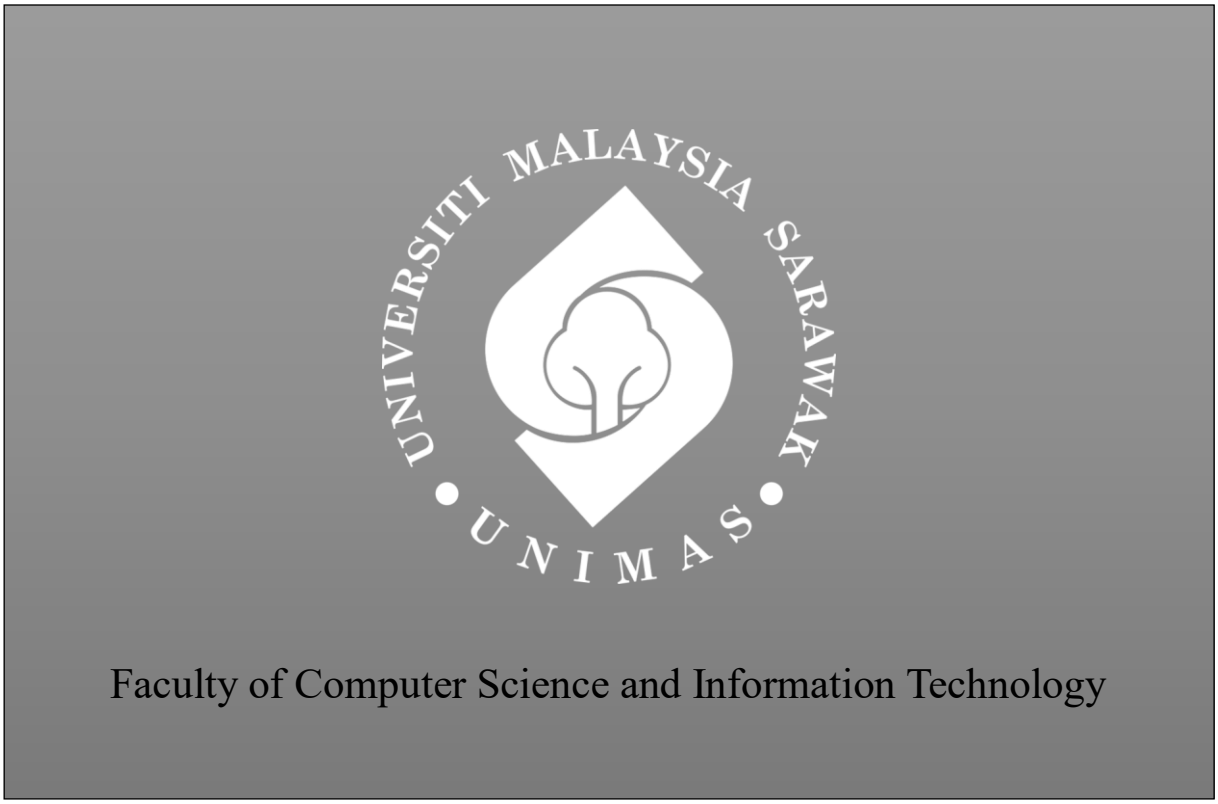


Faculty of Computer Science and Information Technology

DEEP LEARNING-DRIVEN FACE RECOGNITION ATTENDANCE APP

JUSTINE LING KE YI

Bachelor of Software Engineering with Honours

2025

DEEP LEARNING-DRIVEN FACE RECOGNITION ATTENDANCE APP

JUSTINE LING KE YI

This project is submitted in partial fulfilment of
the requirements for the degree of
Bachelor of Computer Science with Honours

Faculty of Computer Science and Information Technology
UNIVERSITI MALAYSIA SARAWAK
2025

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE **DEEP LEARNING-DRIVEN FACE RECOGNITION ATTENDANCE APP**

ACADEMIC SESSION: __2024/2025_____

_____ JUSTINE LING KE YI _____
(CAPITAL LETTERS)

hereby agree that this Thesis* shall be kept at the Centre for Academic Information Services, Universiti Malaysia Sarawak, subject to the following terms and conditions:

1. The Thesis is solely owned by Universiti Malaysia Sarawak
2. The Centre for Academic Information Services is given full rights to produce copies for educational purposes only
3. The Centre for Academic Information Services is given full rights to do digitization in order to develop local content database
4. The Centre for Academic Information Services is given full rights to produce copies of this Thesis as part of its exchange item program between Higher Learning Institutions [or for the purpose of interlibrary loan between HLI]
5. ** Please tick (√)

	CONFIDENTIAL	(Contains classified information bounded by the OFFICIAL SECRETS ACT 1972)
	RESTRICTED	(Contains restricted information as dictated by the body or organization where the research was conducted)
√	UNRESTRICTED	

_____
(AUTHOR'S SIGNATURE)

Permanent Address

No 46, JADE GARDEN
97000 BINTULU
SARAWAK

Date: _23/7/2025_____

Validated by

_____
(SUPERVISOR'S SIGNATURE)

Date: 25/7/2025

Note * Thesis refers to PhD, Master, and Bachelor Degree
** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

DECLARATION OF ORIGINALITY

I hereby declare that this research together with all of its content is none other than that of my own work, with consideration of the exception of research-based information and relative materials that were adapted and extracted from other resources, which have evidently been quoted or stated respectively.

Signed,

…….

JUSTINE LING KE YI

Faculty of Computer Science and Information Technology 22 June 2025
Universiti Malaysia Sarawak.

ACKNOWLEDGEMENT

First of all, I would like to express my heartfelt gratitude to my supervisor, Ts. Dr. Chiu Po Chan, for her continuous and meticulous guidance in data and text processing as well as application development. Without her dedicated teaching, I believe it would have been extremely difficult to successfully complete this thesis.

Next, I would like to express my heartfelt gratitude to Associate Professor Dr. Cheah Wai Shiang, who has provided me with a great deal of professional guidance on Android development, which has greatly helped me to complete this project in an efficient and professional manner.

I am also grateful to the researchers at the University of Massachusetts Amherst who created and maintained the LFW (Labelled Faces in the Wild) face image database, which greatly supported the data collection for this project.

I would also like to thank my friends for their willingness to sacrifice their valuable time to discuss the project with me and for their many helpful comments.

Last but not least, I would like to express my special thanks to my dear family for their encouragement and support, both financially and spiritually, throughout the completion of this project.

Table of Contents

List of Figures

List of Tables

ABSTRACT

Traditional attendance methods, such as manual signing in or swiping cards to punch in, are usually inefficient, error-prone, and risk cheating. To solve these problems, this project proposes an artificial intelligence-based face recognition attendance system and deploys it in an Android mobile application. The system employs deep learning techniques, in particular a metric learning architecture based on Convolutional Neural Networks (CNN), for efficient extraction and comparison of face features. This study compares several classification methods, including a combination of PCA with Random Forest, SVM, KNN and LDA, and the results show that the CNN model achieves the highest recognition accuracy of 88.51% on the LFW dataset. After training, the model was converted to TensorFlow Lite format and successfully integrated into an Android application, achieving real-time recognition and a convenient attendance recording interface. Software tests verified the stability and usability of the system. Although the current system is already usable, the future plan is to integrate the map API to support the geolocation-based verification function to further enhance the user experience. This project validates the feasibility of deep learning-based face recognition in attendance management and provides a secure, efficient and scalable solution for educational institutions.

ABSTRAK

Kaedah kehadiran tradisional seperti tandatangan manual atau sistem kad kehadiran selalunya kurang cekap, mudah berlaku kesilapan, dan terdedah kepada risiko penipuan. Bagi mengatasi masalah ini, projek ini mencadangkan satu sistem kehadiran berasaskan kecerdasan buatan (AI) yang menggunakan pengecaman wajah, dan telah dilaksanakan dalam aplikasi mudah alih Android. Sistem ini menggunakan teknologi pembelajaran mendalam (deep learning), khususnya seni bina pembelajaran metrik berasaskan rangkaian neural konvolusi (CNN), bagi mengekstrak dan membandingkan ciri wajah dengan cekap. Kajian ini membandingkan pelbagai kaedah pengelasan, termasuk gabungan PCA dengan Random Forest, SVM, KNN, dan LDA. Hasilnya menunjukkan bahawa model CNN mencapai ketepatan pengecaman tertinggi iaitu 88.51% pada set data Labeled Faces in the Wild (LFW). Selepas latihan, model ini ditukarkan kepada format TensorFlow Lite dan berjaya diintegrasikan ke dalam aplikasi Android, membolehkan pengecaman masa nyata dan antara muka pencatatan kehadiran yang mesra pengguna. Ujian perisian telah mengesahkan kestabilan dan kebolehgunaan sistem ini. Walaupun sistem ini telah menunjukkan potensi yang baik, pembangunan masa hadapan dirancang untuk mengintegrasikan API peta bagi menyokong pengesahan berasaskan lokasi dan meningkatkan lagi pengalaman pengguna. Projek ini membuktikan kebolehgunaan pengecaman wajah berasaskan pembelajaran mendalam dalam pengurusan kehadiran, menawarkan satu penyelesaian yang selamat, cekap, dan mudah diskalakan untuk institusi pendidikan.

Chapter 1: Introduction

1.1 Introduction

In today's fast-paced world, automation is transforming numerous industries, making processes more efficient and reliable. Particularly in the field of attendance management, traditional methods like roll call or QR code scanning, though widely used, struggle to meet the needs of large institutions and educational organizations. The breakthrough in the field (Mohamed & Raghu, 2012) of automation replacing conventional attendance marking activities is known as an Automated Attendance System. These automated solutions reduce manual labour, enhance accuracy, and minimize errors and fraud risks. When the number of people is large, the traditional method (Sun et al., 2019) of attendance marking becomes exceedingly time-consuming and cumbersome. Automated attendance systems offer an advantage over traditional methods by saving time and enhancing security (Lim et al., 2009). As a result, many organizations are transitioning to automated systems using biometrics, smart cards, and web-based automated time and attendance systems. Among these, facial recognition technology stands out as a promising, non-invasive, and efficient alternative. This field has grown rapidly in recent years and plays an important role in security due to its precise ability to identify and verify individuals (Hua et al., 2011) (Filippidou & Papakostas, 2020). This technology leverages unique facial features for identification and is widely utilized in areas such as security, healthcare, and education.

In educational institutions, tracking student attendance is crucial for monitoring engagement and assessing performance. Schools and colleges need reliable attendance management systems to ensure accurate record-keeping and support quality monitoring. Traditional methods, like calling names or signing papers, are commonly used in most institutions, but they are time-consuming and insecure (Karunakar et al., 2020). These manual

methods present limitations, especially in large classrooms, where maintaining accurate records becomes challenging and time-consuming. Moreover, traditional methods often lead to proxy attendance, further compromising the integrity of attendance records. Therefore, the most suitable method to ensure security and maintain accurate attendance records is through a smart face recognition system (Bhattacharya et al., 2018).

Facial recognition-based attendance systems have gained popularity for their ability to address these challenges efficiently. It automates and streamlines attendance methods. By capturing and analyzing facial images, these systems can accurately verify an individual's identity, providing a quick, seamless check-in process. As a biometric trait, the face provides a highly reliable means of identity verification due to its unique characteristics. With advancements in artificial intelligence, facial recognition has become more accurate and resilient, even when handling variations like changes in lighting, facial expressions, glasses, or beards.

The primary aim of this project is to design and implement an automated time and attendance system based on facial recognition using deep learning algorithms. To achieve this goal, the project uses deep learning models to extract and analyze facial features, comparing live camera feeds with stored facial data for real-time recognition. This approach will not only automate attendance but also improve accuracy and minimize potential manipulation, thus contributing to a safe and efficient classroom environment. This project evaluate multiple machine learning and deep learning models to determine the most suitable model for accurate, real-time attendance tracking. The final model will be integrated into a mobile Android application that allow instructors to monitor attendance through a user-friendly interface while providing an easy, secure check-in experience for students.

Additionally, transfer learning techniques will be utilized to build upon pre-trained models, making the development process more efficient by adapting the model to specific data requirements. Transfer learning particularly helps alleviate difficulties involved in acquiring large datasets (Murel & Kavlakoglu, 2024). Transfer learning proves highly effective in image recognition tasks, as it allows the model to leverage knowledge from previous training, significantly reducing the need for extensive datasets and long training times. In this project, a diverse dataset with multiple categories of facial images will be prepared, pre-processed, and augmented to ensure the model's robustness and accuracy in real-world scenarios.

1.2 Problem Statement

Inefficient attendance management is a common problem in educational institutions. Most universities and schools apply manual attendance system (Sittampalam & Ratnarajah, 2019). Traditional methods of attendance management, such as manual sign-in or code scanning, are not only inefficient but also easily manipulated. As class sizes increase, these attendance methods often result in repetitive, time-consuming processes that add to the administrative burden of educational institutions and create significant hidden costs for schools and parents. Research from the National Center for Education Statistics reveals that the average professor spends approximately 10 minutes per day on attendance- related tasks (*The Hidden Costs: How Manual Attendance Tracking Damages Schools and Disrupts Parent Engagement | Orah Blog*, n.d.). Manual sign-in is particularly susceptible to proxy attendance problems, where one student signs in for another, resulting in inaccurate records and compromising the integrity of attendance tracking. Systems based on QR codes and ID cards, although automated, still require individual scanning and verification for large classes, which is time consuming and disruptive. As a result, manual system is not effective, inefficient, and prone to human error (Adiono et al., 2021).

Face recognition is a part of biometric identification that extracts the facial features of a face, and then stores it as a unique face print to uniquely recognize a person (Dubey, 2020). Face recognition as a biometric method of recognizing an individual by comparing real-time captured images is easily accessible and non-invasive. Facial recognition technology offers a promising alternative to traditional time and attendance systems. As the face recognition system is able to verify the identity of each individual based on his/her unique facial features, it effectively prevents problems such as buddy punching on behalf of a friend, thus safeguarding the accuracy of attendance records and eliminating attendance falsification.

Currently, the research of face recognition attendance system has reached a high level, in which deep learning techniques, especially Convolutional Neural Networks (CNN), have become the core methods. These techniques achieve highly accurate face verification and recognition by extracting key features of a face image. In recent years, the success of deep learning models such as DeepID and FaceNet has further advanced the field. Since 2017, Google has successively proposed MobileNetV1, MobileNetV1, and MobileNetV3, which all of these can be applied to mobile and embedded devices (Nan et al., 2021) for recognizing faces and expressions.

However, despite the significant technological advances, existing systems still face a number of challenges. These challenges include the limitations of real-time and computational resources and the problem of recognition accuracy in complex environments. In real-time attendance management scenarios, the high computational complexity of convolutional neural networks may affect the recognition speed, while lighting variations, shooting angles, and face occlusion can lead to a decrease in recognition accuracy.

Therefore, there is an urgent need for a new automated, secure, and scalable attendance solution for educational institutions to overcome the limitations of traditional methods and improve the accuracy and efficiency of attendance.

1.3 Project Scope

The scope of this project is to develop an AI-based face recognition attendance system designed to streamline and improve the efficiency and accuracy of classroom attendance. The project focus on improving model robustness to address challenges specific to face recognition, such as the impact of different angles and expressions on recognition accuracy, as well as the interference of facial occlusion and lighting changes on model adaptation. To ensure data quality, datasets from public platforms such as Kaggle will be used, and recognition accuracy will be improved by preprocessing techniques such as image enhancement and segmentation.

The deliverable of the project is to develop a user-friendly Android application to help professors conveniently manage and track student attendance, and to promote intelligent and automated classroom management.

1.4 Aim and Objectives

The aim of this study is to propose a deep learning-based face recognition system for class attendance.

The objectives of this project are as follows:

- To develop a face recognition model for class attendance based on deep learning
- To evaluate the performance of different methods in face recognition modelling and compare the advantages and disadvantages of machine learning and deep learning technique
- To design an Android app for class attendance

1.5 Methodology

The method chosen for this project is the Cross-Industry Standard Process for Data Mining Methods (commonly referred to as CRISP-DM). CRISP-DM is the de-facto standard and an industry-independent process model for applying data mining projects (Schröer et al., 2021). The methodology is most used for data mining, analytics and data science projects. The methodology divides the lifecycle of a data mining project into six phases, each with clearly defined roles and tasks. This methodology consists of business understanding, data understanding, data preparation, modelling, evaluation, and deployment (Rotty et al., 2023). Figure 1.1 below shows the six stages of the CRISP-DM methodology.

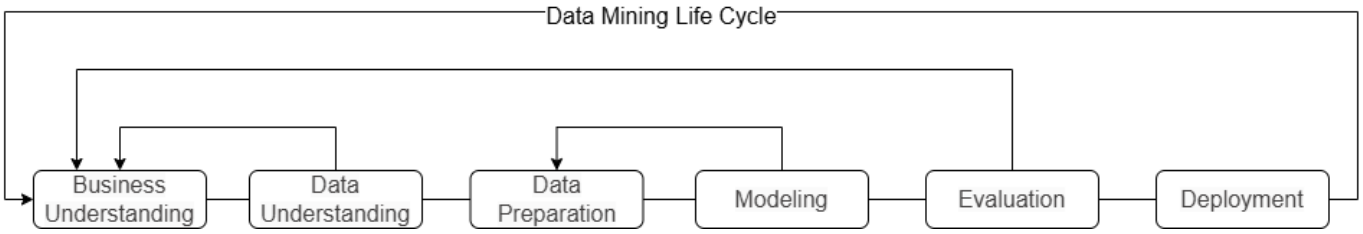


Figure 1.1 : Six phases of CRISP-DM Methodology

1.5.1 Business Understanding

This phase focuses on defining the goals and needs of the project. In this project, the objective is to develop an automated time and attendance system to replace the traditional manual attendance method and improve the efficiency and accuracy of attendance. With face recognition, the system can avoid problems such as substitute punching and improve the authenticity and security of attendance records. In the business understanding phase, the author needs to specifically define the core requirements of the project and set the project objectives. Based on these requirements, transform the project needs into data mining problems to determine the rationality of using artificial intelligence and face recognition technologies and their feasibility.

1.5.2 Data Understanding

In the data understanding phase, the face dataset required for the project needs to be collected and analysed. First, acquire face data under different lighting, angles, and expressions to ensure data diversity and thus improve the model's performance in a variety of real-world scenarios. Meanwhile, through data exploration, the distribution of the dataset is analysed, data quality issues are identified, and it is ensured that the collected data can reflect the scenario requirements of student attendance.

1.5.3 Data Preparation

In the data preparation phase, the collected raw dataset needs to be organized into a format suitable for model training through a series of preprocessing steps. This phase includes operations such as removing low-quality images, cleaning up background interference, and resizing face images to keep the data consistent. In addition, data enhancement operations (e.g., rotation, cropping, flipping) are also considered to improve the generalization ability of the model. Through data processing, a high-quality dataset containing clear and standardized face images will be finally constructed to ensure that the model can learn effective features more efficiently during the training process.

1.5.4 Modelling

The modelling phase focuses on selecting, training and optimizing the most suitable model for face recognition. PCA, RF, SVM, KNN and LDA are used for the machine learning part, while CNN is selected for deep learning. hyperparameter tuning, feature engineering and data preprocessing are used to ensure that the model extracts effective features and achieves high accuracy. After training, the model performance is evaluated using a validation set, and

the optimal model is finally selected by combining metrics such as accuracy, F1 score and ROC curve.

1.5.5 Evaluation

After completing the model training, the performance of the model is tested through a series of evaluation metrics. Particularly in this project, the accuracy of the model in recognizing students' faces needs to be ensured, while considering the model response speed. At this stage, you also need to evaluate whether the model meets the business requirements, such as whether it enables real-time attendance and prevents behaviours such as substitute punching. The robustness of the model under various types of lighting, angles and expressions should also be verified to ensure that the model has a stable recognition effect in real application environments.

1.5.6 Deployment

In the deployment phase, the model will be integrated into the backend system of the Android application to automate attendance management. A clear deployment plan is firstly formulated, including the work of server building and interface development. After the system is online, users can operate attendance through the mobile application, students can complete signing in, and professors can easily view attendance records. At the same time, the attendance data is analysed and reports are generated regularly in order to continuously improve the effectiveness of the system.

1.6 Significance of Project

The significance of this project is to develop an effective automated attendance tracking solution through face recognition technology that provides higher accuracy and security compared to traditional attendance methods. By leveraging deep learning algorithms, the project aims to provide a robust and scalable solution that minimizes attendance operations and administrative overhead. The system will improve the ability of educational institutions to monitor student participation and provide accurate attendance records, which is critical for performance evaluation and quality monitoring.

In addition to its educational applications, this face recognition-based attendance system has broad potential in other environments where contactless attendance is preferred, such as corporate offices and event check-ins. This project's findings will advance the use of biometrics in education by providing a reliable model of a secure, automated attendance system applicable across various institutions and settings. Ultimately, this project has the potential to streamline the attendance management process, reduce manual effort, and promote a safe and efficient academic environment, while also laying foundational support for intelligent, automated management systems of the future.

1.7 Expected Outcome

The expected outcome of this project is to identify and select the most effective facial recognition model by performing a comprehensive comparative analysis of various machine learning and deep learning models. This will include evaluating their accuracy, speed and scalability in a real-time attendance tracking scenario. Additionally, the project aims to develop an Android application that provides a user-friendly interface to enable professors to seamlessly manage and monitor student attendance. The app will integrate advanced face detection and recognition features to enable students to sign in quickly and securely. Ultimately, the successful implementation of the system will result in a reliable automated attendance management solution that will increase operational efficiency and improve the overall educational experience for both students and professors.

1.8 Project Schedule

ID	Task Name	Start	End	Duration
1	Deep Learning-Driven Face Recognition Atten…	2024-10-07	2025-07-25	210 days
2	Building Deep Learning Models for Face Re…	2024-10-07	2025-05-20	162 days
4	Business Understanding	2024-10-07	2024-11-15	30 days
5	Data Understanding	2024-11-18	2024-12-27	30 days
6	Data Preparation	2024-12-30	2025-02-07	30 days
7	Modelling	2025-02-10	2025-03-21	30 days
8	Evaluation	2025-03-24	2025-04-21	21 days
9	Deployment	2025-04-22	2025-05-20	21 days
3	Building Android App for Class Attendance	2025-05-21	2025-07-25	48 days
10	Application Development	2025-05-21	2025-06-18	21 days
11	Backend Integration	2025-06-19	2025-07-17	21 days
12	Testing & Debugging	2025-07-18	2025-07-25	6 days

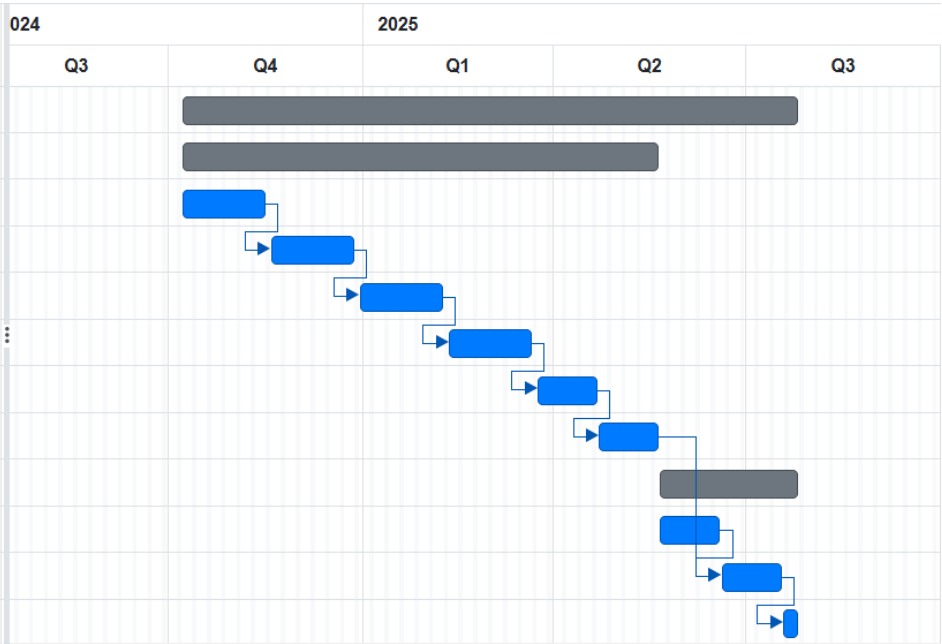


Figure 1.2 : Overall Gantt Chart of the Project

1.9 Thesis Outline

1.9.1 Introduction

The introduction chapter lays the groundwork for the project by providing a comprehensive overview. It begins with the problem statement, which identifies the key issues to be addressed by the project. The chapter then defines the project scope, emphasizing the focus and boundaries of the study. This is followed by the objectives, which clearly outlines the purpose of the study. The chapter provides a brief overview of the methodology of the research used to accomplish the overall project. The chapter also emphasizes the significance of the project, explaining its potential impacts and benefits. The chapter also discusses the expected outcomes of the project to help the reader understand what the project intends to accomplish. In addition, a project schedule is outlined to highlight the key milestones and timeline for completion. Finally, the chapter concludes with a thesis outline that provides the reader with a roadmap to subsequent chapters.

1.9.2 Literature Review

The literature review focus on analysing the potential of machine learning and deep learning algorithms in the field of face recognition. This chapter introduce a variety of machine learning algorithms, including Principal Component Analysis (PCA), Random Forest (RF), Support Vector Machine (SVM), K Nearest Neighbors (KNN), and Linear Discriminant Analysis (LDA), as well as deep learning methods such as Convolutional Neural Networks (CNNs), and discuss in detail the fundamentals of these algorithms and their applications in face recognition. Meanwhile, the literature review also analyses the performance of these algorithms for face recognition in real-world scenarios in related studies and compares them in terms of applicable scenarios, performance evaluation metrics, and the ability to cope with specific challenges. This chapter also discusses the supporting role of relevant technology stacks, such as the importance of tools like Python, OpenCV and TensorFlow in data processing

and model training. By comparing the strengths and weaknesses of different algorithms and their technical implementations, this chapter provides a solid theoretical foundation for the selection and optimization of algorithms in this study. In addition, this chapter also provides a comparative analysis of different attendance methods.

1.9.3 Methodology

This chapter detail the methodology used to address the research problem and achieve the project objectives. Also, this chapter discuss in detail the selection of machine learning and deep learning algorithms and the setup of the training process.

1.9.4 Implementation

This chapter details the project implementation steps from data preparation to Android app building. First, the dataset is pre-processed to ensure that it is suitable for training deep learning models. Subsequently, the chapter builds and evaluates six face recognition models involving algorithms such as Principal Component Analysis (PCA), Random Forest, Support Vector Machine (SVM), K Nearest Neighbor (KNN), Linear Discriminant Analysis (LDA), and Convolutional Neural Networks (CNN) to determine the best algorithms for the face recognition task. Finally, the best performing model is integrated into an Android application to provide a reliable solution for attendance management.

1.9.5 Result and Discussion

This chapter analyse in detail the results of the models trained for performing face recognition. This section provides a comprehensive evaluation and comparison of the model's performance on the training and test sets through performance metrics. By visualizing and analysing the performance of different models, the best model will be selected for implementation.

1.9.6 Conclusion and Future Works

The final chapter summarize the main results of this project, including the identification of the most suitable model for face recognition attendance management and its potential applications in the field of attendance management. The chapter also discuss the limitations of the research and look at future research directions in this area. Meanwhile, the chapter summarize the overall objectives and significance of the project and make further recommendations for improving attendance management.

Chapter 2: Literature Review

2.1 Introduction

Face recognition techniques have become an important research direction in the fields of machine learning (ML) and deep learning (DL), and the main driver for their development is the need for enhanced security, surveillance capabilities, and authentication systems. This literature review examines a variety of ML and DL algorithms, including Principal Component Analysis (PCA), Random Forest (RF), Support Vector Machine (SVM), K Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), and Convolutional Neural Networks (CNN), which have their own functional characteristics ranging from Dimensionality Reduction (DRR) to Complex Pattern Recognition (CPR) to satisfy the diversified needs of facial recognition tasks.

In addition, this chapter explores how the combination of ML and DL techniques can address practical challenges such as occlusion, illumination variations, and pose diversity. Through a critical review of related research, this chapter focuses on analyzing the latest advances and innovative solutions, such as hybrid methods and optimized architectures, which effectively improve the accuracy and adaptability of facial recognition. The chapter also provides a detailed comparison of these techniques based on metrics such as recognition accuracy, thus providing an in-depth analysis of their strengths and limitations. At the same time, this chapter discusses the underlying technology stack including Python, OpenCV, TensorFlow, Kotlin, and Firebase, demonstrating their key role in implementing and optimizing these algorithms. In addition, this chapter covers a variety of time and attendance methods, such as manual attendance, QR code attendance, NFC/RFID attendance, GPS-based attendance, face-recognition attendance, and fingerprint-recognition attendance, each of which provides unique functionality and value for a particular application scenario.

2.2 Machine Learning Algorithms and Deep Learning Algorithms

Machine learning is a complex computational paradigm that improves the performance of a system by extracting insights from data. Data is the basic source of knowledge acquisition for computer systems. The overall goal of machine learning is to design algorithms that can construct predictive models from data. These models are trained by exposing machine learning algorithms to empirical datasets and can accurately predict newly observed outcomes. The constructs derived from the data are often referred to as "models" (Zhou, 2021).

Machine learning algorithms are usually categorized into three main groups: supervised learning, unsupervised learning, and reinforcement learning. These categories vary depending on the learning method, the nature of the input and output data, and the type of problem to be solved. In addition, hybrid strategies and advanced techniques provide extensions to accommodate complex problem types, reflecting the expanding range of machine-learning applications (Sah, 2020).

Supervised learning is characterized by the use of labelled datasets to train algorithms. By being exposed to labelled inputs and outputs, supervised models are able to improve their accuracy over time (Delua, 2021). Popular supervised learning techniques include Random Forests, Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Linear Discriminant Analysis (LDA). These methods are widely used in domains that require accurate prediction and classification.

In contrast, unsupervised learning automatically discovers underlying patterns and structures in data by algorithmically analyzing and clustering unlabeled data sets (Delua, 2021). Principal Component Analysis (PCA) is a well-known unsupervised learning algorithm that is highly regarded for its ability to effectively reduce the dimensionality of the data and preprocess the data, thereby facilitating the efficiency of subsequent analysis.

Deep learning is a specialized subset of machine learning that represents a paradigm shift in data processing and pattern recognition. It utilizes artificial neural networks, a computational model inspired by the human brain, to automatically discover complex patterns and relationships in data. A neural network consists of multiple layers of interconnected processing units, called neurons, that work together to transform inputs into meaningful outputs through a series of computational steps.

The core concept of deep learning is hierarchy. The input layer receives raw data, such as images, text, or numerical values, and passes this information to the hidden layer for analysis. In complex networks, there are typically dozens or even hundreds of hidden layers that perform most of the computational tasks. Each layer extracts increasingly abstract features from the input data, and ultimately, the hidden layer provides a comprehensive representation. The output layer utilizes the computational results of the previous layers to generate predictions or classification results based on the task. Deep learning models use weights and biases to adjust the strength of the connections between neurons and continuously optimize the computational process during training. Activation functions play a crucial role in neural networks by introducing nonlinearities that enable the network to model complex relationships in the data. These functions determine whether a neuron is activated or not, thus effectively controlling the flow of information through the network.

This project test and compare five machine learning algorithms (PCA, Random Forest, SVM, KNN, and LDA) and one deep learning algorithm (CNN), and identify the most suitable algorithms for the face recognition task by evaluating their performances when dealing with face datasets. We compare the accuracy, efficiency and adaptability of the algorithms and ultimately select the best-performing algorithm to improve the predictive accuracy and performance of the face recognition model.

2.2.1 Principal Component Analysis (PCA)

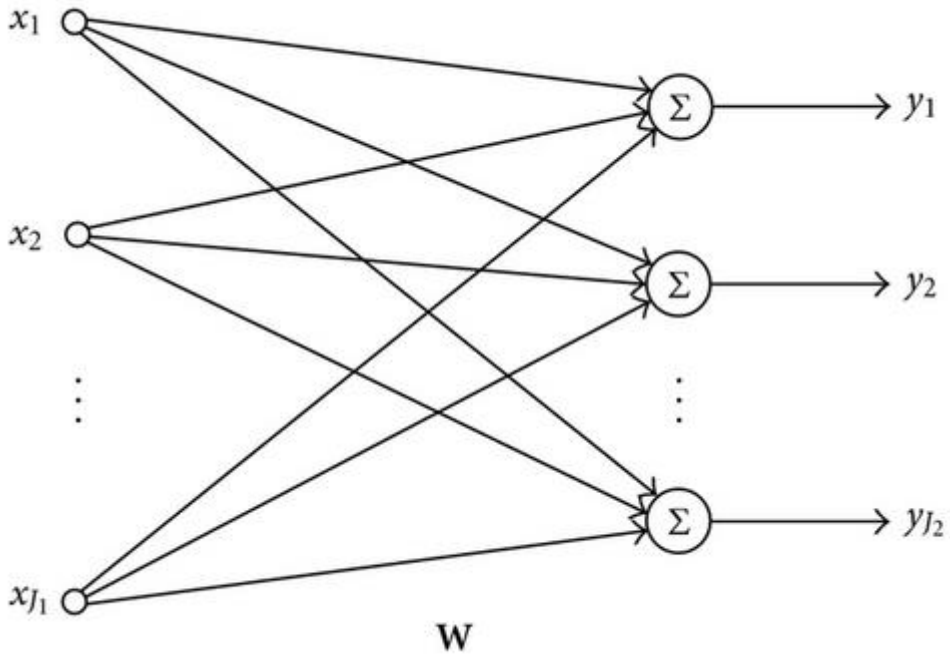


Figure 2.1 : The architecture of the PCA network

Principal Component Analysis (PCA) is a statistical method used to reduce a table of case data categorized by variables to their basic characteristics, known as principal components. It does this by identifying orthogonal axes along which the data shows the greatest variance. These components are linear combinations of the original features and are arranged in descending order of the variance they capture. PCA is widely used in fields such as machine learning, bioinformatics, and finance for tasks such as data visualization, noise reduction, and feature extraction (Jolliffe & Cadima, 2016).

PCA follows a series of well-defined steps to transform data. First, the dataset is centered by subtracting the mean of each feature to ensure that all features have a mean of zero. This step standardizes the data for analysis. Next, the covariance matrix of the centered data is computed to capture the relationship and variability between features. The eigenvectors and eigenvalues of the covariance matrix are then determined, where the eigenvectors represent the direction of the principal components and the eigenvalues represent the variance explained by

each component(Jolliffe, 2002). These components are ranked according to their eigenvalues and the most significant components are selected for dimensionality reduction.

The architecture of a PCA network is given in Figure 2.1(Qiu et al., 2012). PCA works by projecting the original data onto a subspace defined by the selected principal components. This projection reduces the dimensionality of the data while retaining the most important patterns. Mathematically, this process is expressed as y=Wx, where y is the transformed data, W is the principal component matrix, and x is the original data. PCA minimizes reconstruction errors and ensures that the simplified representation is very close to the original data set(Abdi & Williams, 2010). It is particularly useful when features are highly correlated, as it creates a new coordinate system with uncorrelated axes, thus improving the interpretability of the data.

The application of principal component analysis (PCA) in face recognition originated from a technique based on dimensionality reduction and linear algebra proposed by Turk and Pentland in 1991 (GeeksforGeeks, 2021). The method realizes feature extraction and dimensionality reduction by projecting high-dimensional face image data into a low-dimensional feature space. The specific process includes:

1. Pre-processing of face images (e.g., gray scaling, size normalization, and illumination normalization) to normalize the data
2. Extracting eigenvalues and eigenvectors by calculating the covariance matrix of the image data and constructing the projection matrix by selecting principal components based on the cumulative contribution rate
3. Generate Eigenfaces by dimensionality reduction of the image data to the feature space using the projection matrix.

4. Face matching and recognition is achieved by calculating the similarity (e.g., Euclidean distance or cosine similarity) between the image to be recognized and the image in the database that has been downsized.

In conclusion, PCA is a robust and versatile technique for downscaling and feature extraction. Converting data to orthogonal principal components can simplify complex datasets, improve computational efficiency, and reveal underlying patterns. Despite its limitations, such as the assumption of linear relationships and sensitivity to scaling, PCA remains a fundamental method for data preprocessing and analysis (Wold et al., 1987).

2.2.2 Random Forest

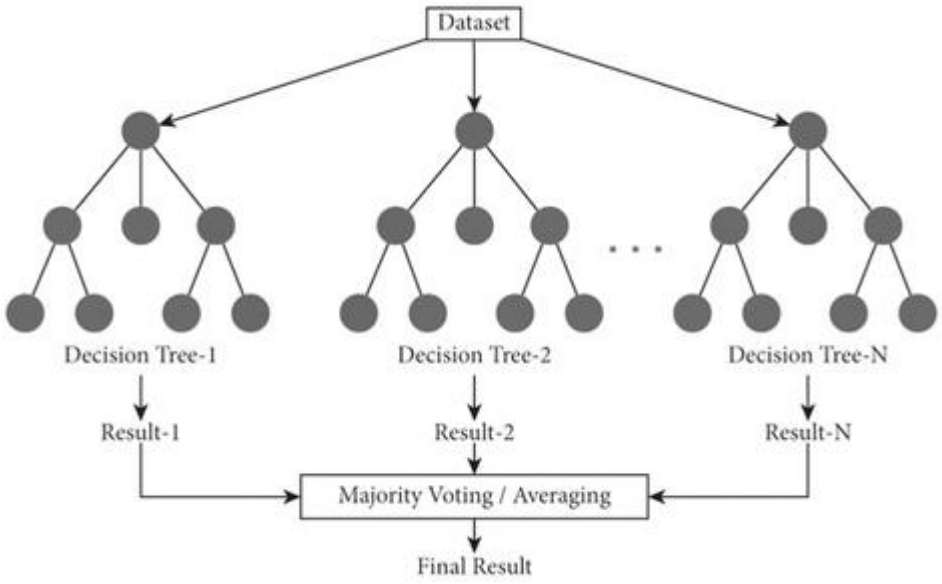


Figure 2.2 : The illustration of Random Forest Tree

The Random Forest algorithm is a widely used and efficient machine learning method known for its simplicity, robustness and versatility. The Random Forest Tree is illustrated in Figure 2.2 (Khan et al., 2021). It belongs to the family of integrated learning algorithms, which combine multiple individual models to produce stronger and more reliable predictions. In the case of Random Forest, it does this by constructing a collection of decision trees during training and using them collectively for prediction through a voting mechanism (Abdulkareem & Abdulazeez, 2021). This design makes random forests ideal for classification and regression tasks.

At its core, Random Forest relies on decision trees, which are the building blocks of the model(Quinlan, 1986). Decision trees operate by recursively partitioning data into subsets based on specific features. Each split corresponds to a yes/no question, e.g. whether the feature value exceeds a threshold (S Mucesh, 2021). The process continues until a stopping criterion is met, such as reaching a minimum number of samples per leaf or a maximum depth.

Individual decision trees, while interpretable and flexible, are prone to overfitting when used alone. Random forests mitigate this problem by combining predictions from multiple trees.

The Random Forest algorithm introduces randomness to enhance diversity and robustness. First, it uses bootstrap sampling in which each tree is trained on a randomly selected subset of data (IBM, 2024). This ensures that each tree receives a slightly different view of the dataset. Second, in each segmentation of the tree, only a random subset of features is considered in determining the optimal segmentation (Kursa and Rudnicki, 2011). This de-correlation between trees enhances the whole as it prevents individual trees from converging to similar patterns. As a result, the randomized forest produces more accurate and stable predictions.

Once the decision tree forest is constructed, the randomized forest uses a voting mechanism for classification task prediction. Each tree in the forest provides its own prediction, and the category that receives the majority of votes is chosen as the final prediction. For the regression task, the algorithm calculates the average of the predictions of all trees. This aggregation process utilizes the principle of "wisdom of the crowd" to produce predictions that are more accurate and less biased than the predictions of any one tree (Breiman, 2001).

Random forest algorithm is an effective method for face recognition, which is mainly realized through the following steps. First, the image dataset is preprocessed to convert all images to a uniform gray value and normalize the size to ensure that all images have the same resolution. Next, effective features are extracted (e.g., by PCA, HOG, or LDA) and features are normalized. Then, multiple decision trees are constructed using random forests, and training samples and feature subsets are randomly selected by bootstrap sampling to reduce the risk of overfitting. In order to optimize the performance, key parameters need to be adjusted, including the number of trees (Number Tree) and the convergence threshold (Epsilon). The robustness and scalability of Random Forest makes it perform well in image recognition tasks.

Random Forest is a powerful algorithm that builds on the strengths of decision trees while addressing their weaknesses through integrated learning. By combining multiple trees, utilizing bootstrap sampling and introducing feature randomness, it achieves high accuracy, robustness and interpretability. These properties make Random Forest a cornerstone of modern machine learning applications.

2.2.3 Support Vector Machine (SVM)

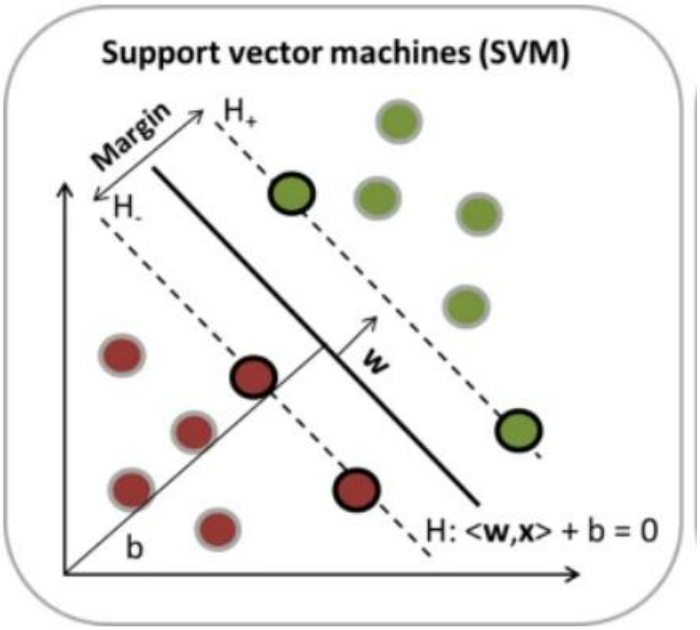


Figure 2.3 : Modeling of Support Vector Machine

Support vector machine (SVM) is one of the widely used classifiers in traditional machine learning algorithms (Chang & Lin, 2011). It is particularly well suited for working with complex and unbalanced datasets and performs especially well on small and medium-sized datasets. The goal of the SVM is to determine whether a data point belongs to the correct category by constructing a hyperplane that delineates a categorical boundary between two datasets (S, David, W, & Sundar, 2019).SVMs define the hyperplane as the categorical boundary between two categories (Rodríguez-Pérez & Bajorath, 2022).SVMs can be used for linear or nonlinear categorization, but the basic SVM that applies to the hyperplane is often referred to as a linear SVM (Khan & Nazir, 2020; FanRong-En et al., 2008). As shown in

Figure 2.3, the construction of SVM models relies on the definition of support vectors (SVs), which play a key role in classification and regression (Rodríguez-Pérez & Bajorath, 2022).SVMs strike a balance between training data fitting and the ability of the model to generalize, which reduces the risk of overfitting, which typically affects the generalization ability of machine learning models.

Support Vector Machines (SVMs) can efficiently handle small to medium sized image data and have good generalization ability in the classification process, which is suitable to cope with complex face recognition tasks. The application of SVM in face recognition distinguishes different face images by constructing an efficient classifier. The specific process includes:

1. Pre-processing (e.g., gray scaling, normalization, and size normalization) of face images to ensure data consistency
2. Converting image data into feature vectors by feature extraction (e.g., using PCA or other methods)
3. Selecting a suitable SVM kernel function (e.g., Gaussian or linear kernel) and optimizing key hyper-parameters (e.g., soft interval weights C and parameters of the kernel function) through cross-validation
4. In the training phase, train the SVM classifier using the extracted feature vectors and define the classification hyperplane by support vectors
5. In the testing phase, feature extraction and SVM classification are performed on the face images of the test set to predict the category they belong to.

2.2.4 K-Nearest Neighbor (KNN)

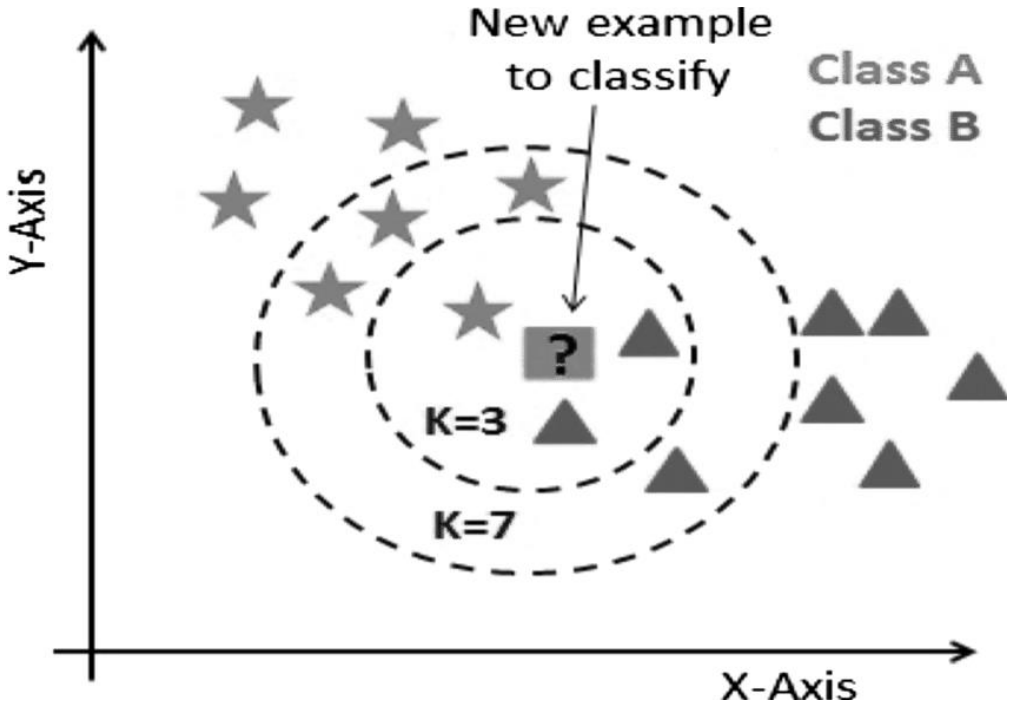


Figure 2.4 : The Block diagram of KNN classification

K-Nearest Neighbors (KNN) is a nonparametric classification method that is both simple and effective in many applications (Smith, 2002). The K-Nearest Neighbors algorithm records all instances in an n-dimensional space that match the training data points. For continuous numerical data, KNN returns the average of the k nearest neighbors; for discrete data, the algorithm examines the closest k stored instances (i.e., the nearest neighbors) when unknown data is received and returns the most common category as a prediction (Guo et al., 2004). The distance-weighted nearest neighbor algorithm weights the contribution of each nearest neighbor by the following query, giving greater weight to the nearest neighbors based on their distance from the query point. Since KNN is classified by the average of k nearest neighbors, it is usually more sensitive to noisy data (Dewi and Imah, 2020). Figure 2.4 shows a block diagram of KNN classification (Kumar et al., 2019), and it can be seen that when the value of k is taken differently, the classification results vary. Therefore, how to choose the value of k is crucial in this algorithm as it will directly affect the final classification result.

In face recognition, KNN uses the distance between the feature vectors extracted from the face image for classification (Happiness, 2023). The KNN-based face recognition algorithm process includes several key steps. First, the original image is preprocessed, including gray scaling, size normalization, and feature extraction (e.g., PCA) to ensure data consistency and validity. Next, based on the extracted features, a feature space is constructed, where each point represents a sample (i.e., face image) and each dimension represents features extracted from the image (e.g., feature vectors extracted by PCA). Then, the images and labels in the training set are used to construct a KNN model, which performs classification by calculating the distance (e.g., Euclidean distance) between the feature vectors of the test image and the samples in the training set. In the testing phase, the preprocessed test images are classified by the KNN model and the accuracy of the model is finally evaluated based on the comparison of the classification results with the real labels. Through these steps, KNN is able to effectively perform feature-based classification tasks for face recognition.

2.2.5 Linear Discriminant Analysis (LDA)

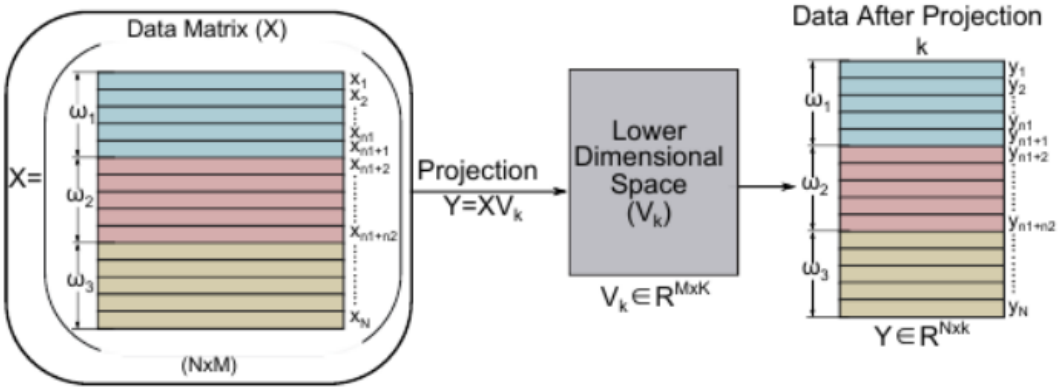


Figure 2.5 : The projection of the original sample into the LDA low-dimensional space

Linear Discriminant Analysis (LDA) is a classical dimensionality reduction method. Unlike Principal Component Analysis (PCA), which is an unsupervised dimensionality reduction technique that does not take into account the category information of the samples,

LDA is a supervised learning dimensionality reduction technique that requires each sample in the dataset to have a category label. As a dimensionality reduction method, LDA is often used in classification problems to classify data samples into two or more groups by finding linear feature combinations (McLachlan, 2004).

The core goal of LDA is to transform features into a lower dimensional space while maximizing the ratio of interclass variance to intraclass variance to achieve maximum class separability (Pan et al., 2014). Specifically, LDA looks for a projection direction that keeps the centroids of different classes as far apart as possible while ensuring that data points within the same class are as close as possible. In some literature, LDA techniques have been categorized into two types: class-dependent and class-independent (Viszlay et al., 2014). Class-dependent methods compute a separate projection space for each category, while class-independent methods differentiate between categories through global projections.

The projection of the original sample (i.e., the data matrix) in the low-dimensional space of LDA demonstrates its downscaling effect, as shown in Figure 2.5 (Tharwat et al., 2017). LDA typically projects d-dimensional data into a space of at most C-1 dimensions (where C is the number of categories), which effectively simplifies the processing of high-dimensional data. Through this process, LDA transforms complex classification problems into linearly differentiable problems in lower dimensional spaces, making subsequent analysis and classification more efficient.

LDA can objectively assess the importance of visual information in different features of a face for recognizing a face (Bala, Manju et al., 2016). The implementation of Linear Discriminant's Analysis (LDA) in face recognition can be achieved in the following steps: first, the image data of multiple face samples is reconstructed to ensure that it contains only the face region and to unify the images normalized to the same dimensions (e.g., normalized to a fixed-

size grayscale image). Next, each 2D image matrix is converted to a 1D representation, thus representing all facial data in a high dimensional feature space. Next, based on the Fisher's discriminant criterion, the intraclass scatter matrix and interclass scatter matrix are computed: the intraclass scatter matrix is used for the degree of segmentation between samples of different classes, and the interclass scatter matrix is used for the degree of separation between samples of different classes. Maximize the ratio of the eigenvalues of the interclass scatter matrix to the intraclass scatter matrix, find the linear projection direction that can best divide the different categories, and project the image data into a low-dimensional feature space. The feature representations generated at this point are called Fisher faces, and they are low-dimensional features extracted from high-dimensional image data with the strongest category separation ability. In the recognition phase, the test image is first projected into the same low-dimensional feature space, and then the feature processing of the test image is compared to that of the training image using a proximity approximation method (e.g., Euclidean distance) to identify the category that is closest to the image test. This process ensures the accuracy of the recognition results by maximizing the separation of different categories and minimizing the closeness of the same categories.

2.2.6 Convolutional Neural Networks (CNN)

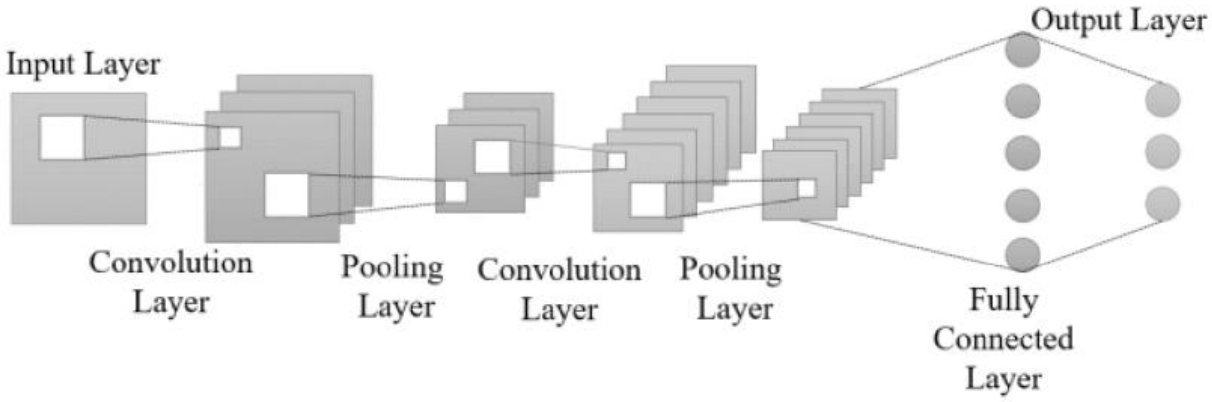


Figure 2.6 : The architecture of the CNN model

Convolutional Neural Network (CNN) is a class of deep learning models specialized for processing structured data, such as images and videos. CNN is inspired by the visual cortex of the human brain and is able to detect hierarchical patterns in the data by using multiple layers of interconnected neurons. Unlike traditional machine learning models, CNN automatically learns features from raw input data through a process of convolution, pooling, and nonlinear transformation (Lecun et al., 1998; Zhang et al., 2012).

A CNN is an artificial neural network with a unique architecture as shown in Figure 2.6(Gu et al., 2019). The main building block of a CNN is the convolutional layer, where a set of learnable filters (kernels) are slid over the input data to generate a feature map. Each filter detects specific features, such as edges or textures, by applying mathematical convolution operations. The pooling layer is located after the convolutional layer and is used to reduce the spatial dimensions of the feature map, thus making the network computationally more efficient and robust to small spatial variations(Deep Learning, n.d.). The most common pooling method is maximum pooling, which selects the highest value in a specified window and retains only the most salient features.

A nonlinear activation function (e.g., ReLU (rectified linear unit)) is applied after each convolution operation to introduce nonlinearity into the model, allowing it to learn complex patterns. Fully connected layers are typically used at the end of the network to integrate extracted features and make predictions (Nair & Hinton, 2010). These layers use functions such as SoftMax to convert features into output labels or probabilities, which is particularly effective for multi-class classification tasks.

During training, CNN updates their weights using a process called backpropagation. This involves calculating the gradient of the loss function concerning the model parameters and using an optimization algorithm such as stochastic gradient descent (SGD) to adjust the

weights. Through iterative updating, the network learns to minimize losses, thereby improving its ability to detect features and make accurate predictions. These features, combined with their hierarchical structure, make CNN a powerful tool for analyzing grid-like data.

The complete process of Convolutional Neural Network (CNN) for face recognition consists of the following steps: first, the input face image is subjected to feature extraction by means of convolutional layers, which utilize multiple learnable convolutional kernels to perform sliding operations on the image in order to capture local features such as edges, textures, etc. and to generate a feature map. Subsequently, these feature maps undergo dimensionality reduction by a pooling layer (e.g., maximum pooling or average pooling), which reduces the amount of data while retaining important information, thereby improving computational efficiency and enhancing robustness to positional offsets. Next, the negative values in the feature map are set to zero by a nonlinear activation function (e.g., ReLU) to enhance the network's ability to learn complex features. Finally, the extracted high-dimensional features are combined into a fixed-length vector through a fully connected layer, and a SoftMax classifier maps these features into category probabilities, which ultimately outputs the predicted category results, i.e., the identity or label corresponding to the face (Coskun et al., 2017). Through the above process, the CNN is able to automatically learn the complex features in the image, providing efficient and accurate face recognition capability.

2.3 Related Works

This section reviews the relevant research work in the field of facial recognition in recent years. Many researchers have explored face recognition in depth using machine learning and deep learning techniques. The commonly used algorithms in these researches include Principal Component Analysis (PCA), Random Forest, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), and Convolutional Neural Network (CNN). These algorithms are widely used in feature extraction, classification, and model optimization due to their unique features and advantages.

Ejaz et al. (2019) investigated the application of principal component analysis (PCA) algorithms in face recognition, focusing on comparing the performance under occlusion and no-occlusion conditions. The background of the study shows that face recognition has attracted much attention due to its wide range of application scenarios, but occlusion (e.g., masks, glasses) significantly reduces the accuracy of recognition. This study uses PCA as the core algorithm to generate the feature space by dimensionality reduction and combines it with the Viola-Jones algorithm for face detection. The experiments use the ORL database and self-collected data, totaling 500 images, of which 300 are used for training and the rest are used as the test set. The results show that PCA has an average accuracy of 95% in unobscured face recognition, while the accuracy in occluded conditions is only 72%, indicating that occlusion has a significant impact on recognition performance. The study concludes that although PCA is suitable for non-obscured face recognition, more advanced machine learning methods should be explored to cope with the occlusion problem and improve recognition performance.

In uncontrolled environments such as video broadcasts, variables such as lighting, pose, and facial expression can significantly affect the accuracy of facial recognition systems. Mady and Hilles (2018) explored the challenges of facial recognition in such environments and their

possible solutions. The study proposes a framework that combines Local Binary Pattern (LBP) and Histogram of Orientation Gradients (HOG) for feature extraction, both of which are known for their robustness and low computational cost. The classification was performed using the Random Forest (RF) algorithm, which was selected for its high accuracy and efficiency, and compared with Support Vector Machines (SVMs). The experiments used the Mediu Staff database, a customized dataset consisting of real video samples. The test results show an average recognition accuracy of 97.6%, which is significantly better than other classifiers. The proposed system employs the Viola-Jones method for initial face detection, followed by feature extraction and classification, and finally, an innovative voting mechanism to improve the reliability of the results. The applicability of the system in real-world scenarios is further demonstrated by validating the implementation in MATLAB.

Pk et al. (2020) developed a multi-class face recognition framework using Support Vector Machines (SVM) and Histogram of Orientation Gradients (HOG) for feature extraction. The study highlights the challenges of coping with different environmental variations (e.g., lighting, facial expression, and orientation changes) in face recognition. The method employs a one-to-one SVM algorithm to classify face images, achieving high accuracy recognition in multi-class scenarios. Experiments are conducted on ORL, YALE, and a self-built database that covers diverse facial features. The results show that the recognition rate of the framework exceeds 96% on all datasets, outperforming many existing methods. The experimental procedure includes the preprocessing of grayscale facial images, feature extraction, and classification using the SVM algorithm. The MATLAB-based implementation validates the system's effectiveness for practical deployment in authentication and surveillance applications.

Guo (2021) investigated the application of the K-Nearest Neighbors (KNN) algorithm to face recognition, especially under challenging conditions such as partial occlusion and

contour views. The study emphasizes the urgent need for reliable contactless recognition systems during the COVID-19 pandemic. The dataset consists of frontal and lateral face images, both unobscured and manually added occlusion images, all of which are taken from online platforms and preprocessed. In the experimental design, the KNN classifier is trained on a subset of unobstructed and occluded frontal faces and tested in six different scenarios including different face orientations and occlusion cases. The results show that KNN performs well in recognizing unobstructed frontal faces with 95% accuracy, but is less effective in recognizing side or occluded faces with a low success rate of 2.22%. The algorithm improves in recognizing partially occluded frontal faces by additional training on partially occluded images, but the overall performance is still insufficient under constrained conditions. The conclusion of the study shows that the KNN algorithm is suitable for scenes with unobstructed frontal faces, but needs to be further optimized for a wider range of real-world applications, and suggests the introduction of advanced feature weighting techniques to improve performance in the future.

Wei (2017) proposed a facial recognition method based on an improved linear discriminant analysis (LDA) algorithm, aiming to solve the problem of inaccurate estimation of class means by traditional LDA models in the case of small samples. The traditional LDA method may contain outliers in the sample set due to the influence of non-ideal conditions such as illumination, expression and gesture variations, which reduces the robustness and recognition performance of the model. For this reason, the researcher proposed a complete fuzzy LDA (CFLDA) algorithm by combining the fuzzy set theory with the K-nearest neighbor algorithm (FKNN). The algorithm calculates the fuzzy affiliation matrix of the training samples through FKNN, and then obtains the fuzzy mean value of each class of samples, and replaces the traditional sample mean with the fuzzy mean value, which is used to redefine the intraclass scatter matrix and interclass scatter matrix. Ultimately, the algorithm obtains the optimal projection matrix for feature extraction by maximizing the objective function of the fuzzy

scatter ratio. In the experiments, the researchers validated the CFLDA algorithm using three datasets, ORL, YALE, and FERET. The ORL database contains 400 images of 40 people, and the experiments were conducted using 3, 5, and 7 images per person as the training samples, respectively; the YALE database contains 165 images of 15 people with significant variations in lighting, expression, and pose; the FERET database was a standard database for testing and evaluating advanced facial recognition algorithms. In terms of experimental methodology, the study designed comparative tests with PCA, 2DPCA, LDA and FLDA methods, evaluated the classification performance using nearest neighbor classifiers with Euclidean distances and improved the stability of the results through cross-validation. The experimental results show that the CFLDA algorithm outperforms the other methods in terms of recognition rate on all datasets, and especially exhibits higher robustness with fewer training samples. For example, on the ORL database, CFLDA achieves a 100% recognition rate when trained using 7 images per person. On the FERET database, CFLDA improves the average recognition rate by about 1% compared to FLDA. This result shows that CFLDA effectively improves the model's performance in dealing with uncertainty and improving facial recognition performance by fully integrating fuzzy set theory into the LDA model.

Jiang and Learned-Miller (2017) explored the application of improved regional convolutional neural networks (Faster R-CNN) in face detection tasks, aiming to address the limitations of traditional R-CNN methods in terms of accuracy and speed. Despite the significant progress of deep learning in the field of target detection, the development of facial detection methods is relatively lagging. To this end, the researchers propose a Faster R-CNN-based framework combined with the large-scale WIDER face dataset to significantly improve the performance of facial detection. Faster R-CNN achieves efficient generation of target region proposals and accurate classification by combining the region proposal network (RPN) with the Fast R-CNN module, which shares the properties of convolutional layers. This

architecture not only significantly reduces the computational overhead, but also allows the extraction of high-quality features using deep networks such as VGG16. Three benchmark datasets, WIDER, FDDB, and IJB-A, are selected for experimental validation. The WIDER dataset contains 12880 images and 159424 faces covering a challenging variety of scales, poses, and densities; the FDDB dataset focuses on the task of face detection in complex scenes and occlusions; and the IJB-A dataset emphasizes the pose, expression, and light significant variations. In the experiments, the researchers trained the model on the WIDER dataset, optimized it with 50,000 iterations using stochastic gradient descent (SGD), and further improved the performance with an additional 30,000 fine-tuning. The evaluation method uses discrete and continuous scores to analyze the model performance through ROC curves, as well as comparing the performance with R-CNN and Fast R-CNN methods and comparing the performance of proposal generation modules such as EdgeBox and Faceness. The experimental results show that Faster R-CNN exhibits excellent performance on all test datasets. In the WIDER dataset, Faster R-CNN outperforms other methods in “easy”, “medium”, and “difficult” scenarios, especially when dealing with thick This is especially true for dense targets and complex scenarios. In the FDDB dataset, the true-positive rate of this method is significantly higher than that of other models, and the computational efficiency is greatly improved compared with traditional methods. On the IJB-A dataset, the detection ability of the model is further enhanced by a fine-tuning strategy adapted to the annotation style. The results show that Faster R-CNN not only achieves higher accuracy in facial detection but also demonstrates strong robustness in complex scenes, pointing out the direction for further optimizing the performance of facial detection by introducing contextual information in the future.

Li et al. (2015) proposed a cascade architecture based on convolutional neural networks (CNNs) for fast and efficient face detection in response to the challenges of face detection in the real world. The research background points out that traditional facial detection methods

(e.g., Viola-Jones algorithm) are not robust enough to cope with complex conditions such as pose, expression, and illumination changes, while convolutional neural networks can effectively solve these problems by automatically learning features. However, due to the high computational overhead of CNNs, direct multi-scale scanning of the whole image is not feasible in practice. To this end, the researchers designed a CNN cascade architecture that balances detection accuracy and efficiency by quickly screening out background regions in the low-resolution stage and accurately evaluating a small number of candidate regions in the high-resolution stage. In addition, to further improve the localization accuracy and reduce the number of candidate regions in subsequent stages, the architecture incorporates a CNN-based calibration module after each detection stage. The study uses two public datasets for experimental validation: Annotated Faces in the Wild (AFW) and Face Detection Dataset and Benchmark (FDDB). In the experimental methodology, the researchers designed three CNN modules (12-net, 24-net, and 48-net) with step-by-step resolution increase and their corresponding calibration networks to gradually filter a large number of candidate windows through a multi-stage cascade structure. During the training process, face samples from the AFLW dataset were used as positive samples, while negative samples were generated by random sampling from non-face backgrounds. In the testing phase, the researchers used sliding windows to scan the images densely and further reduced the overlapping detection windows by non-maximum suppression (NMS) at each stage. The experimental results show that the performance of the method exhibits significant advantages on both datasets. In the AFW dataset, the detection accuracy is comparable to the state-of-the-art existing methods; in the FDDB dataset, the method outperforms all existing methods in intermittent score evaluation and approaches the top model in continuous score evaluation. In addition, the computational efficiency of this method is significantly better than traditional methods, which can process VGA images at 14 frames per second (FPS) on a single-core CPU and further accelerate to 100

FPS on a GPU. In summary, this CNN-based cascade architecture not only achieves fast and accurate face detection but also significantly improves the localization accuracy and efficiency of detection by introducing a calibration module. Future research can explore optimizing this method for more complex scenarios, including dense target detection and extreme lighting conditions.

Winarno et al. (2019) proposed a facial recognition attendance system based on a hybrid CNN-PCA approach to overcome the shortcomings of traditional attendance systems in terms of accuracy and efficiency. The background of the study states that traditional facial recognition techniques do not perform well enough to cope with complex conditions such as lighting, expression, and gesture changes, while convolutional neural networks (CNNs) provide an effective solution by automatically extracting high-dimensional features, while principal component analysis (PCA) further improves the efficiency through feature dimensionality reduction. The system captures facial images from a real-time camera and matches them with face information from a database to achieve efficient and reliable attendance recording. The research methodology covers five phases: data acquisition, facial detection, preprocessing, feature extraction and classification. In the facial detection stage, the system uses the Viola-Jones algorithm implemented based on OpenCV to detect and locate the facial region, generating the original image containing the background and face. In the preprocessing stage, cropping, gray scaling, histogram equalization, and brightness contrast adjustment are applied to mitigate the effect of lighting variations on image quality. In the feature extraction process, CNN is used to reconstruct the 2D face image into a 3D image to extract key features from the high-dimensional data; subsequently, PCA downsizes the features to improve processing efficiency. The classification stage uses the Mahalanobis distance method to compare the test features with the training features in the database to complete the facial recognition. Experimental results validate the effectiveness of the method. Tests show that the

recognition accuracy using PCA alone is 90%-96%, while the accuracy is increased to 90%-98% with the combination of CNN-PCA. The system demonstrated excellent real-time performance and reliability and was able to quickly generate high-precision attendance records. The conclusion of the study shows that the CNN-PCA method is effective in improving the accuracy and robustness of facial recognition, and provides strong technical support for the facial recognition-based attendance system. Future work suggests further optimization of the 3D reconstruction model to adapt to more complex scenes and environmental requirements.

In recent years, facial recognition technology has made significant progress, thanks to the combination of traditional machine learning methods and deep learning techniques. Researchers have made breakthroughs in facial recognition performance and efficiency in complex environments by innovatively improving existing algorithms such as PCA, LDA, SVM, and KNN, and combining deep learning models such as CNN. Nevertheless, existing methods still have limitations in the face of challenges such as occlusion, illumination, pose change, and dense target detection. By introducing hybrid methods (e.g., CNN-PCA) or optimizing existing frameworks (e.g., Faster R-CNN, cascaded CNN architectures), the research further improves the robustness and real-time performance of the models, providing stronger support for facial recognition technology in practical applications such as time and attendance, surveillance, and identity verification.

2.4 Comparison of Related Works

In this study, a variety of machine learning and deep learning algorithms were selected and evaluated for face recognition tasks, including Principal Component Analysis (PCA), Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), and Convolutional Neural Network (CNN). Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), and Convolutional Neural Network (CNN). In order to have a comprehensive understanding of the performance and applicability of these algorithms, this chapter conducts a systematic review of academic papers in related fields and compares the performance of each algorithm in different studies, and the results are shown in Table 2.1.

No.	Reference	**Evaluation Metrics**	Brief Description	Machine Learning Algorithms & Deep Learning Algorithms	Strengths	Limitations
1	Ejaz et al. (2019), *Implementation of Principal*	Accuracy: • 72%: masked face	The study applies PCA to masked and non-masked face recognition, leveraging its feature dimensionality	PCA	Effective at reducing data dimensionality while preserving	Limited robustness to illumination and pose variations.

	Component Analysis on Masked and Non-masked Face Recognition	• 95%：non-masked face	reduction capabilities. Comparisons were made between these two scenarios using datasets augmented with real-world masked and non-masked face images.		essential features; works well for non-masked faces.	Poor performance under severe occlusions.
2	Mady & Hilles (2018), *Face Recognition and Detection Using Random Forest and Combination of LBP and HOG Features*	Accuracy: • 97.6%	This paper combines LBP (Local Binary Patterns) and HOG for feature extraction and uses the Random Forest classifier for video-based real-time facial recognition in uncontrolled environments. It evaluates results on Mediu Staff video datasets.	Random Forest	Low computational cost, efficient in real-time scenarios, and effective under illumination variations.	Performance is sensitive to occlusions, complex backgrounds, and severe pose variations.

3	Pk et al. (2020), *An Integrated Approach for Face Recognition Using Multi-class SVM*	Accuracy: 96-97%	The study introduces a multi-class SVM approach for facial recognition using HOG (Histogram of Oriented Gradients) features. The "one-versus-one" SVM technique is employed to handle multi-class problems, and experiments were conducted on ORL and YALE databases.	SVM	High accuracy in small-sample, high-dimensional datasets. Robust to various facial variations.	Performance depends heavily on hyperparameter tuning; sensitive to noise in high-dimensional data.
4	Guo (2021), *A KNN Classifier for Face Recognition*	Accuracy: • 95%: uncovered frontal faces • 22.2%: profile or masked faces	This paper investigates the KNN classifier for face recognition under diverse conditions, including profile views and masked faces. A weighted voting strategy is	KNN	Simple and intuitive algorithm, adaptable to diverse datasets, and effective in small datasets.	Computationally expensive for high-dimensional data; poor performance in

			employed to improve classification under limited datasets, especially in profile and occluded scenarios.			recognizing occluded faces.
5	Wei (2017), *Face Recognition Method Based on Improved LDA*	Accuracy: • ORL database: 94.50% • YALE database: 91.80% • FERET database: 53.29%	The paper proposes an improved fuzzy LDA (Linear Discriminant Analysis) method using fuzzy set theory to address outliers in training samples. Experiments demonstrate its robustness on ORL, YALE, and FERET datasets under variations in illumination and pose.	LDA	Excels in small-sample size problems, effective in handling class separability, and robust to outliers.	Struggles with intraclass variability; sensitive to insufficient data diversity.

6	Jiang & Learned-Miller (2017), Face Detection with Faster R-CNN	Area under P-R curve: • Easy set: 0.903 • Medium set: 0.836 • Hard set: 0.464	This study explores Faster R-CNN for face detection using the WIDER dataset, achieving state-of-the-art results on FDDB and IJB-A benchmarks. Faster R-CNN integrates a Region Proposal Network (RPN) to generate efficient object proposals.	CNN	High accuracy and robustness to occlusions; end-to-end feature learning with efficient region proposals.	Computationally expensive during training, requiring significant resources.
7	Li et al. (2015), A CNN Cascade for Face Detection	Area under P-R curve: 0.980	This paper proposes a CNN cascade architecture for efficient face detection across multiple resolutions. Calibration steps are incorporated at each stage to	CNN	Efficient rejection of false positives; high speed and accuracy for real-time detection.	Performance declines with occluded faces and very small input sizes.

			improve detection precision and reduce false positives.			
8	Winarno et al. (2019), Attendance System Based on Face Recognition Using CNN-PCA	Accuracy: 98%	This study develops a face recognition-based attendance system using a hybrid CNN-PCA approach. PCA reduces dimensionality while CNN enhances feature extraction to improve recognition performance.	CNN	High accuracy and efficient dimensionality reduction.	Limited robustness to lighting variations despite preprocessing.

Table 2.1 : Comparison of Previous Work

2.5 Comparison and Selection of Face Recognition Algorithms

According to the comparative analysis in Table 2.1, CNN shows a significant advantage in the face recognition task with an accuracy of up to 98%, which is significantly better than other algorithms such as PCA, Random Forest (RF), Support Vector Machine (SVM), K Nearest Neighbors (KNN), and Linear Discriminant Analysis (LDA).

CNN is able to automatically learn multi-level, complex features from images, avoiding the limitations of PCA and LDA that rely on dimensionality reduction, making it more efficient in processing high-dimensional image data. At the same time, CNN is able to effectively cope with problems such as illumination changes, pose changes and background interference through convolutional operations and pooling layers, which makes its performance especially good in standard datasets and shows excellent robustness. Combined with migration learning and regularization strategies (e.g., Data Augmentation and Dropout), CNN not only performs well on small- and medium-sized datasets, but also adapts well to large-scale datasets.

Although the training process of CNN is time-consuming and requires high computational resources, its significant advantages in performance, accuracy and adaptability make it the most recommended algorithm for face recognition tasks.

2.6 Related Technology Stack

The main technology stack used in this system includes Python, OpenCV, TensorFlow, Kotlin, Jetpack Compose, Firebase, and Java. Each technology has its unique role and advantages.

Firstly, Python has been chosen as one of the programming languages for the development of the system and is mainly used for processing datasets, face detection in images, data enhancement, and training of deep learning models. Python has a concise syntax and a rich set of image processing and machine learning libraries such as OpenCV and TensorFlow, enabling developers to efficiently process data and train deep learning models.

OpenCV is an open-source computer vision library mainly used for face detection in this project. Using OpenCV, the system can process the image to identify and extract the face region. In this way, the system focuses only on the face region rather than the entire image, reducing computational complexity and improving recognition efficiency.

For model training, TensorFlow is used to build and train machine learning and deep learning models. TensorFlow provides powerful tools and libraries to build and train deep neural networks. In this project, TensorFlow was used to create an efficient face recognition model to process image data and train the neural network.

In application development, Kotlin and Jetpack Compose are used to build user interfaces. Kotlin is the recommended language for Android development, and its combination with Jetpack Compose makes UI development more efficient and concise. Jetpack Compose provides a declarative programming approach that simplifies the UI building process, allowing developers to focus on implementing functionality rather than layout details.

To manage user data and photos, the project uses Firebase. Firebase Authentication is used for user authentication, Firestore is used to store and manage user information, and Firebase Storage is used to store student photos. Synchronization and security of data is ensured through real-time database and storage services provided by Firebase.

For back-end development, Java was chosen as the primary programming language for building RESTful API services and handling requests and responses from the Android application. Java's responsibilities included data interaction with Firebase, managing user data, and transmitting student photos to the recognition model.

To summaries, each of the above technology stacks has its own role to play, from data processing, and model training to UI development and back-end management, all of which provide strong support for the development of the face recognition system. The selection of each technology has been fully considered to ensure that the system can run efficiently and stably while providing a good user experience.

2.6.1 Python

Python is a cross-platform, general-purpose interpreted high-level programming language that runs seamlessly on operating systems such as Windows, macOS, and Linux. In recent years, Python has been used in a wide range of applications such as data science, software development, web development, automation, and integrated task completion (Staff, 2024). Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than C (Summerfield, 2007). Python's language structure supports users in writing clear small and large programs (Kuhlman, 2011). Its important feature is its support for multiple programming paradigms, including object-oriented, imperative, functional, and procedural programming (Srinath, 2017). Python supports a dynamic type system and automatic memory management and has a large and comprehensive

standards library. According to a study by Statista, Python is the third most frequently used programming language by developers worldwide (Statista, 2024).

2.6.2 OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library (TheKnowledgeAcademy, n.d.). OpenCV is a collection of algorithms, C/C++ functions and some classes. The library contains a complete general-purpose machine-learning library focused on statistical pattern recognition and clustering (Boesch, 2024). Currently, the library offers more than 2,500 algorithms, including classic algorithms such as Support Vector Machines (SVMs), K-neighborhood Algorithms (KNNs), as well as state-of-the-art deep learning techniques (Writer, 2024). OpenCV is widely used for a variety of computer vision tasks, including detecting and recognizing faces, recognizing objects, classifying human actions in video, tracking moving objects, stitching images to produce high-resolution images of an entire scene, finding similar images from image databases, and recognizing a scene and creating markers for overlaying it with augmented reality techniques (OpenCV, 2020). In recent years, OpenCV has increased its integration with deep learning frameworks. With its DNN module, developers can load and run models trained in frameworks such as TensorFlow, Caffe, PyTorch and ONNX, making it easy to apply deep learning to computer vision tasks.

2.6.3 TensorFlow

TensorFlow is a machine learning system that can run in large-scale heterogeneous environments, using the concept of a data flow graph with nodes and edges (BasuMallick, 2022b). Nodes in a data flow graph represent mathematical operations, and edges represent multidimensional arrays (tensors) of data flowing between nodes. The workflow of TensorFlow consists of three main parts: data preprocessing, model building, and predictive model training (What Is TensorFlow, n.d.). It can map nodes of the data flow graph to multiple computers in

a cluster, or multiple computing devices on a single machine, including multi-core CPUs, general-purpose GPUs, and specialized tensor processing units (TPUs) (TensorFlow: A System for Large-scale Machine Learning, n.d.). Unlike traditional “parameter server” designs, TensorFlow allows developers to experiment with new optimization and training algorithms while supporting a wide range of applications and has demonstrated its power, especially in deep neural network training and inference (Abadi et al., 2016).

2.6.4 Kotlin

Kotlin is a statically typed, object-oriented programming language widely used for Android mobile application development and other multi-platform application development (Lutkevich, 2022). Kotlin was developed by the global software company JetBrains and refined in conjunction with open-source contributors and was officially launched in 2011. With the widespread use of Kotlin in Android development and cross-platform applications, Google announced a Kotlin-first approach in 2019 (Android's Kotlin-first Approach, n.d.). Kotlin was originally designed to improve the Java programming language. Originally designed to improve the Java programming language, Kotlin is intended to run on the JVM (Java Virtual Machine) and interoperate seamlessly with Java code; Kotlin's interoperability with the JVM makes it compatible with existing Java codebases and allows organizations to take full advantage of the Java ecosystem (Silverio, 2022). It combines features of both object-oriented and functional programming, emphasizing code readability and simplicity, and is particularly efficient when declaring objects and classes with many attributes (Hellbrück, 2019).

2.6.5 Jetpack Compose

Android Jetpack Compose is a declarative UI toolkit for Android. Compose makes it easier to write and maintain application UI by providing a declarative API that renders the application UI without forcing changes to the front-end view (Thinking in Compose, n.d.). Compose is a fundamental rethinking of the way Android UI is written. The technique works

by regenerating the entire screen from scratch and then applying only the necessary changes. This approach avoids the complexity of manually updating the state view hierarchy. UI layouts and behaviour that used to be implemented across multiple files and formats can now be represented as composable function trees (Partners, 2024).

Jetpack Compose offers several significant advantages over the traditional Android View system. First, Compose reduces the need to write sample code, making UI development more concise. With a declarative syntax, developers only need to describe the final state of the UI and Compose handles view updates automatically. This is different from traditional View systems, which often require tedious callbacks and view updates to keep the state of the interface consistent. Second, Compose seamlessly supports complex animations and transitions, making it easier to achieve a smooth UI interaction experience. In addition, Compose makes the separation between UI and business logic much clearer, reducing problems caused by incorrectly updated view hierarchies.

Jetpack Compose introduces several performance improvements. For example, see (Arora, 2022) and recognize the significant reduction in freeze frames and the reduction in median page load duration. Compose also improves build times and APK sizes. At this point, the main performance problems are caused by composable that may or may not be skipped during recomposition (Trengrove, 2022).

2.6.6 Firebase

Firebase is a powerful toolset for building, improving, and extending applications (Stevenson, 2020). Firebase is based on the NoSQL database model, and instead of storing data in a traditional tabular format (rows and columns), Firebase uses the JavaScript Object Notation (JSON) format. This structure allows Firebase to avoid using traditional query language when inserting, updating, deleting, or adding data, thus simplifying the data manipulation.

Firebase provides two main database services: Firebase Realtime Database and Firestore. Firebase Realtime Database is a JSON-based real-time cloud database that supports instant synchronization of data across multiple clients, especially for applications requiring real-time data updates. In contrast, Firestore is a more powerful and flexible NoSQL database that offers more complex queries, greater scalability, and more efficient data storage, especially for applications that need to handle large amounts of data and complex queries.

Firebase Storage is powered by Google Cloud Storage, which developers can use to store user-generated content such as images, audio, video, and more (Khawas & Shah, 2018). The seamless integration of this service with Google Cloud makes it easy for developers to manage and store files in the cloud, ensuring data security and scalability.

Firebase Authentication provides an easy user authentication solution (GeeksforGeeks, 2024) that supports multiple authentication methods including email/password, social media accounts (e.g., Google, Facebook, Twitter), and anonymous authentication. With Firebase Authentication, developers can easily implement user registration, login, password reset, and account management features while ensuring data and application security.

2.6.7 Java

Java is a widely used class-based object-oriented high-level programming language designed to realize the concept of “write once, run anywhere” (WORA). This means that compiled Java code can run on all Java-enabled platforms without having to be recompiled (Everythingcomputerscience, 2016). Java is known for its simplicity, portability, and platform independence, making it a popular choice for a wide range of scenarios, including mobile apps, web apps, enterprise software, and the Internet of Things (IoT), etc. Java's platform independence is largely due to its unique compilation mechanism: Java programs are converted to bytecode when compiled, rather than operating system-specific machine code. Java programs are compiled into bytecode rather than operating system-specific machine code.

These bytecodes are platform-independent and can be run in any environment that supports Java, allowing developers to achieve cross-platform compatibility without having to be concerned about platform differences (Hachadi, 2023). In addition, Java is widely used in industry and academia and is the language of choice for many organizations when building enterprise-level Internet solutions. This popularity and strong ecosystem further solidifies Java's importance as a programming language.

2.7 Existing Attendance Method

Attendance methods play an important role in meetings, events, and classes; they not only record the attendance of participants, but also help organizers manage data efficiently. As technology continues to evolve, many check-in methods have emerged, including Manual Attendance System, QR code Attendance System, NFC/RFID Attendance System, GPS-Based Attendance System, Face Recognition Attendance System and Fingerprint Recognition Attendance System. Each method has its unique characteristics and specific use cases.

Manual Attendance System

Manual attendance systems are a simple and intuitive way to record information such as participants' names, time of arrival, etc., through paper sign-in sheets or notebooks (Jibble, 2025), and are widely used for small events due to their lack of technical support and low cost. Its flexibility allows sign-in sheets to be customized according to needs, but inefficiency and high risk of errors are its main drawbacks, especially when the number of participants is large and is prone to problems such as incorrectly filled in information, omissions, or signature substitutions, and time-consuming data collation. The limitations of this method are even more apparent at large events, where it is difficult to ensure the timeliness and accuracy of the data, so despite its low cost, its applicability in modern event management is gradually diminishing.

QR code Attendance System

QR code check-in is widely used in meetings, trainings, events and other occasions due to its efficient and convenient features. Participants can complete check-in by simply scanning the QR code with their cell phones (Manori et al., 2017), without the need to queue up or manually fill in the information, which significantly improves the efficiency of check-in. It supports large-scale check-in and real-time uploading and synchronization of data, which

provides an accurate basis for subsequent statistics and analysis. In addition, QR code check-in can be combined with identity verification functions, such as binding a cell phone number or limiting the number of times the code is scanned, thus enhancing security and accuracy. However, this approach relies on a stable network environment and smart devices, and the check-in process may be affected if the network is poor or the devices fail. In addition, the easy dissemination of QR codes brings the risk of cheating, and unauthorized sharing may lead to data distortion. Therefore, additional safeguards, such as dynamic QR codes or login authentication, are required when implementing QR code check-ins to minimize potential risks and ensure the reliability of check-ins.

NFC/RFID Attendance System

As NFC and RFID technologies continue to mature, check-in based on these two technologies is favored for its efficiency and safety. Attendees can check in by simply placing an NFC tag or RFID card close to a sensing device (Saxena, 2021), eliminating the need for manual operations and making the check-in process extremely fast and simple, greatly improving efficiency. At the same time, due to the encrypted transmission of data and the need for close proximity sensing equipment, security is much higher than the traditional way, almost eliminating the possibility of signature or data tampering, so it is particularly suitable for occasions with high security requirements, such as corporate meetings or academic seminars. In addition, this method can also realize real-time data transmission and automatic storage, which is convenient for subsequent analysis and management. However, its limitations lie in its reliance on specialized hardware devices and tags, and high initial investment costs, especially for large-scale events. At the same time, the failure of sensing devices or damage to cards may lead to interruptions in check-in, so it is necessary to have a back-up solution to cope

with unforeseen circumstances. Nonetheless, NFC/RFID check-in is the modern check-in technology of choice for its speed and security advantages.

GPS-Based Attendance System

Global Positioning System (GPS)-based time and attendance systems utilize positioning technology to allow attendees to complete check-ins via mobile devices within a specific area (Grover, 2025), offering a high degree of flexibility due to the fact that it eliminates the need for a physical device for decentralized or remote scenarios. This approach is particularly well suited to the attendance needs of cross-regional teams, outdoor events or remote work, and can be used in conjunction with technologies such as QR codes, fingerprints or face recognition to enhance authentication accuracy and data reliability. In addition, GPS check-in system can record real-time location and time information, providing detailed data for subsequent attendance statistics and abnormal situation investigation. However, its limitation lies in its high dependence on network and device performance, which may lead to check-in failure or delay when the signal is poor or the device performance is insufficient. In addition, the existence of analog positioning technology also poses a certain risk to security, and anti-cheating measures are required to identify false locations. Nonetheless, GPS time and attendance systems have become a popular method of check-in in modern scenarios due to the convenience and scalability of remote operation.

Face Recognition Attendance System

Facial recognition attendance is an efficient way of signing in using a camera to capture images of attendees' faces and compare them with a pre-stored face database (Smitha et al., 2020). This non-contact method is hygienic and convenient, can effectively avoid the risk of contact infection, and its accuracy is high, not easy to appear the phenomenon of fraudulent

signing or signing on behalf of the attendees, and also can automatically record the attendees' signing time, which provides a great convenience for the subsequent data statistics and management. In addition, face recognition attendance can reduce manual operation and improve work efficiency, and has wide application potential in large-scale meetings, school classes and corporate office scenarios. However, the system is sensitive to environmental factors such as light intensity, camera angle, facial occlusion, and other conditions, which may affect the recognition effect; at the same time, the system's pre-development and subsequent maintenance require high-cost investment. What's more, privacy protection of face data is a key issue, which, if not properly managed, may trigger the risk of information leakage or misuse, which in turn poses challenges to user trust and technology promotion. Therefore, in practical applications, it is necessary to comprehensively consider factors such as technology optimization, privacy protection and cost control to ensure the security and reliability of face recognition attendance systems.

Fingerprint Recognition Attendance System

Fingerprint recognition check-in is an efficient technology for identity verification through fingerprint scanning devices that relies on the uniqueness and stability of an individual's fingerprint to ensure accuracy and security (Akinduyite et al., 2013). This technology is widely used in airports, self-service check-in, and other scenarios that require quick identity verification because fingerprints are difficult to forge and can significantly reduce the risk of identity fraud. Compared with traditional identity verification methods, fingerprint identification requires no additional documents, and the operation is simple and convenient, which greatly improves work efficiency. However, fingerprint identification also has certain limitations, such as a high dependence on fingerprint clarity, which may fail to recognize groups with severely worn fingerprints (e.g., manual laborers); at the same time,

damage to the device or defacement of the sensor may also affect the normal use of the device. In addition, fingerprint identification is a contact operation, which is not suitable for occasions with high hygiene requirements, such as during epidemics of infectious diseases, where there may be a potential risk of cross-infection. Therefore, it is necessary to optimize the technology for specific scenarios when promoting and applying it, and at the same time equip other verification methods as a supplement to enhance the applicability and reliability of the system.

2.8 Summary

This chapter comprehensively analyzes machine learning and deep learning algorithms for face recognition, focusing on their applications in attendance management systems. Machine learning methods such as PCA, LDA, RF, SVM, and KNN excel in feature extraction and classification, but have limitations in complex scenarios. In contrast, deep learning (especially CNN) significantly outperforms traditional algorithms by virtue of its ability to learn hierarchical features with up to 98% accuracy on standard datasets and good robustness to illumination, pose, and background variations. Combining hybrid models (e.g., CNN-PCA) and advanced frameworks (e.g., Faster R-CNN) can further improve recognition accuracy and efficiency. This chapter also emphasizes the key role of technology stacks such as Python, OpenCV, and TensorFlow in developing efficient and scalable systems. Furthermore, various attendance methods from manual attendance systems to modern technologies such as NFC/RFID, GPS, and biometrics are systematically compared, with each method having its own unique strengths and limitations. This chapter further points out that improving algorithm accuracy, solving privacy issues, and optimizing for complex real-world scenarios are important directions to drive the continued development of face recognition technology.

Chapter 3: Methodology

3.1 Introduction

The Cross Industry Standard Process for Data Mining (CRISP-DM) methodology be used to evaluate machine learning algorithms and deep learning algorithms. The machine learning algorithms used in this project include Principal Component Analysis (PCA), Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Linear Discriminant Analysis (LDA); and Convolutional Neural Networks (CNN) will be used for the deep learning algorithms. Through a comparative study of different models, the best model will be finally determined. Chapter 3 discuss in detail the CRISP-DM method applied in this project.

3.2 Cross Industry Standard Process for Data Mining (CRISP-DM)

The CRISP-DM framework is the core methodology for developing and evaluating AI-based face recognition time and attendance applications. Covering six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment, the framework provides a systematic and flexible process to ensure that the development process is well-structured, rigorously executed, and capable of iterative improvement. By applying the CRISP-DM framework, this study was able to deeply understand the requirements of the attendance system, effectively collect, clean and analyze the face recognition data, and identify key features in combination with domain knowledge to develop a high-performance predictive model to satisfy the objectives of attendance management.

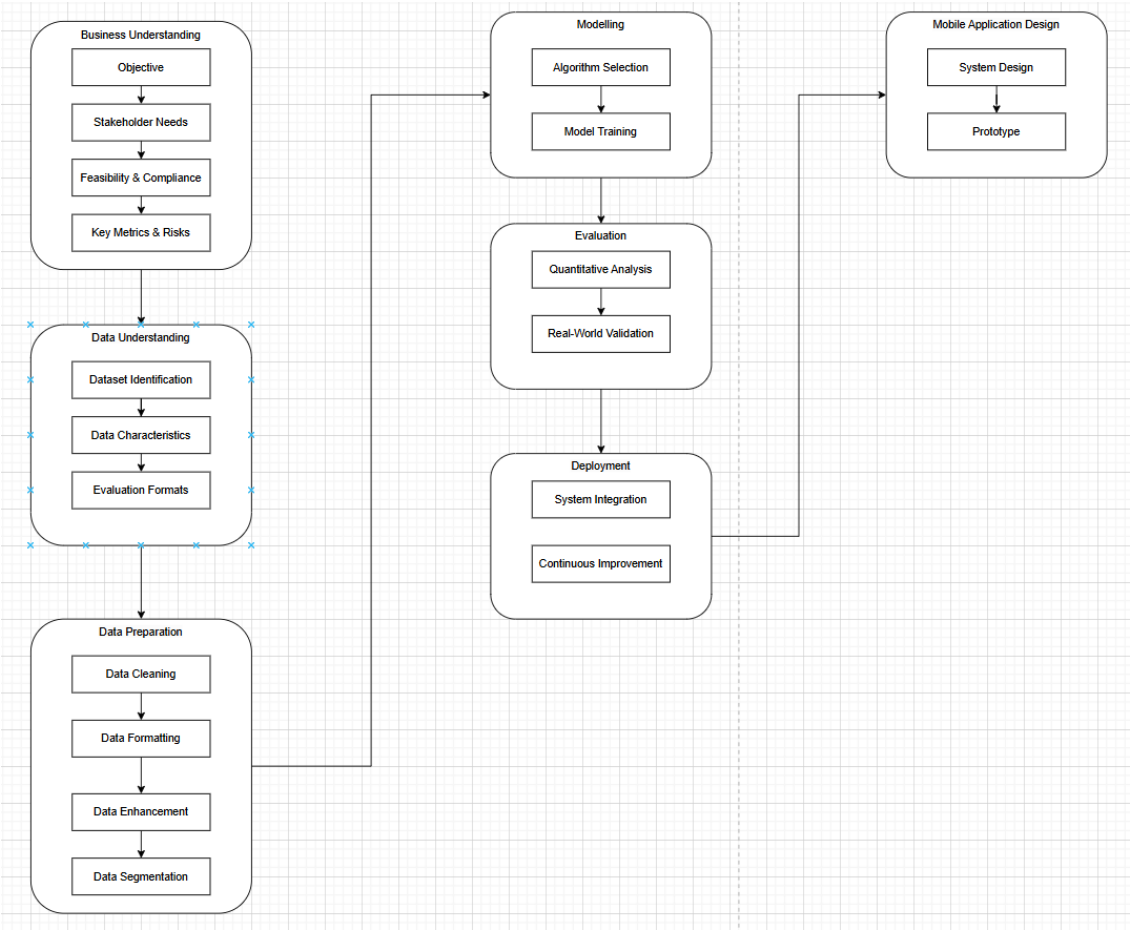


Figure 3.1 : Framework of CRISP-DM

In the modeling phase, the project uses machine learning algorithms such as Principal Component Analysis (PCA), Random Forest (RF), Support Vector Machine (SVM), K Nearest Neighbors (KNN), and Linear Discriminant Analysis (LDA), as well as Convolutional Neural Networks (CNN), which is a deep learning method. Machine learning models are mainly used for feature dimensionality reduction and classification tasks, while deep learning models focus on extracting high-level features from complex facial images. This combination of approaches ensures comprehensive and accurate recognition performance. In the evaluation phase, the models are tested using rigorous performance metrics, and the algorithm parameters are adjusted for the needs of the application scenarios to optimize their performance in real-world deployments.

In addition, the CRISP-DM framework demonstrates significant advantages in biometric data management. Its structured process makes data collection, preprocessing and feature engineering more efficient, while improving data consistency and quality. The iterative nature of the framework allows research to continuously improve model performance and adapt to new needs and challenges in the face of dynamic data changes. This feature is particularly important for evolving time and attendance systems.

By adopting the CRISP-DM framework, this project develops an AI-based face recognition time and attendance application in a systematic way, ensuring high accuracy, reliability, and excellent scalability of the model, which provides an innovative and practical solution for the field of time and attendance management.

3.2.1 Business Understanding

During the business understanding phase, the goal of the AI-driven face recognition attendance application project was clarified as replacing traditional manual attendance methods with an automated system to improve the efficiency and accuracy of attendance. The core objectives included preventing the occurrence of proxy signing issues and improving the authenticity and security of attendance records. The project identifies key stakeholders such as students, professors and administrators and analyses their specific needs in order to design user-friendly attendance and management features.

The technical feasibility assessment included the rationalization of using face recognition and artificial intelligence technologies, taking into account budget, time, and resource constraints. In addition, the project examined the legal and ethical implications of processing biometric data to ensure compliance with data privacy regulations. The project established success criteria, such as face recognition accuracy of 98% or higher and response time of less than two seconds, to assess project outcomes.

At the same time, the project predicted possible risk factors, such as environmental challenges like low light levels and potential user resistance to new technologies, and developed mitigation strategies accordingly. Finally, the project also highlighted potential benefits, such as reduced administrative workload and improved reliability of attendance data, to ensure that the project objectives were aligned with the organization's needs.

3.2.2 Data Understanding

In the data understanding phase, for the AI-driven face recognition attendance application, we need to deeply analyse the dataset used, the LFW (Labelled Faces in the Wild) dataset. In this phase, it is important to first identify the source of the data. The LFW dataset contains more than 13,000 facial images from the Internet, with 5,749 different individuals involved in the images, which are mainly used for facial recognition and verification tasks. Although this data is not specific to students and professors, it provides a rich sample for the evaluation of facial recognition algorithms covering a wide range of lighting, angle, expression, and background conditions, which is highly informative for the performance of this project in the face recognition attendance application.

Characteristic	**Value**
Number of images	13,233
Number of classes	5,749 (distinct individuals)
Image dimensions	250 × 250 pixels (JPEG format)
Images per class	Ranges from 1 to several hundred
Variation in data	Pose, expression, lighting, occlusion, age, gender, race, background
Collection setting	Unconstrained, real-world environments

Table 3.1: Key Characteristics of the LFW dataset

The diversity of the LFW dataset is reflected in its shooting conditions. The uncontrolled environment in which these images are captured results in the presence of large variations in the images, such as different facial poses, expressions, lighting, and backgrounds. This uncontrolled condition makes the dataset an ideal tool for testing the robustness and adaptability of face recognition algorithms in real-life situations.

The LFW dataset provides two main data formats for the facial verification task: the Pairs format, which contains pairs of images labelled to see if they belong to the same person, and the People format, which categorizes images by person. Both formats can provide a testing basis for the facial recognition time and attendance system, helping to evaluate the performance of different algorithms in the face of diverse facial data.

In terms of data quality, the LFW dataset, while having better labelling accuracy, still needs to deal with noise in the dataset, such as poor image quality or partial mislabelling. Some preprocessing operations, such as removing low-quality images and ensuring labelling accuracy, are also required to ensure the applicability of the data in time and attendance systems. In addition, Exploratory Data Analysis (EDA) will help to identify potential patterns, outliers, and feature distributions in the data to support algorithm selection and data preprocessing in the subsequent modelling phase.

Thus, while the LFW dataset is designed with the goal of conducting facial verification and face recognition research, its diversity and challenging nature make it ideal to be used as a benchmark dataset for the AI-driven face recognition time and attendance application in this project.

3.2.3 Data Preparation

In the Data Preparation phase, a series of necessary preprocessing steps are performed on the collected raw data to ensure that the data is suitable for subsequent model training and testing. The core objective of this phase is to create a high-quality, standardized and consistent dataset through systematic processing to provide a solid foundation for subsequent modelling and training.

First, data cleaning is the first step in data preparation, aiming to remove invalid, low-quality or redundant samples. Especially in attendance application scenarios, there may be some images that contain multiple people or have cluttered backgrounds, and this type of data needs to be removed by image preprocessing. It is important to ensure that all images have high quality and remove samples that are blurry, low resolution, and have serious background interference. It is also necessary to ensure labelling accuracy and make sure that the label of each image is consistent with the actual identity to avoid the negative impact of wrong labels on model training.

Next, data formatting and conversion are key steps to ensure data consistency. To ensure that the input data can be adapted for model training, all images need to be resized to a uniform size. In addition, the pixel values of the images need to be normalized by converting the range of values from 0-255 to 0 to 1, which helps speed up the training process. Label formats also need to be converted according to the task requirements.

After data cleaning and formatting, data enhancement is a key step to improve the generalization ability of the model. Techniques such as rotation, cropping, flipping, panning and scaling are used to enhance the diversity of training data and avoid model overfitting. In attendance scenarios, considering the variability of actual shooting angles, expressions, and

lighting, the image enhancement strategy can also include simulating different expression changes and wearing masks to enhance the model's adaptability in real-world environments.

In addition, data segmentation is crucial. At this stage, the dataset needs to be reasonably divided into training, validation and testing sets. The training set accounts for 70%-80% of the dataset for model training; the validation set accounts for 10%-15% for hyper-parameter tuning; and the test set accounts for 10%-15% for the final evaluation of the model's performance to ensure its generalization ability.

For the problem of category imbalance in the dataset, data balancing treatment is necessary. In attendance applications, the number of images of certain individuals (e.g., professors) may be low while the number of images of students is high, resulting in data imbalance. At this point, the data can be balanced by over-sampling or under-sampling, or generating synthetic samples using Generative Adversarial Networks (GAN) to enhance the diversity of the training data.

In terms of feature selection and extraction, Convolutional Neural Networks (CNNs) are able to automatically extract key features from facial images, but in some cases, manually extracting features can still improve model performance. For example, features can be extracted based on facial key points (e.g., locations of eyes, nose, mouth, etc.), or information such as facial lighting and contrast can be analysed to further improve recognition accuracy.

Finally, data storage and management is the basis for ensuring data security and privacy protection. All data should be stored in a unified format to facilitate subsequent processing. Considering that attendance applications involve sensitive personal information, it is important to ensure data encryption and access control measures during storage and transmission to ensure data privacy and security. In addition, a sound data backup mechanism is established to avoid data loss or damage.

In summary, in the data preparation stage, a series of steps such as cleaning, formatting, enhancement, and segmentation are performed to ensure the quality of the data and to improve its diversity and robustness, so as to lay a solid foundation for the subsequent model training. These operations can not only improve the accuracy of the model, but also ensure the stability and practicality of the model in the actual attendance scenario.

3.2.4 Modelling

In the modelling phase, this study explores a variety of algorithms, including classical machine learning methods such as Support Vector Machines (SVMs), Random Forests (RFs), and K-Nearest Neighbours (KNNs), as well as deep learning methods based on Convolutional Neural Networks (CNNs). Each algorithm will be evaluated on a standardized test set to analyse its performance in terms of accuracy, efficiency, and generalization ability in order to filter out the most promising candidate models.

For the input image data, the necessary preprocessing is first performed, including size normalization, gray scaling, and noise removal. The feature extraction strategy varies depending on the type of algorithm: traditional machine learning methods use dimensionality reduction techniques such as Principal Component Analysis (PCA) to extract and optimize key features, thus improving the separability of the data, while deep learning methods automatically extract complex and discriminative features directly from the image via CNN, eliminating the need to manually design features.

In the model training phase, the candidate models are trained using the training set, while the model performance is optimized by adjusting the hyperparameters such as learning rate, regularization strength and network depth. In addition, this study employs data enhancement techniques, such as rotation, scaling and flipping, to extend the dataset to enhance the robustness and generalization of the model, which is particularly important for the training of deep learning models.

In the validation phase, the performance of the model is evaluated through the validation set, and key metrics such as accuracy, recall, and F1 score are calculated to measure the predictive ability of the model. For underperforming models, further improvements will be made by adjusting the network architecture, optimizing the training strategy, or introducing

regularization techniques. For CNN models, techniques such as network pruning and weight quantization will also be applied to improve the computational efficiency of the models and ensure that they can operate efficiently even in resource-limited environments.

Finally, based on the combined results of testing and validation, the key performance indicators of each model are compared and the model with the best performance is selected. In this way, it is ensured that the final selected model not only performs well in the experimental environment, but also meets the requirements of practical application scenarios.

3.2.5 Evaluation

In the evaluation phase, in-depth analysis and validation around the actual performance and business requirements of the model are required. First, the core objective of the model evaluation should be clarified, i.e., to ensure that the system meets the business requirements of time and attendance in terms of accuracy, efficiency, and applicability of face recognition, especially under the conditions of complex scenarios (e.g., changing light, occlusion, and multi-people scenarios), and can provide stable performance. Second, the model's performance on standard test data and interference data is quantified through the core metrics of the classification task, including Accuracy, F1-Score and ROC curve. These metrics can help identify model strengths and potential room for improvement.

Validation of real-world application scenarios is also one of the key tasks in the evaluation phase. By simulating the complexities of real-world environments with changing light, partial occlusion, and multiple people in the frame at the same time, the robustness and adaptability of the model can be tested. In addition, the response speed of the model needs to be evaluated on mobile devices to ensure the real-time performance of the recognition process, especially on low-end and mid-range devices.

Comparative analysis of models is another important aspect. The improved advantages of the new system can be verified by comparing the performance and resource consumption of the current model with other candidate models, and comparing the efficiency and reliability with traditional manual attendance methods. In addition, error analysis is used to locate the defects of the model under specific conditions, such as insufficient recognition of specific angles or expressions, and targeted optimization suggestions are made, such as performance enhancement through data augmentation or model architecture optimization.

Finally, all evaluation results and user feedback should be integrated into the final evaluation report to clarify whether the model meets the conditions for deployment. If the model meets the business requirements, a deployment plan can be formulated; if there are still deficiencies, the optimization direction, improvement strategy and timeline need to be clarified to ensure that the system can be successfully applied to actual attendance scenarios.

Confusion Matrix

In the evaluation phase, the performance of the model needs to be comprehensively evaluated by the core metrics of the classification task based on the confusion matrix. Confusion matrix is a tool used in data science and machine learning to summarize the prediction results of a classification model, which provides a comparison of the model's prediction results on test data with the actual labels, and it demonstrates the prediction performance of a classification model through an n × n matrix. The structure of the confusion matrix is shown in Figure 3.2 below:

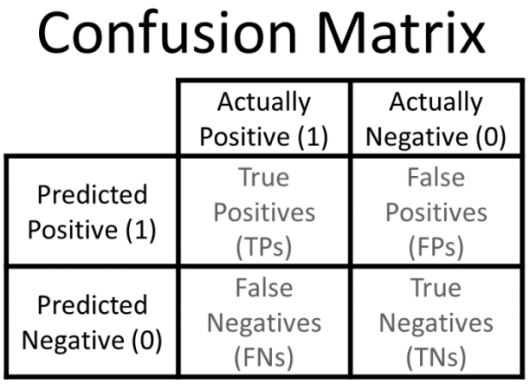


Figure 3.2 : The Confusion Matrix

Among them:

- TP (True Positive) : the number of samples that are actually positive classes and are predicted to be positive

- FP (False Positive) : the number of samples that are actually in the negative category but are predicted to be in the positive category
- FN (False Negative) : the number of samples that are actually positive but are predicted to be negative
- TN (True Negative) : the number of samples that are actually in the negative category and are predicted to be in the negative category

Also:

- TP + FP = P' : the number of all samples predicted as positive classes
- FN + TN = N' : the number of all samples that were predicted to be negative classes
- TP + FN = P : the number of samples that are actually positive
- FP + TN = N : the number of samples that are actually in the negative category

Evaluation Metrics for Classification Task

The evaluation metrics based on the confusion matrix will be used to analyse the performance of the model on the test dataset and to quantify its applicability in real time and attendance scenarios. Table 3.1 below lists the key metrics calculated based on the confusion matrix and explains each evaluation metric in detail.

Metric	Description	Equation
Accuracy	Percentage of correctly categorized samples out of the total number of samples	$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$

Precision	Proportion of samples predicted to be positive by the model that are actually also positive as a percentage of those predicted to be positive	$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$
Recall	Proportion of actual positive samples that were predicted to be positive as a percentage of actual positive samples	$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$
F1-Score	Evaluation metrics for balancing accuracy and recall. F1-Score measure the model's performance in balancing between positive and negative classes	$F1\ Score = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$
Receiver Operator Characteristic (ROC) curve	A two-dimensional graph used to represent the classification performance of a classifier between positive and negative samples	$True\ Positive\ Rate(TPR) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$ $False\ Positive\ Rate(FPR) = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$

Table 3.1: The Evaluation Metrics

3.2.6 Deployment

In the deployment phase of the AI-driven face recognition attendance application, trained models will be integrated into the back-end system to automate the attendance management process. Before deployment, a detailed plan will be developed, including the construction of servers, the development of API interfaces, and ensuring efficient integration between the back-end and the Android app to guarantee the stability and security of data transmission.

After the system goes live, students can quickly sign in through the mobile application, while professors can view attendance data and perform management operations in real time through an intuitive management interface. In order to enhance the effectiveness of the system, the system will regularly analyse the attendance data and generate detailed statistical reports. These reports not only help administrators monitor student attendance patterns and identify potential anomalies, but also provide data support for schools to optimize their attendance policies and processes.

Through this data-driven iterative approach, the system is able to continuously optimize the performance of the model and improve the accuracy and reliability of the recognition. At the same time, the system will always be designed with user needs in mind, providing a stable, efficient and easy-to-use attendance management solution that meets the actual needs of educational institutions.

3.3 Mobile Application Design

This section discusses Context Diagram (CD), Data Flow Diagram (DFD), Entity Relationship Diagram (ERD) and Data Dictionary related to the Mobile Attendance application. In addition, this section demonstrates a prototype user interface design for the Mobile Attendance application.

3.3.1 Context Diagram

The context diagram illustrates the interaction between the system and external entities involving two main entities: students and professors/professors. The context diagram of the system is shown in Figure 3.3, which specifies the interaction pattern between the Face Recognition Attendance App and external entities.

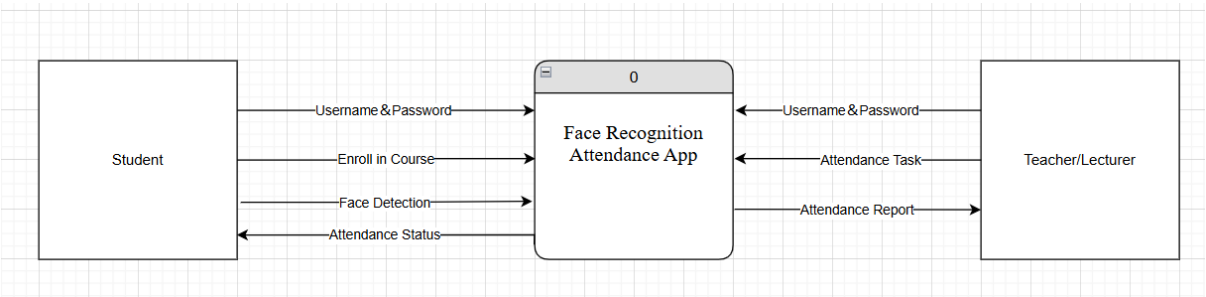


Figure 3.3 : Context Diagram of Face Recognition Attendance App

3.3.2 Level 1 Data Flow Diagram

The Level 1 Data Flow Diagram (DFD) is an extension of the context diagram that further refines and presents the internal data flow and operational details of the proposed system. Figure 3.4 illustrates the Level 1 Data Flow Diagram of the Face Recognition Attendance App, which further breaks down and complements the functional modules and data interactions of the system.

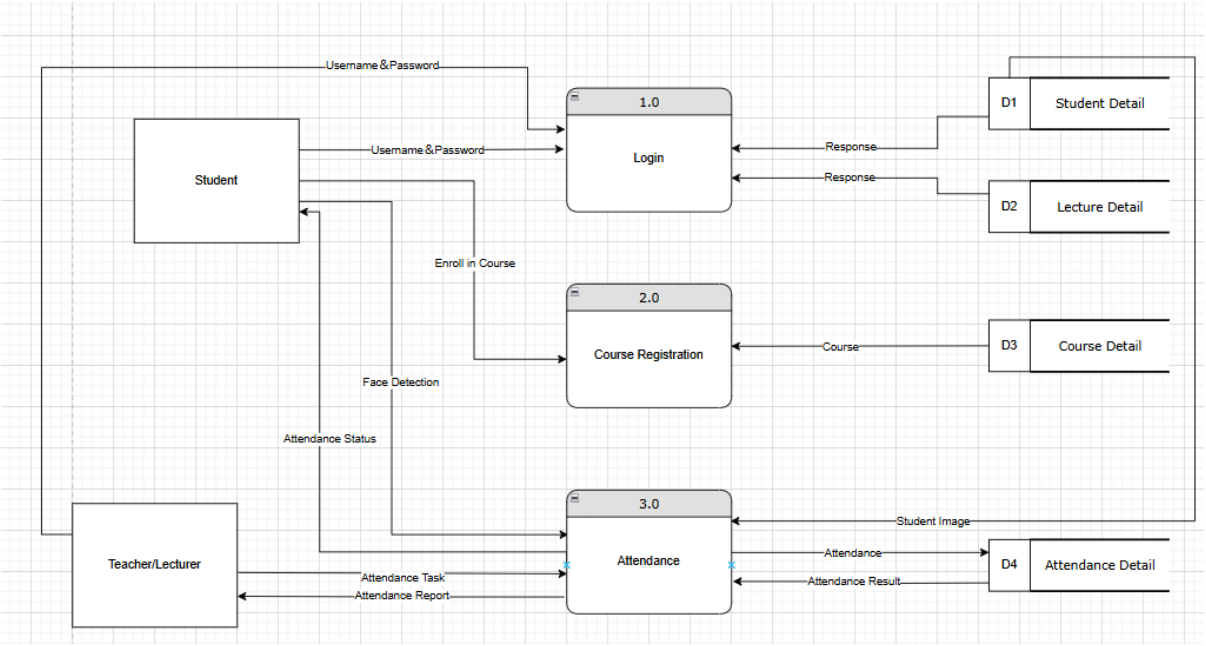


Figure 3.4 : Level 1 DFD of Face Recognition Attendance App

3.3.3 Entity Relationship Diagram

Entity Relationship Diagram (ERD) is used to show entities, attributes and their relationships with each other, and is an important visualization tool before database design and implementation. Figure 3.5 illustrates the Logical Entity Relationship Diagram of the Face Recognition Attendance App, which clearly reflects the core structure of the database. The database design of the proposed system consists of four main tables for storing key data such as user information, attendance records, etc., which provides a reliable data base for the efficient operation of the system.

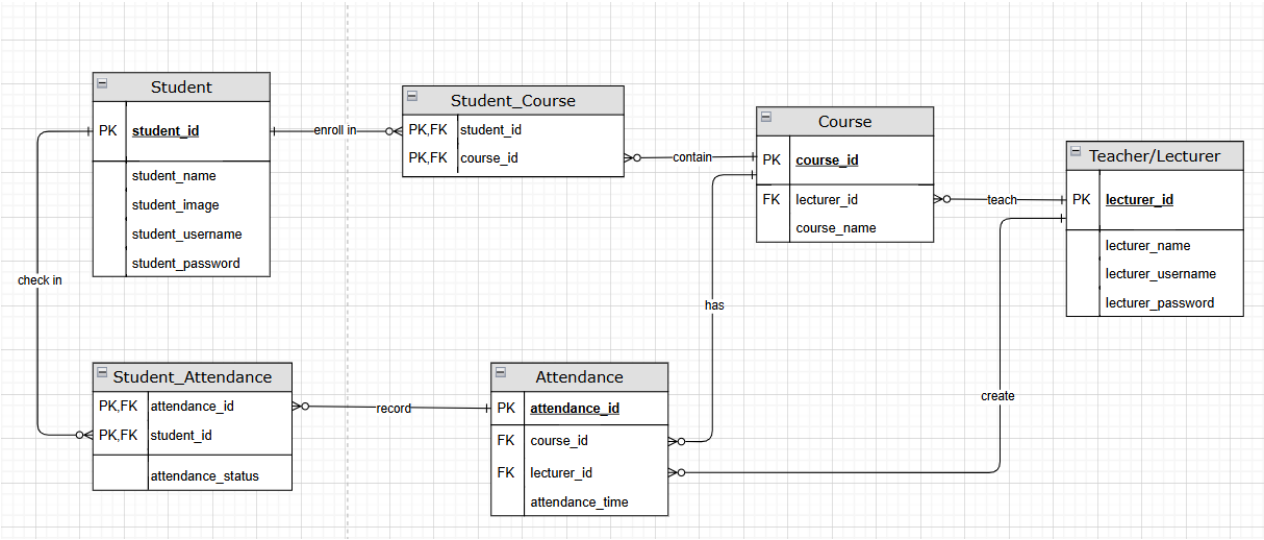


Figure 3.5 : Logical ERD of Face Recognition Attendance App

3.3.4 Data Dictionary

A data dictionary is a tool used to document the structure of a database table, which stores the definitions of data elements and their relationships. A data dictionary includes field names, data types, constraints, and field descriptions. The following tables show the contents of the data dictionary used by the Face Recognition Time and Attendance application.

Field	Data Type	Constraints	Description
Student_id	Integer (10)	Primary Key	Student ID
Student_name	Varchar (255)	Not Null	Full name of Student
Student_image	Text	Not Null	Face Image of Student
Student_username	Integer (10)	Not Null	Account username (StudentID of Student)
Student_password	Integer (12)	Not Null	Account password (IC No. of Student)

Table 3.2: Student Table

Field	Data Type	Constraints	Description
Student_id	Integer (10)	Primary Key, Foreign key	Student ID
Course_id	Varchar (8)	Primary Key, Foreign key	Course ID

Table 3.3: Student_Course Table

Field	Data Type	Constraints	Description
Course_id	Varchar (8)	Primary Key	Course ID
Professor_id	Integer (10)	Foreign key	Professor ID
Course_name	Varchar (255)	Not Null	Full name of Course

Table 3.4: Course Table

Field	Data Type	Constraints	Description
Professor_id	Integer (10)	Primary key	Professor ID
Professor _name	Varchar (255)	Not Null	Full name of Professor
Professor _username	Integer (10)	Not Null	Account username (ProfessorID of Professor)
Professor _password	Integer (12)	Not Null	Account password (IC No. of Professor)

Table 3.5: Professor Table

Field	Data Type	Constraints	Description
Attendance_id	Integer	Primary Key	Attendance ID

Course_id	Varchar (8)	Foreign key	Course ID
Professor_id	Integer (10)	Foreign key	Professor ID
Attendance_time	Boolean	Not Null	Time allowed for attendance

Table 3.6: Attendance Table

Field	Data Type	Constraints	Description
Attendance_id	Integer	Primary Key, Foreign key	Attendance ID
Student_id	Integer (10)	Primary Key, Foreign key	Student ID
Attendance_status	Boolean	Not Null	Attendance Status of Student

Table 3.7: Student_Attendance Table

3.3.5 Prototype

This section shows the prototype design to provide early visualization of the Attendance Android application. The key pages in the proposed system are drawn in the following figures.

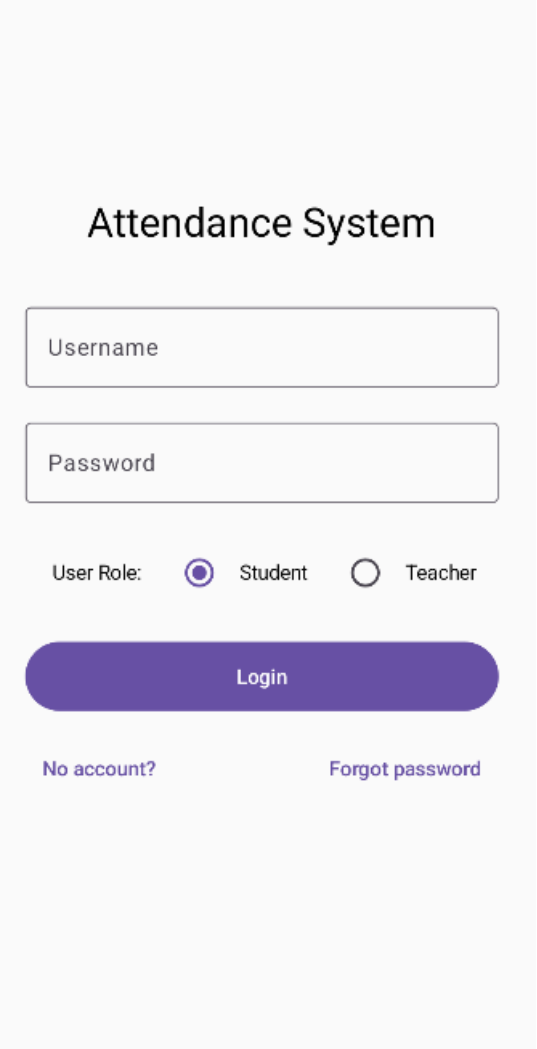


Figure 3.6 : Login Page of Face Recognition Attendance App

Figure 3.6 shows the login page of the Attendance Android application, where the user is required to select an identity and enter a name and password to complete the login operation.

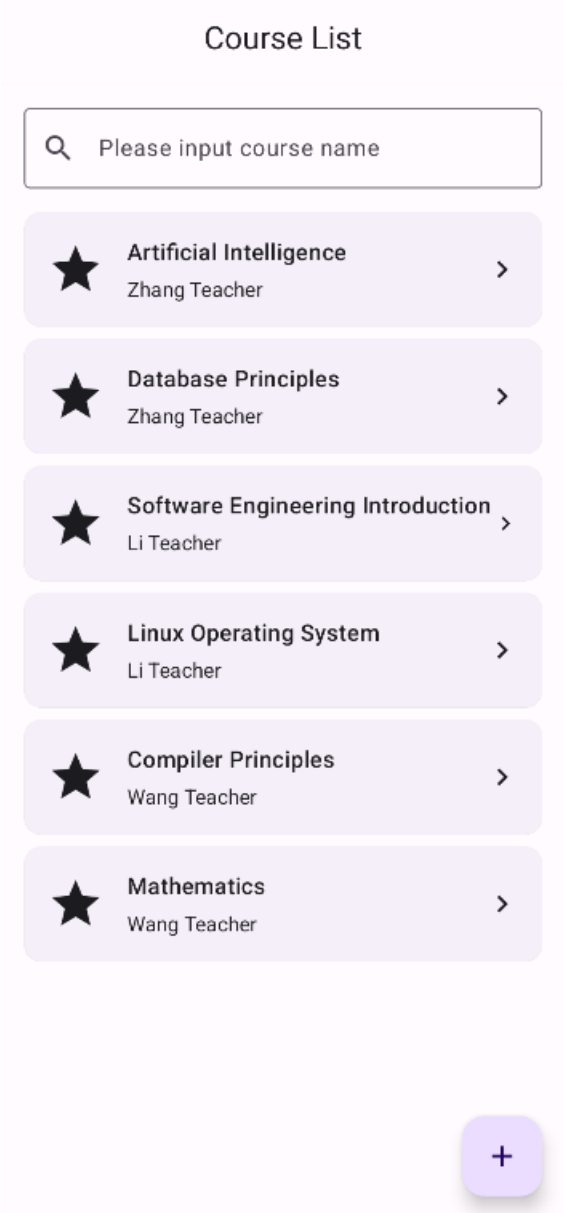


Figure 3.7 : Main Page of Face Recognition Attendance App

Figure 3.7 shows the main page of the student view in the Attendance Android application. The page lists the courses in which the student is enrolled, and the student can view the course details and access the attendance screen, or enrol in a new course.

Figure 3.8 : Attendance Page of Face Recognition Attendance App

Figure 3.8 shows the Attendance page of the Attendance Android application. This page displays the details of the attendance tasks for the selected course, where the user can view the attendance tasks to be completed and sign in.

3.4 Summary

This chapter provides a comprehensive overview of the application of the CRISP-DM methodology to the development of an AI-driven face recognition time and attendance system. The methodology is divided into six phases - Business Understanding, Data Understanding, Data Preparation, Modelling, Evaluation, and Deployment - ensuring a systematic and efficient development process.

In the business understanding phase, the goals of automating attendance and improving security were clarified to align with stakeholder requirements and organizational goals. In addition, legal and ethical considerations, especially regarding biometric data privacy, were carefully considered to ensure compliance with relevant standards. The data understanding phase focused on analysing the Labelled Face in the Wild (LFW) dataset to assess its suitability for face recognition tasks. Challenges such as data noise and inconsistency were identified and preprocessing steps such as data cleaning and enhancement were implemented to prepare the dataset for subsequent phases.

In the data preparation phase, efforts were made to standardize and optimize the dataset using techniques such as data augmentation, balancing, and feature extraction to improve model performance. In the modelling phase, various algorithms including machine learning and deep learning models such as Support Vector Machine (SVM), Random Forest (RF), K Nearest Neighbors (KNN) and Convolutional Neural Networks (CNN) were evaluated. The models were trained, validated and compared based on performance metrics, and the most appropriate model was finally selected for the system. The evaluation phase then rigorously tested the selected model through real-world simulations and performance metrics to ensure its robustness, accuracy, and efficiency in a dynamic environment. The final evaluation confirmed the utility and feasibility of deploying the system. Finally, during the deployment phase, the

trained models were integrated into the backend system to facilitate seamless interaction with the mobile application.

This chapter also discusses the design of the mobile application, detailing its architecture and functionality. Context diagrams illustrate the interactions between the application and external entities, focusing primarily on students and professors. A Level 1 Data Flow Diagram (DFD) builds on the contextual diagrams to detail the internal data flows and operational processes of the proposed system. The Entity Relationship Diagram (ERD) provides a clear visualization of the database structure, including the core entities such as students, courses, professors, and attendance, as well as their attributes and relationships. The data dictionary provides a detailed record of the database schema, specifying field names, data types, constraints, and descriptions to ensure consistency and standardization. In addition, prototyping provides early visualization of the user interface, showing key pages such as the login page, home page, and attendance page, thus demonstrating the system's functional flow and user interactions.

In summary, this structured approach to system development ensures that the final product is scalable, reliable, and well suited to meet user needs while conforming to industry best practices and standards.

Chapter 4: Model Implementation and Experimental Result

4.1 Introduction

This chapter describes the complete experimental flow of face recognition models for intelligent time and attendance systems, covering model construction and performance evaluation. The research goal is to explore robust and scalable solutions applicable to face recognition challenges in real-world environments.

Firstly, this chapter constructs and evaluates traditional machine learning methods (e.g. SVM+PCA/LDA, Random Forest, KNN) and customized CNN classifiers as typical paradigms in face recognition. Comparative experiments show that CNN performs optimally in terms of classification accuracy, but it is also pointed out that it is difficult to cope with open-set scenarios of dynamic user registration, just like traditional models.

In order to meet the system requirements for open-set recognition, this study designs a deep metric learning network based on the CNN architecture, which optimizes the embedding space by using the triple loss to make similar face features closer and different face features more separated, so as to solve the problem of recognizing individuals not included in the training set.

This chapter validates the potential of the metric learning network in terms of scalability and generalization ability by comparing the performance of each model under their respective applicable evaluation strategies. It also analyses the challenges of the model in terms of embedding spatial discriminability, in particular the problem of overlapping Euclidean distance distributions and its impact on the actual recognition results. In addition, this chapter briefly describes the experimental setup, the LFW dataset used and the software environment to provide a technical basis for subsequent model training and evaluation.

4.2 Experimental Environment Configuration

This section describes in detail the hardware and software environments for model development and experimentation, analyzes hardware performance bottlenecks and proposes optimization strategies to ensure efficient and reproducible experiments. The development environment supports the training and evaluation of traditional machine learning, CNN classifiers and FaceNet style models, and provides technical preparations for the subsequent mobile deployment of the Android attendance application.

4.2.1 Hardware Specifications

Model training tasks in this study were primarily conducted on a computing device specifically configured for deep learning workloads. To significantly accelerate the deep learning model training process, this research fully leveraged the parallel computing capabilities of NVIDIA GPUs, specifically utilizing an NVIDIA GeForce GTX 1650 Ti Max-Q Design GPU equipped with 4GB of GDDR6 video memory. The integration of this GPU proved crucial for substantially reducing training times and its high-efficiency computing power was particularly vital when processing complex convolutional neural network architectures, such as the FaceNet-style model. Furthermore, model training also imposed distinct requirements on Central Processing Unit (CPU) performance and system memory capacity. The development host for this research was outfitted with an Intel Core i5-1135G7 processor and 16GB of DDR5 memory (RAM), a configuration that collectively provided ample and stable computational resources for data loading, preprocessing, and the iterative training of the models.

In order to clearly outline the hardware platform used in the model training phase of this research, the main configurations are summarized in Figure 4.1.

Category	Sub-category/Component	Specific Configuration/Description
Development Host	Operating System	Windows 11 Professional
	Processor (CPU)	Intel Core i5-1135G7
	Memory (RAM)	16GB DDR4
	Graphics Processor (GPU)	NVIDIA GeForce GTX 1650 Ti Max-Q Design
	Video Memory (VRAM)	4GB GDDR6

Figure 4.1 : Model Training Hardware Environment Summary

4.2.2 Software Environment

The software environment configuration utilized in this research for model training was meticulously selected and set up, with its specific details outlined below. Regarding the operating system, this study employed the Windows 11 Professional operating system. As the core programming language for model training, Python 3.11.5 was the ideal choice for constructing and training complex facial recognition models, given its robust capabilities and extensive ecosystem in scientific computing and deep learning.

Concerning deep learning frameworks and libraries, this project adopted TensorFlow 2.10.0 as the primary deep learning framework, concurrently leveraging Keras as its high-level Application Programming Interface (API), which considerably streamlined the neural network construction and training workflows. To fully harness the computational potential of the NVIDIA GPU, this research also installed and configured NVIDIA CUDA Toolkit 11.2 and cuDNN 8.1; these underlying libraries provided essential GPU-accelerated runtime support, playing a decisive role in ensuring the efficient operation of TensorFlow, particularly in accelerating the training process of the FaceNet-style model. Additionally, the Scikit-learn library was utilized for implementing various traditional machine learning models, including Principal Component Analysis (PCA), Support Vector Machines (SVM), Random Forests, K-

Nearest Neighbors (KNN), and Linear Discriminant Analysis (LDA), and for computing relevant classification performance metrics. NumPy, as Python's foundational scientific computing library, provided the necessary support for efficient array and matrix operations, while Matplotlib and Seaborn served as professional data visualization tools, widely applied in plotting training curves, confusion matrices, and distance distribution graphs. The Joblib library was employed for conveniently saving and loading the trained traditional machine learning models.

In the selection of Integrated Development Environments (IDEs), this research primarily leveraged Jupyter Notebook and Visual Studio Code as interactive development platforms, which collectively offered an efficient and user-friendly environment for Python code authorship, experimental validation of models, and result visualization. To maintain the purity and stability of the development environment, this study also utilized the Anaconda tool to create and manage an independent Python virtual environment named lfw-env, version 23.1.0. This virtual environment management approach effectively isolated project dependencies, thereby mitigating potential conflicts arising from differing library versions across projects and ensuring a clean and stable development environment. Finally, Git, as the central version control tool, was employed to manage the iterative versions of the model training code, significantly facilitating the tracking of modification history and fostering collaborative efforts.

To clearly outline the software tool stack used in the model training phase of this study, the main configurations are summarized in Figure 4.2.

Category	Component/Library/Tool	Specific Version/Description
Operating System	Windows	Windows 11 Professional
Core Programming Language	Python	3.11.5
Deep Learning Frameworks & Libraries	TensorFlow	2.10.0
	Keras	(As a high-level API of TensorFlow)
	NVIDIA CUDA Toolkit	11.2
	cuDNN	8.1
Traditional Machine Learning Library	Scikit-learn	Used for PCA, SVM, RF, KNN, LDA
Scientific Computing & Data Processing	NumPy	Core scientific computing
Data Visualization	Matplotlib	
	Seaborn	
Model Serialization	Joblib	Used for saving/loading traditional models
Development Environment (IDE)	Jupyter Notebook	
	Visual Studio Code	
Environment Management	Anaconda	(Virtual environment name: lfw-env, version 23.1.0)
Version Control	Git	(Integrated with GitHub)

Figure 4.2 : Model Training Software Environment Summary

```
# Cell 1: Import necessary libraries
import joblib
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split
from sklearn.metrics import (
    classification_report, accuracy_score, precision_score,
    recall_score, f1_score, confusion_matrix, roc_curve, auc
)
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import label_binarize

import tensorflow as tf
from tensorflow.keras.models import Model  # Import Model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import (
    Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Input, Lambda, BatchNormalization
)
from tensorflow.keras import backend as K
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator  # For image preprocessing

sns.set(style="whitegrid")
print("Cell 1 completed")
```

Figure 4.3 : Importing Required Libraries for Face Recognition Model and Evaluation

4.3 LFW Dataset Preprocessing

This section details the refined preprocessing process of the Labeled Faces in the Wild (LFW) dataset before it is loaded and fed into various types of face recognition models to ensure that the data format accurately matches the specific input requirements of different model types and to optimize the performance of model training and evaluation.

The LFW dataset is a widely used public dataset in the field of face recognition, which is characterized by a large number of real-world face images in uncontrolled environments, covering a wide variety of poses, lighting, expressions, and backgrounds, which poses a challenge to the model generalization capability.

4.3.1 Dataset Loading and Initial Screening

In this study, the LFW dataset was loaded by sklearn.datasets.fetch_lfw_people function. To ensure that each identity category has enough training samples, the parameter min_faces_per_person=70 is set to strictly filter out people containing at least 70 photos. This contributes to the adequacy of subsequent model training and evaluation. In addition, the resize=0.4 parameter is applied during initial loading to uniformly scale the original images to 50×37 pixels, which effectively reduces the computational complexity of the subsequent processing. After this loading and initial screening, the original grayscale image size of the dataset is denoted as (n_samples, 50, 37), where n_samples denotes the total number of images, and the dataset contains n_classes of different character categories. It should be noted that the fetch_lfw_people function already provides pre-cropped and roughly aligned face images, so there is no need for additional independent face detection and alignment steps.

The following figure shows a visual flowchart of data loading and initial filtering.

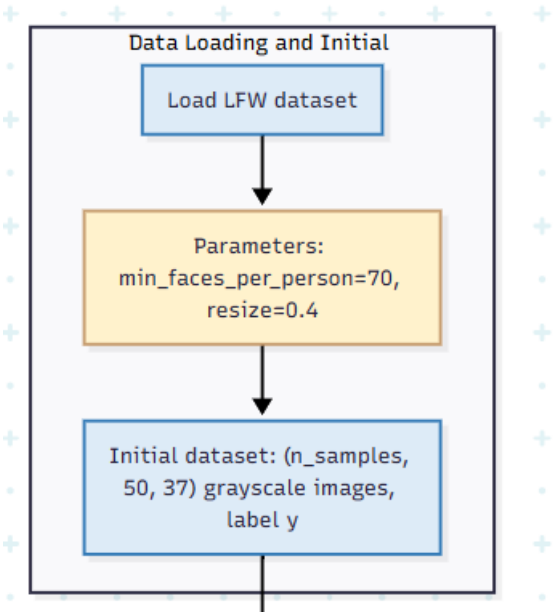


Figure 4.4 : Structured presentation of data loading and initial screening

4.3.2 Preprocessing for Machine Learning Models

For machine learning models based on Scikit-learn, including PCA+SVM, Random Forest, KNN, and LDA+SVM, the image data needs to be converted into one-dimensional feature vectors. Therefore, the original grayscale image is first reshaped into a two-dimensional matrix of (n_samples, 50 × 37). Subsequently, principal component analysis (PCA) is applied for dimensionality reduction; by setting the number of principal components to n_components = 150 and performing whitening (whiten), the correlation between features is effectively eliminated and their variance is normalized. This series of preprocessing steps significantly improves the performance of the subsequent classifier.

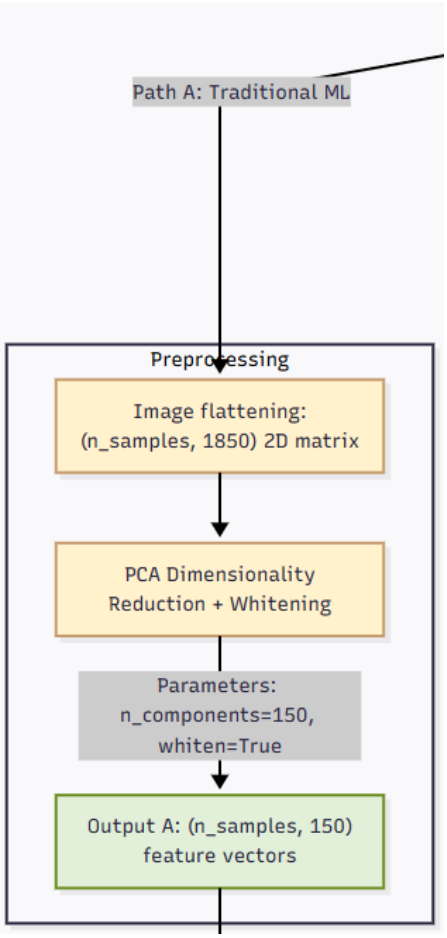


Figure 4.5 : Structured presentation of Preprocessing for Machine Learning Models

4.3.3 Preprocessing for CNN Classification Models

CNN classification models require that the input image retains its intrinsic two-dimensional spatial structure and requires explicit channel dimensionality. To this end, the original grayscale image is carefully reshaped into a four-dimensional tensor of (n_samples, 50, 37, 1), where the fourth dimension explicitly indicates the single-channel (i.e., grayscale map) nature of the image. At the same time, the corresponding label y is precisely converted to One-Hot coding format to ensure compatibility with the requirements of the output layer of the classification model.

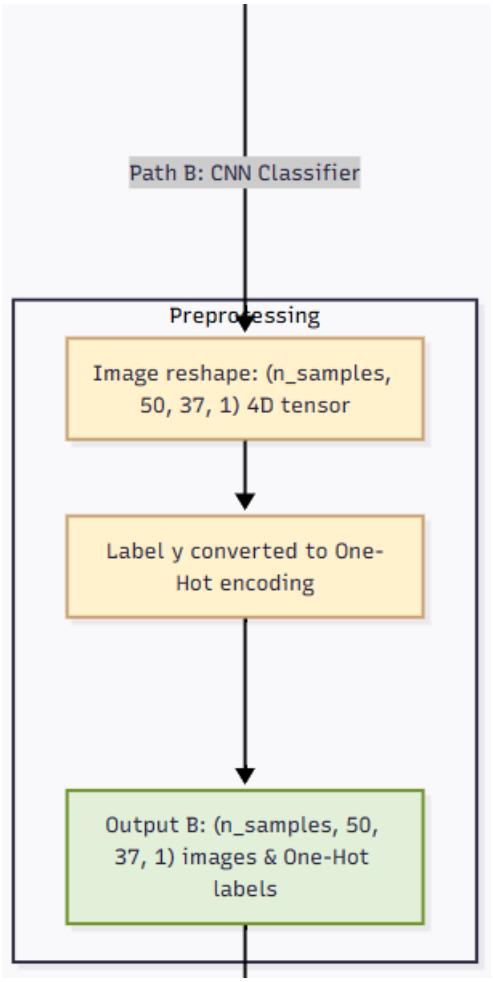


Figure 4.6 : Structured presentation of Preprocessing for CNN Classification Models

4.3.4 Preprocessing for Metric Learning Network

The Metric Learning Network has specific requirements on the size and number of channels of the input image, usually expecting a color image and a large size. Therefore, the original 50×37-pixel grayscale LFW images undergo the following key transformations:

- Channel Expansion: The original 50x37 single-channel grayscale image is first expanded to a simulated RGB three-channel image with dimensions (n_samples, 50, 37, 3) by copying its data three times in the added channel dimension via NumPy's np.repeat function. This conversion is critical because most pre-trained

deep learning models (including FaceNet style models) are typically trained on color image datasets.

- Uniform size: The extended channel images are then uniformly resized to 160x160 pixels using the tf.image.resize function. tf.image.resize is chosen for this scaling to ensure that the image reaches an accurate and uniform input size (160, 160, 3) before entering the FaceNet model, which is critical for model stability and performance.
- Pixel value normalization: Finally, to optimize model convergence performance, the image pixel values are normalized from the original [0, 255] range to the [0, 1] range, which is standard practice in deep learning model training.

After the above transformation, the input image tensor size of the Metric Learning Network becomes (n_samples, 160, 160, 3).

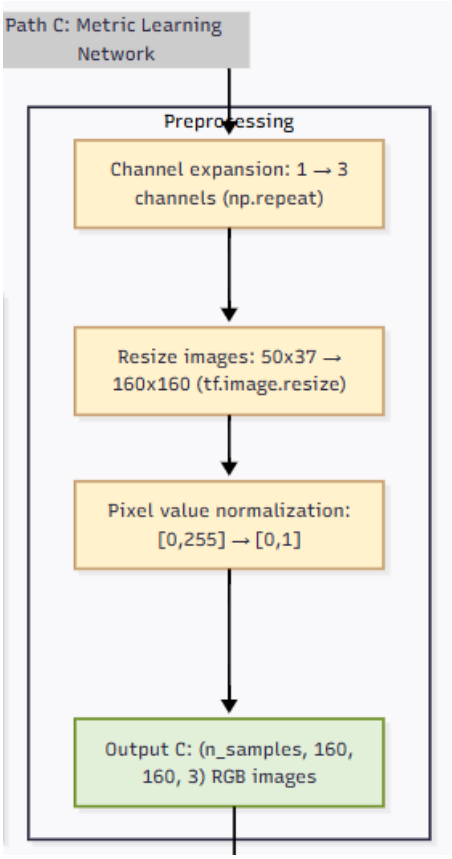


Figure 4.7 : Structured presentation of Preprocessing for Metric Learning Network

4.3.5 Dataset Partitioning

After completing all data preprocessing, the dataset was divided into training, validation, and testing sets. A consistent dataset partitioning strategy was used for all models. Specifically, this study first used the sklearn.model_selection.train_test_split function to divide the data into the main training set (75%) and the final test set (25%). For model that requires an independent validation set during training (CNN), 10% of this master training set is further delineated as the validation set. Thus, the original dataset is effectively divided into approximately 67.5% for training, 7.5% for validation, and 25% for testing. The stratify=y parameter is applied in all the partitioning steps in this study to ensure the consistency of the distribution of the categories in the different subsets, so as to avoid data skewing that may affect the model evaluation.

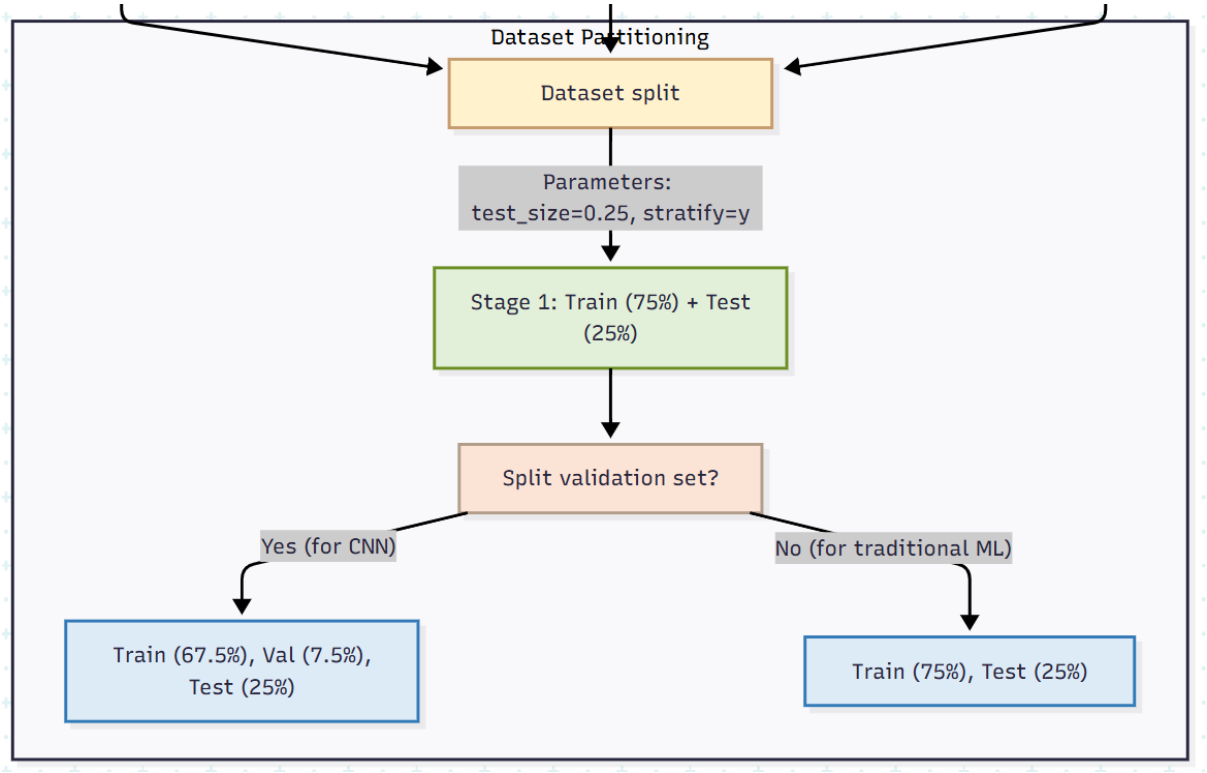


Figure 4.8 : Structured presentation of Dataset Partitioning

After the final division, the number of samples and the shape of each dataset are shown in Figure 4.9.

Dataset Type	Sample Size (N)	Original Grayscale Image Shape (H, W)	FaceNet Style Input Image Shape (H, W, C)	Remarks
Total Dataset	1288	(50, 37)	(160, 160, 3)	
Training Set	966	(966, 50, 37)	(966, 160, 160, 3)	75% of total dataset
Test Set	322	(322, 50, 37)	(322, 160, 160, 3)	25% of total dataset
Validation Set	~97	(~97, 50, 37)	(~97, 160, 160, 3)	10% dynamically split from training set (966 * 0.1 ≈ 97)

Figure 4.9 : Details of Dataset Segmentation

4.4 Face Classification Models: Implementation and Comparative Evaluation

The aim of this section is to detail the implementation, training and evaluation process of traditional machine learning classifiers and convolutional neural network (CNN) classifiers. These models are considered as key performance benchmarks for in-depth analysis of their face recognition capabilities under the closed set classification paradigm. This evaluation phase highlight their effectiveness in recognizing known individuals, and in doing so, demonstrate their fundamental limitations when faced with the open-set challenges inherent in time and attendance systems.

4.4.1 Machine Learning Models Implementation

The aim of this section is to implement and evaluate a series of traditional machine learning classifiers to establish a performance baseline for face classification and to provide a basis for performance comparisons of subsequent deep learning methods.

As described in Section 4.3.2, the original image data (50 × 37 = 1850 pixels) is first flattened into a one-dimensional vector. Subsequently, we apply Principal Component Analysis (PCA) for dimensionality reduction to project the 1850-dimensional flattened image data to a lower-dimensional space of 150 dimensions. This process includes whitening (whiten=True), which not only effectively reduces the data redundancy by identifying the orthogonal components with the largest variance, but also ensures that the variance of each principal component is normalized to 1 and further enhances the decorrelation of the features, which significantly improves the processing efficiency and stability of the subsequent classifiers.

The figure below visualizes how the data (X_flat_train) is degraded and whitened by PCA to generate features (X_pca_train) for model training.

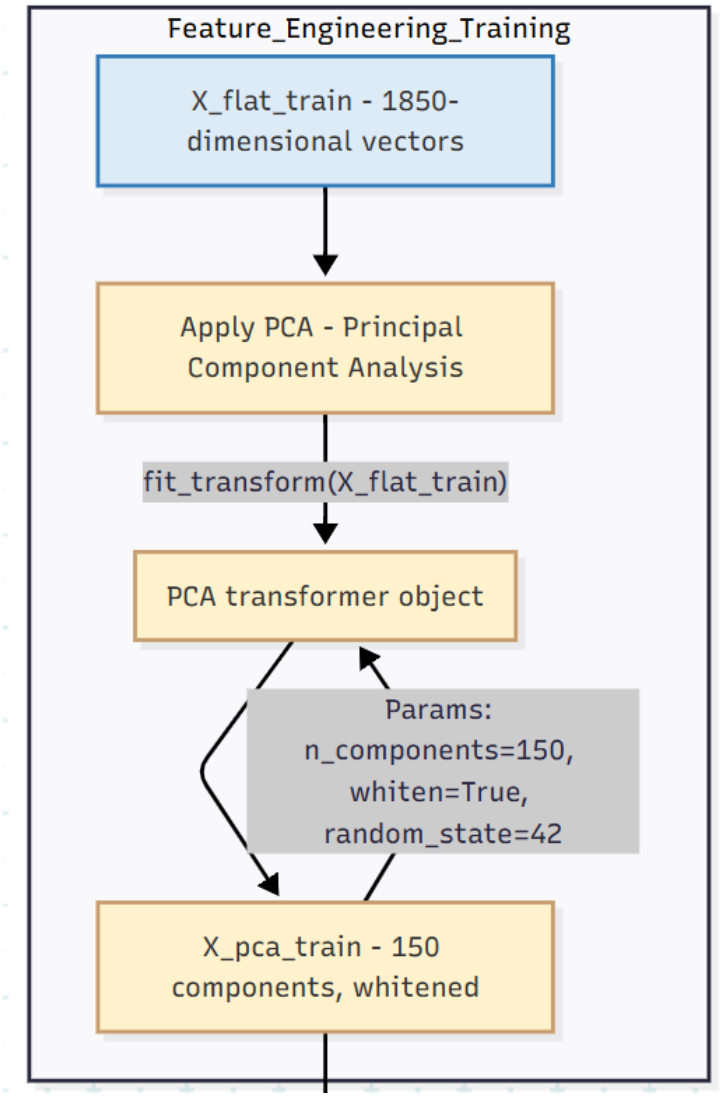


Figure 4.10 : Feature Engineering Stage in the Machine Learning Process

In the feature space after PCA dimensionality reduction, the following four classical classification algorithms are applied in this study for model training and evaluation:

1. Support Vector Machine (SVM): A Radial Basis Function (RBF) kernel is used and the potential class imbalance problem is effectively addressed by setting the class_weight='balanced' parameter. The model explicitly enables probability=True to obtain a probabilistic output, which is essential for subsequent ROC curve analysis.

2. Random Forest (RF): As an integrated learning method, RF aims to improve classification accuracy and generalization by constructing multiple decision trees and aggregating their predictions.
3. K-Nearest Neighbors (KNN) algorithm: As an instance-based non-parametric learning method, KNN determines the classification of a new sample by identifying the classes of the K nearest training samples.
4. Linear Discriminant Analysis (LDA) + SVM: This study also explores the combined application of LDA and SVM, which is a supervised dimensionality reduction technique whose goal is to find the optimal projection direction to maximize the category spacing while minimizing the scatter within the same category. In this implementation, LDA performs a secondary dimensionality reduction on the PCA-processed data, and its output is then used as a feature input to an SVM classifier using a linear kernel, which is also enabled with probability=True.

The following figure clearly shows the training process of four different types of traditional machine learning models (SVM, Random Forest, KNN, LDA+SVM), including their specific algorithms and key hyperparameters.

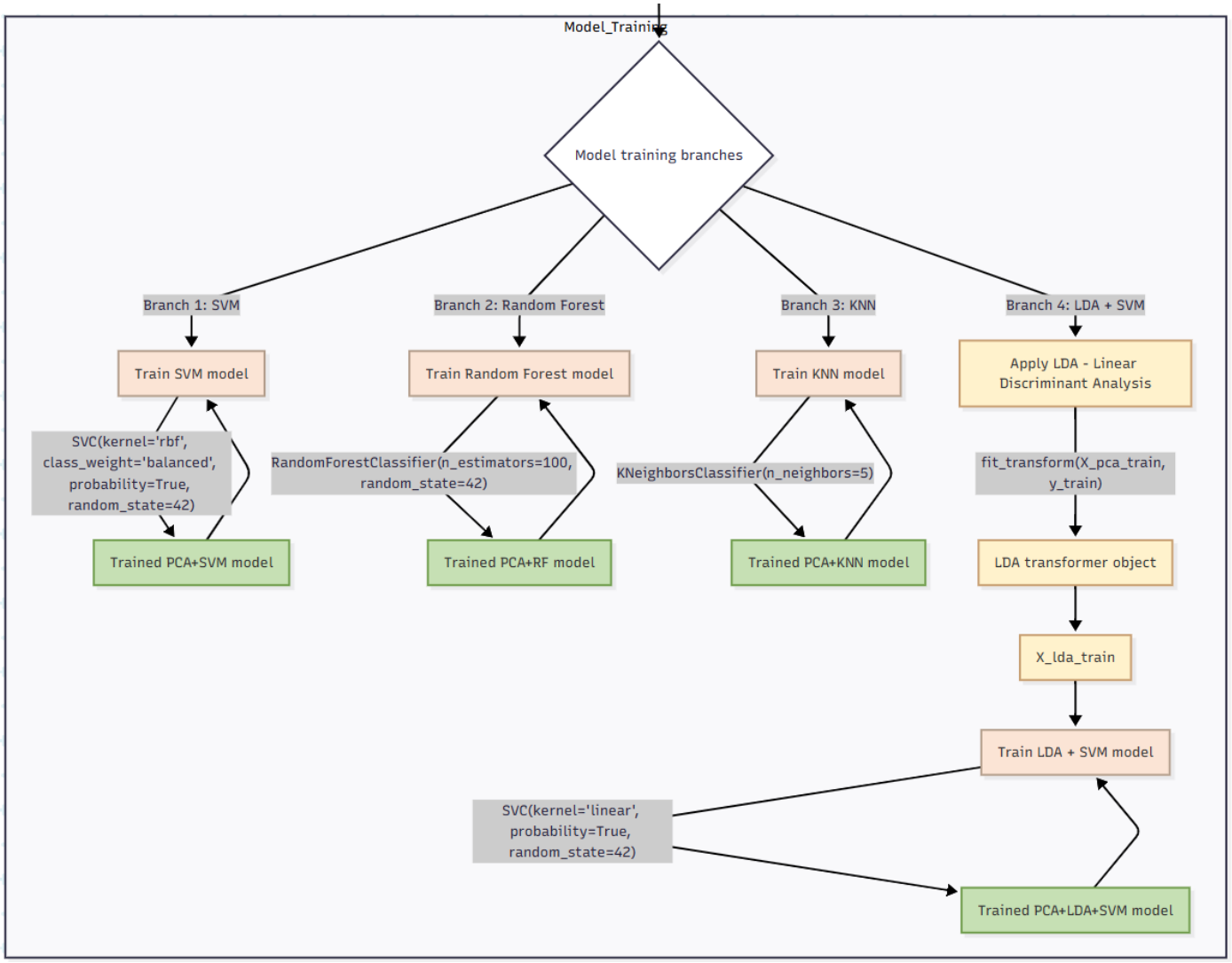


Figure 4.11 : Training of Machine Learning Models

The key hyperparameters of these classifiers, such as the number of decision trees in the random forest, the number of nearest neighbors for the KNN, and the kernel function and weight configurations for the SVM, have been set directly in the code and are designed to provide effective performance on this benchmark dataset.

Figure 4.12 details the hyperparameter configurations of each machine learning model used in this study and the basis for their selection.

Model Name	Hyperparameter	Value	Remarks (Parameter Meaning)	Selection Rationale (Why This Value/Configuration)
PCA	n_components	150	Number of features after dimensionality reduction	Empirically chosen to retain key LFW face image information (sufficient for distinguishing facial features) while reducing dimensionality to improve computational efficiency and mitigate overfitting risks due to the curse of dimensionality.
	whiten	True	Whiten the principal components	Ensures variance of each principal component is normalized to 1 and removes linear correlations between features, making feature contributions more balanced, which enhances performance and stability of subsequent classifiers (especially SVM and LDA).
	random_state	42	Random seed for reproducibility	Ensures consistent experimental results across multiple runs, improving research reliability and transparency for reproducibility.
PCA + SVM	kernel	'rbf'	Radial basis function kernel	RBF is a versatile nonlinear kernel capable of handling complex nonlinear relationships in the dataset, performing well in various image recognition tasks and often chosen as the default kernel.
	class_weight	'balanced'	Automatically adjust weights for class imbalance	The LFW dataset, after filtering (min_faces_per_person=70), may still have uneven sample sizes across classes. This parameter assigns higher weights to minority classes, addressing imbalance, preventing bias toward majority classes, and improving classification fairness and robustness.
	probability	True	Enable probability estimation	Allows obtaining classification probability outputs, essential for generating ROC curves and calculating AUC values, which provide a comprehensive evaluation of classifier performance.
	random_state	42	Random seed for reproducibility	Ensures reproducibility of the model training process (e.g., if internal algorithms involve randomness).
PCA + Random Forest	n_estimators	100	Number of decision trees in the forest	100 trees is a common choice in ensemble learning, effectively reducing model variance and improving generalization and stability without significantly increasing computational cost.
	random_state	42	Random seed for reproducibility	Ensures reproducibility of the random forest construction process (e.g., bootstrap sampling and feature selection at node splits).
PCA + KNN	n_neighbors	5	Number of neighbors for classification	5 is a commonly used empirical value for KNN. Smaller K values may be sensitive to noise, while larger K values may blur class boundaries; 5 often strikes a good balance, suitable for various datasets.
PCA + LDA + SVM	LDA	(no explicit hyperparameters)	Default parameters, performs dimensionality reduction	LinearDiscriminantAnalysis uses the SVD solver by default, which is generally robust. As a supervised dimensionality reduction method, it maximizes inter-class separability, complementing PCA to optimize the feature space and enhance classification performance.
	SVM kernel	'linear'	Linear kernel	After LDA, data in the projected space is more linearly separable. A linear kernel maintains classification performance while being computationally efficient and better reflecting LDA's linear discriminant properties.
	probability	True	Enable probability estimation	Same as PCA + SVM, used for ROC curve analysis.
	random_state	42	Random seed for reproducibility	Same as PCA + SVM, ensures reproducibility.

Figure 4.12 : Machine Learning Model Hyperparameter Configuration

Figure 4.13 visually illustrates the training process of these machine learning models based on PCA feature compression for face recognition.

```
# Flatten the image data into vectors for traditional ML models
X_flat_train = X_train_orig.reshape((X_train_orig.shape[0], -1))
X_flat_test = X_test_orig.reshape((X_test_orig.shape[0], -1))

# Apply PCA for dimensionality reduction
pca = PCA(n_components=150, whiten=True, random_state=42)
X_pca_train = pca.fit_transform(X_flat_train)
X_pca_test = pca.transform(X_flat_test)

# --- Model 1: PCA + SVM ---
svm = SVC(kernel='rbf', class_weight='balanced',
          probability=True, random_state=42)
svm.fit(X_pca_train, y_train)
y_pred_svm = svm.predict(X_pca_test)
y_proba_svm = svm.predict_proba(X_pca_test)
predictions_proba_traditional["PCA + SVM"] = y_proba_svm

# --- Model 2: PCA + Random Forest ---
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_pca_train, y_train)
y_pred_rf = rf.predict(X_pca_test)
y_proba_rf = rf.predict_proba(X_pca_test)
predictions_proba_traditional["PCA + Random Forest"] = y_proba_rf

# --- Model 3: PCA + KNN ---
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_pca_train, y_train)
y_pred_knn = knn.predict(X_pca_test)
y_proba_knn = knn.predict_proba(X_pca_test)
predictions_proba_traditional["PCA + KNN"] = y_proba_knn

# --- Model 4: PCA + LDA + SVM ---
lda = LinearDiscriminantAnalysis()
X_lda_train = lda.fit_transform(X_pca_train, y_train)
X_lda_test = lda.transform(X_pca_test)

svm_lda = SVC(kernel='linear', probability=True, random_state=42)
svm_lda.fit(X_lda_train, y_train)
y_pred_lda = svm_lda.predict(X_lda_test)
y_proba_lda = svm_lda.predict_proba(X_lda_test)
predictions_proba_traditional["PCA + LDA + SVM"] = y_proba_lda
```

Figure 4.13 : Training process of machine learning model based on PCA feature compression

4.4.2 Deep Learning Model (CNN Classifier) Implementation

Convolutional neural networks (CNNs) have become a cornerstone of deep learning given the limitations inherent in traditional machine learning approaches for processing high-dimensional, nonlinear image data, such as the need for complex manual feature engineering and the difficulty in capturing deep semantic information. Their intrinsic ability to automatically learn and extract hierarchical features from raw images makes them well suited for image classification tasks. Therefore, this study introduces a customized CNN model to be explored as a basis for the face classification task.

The custom CNN model employs a typical convolution-pooling architecture designed to efficiently extract features from preprocessed LFW grayscale images for classification. The input layer of the model is precisely configured to have a dimension of (50,37,1), which perfectly matches the LFW image dimensions determined during the preprocessing stage (see Section 4.3 for details). The network contains two main convolutional blocks:

1. First convolutional block: This block consists of a Conv2D layer using 32 3×3 convolutional kernels combined with a ReLU activation function. This is followed by a MaxPooling2D layer with a pooling size of 2×2, which is used to down sample the feature map and add translation invariance.
2. Second Convolutional Block: After the first block, another Conv2D layer is used, this time with 64 3×3 convolutional kernels, again using the ReLU activation function. A subsequent MaxPooling2D layer (with a pooling size of 2×2) further reduces the feature dimensions and enhances the extraction of salient spatial attributes.

After these convolutional layers, a Flatten layer converts the 2D feature map into a 1D vector. This vector is then fed into a fully connected (Dense) layer containing 128 neurons,

which also uses the ReLU activation function for higher level feature abstraction. In order to mitigate model overfitting, a Dropout layer with a dropout rate of 0.5 is strategically introduced after this fully connected layer. Finally, the output layer is structured as a Dense layer containing n_classes of neurons (where n_classes represents the total number of filtered people in the LFW dataset) and a SoftMax activation function is applied to output the predicted probability distribution for each class. The complete architecture of this baseline CNN classifier is visualized in Figure 4.14.

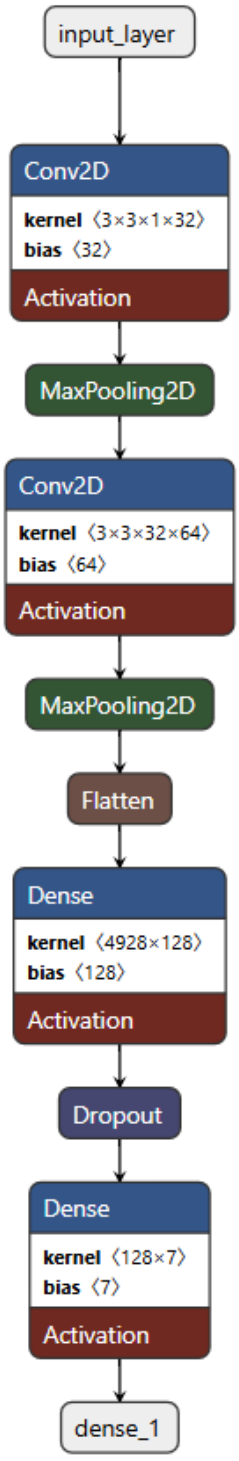


Figure 4.14 : Fundamental CNN classifier architecture with Model Architecture Hyperparameters

In order to clearly present the architectural design of the custom CNN classification model in this study, its main hyperparameters and the rationale for their selection are summarized in Figure 4.15.

Category	Hyperparameter	Value/Selection	Brief Explanation
Input Layer	Input Dimension	(50, 37, 1)	Matches the preprocessed LFW grayscale image dimensions.
Convolutional Block (Conv2D)	Number of Filters	32, 64	Increases layer-by-layer to capture complex features, commonly powers of 2.
	Filter Size	3x3	Popular choice, effectively captures local features with high computational efficiency.
	Activation Function	ReLU	Addresses vanishing gradient problem, improves computational efficiency.
Pooling Layer (MaxPooling2D)	Pool Size	2x2	Common downsampling strategy, reduces feature map size and enhances translation invariance.
Fully Connected Layer (Dense)	Number of Neurons	128	Empirical choice for high-level feature abstraction with moderate dimensionality.
	Activation Function	ReLU	Same as above.
Dropout Layer	Dropout Rate	0.5	Common rate, effectively mitigates overfitting.
Output Layer	Number of Neurons	n_classes	Matches the number of classes in the filtered LFW dataset.
	Activation Function	SoftMax	Standard activation function for outputting probability distributions in multi-class tasks.

Figure 4.15 : CNN Classifier Model Architecture Hyperparameters

In the compilation and training phases, the CNN model uses the Adam optimizer with an initial learning rate of 0.001. Adam is chosen because it has the advantage of adaptive learning rate tuning in deep learning tasks, can effectively deal with sparse gradient problems, and has been widely proved to perform well on a wide range of tasks. The model chooses categorical_crossentropy as the loss function, which is suitable for multiple classification tasks using One-Hot coded labels. Accuracy is specified as the main evaluation metric. The model was trained for 100 epochs on a prepared training set with a batch size of 32. The batch size of 32 is a common choice to strike a balance between computational resources and convergence efficiency. Although no explicit learning rate scheduling strategy was applied during the training process and a default constant learning rate was maintained, this setting has enabled the model to converge efficiently in preliminary experiments. In addition, 10% of the training data was partitioned into a validation set, which was used to rigorously monitor the model's performance on unseen data and to guide the training process, thus effectively aiding in the detection of overfitting.

```
# Cell 5: Define and train CNN Classifier

# CNN need input format [samples, height, width, channels]
X_cnn_train_orig = X_train_orig.reshape(-1,
                                        X_train_orig.shape[1], X_train_orig.shape[2], 1)
X_cnn_test_orig = X_test_orig.reshape(-1,
                                      X_test_orig.shape[1], X_test_orig.shape[2], 1)

# Convert label to one-hot encoded format
y_cnn_train_cat = to_categorical(y_train, num_classes=n_classes)
y_cnn_test_cat = to_categorical(y_test, num_classes=n_classes)

# Build CNN Model
cnn_model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(
        X_train_orig.shape[1], X_train_orig.shape[2], 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(n_classes, activation='softmax')
])
cnn_model.compile(optimizer='adam',
                  loss='categorical_crossentropy', metrics=['accuracy'])

# Train CNN Model
history_cnn_classifier = cnn_model.fit(
    X_cnn_train_orig, y_cnn_train_cat,
    epochs=100, batch_size=32,
    validation_split=0.1, verbose=1
)

y_proba_cnn_classifier = cnn_model.predict(X_cnn_test_orig)
predictions_proba_cnn_classifier["CNN Classifier"] = y_proba_cnn_classifier
y_pred_cnn_classifier = np.argmax(y_proba_cnn_classifier, axis=1)
predictions["CNN Classifier"] = y_pred_cnn_classifier
```

Figure 4.16 : CNN Classifier with Categorical Cross-Entropy Losses

The key hyperparameters used in the training process of the CNN classification model in this study and the reasons for their selection are summarized in Figure 4.17.

Category	Hyperparameter	Value/Selection	Brief Explanation
Optimizer	Type	Adam	Offers adaptive learning rate adjustment, widely proven to perform well across various tasks.
	Initial Learning Rate	0.001	Industry-standard default, enables effective model convergence in preliminary experiments.
Loss Function	Type	categorical_crossentropy	Suitable for multi-class tasks with one-hot encoded labels.
Evaluation Metric	Primary Metric	Accuracy	Direct metric for measuring model classification performance.
Training Epochs	Epochs	100	Preliminary observations show effective convergence within this number of epochs.
Batch Size	Batch Size	32	Common choice balancing computational resources and convergence efficiency.
Learning Rate Schedule	Strategy	None (Constant)	Preliminary experiments show constant learning rate enables effective convergence.
Data Split	Validation Split	10%	Used to strictly monitor model performance on unseen data, aiding in detecting overfitting.

Figure 4.17 : CNN Classifier Training Process Hyperparameters

4.4.3 Evaluation Methodology

The main goal of this evaluation phase was to quantify the performance of these models in a closed-set classification task to reflect their accurate ability to recognize individuals known at the time of training. The following standard metrics were used:

1. Accuracy: The percentage of correctly classified instances out of the total instances. It is a direct measure of the overall correctness of the model and is calculated as the number of correct predictions on the test set divided by the total number of predictions.
2. F1 Score: The reconciled average of Precision and Recall. It provides a balanced measure of model performance, especially for multi-categorization scenarios or when there is an imbalance between false positives and false negatives.

```
# Cell 10: Compare Accuracy and F1 Score of All Classification Models
# (excluding FaceNet-style embedding model)

# Extract results (only for classification models)
classification_model_names = [
    name for name in results.keys() if name != "FaceNet-style Embedding"]
accuracies = [results[m]['accuracy'] for m in classification_model_names]
f1_scores = [results[m]['f1'] for m in classification_model_names]

# Accuracy comparison
plt.figure(figsize=(10, 6))
bars_acc = plt.bar(classification_model_names, accuracies, color='skyblue')
plt.xticks(rotation=45, ha='right')
plt.ylim(0, 1)
plt.ylabel('Accuracy')
plt.title('Classification Model Accuracy Comparison')
plt.tight_layout()

for bar in bars_acc:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.02,
             round(yval, 4), ha='center', va='bottom')
plt.show()

# F1 score comparison
plt.figure(figsize=(10, 6))
bars_f1 = plt.bar(classification_model_names, f1_scores, color='lightcoral')
plt.xticks(rotation=45, ha='right')
plt.ylim(0, 1)
plt.ylabel('F1 Score')
plt.title('Classification Model F1 Score Comparison')
plt.tight_layout()

for bar in bars_f1:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.02,
             round(yval, 4), ha='center', va='bottom')
plt.show()
```

Figure 4.18 : Comparative bar charts for accuracy and F1 scores of Classification Models

3. Confusion Matrix: A detailed table that shows the performance of the classification model. It shows the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) for each category, thus providing a detailed insight into the model's success in categorizing and the occurrence of errors (misclassification).

```
for model_name, y_pred in predictions.items():
    evaluate_model(model_name, y_test, y_pred)
```

Figure 4.19 : Confusion Matrix of Classification Models

4. Receiver Operating Characteristics (ROC) Curve vs. Area Under the Curve (AUC): The ROC curve, with True Positive Rate (TPR) on the Y-axis and False Positive Rate (FPR) on the X-axis, depicts the trade-offs of the model's performance at different classification thresholds. It visualizes the performance of the model at different sensitivities and specificities. The area under the curve (AUC) serves as a comprehensive metric that quantifies the overall ability of the model to discriminate between positive and negative samples; an AUC value closer to 1 indicates superior model performance. In this study, for Mult categorical problems, the ROC curves are presented using the Micro-Average method, which is calculated by aggregating the true positives, false positives, true negatives, and false negatives across all categories.

```
# Cell 11: Plot ROC Curves for Classification Models (excluding FaceNet-style model)

# One-hot encode true labels
y_test_binarized = label_binarize(y_test, classes=range(n_classes))

plt.figure(figsize=(12, 8))

# Combine predicted probabilities from traditional models and original CNN classifier
all_predictions_proba_for_roc = {
    **predictions_proba_traditional, **predictions_proba_cnn_classifier}
colors = sns.color_palette("husl", len(all_predictions_proba_for_roc))

for i, (model_name, y_proba) in enumerate(all_predictions_proba_for_roc.items()):
    # Compute ROC curve and AUC score (One-vs-Rest strategy for multi-class classification)
    fpr, tpr, _ = roc_curve(y_test_binarized.ravel(), y_proba.ravel())
    roc_auc = auc(fpr, tpr)

    plt.plot(fpr, tpr, color=colors[i], linestyle='-',
             label=f'{model_name} (AUC = {roc_auc:.2f})', lw=2)

# Plot diagonal line as chance level
plt.plot([0, 1], [0, 1], 'k--', lw=2, label='Chance level (AUC = 0.50)')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve - Micro-Average for Classification Models')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

Figure 4.20 : ROC Curve Analysis of Classification Models (Micro-Averaged)

Together, these metrics provide a comprehensive understanding of the precision, recall, and overall effectiveness of each model in classifying known face identities.

4.4.4 Experimental Results and Comparative Analysis

The performance of machine learning classification models (PCA+SVM, Random Forest, KNN, LDA+SVM) and CNN classifiers were systematically evaluated on the test set by accuracy, F1 scores, confusion matrices and ROC curves.

Performance Metrics and Visualization Comparison

Figures 4.21 to 4.25 show the accuracy, F1 scores, and confusion matrices for all evaluated traditional machine learning models and CNN classifiers, respectively. Each figure clearly demonstrates the model's classification ability and potential misclassification, thus enabling a direct quantitative comparison of their performance.

```
===== Evaluating Model: PCA + SVM =====
                   precision    recall  f1-score   support

     Ariel Sharon       1.00      0.63      0.77        19
     Colin Powell       0.76      0.93      0.84        59
  Donald Rumsfeld       0.86      0.63      0.73        30
    George W Bush       0.84      0.95      0.89       133
Gerhard Schroeder       0.91      0.78      0.84        27
      Hugo Chavez       1.00      0.61      0.76        18
       Tony Blair       1.00      0.83      0.91        36

         accuracy                           0.85       322
        macro avg       0.91      0.77      0.82       322
     weighted avg       0.87      0.85      0.85       322

Accuracy: 0.8540, F1 Score (macro): 0.8205
```

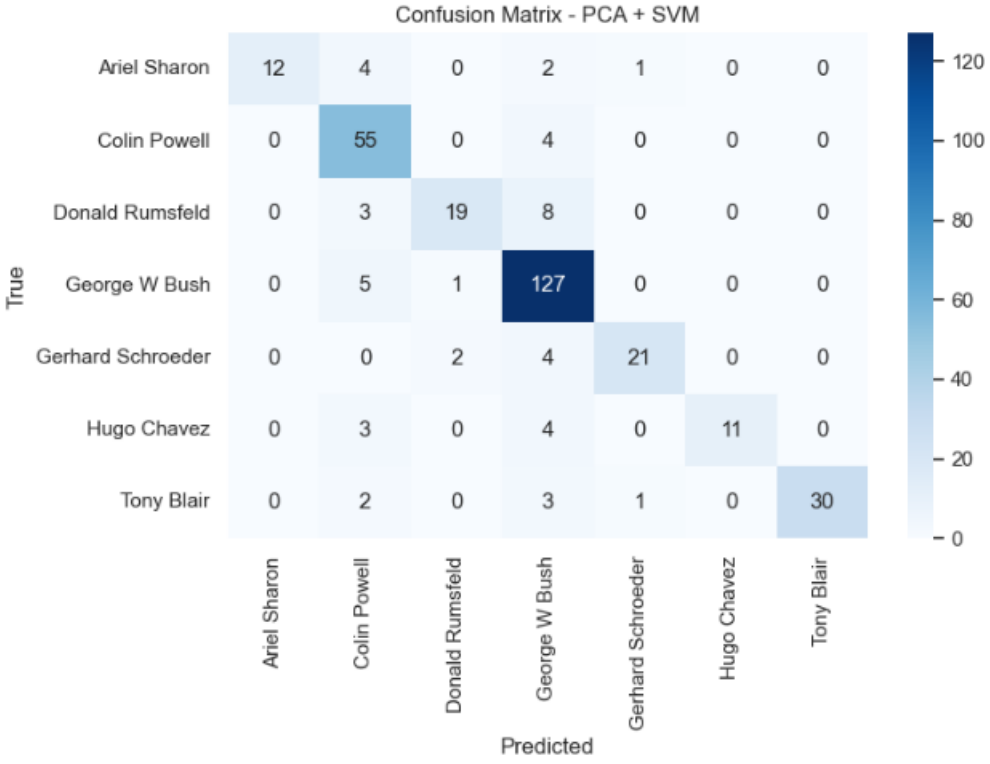


Figure 4.21 : Performance Evaluation Result of PCA + SVM

```
===== Evaluating Model: PCA + Random Forest =====
                   precision    recall  f1-score   support

     Ariel Sharon       0.00      0.00      0.00        19
     Colin Powell       0.82      0.61      0.70        59
  Donald Rumsfeld       1.00      0.17      0.29        30
    George W Bush       0.49      0.98      0.65       133
Gerhard Schroeder       1.00      0.04      0.07        27
      Hugo Chavez       0.00      0.00      0.00        18
       Tony Blair       1.00      0.08      0.15        36

         accuracy                           0.55       322
        macro avg       0.62      0.27      0.27       322
     weighted avg       0.64      0.55      0.45       322

Accuracy: 0.5466, F1 Score (macro): 0.2660
```

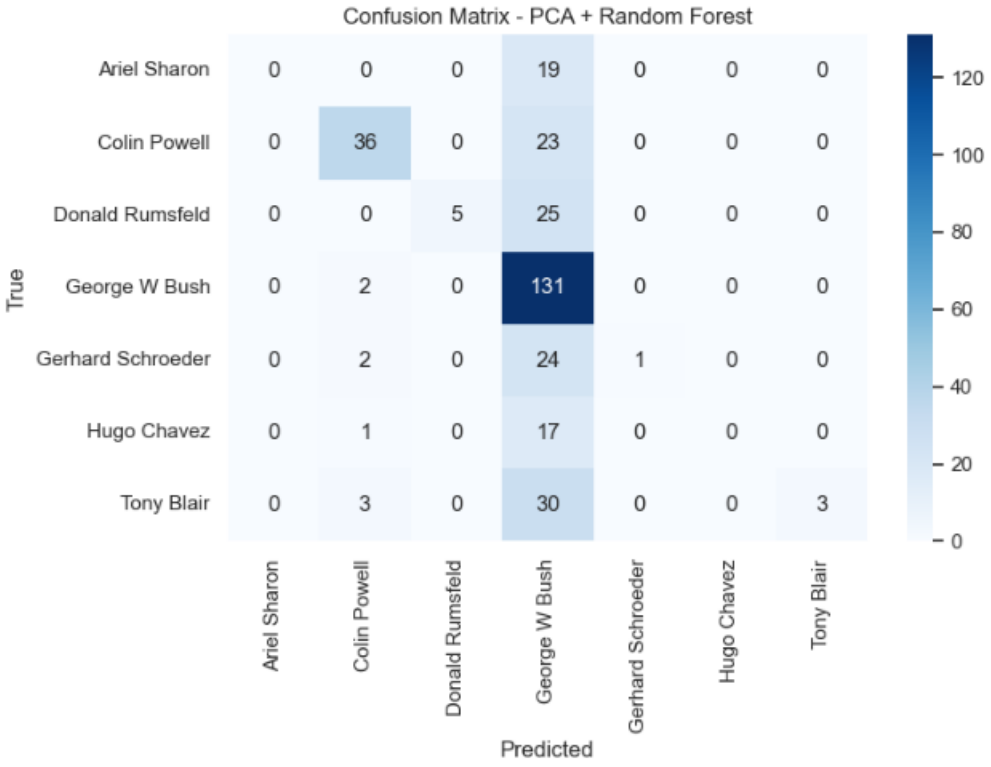


Figure 4.22 : Performance Evaluation Result of PCA + Random Forest

```
===== Evaluating Model: PCA + KNN =====
                   precision    recall  f1-score   support

     Ariel Sharon       0.77      0.53      0.62        19
     Colin Powell       0.68      0.69      0.69        59
  Donald Rumsfeld       0.62      0.43      0.51        30
    George W Bush       0.64      0.90      0.75       133
Gerhard Schroeder       0.75      0.33      0.46        27
      Hugo Chavez       0.43      0.17      0.24        18
       Tony Blair       0.64      0.39      0.48        36

         accuracy                           0.65       322
        macro avg       0.65      0.49      0.54       322
     weighted avg       0.65      0.65      0.63       322

Accuracy: 0.6522, F1 Score (macro): 0.5369
```

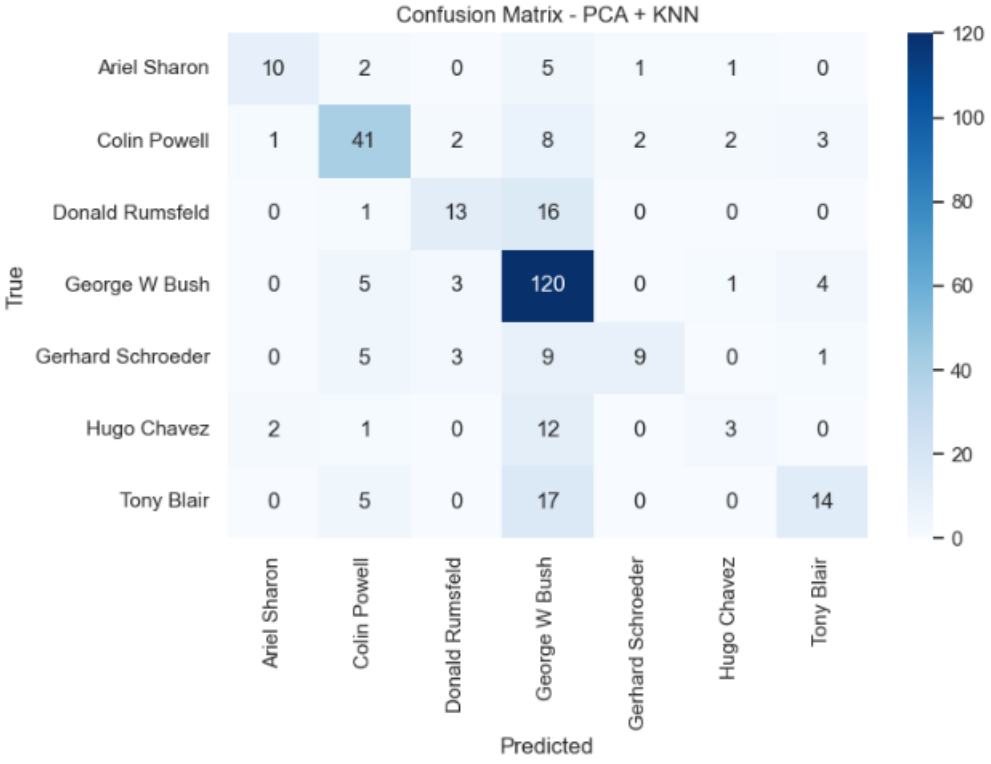


Figure 4.23 : Performance Evaluation Result of PCA + KNN

```
===== Evaluating Model: PCA + LDA + SVM =====
                   precision    recall  f1-score   support

     Ariel Sharon       0.80      0.63      0.71        19
     Colin Powell       0.86      0.86      0.86        59
  Donald Rumsfeld       0.72      0.60      0.65        30
    George W Bush       0.86      0.93      0.89       133
Gerhard Schroeder       0.74      0.74      0.74        27
      Hugo Chavez       0.82      0.78      0.80        18
       Tony Blair       0.91      0.86      0.89        36

         accuracy                           0.84       322
        macro avg       0.82      0.77      0.79       322
     weighted avg       0.84      0.84      0.84       322

Accuracy: 0.8385, F1 Score (macro): 0.7919
```

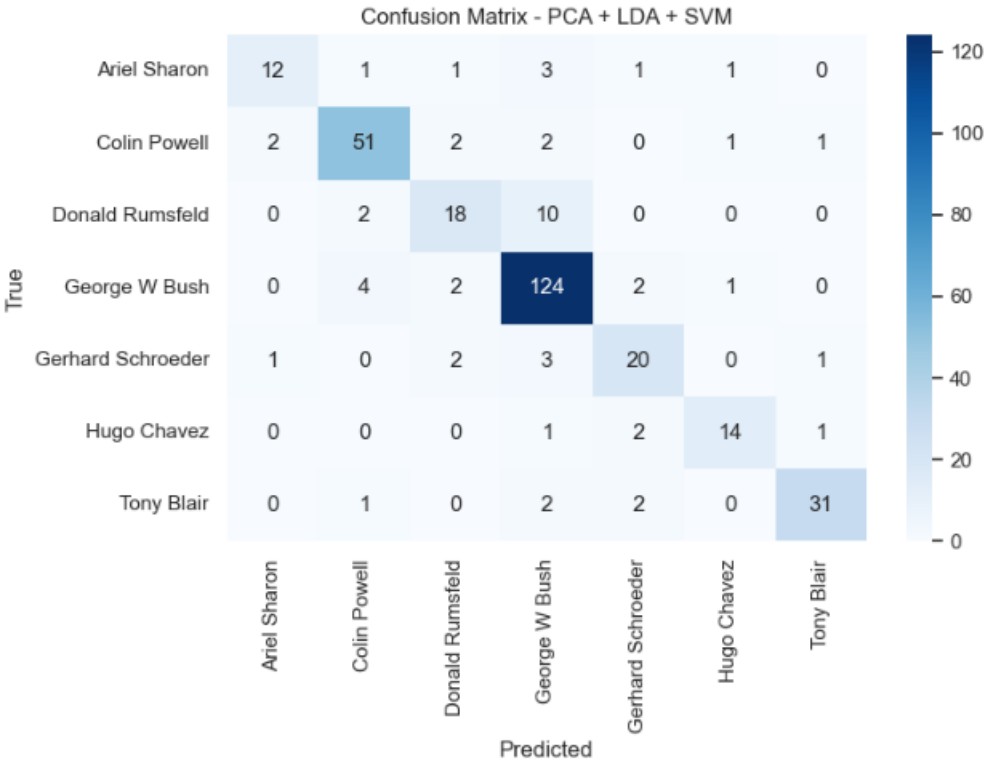


Figure 4.24 : Performance Evaluation Result of PCA + LDA + SVM

```
===== Evaluating Model: CNN =====
                   precision    recall  f1-score   support

     Ariel Sharon       0.89      0.89      0.89        19
     Colin Powell       0.90      0.93      0.92        59
  Donald Rumsfeld       0.88      0.73      0.80        30
    George W Bush       0.88      0.98      0.93       133
Gerhard Schroeder       0.88      0.78      0.82        27
      Hugo Chavez       1.00      0.61      0.76        18
       Tony Blair       0.83      0.81      0.82        36

         accuracy                           0.89       322
        macro avg       0.89      0.82      0.85       322
     weighted avg       0.89      0.89      0.88       322

Accuracy: 0.8851, F1 Score (macro): 0.8484
```

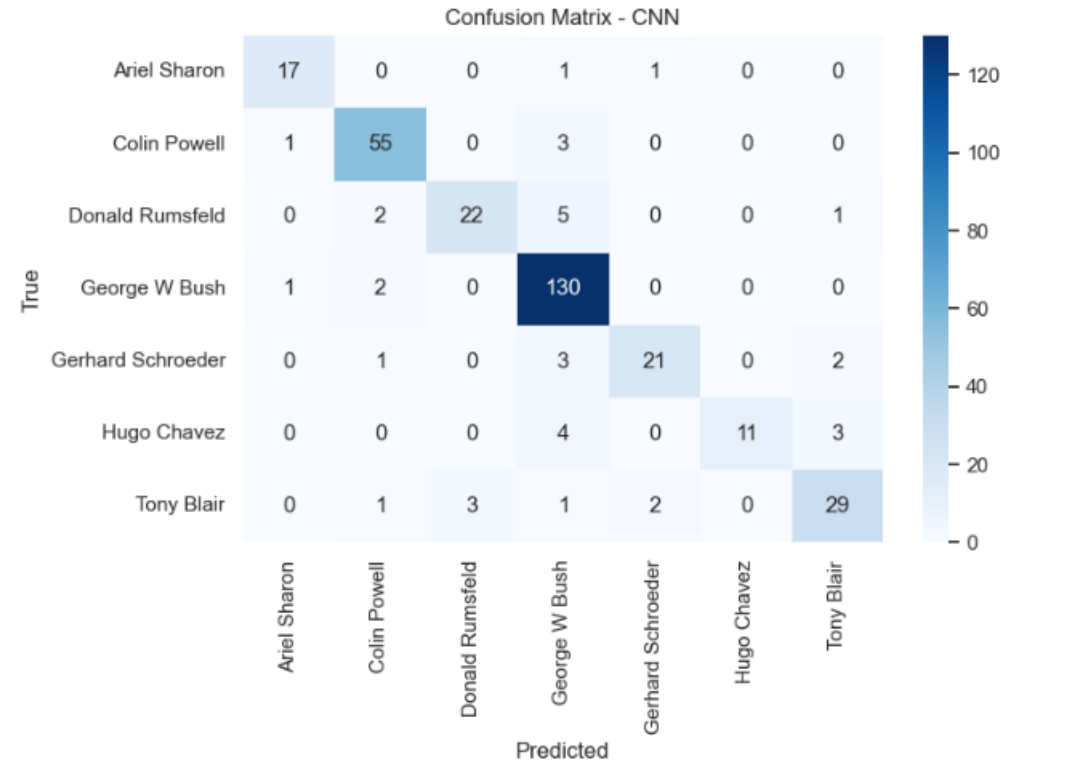


Figure 4.25 : Performance Evaluation Result of CNN Classifier

To visually emphasize the comparative performance between different classification models, Figure 4.26 provides a comparative histogram of the accuracy of the machine learning model and the CNN classifier, while Figure 4.27 shows a comparative histogram of the F1 scores.

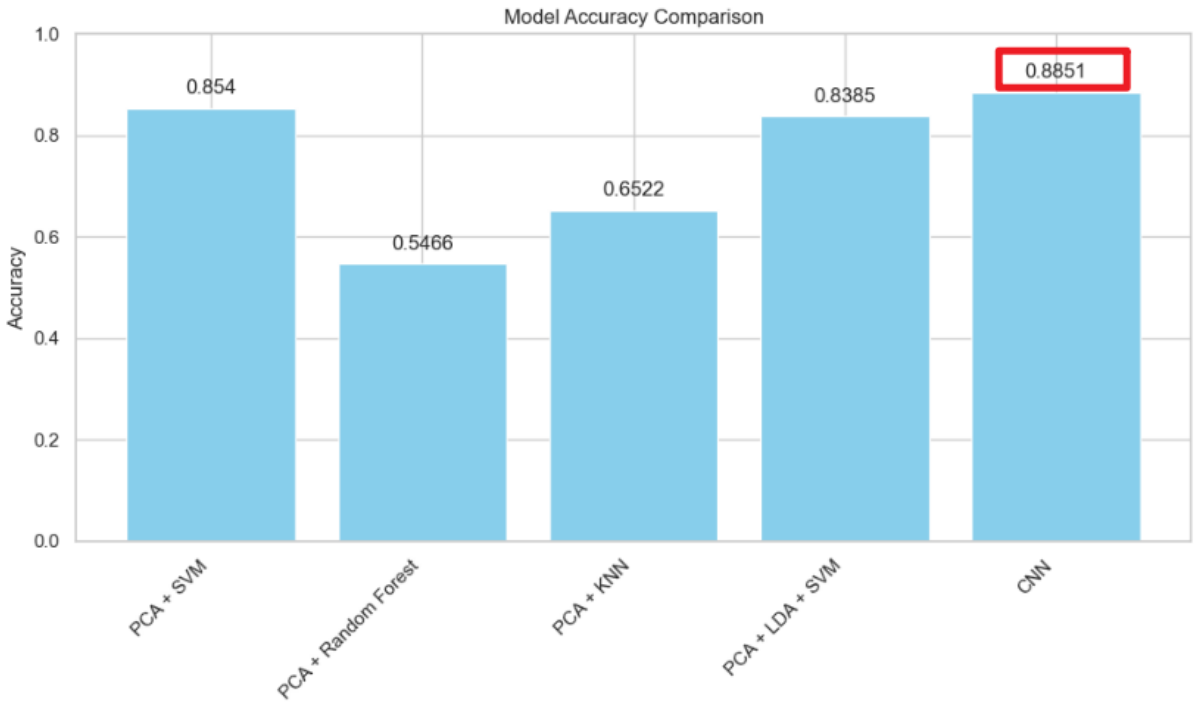


Figure 4.26 : Comparative Bar Chart of Accuracy of Classification Models

Visual analysis shows that the CNN classifier achieves the highest accuracy of 0.8851. This highlights its power in automatically learning hierarchical features from raw images for classification. Close behind, the PCA + SVM model also shows strong performance with an accuracy of 0.854, which demonstrates that traditional methods can also achieve good results when optimized with a dimensionality reduction configuration. Among other traditional models, the combination of PCA + LDA + SVM also performs well with an accuracy of 0.8385, highlighting the strength of LDA in enhancing the separability of subsequent classification categories. In contrast, PCA + KNN has a more mediocre accuracy of 0.6522, while PCA +

Random Forest has the lowest accuracy of 0.5466, which hints at its relative inapplicability or sub-optimal configuration in this particular dataset and feature space.

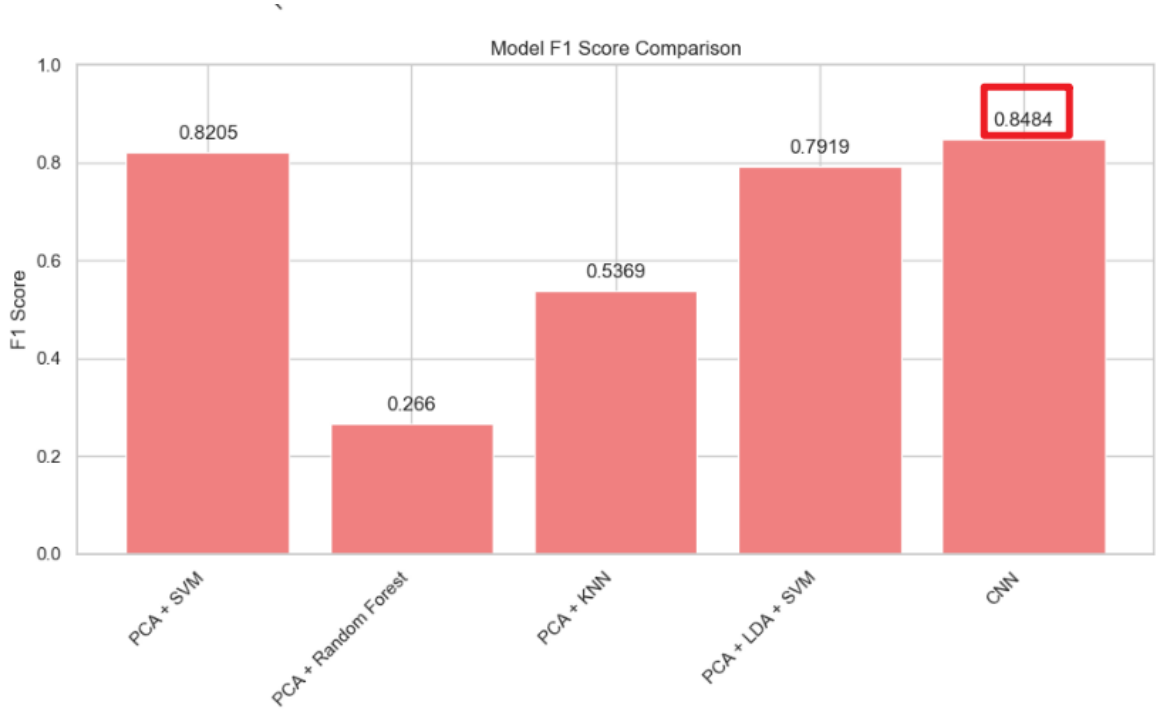


Figure 4.27 : Comparative Bar Chart of F1 Scores of Classification Models

Accordingly, in terms of F1 scores, the CNN classifier achieved the highest F1 score of 0.8484, which shows that it achieved an excellent balance between precision and recall in this task. PCA + SVM follows closely with an F1 score of 0.8205, showing its robust performance. The PCA + LDA + SVM model also performs well with an F1 score of 0.7919. These F1 scores generally corroborate the accuracy results, highlighting that the best performing models maintain a good balance between identifying positive examples and minimizing false positives. In contrast, the PCA + KNN model had an F1 score of 0.5369, while PCA + Random Forest again achieved the lowest F1 score of 0.266, further confirming its weaker performance in this context. The F1 scores provide a more robust way of evaluating, especially in the presence of possible category imbalance, and reinforce the observation that deep learning (represented by

CNN classifiers) provides competitive and often leading performance in this face recognition task.

Figure 4.28 illustrates the micro-averaged receiver operating characteristic (ROC) curves for all the multiclass classification models, providing a comprehensive picture of their discriminative ability under different thresholds.

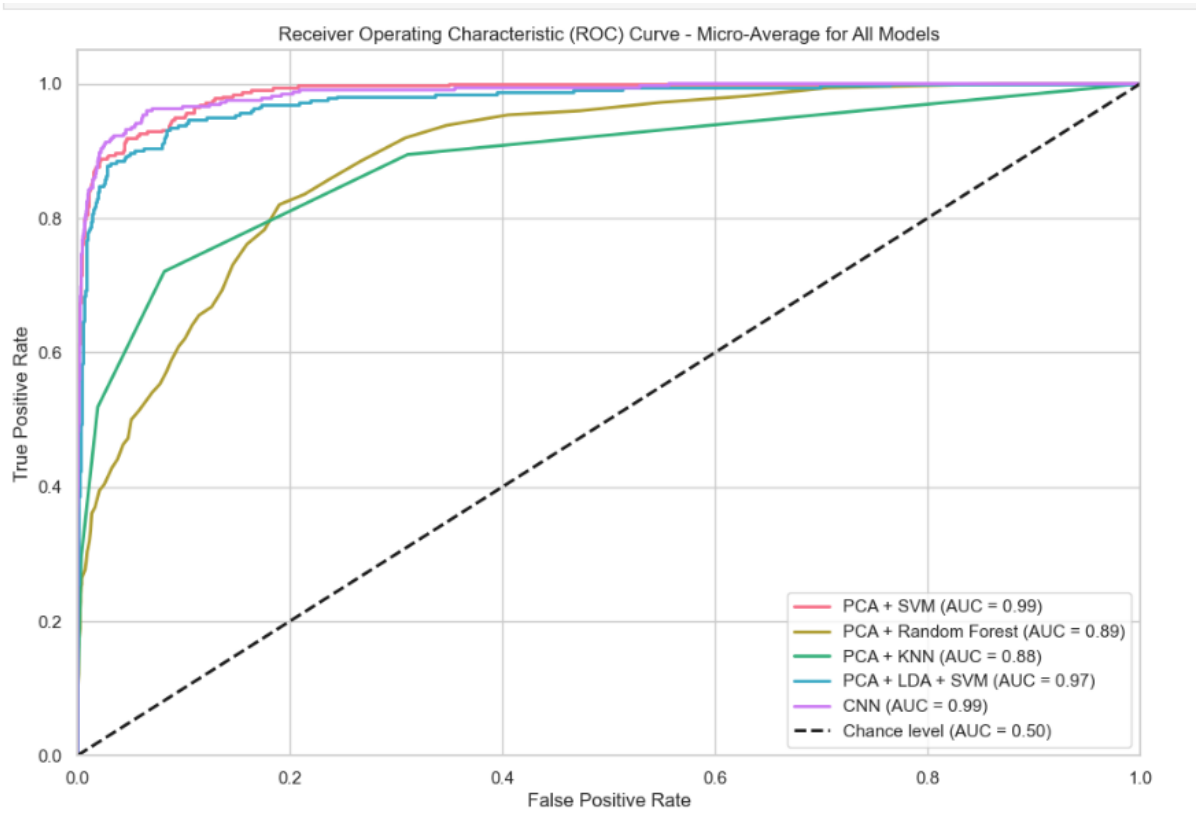


Figure 4.28 : ROC Curve for Classification Model

Figure 4.28 illustrates the micro-averaged Receiver Operating Characteristic curves (ROCs) of all the evaluated multi-class classification models, which fully elucidates their discriminative abilities under different classification thresholds. As shown, the PCA + SVM model and the CNN classifier have the highest overall discriminative performance, with micro-average AUC values of 0.99 for both. Similarly, the PCA + LDA + SVM combination also performs well with an AUC value of 0.97. In contrast, the PCA + Random Forest model has a

significantly lower AUC value of 0.89, while the PCA + KNN model has an AUC value of 0.88, showing the weakest discriminative ability. These AUC values coincide with the results of the accuracy and F1 scores, and together confirm the strong ability of the best performing model to effectively discriminate between different identities in the closed-set recognition paradigm.

CNN Training Dynamics and Overfitting Analysis

Regarding the training dynamics of the custom CNN classification model, Figure 4.29 illustrates its loss curves and accuracy curves for 100 training cycles (epochs).

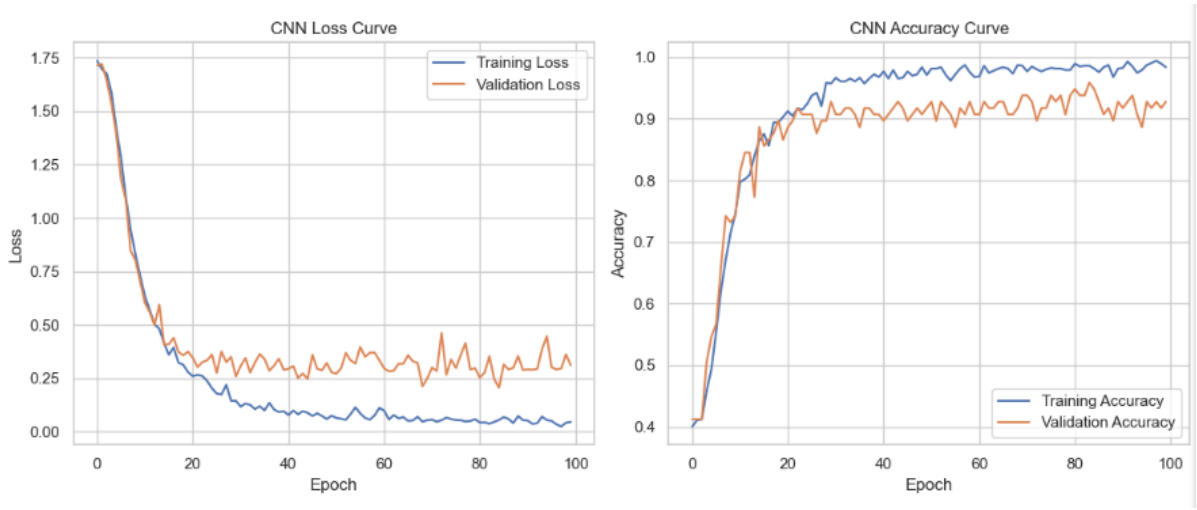


Figure 4.29 : Loss and Accuracy curves for CNN Classification Model

Observing these curves, we can derive the following specific insights:

In the early stage of training (about 0-20 training cycles), both the training loss (blue curve) and validation loss (orange curve) show a sharp downward trend, indicating that the model is learning efficiently and capturing the essential features in the data quickly. At the same time, the training accuracy and validation accuracy also increase rapidly, with the training accuracy rapidly increasing from about 0.4 to about 0.90 and the validation accuracy reaching about 0.85, confirming the model's ability to learn efficiently at this initial stage.

Moving into the middle stage of training (about 20-70 training cycles), the rate of decline in training loss begins to slow down and gradually stabilizes at a very low level (close to 0.05). Training accuracy continues to climb steadily and eventually stabilizes at around 0.95-0.98. However, over this longer period, the validation loss curve begins to fluctuate more markedly and shows a subtle upward trend, while the validation accuracy, while remaining generally high (around 0.90-0.94), increases in volatility and shows a clearer gap with the training accuracy. This suggests that the model is beginning to overfit the training data.

From approximately 70 training cycles to 100 training cycles (late stage), the training loss continued to hover at very low values and the training accuracy remained at its peak (close to 1.0). However, the volatility of the validation loss increases further and shows a more pronounced upward trend, while the validation accuracy remains relatively stable in this range, but is always significantly lower than the training accuracy. This phenomenon, i.e., the training loss continues to decline while the validation loss exhibits a more pronounced rebound, while the training accuracy is significantly higher than the validation accuracy, clearly indicates that the model experiences significant overfitting during this expansion phase. This implies that the model has learnt too much ‘specificity’ on the training data, which may contain noise or unique patterns, leading to a slight decrease in its generalization ability when faced with new, unseen data (validation set).

Overall, Figure 4.29 confirms that the CNN classification model has a strong learning ability on the LFW dataset and achieves high classification accuracy. Despite the significant overfitting observed in the later stages of training, the model exhibits robust initial performance. Future work could consider introducing stronger regularization techniques or employing an early-stop strategy to stop training when the validation loss starts to increase consistently, thus further optimizing the model's generalization performance and preventing excessive overfitting.

4.4.5 Comprehensive Discussion on Classification Models

A comprehensive evaluation of traditional machine learning models and deep learning classifiers (evaluated by accuracy, F1 scores, confusion matrices, and ROC curves) reveals several key insights. The following table summarizes the performance of all classification models on the test set.

Model	Accuracy	F1 Score	Micro-Average AUC
CNN Classifier	0.8851	0.8484	0.99
PCA + SVM	0.854	0.8205	0.99
PCA + LDA + SVM	0.8385	0.7919	0.97
PCA + KNN	0.6522	0.5369	0.88
PCA + Random Forest	0.5466	0.266	0.89

Table 4.1: Model Performance Summary

Among the classical methods, the **PCA + SVM** model demonstrated excellent baseline performance with an accuracy of 0.854, an F1 score of 0.8205, and achieved an excellent AUC value of 0.99. This demonstrates that the classical algorithm is capable of establishing a robust performance baseline when appropriate dimensionality reduction techniques such as Principal Component Analysis (PCA) are applied. This is consistent with the findings of Chen and Jenkins (2017), who noted that PCA + SVM can achieve recognition rates in excess of 95% on certain datasets and feature face sizes, highlighting its robustness when combined with appropriate dimensionality reduction methods.

However, the CNN classifier demonstrated better overall performance, with an accuracy of 0.8851, an F1 score of 0.8484, and a micro-averaged AUC value of 0.99, which was tied for the highest with the PCA + SVM model. This not only highlights CNN's powerful ability to automatically extract hierarchical features without manual feature engineering, but

also shows that it has top discriminative performance at different classification thresholds. Despite the significant overfitting observed during training (see Section 4.4.4 for more details), the overall discriminative performance of CNN in closed-set classification tasks is still very strong. This has been supported by several comparative studies. For example, Saha et al. (2024) demonstrated that the CNN model outperforms the SVM in terms of accuracy, recall and F1 scores on different facial datasets, despite requiring longer training and prediction times.

Crucially, while the CNN's accuracy of 88.51% is commendable in a research context, it presents significant limitations for high-security time and attendance systems. In such real-world deployments, where false positives or false negatives can have critical implications (e.g., unauthorized access), an 88.51% accuracy rate is generally insufficient. For applications demanding high security and reliability, a minimum acceptable accuracy threshold typically needs to be well above 95%, often striving for 99% or higher. This higher threshold is necessary to minimize errors that could compromise security, operational efficiency, and data integrity.

In contrast, models such as PCA + Random Forest and PCA + KNN performed relatively weakly, reaffirming that not all algorithms are equally suited to high-dimensional image data, even after dimensionality reduction or in configurations that are not suitable for this particular dataset and feature space.

Although CNN classifiers exhibit high recognition performance in closed-set classification tasks, they have inherent limitations in open-set face recognition time and attendance system applications, including the need for model retraining when new users are added, scalability issues with increasing categories, and the inability to directly measure identity similarity. These fundamental challenges force the research strategy to shift towards metric learning. Thus, a deep metric learning network is introduced that aims to address the

above challenges by mapping faces into the feature space such that distance directly represents similarity.

4.5 Deep Metric Learning Network: Necessity, Implementation, and Evaluation

This section details the design, implementation and rigorous evaluation of the deep metric learning network based on CNN architecture. The model is chosen to overcome the inherent limitations of closed-set classifiers and to meet the urgent need for open-set recognition in smart attendance systems.

4.5.1 Necessity of Metric Learning for Open-Set Face Recognition

As shown in Section 4.4, traditional machine learning classifiers, even Convolutional Neural Networks (CNNs), have shown commendable performance in face classification tasks in closed scenes. However, their basic architecture poses significant limitations in the face of the dynamic and ever-changing requirements of smart time and attendance systems, which inherently operate in open environments. These key limitations include:

1. Scalability issues and retraining requirements: Classification models require a dedicated output neuron for each known identity. As a result, each new user addition requires complete or partial retraining of the model, which is not only computationally expensive, but also impractical for real-world deployments where the user base changes frequently.
2. Increased Model Complexity: As the number of registered users (categories) increases, the output layer of the classification CNN becomes too large, resulting in increased model complexity, memory footprint, and slower inference.
3. Inability to quantify similarity: Classification models output probability distributions for known categories, but inherently do not directly measure the

similarity or dissimilarity between arbitrary face pairs, which is critical for verifying unseen individuals or performing one-shot recognition.

In order to address the adaptability of traditional face recognition methods in open environments, this project adopts Deep Metric Learning approach and uses it as the core of the face recognition module in the attendance system. Traditional classification-based models have limited generalization ability when faced with unseen identities, so the shift from a direct classification paradigm to a metric learning paradigm becomes crucial to improve the robustness of the system.

Metric learning allows the Euclidean distance to measure the semantic similarity between images by mapping face images into a low-dimensional embedding space: images of the same identity are clustered in this space and different identities are separated. This mechanism supports the recognition of new identities without retraining, which significantly enhances the scalability and utility of the system in open-set scenarios.

In this system, a CNN architecture is used to extract face features and trained with Triplet Loss to learn discriminative embedding spaces. This method fuses the feature representation capability of deep learning with the discriminative capability of metric learning, which improves the recognition accuracy and also enhances the robustness and generalization of the system in complex environments.

4.5.2 Architecture of Deep Metric Learning Network

The core goal of the deep metric learning network is to map face images into a low-dimensional Euclidean feature space (embedding space) such that embedding vectors of the same face are close to each other and embedding vectors of different faces are far away from each other.

This deep metric learning network uses a customized Convolutional Neural Network (CNN) as its backbone network for feature extraction. The architecture is designed to efficiently capture hierarchical visual features from raw image data:

1. Input Layer: The network accepts as input an RGB image of size 160x160.
2. Convolutional Blocks: The backbone network consists of a series of Conv2D layers with an increasing number of convolutional kernels (64, 128, 256). Each Conv2D layer uses 3x3 convolutional kernels and ReLU activation functions, and is immediately followed by a Batch Normalization layer and a MaxPooling2D layer (with a pooling size of 2x2). This design aims to extract increasingly abstract and robust features.
3. Spreading and Dense Layer: After the convolutional block, the feature map is spread and passed to a Dense layer containing 512 neurons, which also uses the ReLU activation function. This is followed by a Batch Normalization layer and a Dropout layer (with a dropout rate of 0.5) to prevent overfitting.
4. Embedding Layer: The final layer of the network is a fully connected layer containing embedding_dim=512 units, which does not use an activation function. Its raw output is then explicitly L2 normalized by a Lambda layer (Lambda(lambda x: K.l2_normalize(x, axis=1))). This normalization is crucial because it ensures that all embedding vectors lie on the hypersphere, thus simplifying distance comparisons and maintaining consistency of embedding magnitudes, which is essential for triple loss stability.

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 160, 160, 3)	0
conv2d_2 (Conv2D)	(None, 158, 158, 64)	1,792
batch_normalization (BatchNormalization)	(None, 158, 158, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 79, 79, 64)	0
conv2d_3 (Conv2D)	(None, 77, 77, 128)	73,856
batch_normalization_1 (BatchNormalization)	(None, 77, 77, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 38, 38, 128)	0
conv2d_4 (Conv2D)	(None, 36, 36, 256)	295,168
batch_normalization_2 (BatchNormalization)	(None, 36, 36, 256)	1,024
max_pooling2d_4 (MaxPooling2D)	(None, 18, 18, 256)	0
flatten_1 (Flatten)	(None, 82944)	0
dense_2 (Dense)	(None, 512)	42,467,840
batch_normalization_3 (BatchNormalization)	(None, 512)	2,048
dropout_1 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 512)	262,656
lambda (Lambda)	(None, 512)	0

Total params: 43,105,152 (164.43 MB)

Trainable params: 43,103,232 (164.43 MB)

Non-trainable params: 1,920 (7.50 KB)

Figure 4.30 : Architecture of Deep Metric Learning Network

In order to clearly present the architecture design strategy of the deep metric learning network in this study, its core hyperparameters and their selection rationale will be summarized as shown in Figure 4.31.

Hyperparameter Category	Hyperparameter	Value/Choice	Brief Explanation
Input Layer	Input Dimension	160x160 (RGB)	Uses higher resolution RGB images to capture richer facial details.
Convolutional Blocks (Conv2D)	Number of Kernels	64, 128, 256	Progressively increased to extract more abstract and robust features, a common pattern in deep networks.
	Kernel Size	3x3	Popular choice for effective local feature extraction and computational efficiency.
	Activation Function	ReLU	Addresses vanishing gradient problem and improves computational efficiency.
Pooling Layer (MaxPooling2D)	Pooling Size	2x2	Common down-sampling strategy, effectively reduces feature map size and enhances translation invariance.
Fully Connected Layer (Dense)	Number of Neurons	512	Empirically chosen dimension for high-level feature abstraction.
	Activation Function	ReLU	Same as above.
Dropout Layer	Dropout Rate	0.5	Common rate, effectively mitigates overfitting by randomly deactivating neurons.
Embedding Layer	Embedding Dimension	512	Dimension of output feature vectors, suitable for high-dimensional embedding in face recognition.
Normalization Strategy		L2 Normalization	Ensures embeddings lie on a hypersphere, stabilizing triplet loss and simplifying distance comparisons.

Figure 4.31 : Hyperparameters of the Architecture of Deep Metric Learning Network

4.5.3 Triple loss Functions and Training Strategies

Deep metric learning networks are trained using the Triplet Loss function, which is the cornerstone of face recognition metric learning. Triplet Loss acts on a triad of three images:

- Anchor (A): a reference image.
- Positive (P): an image of the same person as the anchor.
- Negative (N): an image of a different person than the anchor.

The goal of the triple loss is to enforce the condition $d(A,P)+\alpha<d(A,N)$, where d denotes the Euclidean distance between the embedding vectors, and α (0.2 in code) is a hyperparameter that represents the minimum margin required between pairs of positive and negative samples. This loss function directly motivates the model to learn an embedding space where intra-class distances are minimized and inter-class distances are maximized.

Triplet Generation Strategy: A key aspect of training with triple loss is the triple mining strategy. Based on the get_triplets' function in Cell 7 of the code, the implementation in this study employs a simplified offline, per-round, randomized triple generation strategy. Each training round, a new set of NUM_TRIPLETS_PER_EPOCH (5000) triplets is randomly selected from the training data.

```python
def get_triplets(X, y, num_triplets_per_epoch=2000, seed=None):
    if seed is not None:
        np.random.seed(seed)

    unique_labels = np.unique(y)
    triplets = []

    # Generate as many triplets as possible until the desired number is reached
    for _ in range(num_triplets_per_epoch):
        # Randomly select an anchor class
        anchor_class_idx = np.random.choice(unique_labels)

        # Get all indices belonging to the anchor class
        anchor_indices = np.where(y == anchor_class_idx)[0]

        # Skip if there are not enough samples to form an anchor-positive pair
        if len(anchor_indices) < 2:
            continue

        # Randomly select an anchor and a positive (same class)
        ap_indices = np.random.choice(anchor_indices, 2, replace=False)
        anchor_img = X[ap_indices[0]]
        positive_img = X[ap_indices[1]]

        # Select a negative class different from the anchor class
        other_labels = unique_labels[unique_labels != anchor_class_idx]
        if len(other_labels) == 0:
            continue  # Cannot select a negative if only one class is present

        negative_class_idx = np.random.choice(other_labels)

        # Get all indices of the negative class
        negative_indices = np.where(y == negative_class_idx)[0]

        # Skip if no samples are found for the negative class
        if len(negative_indices) == 0:
            continue

        # Randomly select a negative image
        negative_img = X[np.random.choice(negative_indices, 1)[0]]

        triplets.append((anchor_img, positive_img, negative_img))

    anchors = np.array([t[0] for t in triplets])
    positives = np.array([t[1] for t in triplets])
    negatives = np.array([t[2] for t in triplets])

    return anchors, positives, negatives
```

Figure 4.32 : Triple mining strategy for Deep Metric Learning Networks

The model is compiled using the Adam optimizer. The training process consists of a custom loop that iterates over 120 training rounds, processing 32 triples at a time. The training process mainly monitors the decrease in triple loss.

The key hyperparameters used in the training process of the deep metric learning network in this study and their selection rationale are summarized in Figure 4.33.

Hyperparameter Category	Hyperparameter	Value/Choice	Brief Explanation
Loss Function	Type	Triplet Loss	Core for face recognition metric learning, optimizes intra-class compactness and inter-class separation.
	Margin	$\alpha = 0.2$	Enforces a minimum distance difference between positive and negative pairs, ensuring effective discrimination.
Triplet Generation	Strategy	Offline, per-round random generation	Simplified triplet generation strategy, randomly selected each round to increase sample diversity.
	Triplet Count per Epoch	5000	Number of triplets generated per training epoch, balancing training data volume.
Optimizer	Type	Adam	Adaptive learning rate, suitable for handling sparse gradients, with good convergence performance.
Training Process	Training Rounds/Epochs	120	Total iterations for model training; effective convergence observed within this period.
	Batch Size	32	Number of triplets processed per gradient update, balancing computational resources and convergence efficiency.

Figure 4.33 : Hyperparameters for the training process of Deep Metric Learning Networks

4.5.4 Evaluation Methodology (open set validation tasks)

Deep metric learning networks are specifically evaluated for their performance in open-set verification tasks, which is crucial to assess their ability to distinguish identity similarities and to handle individuals not present in training. Given the focus of this study and the data limitations, we assessed the embedding quality of the model primarily through distance distribution analysis.

Distance Distribution Plots are (histogram) visualizations of the distribution of intra-class distances (distances between embedding vectors for the same face) and inter-class distances (distances between embedding vectors for different faces). Ideally the discriminative embedding space should show two distinct, non-overlapping distributions, where the intra-class distances are clustered at values close to zero and the inter-class distances are significantly larger. By analyzing the mean and standard deviation of these two distance distributions, as well as the degree of overlap between them, we can visually and quantitatively assess the model's ability to learn discriminative features.

```python
# --- Evaluation of FaceNet-style Embedding Quality ---
# Compute embeddings on test set
test_embeddings = facenet_embedding_network.predict(X_test_facenet)

# Calculate intra-class and inter-class distances
intra_class_distances = []
inter_class_distances = []

for i in range(n_classes):
    class_indices = np.where(y_test == i)[0]
    if len(class_indices) < 2:
        continue  # At least two samples required to compute intra-class distances

    class_embeddings = test_embeddings[class_indices]

    # Intra-class distances
    for j in range(len(class_embeddings)):
        for k in range(j + 1, len(class_embeddings)):
            dist = np.linalg.norm(class_embeddings[j] - class_embeddings[k])
            intra_class_distances.append(dist)

    # Inter-class distances (against all other classes)
    other_classes_indices = np.where(y_test != i)[0]
    if len(other_classes_indices) > 0:
        other_embeddings = test_embeddings[other_classes_indices]
        for emb1 in class_embeddings:
            for emb2 in other_embeddings:
                dist = np.linalg.norm(emb1 - emb2)
                inter_class_distances.append(dist)

if intra_class_distances:
    print(f"FaceNet-style: Average intra-class distance: {np.mean(intra_class_distances):.4f} +/- {np.std(intra_class_distances):.4f}")
else:
    print("Not enough samples to calculate intra-class distances for FaceNet-style model.")

if inter_class_distances:
    print(f"FaceNet-style: Average inter-class distance: {np.mean(inter_class_distances):.4f} +/- {np.std(inter_class_distances):.4f}")
else:
    print("Not enough samples to calculate inter-class distances for FaceNet-style model.")

# Optional: Visualize distance distributions
plt.figure(figsize=(10, 6))
sns.histplot(intra_class_distances, color='skyblue', label='Intra-class Distances', kde=True, stat='density', alpha=0.6)
sns.histplot(inter_class_distances, color='lightcoral', label='Inter-class Distances', kde=True, stat='density', alpha=0.6)
plt.title('FaceNet-style Embedding Distance Distributions')
plt.xlabel('Euclidean Distance')
plt.ylabel('Density')
plt.legend()
plt.grid(True)
plt.show()
```

Figure 4.34 : Code for visualizing distance distribution plot

This assessment directly reflects the model's ability to distinguish between different identities in the embedding space, and is particularly critical for open set systems that perform recognition based on similarity.

4.5.5 Experimental Results and Analysis

The evaluation of the deep metric learning network focuses on its ability to generate high-quality face embedding vectors, where the desirable features are minimization of the intra-class distance and maximization of the inter-class distance. The experimental results on the LFW test set are shown in Figure 4.35, where the model has an average intra-class distance of 0.5521 ± 0.2868 and an average inter-class distance of 1.3544 ± 0.2182.

```
Average intra-class distance: 0.5521 +/- 0.2868
Average inter-class distance: 1.3544 +/- 0.2182
```

Figure 4.35 : Mean and Standard Deviation of Intraclass and Interclass Distances for Deep Metric Learning Network

These quantitative values clearly indicate that the model has significant discriminatory power in distinguishing different identities. The large difference between the average interclasses distance (1.3544) and the average intraclass distance (0.5521) suggests that the network succeeds in tightly clustering embedded data with the same identity while effectively separating embedded data with different identities.

In order to provide a more intuitive and comprehensive picture of the embedding quality, Figure 4.36 (Intra- and inter-class distance distributions for deep metric learning networks) clearly depicts the histograms of the intra- and inter-class distance distributions. This visual representation effectively shows the degree of separation between the two distance distributions. An ideal embedding model should exhibit a smaller average intra-class distance and a larger average inter-class distance, and the overlap region between the two distributions should be as small as possible.

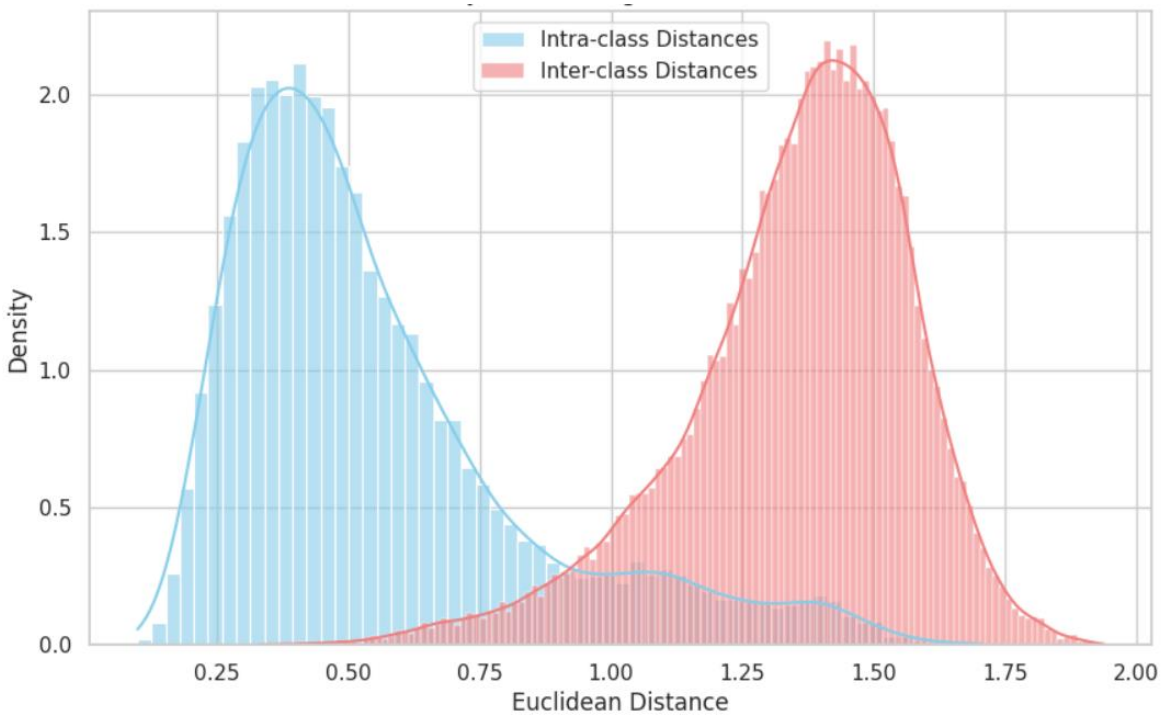


Figure 4.36 : Distributions of histograms of intra- and inter-class distances for deep metric learning network

As shown in Figure 4.36, the histogram reveals a clear and satisfactory separation between the intraclass (blue) and interclass (red) distance distributions. The intra-class distances are mainly clustered at smaller values, specifically around 0.35, while the inter-class distances are mainly clustered at larger values, around 1.4. Although there is still a certain degree of overlap in the tails of the two distributions, which may lead to a small amount of misclassification in practical applications (e.g., false positives where different identities are incorrectly matched, or misses where true matches are missed), this degree of separation demonstrated significantly proves the effectiveness of the model in learning discriminative features compared to the case where there is a large amount or complete overlap. This implies that in practical applications, a relatively robust distance threshold can be established for identity verification, which can help to reduce the rate of false positives and omissions.

4.5.6 Comprehensive Discussion on Deep Metric Learning Network

This section explores the performance of a deep metric learning network in an open-set face recognition task, and analyses its results to suggest future research directions.

It is shown that the network successfully learns face embeddings that can effectively discriminate different identities, and its performance on the LFW test set demonstrates good intra-class compactness and inter-class separation. This is similar to the performance of the FaceNet model proposed by Schroff et al. (2015), which achieves an efficient face feature discrimination capability through a triple loss function that significantly improves the performance of open-set recognition. This discriminative capability is particularly critical for open-set face recognition systems, as it eliminates the need for frequent re-training when supporting new user registrations, thus significantly improving the scalability and deployment efficiency of the system.

According to the visual analysis results in Section 4.5.5, in most cases, the Euclidean distances between people with the same identity are concentrated below 1.0, while the distances between different people are generally greater than 1.0. This is consistent with the observations of J. Wang et al. (2017) regarding the distribution of distances in metric learning. Although there are some overlapping regions in the data distribution, based on this initial discriminative property and the robustness requirements of the system in real-world applications, this project plans to deploy the model into an Android application in Chapter 5 with an initial Euclidean distance threshold of 1.0. This strategy is also supported by studies such as Deng et al. (2019), who emphasize the importance of reasonable threshold settings for the performance of real face recognition systems.

However, despite the overall effective separation, there is still overlap in the tails of the distance distribution. This means that in extreme cases, the model may produce false alarms or

omissions. Potential reasons for this include the inherent challenges of the LFW dataset (e.g., pose, lighting, and expression variations), the underutilization of 'difficult samples' by the random triad generation strategy used during training, and the inherent limitations of the current CNN backbone architecture.

Therefore, in order to build a more robust intelligent time and attendance system, future research should focus on improving the discriminative ability of face embeddings and minimizing the overlap of Euclidean distance distributions. To achieve this goal, the following measures can be taken, which are introducing more efficient triple mining strategies (e.g., online difficult triple mining), exploring and integrating more advanced CNN backbone network architectures, and using larger and more diverse face datasets for training.

4.6 Model Saving and Conversion

This section details the preservation strategy for all the training models in this study, in particular the conversion process of the deep learning models to the lightweight TensorFlow Lite (TFLite) format, which is a key component for subsequent successful deployment on resource-constrained devices.

Machine learning components, including PCA transformers, LDA transformers, and trained Support Vector Machines (SVMs), Random Forests, and K-Nearest Neighbor (KNN) classifiers, are systematically serialized using the Joblib library. Each component is saved as a separate .pkl file, ensuring its persistent storage and easy reloading for subsequent evaluation or integration. This approach ensures the reproducibility of the exact state of the model after training.

For CNN classification models, trained networks are first saved in Keras' native .keras format. This format preserves the model's architecture, weights, and optimizer state intact. The Keras model is then converted to TFLite format using the

tf.lite.TFLiteConverter.from_keras_model() tool. This conversion is critical for device-side deployments and provides significant benefits, including a dramatic reduction in model size and a significant increase in inference speed. These optimizations are essential for deploying deep learning models on mobile and embedded platforms where computational resources are limited.

The deep metric learning network underwent a similar saving and transformation process. It is important to emphasize that we save the embedding network (which is the core feature extractor that generates the embedding vectors) rather than the facenet_triplet_model that is only used for triplet loss training. This embedding network is subsequently converted to TFLite format using the same tf.lite.TFLiteConverter.from_keras_model() method, thus gaining the same advantages of model size reduction and inference acceleration. This is critical for its integration into the target application.

All saved model files, covering both machine learning models and deep learning models, are uniformly organized in their respective formats and stored in a dedicated directory called models. This standardized file structure ensures easy access and clear management of all training products.

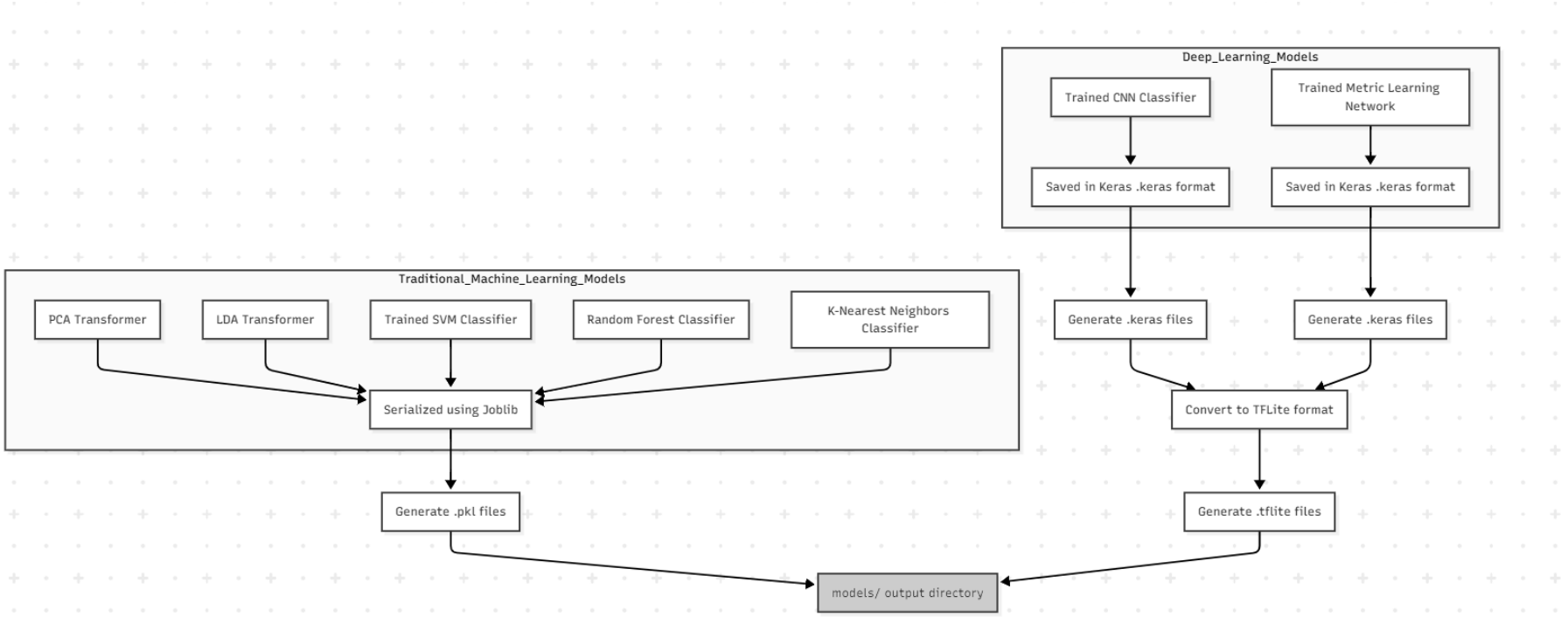


Figure 4.37 : Visual overview of the model saving and transformation process

```
lda_filename = os.path.join(models_dir_combined, 'lda_transformer.pkl')
joblib.dump(lda, lda_filename)
print(f"LDA transformer saved to: {lda_filename}")

svm_lda_filename = os.path.join(models_dir_combined, 'pca_lda_svm_model.pkl')
joblib.dump(svm_lda, svm_lda_filename)
print(f"PCA + LDA + SVM model saved to: {svm_lda_filename}")


# --- Save the original CNN classifier model and convert to TFLite ---
print("\nSaving the original CNN classifier model and converting to TFLite...")

cnn_keras_filename = os.path.join(
    models_dir_combined, 'cnn_classifier_model.keras')
cnn_model.save(cnn_keras_filename)
print(f"Original CNN classifier Keras model saved to: {cnn_keras_filename}")

converter_cnn_classifier = tf.lite.TFLiteConverter.from_keras_model(cnn_model)
tflite_model_cnn_classifier = converter_cnn_classifier.convert()

cnn_tflite_filename = os.path.join(
    models_dir_combined, 'cnn_classifier_model.tflite')
with open(cnn_tflite_filename, 'wb') as f:
    f.write(tflite_model_cnn_classifier)
print(f"Original CNN classifier TFLite model saved to: {cnn_tflite_filename}")

# --- Save the FaceNet-style embedding model and convert to TFLite ---
print("\nSaving the FaceNet-style embedding model and converting to TFLite...")

facenet_embedding_keras_filename = os.path.join(
    models_dir_combined, 'facenet_embedding_network.keras')
facenet_embedding_network.save(facenet_embedding_keras_filename)
print(f"FaceNet embedding network Keras model saved to: {facenet_embedding_keras_filename}")

converter_facenet_embed = tf.lite.TFLiteConverter.from_keras_model(
    facenet_embedding_network)
tflite_model_facenet_embed = converter_facenet_embed.convert()

facenet_embedding_tflite_filename = os.path.join(
    models_dir_combined, 'facenet_embedding_network.tflite')
with open(facenet_embedding_tflite_filename, 'wb') as f:
    f.write(tflite_model_facenet_embed)
print(f"FaceNet embedding network TFLite model saved to: {facenet_embedding_tflite_filename}")
```

Figure 4.38 : Code snippets for training model saving and deep learning model to TensorFlow Lite format conversion

4.7 Summary

This chapter details the experimental path of the Smart Attendance Face Recognition module, which gradually transitions from traditional machine learning to deep learning, eventually focusing on open set recognition.

The experiments first emphasize and implement the criticality of dataset preprocessing and enhancement. Subsequently, conventional classifiers combining PCA/LDA are evaluated, revealing their inherent limitations in handling high-dimensional face data and open-set problems.

The key finding is that Convolutional Neural Networks (CNNs) demonstrate significant superiority in face classification tasks, providing strong evidence of the powerful feature extraction capabilities of deep learning. However, despite the superior performance of CNN classifiers in closed-set tasks, the fundamental limitation that they cannot be directly adapted in open-set scenarios (e.g., new user registration) comes to the fore.

With this in mind, this research turns to a deep metric learning network based on a CNN architecture. The network successfully learns discriminative face embeddings by triple loss training. Experimental results show that the average intra-class distance is much smaller than the average inter-class distance (0.5521 ± 0.2868 vs. 1.3544 ± 0.2182), which effectively realizes the separation of different identities and meets the core requirements of an open-set attendance system. Despite the overlap in the tails of the distance distribution, which is mainly attributed to the complexity of the dataset, the limitations of the triple mining strategy, and the inherent characteristics of the backbone network; future research will be devoted to further improving the model robustness by optimizing the triple mining, exploring the advanced network, and utilizing the large-scale dataset.

In summary, this chapter verifies the excellence of CNN in face feature extraction through comparative experiments, and on this basis, it proves that the deep metric learning network based on CNN architecture can effectively overcome the limitations of traditional classifiers in open-set scenarios, which provides a solid technical support for the construction of a scalable, efficient, and robust intelligent attendance face recognition system.

Chapter 5: System Implementation and Testing

5.1 Introduction

The purpose of this chapter is to elaborate the concrete implementation and comprehensive testing process of a deep learning-based face recognition attendance application. Following the in-depth design and planning of the system requirements, architecture and functional modules in Chapter 3, and the successful deployment of the trained models in Chapter 4 into real applications, the core task of this chapter is to transform these abstract design blueprints into practically runnable Android applications. To this end, we focus on implementing various system features and conducting comprehensive systematic validation to ensure that they meet the expected functionality and performance requirements.

This chapter is organized as follows: firstly, we introduce the hardware and software environment of the system; secondly, we analyze the design concepts and implementation details of each module; and finally, we show the carefully planned testing process and results.

5.2 System Implementation Environment

The purpose of this section is to detail the hardware platform and software tool stack used in the development and testing phases of this deep learning-based face recognition attendance application to ensure the reproducibility and professionalism of the research as well as to provide readers with a clear technical background.

5.2.1 Hardware Environment

The hardware configurations for the development and testing phases of the system are as follows: the development mainframe is equipped with Windows 11 Professional operating system, Intel Core i5-1135G7 processor, 16GB of DDR4 memory (RAM), and 1TB of NVMe SSD as storage. This configuration provides sufficient computing resources and responsiveness for the development and testing of deep learning models, as well as the compilation and running of Android applications.

To clearly outline the hardware platform used for the development and testing of this system, the main configurations are summarized in Figure 5.1.

Category	Sub-category/Component	Specific Configuration/Description
Development Host	OS	Windows 11 Professional
	Processor	Intel Core i5-1135G7
	RAM	16GB DDR4
	Storage	1TB NVMe SSD

Figure 5.1 : System Hardware Environment Summary

5.2.2 Software Environment

The software tool stack used in the development of the system is described in detail here.

Development & Build Tools

The core development and debugging platform of the system is Android Studio, which provides a complete tool chain including code editor, debugger, performance analyzer, and simulator, which significantly improves the development efficiency of Android applications.

Kotlin 2.1.20 was chosen as the main development language, which lays the foundation for building high-quality applications with its concise syntax, built-in null security features, and high interoperability with Java.

For Android SDK configuration, minSdkVersion is set to 26 (Android 8.0 Oreo), and targetSdkVersion and compileSdkVersion are both set to 34 (Android 14), which ensures the compatibility of the application with the mainstream devices and the optimal utilization of the latest API features.

Core Functionality Frameworks & Libraries

1. Face Detection: The Face Detection API, integrated with Google ML Kit, is used to efficiently and accurately localize face regions in images and video frames, and to detect key facial features and head pose. Its lightweight and high-accuracy features are the core reasons for choosing this library.
2. Deep Learning Inference: TensorFlow Lite (TFLite), a lightweight cross-platform deep learning inference framework optimized for mobile and embedded devices, is used to efficiently execute pre-trained face recognition models on the device side to achieve low-latency local face feature vector extraction and comparison.
3. Recognition Logic Encapsulation: TFLiteFaceRecognition module is developed independently to encapsulate TFLite model loading, image preprocessing (e.g., cropping, normalization), face feature extraction, and feature matching logic based on cosine similarity or Euclidean distance, which is the key component for the system to realize the core face recognition functions.
4. User Interface (UI) Construction: Jetpack Compose was chosen as a modern declarative UI toolkit from Google, which greatly simplifies the native Android UI development process and improves code readability, maintainability, and development efficiency.

Data Management & Backend Services

1. Network communication and data synchronization: The system relies on Firebase SDK for Android, which is fully integrated with Google Firebase back-end services. Firebase Firestore, as a NoSQL document database, is mainly used to store structured data such as basic user information, attendance tasks and check-in records, and supports real-time data synchronization.

2. Local data persistence: A custom DB Helper module based on SQLite is implemented to securely store the basic information of registered users and their corresponding face feature vectors locally on the device. The local storage is designed to accelerate the face matching process and improve the system response speed and robustness in offline environment.

Auxiliary Development Tools

1. Asynchronous Programming: Use Kotlin Coroutines to manage background tasks and asynchronous operations (e.g., network requests, database I/O, complex calculations) to ensure that the main thread is not blocked and to maintain a smooth user experience.
2. Version Control: Git is used as the core tool, and GitHub is used for code versioning and collaborative development to ensure the standardization and traceability of the development process.

In order to clearly outline the software tool stack used for the development and testing of this system, the main configurations are summarized in Figure 5.2.

Category	Sub-category/Component	Specific Configuration/Version/Description
Development & Build Tools	IDE	Android Studio
	Programming Language	Kotlin 1.8.20
	Android SDK	minSdkVersion 26, target/compileSdkVersion 34
Core Functionality Frameworks & Libraries	Face Detection	Google ML Kit Face Detection API
	Deep Learning Inference	TensorFlow Lite (TFLite)
	Recognition Logic Encapsulation	TFLiteFaceRecognition (Custom Module)
	UI Construction	Jetpack Compose
Data Management & Backend Services	Network Comm./Data Sync	Firebase SDK for Android
	Backend DB	Firebase Firestore (NoSQL)
	Local Data Persistence	SQLite (Custom DB Helper)
Auxiliary Development Tools	Asynchronous Programming	Kotlin Coroutines
	Version Control	Git + GitHub

Figure 5.2 : System Software Environment Summary

5.3 System Function Implementation

This section elaborate the specific function implementation details of this Android face recognition attendance system, including its MVVM architecture design and the implementation of each core module.

5.3.1 System Architecture Design: Application of MVVM Patterns

This system adopts MVVM (Model-View-ViewModel) architecture pattern, combining with the characteristics of Android development, and builds a layered structure of View - ViewModel - Repository - DataSource. This architecture aims to realize the clear separation of UI logic and business logic, effectively improve the testability, maintainability and scalability of the code, so as to lay the foundation for building a stable and efficient Android face recognition attendance system.

View is responsible for rendering the user interface and capturing user interaction events. In this project, the View layer consists of Jetpack Compose's Composable functions, such as LoginScreen, StudentCourseScreen and AttendanceProcessScreen. The view layer is only responsible for passing user actions to the ViewModel and observing the UI state exposed by the ViewModel to update the interface. It does not deal with business logic directly, maintaining its character as a dumb component and ensuring that the interface is loosely coupled with the business logic.

ViewModel (View Model Layer) serves as a bridge between View and Model, and assumes the responsibility of UI-related business logic and state management. It requests data

from the Repository and transforms the data into UI state that can be directly consumed by the View, such as StateFlow or LiveData. For example, LoginViewModel and StudentCourseViewModel manage the data and logic of the corresponding interfaces. ViewModel does not directly hold a reference to the View, but communicates with the View through data binding and event callback mechanisms, which effectively avoids memory leakage and ensures the independence of UI logic and life cycle.

The Model layer is further subdivided into the Repository and DataSource layers in this architecture.

Repository is responsible for unified scheduling and management of data, shielding the complexity of the underlying data sources. It obtains data from one or more Data Sources according to business requirements and provides a simple data access interface to the ViewModel. The Repository abstracts the access logic for different types of data and ensures that the ViewModel doesn't have to care about where the data comes from.

DataSource represents the specific underlying data source. In this project, the data sources include remote servers (Firestore via Firebase SDK) and local databases (SQLite). In this system, for the core functionality of face recognition, the specialized data sources also include the TFLite Model-based face recognition module (TFLiteFaceRecognition) and the Camera and Machine Learning Kit (CameraX/MLKit). These specific data sources provide the system with image capture and face feature extraction capabilities. The Repository coordinate these different types of data sources to meet the data requirements of the upper layer business logic.

Figure 5.3 shows an overview of the MVVM architecture of this Deep Learning-based Face Recognition Time and Attendance Android application. This architecture is designed to improve the decoupling between modules and provide a clear structural support for the implementation of the core functions such as face recognition detailed in the subsequent chapters.

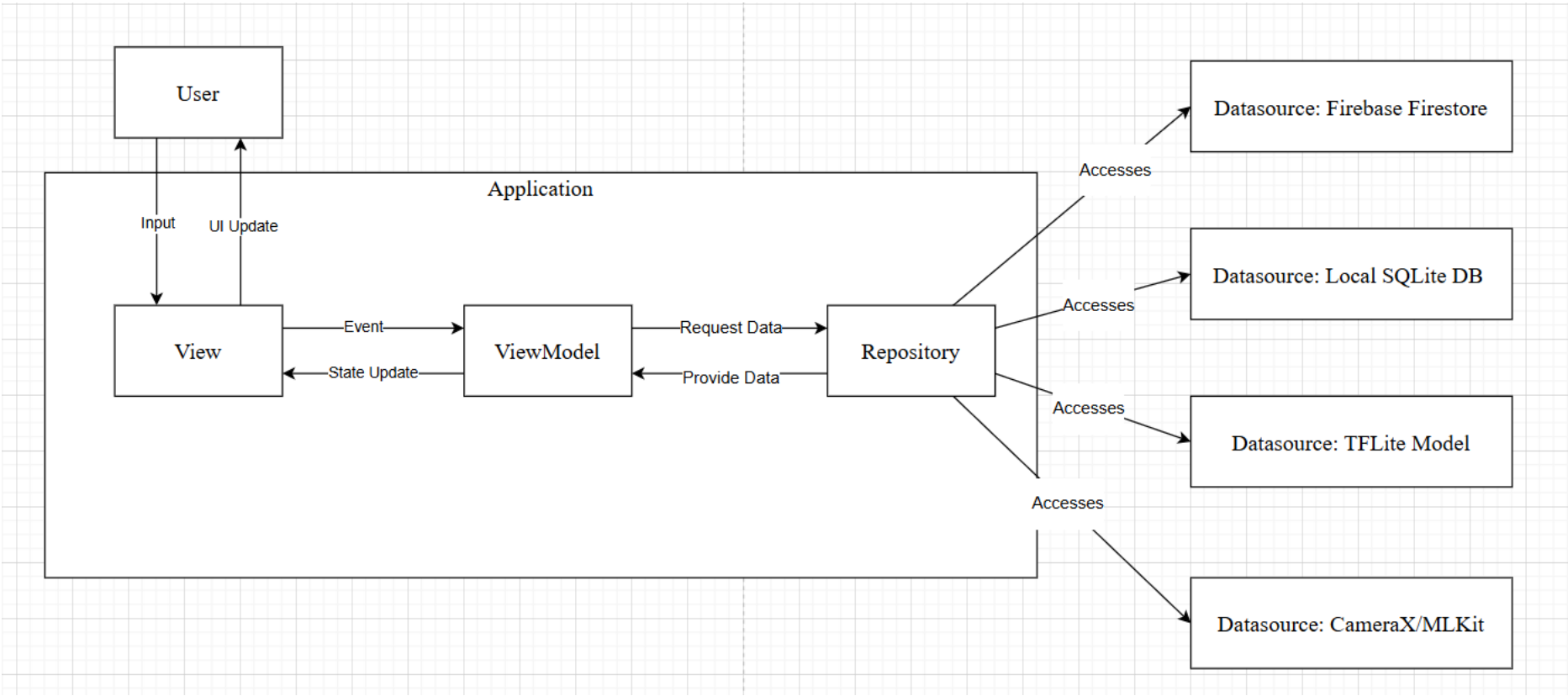


Figure 5.3 : MVVM Architecture for Deep Learning-based Face Recognition Attendance Android Application

5.3.2 System Module Development

1. Login Module

This section details the design and implementation of the login module, which serves as the primary entry point for users to access the system and aims to provide a secure and user-friendly authentication mechanism. The login module not only ensures secure access to user accounts but also provides customized functional entry points tailored for different roles, specifically for students and professors.

The login module's core function is to verify a user's identity and subsequently redirect the user to the main interface corresponding to their assigned role based on the verification result. The system supports two preset roles, which are student and professor. Students gain access to course information and attendance records upon successful login, while professors are authorized to view course management tools and attendance records. This meticulously designed role-based access control mechanism ensures the rational allocation of system functions and robust data security.

The login interface adopts a simple and intuitive design to optimize the user experience. The interface consists of several key interactive elements: a user name input box for the user to enter his/her registered user name, and a password input box for the user to enter the corresponding secure password. To enhance user experience and security, the password input box integrates a show/hide button represented by an “eye” icon that allows users to toggle the visible state of the password as needed. Additionally, the role selector provides different “Student” and “Professor” options, requiring the user to explicitly select their current identity to ensure that the system correctly matches roles and verifies permissions. Finally, after completing the information entry and role selection, the user clicks the “Login” button to submit the login request, which triggers the backend authentication process.

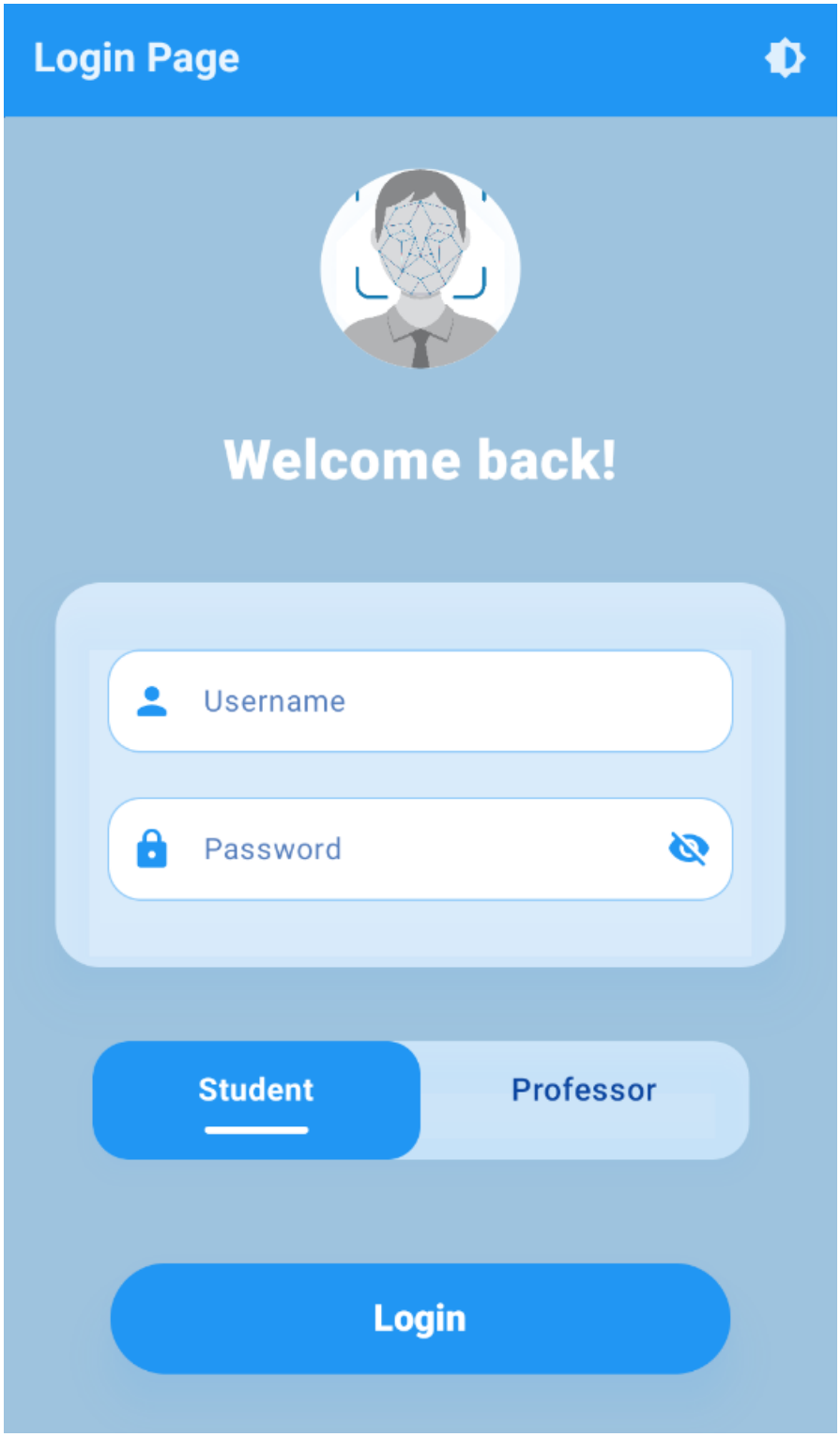


Figure 5.4 : UI for Login Module

The user login process follows a prescribed sequence of steps to ensure accurate authentication and system responsiveness. After the user launches the application, the system automatically loads and displays the login screen. The user then enters the account name in the designated “User Name” input box and the account password in the “Password” input box. Next, the user explicitly selects their identity, using the role selector to choose either “Student” or “Professor”. Finally, after clicking on the “Login” button, the system will obtain the user's credentials and selected role and transfer them to the backend for verification. The system will direct the user to the appropriate homepage or return an error message based on the backends’ validation results.

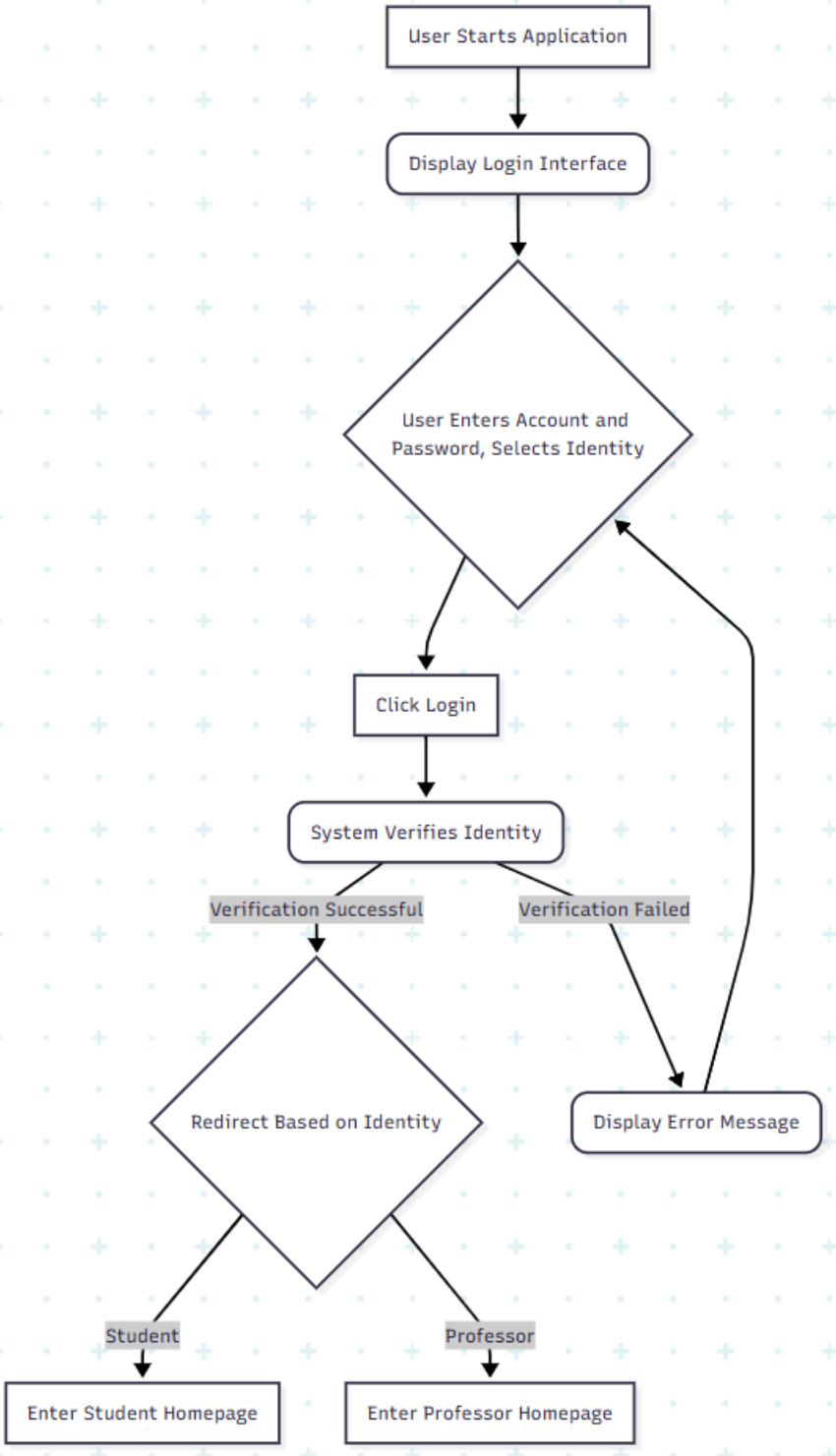


Figure 5.5 : Business Flowchart for Login Module

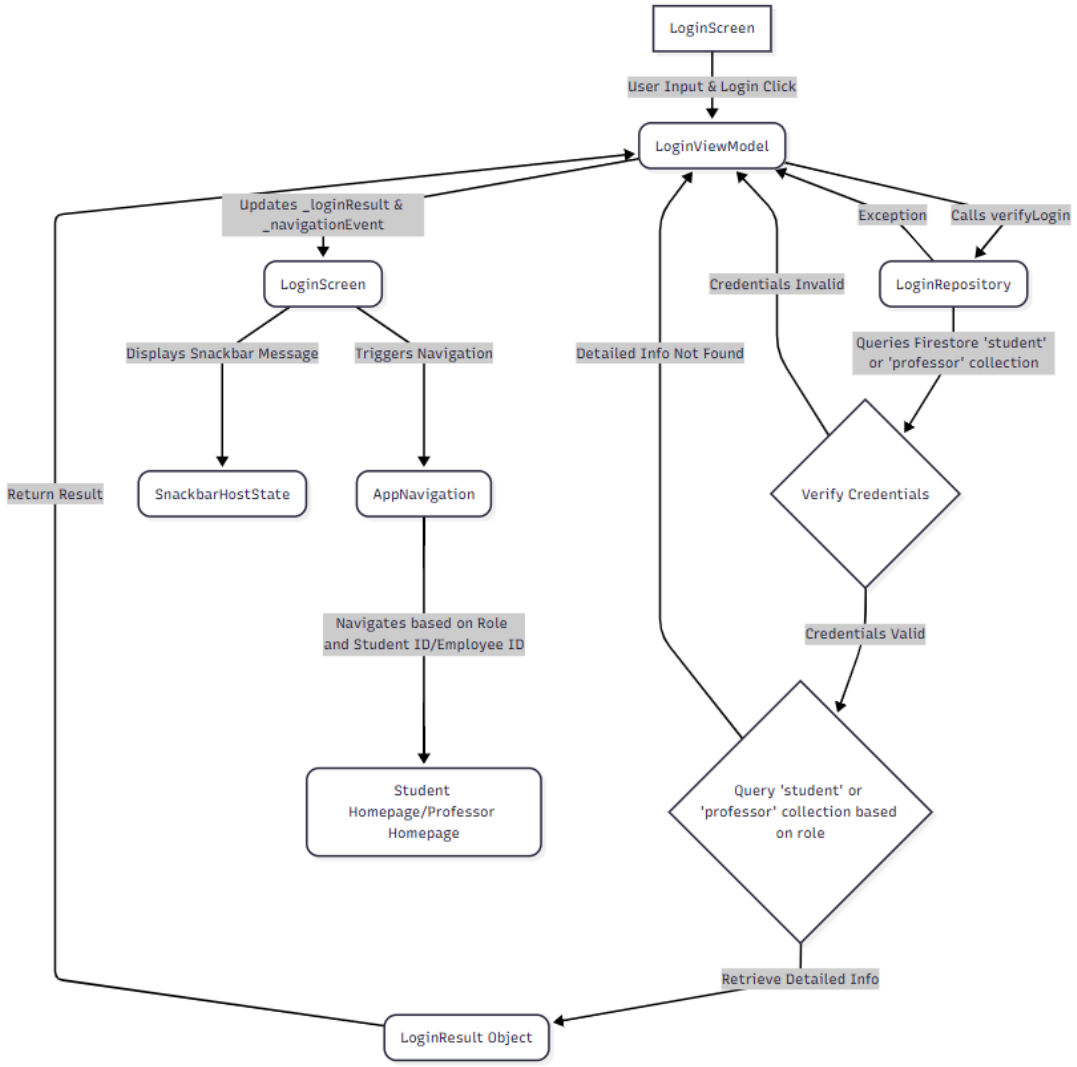


Figure 5.6 : Workflow for Login Module

2. Home Module

This section details the design and implementation of the homepage module, which is the main interactive interface for students or professors after logging in. It integrates a number of core functions such as course management, attendance query, personal information view, etc., aiming to provide an efficient and intuitive learning management experience.

The core function of the homepage module is to provide a centralized operation platform for users. On this page, students or professors can easily view the list of all registered courses. To improve search efficiency, the homepage also provides the ability to search for specific courses. Students or professors can enroll in new courses directly from the homepage and easily view and manage their personal information. The homepage is also the main portal for users to manage their course attendance information. For a student, he is able to track real-time attendance for each course. In addition, the system clearly display any pending course tasks to remind students to complete their attendance on time. To improve search efficiency, the homepage also provides the ability to search for specific courses. Students or professors can enroll in new courses directly from the homepage and easily view and manage their personal information. The homepage is also the main portal for users to manage course attendance information.

The interface design of the student homepage follows the principle of user-friendliness and is divided into several areas to ensure effective presentation of information and ease of operation. The search box is prominently placed to allow users to quickly locate enrolled courses. To accommodate different user preferences and ambient lighting, a theme switching button is integrated into the top bar, allowing users to switch between dark and light themes. In addition, a logout button provides a convenient point for users to safely exit the system. The main content area is the core information display area of the student homepage. At the top of

this area is a welcome message containing the student's name, enhancing the personalized experience. Below this is a list of enrolled courses in the form of cards that clearly display all the courses the student is enrolled in. Each card succinctly displays the course number, title, and attendance, as well as an eye-catching reminder of pending check-ins, indicated by a warning icon if such a task exists. The “Add Course” floating button in the bottom right corner of the interface allows users to quickly initiate the enrollment process for a new course. The bottom navigation bar provides easy global navigation, including a “Home” button that points to the current page and a “Profile” button that can be clicked to access the personal information viewing and editing page.

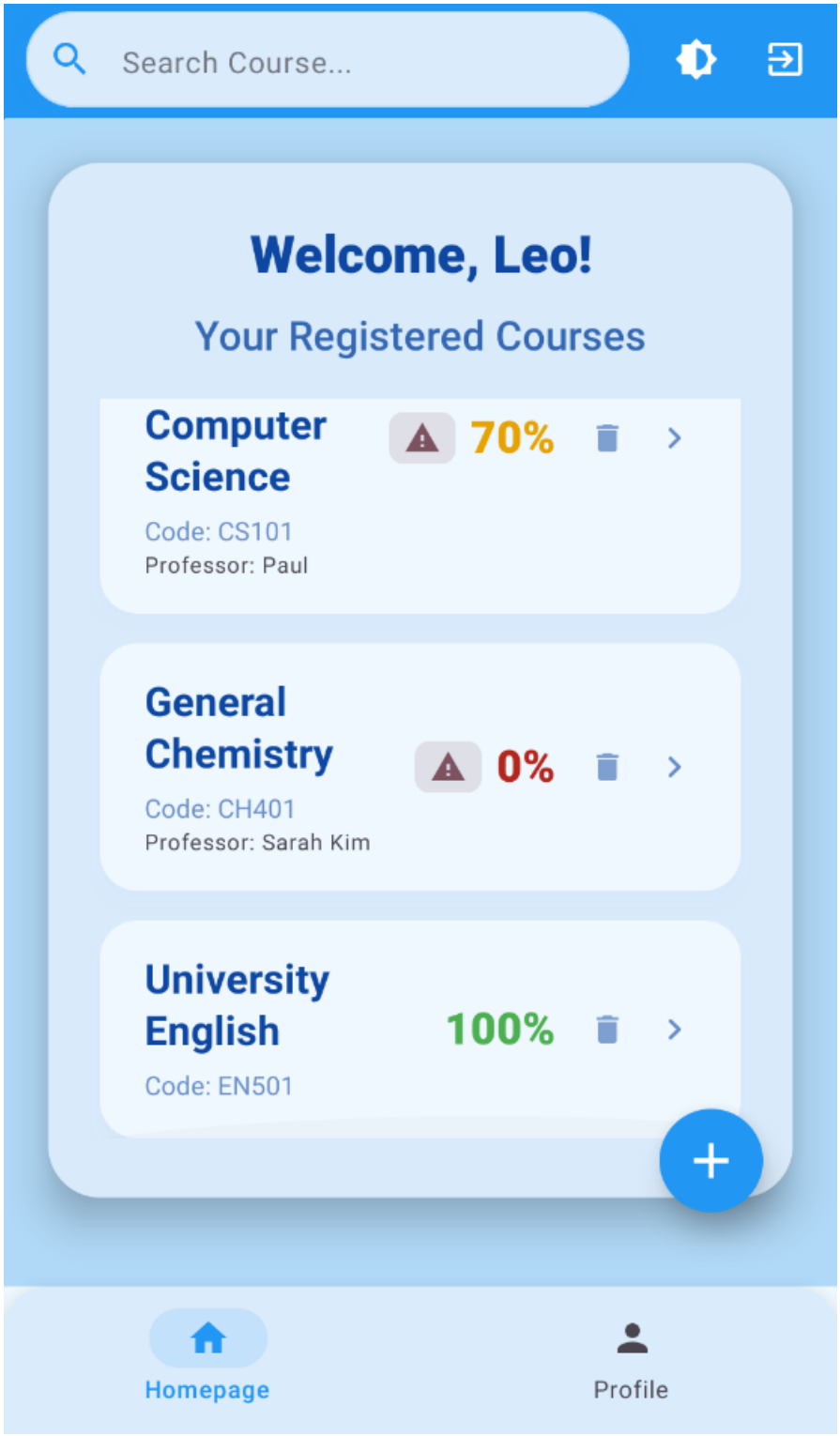


Figure 5.7 : UI for Student Home Module

The interface of the Professor's Home Page has been carefully designed to balance functionality and user experience and includes several key areas. A comprehensive search box at the top allows professors to quickly retrieve enrolled courses. To accommodate different user visual preferences, a theme toggle button allows professors to switch between dark and light themes. In addition, a “Logout” button makes it easy to safely exit the system. The main content area is the centerpiece of the professor's homepage information display. At the top of this area is a personalized welcome message containing the professor's name, enhancing the user experience. Below this is a comprehensive list of enrolled courses, clearly displaying all courses for which the professor is responsible, with each course card succinctly displaying the course number and title. In the lower right corner of the interface, a floating “Add Course” button provides a quick entry point for professors to register for new courses. The bottom navigation bar provides easy global navigation, including a “ Homepage” button that points to the current page and a “ Profile” button for viewing and editing profiles.

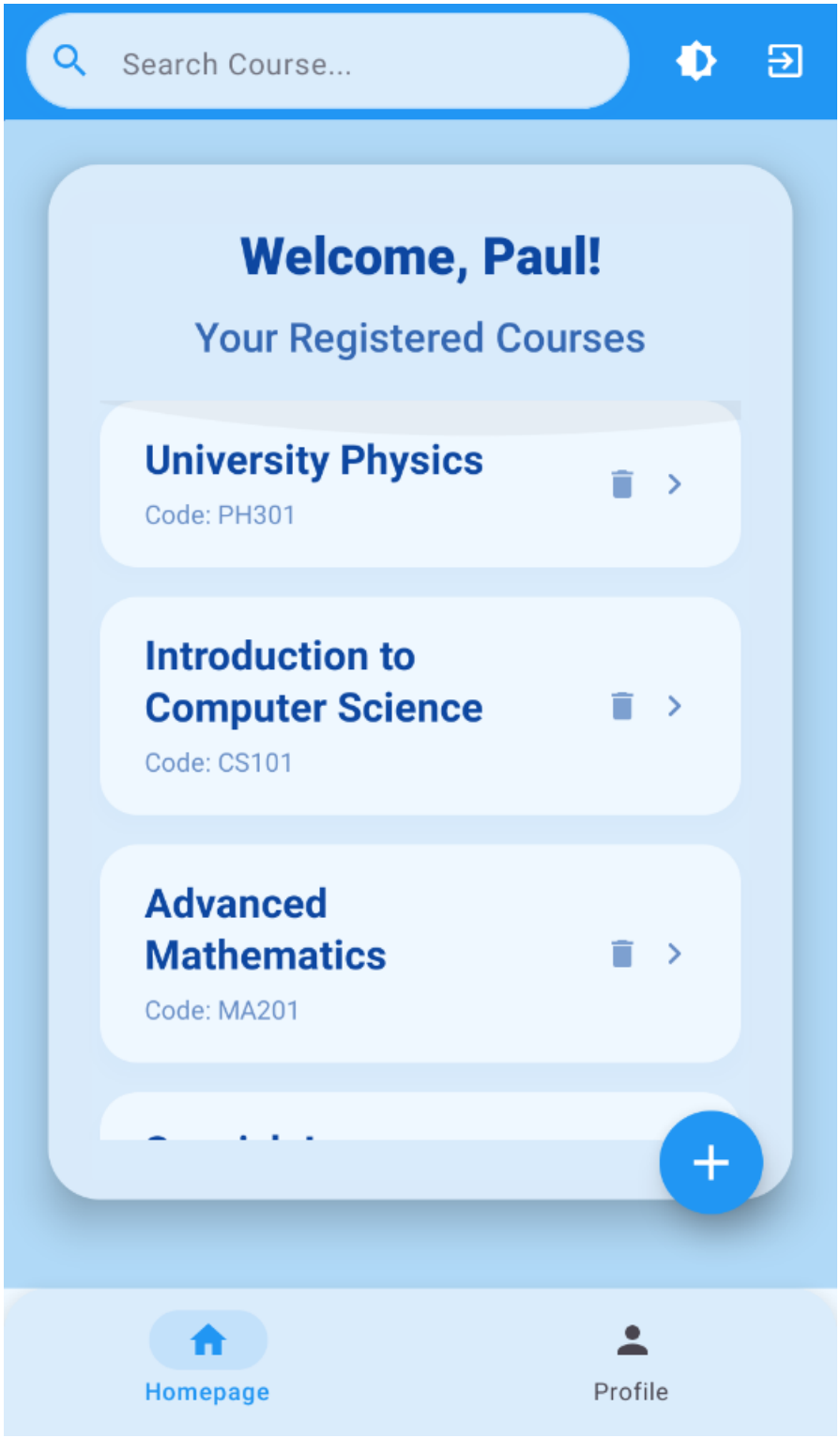


Figure 5.8 : UI of Professor Home Module

The Home module is the main interface for users, providing the core functions of course management and personal settings. Upon successful login, the system automatically displays all registered courses. Users can click on any course card to view detailed attendance information. For course search, users can use the search box at the top to filter courses by course number or name in real time; clearing the search box redisplay all courses. The module also supports enrollment in new courses via the "+" button in the lower right corner. The user enters the course number in the pop-up window and clicks "Register". The system provides immediate feedback such as "Course registration successful" or reason for failure (e.g. 'Course does not exist', "Already registered"). Withdrawing a course is handled by selecting a course from the list; the system asks for confirmation before updating the list of courses and displaying the result. In addition, users can manage their personal information by clicking on the 'Profile' button in the bottom navigation bar.

For student users, each course card not only display the course number and name, but will also be color-coded to reflect the attendance status: green for over 80% (good), orange for 60-80% (moderate), and red for less than 60% (poor). At the same time, a warning icon appears on the course card that needs to be signed in to remind the student to sign in.

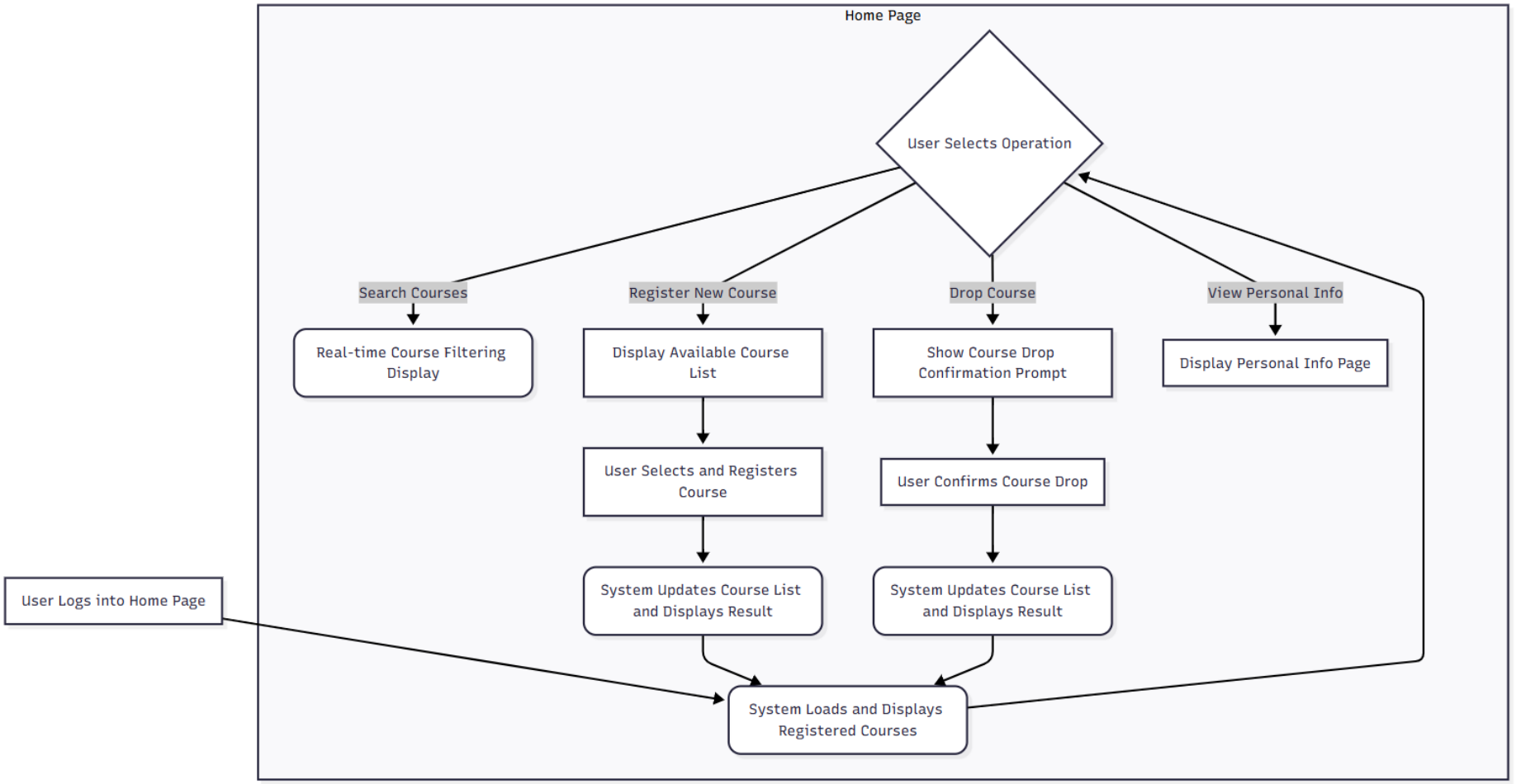


Figure 5.9 : Business Flowchart of Home Module

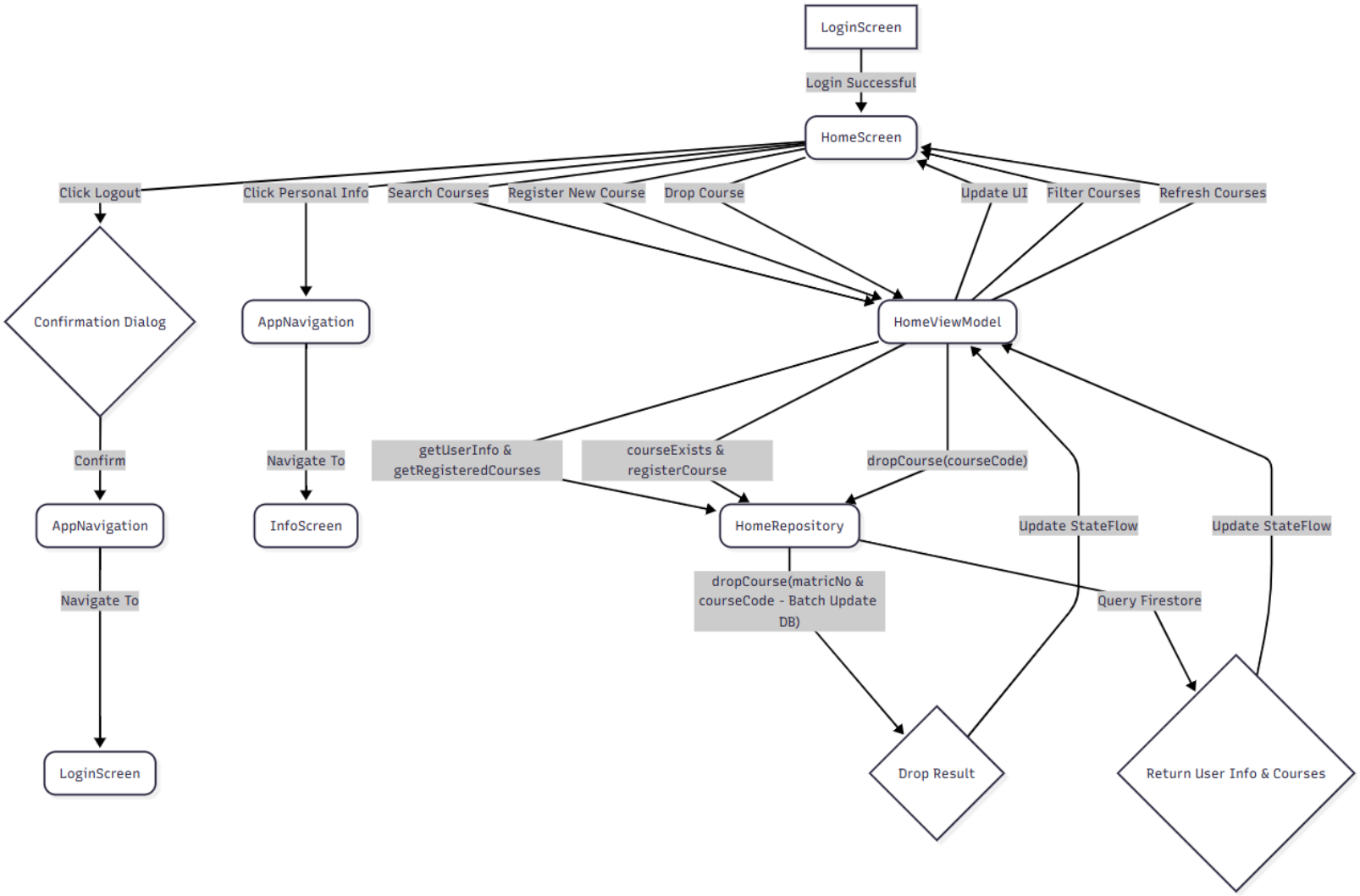


Figure 5.10 : Workflow of Home Module

3. Personal Information Module

This section details the design and implementation of the Personal Information module, which is designed to provide users with a centralized interface to view and manage their personal information. The module ensures the accuracy of user data and facilitates the smooth operation of the attendance recognition system.

The core functionality of the Personal Information module is to allow users to view their basic personal information, including name, matric number, and e-mail address.

Additionally, the module provides student users with the critical ability to upload face photos, a key part of the attendance recognition process. To enhance the user experience, the module is also equipped with a theme switching feature and a convenient way to return to the homepage.

The top bar of the information page is the navigation and function control area of the page. The “Back” button on the left side allows the user to easily return to the previous page, while the “Theme Switch” button on the right side allows the user to quickly switch between dark and light themes to suit different preferences. The main content area is the centerpiece of the personal information display.

For student users, this is followed by a separate “Upload Photo” card, specifically designed to direct the student user to upload a face photo on which the Attendance Recognition System will base attendance records. The bottom navigation bar provides easy global navigation, including a “ Homepage” button that takes user directly to the home page and a “ Profile” button that displays the current page.

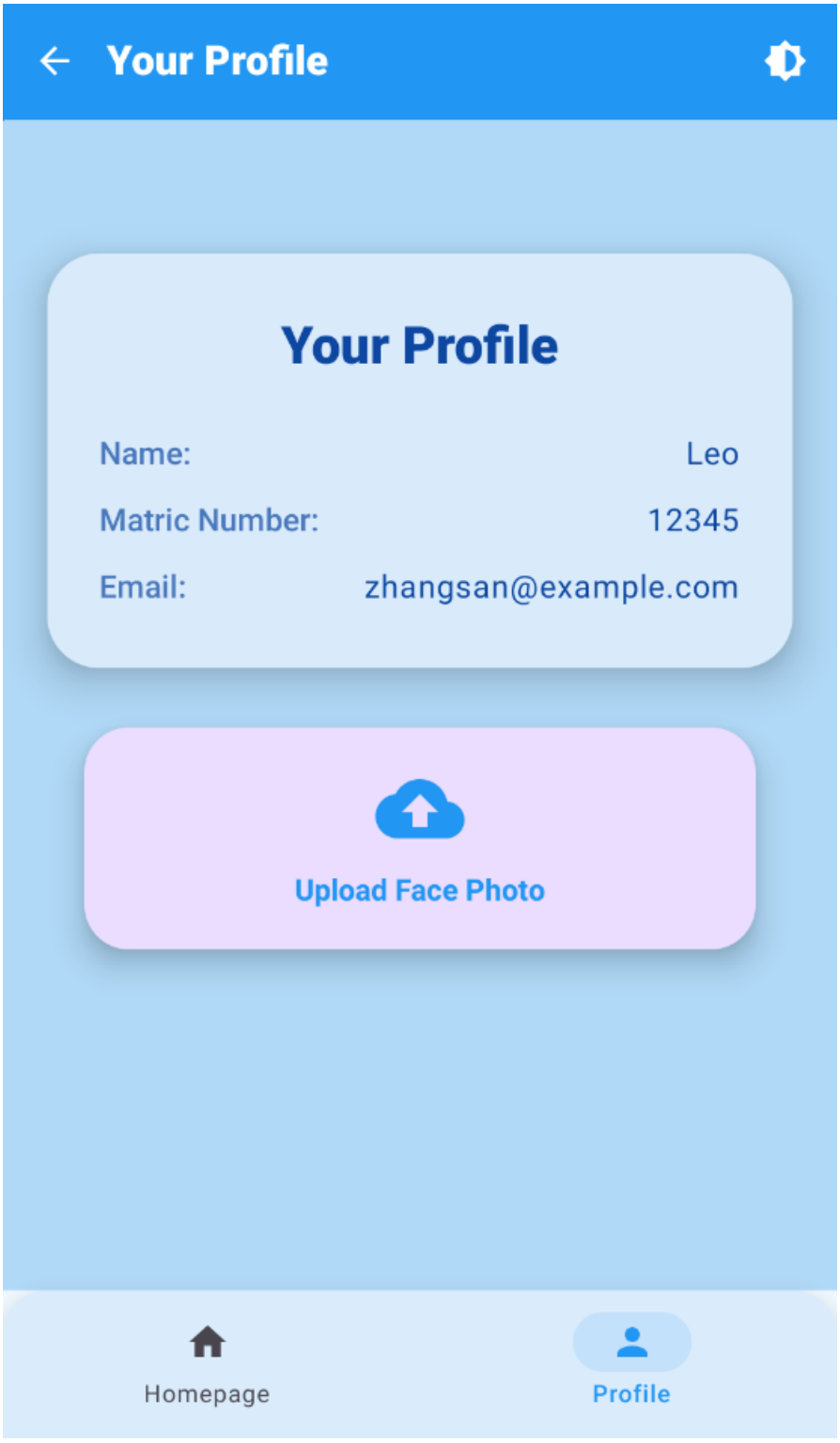


Figure 5.11 : UI of Student Information Module

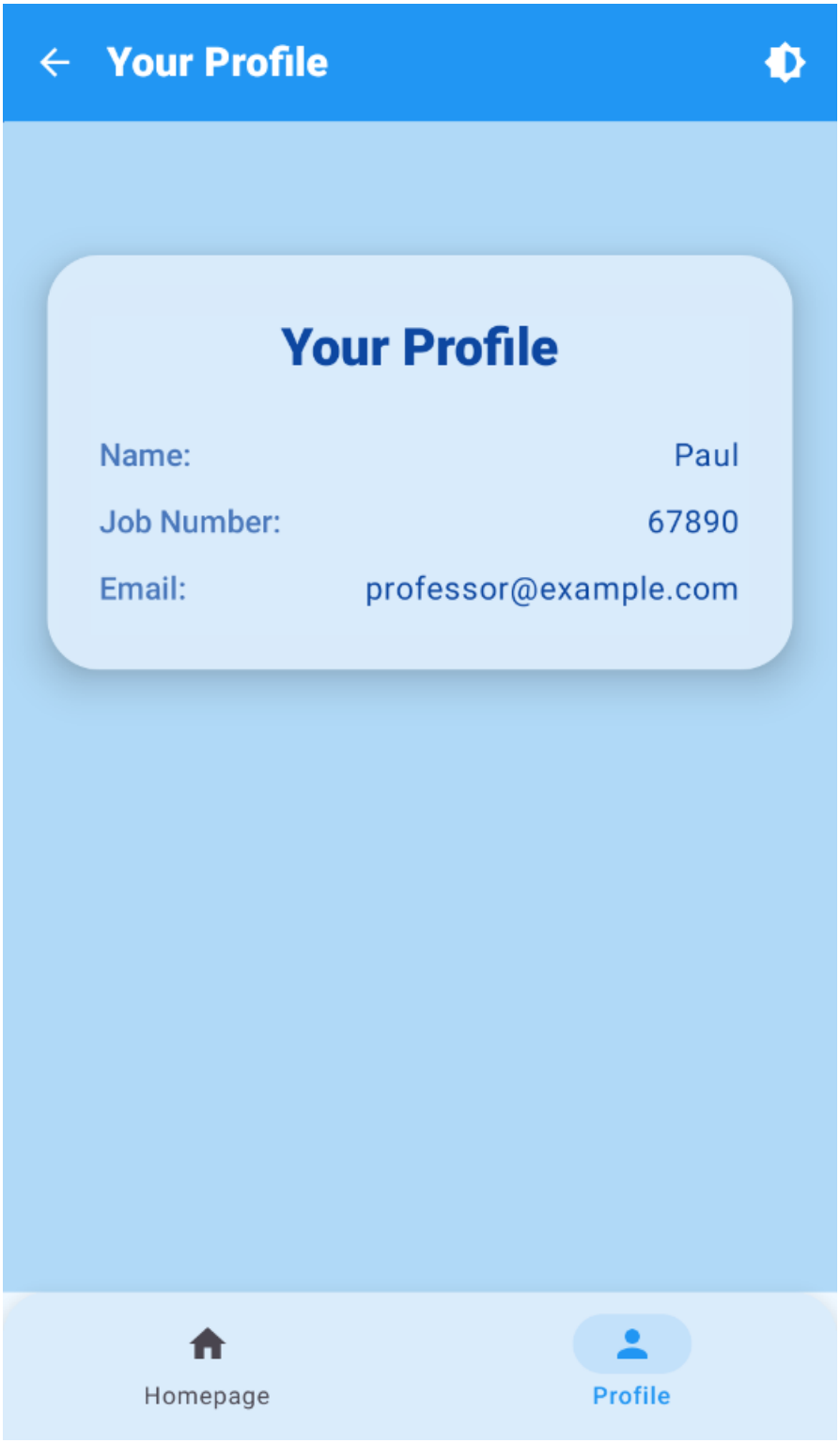


Figure 5.12 : UI of Professor Information Module

When a user clicks on a profile, the system first displays the user's information and then loads detailed user data. A condition is then evaluated to determine if the user is a student. If the user is a student, the system displays the Upload Photo button, and when clicked, the user is redirected to the photo upload page. If the user is not a student, the upload button will be hidden. Additionally, on the profile page, the user can click the Back button to go back to the previous page.

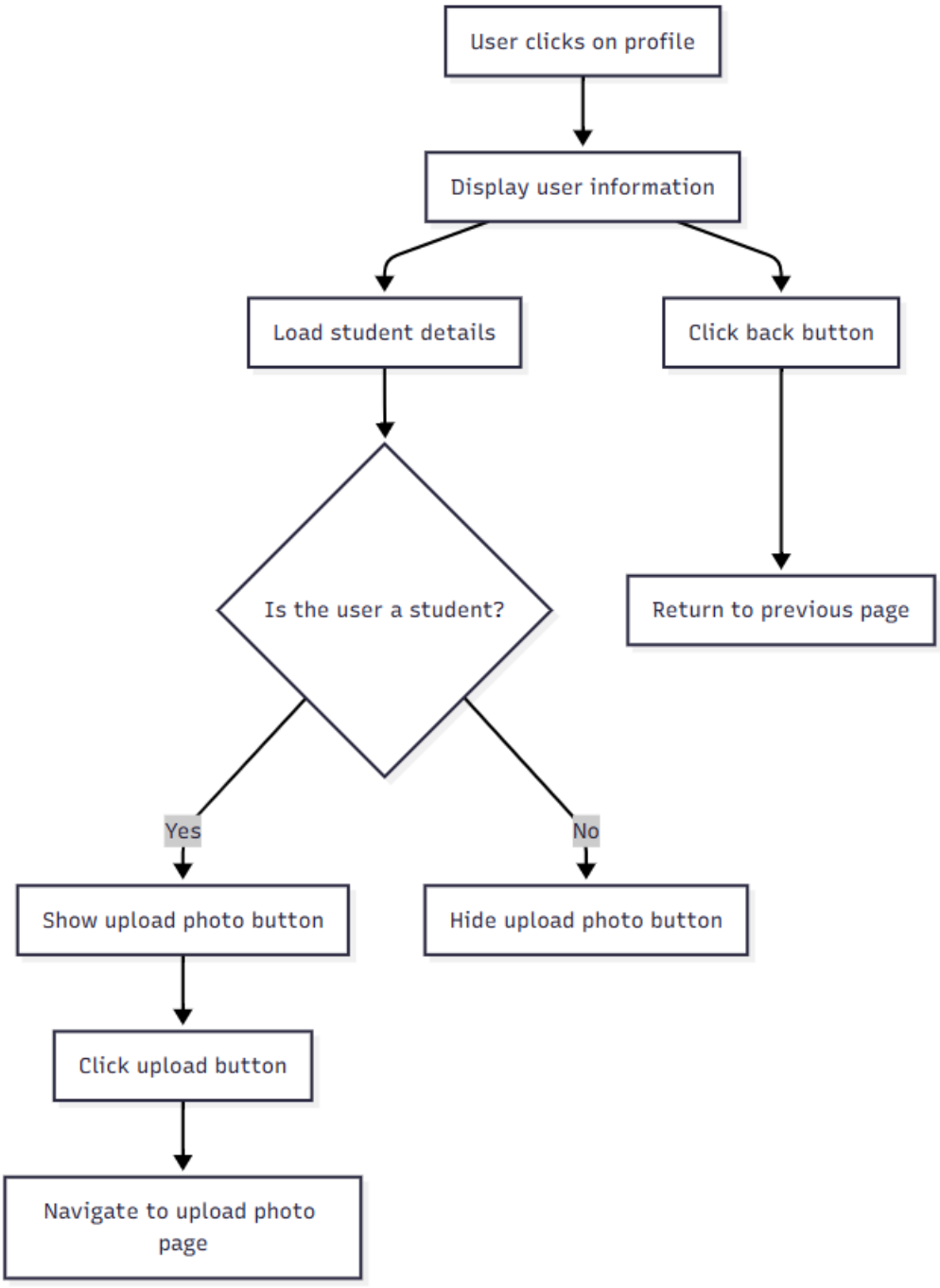


Figure 5.13 : Business Flowchart of Information Module

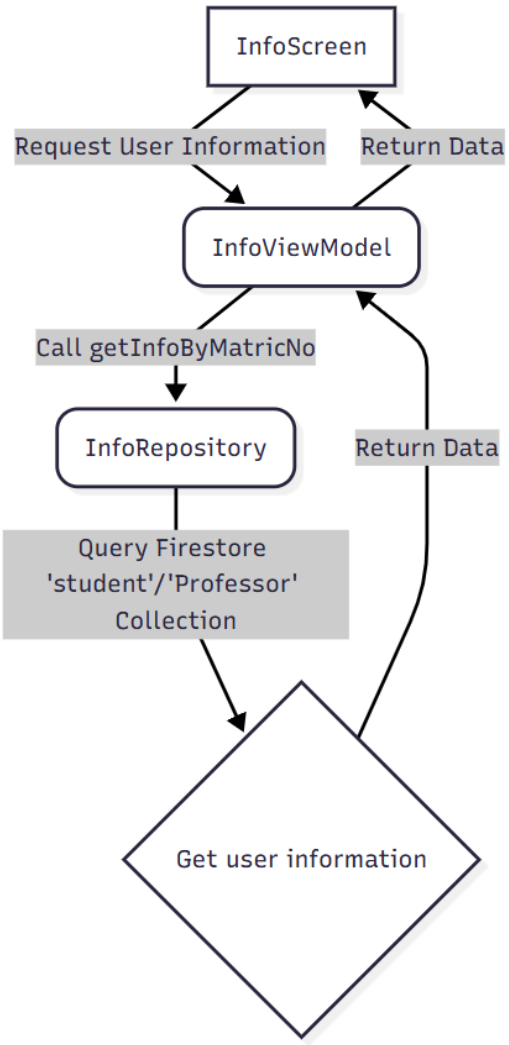


Figure 5.14 : Workflow of Information Module

4. Student Course Module

This section details the design and implementation of the Student Course Module, which serves as the core tool for students to manage and participate in course attendance. It is designed to provide an intuitive and functional interface enabling students to view course information, attendance tasks, and statistics, as well as to complete attendance sign-ins.

The Student Course Module's fundamental purpose is to provide comprehensive attendance management capabilities for students. Within this module, students can view basic course information and browse all relevant attendance tasks. The module further supports students in performing real-time attendance sign-ins and sign-outs and viewing their attendance statistics to fully understand their attendance status.

The course page interface is designed to provide a clear display of information and user-friendly operation and consists of several key areas. The top bar is the navigation and function control area of the page, prominently displaying the name of the course the user is currently viewing. A back button on the left side allows the user to easily return to the student home page, while a theme toggle button on the right side allows the user to quickly switch between dark and light themes to accommodate different visual preferences. The Attendance Task List is the core content area of the page, clearly displaying all attendance tasks in card format. Each task card includes the task serial number, start time, end time, check-in status and task status, allowing students to fully understand their attendance status. The bottom navigation bar facilitates easy global navigation and includes a “Statistics” button for viewing attendance statistics and an “Attendance” button for displaying a list of attendance tasks on the current page.

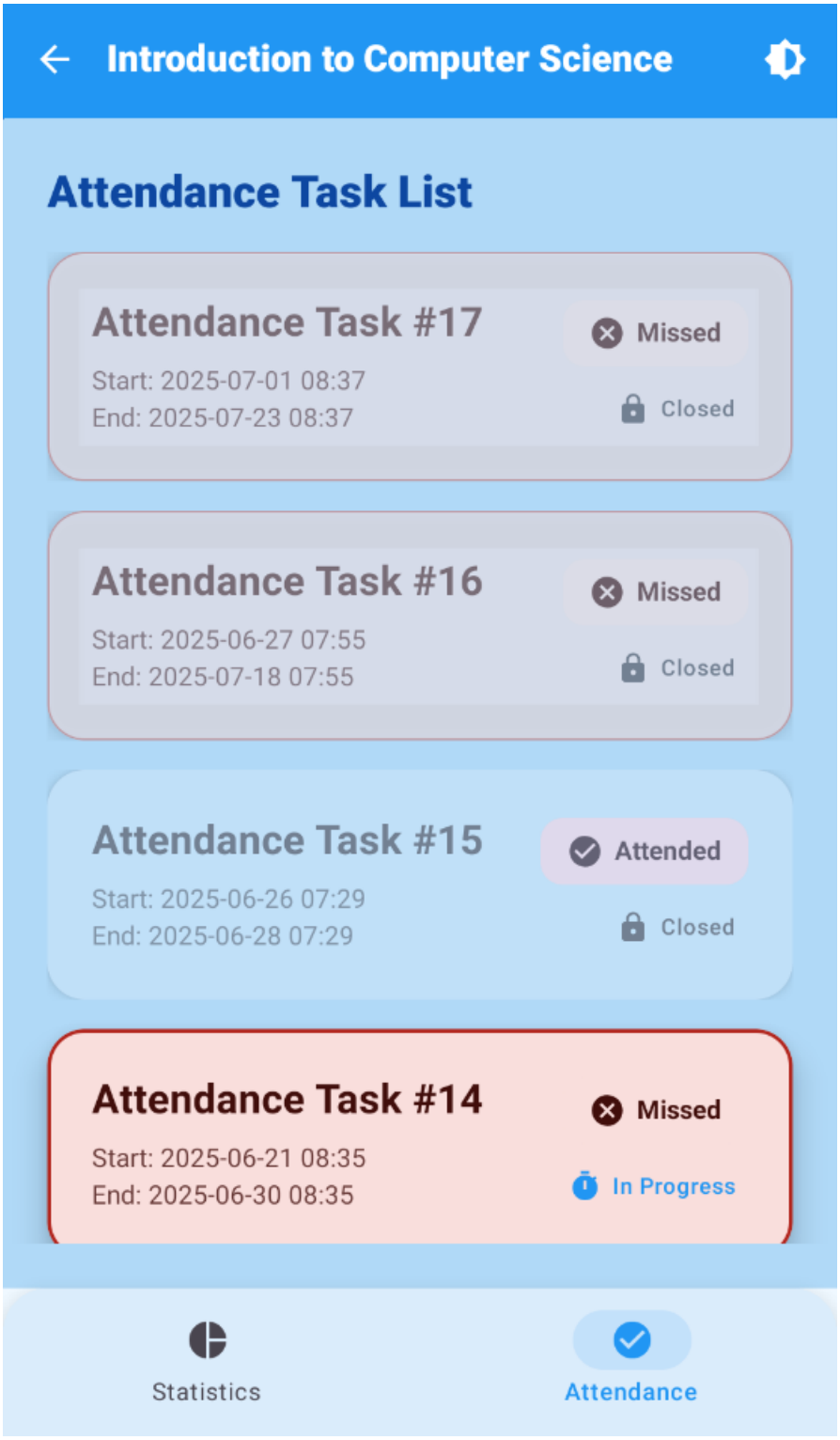


Figure 5.15 : UI of Student Course Module

The Student Courses module supports a variety of basic operations to ensure that students can effectively manage their attendance affairs. Upon entering the course page, the system automatically displays the basic information of the course. Attendance task list shows all the tasks clearly in order. Detailed information of each task, such as start time, end time, sign-in status, etc., is clear at a glance. Regarding the task status, “Attended” is indicated by a check mark label, which confirms that the task has been completed, while “Missed” is indicated by a cross label, which means that the task has not been completed yet. “In Progress” indicates that the task is currently available for check-in, while “Closed” indicates that the task has not yet started or has already been completed. In the case of attendance check-in, when an attendance task is in the “In Progress” state and its check-in status is “Missed”, the task card will be highlighted to remind the user to check in. When a student clicks on a task card that is in the “In Progress” state and its check-in status is “Missed”, the system will automatically redirect the user to the attendance page and the user will follow the instructions to complete the check-in process.

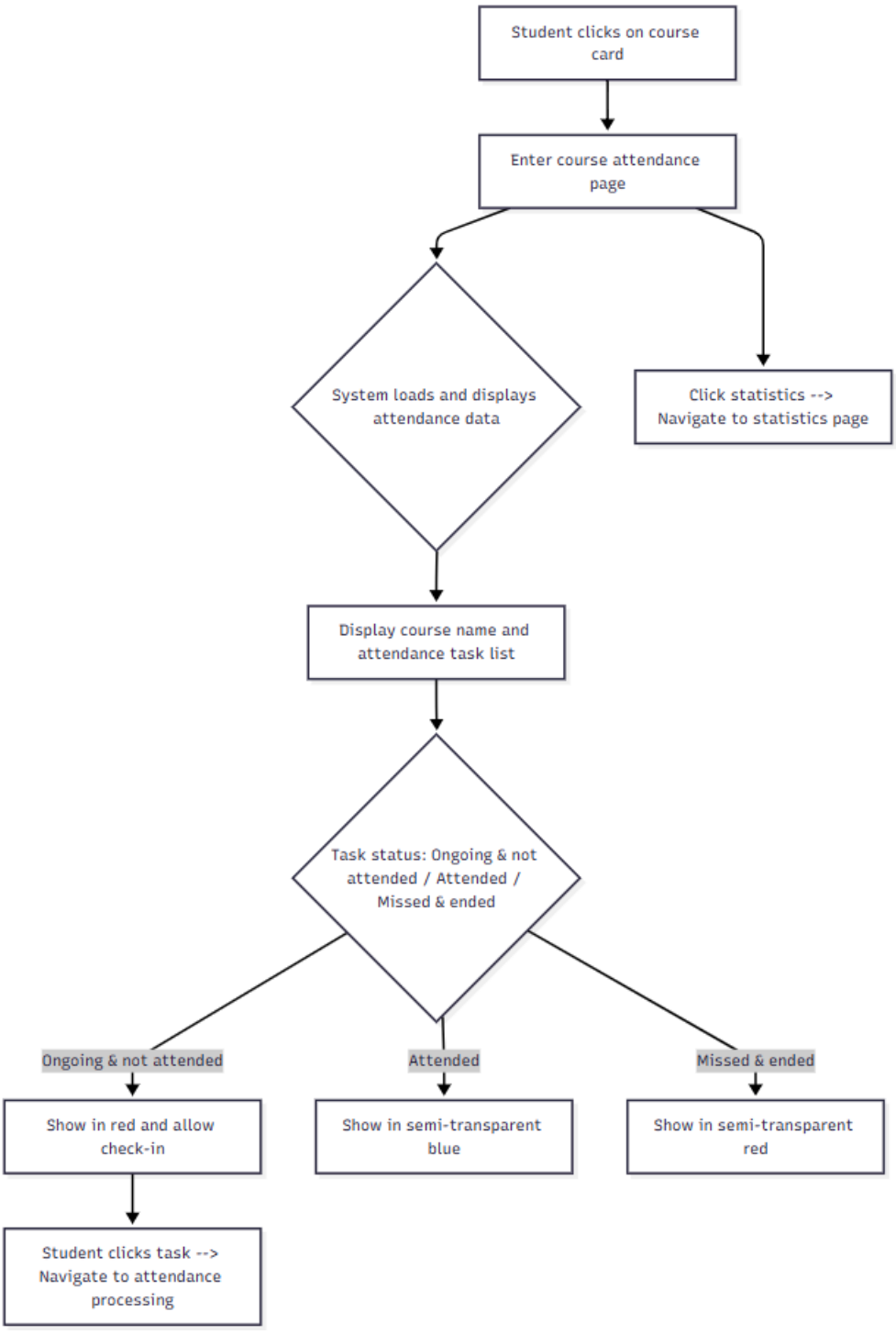


Figure 5.16 : Business Flowchart of Student Course Module

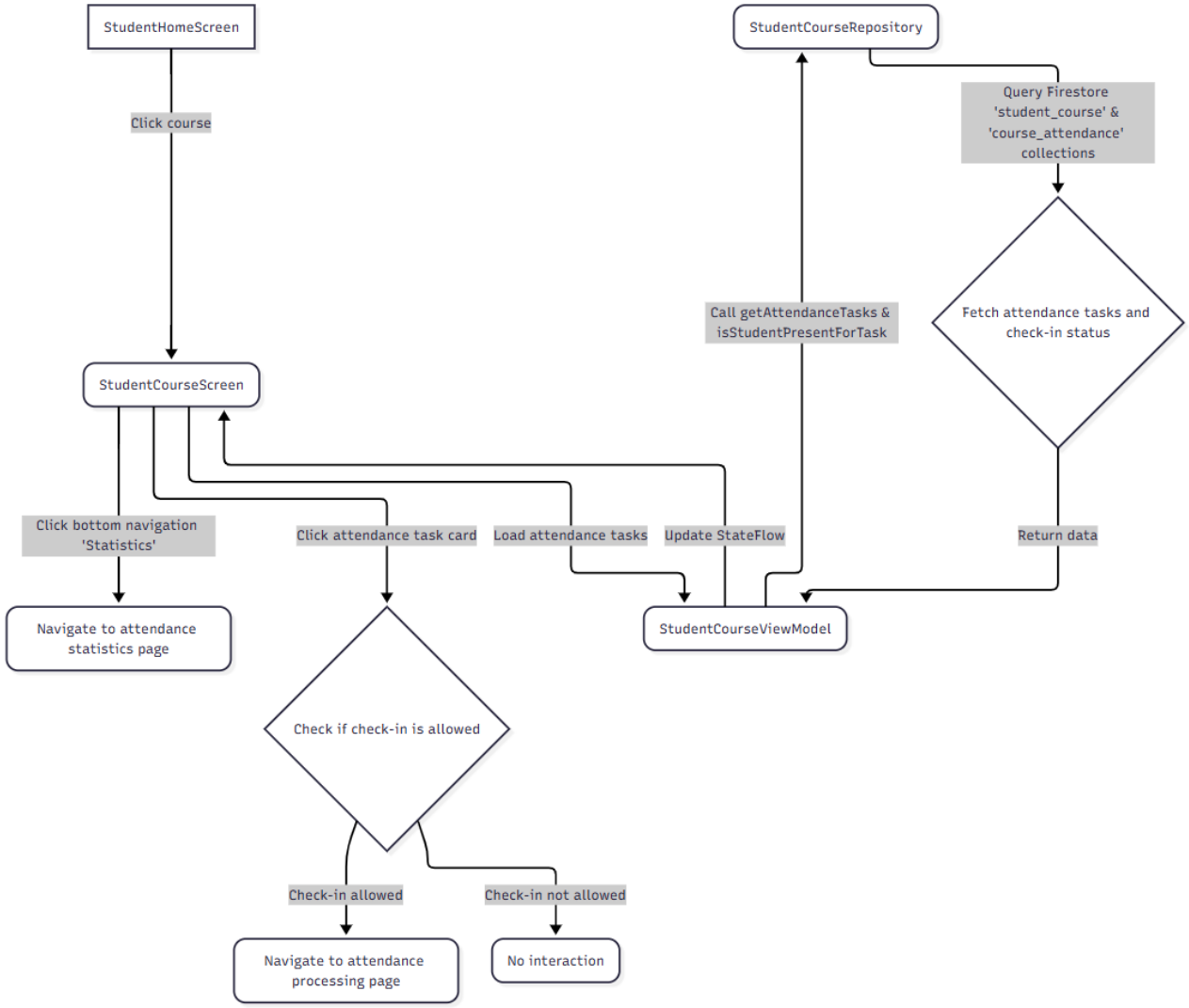


Figure 5.17 : Workflow of Student Course Module

5. Professor Course Module

This section details the design and implementation of the Professor Course Module, which serves as the core functional area enabling professors to manage course attendance. It aims to provide a comprehensive interface supporting professors in viewing basic course information, managing attendance tasks, reviewing attendance statistics, overseeing course members, and conducting real-time attendance sessions.

The Professor Course Module's core functionality is to offer centralized course attendance management capabilities for professors. This module enables a professor to view basic information about a course and efficiently manage attendance tasks, including adding new tasks and viewing details of existing ones. Furthermore, the module allows professors to review detailed attendance statistics, manage the list of course members, and perform real-time attendance.

The top bar of the course page interface is the navigation and functional control area of the page, prominently displaying the course name to identify the current course. The “Back” button on the left side allows professors to easily return to the ‘Professor Home’ page, while the “Theme Toggle” button on the right side allows for quick switching between dark and light themes. The course information card immediately below the top bar succinctly displays the professor's number and the course code for the current course, ensuring that the user understands the contextual information. The Attendance Task List is the core content area of the page, clearly displaying all attendance tasks in a card format. Each task card includes the task serial number, ID, start time, end time, and attendance (shown as a percentage), giving professors a comprehensive view of their attendance tasks. The bottom navigation bar provides easy global navigation. The “Members” button is used to view the list of students in the course,

the ‘Statistics’ button is used to view attendance statistics, and the “Tasks” button indicates that the current page displays a list of attendance tasks.

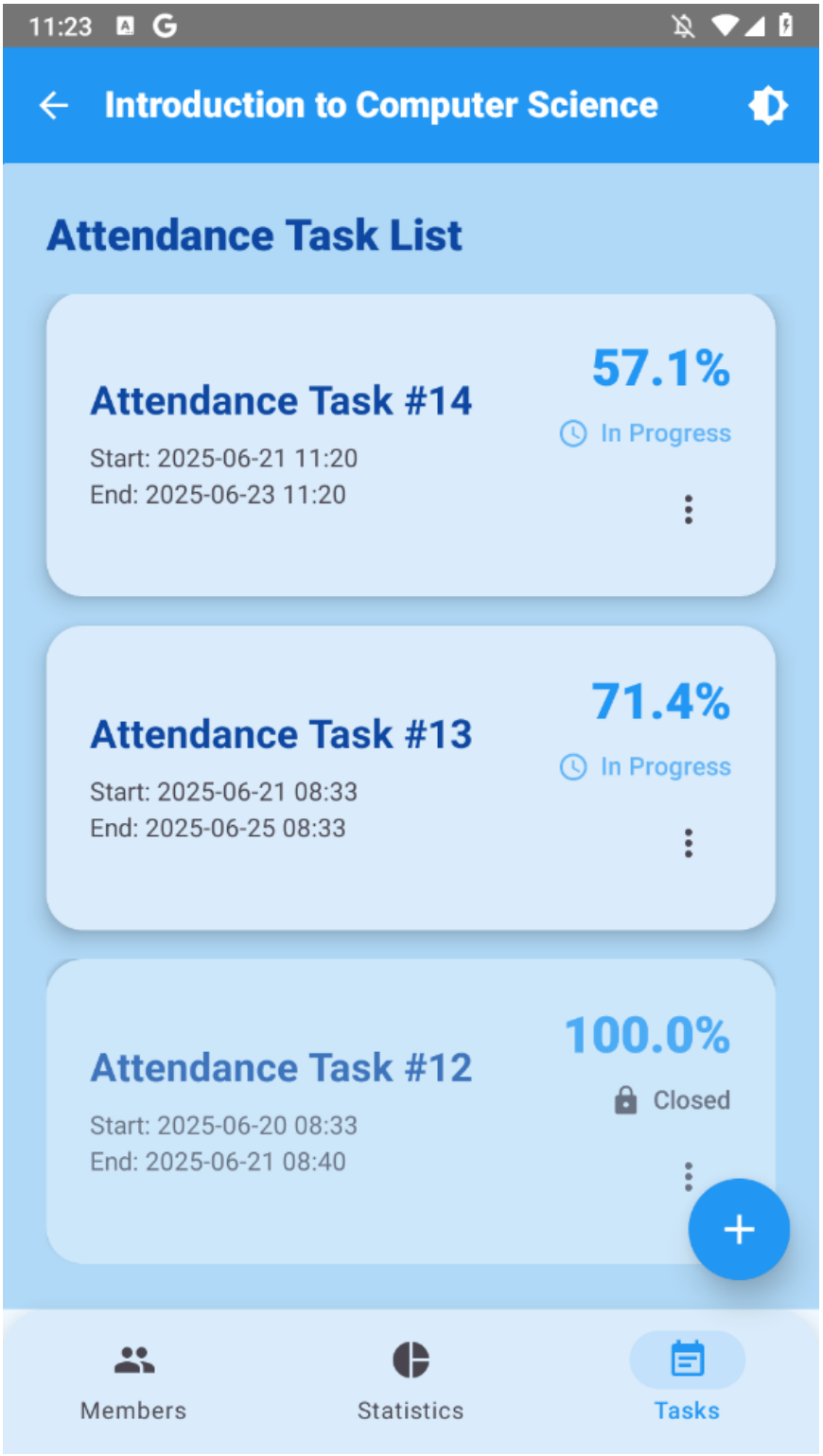


Figure 5.18 : UI of Professor Course Module

The Professor Courses module supports a variety of basic operations to ensure that professors can effectively manage their teaching affairs. Upon entering the course page, the system automatically displays basic information about the course, including the professor's student number and the corresponding course code. Professors can manage attendance tasks by viewing a chronological list that displays the attendance rate for each task; clicking on a task card takes the professor to the detailed attendance report page for that task. To add a new attendance task, the professor must click on the "+" hover button at the bottom right corner; in the subsequent pop-up dialog box, the professor needs to select the start and end date and time of the attendance task, and then confirm the addition. Using the bottom navigation bar, the professor can click the "Members" button to view the list of students in the course. Clicking on the "Statistics" button will take user to the attendance statistics, and clicking on the "Check-in" button will return user to the Attendance Task List page.

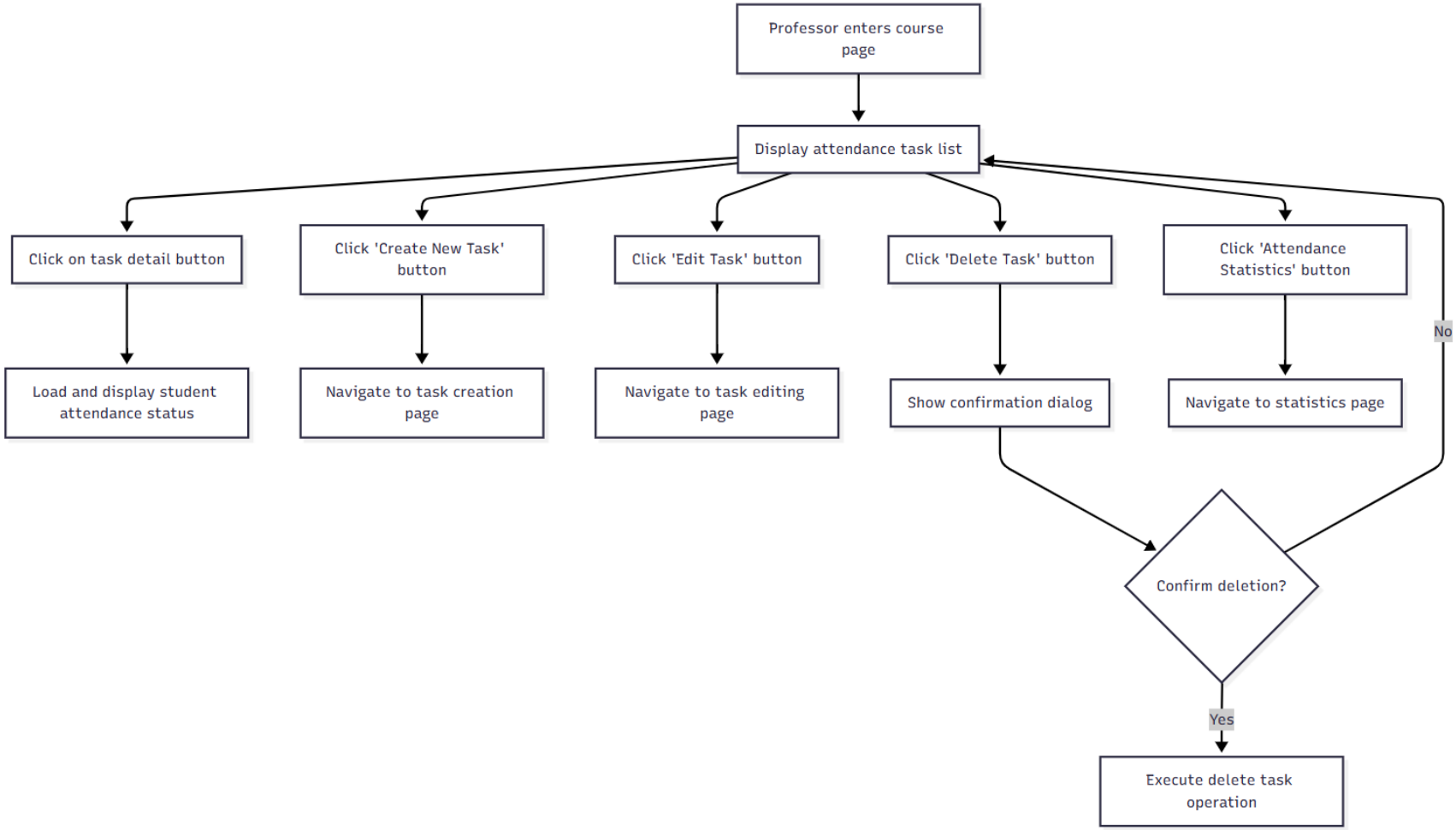


Figure 5.19 : Business Flowchart of Professor Course Module

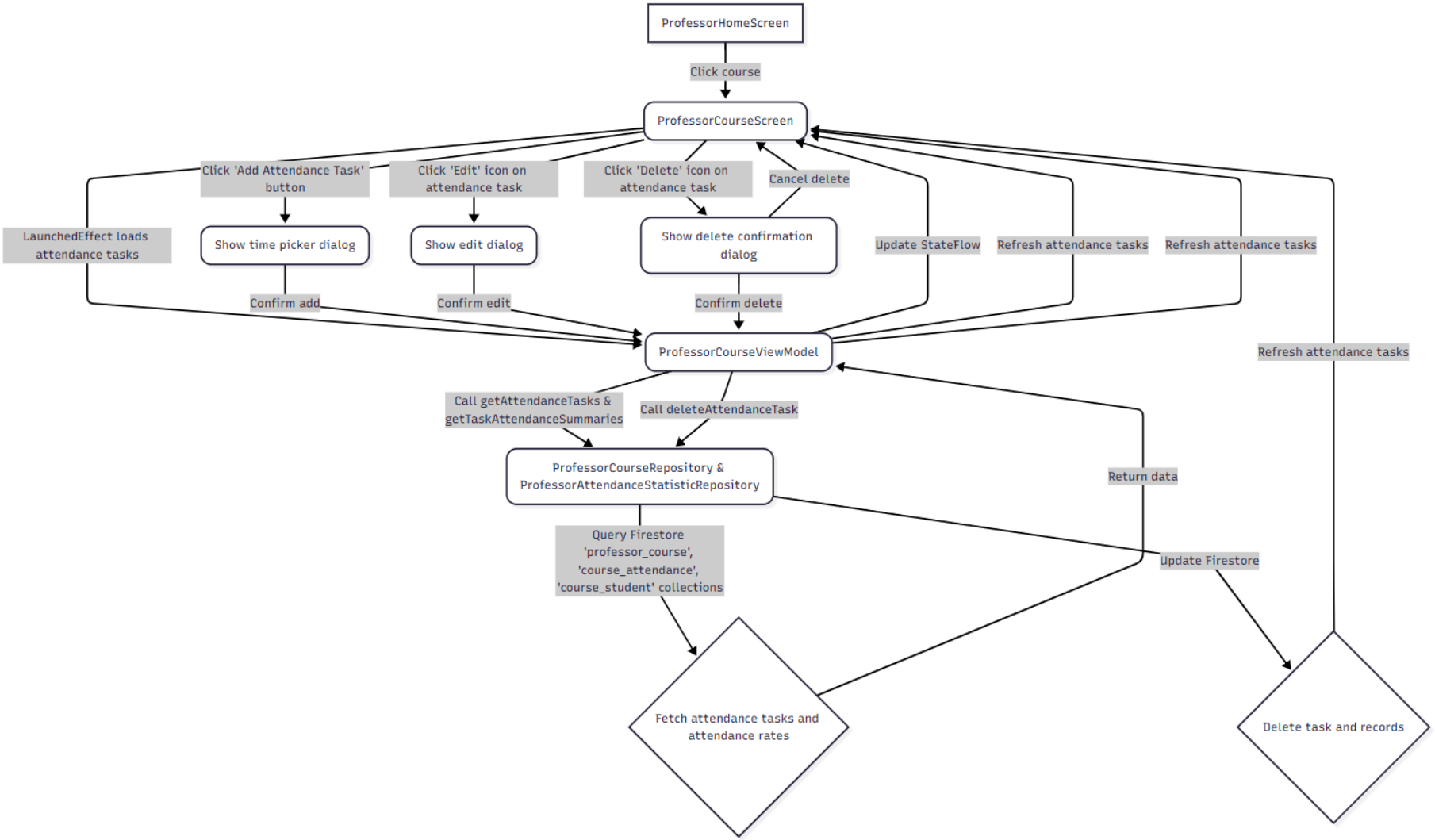


Figure 5.20 : Workflow of Professor Course Module

6. Student Course Attendance Statistics Module

This section details the design and implementation of the Student Course Attendance Statistics Module, specifically engineered to offer students an intuitive statistical and visual display of their attendance data. This tool helps students gain a comprehensive understanding of their attendance performance within a particular course.

The Student Course Attendance Statistics Module's core function is to provide a comprehensive summary of course attendance. Through this module, students can easily understand their personal attendance status and clearly view the number of tasks they have signed in for versus those they have missed. By presenting information through charts and graphs, the module is designed to help students quickly grasp their overall attendance standing.

The interface design of the Course Attendance Statistics page focuses on data visualization and user-friendliness and is divided into several key areas. The top bar is the navigation and functional control area of the page, prominently displaying the page title and course name. The statistical information card immediately below the top bar succinctly displays the course code and student number, providing a quantitative overview of student attendance by offering key attendance metrics including total tasks, signed-in tasks, and unsigned-in tasks. The Attendance Chart is the central visualization on the page, showing attendance visually in a circular chart with specific percentage values clearly marked in the center of the chart. The legend below the chart clearly illustrates the percentage of “checked in” tasks in blue and the percentage of “not checked in” tasks in red, greatly improving the readability of the data. The bottom navigation bar provides easy global navigation, including a “Statistics” button (which displays the current page) and a “Tasks” button (which users can click to check in).

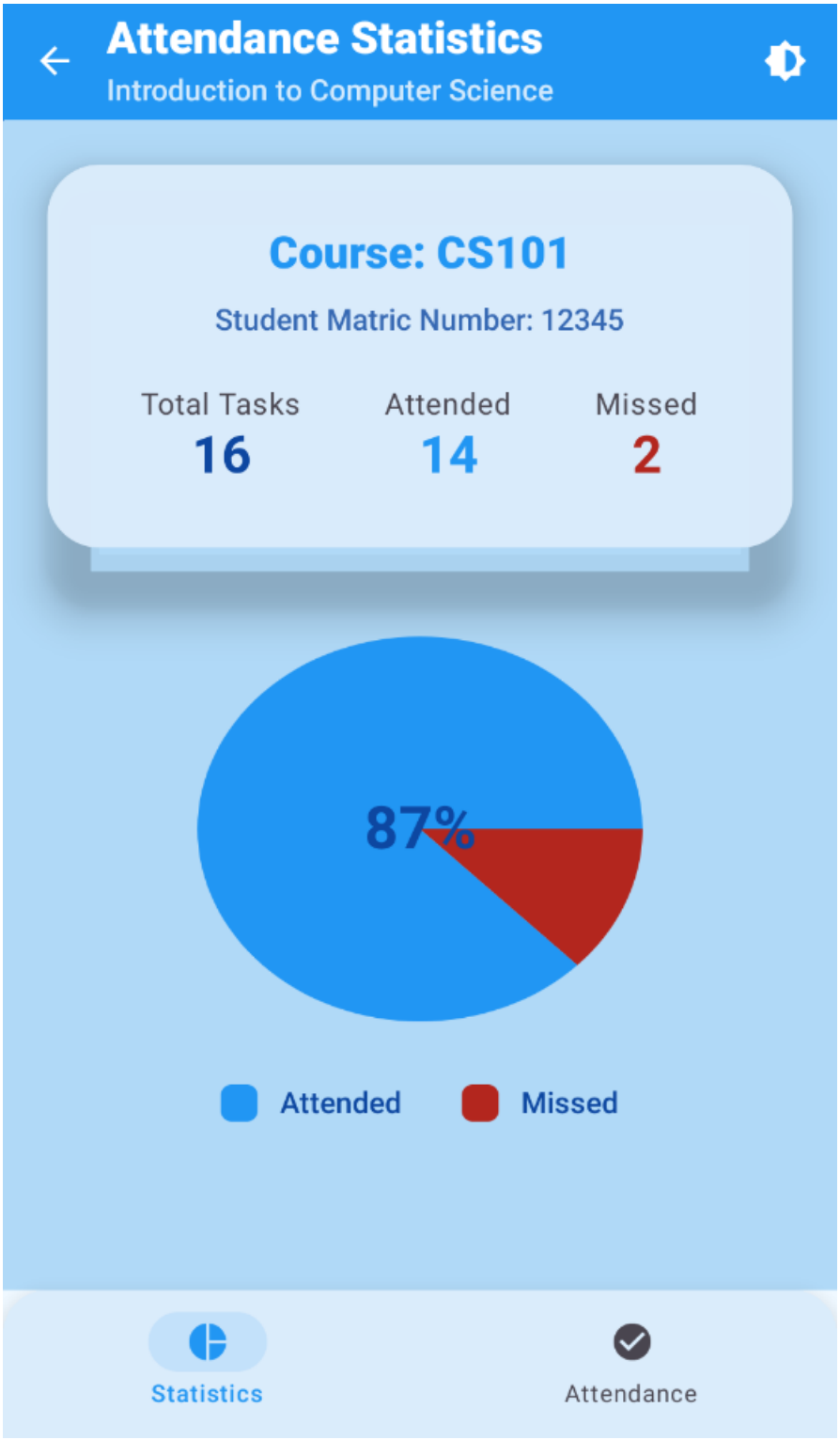


Figure 5.21 : UI of Student Course Attendance Statistics Module

The Student Course Attendance Statistics module supports a variety of basic operations to ensure that students can effectively access and understand their attendance data. Students

can access the Attendance Statistics page by clicking the Statistics button on the bottom navigation bar of the Course Details page. The system automatically displays the relevant attendance statistics as the page loads. On the page, the stats card displays overall attendance data, while the ring graph visually depicts attendance with a legend to help users understand the meaning of the different colors.

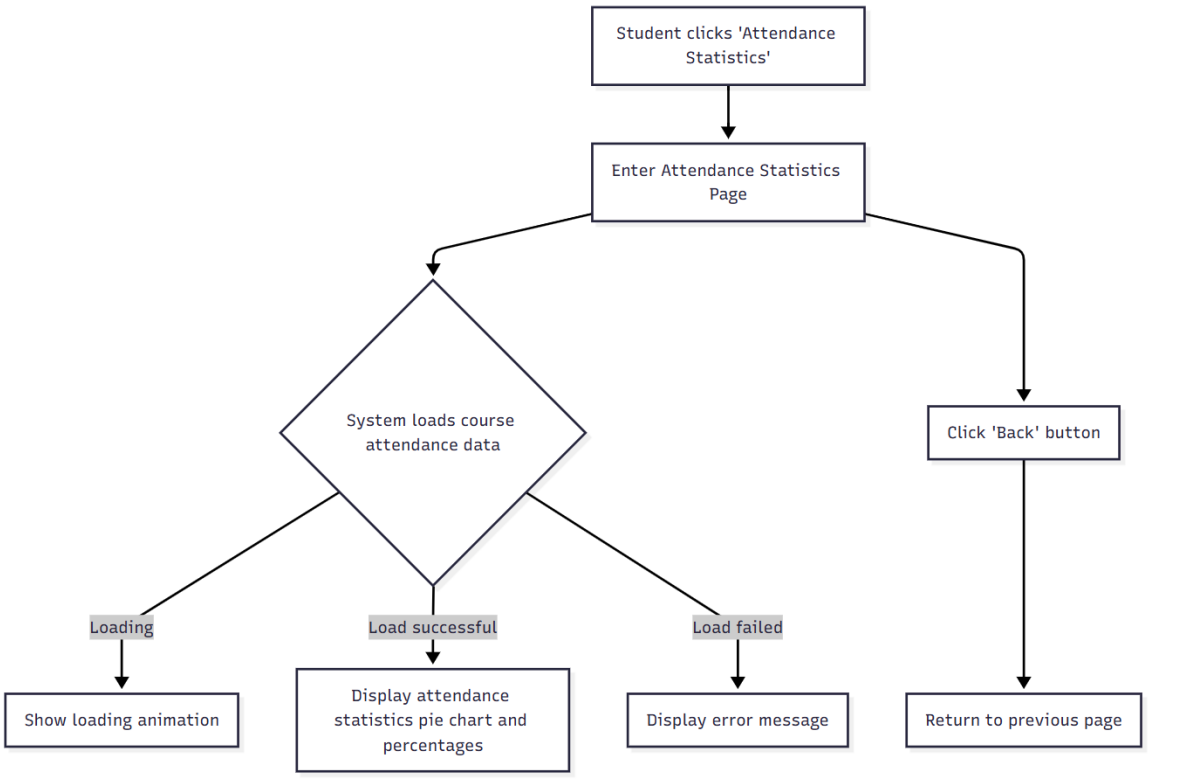


Figure 5.22 : Business Flowchart of Student Course Attendance Statistics Module

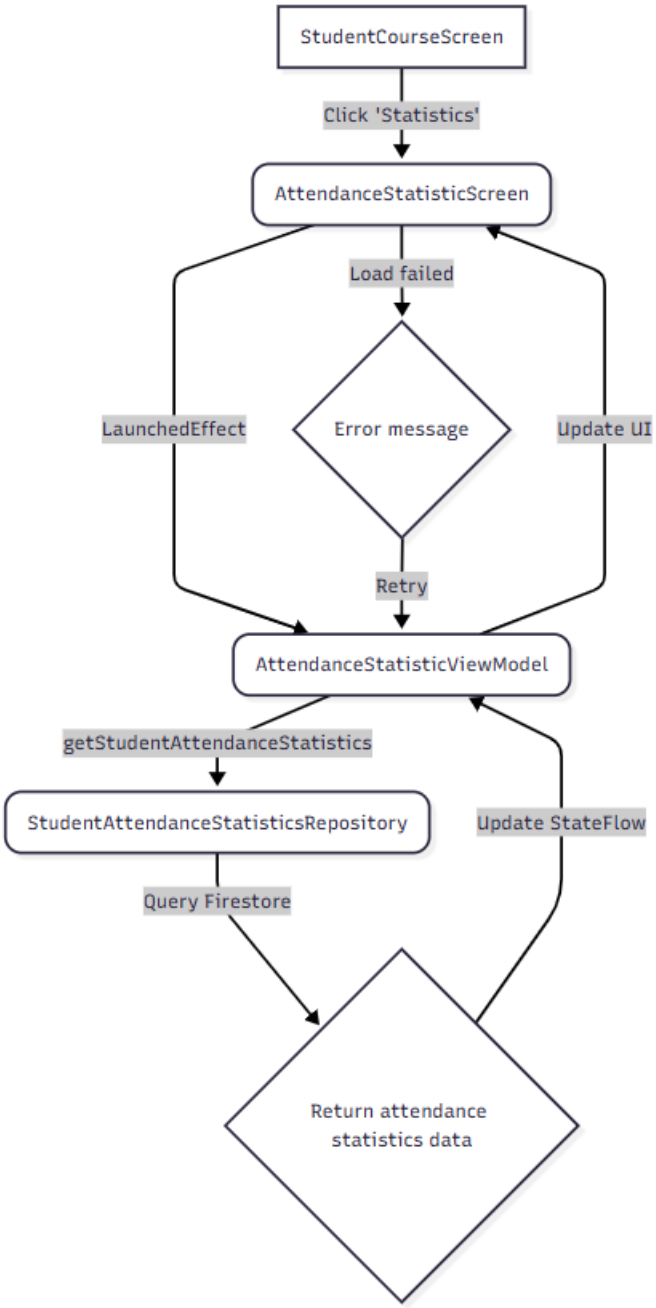


Figure 5.23 : Workflow of Student Course Attendance Statistics Module

7. Professor Course Attendance Statistics Modul

This section details the design and implementation of the Professor Course Attendance Statistics module, which provides users with a powerful tool for visualizing and analyzing course attendance data. Through this tool, professors can visualize attendance trends, grasp the attendance status of each specific attendance task, and analyze changes in attendance data to get a comprehensive understanding of the overall attendance situation.

The core function of the Professor Course Attendance Statistics module is to graphically display course attendance trends to help professors visualize attendance for each attendance task. By visually analyzing attendance data, professors can effectively interpret changes in attendance patterns and gain a comprehensive understanding of the overall attendance situation, ultimately providing important data support for adjusting teaching strategies.

The interface design of the Professor Course Attendance Statistics page emphasizes clear data display and easy interaction, and consists of several key areas. The top bar is the navigational and functional control area of the page, prominently displaying the page title and indicating the name of the course currently being viewed. The main content area is the core of the attendance data visualization. This area displays a graph of attendance trends, where the X-axis represents the attendance task sequence number and the Y-axis represents the percentage of attendance. The lines in the graph visually depict the attendance trends in the task sequence, and the individual data points represent the specific attendance values for each attendance task. In addition, this area shows the total number of attendance tasks, providing important overall volume information.

Figure 5.24 : UI of Professor Course Attendance Statistics Module

The Professor Attendance Statistics module supports a variety of basic operations to ensure that professors can effectively access and analyze attendance data. Upon entering the page, the system automatically loads and displays attendance statistics; users must wait for the loading animation to complete before they can see the attendance trend graph. To interpret

overall attendance trends, professors can observe the overall trajectory of the folded lines to determine the general direction of attendance. Professors can drag the chart to view data from different time periods and use the two-finger zoom feature to adjust the chart's display for more detailed analysis.

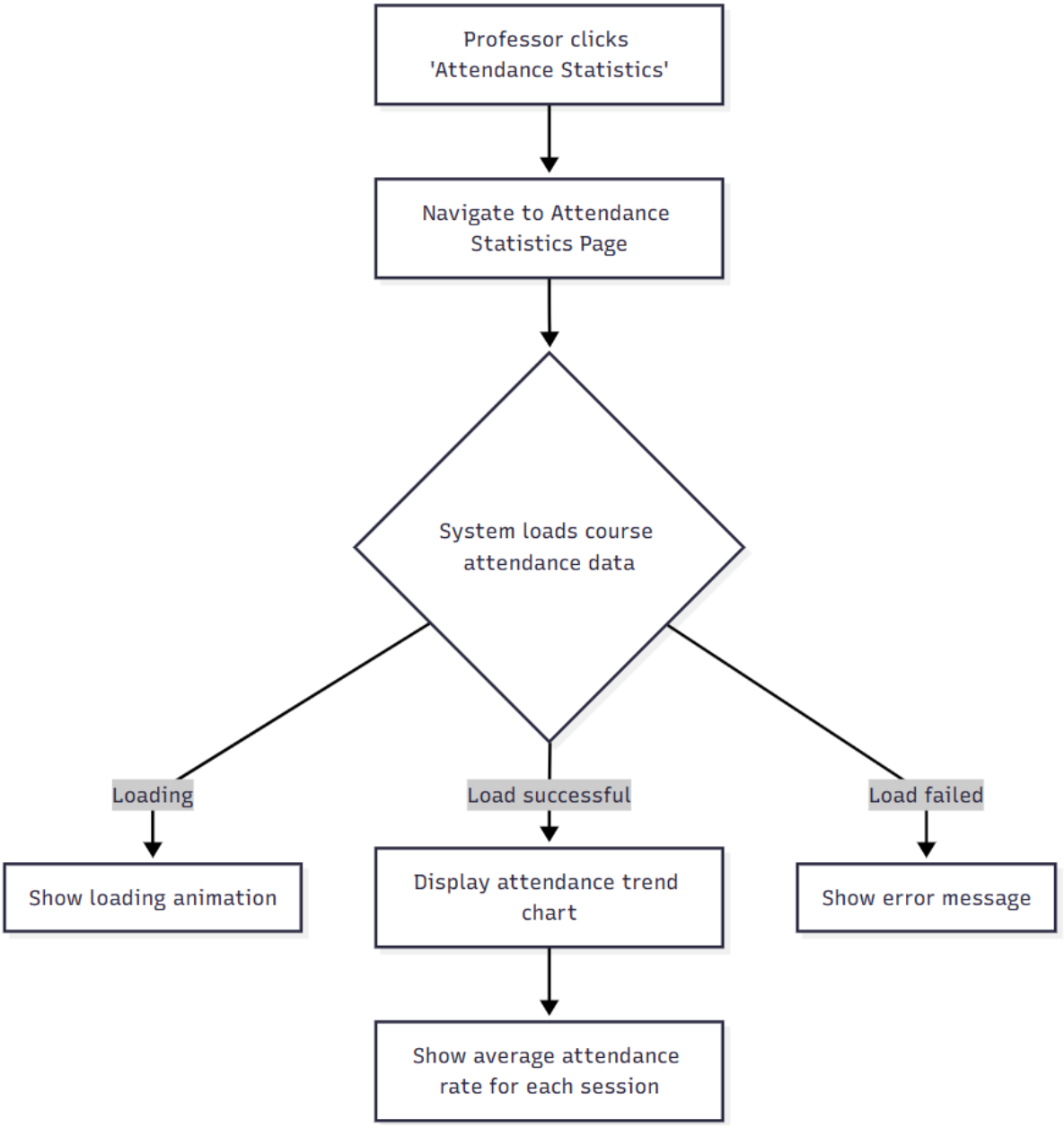


Figure 5.25 : Business Flowchart of Professor Course Attendance Statistics Module

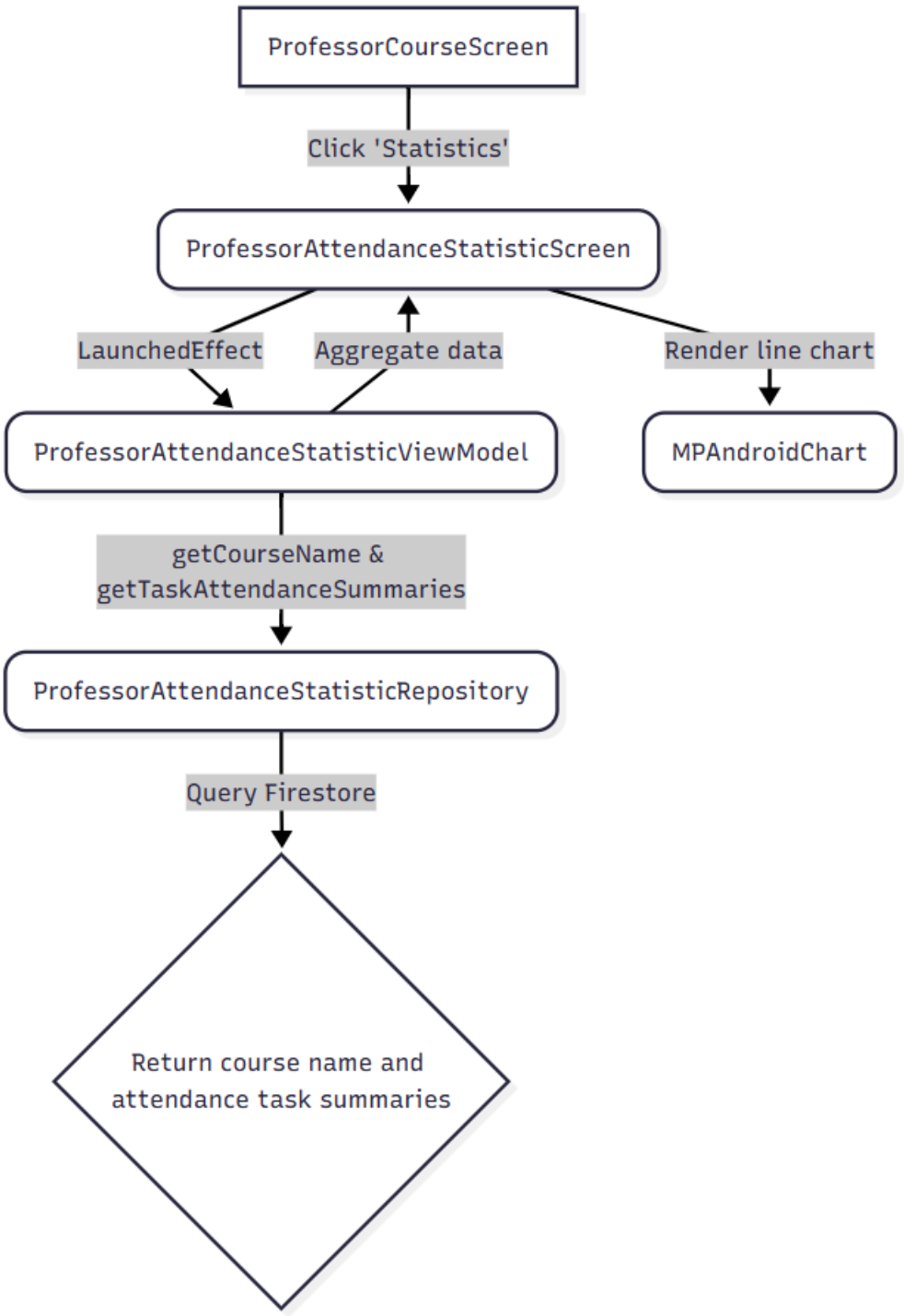


Figure 5.26 : Workflow of Professor Course Attendance Statistics Module

8. Course Student Module

This section details the design and implementation of the course student module, which is intended to provide a comprehensive interface for instructors. This interface allows the instructor to view and manage information about all members of the course for which he or she is responsible and to closely monitor their attendance.

The core functionality of the course student module is to allow the professor to view a comprehensive list of all students enrolled in the course. For each student, the professor can clearly see their attendance and view further details about the student.

The main content area of the course student page is the core display space for student information. This section clearly displays the details of all students enrolled in the course in a list format, including each student's name, student number, and attendance (displayed as a percentage). Attendance is visually color-coded. Green represents 80% and above, indicating good attendance; orange represents 60% to 79%, indicating fair attendance; and red represents less than 60%, warning of poor attendance.

Figure 5.27 : UI of Course Student Module

The Course Student module supports a variety of basic operations to ensure that professors can effectively manage course membership information. Upon entering the page, the system automatically loads and displays a list of all students enrolled in the course. If there are no students currently enrolled, the page displays the message “There are no students in this course”. Professors can view student details by clicking on any student's card, and the system will seamlessly navigate to that student's dedicated details page to view their personal information. Additionally, instructors can use the bottom navigation bar to easily switch between the “Statistics” page for viewing course attendance statistics and the “Tasks” page for course attendance management.

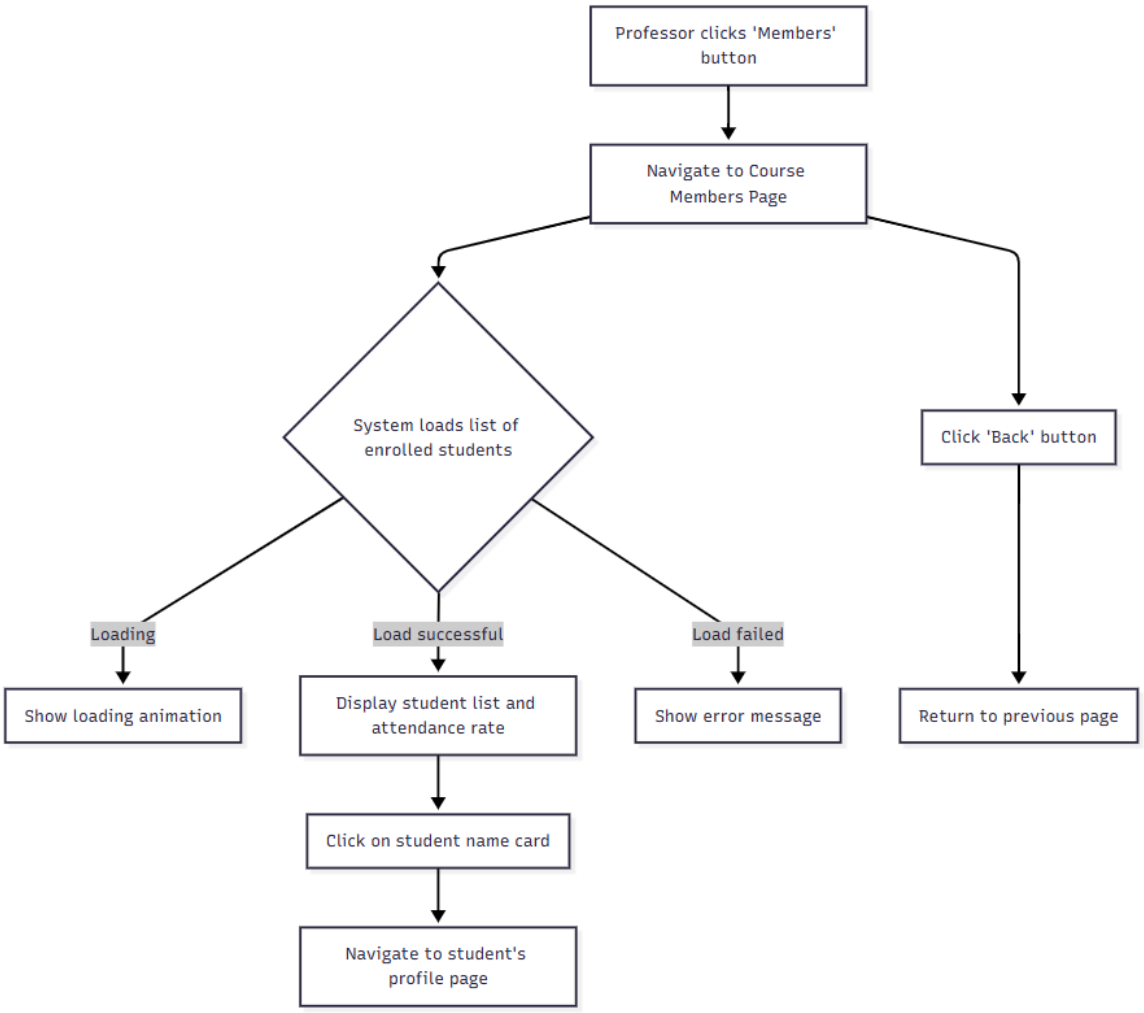


Figure 5.28 : Business Flowchart of Course Student Module

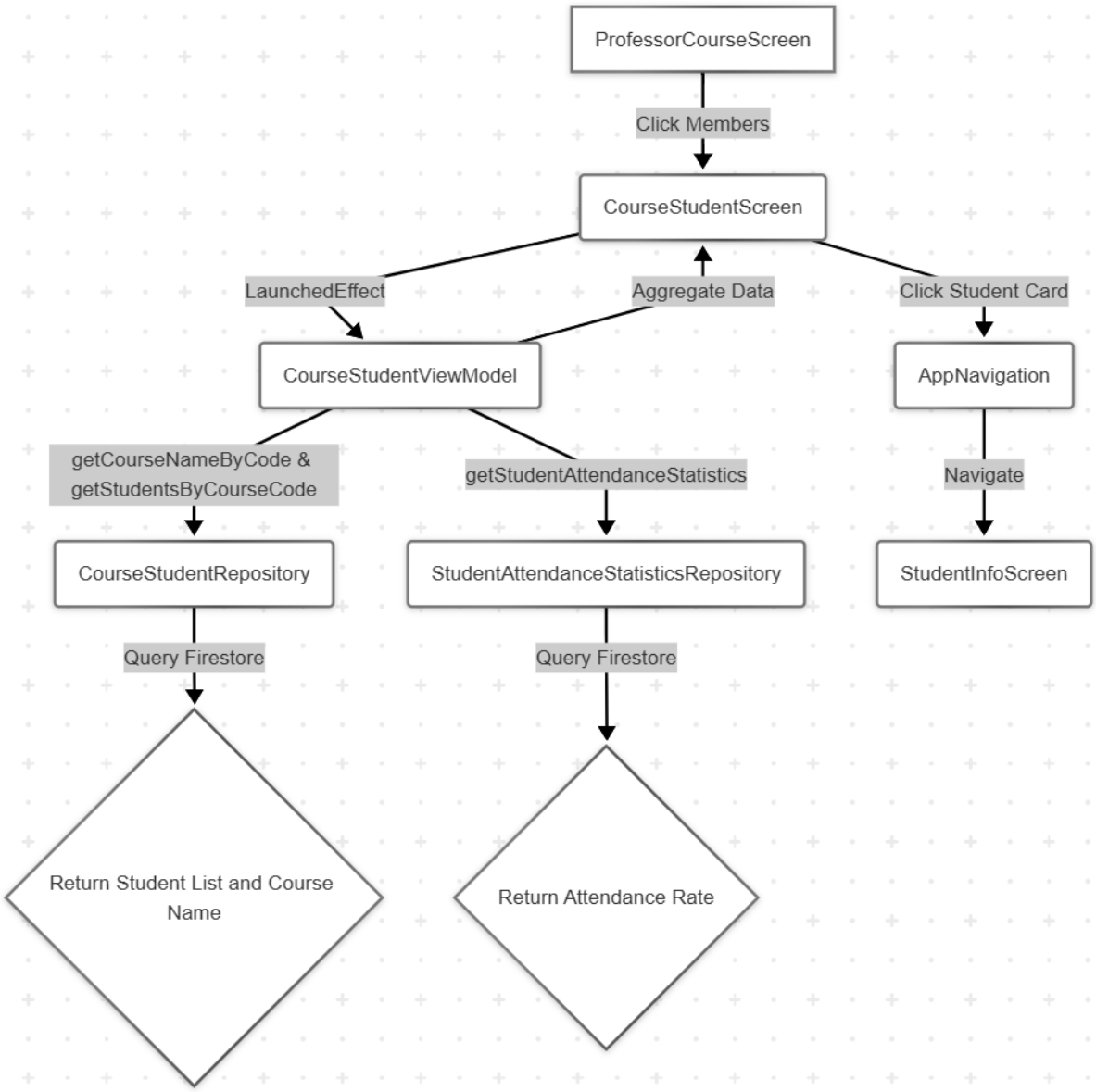


Figure 5.29 : Workflow of Course Student Module

9. Attendance Report Module

This section details the design and implementation of the Attendance Reporting module, which is intended to provide professors with a comprehensive, filterable interface for viewing and managing the attendance status of all students associated with a specific attendance task and to provide a quick overview of the overall attendance status.

The core functionality of the Attendance Reports module is to display the attendance status of all students under a specific attendance task. Professors can utilize this module to filter and view students who are present or absent, and can further manually complete student attendance tasks.

The interface design of the Attendance Reports page prioritizes data readability and ease of filtering and includes several key areas. The statistical overview area is located directly below the top bar and visualizes the overall status of the current attendance task. This area presents the number of students present and absent and displays the overall attendance rate to help professors get a quick overview of attendance. The Filter Control area provides flexible data filtering options. Professors can use the segmentation buttons to toggle between All, Present, and Absent, with the currently selected option highlighted for quick attention to the status of a specific student list. The main content area is the centralized display area for student attendance information, clearly displaying data in the form of a list of students, including their names and student numbers. Attendance status labels are attached to each student entry, with a check mark icon indicating that the student is present and a cross icon indicating that the student is absent.

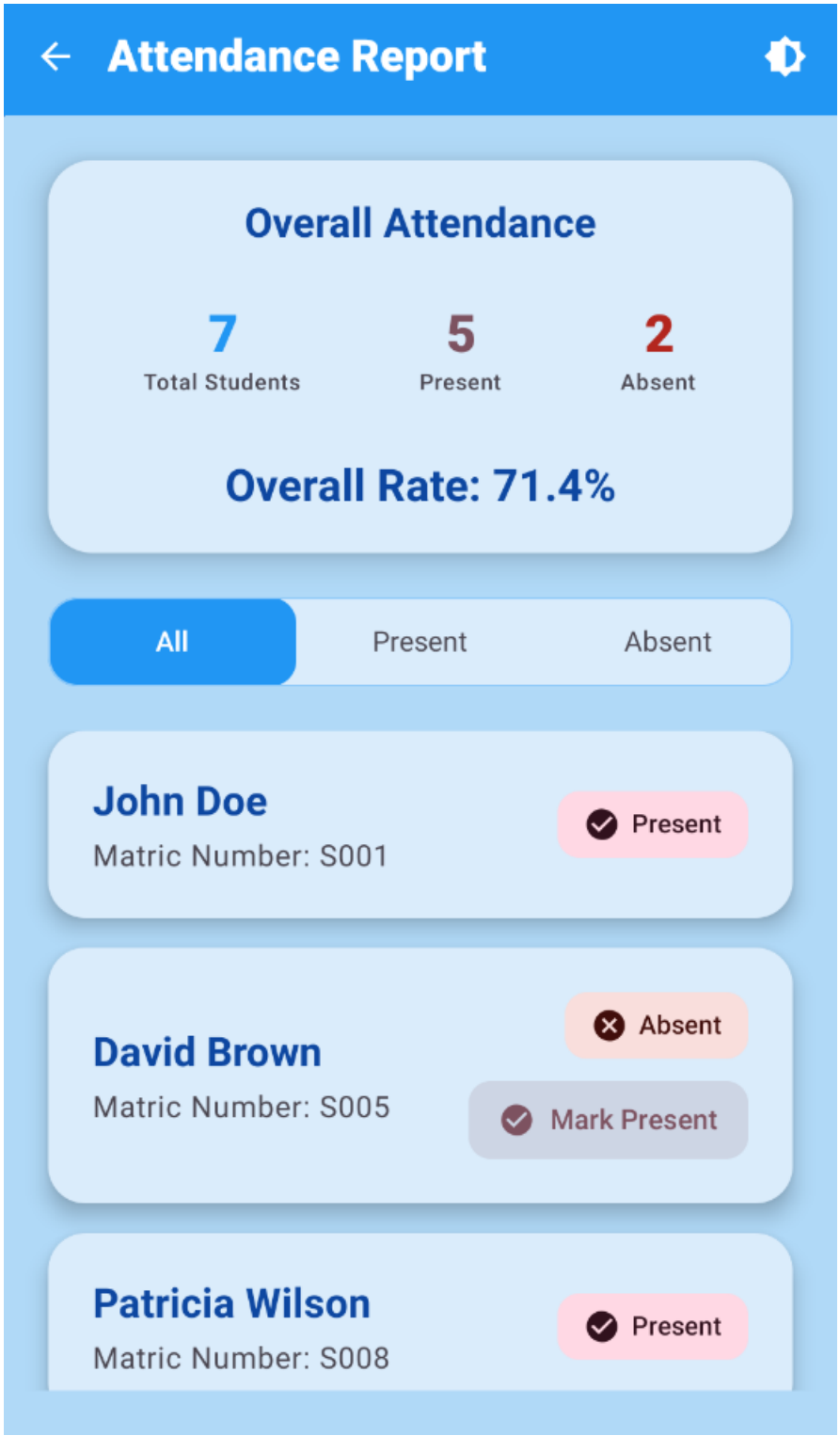


Figure 5.30 : UI of Attendance Report Module

The Attendance Report module provides a series of basic operations designed to help teachers efficiently view and analyze student attendance. When the professor enters the Attendance Report page, the system will automatically load and present the detailed data of the current attendance task. If the task has not yet recorded any attendance information, the system will prompt “No attendance data”. To enhance the flexibility of data browsing, the interface is equipped with an attendance status filtering function. Professors can use the filter button to categorize the list of students. Selecting “All” will display all students' attendance status; selecting ‘Present’ will display only those who have signed in; selecting “Absent” will list only those who have not yet signed in.

In order to cope with special circumstances, such as individual students who missed signing in due to network or system anomaly, the system also provides “make-up” function for “absent” students. When the filter view is set to “Absent”, a “Mark Present” button will be displayed next to the corresponding student entry. By clicking on this button, the professor can change the student's attendance status from “absent” to “present”, ensuring the accuracy and control of the attendance data.

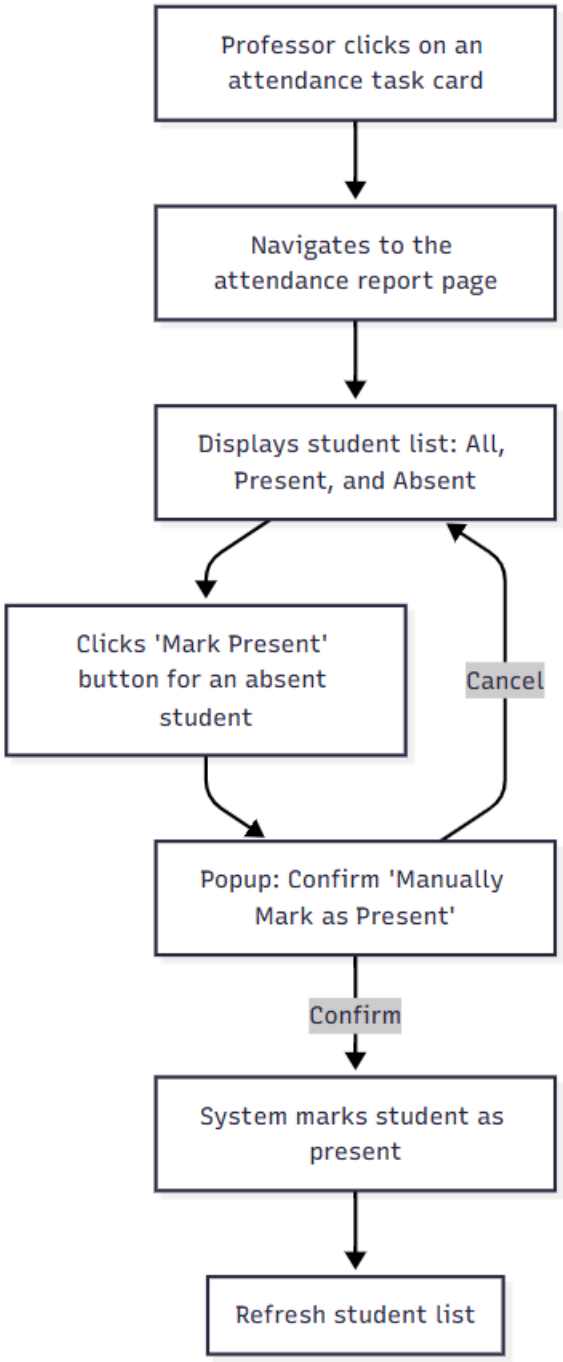


Figure 5.31 : Business Flowchart of Attendance Report Module

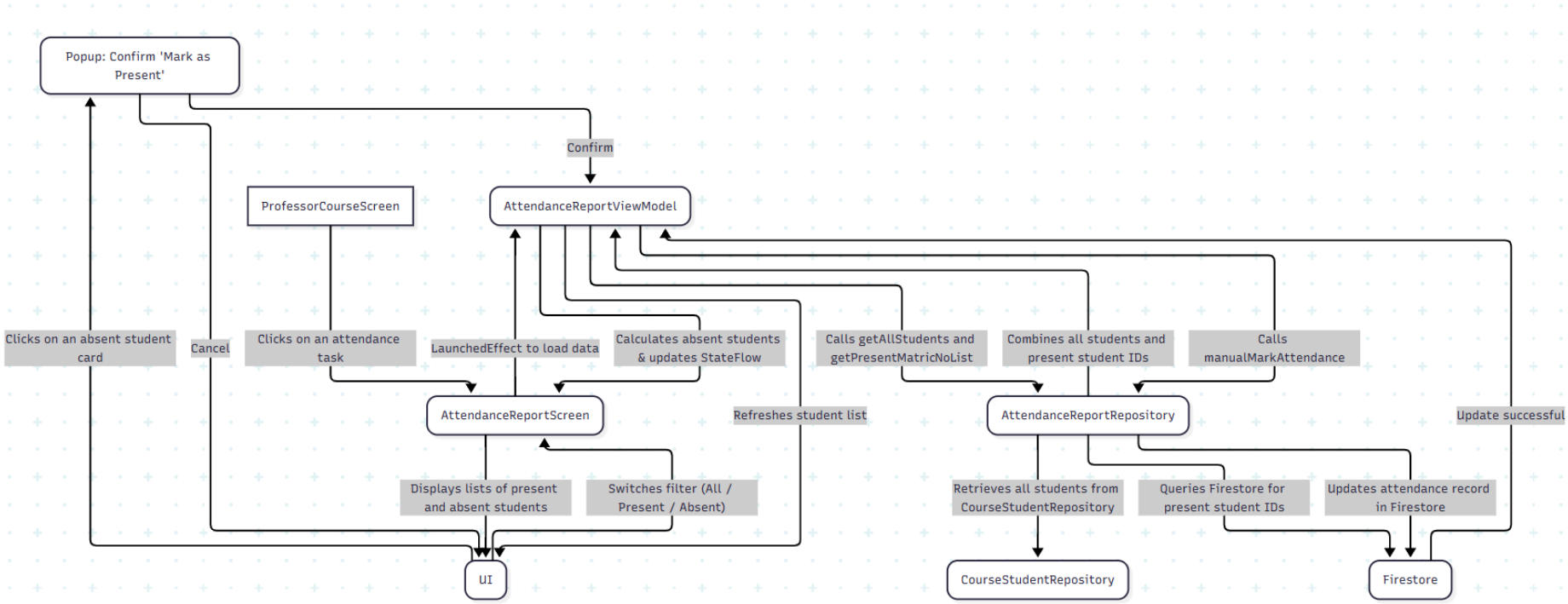


Figure 5.32 : Workflow of Attendance Report Module

10. Face Registration and Face Recognition Module

This section details the design and implementation of the Face Enrollment and Attendance Recognition module at the user level. The module provides key biometric functions for students, aiming to realize secure identity verification and automatic attendance recording through face data, thus replacing the traditional manual sign-in method and significantly improving the efficiency and accuracy of attendance.

The Face Registration and Attendance Recognition module consists of two main functions. The first function is facing registration, where students can securely input their unique facial biometric data into the system by uploading an existing photo or capturing a real-time image, and this facial data will be used as the only credential for attendance verification. The second feature is attendance recognition, where students can take a photo of themselves in each class; the system automatically recognize the captured face and subsequently complete the attendance record, completely eliminating the need for manual sign-in operations.

Face registration is an important step that students must complete before using the attendance system for the first time, and its process is carefully designed to ensure the accuracy and validity of the face data collected. After logging in to the system, users need to first enter the “Profile” page and click the “Upload Photo” button to enter the exclusive face registration interface. In this interface, the system provides two ways to submit photos. Users can click “ Gallery” to select a clear front face photo from the device, or click “Take Photo” to activate the device camera and immediately take a frontal photo that meets the requirements.

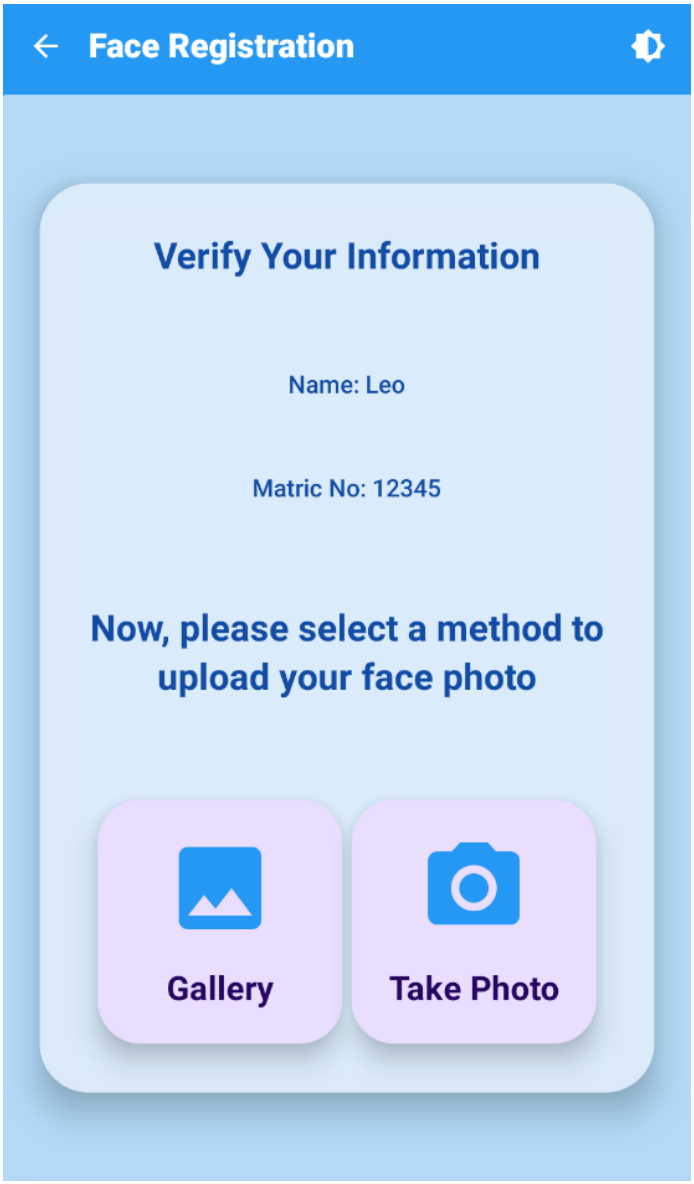


Figure 5.33 : UI of Face Registration Module

It is important to note that in order to ensure successful registration and subsequent accurate identification, the submitted photo must meet strict criteria, i.e., it must be a photo of the person's face, well-lit, with no face cover, and with only one person in the frame. If the system requests camera or album access for the first time, the user must click “Allow” to authorize the application to access.

After submitting the photo, the system automatically enters the processing stage for face detection and feature registration; during this process, the page will display prompts such as “Processing photo and registering face .....”, and users need to wait patiently for the processing to complete. The registration result will be clearly feedback to the user. If the registration is successful, the page will display "Face registration successful! and the user can return to the personal information page. If registration fails, the system provide specific reasons for the failure, for example, “No face detected” prompts the user to make sure that the photo is clear and is a frontal face, and “Multiple faces detected” prompts the user to make sure that only the user is in the photo.

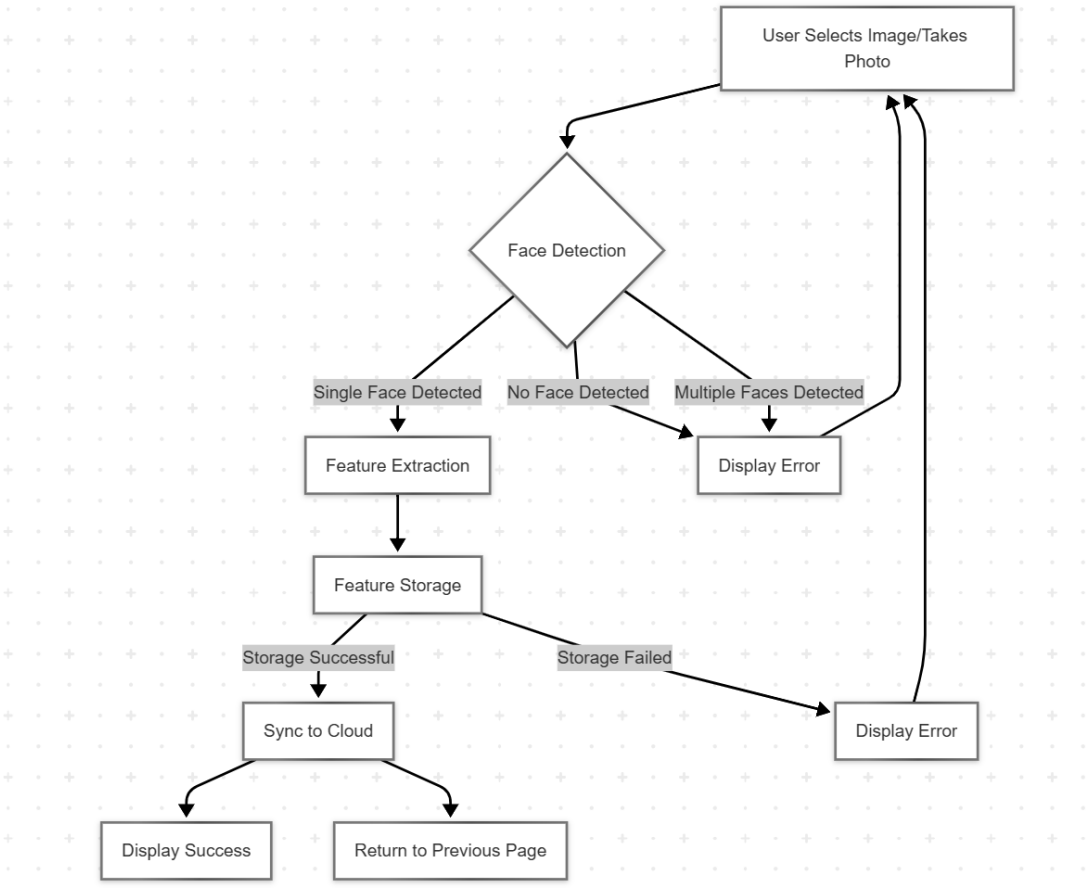


Figure 5.34 : Business Flowchart of Face Registration Module

Attendance recognition is the core operation of every course attendance session, and the process is designed to enable automated and efficient facial identity verification. Students can access the Attendance Recognition interface by clicking on the Attendance Task Card on the course page. In this interface, the system provides users with an identification method. Users can click on the “Take Photo” button to activate the device camera and take a current frontal photo for identification.

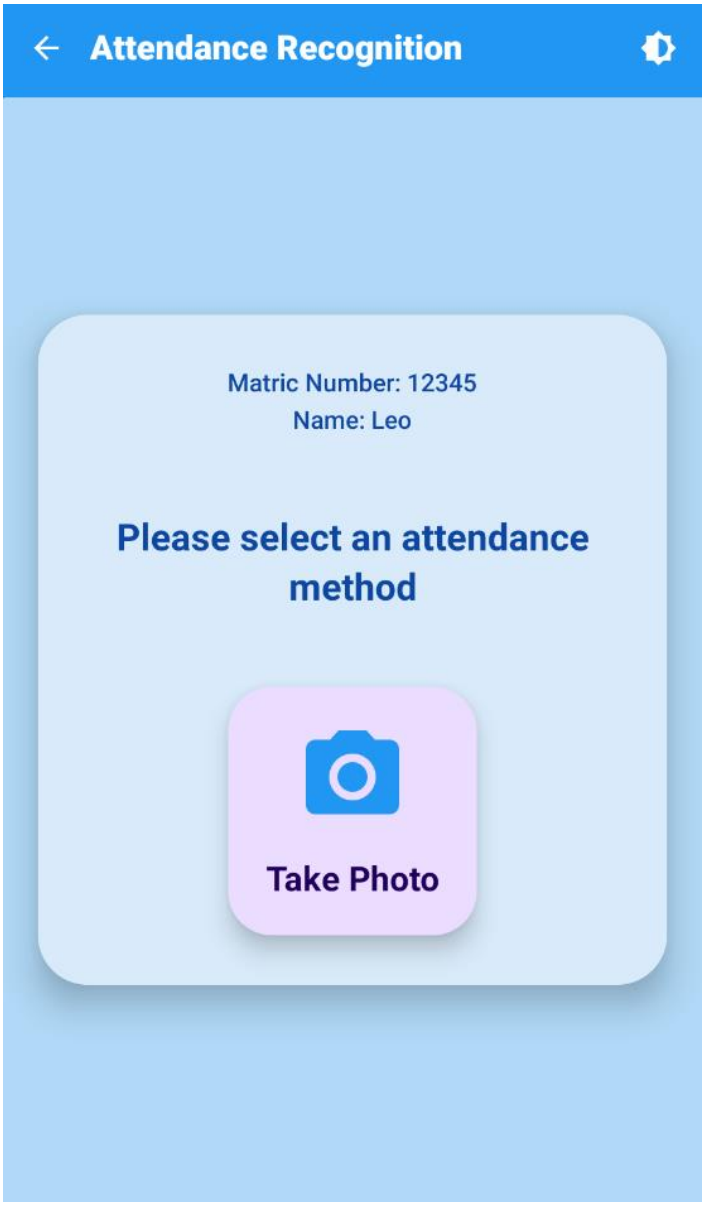


Figure 5.35 : UI of Face Recognition Module

In order to ensure the accuracy of identification, the uploaded photo must be the user's own, and try to take the photo in an environment similar to that of the registration, avoiding bright light, face blocking or multiple faces in the picture. If the system requests camera or album permissions, the user must click “Allow” to continue. After the photo is submitted, the system automatically performs face detection, feature extraction and comparison, and automatically complete the attendance record after successful recognition.

During the process, the page displays a prompt, such as “Face recognition in .....”, users need to wait patiently for the recognition results. The recognition results then be clearly displayed to the user. If the recognition is successful, the page display "Check-in Successful! which indicates that the system has accurately recorded the user's attendance information. If recognition fails, the system provides a specific reason for the failure and allow the user to try again. “No face detected” prompts the user to adjust the camera angle or lighting to ensure that the face is clear. “Multiple faces detected” prompts the user to make sure that only the user is in the picture. “Face does not match” suggests the user to confirm the identity or re-register the face. “Face not registered” clearly informs the user that face registration must be completed before checking in.

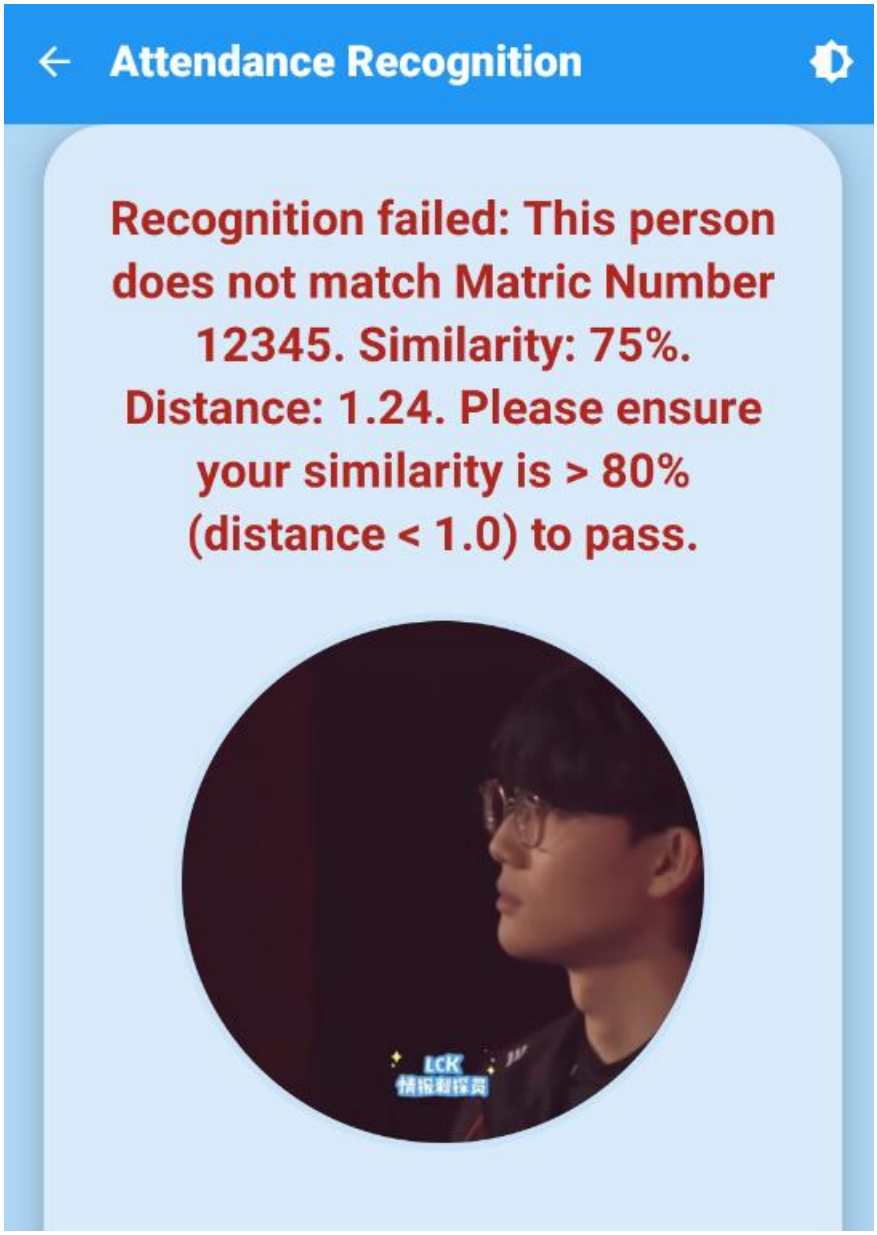


Figure 5.36 : UI when user's face data does not match

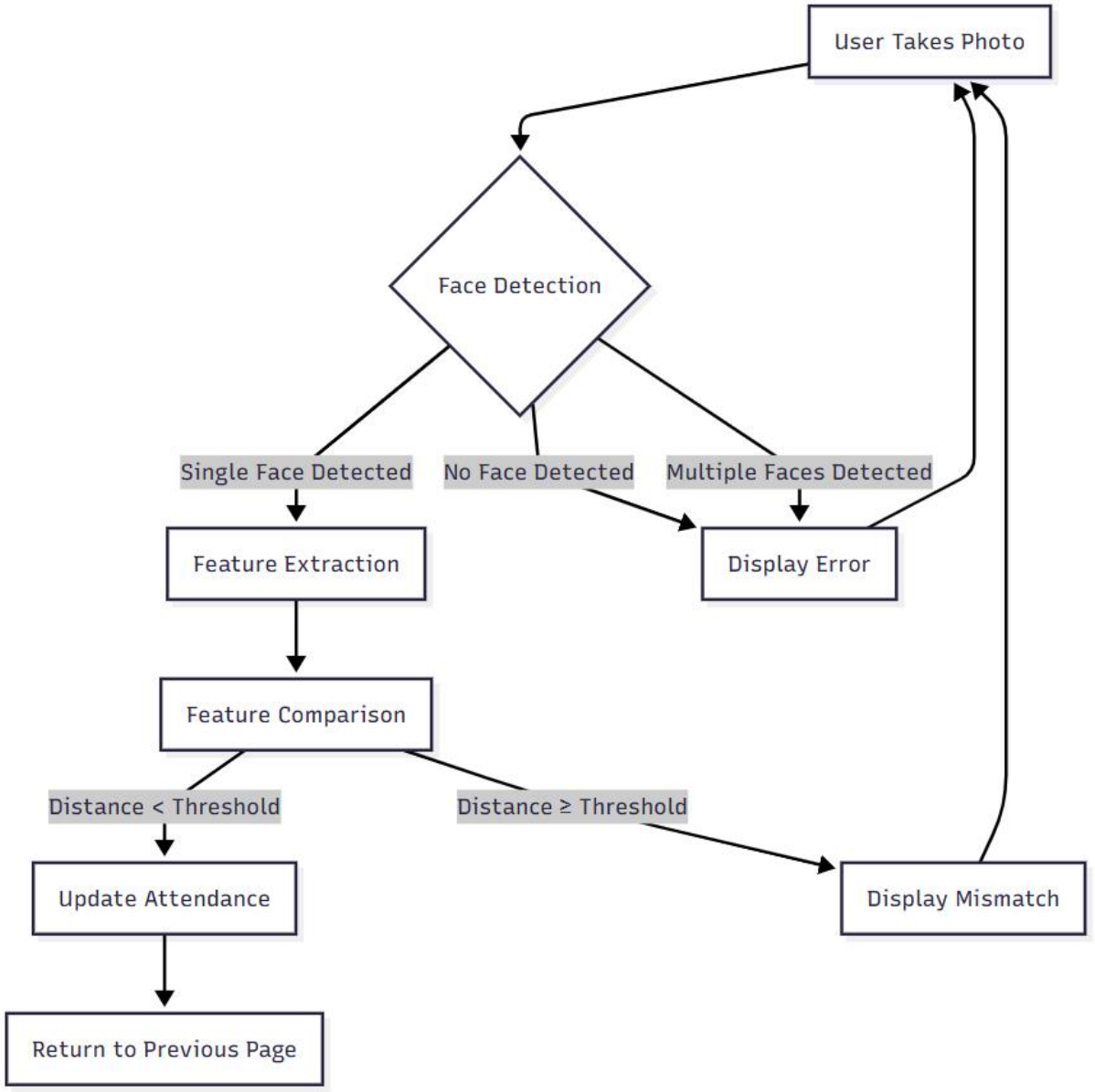


Figure 5.37 : Business Flowchart of Face Recognition Module

5.3.3 Face Recognition Technology Principles and Realization

This section discusses the core technical principles and implementation of the attendance system's face recognition function. Face recognition technology identifies and authenticates a user's identity through the automated processing of image data, thereby providing a fast, secure, and efficient solution for attendance recording and identity verification, effectively replacing traditional manual sign-in or card-based methods. The primary advantages of this technology lie in its unparalleled convenience, inherent uniqueness, and robust anti-cheating capabilities.

Face recognition technology fundamentally relies on advancements in computer vision and pattern recognition, with its core principle based on the extraction and comparison of features from face images. Initially, upon receiving a face image (whether uploaded during registration or captured during attendance), the system employs a face detection algorithm (Google ML Kit) to precisely locate the face area within the image. This detected face is then cropped and aligned to mitigate the influence of variations in posture, lighting, and other environmental factors, preparing it for subsequent feature extraction. After preprocessing, the face images are processed by a face feature extraction network (a pre-trained deep metric learning network based on the CNN architecture in Chapter 4). This network maps the face image to a high-dimensional feature vector space, known as the face embedding space. In this space, feature vectors representing different individuals exhibit large Euclidean distances, while feature vectors from the same person under different conditions maintain small Euclidean distances, making these feature vectors unique digital representations of the face. Finally, in the face matching phase, the system calculates the similarity (using Euclidean distance) between the feature vectors extracted in real time and the feature vectors of the registered user stored in the database. If the calculated similarity exceeds a preset threshold, the identity matching is considered successful; otherwise, the matching fails.

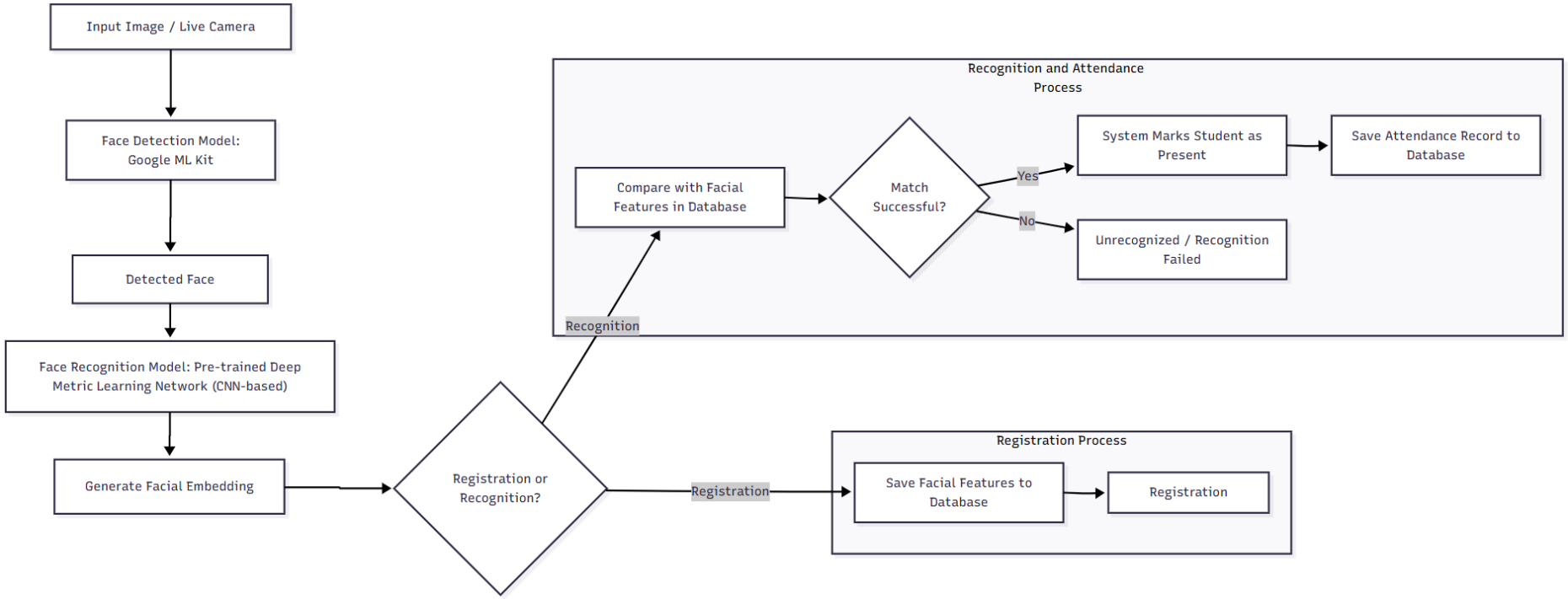


Figure 5.38 : Business Flowchart of Perform Face Recognition System in Application

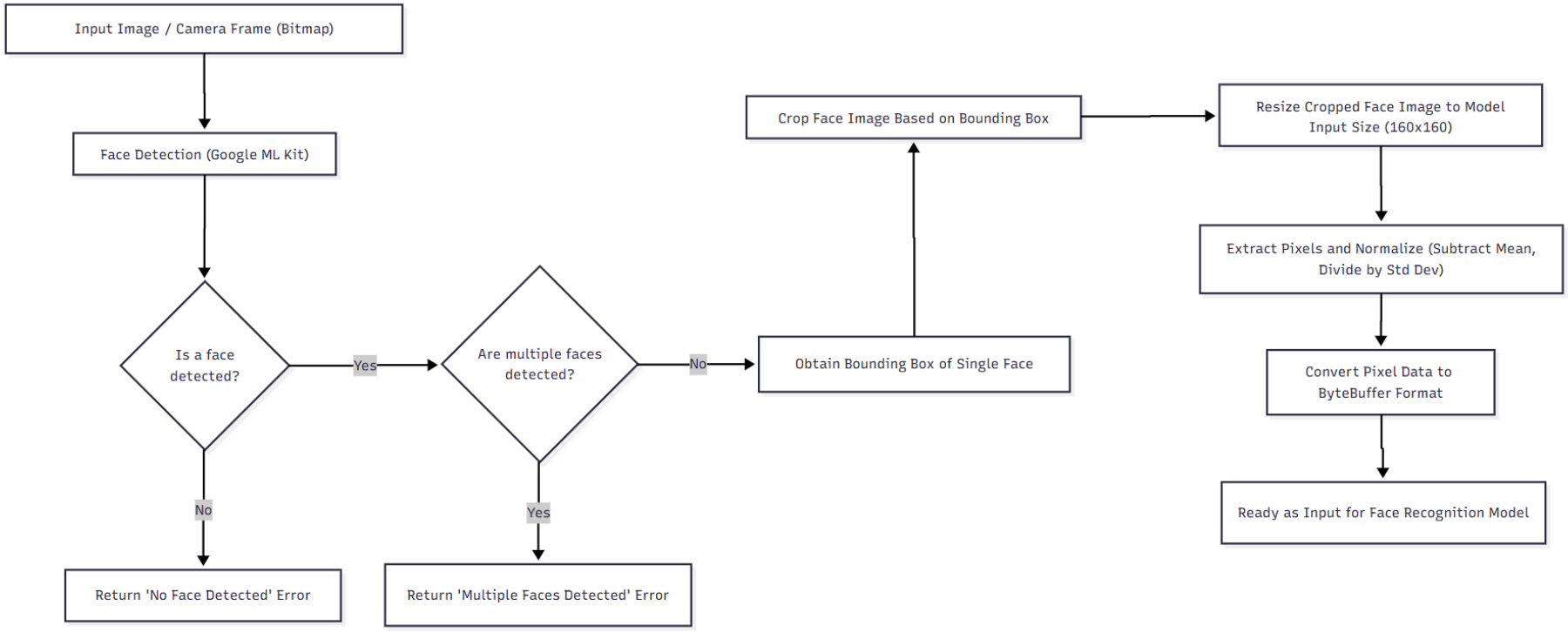


Figure 5.39 : Workflow of Face Detection and Feature Extraction

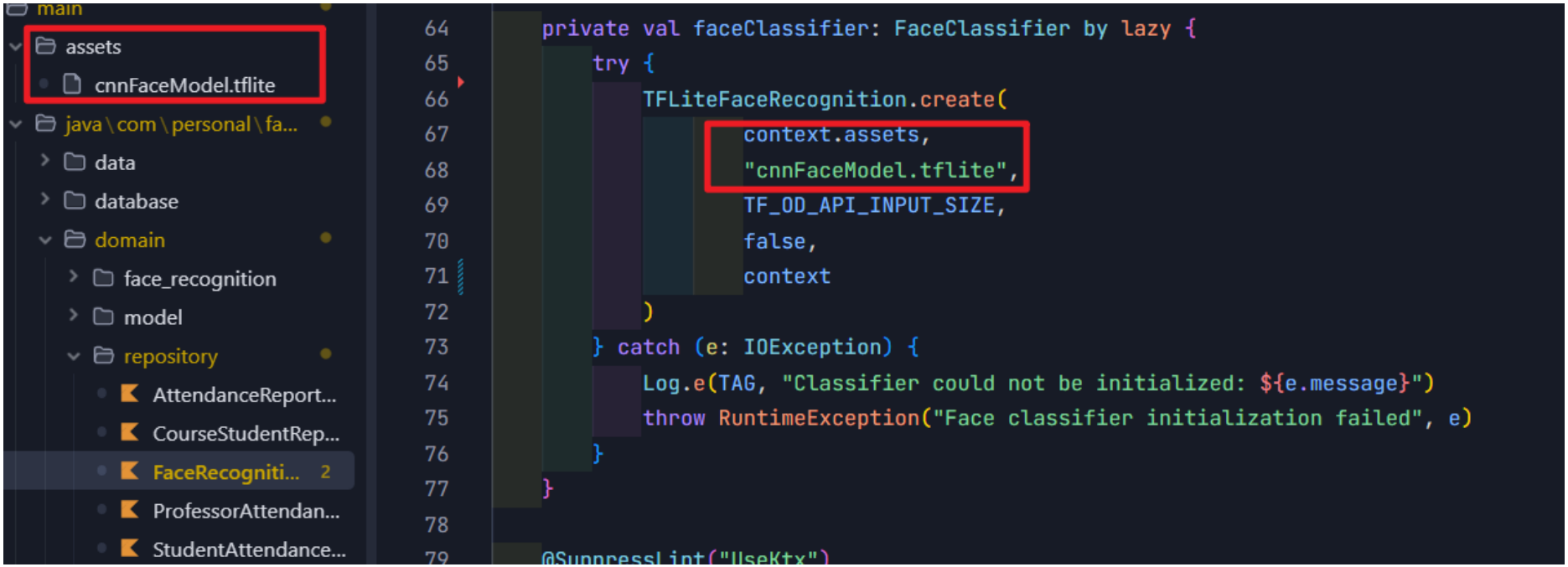


Figure 5.40 : Code snippet for applying the Pretrained Model to the Application

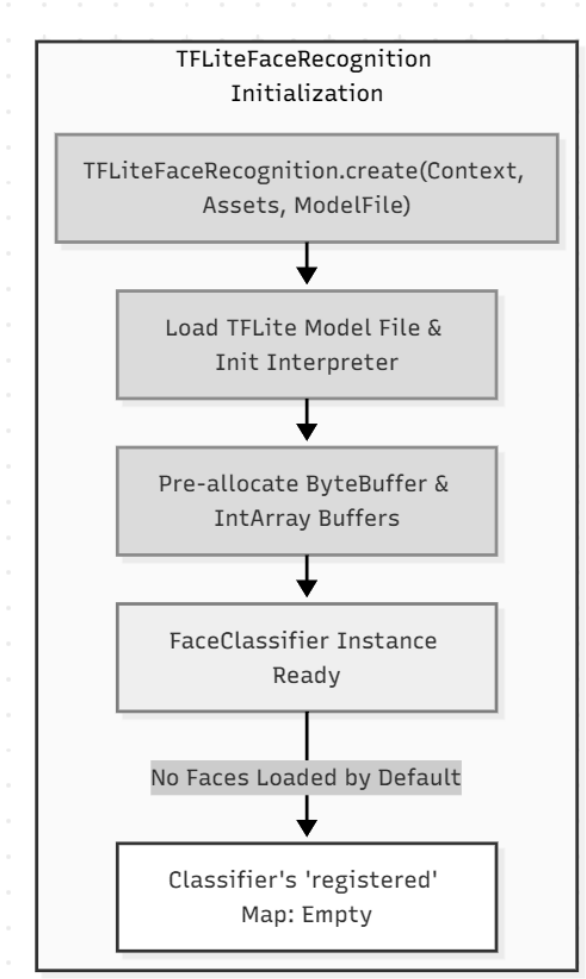


Figure 5.41 : Workflow of TFLiteFaceRecognition Initialization

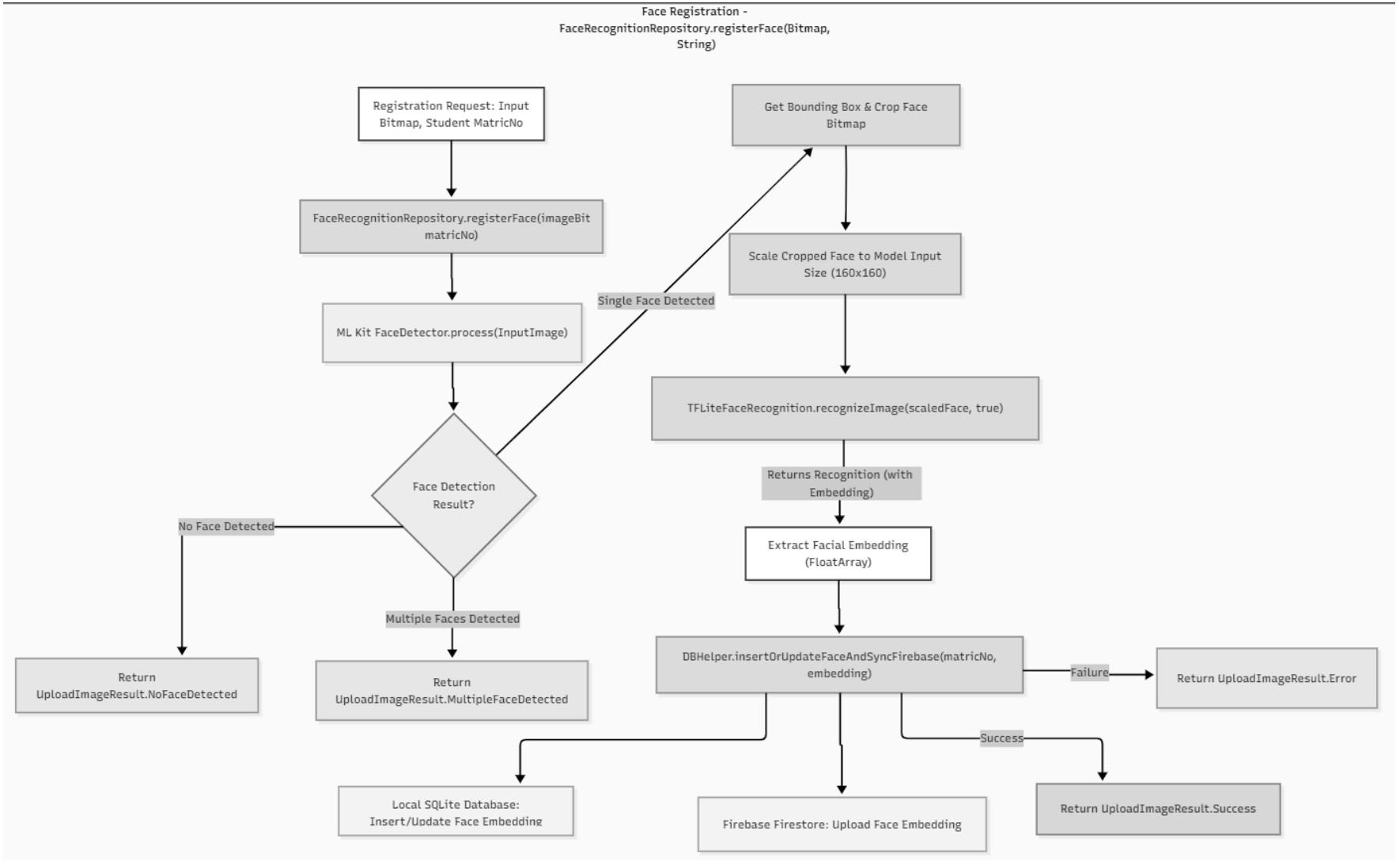
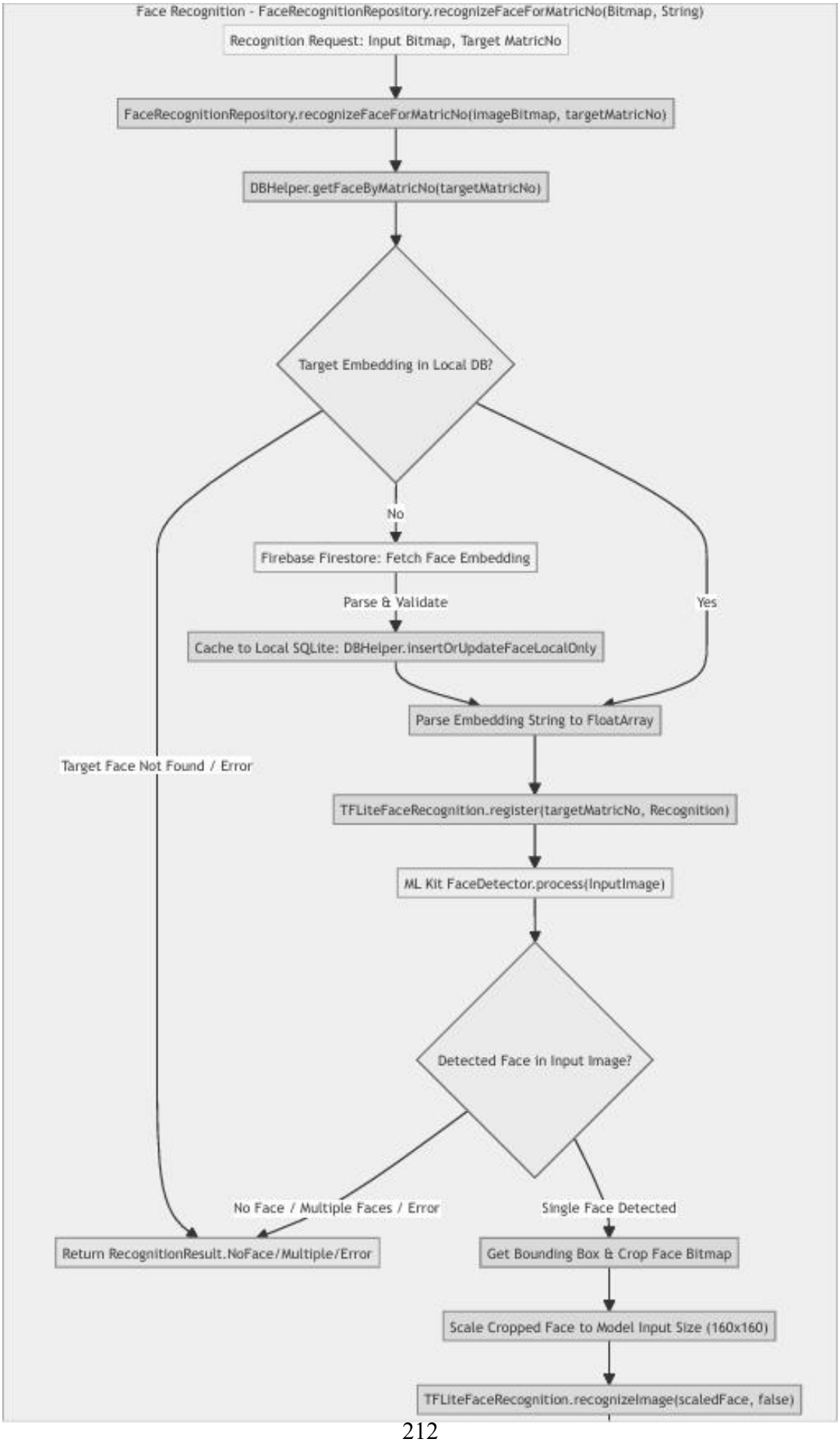


Figure 5.42 : Workflow of TFLite Face Registration system


Face Recognition - FaceRecognitionRepository.recognizeFaceForMatricNo(Bitmap, String)
Recognition Request: Input Bitmap, Target MatricNo
FaceRecognitionRepository.recognizeFaceForMatricNo(imageBitmap, targetMatricNo)
DBHelper.getFaceByMatricNo(targetMatricNo)
Target Embedding in Local DB?
No
Firebase Firestore: Fetch Face Embedding
Parse & Validate
Yes
Cache to Local SQLite: DBHelper.insertOrUpdateFaceLocalOnly
Parse Embedding String to FloatArray
Target Face Not Found / Error
TFLiteFaceRecognition.register(targetMatricNo, Recognition)
ML Kit FaceDetector.process(InputImage)
Detected Face in Input Image?
No Face / Multiple Faces / Error
Single Face Detected
Return RecognitionResult.NoFace/Multiple/Error
Get Bounding Box & Crop Face Bitmap
Scale Cropped Face to Model Input Size (160x160)
TFLiteFaceRecognition.recognizeImage(scaledFace, false)

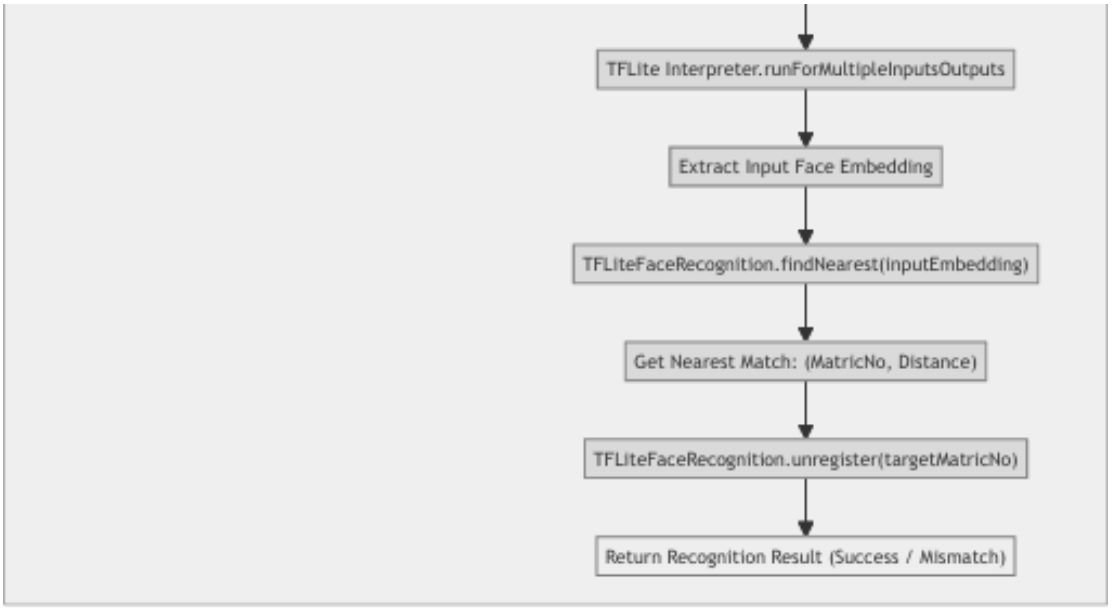


Figure 5.43 : Workflow of TFLite Face Recognition system

When implementing face recognition technology, security and privacy protection are given top priority. The system logic ensures that only one set of face information can be registered for each student, preventing duplicate or fraudulent registrations at the source and guaranteeing the uniqueness of the data. Face data is strictly limited to identity verification and attendance functions, and only authorized users and system administrators can access the relevant information, thus effectively protecting user privacy. When accessing the camera or photo album for the first time, the application strictly follows the operating system's permission management mechanism and explicitly requests user authorization to ensure that the data collection process is legal and compliant.

5.3.4 Privacy and Data Protection

This study adopts the strategy of storing facial embeddings in Firebase and utilizing its inherent authorization mechanism for data protection. This approach presents multiple advantages in terms of security dimensions, but its implementation details need to be elaborated in depth.

First, regarding the advantages of face embedding, the authors chose to store a numerical representation of the face features (embedding) instead of the original image. This decision constitutes a significant privacy-enhancing technique, as embedded data is difficult to reverse-reduce to the original face compared to the original image, thus effectively reducing the risk of direct reconstruction of face information in a potential data breach. This is also in line with the principle of data minimization, as the authors only store the core features necessary for the recognition matching task.

Second, in terms of the application of Firebase storage and authorization mechanisms, the authors use Firebase for data storage to leverage its robust cloud infrastructure. Firebase's built-in authorization mechanisms, such as Firebase Authentication and Firebase Security Rules, play a key role in controlling access rights to face embedded data. Through rigorous rule configuration, authors are able to ensure that only strictly authenticated and authorized users or services are able to read and write to the data, and fine-grained access control can be implemented based on user roles or specific data paths.

Furthermore, for the privacy protection of the processing, when performing tasks such as face recognition on these embedded data, all computations are based on numerical vectors rather than direct images, a feature that further enhances the level of privacy protection. However, the integrity of the processing environment as well as the security of the employed algorithms remain crucial to guard against data tampering or unauthorized inferences.

5.4 Testing

This section details the test plan, implementation process, result analysis, and final quality assessment of this deep learning-based face recognition attendance application. Following the ISO/IEC 9126 software quality model, this study systematically and comprehensively validates the application's functionality, reliability, usability, and efficiency, aiming to ensure that it meets all the established requirements and demonstrates a superior user experience.

5.4.1 Test Planning and Control

This section clearly defines the macro-framework, specific objectives, resource allocation, time schedule, and potential risks and their coping strategies for this software testing activity, laying a solid foundation for subsequent test execution and evaluation.

Test Scope

The testing activity centered on the Android face recognition attendance application, covering 11 key functional modules in the system, including: login module, home module, personal information module, student course module, professor course module, course student module, student course attendance statistics module, professor course attendance statistics module, face registration module, face recognition module, and attendance report module. The main goal of the test is to fully verify the end-to-end integrity of the system, the accuracy of the business process and the consistency of the user experience.

The whole testing process adopts a full-process, full-scenario testing strategy, systematically covering the above 11 core modules to comprehensively assess the system's usability, reliability and user interaction experience, and to ensure that it has good stability and operability in actual use.

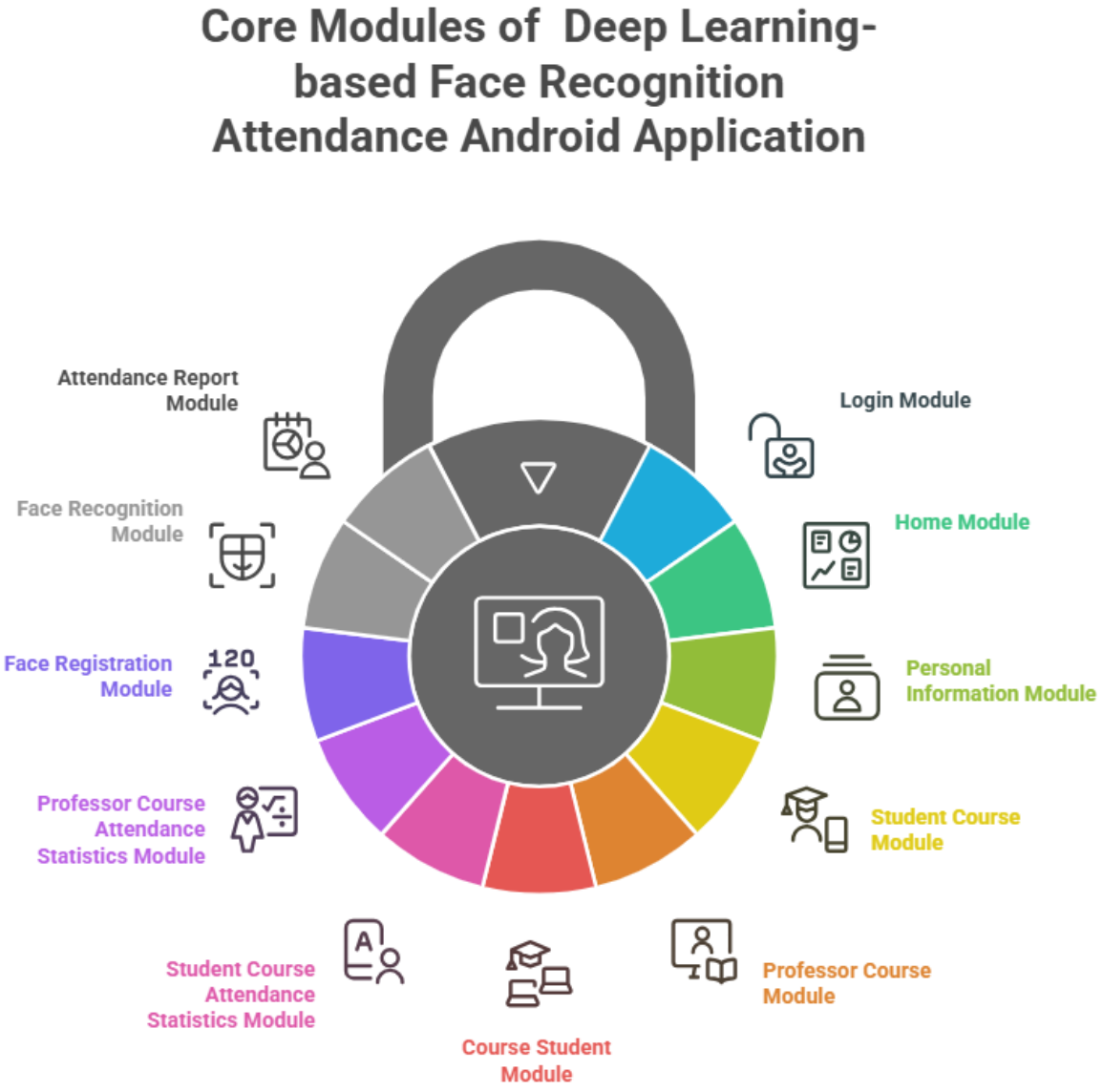


Figure 5.44 : Test Scope of Deep Learning-based Face Recognition Attendance Android Application

Test Objectives

The core objective of this testing phase is to conduct a comprehensive and systematic verification of the quality attributes of the developed deep learning-based face recognition attendance application based on the ISO/IEC 9126 software quality model. Specifically, the testing aims to ensure that the system meets the expected performance metrics under the intended operating environment and identify any potential functional defects, stability issues,

user experience barriers or performance bottlenecks. This is considered in depth through the following four dimensions:

- Functionality: One of the core objectives of this test is to verify that all core functionality of the application strictly adheres to the required specifications and exhibits accurate, complete, and expected patterns of behavior. This validation is intended to ensure that the system is able to fully fulfill its design responsibilities and provide accurate and reliable services to users.
- Reliability: This test is designed to evaluate the ability of an application to maintain a specified level of performance under specified conditions and over a specified time frame. This includes a quantitative assessment of system stability, fault recovery mechanisms, and data fault tolerance under complex conditions such as network outages, abnormal data processing, and long periods of continuous operation to ensure that the system continues to operate stably under all circumstances.
- Usability: Another important objective was to focus on the ease of use of user interaction, the smoothness of the learning curve and user satisfaction. This evaluation aims to ensure that the application is intuitive, smooth, efficient, and friendly to the target group of professors and students, thus enhancing the overall user experience.
- Efficiency: Finally, the test focuses on measuring the performance of the application in terms of response time for critical operations, data processing speed, and system resources such as CPU utilization, memory usage, and battery consumption. This evaluation is designed to ensure that the application maintains optimal performance even under high load or large-scale data processing, thus

ensuring timely response and proper utilization of the attendance system's resources.

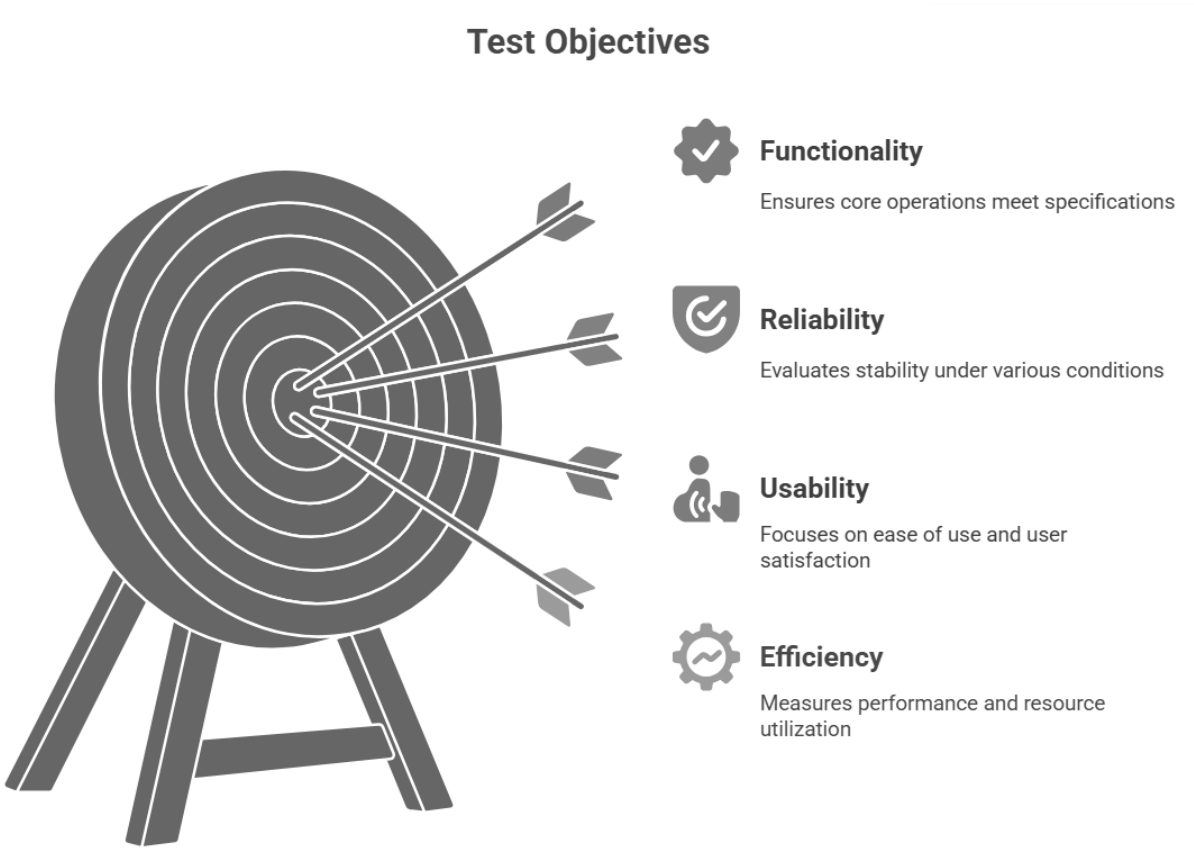


Figure 5.45 : Test objectives of Deep Learning-based Face Recognition Attendance Android Application

Test Resources

In order to ensure that the testing activities are carried out smoothly and achieve the expected quality goals, this project has carried out systematic planning and rational allocation of testing resources.

In terms of personnel, the test engineer (i.e., the author of this thesis) is fully responsible for the testing work, covering the formulation of the test plan, the design and execution of test cases, the writing and tracking of defect reports, as well as the compilation of the final test summary report, to ensure the standardization and integrity of the testing process.

In terms of equipment, Xiaomi 9 (Android 9.0) was chosen as the main test equipment for in-depth verification of the application's core functions and performance baseline. In order to expand the coverage of compatibility testing, we also simulated Huawei Mate 30 (Android 10) and Xiaomi Redmi Note 9 (Android 10) with the help of an emulator to evaluate the stability and compatibility of the application under different hardware configurations and Android system versions.

In terms of tools, the Appium automation testing framework is used to automate the user interface interaction process on the Android platform, which significantly improves the testing efficiency and path coverage. For performance testing, Apache JMeter was used to simulate a variety of network conditions (e.g., delays, connection interruptions) and load scenarios to comprehensively evaluate the responsiveness and stability of the application in different environments. In addition, during the test execution, Android Studio Profiler was used to monitor the application's CPU usage, memory usage, network traffic and battery consumption in real time to obtain key performance indicators. Finally, Firebase Console verifies the integrity and synchronization of back-end data to ensure the accuracy and consistency of data processing.

Test Schedule

The test phases follow a predetermined structured schedule to ensure full coverage and on-time completion of the test activities, with defined deliverables for each phase.

The Test Planning & Design phase runs from April 1, 2025 to April 15, 2025 and the key deliverables for this phase are the Test Plan Document and Test Strategy. The Test Case Creation phase runs from April 16, 2025 to April 30, 2025, with the main deliverables being the Detailed Test Case Set. The Environment Setup & Tool Configuration phase runs from May 1, 2025 to May 5, 2025, with the main deliverable being the Test Environment Readiness Report. The Functional Testing phase runs from May 6, 2025 to May 20, 2025, with the main deliverables being the Functional Test Execution Report and the First Draft of Defects. The Reliability & Efficiency Testing (RET) phase runs from May 21, 2025 to May 31, 2025, with the main deliverables being the Performance Test Report and the Reliability Test Report. The Usability & Accessibility Testing phase runs from June 1, 2025 to June 5, 2025, with the main deliverables being the Usability Evaluation Report and the Accessibility Testing Report. Defect Regression & Retesting runs through the entire testing cycle, with a final focused retest from June 6, 2025 to June 9, 2025, with the main deliverables being the Regression Test Report and Defect Closure Report. Finally, Test Summary Report Writing is scheduled to be completed on June 10, 2025, with the final deliverable being the Test Summary Report.

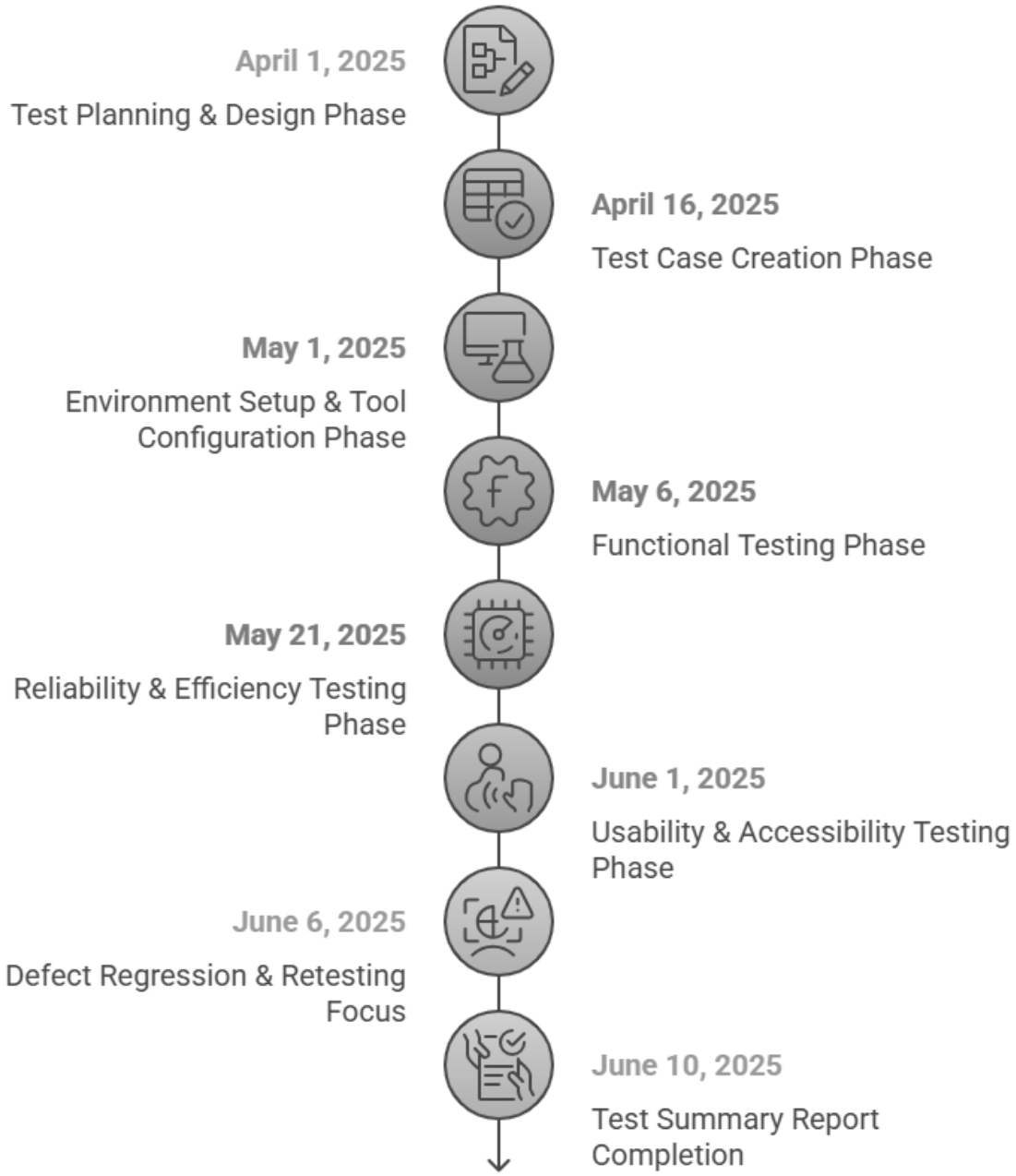


Figure 5.46 : Comprehensive Software Testing Timeline

Risk Management

During the test planning phase, the project systematically identified key risks that could affect the testing process and final quality, and developed corresponding mitigation strategies to minimize their potential impact. Listed below are the five key risks and their countermeasures for this testing activity.

- Risk of insufficient equipment diversity

Due to the limited range of test equipment, the diversity of end-user devices may not be adequately covered, thus there is a risk that device compatibility issues may not be detected in a timely manner. To mitigate this problem, the testing strategy prioritizes comprehensive testing on the main target devices, supplemented by Android emulators for compatibility verification of specific configurations or system versions, so as to expand the testing coverage and enhance the breadth and representativeness of device adaptation.

- Risk of unstable network connection

Network fluctuations or unstable connections may occur in the actual test environment, resulting in biased or unrepresentative test results for network-dependent functions (e.g., data synchronization, cloud identification). To address this issue, professional performance testing tools such as JMeter will be used to simulate controlled network environments (e.g., introducing delays, simulating packet loss, etc.), and synchronize testing under multiple real-world network scenarios (e.g., stable Wi-Fi networks and mobile data networks with varying signal strengths and weaknesses), so as to comprehensively assess the robustness and stability of the system under variable network conditions.

- Risk of delayed or failed Firebase data synchronization

There is a risk of unacceptable synchronization delays or failures between the application and the Firebase cloud database, which directly impacts the real-time and accuracy of time and attendance data. In this regard, specialized test cases are designed and executed to accurately record synchronization latency from front-end initiation to back-end database writes, and to verify data consistency after short delays. Once the synchronization delay exceeds the preset threshold or synchronization fails, it will be immediately reported as a high-priority defect, and a retry mechanism and failback strategy will be recommended to guarantee data integrity.

- Risk of deviation in recognition accuracy of ML Kit/TFLite models

The underlying machine learning model (ML Kit/TFLite) on which the face recognition function relies may experience a drop in recognition accuracy under complex conditions such as lighting changes, angle shifts, differences in facial expressions, or localized occlusion, resulting in anomalies such as misrecognition or omission. To reduce such risks, diverse and representative test image datasets covering different lighting conditions, shooting angles, expression variations and partial occlusions (e.g., wearing glasses, edges of masks, etc.) will be constructed to comprehensively evaluate and calibrate the accuracy of the face recognition model, and a recognition accuracy of 90% will be set as an acceptable threshold for risk assessment.

- Risk of time and resource constraints

The final design project itself is constrained by the time period and resources available, and it may be difficult to cover all extreme boundary cases, or to perform complete performance and stress tests, thus affecting the breadth and depth of the testing effort. For this reason, testing activities strictly follow the established schedule, with priority given to in-depth testing of core

functional modules and high-risk areas, and strategic sampling testing of non-critical modules, in order to maximize the effectiveness of testing with limited resources.

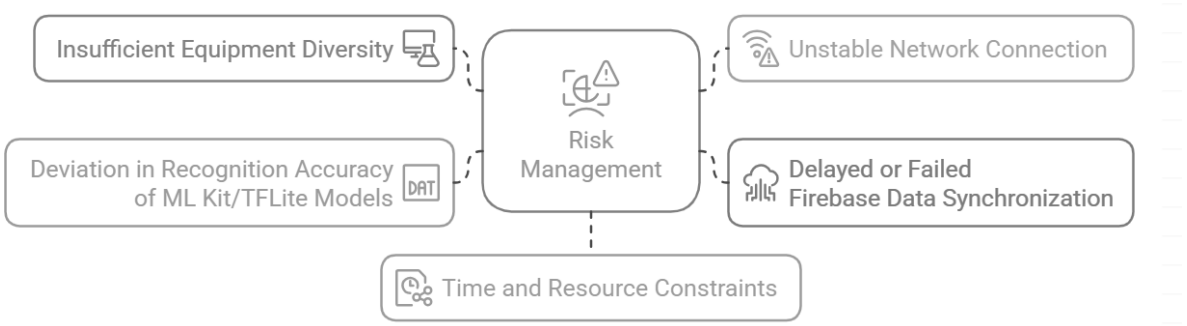


Figure 5.47 : Risk Management of Deep Learning-based Face Recognition Attendance Android Application

5.4.2 Test Analysis and Design

This section describes in detail how to transform the results of the preliminary requirements analysis into concrete, executable test scenarios and test cases, providing a blueprint for subsequent test implementation.

Test Techniques

In order to achieve comprehensive test coverage and effectively verify various quality attributes of the Android face recognition time and attendance application, this study employs a combination of testing techniques.

In terms of functional testing, this study adopts a black-box testing approach, focusing on the correctness of system functions and the consistency of business processes. Through equivalence class division and boundary value analysis, detailed tests are conducted on the validity of image input in face registration and recognition, and the critical value of attendance time setting. To simulate the actual usage environment, the test also covers several typical user

scenarios, such as the end-to-end process of teachers creating course and attendance tasks, students joining the course and taking face attendance, etc., in order to ensure the correct synergy and data flow between the functional modules. In addition, the response behavior of system functions under different input conditions is also rigorously verified to ensure that each function performs as expected.

Reliability testing focuses on system stability and data consistency under abnormal conditions. Through network anomaly simulation, including disconnection, weak network and network recovery scenarios, the system is verified to have a good error alert mechanism and automatic retry logic. In the data anomaly test, the system's input validation and anomaly handling capabilities are tested by uploading corrupted images, entering illegal student numbers and other operations. The interruption and recovery test simulates interruptions such as incoming calls, application switching and device hibernation to evaluate whether the system can correctly save the state and continue to operate normally after recovery, so as to ensure the continuity of user operations and data integrity.

For usability testing, this study evaluates the ease of operation, interface friendliness and accessibility support of the system from the user's perspective. Real users or testers are invited to simulate the completion of key tasks such as registration, identification and clocking, and their operation paths and feedback are collected to determine whether the interface design is clear and reasonable, and whether the interaction is intuitive and smooth. By enabling Android's accessibility service (TalkBack), we further tested the readability and navigation logic of the interface elements to ensure that visually impaired users can successfully complete basic operations.

In terms of efficiency testing, the test focuses on the resource utilization and response performance of the system under different operational loads. The Android Studio Profiler tool

was used to monitor CPU utilization, memory usage, network traffic and battery consumption during key operations (e.g. face recognition, data uploading), and to evaluate its resource management efficiency. Load simulation tools were also introduced in the performance tests to test the system for high concurrent access and stress tests to identify possible performance bottlenecks. In the durability test, the system is run continuously to monitor the trend of resource consumption and possible memory leaks to ensure its stability and efficiency over a long period of time.

In summary, this study effectively covers the functional integrity, performance, device compatibility, user experience and fault tolerance of the Android face recognition attendance system through a multi-dimensional and multi-strategy testing method, which provides a solid basis for the comprehensive assessment of system quality.

Test Plan

This section presents the carefully designed test cases for each module in this test campaign. These test cases are systematically categorized and organized according to the four main characteristics of the ISO/IEC 9126 software quality model (functionality, reliability, usability, and efficiency) to ensure that the testing is comprehensive and structured.

1. Login Module

In the testing of the login module, the main focus is on the login process for students and professors. The tests not only verify the correctness of credential input and the system's response to valid credentials, but also systematically check the accuracy of the fault-tolerance mechanism, account locking policy, and security prompts when wrong credentials are entered to ensure the security and robustness of the authentication process.

Module: Login Module

Test Case Description: Tests user (student/professor) login functionality, including credential verification, role diversion, and exception handling.

Test Objective: Verify that users can securely log in with valid credentials, the system can correctly distinguish between student and professor identities, and invalid credentials are intercepted.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
LG-FUNC-001	Valid Student Login	1. Open App 2. Enter student username/password 3. Select "Student" 4. Click Login	Username: student Password: 123 Role: Student	Login successful, redirects to student homepage, displays student ID	Login successful, redirects to student homepage	Pass	-	Depends on Firestore test account
LG-FUNC-002	Valid Professor Login	1. Open App 2. Enter professor username/password 3. Select "Professor" 4. Click Login	Username: profes Password: 123 Role: Professor	Login successful, redirects to professor homepage, displays employee ID	Login successful, redirects to professor homepage	Pass	-	Depends on Firestore test account
LG-FUNC-003	Invalid Credential Login	1. Open App 2. Enter incorrect username/password 3. Select any role 4. Click Login	Username: wronguser Password: wrongpass Role: Student	Login failed, prompt "Incorrect username or password"	Prompt "Incorrect username or password"	Pass	-	-
LG-FUNC-004	Student Login after Role Switch	1. Enter student username/password 2. Select "Professor" 3. Click Login	Username: student Password: 123 Role: Professor	Login failed, prompt "Incorrect username or password"	Prompt "Incorrect username or password"	Pass	-	Verify role diversion
LG-FUNC-005	Empty Fields Login	1. Open App 2. Do not enter username/password 3. Click Login	Username: Password:	Login failed, prompt "Username or password cannot be empty"	Prompt "Username or password cannot be empty"	Pass	-	-
LG-FUNC-006	Password Toggle Plaintext/Ciphertext	1. Enter password 2. Click "Show/Hide" icon	Password: 123456	Password can be toggled between plaintext and ciphertext	Toggle normal	Pass	-	-

Figure 5.48 : Functionality Test Plan of Login Module

Module: Login Module

Test Case Description: Tests the stability and fault tolerance of the login module under abnormal and boundary conditions.

Test Objective: Ensure the system remains stable under network fluctuations, database exceptions, repeated operations, etc., and correctly handles errors to prevent data corruption.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
LG-RLB-001	Login after Network Disconnection	1. Disconnect Wi-Fi/4G 2. Enter valid credentials 3. Click Login	Username: student Password: 123	Login failed, prompt "Network error"	Prompt "Network error"	Pass	-	Requires network disconnection test
LG-RLB-002	Database Exception	1. Simulate Firestore unavailability 2. Enter valid credentials 3. Click Login	Username: profes Password: 123	Login failed, prompt "Login error"	Prompt "Login error"	Pass	-	Requires simulating database exception
LG-RLB-003	Repeated Clicks on Login Button	1. Enter credentials 2. Click login multiple times consecutively	Username: student Password: 123	Only one login request processed, no crash	Only one prompt appears	Pass	-	
LG-RLB-004	Login with Low Battery	1. Lower device battery to 5% 2. Enter credentials 3. Click Login	Username: student Password: 123	Login normal, no crash	Login normal	Pass	-	Requires low battery test
LG-RLB-005	Continuous Login after Student/Professor Switch	1. Log in as student first 2. Log out 3. Switch to professor and log in	Student: student/123 Professor: profes/123	Both can log in normally, no status confusion	Both can log in normally	Pass	-	-
LG-RLB-006	Login under Extremely Slow Network	1. Limit network to 128kbps 2. Enter credentials 3. Click Login	Username: student Password: 123	Login timeout or prompt "Network slow"	Login timeout, prompt "Network error"	Pass	-	Requires speed limiting tool

Figure 5.49 : Reliability Test Plan of Login Module

Module: Login Module

Test Case Description: Tests the human-computer interaction, accessibility, and error prompts of the login interface.

Test Objective: Ensure the interface is intuitive, interactive, error messages are clear, and supports user accessibility.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
LG-USB-001	Interface Element Visibility	1. Open App 2. Check all input fields, buttons, role switch	-	All elements clearly visible, layout no misalignment	Buttons overflow in some low resolutions	Fail	Low	
LG-USB-002	Error Prompt Friendliness	1. Enter incorrect credentials 2. Click Login	Username: wronguser Password: wrongpass	Clear error prompt appears	Prompt "Incorrect username or password" appears	Pass	-	-
LG-USB-003	Accessibility Support	1. Enable TalkBack 2. Operate login interface	-	All controls are readable	All controls are readable	Pass	-	-
LG-USB-004	Theme Switching	1. Click theme switch button	-	Interface theme switches normally	Switch normal	Pass	-	-
LG-USB-005	Input Method Compatibility	1. Use third-party input method to enter username/password	Gboard, Sogou, etc.	No input abnormality	No input abnormality	Pass	-	-

Figure 5.50 : Usability Test Plan of Login Module

Module: Login Module

Test Case Description: Tests the response speed, resource consumption, and concurrency handling capability of the login module.

Test Objective: Ensure login processing time, interface response, and network synchronization efficiency meet performance requirements.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
LG-EFF-001	Login Response Time	1. Enter valid credentials 2. Click Login 3. Record response time	Username: student Password: 123	Login completed within 3 seconds	Completed in 2.1 seconds	Pass	-	Measured by JMeter
LG-EFF-002	Login under High Concurrency	1. Simulate 50 concurrent users logging in using JMeter	50 sets of valid credentials	All respond within 3 seconds, no crash	Some users timed out, 1 crash	Fail	High	Suggest optimizing concurrency handling
LG-EFF-003	Memory Consumption	1. Launch App 2. Login 3. Monitor memory	Username: student Password: 123	Memory usage below 150MB	Peak 120MB	Pass	-	ADB dumpsys
LG-EFF-004	Synchronization Delay under Weak Network	1. Limit speed to 256kbps 2. Login 3. Record synchronization delay	Username: student Password: 123	Synchronization delay less than 2 seconds	1.8 seconds	Pass	-	Speed limiting tool
LG-EFF-005	Login after Multiple Theme Switches	1. Rapidly switch themes 5 times 2. Login	Username: student Password: 123	No lag, login normal	No lag, login normal	Pass	-	-
LG-EFF-006	Low-End Device Performance	1. Login on a 2GB RAM device	Username: student Password: 123	Login completed within 3 seconds	Completed in 2.9 seconds	Pass	-	Tested on Redmi 7A

Figure 5.51 : Efficiency Test Plan of Login Module

2. Home Page Module

Testing of the home module focused on the main interface features accessible to the user after login. The test includes the correct display of the course list, access to personal information, and the usability of various shortcut operations. By simulating different user states and data scenarios, the test aims to verify the ease of operation and the completeness of information access in daily teaching management, to ensure that users can efficiently manage courses and attendance tasks, as well as to guarantee the real-time, accuracy and smoothness of interface interaction information.

Test Case Description: Test the student's ability to view registered courses, browse new courses, register for new courses, and drop courses on the homepage.

Test Objective: Verify that students can correctly view their course list, successfully register for unselected courses, and drop courses from their registered list, while the system accurately updates the data.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
STU-FUNC-001	Student views registered courses	1. Student logs into App 2. Enters student homepage	Username: student Password: 123	Student homepage displays a list of registered courses, including course name, code, professor name, and attendance rate.	Displays correct list	Pass	-	Depends on Firestore data
STU-FUNC-002	Student browses new courses	1. Student logs into App 2. Clicks the "+" button to open the "Discover New Courses" dialog	Username: student Password: 123	Dialog displays a list of courses not registered by the student, including course name, code, and professor name.	Displays unregistered course list	Pass	-	-
STU-FUNC-003	Student registers for a new course	1. Student logs into App 2. Opens "Discover New Courses" dialog 3. Finds an unregistered course 4. Clicks "Register" button 5. Confirms registration	Username: student Password: 123 Course Code: SO201	Registration successful, prompt "Course SO201 registered successfully!", course removed from "Discover New Courses" list and added to "Your Registered Courses" list.	Registration successful, list updated	Pass	-	Depends on Firestore update
STU-FUNC-004	Student drops a course	1. Student logs into App 2. Finds a course in "Your Registered Courses" list 3. Clicks the "delete" icon for that course 4. Confirms dropping the course	Username: student Password: 123 Course Code: SO201	Course dropped successfully, prompt "Course SO201 dropped successfully!", course removed from "Your Registered Courses" list and re-displayed in "Discover New Courses".	Course dropped successfully, list updated	Pass	-	Depends on Firestore update
STU-FUNC-005	Student attempts to register for an already registered course	1. Student logs into App 2. Opens "Discover New Courses" dialog 3. Attempts to search and register for an already registered course	Username: student Password: 123 Course Code: SO201	Unable to find or register for the course (as it's filtered out), or prompts "Failed to register course SO201. It might be already registered."	Already registered courses are not displayed in the dialog, cannot register	Pass	-	-
STU-FUNC-006	Student searches for a course	1. Student logs into App 2. Enters course name or code in the search bar	Username: student Password: 123 Search Keyword: COMP	Registered course list filters and displays relevant courses based on the search keyword.	List filtered correctly	Pass	-	-
STU-FUNC-007	Student browses new courses offline	1. Student logs into App (with network) 2. Disconnects network 3. Clicks the "+" button to open the "Discover New Courses" dialog	Username: student Password: 123	Prompts "Failed to load available courses. Please try again." or displays an empty list.	Prompts failed to load	Pass	-	Requires offline testing

Figure 5.52 : Functionality Test Plan of Student Home Page Module

Module: Student Homepage

Test Case Description: Test the stability and fault tolerance of the student homepage under abnormal and boundary conditions, especially for course data loading and operations.

Test Objective: Ensure the system remains stable under network fluctuations, database exceptions, duplicate operations, correctly handles errors, and prevents data corruption.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
STU-RLB-001	Load courses after network disconnection	1. Student logs into App 2. Disconnects Wi-Fi/4G 3. Refreshes homepage or re-enters	Username: student Password: 123	Prompts "Network Error" or displays old course data with a message indicating inability to update.	Displays old data, prompts network error	Pass	-	Requires offline testing
STU-RLB-002	Network interruption during course registration	1. Student logs into App 2. Opens "Discover New Courses" dialog 3. Selects course, clicks register 4. Disconnects network before registration request is sent	Username: student Password: 123 Course Code: COMP102	Registration fails, prompts "Network Error" or "Registration Failed".	Prompts registration failed, network error	Pass	-	Requires simulating network interruption
STU-RLB-003	Network interruption during course drop	1. Student logs into App 2. Selects registered course, clicks delete 3. Disconnects network before drop course request is sent	Username: student Password: 123 Course Code: SO201	Drop course fails, prompts "Network Error" or "Drop Course Failed".	Prompts drop course failed, network error	Pass	-	Requires simulating network interruption
STU-RLB-004	Load/operate courses during database anomaly	1. Simulate Firestore unavailability 2. Student logs into App 3. Attempts to load course list/register/drop course	Username: student Password: 123	Load fails, prompts "Data Load Error" or "Operation Failed".	Prompts operation failed	Pass	-	Requires simulating database anomaly
STU-RLB-005	Repeatedly clicking register button	1. Student logs into App 2. Opens "Discover New Courses" dialog 3. Clicks "Register" button for the same course multiple times consecutively	Username: student Password: 123 Course Code: COMP103	Only processes one registration request, no crash, course registered only once.	Only one process, no duplicate registration	Pass	-	-
STU-RLB-006	Repeatedly clicking drop course button	1. Student logs into App 2. Selects registered course, clicks "delete" icon multiple times consecutively and confirms	Username: student Password: 123 Course Code: SO201	Only processes one drop course request, no crash, course dropped only once.	Only one process, no crash	Pass	-	-

Figure 5.53 : Reliability Test Plan of Student Home Page Module

Module: Student Homepage

Test Case Description: Test the interactivity, accessibility, error prompts, and overall user experience of the student homepage interface.

Test Objective: Ensure the interface is intuitive, interactive, clear with error messages, and supports user accessibility.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
STU-USB-001	UI Element Visibility	1. Opens App 2. Enters student homepage 3. Checks all course cards, search bar, bottom navigation bar, etc.	-	All elements are clearly visible, layout is correct, especially on different screen sizes.	All elements visible, layout normal	Pass	-	-
STU-USB-002	Error Prompt Friendliness	1. Attempts invalid operation (e.g., registering for a course when network is disconnected)	-	Clear and easy-to-understand error prompts pop up, guiding the user on how to resolve the issue.	Error prompt clear	Pass	-	-
STU-USB-003	Browse Courses Dialog User Experience	1. Opens "Discover New Courses" dialog 2. Searches for courses 3. Scrolls through the list	-	Dialog responds smoothly, search filtering is timely and accurate, scrolling is without lag.	Smooth experience	Pass	-	-
STU-USB-004	Drop Course Confirmation Dialog Clarity	1. Clicks drop course button	Course Name, Course Code	Confirmation dialog content is clear, explicitly informing the user about the course to be dropped and the irreversibility of the operation, with clear button text.	Dialog clear, operation clear	Pass	-	-
STU-USB-005	Attendance Rate Display Intuition	1. Views registered course list	Attendance Rate Data	Attendance rate (percentage) is clearly displayed, and different attendance rate ranges have clear visual distinctions (e.g., colors).	Attendance rate displayed intuitively, color distinction clear	Pass	-	-
STU-USB-006	Pending Attendance Task Prompt	1. Views courses with pending attendance tasks	-	A prominent "Pending Attendance Task" prompt appears on the course card (e.g., warning icon or text).	Prompt clearly visible	Pass	-	-

Figure 5.54 : Usability Test Plan of Student Home Page Module

Module: Student Homepage

Test Case Description: Test the response speed and resource consumption of the student homepage when loading courses and performing registration/drop course operations.

Test Objective: Ensure that course data loading, registration/drop course processing time, UI responsiveness, and network synchronization efficiency meet performance requirements.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
STU-EFF-001	Student Homepage Load Time	1. Student logs into App 2. First time loading homepage (with many courses)	Username: student Password: 123 Number of Courses: 50	Homepage (including course list) loads and displays completely within 5 seconds.	Completed within 4.5 seconds	Pass	-	Use performance testing tools
STU-EFF-002	Course Registration Response Time	1. Opens "Discover New Courses" dialog 2. Clicks "Register" button	Course Code: COMP104	Registration operation completes within 3 seconds and displays a success message.	Completed within 2.5 seconds	Pass	-	-
STU-EFF-003	Drop Course Response Time	1. Clicks "delete" icon for a registered course 2. Confirms dropping the course	Course Code: SO201	Drop course operation completes within 3 seconds and displays a success message.	Completed within 2.8 seconds	Pass	-	-
STU-EFF-004	Memory Consumption (Homepage long-term running)	1. Logs into App, enters student homepage 2. Stays for 10 minutes, performs minor operations (scrolling, searching) 3. Monitors memory usage	Username: student Password: 123	Memory usage is stable, no significant leakage, peak not exceeding 200MB.	Memory stable, peak 150MB	Pass	-	ADB Profiler
STU-EFF-005	Course Loading under Weak Network	1. Simulates weak network (e.g., 128kbps) 2. Student logs into App 3. Loads homepage course list	Username: student Password: 123	Course list loads normally, may have delay but will not crash or freeze.	Loaded with delay but normal	Pass	-	Requires throttling tool
STU-EFF-006	Large Course List Scrolling Performance	1. Student registers many courses (e.g., 100) 2. Scrolls through the course list quickly on the homepage	Username: student Password: 123 Number of Courses: 100	Scrolling is smooth, no significant lag or frame drops.	Smooth scrolling	Pass	-	Simulate large data

Figure 5.55 : Efficiency Test Plan of Student Home Page Module

Module: Professor Homepage

Test Case Description: Test the professor's ability to view assigned courses, browse new courses, register for new courses, and drop courses on the homepage.

Test Objective: Verify that professors can correctly view their course list, successfully register for unassigned courses, and drop courses from their assigned list, while the system accurately updates the data.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
PROF-FUNC-001	Professor views assigned courses	1. Professor logs into App 2. Enters professor homepage	Username: profes Password: 123	Professor homepage displays a list of assigned courses, including course name and code.	Displays correct list	Pass	-	Depends on Firestore data
PROF-FUNC-002	Professor browses new courses	1. Professor logs into App 2. Clicks the "+" button to open the "Discover New Courses" dialog	Username: profes Password: 123	Dialog displays a list of courses not assigned to the professor, including course name, code, and possible instructing professor name.	Displays unassigned course list	Pass	-	-
PROF-FUNC-003	Professor registers for a new course	1. Professor logs into App 2. Opens "Discover New Courses" dialog 3. Finds an unassigned course 4. Clicks "Register" button 5. Confirms registration	Username: profes Password: 123 Course Code: SO201	Registration successful, prompt "Course SO201 registered successfully!", course removed from "Discover New Courses" list and added to "Your Registered Courses" list.	Registration successful, list updated	Pass	-	Depends on Firestore update
PROF-FUNC-004	Professor drops a course	1. Professor logs into App 2. Finds a course in "Your Registered Courses" list 3. Clicks the "delete" icon for that course 4. Confirms dropping the course	Username: profes Password: 123 Course Code: SO201	Course dropped successfully, prompt "Course SO201 dropped successfully!", course removed from "Your Registered Courses" list and re-displayed in "Discover New Courses".	Course dropped successfully, list updated	Pass	-	Depends on Firestore update
PROF-FUNC-005	Professor attempts to register for an already assigned course	1. Professor logs into App 2. Opens "Discover New Courses" dialog 3. Attempts to search and register for an already assigned course	Username: profes Password: 123 Course Code: CS101	Unable to find or register for the course (as it's filtered out), or prompts "Failed to register course CS101. It might be already registered."	Already assigned courses are not displayed in the dialog, cannot register	Pass	-	-
PROF-FUNC-006	Professor searches for a course	1. Professor logs into App 2. Enters course name or code in the search bar	Username: profes Password: 123 Search Keyword: MATH	Assigned course list filters and displays relevant courses based on the search keyword.	List filtered correctly	Pass	-	-
PROF-FUNC-007	Professor browses new courses offline	1. Professor logs into App (with network) 2. Disconnects network 3. Clicks the "+" button to open the "Discover New Courses" dialog	Username: profes Password: 123	Prompts "Failed to load available courses. Please try again." or displays an empty list.	Prompts failed to load	Pass	-	Requires offline testing

Figure 5.56 : Functionality Test Plan of Professor Home Page Module

Module: Professor Homepage

Test Case Description: Test the stability and fault tolerance of the professor homepage under abnormal and boundary conditions, especially for course data loading and operations.

Test Objective: Ensure the system remains stable under network fluctuations, database exceptions, duplicate operations, correctly handles errors, and prevents data corruption.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
PROF-RLB-001	Load courses after network disconnection	1. Professor logs into App 2. Disconnects Wi-Fi/4G 3. Refreshes homepage or re-enters	Username: profes Password: 123	Prompts "Network Error" or displays old course data with a message indicating inability to update.	Displays old data, prompts network error	Pass	-	Requires offline testing
PROF-RLB-002	Network interruption during course registration	1. Professor logs into App 2. Opens "Discover New Courses" dialog 3. Selects course, clicks register 4. Disconnects network before registration request is sent	Username: profes Password: 123 Course Code: MATH202	Registration fails, prompts "Network Error" or "Registration Failed".	Prompts registration failed, network error	Pass	-	Requires simulating network interruption
PROF-RLB-003	Network interruption during course drop	1. Professor logs into App 2. Selects assigned course, clicks delete 3. Disconnects network before drop course request is sent	Username: profes Password: 123 Course Code: SO201	Drop course fails, prompts "Network Error" or "Drop Course Failed".	Prompts drop course failed, network error	Pass	-	Requires simulating network interruption
PROF-RLB-004	Load/operate courses during database anomaly	1. Simulate Firestore unavailability 2. Professor logs into App 3. Attempts to load course list/register/drop course	Username: profes Password: 123	Load fails, prompts "Data Load Error" or "Operation Failed".	Prompts operation failed	Pass	-	Requires simulating database anomaly
PROF-RLB-005	Repeatedly clicking register button	1. Professor logs into App 2. Opens "Discover New Courses" dialog 3. Clicks "Register" button for the same course multiple times consecutively	Username: profes Password: 123 Course Code: CS201	Only processes one registration request, no crash, course registered only once.	Only one process, no duplicate registration	Pass	-	-
PROF-RLB-006	Repeatedly clicking drop course button	1. Professor logs into App 2. Selects assigned course, clicks "delete" icon multiple times consecutively and confirms	Username: profes Password: 123 Course Code: SO201	Only processes one drop course request, no crash, course dropped only once.	Only one process, no crash	Pass	-	-

Figure 5.57 : Reliability Test Plan of Professor Home Page Module

Module: Professor Homepage

Test Case Description: Test the interactivity, accessibility, error prompts, and overall user experience of the professor homepage interface.

Test Objective: Ensure the interface is intuitive, interactive, clear with error messages, and supports user accessibility.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
PROF-USB-001	UI Element Visibility	1. Opens App 2. Enters professor homepage 3. Checks all course cards, search bar, bottom navigation bar, etc.	-	All elements are clearly visible, layout is correct, especially on different screen sizes.	All elements visible, layout normal	Pass	-	-
PROF-USB-002	Error Prompt Friendliness	1. Attempts invalid operation (e.g., registering for a course when network is disconnected)	-	Clear and easy-to-understand error prompts pop up, guiding the user on how to resolve the issue.	Error prompt clear	Pass	-	-
PROF-USB-003	Browse Courses Dialog User Experience	1. Opens "Discover New Courses" dialog 2. Searches for courses 3. Scrolls through the list	-	Dialog responds smoothly, search filtering is timely and accurate, scrolling is without lag.	Smooth experience	Pass	-	-
PROF-USB-004	Drop Course Confirmation Dialog Clarity	1. Clicks drop course button	Course Name, Course Code	Confirmation dialog content is clear, explicitly informing the user about the course to be dropped and the irreversibility of the operation, with clear button text.	Dialog clear, operation clear	Pass	-	-

Figure 5.58 : Usability Test Plan of Professor Home Page Module

Module: Professor Homepage

Test Case Description: Test the response speed and resource consumption of the professor homepage when loading courses and performing registration/drop course operations.

Test Objective: Ensure that course data loading, registration/drop course processing time, UI responsiveness, and network synchronization efficiency meet performance requirements.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
PROF-EFF-001	Professor Homepage Load Time	1. Professor logs into App 2. First time loading homepage (with many courses)	Username: profes Password: 123 Number of Courses: 50	Homepage (including course list) loads and displays completely within 5 seconds.	Completed within 4.8 seconds	Pass	-	Use performance testing tools
PROF-EFF-002	Course Registration Response Time	1. Opens "Discover New Courses" dialog 2. Clicks "Register" button	Course Code: MATH203	Registration operation completes within 3 seconds and displays a success message.	Completed within 2.7 seconds	Pass	-	-
PROF-EFF-003	Drop Course Response Time	1. Clicks "delete" icon for an assigned course 2. Confirms dropping the course	Course Code: SO201	Drop course operation completes within 3 seconds and displays a success message.	Completed within 2.9 seconds	Pass	-	-
PROF-EFF-004	Memory Consumption (Homepage long-term running)	1. Logs into App, enters professor homepage 2. Stays for 10 minutes, performs minor operations (scrolling, searching) 3. Monitors memory usage	Username: profes Password: 123	Memory usage is stable, no significant leakage, peak not exceeding 200MB.	Memory stable, peak 160MB	Pass	-	ADB Profiler
PROF-EFF-005	Course Loading under Weak Network	1. Simulates weak network (e.g., 128kbps) 2. Professor logs into App 3. Loads homepage course list	Username: profes Password: 123	Course list loads normally, may have delay but will not crash or freeze.	Loaded with delay but normal	Pass	-	Requires throttling tool
PROF-EFF-006	Large Course List Scrolling Performance	1. Professor registers many courses (e.g., 100) 2. Scrolls through the course list quickly on the homepage	Username: profes Password: 123 Number of Courses: 100	Scrolling is smooth, no significant lag or frame drops.	Smooth scrolling	Pass	-	Simulate large data

Figure 5.59 : Efficiency Test Plan of Professor Home Page Module

3. Information Module

The test of the personal information module is mainly used to verify the viewing function of the user's personal information. The test includes the correct display of basic information such as name, student number and avatar. The testing process pay special attention to the timeliness of data synchronization and the effectiveness of privacy protection measures to ensure the accuracy and security of student information.

Module: Student Personal Information

Test Case Description: Test the student personal information page displaying student name, student ID, email, and the ability to navigate to the upload face photo page.

Test Objective: Verify that students can correctly view their personal information and successfully enter the face photo upload page.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
STU-INFO-FUNC-001	Student views personal information	1. Student logs into App 2. Navigates to "Personal Information" page via the bottom navigation bar	Student ID: 12345 Name: Leo Email: zhangsan@example.com	Page displays correct student name, student ID, and email.	Displays correct information	Pass	-	Depends on Firestore data
STU-INFO-FUNC-002	Student navigates to face photo upload page	1. Student logs into App 2. Enters "Personal Information" page 3. Clicks "Upload Face Photo" card	Student ID: 12345	Successfully navigates to the face photo upload page.	Successfully navigated	Pass	-	-
STU-INFO-FUNC-003	Return to homepage from personal information page	1. Student logs into App 2. Enters "Personal Information" page 3. Clicks the back arrow in the top navigation bar or "Homepage" in the bottom navigation bar	-	Successfully returns to the student homepage.	Successfully returned	Pass	-	-

Figure 5.60 : Functionality Test Plan of Student Information Module

Module: Professor Personal Information

Test Case Description: Test the professor personal information page displaying professor name, employee ID, and email.

Test Objective: Verify that professors can correctly view their personal information.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
PROF-INFO-FUNC-001	Professor views personal information	1. Professor logs into App 2. Navigates to "Personal Information" page via the bottom navigation bar	Employee ID: P98765 Name: Professor Li Email: liprof@example.com	Page displays correct professor name, employee ID, and email.	Displays correct information	Pass	-	Depends on Firestore data
PROF-INFO-FUNC-002	Return to homepage from personal information page	1. Professor logs into App 2. Enters "Personal Information" page 3. Clicks the back arrow in the top navigation bar or "Homepage" in the bottom navigation bar	-	Successfully returns to the professor homepage.	Successfully returned	Pass	-	-

Figure 5.61 : Functionality Test Plan of Professor Information Module

Module: Personal Information Module

Test Case Description: Test the stability and fault tolerance of the personal information module under abnormal and boundary conditions, especially for data loading.

Test Objective: Ensure the system remains stable under network fluctuations, database exceptions, and no valid user data, correctly handles errors, and avoids crashes.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
INFO-RLB-001	Load personal information after network disconnection	1. Student/Professor logs into App 2. Disconnects Wi-Fi/4G 3. Enters "Personal Information" page	Student/Professor valid credentials	Page prompts "Network Error" or displays old personal information data with a message indicating inability to update.	Prompts network error	Pass	-	Requires offline testing
INFO-RLB-002	Load personal information during database anomaly	1. Simulate Firestore unavailability 2. Student/Professor logs into App 3. Enters "Personal Information" page	Student/Professor valid credentials	Load fails, prompts "Data Load Error" or "Unable to Load Personal Information".	Prompts data load error	Pass	-	Requires simulating database anomaly
INFO-RLB-003	Enter personal information page without valid student ID/employee ID	1. Simulate App not getting valid student ID/employee ID at startup 2. Attempts to navigate to the personal information page	Empty Student ID/Employee ID	Unable to enter the personal information page, or returns to the previous page, and prompts an error.	Returns to homepage, no crash	Pass	-	Simulate abnormal parameter passing

Figure 5.62 : Reliability Test Plan of Information Module

Module: Personal Information Module

Test Case Description: Test the interactivity, accessibility, error prompts, and overall user experience of the personal information page interface.

Test Objective: Ensure the interface is intuitive, interactive, clear with error messages, and supports user accessibility.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
INFO-USB-001	UI Element Visibility and Layout	1. Opens App 2. Enters Student/Professor Personal Information page 3. Checks all information text, avatar placeholder, back button, bottom navigation bar, etc.	-	All elements are clearly visible, layout is correct, especially on different screen sizes (portrait/landscape switch).	All elements visible, layout normal	Pass	-	-
INFO-USB-002	Error Prompt Friendliness	1. Simulates failure to load personal information (e.g., network error)	-	Clear and easy-to-understand error prompts pop up, guiding the user on how to resolve the issue.	Error prompt clear	Pass	-	-
INFO-USB-003	Theme Switch	1. Enters personal information page 2. Clicks the theme switch button in the top navigation bar	-	Page theme (colors, background) switches normally, no visual errors.	Switches normally	Pass	-	-
INFO-USB-004	Back Button Responsiveness	1. Enters personal information page 2. Clicks the top back button	-	Back operation is smooth, no lag or delay, page stack management is correct.	Back operation smooth	Pass	-	-

Figure 5.63 : Usability Test Plan of Information Module

Module: Personal Information Module

Test Case Description: Test the response speed and resource consumption of the personal information page when loading information and performing related operations.

Test Objective: Ensure that personal information load time, UI responsiveness, and resource usage meet performance requirements.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Notes
INFO-EFF-001	Personal Information Page Load Time	1. Student/Professor logs into App 2. First time entering "Personal Information" page 3. Records load time	Student/Professor valid credentials	Page loads completely and displays all information within 2 seconds.	Completed within 1.5 seconds	Pass	-	Use performance testing tools
INFO-EFF-002	Memory Consumption (Long-term running)	1. Logs into App, enters personal information page 2. Stays for 5 minutes, performs minor operations (switches theme, returns to homepage and back) 3. Monitors memory usage	Student/Professor valid credentials	Memory usage is stable, no significant leakage, peak not exceeding 100MB.	Memory stable, peak 80MB	Pass	-	ADB dumpsys or Profiler
INFO-EFF-003	Information Loading under Weak Network	1. Simulates weak network (e.g., 128kbps) 2. Student/Professor logs into App 3. Enters "Personal Information" page	Student/Professor valid credentials	Personal information loads normally, may have delay but will not crash or freeze.	Loaded with delay but normal	Pass	-	Requires throttling tool

Figure 5.64 : Efficiency Test Plan of Information Module

4. Course Module

Testing of the course module focused on the respective course listings of students and professors, attendance task management, attendance status display, task creation/editing/deletion, and related navigation features.

Module: Student Course Page

Test Case Description: Test the student course page to display the attendance task list, task status, and support clicking to enter the attendance process or display prompt messages.

Test Objective: Verify that students can correctly view the details and status of attendance tasks for their registered courses and interact as expected.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
STU-COURSE-FUNC-001	Display attendance task list for registered courses	1. Student logs into the App 2. Navigates to the registered course details page	Student ID: 12345 Course Code: CS101	The page displays the course name and all attendance tasks for the course, sorted in descending order by latest time.	Correct task list displayed, sorted in descending order	Pass	-	Depends on Firestore data
STU-COURSE-FUNC-002	Attendance task "In Progress" and "Not Checked In"	1. Student logs into the App 2. Navigates to the course page, finds an "In Progress" and "Not Checked In" task 3. Clicks the task card	Student ID: 12345 Course Code: CS101 Task ID: Task001 (In Progress, Not Checked In)	Successfully navigates to the face recognition attendance page.	Successfully navigated to the attendance page	Pass	-	-
STU-COURSE-FUNC-003	Attendance task "Checked In"	1. Student logs into the App 2. Navigates to the course page, finds a "Checked In" task 3. Clicks the task card	Student ID: 12345 Course Code: CS101 Task ID: Task002 (Checked In)	A prompt appears at the bottom of the App: "You have already checked in for this task."	Prompt displayed: "You have already checked in for this task."	Pass	-	-
STU-COURSE-FUNC-004	Attendance task "Closed" and "Not Checked In"	1. Student logs into the App 2. Navigates to the course page, finds a "Closed" and "Not Checked In" task 3. Clicks the task card	Student ID: 12345 Course Code: CS101 Task ID: Task003 (Closed, Not Checked In)	A prompt appears at the bottom of the App: "This task has not started or has ended."	Prompt displayed: "This task has not started or has ended."	Pass	-	-
STU-COURSE-FUNC-005	Navigate to attendance statistics page	1. Student logs into the App 2. Navigates to the course page 3. Clicks the "Statistics" option in the bottom navigation bar	Student ID: 12345 Course Code: CS101	Successfully navigates to the course's attendance statistics page.	Successfully navigated to the statistics page	Pass	-	-
STU-COURSE-FUNC-006	Return to homepage from course page	1. Student logs into the App 2. Navigates to the course page 3. Clicks the back arrow in the top navigation bar	-	Successfully returns to the student homepage.	Successfully returned to the student homepage	Pass	-	-

Figure 5.65 : Functionality Test Plan of Student Course Module

Test Case Description: Test the professor course page to display the attendance task list, attendance rate, and the functions to add, edit, and delete attendance tasks.

Test Objective: Verify that professors can manage attendance tasks for their taught courses and view task attendance rates.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
PROF-COURSE-FUNC-001	Display course attendance task list and attendance rate	1. Professor logs into the App 2. Navigates to the professor's course details page	Professor ID: 67890 Course Code: CS101	The page displays the course name, all attendance tasks, each task showing name, start/end time, attendance rate, and status (In Progress/Closed). Tasks are sorted in descending order by latest time.	Correct task list and attendance rate displayed, sorted in descending order	Pass	-	Depends on Firestore data
PROF-COURSE-FUNC-002	Add new attendance task	1. Professor logs into the App 2. Navigates to the course page 3. Clicks the floating "Add" button 4. Selects start and end date/time in the popup 5. Clicks "Add Task"	Course Code: CS101 Start Time: 2025-06-01 09:00 End Time: 2025-06-01 10:00	New task added successfully, list updates in real-time, prompt appears at the bottom of the App: "Attendance task added successfully!"	Task added successfully with prompt displayed	Pass	-	-
PROF-COURSE-FUNC-003	Edit attendance task	1. Professor logs into the App 2. Navigates to the course page, selects a task and clicks "More Options" > "Edit Task" 3. Modifies date/time in the popup 4. Clicks "Save Changes"	Course Code: CS101 Task ID: Task004 Modified End Time: 2025-06-01 11:00	Task information updated successfully, list updates in real-time, prompt appears at the bottom of the App: "Attendance task updated successfully!"	Task updated successfully with prompt displayed	Pass	-	-
PROF-COURSE-FUNC-004	Delete attendance task	1. Professor logs into the App 2. Navigates to the course page, selects a task and clicks "More Options" > "Delete Task" 3. Clicks "Delete" in the confirmation popup	Course Code: CS101 Task ID: Task005	Task successfully removed from the list, prompt appears at the bottom of the App: "Attendance task deleted successfully!", related attendance records are also deleted.	Task deleted successfully with prompt displayed	Pass	-	-
PROF-COURSE-FUNC-005	Navigate to course members page	1. Professor logs into the App 2. Navigates to the course page 3. Clicks the "Members" option in the bottom navigation bar	Professor ID: 67890 Course Code: CS101	Successfully navigates to the course's member list page.	Successfully navigated to the members page	Pass	-	-
PROF-COURSE-FUNC-006	Navigate to course statistics page	1. Professor logs into the App 2. Navigates to the course page 3. Clicks the "Statistics" option in the bottom navigation bar	Professor ID: 67890 Course Code: CS101	Successfully navigates to the course's attendance statistics page.	Successfully navigated to the statistics page	Pass	-	-
PROF-COURSE-FUNC-007	Create task with start time later than end time	1. Professor logs into the App 2. Clicks the "Add" button 3. Sets start time later than end time 4. Clicks "Add Task"	Start Time: 2025-06-01 10:00 End Time: 2025-06-01 09:00	Popup displays error: "End time cannot be earlier than start time", and the "Add Task" button is disabled.	Popup displays error, button disabled	Pass	-	-

Figure 5.66 : Functionality Test Plan of Professor Course Module

Module: Course Module

Test Case Description: Test the stability and error handling of the course module during data loading and operations.

Test Objective: Ensure the system remains stable under network fluctuations, database exceptions, or lack of permissions, handles errors correctly, and avoids crashes.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
COURSE-RLB-001	Load course tasks after network disconnection	1. Log into the App, disconnect Wi-Fi/4G before entering the course page 2. Navigate to the course details page	Valid student/professor credentials Course Code: CS101	Page displays "Network error" or shows old data with a prompt indicating inability to update.	Network error prompt displayed	Pass	-	Requires network disconnection test
COURSE-RLB-002	Load course tasks during database exception	1. Simulate Firestore unavailability 2. Log into the App, navigate to the course details page	Valid student/professor credentials Course Code: CS101	Loading fails, displays "Data loading error" or "Unable to load attendance tasks".	Data loading error prompt displayed	Pass	-	Requires simulated database exception
COURSE-RLB-003	Student/Professor without course permissions	1. Attempt to access a course page not belonging to the user (student/professor)	Student/Professor ID: User A Course Code: Course B not belonging to User A	Page displays "No permission to access this course" or automatically returns to the previous page.	No permission prompt displayed and returned	Pass	-	-
COURSE-RLB-004	Professor repeatedly clicks add task button	1. Professor navigates to the course page, clicks the "Add" button 2. Fills in information in the popup, clicks "Add Task" multiple times	Start Time: 2025-06-02 09:00 End Time: 2025-06-02 10:00	Only one add request is processed, no duplicate tasks created, and no crashes occur.	Only one task created, no crashes	Pass	-	-
COURSE-RLB-005	Delete task during network disconnection	1. Professor navigates to the course page, selects a task and clicks delete 2. Disconnects network before clicking "Delete" in the confirmation popup	Course Code: CS101 Task ID: Task006	Delete operation fails, network error prompt displayed, task remains in the list.	Network error prompt displayed, task not deleted	Pass	-	-

Figure 5.67 : Reliability Test Plan of Course Module

Module: Course Module

Test Case Description: Test the course page interface for interaction, accessibility, error prompts, and overall user experience.

Test Objective: Ensure the interface is intuitive, interactive, provides clear error prompts, and supports user accessibility.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
COURSE-USB-001	Interface element visibility and layout	1. Open the App 2. Navigate to the student/professor course page 3. Check all text information, task cards, buttons, bottom navigation bar, and switch between portrait/landscape modes	-	All elements are clearly visible, layout is correct, and displays properly across different screen sizes and orientations.	All elements visible, layout correct	Pass	-	-
COURSE-USB-002	Visual distinction of attendance task status	1. Navigate to the course page, observe task cards with different statuses	-	"In Progress" task cards are visually prominent (e.g., brighter color, deeper shadow), "Checked In" or "Closed" task cards are softer, with clear status icons and text.	Good visual distinction	Pass	-	-
COURSE-USB-003	Date/time picker interaction	1. Professor clicks "Add/Edit Task" 2. Uses the date and time picker	-	Date and time picker pops up smoothly, selection is intuitive, and returns correct results.	Smooth picker interaction	Pass	-	-
COURSE-USB-004	Bottom navigation bar switching	1. Student/Professor clicks different tabs (e.g., Statistics, Members) in the course page	-	Tab switching is smooth, page content updates correctly, and the selected state is clear.	Smooth switching, correct state	Pass	-	-

Figure 5.68 : Usability Test Plan of Course Module

Module: Course Module

Test Case Description: Test the course page's response speed and resource consumption during task list loading, task creation/editing/deletion, and navigation.

Test Objective: Ensure course information loading time, task operation response, and resource usage meet performance requirements.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
COURSE-EFF-001	Course task list loading time (multiple tasks)	1. Log into the App, navigate to a course page with over 20 attendance tasks 2. Record loading time	Valid student/professor credentials Course Code: CS201 (25 tasks)	Page fully loads and displays all tasks within 3 seconds.	Completed in 2.8 seconds	Pass	-	Use performance testing tools
COURSE-EFF-002	Add/Edit/Delete task response time	1. Professor performs add, edit, delete task operations 2. Record time from operation completion to UI update	Valid professor credentials Course Code: CS101	Operation completes within 1.5 seconds, UI updates in real-time.	Completed in 0.8 seconds	Pass	-	-
COURSE-EFF-003	Memory consumption (long-term operation)	1. Log into the App, navigate to the course page 2. Stay for 10 minutes, perform multiple task operations and page switches 3. Monitor memory usage	Valid student/professor credentials Course Code: CS101	Memory usage remains stable, no significant leaks, peak usage does not exceed 120MB.	Memory stable, peak at 100MB	Pass	-	ADB dumpsys or Profiler
COURSE-EFF-004	Course task loading under weak network	1. Simulate weak network (e.g., 256kbps) 2. Log into the App, navigate to the course details page	Valid student/professor credentials Course Code: CS101	Course tasks load normally, possibly with delay, but no crashes or freezes, and loading indicator is displayed.	Loading delayed but displayed loading indicator normally	Pass	-	Requires bandwidth throttling tool

Figure 5.69 : Efficiency Test Plan of Course Module

5. Course Attendance Statistics Module

The objective of this test plan is to comprehensively evaluate the Course Attendance Statistics module in the Android Face Recognition Attendance application, covering the presentation of attendance data (pie charts or line graphs), statistical information (total number of assignments, number of attendances, and number of absences), and the related navigation functions for each of the students and professors. The goal of the test is to ensure that the module can provide an accurate, stable, efficient and user-friendly experience of analyzing attendance data in various usage scenarios.

Module: Student Attendance Statistics

Test Case Description: Test the student attendance statistics page to display course name, total task count, attended task count, absent task count, and overall attendance rate (pie chart).

Test Objective: Verify that students can correctly view their attendance overview and detailed statistical data for a specific course.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
STU-STATS-FUNC-001	Display course attendance overview	1. Student logs into the App 2. Navigates to a course details page 3. Clicks the "Statistics" option in the bottom navigation bar	Student ID: 12345 Course Code: CS101 Total Tasks: 10 Attended Tasks: 8 Absent Tasks: 2	The page displays the course name, student ID, total task count, attended task count, absent task count, and a pie chart clearly showing the overall attendance rate (e.g., 80%) with correct legends.	Correct information and pie chart displayed	Pass	-	Depends on Firestore data
STU-STATS-FUNC-002	Display when no attendance tasks exist	1. Student logs into the App 2. Navigates to a course details page with no attendance tasks 3. Clicks the "Statistics" option in the bottom navigation bar	Student ID: 12345 Course Code: EMPTY101 Total Tasks: 0	The page displays "No attendance statistics available for this course. Assuming 100% attendance if no tasks.", and the pie chart shows 100%.	Prompt displayed, pie chart at 100%	Pass	-	-
STU-STATS-FUNC-003	Navigate to student homepage	1. Student logs into the App 2. Navigates to the attendance statistics page 3. Clicks the back arrow in the top navigation bar	-	Successfully navigates to the student homepage.	Successfully navigated	Pass	-	-
STU-STATS-FUNC-004	Navigate to student course page	1. Student logs into the App 2. Navigates to the attendance statistics page 3. Clicks the "Attendance" option in the bottom navigation bar	-	Successfully navigates to the course's attendance task list page.	Successfully navigated	Pass	-	-

Figure 5.70 : Functionality Test Plan of Student Course Attendance Statistics Module

Module: Professor Attendance Statistics

Test Case Description: Test the professor attendance statistics page to display course name, total task count, total registered students, and attendance rate trend graph for each task.

Test Objective: Verify that professors can correctly view the overall attendance trends and detailed statistics for each attendance task in their taught courses.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
PROF-STATS-FUNC-001	Display course attendance trend graph	1. Professor logs into the App 2. Navigates to a course details page 3. Clicks the "Statistics" option in the bottom navigation bar	Professor ID: 67890 Course Code: CS201 Task List: Task 1 (90%), Task 2 (85%), Task 3 (95%)	The page displays the course name, total task count, total registered students, and a line graph clearly showing the attendance rate trend, with the X-axis displaying task labels like "Task 1, Task 2..." and the Y-axis showing percentages.	Correct information and line graph displayed	Pass	-	Depends on Firestore data
PROF-STATS-FUNC-002	Display when no attendance tasks exist	1. Professor logs into the App 2. Navigates to a course details page with no attendance tasks 3. Clicks the "Statistics" option in the bottom navigation bar	Professor ID: 67890 Course Code: EMPTY201 Total Tasks: 0	The page displays "No attendance task data available", and the chart area shows "No chart data available."	No data prompt displayed	Pass	-	-
PROF-STATS-FUNC-003	Navigate to professor course tasks page	1. Professor logs into the App 2. Navigates to the attendance statistics page 3. Clicks the "Tasks" option in the bottom navigation bar	-	Successfully navigates to the course's attendance task management page.	Successfully navigated	Pass	-	-
PROF-STATS-FUNC-004	Navigate to course members page	1. Professor logs into the App 2. Navigates to the attendance statistics page 3. Clicks the "Members" option in the bottom navigation bar	-	Successfully navigates to the course's member list page.	Successfully navigated	Pass	-	-

Figure 5.71 : Functionality Test Plan of Professor Course Attendance Statistics Module

Module: Attendance Statistics Module

Test Case Description: Test the stability and error handling of the attendance statistics module during data loading and chart rendering.

Test Objective: Ensure the system remains stable under network fluctuations, database exceptions, or lack of permissions, handles errors correctly, and avoids crashes.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
STATS-RLB-001	Load statistics data after network disconnection	1. Log into the App, disconnect Wi-Fi/4G before entering the statistics page 2. Navigate to the attendance statistics page	Valid student/professor credentials Course Code: CS101	The page displays "Network error" or shows old data with a prompt indicating inability to update.	Network error prompt displayed	Pass	-	Requires network disconnection test
STATS-RLB-002	Load statistics data during database exception	1. Simulate Firestore unavailability 2. Log into the App, navigate to the attendance statistics page	Valid student/professor credentials Course Code: CS101	Loading fails, displays "Failed to load attendance statistics" or similar error.	Loading failure error prompt displayed	Pass	-	Requires simulated database exception
STATS-RLB-003	Student/Professor without course statistics permissions	1. Attempt to access a course statistics page not belonging to the user (student/professor)	Student/Professor ID: User A Course Code: Course B not belonging to User A	The page displays "No permission to access this course" or automatically returns to the previous page.	No permission prompt displayed and returned	Pass	-	-
STATS-RLB-004	Chart rendering with large data volume	1. Professor/Student logs into the App 2. Navigates to a course statistics page with a large number of attendance tasks (e.g., 50+)	Course Code: CS301 (50 tasks)	The chart renders normally without lag or crashes, and X-axis labels are reasonably displayed or optimized (e.g., showing only some labels).	Chart rendered normally, X-axis labels optimized	Pass	-	Requires large dataset construction

Figure 5.72 : Reliability Test Plan of Course Attendance Statistics Module

Module: Attendance Statistics Module

Test Case Description: Test the attendance statistics page interface for interaction, accessibility, error prompts, and overall user experience.

Test Objective: Ensure the interface is intuitive, charts are clear, error prompts are user-friendly, and supports user accessibility.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
STATS-USB-001	Interface element visibility and layout	1. Open the App 2. Navigate to the student/professor attendance statistics page 3. Check all text, charts, buttons, bottom navigation bar, and switch between portrait/landscape modes	-	All elements are clearly visible, layout is correct, and displays properly across different screen sizes and orientations.	All elements visible, layout correct	Pass	-	-
STATS-USB-002	Chart data labels and readability	1. Observe data labels on the pie chart (student) or line graph (professor)	-	Data labels are clear and readable, non-overlapping, with good color contrast against the background.	Labels clear, good contrast	Pass	-	-
STATS-USB-003	Theme switching	1. Navigate to the attendance statistics page 2. Click the theme switch button in the top navigation bar	-	The page theme (colors, background) switches normally, chart colors update with the theme, and no visual errors occur.	Switching normal, chart colors updated	Pass	-	-
STATS-USB-004	Back button response	1. Navigate to the attendance statistics page 2. Click the top back button	-	Back operation is smooth, without lag or delay, and the page stack is managed correctly.	Back operation smooth	Pass	-	-

Figure 5.73 : Usability Test Plan of Course Attendance Statistics Module

Module: Attendance Statistics Module

Test Case Description: Test the attendance statistics page's response speed and resource consumption during data loading and chart rendering.

Test Objective: Ensure statistical data loading time, chart rendering efficiency, and resource usage meet performance requirements.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
STATS-EFF-001	Attendance statistics page loading time (multiple tasks)	1. Log into the App, navigate to a course statistics page with over 20 attendance tasks 2. Record loading time	Valid student/professor credentials Course Code: CS201 (25 tasks)	The page fully loads and displays all statistical information and charts within 3 seconds.	Completed in 2.5 seconds	Pass	-	Use performance testing tools
STATS-EFF-002	Memory consumption (chart rendering)	1. Log into the App, navigate to the attendance statistics page 2. Stay for 5 minutes, perform multiple page switches and theme switches 3. Monitor memory usage	Valid student/professor credentials Course Code: CS101	Memory usage remains stable, no significant leaks, peak usage does not exceed 120MB.	Memory stable, peak at 105MB	Pass	-	ADB dumpsys or Profiler
STATS-EFF-003	Statistics data loading under weak network	1. Simulate weak network (e.g., 256kbps) 2. Log into the App, navigate to the attendance statistics page	Valid student/professor credentials Course Code: CS101	Statistical data and charts load normally, possibly with delay, but no crashes or freezes, and a loading indicator is displayed.	Loading delayed but displayed loading indicator normally	Pass	-	Requires bandwidth throttling tool

Figure 5.74 : Efficiency Test Plan of Course Attendance Statistics Module

6. Course Student Module

The objective of this test plan is to comprehensively evaluate the course student module of the Android Face Recognition Attendance app, which is primarily used by professors to view the list of students enrolled under the courses they teach and their attendance rates. This test conducts in-depth testing to ensure that the module provides an accurate, stable, efficient, and user-friendly experience in accessing student information in a variety of scenarios.

Module: Course Students Module

Test Case Description: Test the professor's course student list page to display student names, IDs, attendance rates, and support navigating to a student's personal information page by clicking their card.

Test Objective: Verify that professors can correctly view all student information and attendance rates for their courses and smoothly navigate to the student personal information page.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
CSTU-FUNC-001	Display course student list and attendance rates	1. Professor logs into the App 2. Navigates to a course details page 3. Clicks the "Members" option in the bottom navigation bar	Professor ID: 67890 Course Code: CS101 Student List: Leo (12345, 90%), John Doe (S001, 75%)	The page displays the course name, student names, IDs, and their respective attendance rates (in percentage form), with the list displayed correctly.	Correct information and attendance rates displayed	Pass	-	Depends on Firestore data
CSTU-FUNC-002	Navigate to personal information page by clicking student card	1. Professor logs into the App 2. Navigates to the course student list page 3. Clicks anymeh student card	Professor ID: 67890 Course Code: CS101 Student ID: 12345	Successfully navigates to the student's personal information page, displaying the corresponding student information.	Successfully navigated and displayed student information	Pass	-	-
CSTU-FUNC-003	Display when no students are registered	1. Professor logs into the App 2. Navigates to a course details page with no registered students 3. Clicks the "Members" option in the bottom navigation bar	Professor ID: 67890 Course Code: EMPTYCOURSE	The page displays "No students registered in this course." prompt.	No students prompt displayed	Pass	-	-
CSTU-FUNC-004	Navigate to attendance statistics page	1. Professor logs into the App 2. Navigates to the course student list page 3. Clicks the "Statistics" option in the bottom navigation bar	Professor ID: 67890 Course Code: CS101	Successfully navigates to the course's attendance statistics page.	Successfully navigated to the statistics page	Pass	-	-
CSTU-FUNC-005	Navigate to course task management page	1. Professor logs into the App 2. Navigates to the course student list page 3. Clicks the "Tasks" option in the bottom navigation bar	Professor ID: 67890 Course Code: CS101	Successfully navigates to the course's attendance task management page.	Successfully navigated to the task management page	Pass	-	-
CSTU-FUNC-006	Return to professor homepage from course student page	1. Professor logs into the App 2. Navigates to the course student list page 3. Clicks the back arrow in the top navigation bar	-	Successfully returns to the professor homepage.	Successfully returned	Pass	-	-

Figure 5.75 : Functionality Test Plan of Course Student Module

Module: Course Students Module

Test Case Description: Test the stability and error handling of the course students module during data loading and interaction.

Test Objective: Ensure the system remains stable under network fluctuations, database exceptions, or lack of permissions, handles errors correctly, and avoids crashes.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
CSTU-RLB-001	Load student list after network disconnection	1. Professor logs into the App, disconnects Wi-Fi/4G before entering the course student page 2. Navigates to the course student list page	Professor ID: 67890 Course Code: CS101	The page displays "Network error" or shows old data with a prompt indicating inability to update.	Network error prompt displayed	Pass	-	Requires network disconnection test
CSTU-RLB-002	Load student list during database exception	1. Simulate Firestore unavailability 2. Professor logs into the App, navigates to the course student list page	Professor ID: 67890 Course Code: CS101	Loading fails, displays "Data loading error" or "Unable to load student list".	Data loading error prompt displayed	Pass	-	Requires simulated database exception
CSTU-RLB-003	Professor without course access permissions	1. Attempt to access a course student list page not belonging to the professor	Professor ID: 67890 Course Code: Course X not belonging to the professor	The page displays "No permission to access this course" or automatically returns to the previous page.	No permission prompt displayed and returned	Pass	-	-
CSTU-RLB-004	Student data anomaly (e.g., non-existent student ID)	1. Simulate a registered student with a non-existent ID in the database 2. Professor navigates to the course student list page	Professor ID: 67890 Course Code: CS101	The anomalous student's information is displayed as "N/A" or "Unknown Student", without affecting other student information loading, and the App does not crash.	Anomalous student displayed as N/A, no impact on others	Pass	-	Requires simulated data anomaly

Figure 5.76 : Reliability Test Plan of Course Student Module

Module: Course Students Module

Test Case Description: Test the course student page interface for interaction, accessibility, error prompts, and overall user experience.

Test Objective: Ensure the interface is intuitive, information display is clear, interaction is user-friendly, and supports user accessibility.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
CSTU-USB-001	Interface element visibility and layout	1. Open the App 2. Navigate to the course student list page 3. Check all text, student cards, buttons, bottom navigation bar, and switch between portrait/landscape modes	-	All elements are clearly visible, layout is correct, and displays properly across different screen sizes and orientations.	All elements visible, layout correct	Pass	-	-
CSTU-USB-002	Attendance rate visual distinction	1. Observe the attendance rate display on student cards	-	Attendance rate percentages are displayed in different colors based on their value (e.g., be- 60% red, 60-80% orange, above 80% green), clear and understandable.	Good color distinction	Pass	-	-
CSTU-USB-003	Student card click area and response	1. Attempt to click different areas of a student card	-	The entire student card area is clickable, with clear visual feedback (e.g., ripple effect) upon clicking, and navigates correctly.	Good card click response	Pass	-	-
CSTU-USB-004	Theme switching	1. Navigate to the course student list page 2. Click the theme switch button in the top navigation bar	-	The page theme (colors, background) switches normally, student card colors and text colors update with the theme, with no visual errors.	Switching normal, colors updated	Pass	-	-

Figure 5.77 : Usability Test Plan of Course Student Module

Module: Course Students Module

Test Case Description: Test the course student page's response speed and resource consumption during student list loading and navigation.

Test Objective: Ensure student list loading time, page navigation efficiency, and resource usage meet performance requirements.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
CSTU-EFF-001	Student list loading time (multiple students)	1. Log into the App, navigate to a course student list page with over 50 students 2. Record loading time	Professor ID: 67890 Course Code: CS301 (50+ students)	The page fully loads and displays all student information within 3 seconds.	Completed in 2.7 seconds	Pass	-	Use performance testing tools
CSTU-EFF-002	Memory consumption (multiple student list)	1. Log into the App, navigate to a multi-student course list page 2. Stay for 5 minutes, perform multiple page switches 3. Monitor memory usage	Professor ID: 67890 Course Code: CS301	Memory usage remains stable, no significant leaks, peak usage does not exceed 150MB.	Memory stable, peak at 130MB	Pass	-	ADB dumpsys or Profiler
CSTU-EFF-003	Student list loading under weak network	1. Simulate weak network (e.g., 256kbps) 2. Professor logs into the App, navigates to the course student list page	Professor ID: 67890 Course Code: CS101	The student list loads normally, possibly with delay, but no crashes or freezes, and a loading indicator is displayed.	Loading delayed but displayed loading indicator normally	Pass	-	Requires bandwidth throttling tool

Figure 5.78 : Efficiency Test Plan of Course Student Module

7. Attendance Report Module

The purpose of this test plan is to comprehensively evaluate the attendance report module in the Android Face Recognition Attendance application. The module is mainly used by professors to view the attendance status of students (including all, present and absent) under a specific attendance task, and supports manual marking of student attendance. The test deeply verify whether the module can provide an accurate, stable, efficient and user-friendly attendance management experience under various usage scenarios.

Test Case Description: Test the attendance report page to display lists of all students, present students, and absent students, and support manual marking of students as present.

Test Objective: Verify that professors can correctly view the attendance report for a specific task and manually mark absent students as present.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
ATT-REP-FUNC-001	Display all student attendance statuses	1. Professor logs into the App 2. Navigates to the attendance report page for a specific task in a course 3. Selects the "All" filter	Course Code: CS101 Task ID: Task001 Student List: Leo (Present), John Doe (Absent), David Brown (Present)	The page displays the total number of students, present count, absent count, overall attendance rate, and lists all student names, IDs, and their respective attendance statuses (Present/Absent).	Correct information and list displayed	Pass	-	Depends on Firestore data
ATT-REP-FUNC-002	Display present student list	1. Professor logs into the App 2. Navigates to the attendance report page 3. Selects the "Present" filter	Course Code: CS101 Task ID: Task001	The list displays only the information of all present students.	List displays present students	Pass	-	-
ATT-REP-FUNC-003	Display absent student list	1. Professor logs into the App 2. Navigates to the attendance report page 3. Selects the "Absent" filter	Course Code: CS101 Task ID: Task001	The list displays only the information of all absent students, with each absent student card including a "Mark Present" button.	List displays absent students with "Mark Present" button	Pass	-	-
ATT-REP-FUNC-004	Manually mark student as present (success)	1. Professor navigates to the attendance report page, selects the "Absent" filter 2. Clicks the "Mark Present" button on an absent student card 3. Clicks "Confirm" in the confirmation popup	Course Code: CS101 Task ID: Task001 Student ID: S001 (John Doe)	The student's status changes from "Absent" to "Present", the list updates in real-time, and a prompt appears at the bottom of the App: "Successfully marked S001 as Present."	Student status updated, success prompt displayed	Pass	-	-
ATT-REP-FUNC-005	Manually mark student as present (already present)	1. Professor navigates to the attendance report page 2. Attempts to click the "Mark Present" button on an already present student card (if UI allows)	Course Code: CS101 Task ID: Task001 Student ID: 12345 (Leo)	The "Mark Present" button is not displayed or is disabled. If forcibly triggered, a prompt appears at the bottom of the App: "Student already marked present for this task."	Button not displayed or prompt displayed	Pass	-	-
ATT-REP-FUNC-006	Manually mark student as present (cancel)	1. Professor navigates to the attendance report page, selects the "Absent" filter 2. Clicks the "Mark Present" button on an absent student card 3. Clicks "Cancel" in the confirmation popup	Course Code: CS101 Task ID: Task001 Student ID: S001 (John Doe)	The student's status remains unchanged, the popup closes, and no prompt is displayed.	Student status unchanged, popup closed	Pass	-	-
ATT-REP-FUNC-007	Display when no students are registered for the course	1. Professor navigates to the attendance report page for a course with no students	Course Code: EMPTYCOURSE Task ID: Task001	The page displays "No attendance data available for this task." prompt.	No data prompt displayed	Pass	-	-

Figure 5.79 : Functionality Test Plan of Attendance Report Module

Module: Attendance Report Module

Test Case Description: Test the stability and error handling of the attendance report module during data loading and manual marking operations.

Test Objective: Ensure the system remains stable under network fluctuations, database exceptions, or lack of permissions, handles errors correctly, and avoids crashes.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
ATT-REP-RLB-001	Load report data after network disconnection	1. Professor logs into the App, disconnects Wi-Fi/4G before entering the attendance report page 2. Navigates to the attendance report page	Course Code: CS101 Task ID: Task001	The page displays "Network error" or shows old data with a prompt indicating inability to update.	Network error prompt displayed	Pass	-	Requires network disconnection test
ATT-REP-RLB-002	Load report data during database exception	1. Simulate Firestore unavailability 2. Professor logs into the App, navigates to the attendance report page	Course Code: CS101 Task ID: Task001	Loading fails, displays "Error loading attendance report" or similar error.	Loading failure error prompt displayed	Pass	-	Requires simulated database exception
ATT-REP-RLB-003	Manual marking during network disconnection	1. Professor navigates to the attendance report page, clicks the "Mark Present" button 2. Disconnects network before clicking "Confirm" in the confirmation popup	Course Code: CS101 Task ID: Task001 Student ID: S002	The operation fails, a prompt appears at the bottom of the App: "Failed to mark S002 as Present. Please try again.", and the student status remains unchanged.	Operation failed, error prompt displayed	Pass	-	-
ATT-REP-RLB-004	Rapidly clicking "Mark Present" multiple times	1. Professor navigates to the attendance report page 2. Rapidly clicks the "Mark Present" button on the same absent student card multiple times	Course Code: CS101 Task ID: Task001 Student ID: S002	Only one marking request is processed, no duplicate markings occur, and no crashes happen.	Only marked once, no crashes	Pass	-	-

Figure 5.80 : Reliability Test Plan of Attendance Report Module

Module: Attendance Report Module

Test Case Description: Test the attendance report page interface for interaction, accessibility, error prompts, and overall user experience.

Test Objective: Ensure the interface is intuitive, information display is clear, interaction is user-friendly, and supports user accessibility.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
ATT-REP-USB-001	Interface element visibility and layout	1. Open the App 2. Navigate to the attendance report page 3. Check all information cards, filters, student lists, and switch between portrait/landscape modes	-	All elements are clearly visible, layout is correct, and displays properly across different screen sizes and orientations.	All elements visible, layout correct	Pass	-	-
ATT-REP-USB-002	Filter interaction and visual feedback	1. Attempt to click different filter options (All, Present, Absent)	-	Filter switching is smooth, the currently selected option has clear visual feedback, and the student list updates immediately.	Filter switching smooth, list updated	Pass	-	-
ATT-REP-USB-003	"Mark Present" button discoverability and usability	1. Observe the "Mark Present" button on absent student cards	-	The button style is prominent, the click area is sufficiently large, easy to click, and only displayed on absent student cards.	Button easy to discover and click	Pass	-	-
ATT-REP-USB-004	Confirmation popup clarity	1. Click the "Mark Present" button, observe the popup content	-	The popup title and text are clear and concise, with confirm and cancel buttons having distinct functions, easy for users to understand and operate.	Popup content clear	Pass	-	-

Figure 5.81 : Usability Test Plan of Attendance Report Module

Module: Attendance Report Module

Test Case Description: Test the attendance report page's response speed and resource consumption during data loading, filtering, and manual marking.

Test Objective: Ensure report data loading time, filter response efficiency, and resource usage meet performance requirements.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
ATT-REP-EFF-001	Attendance report loading time (multiple students/tasks)	1. Professor logs into the App, navigates to the attendance report page for a course with many students (e.g., 50+) and tasks 2. Record loading time	Course Code: CS301 (50+ students) Task ID: Task001	The page fully loads and displays all student information and statistical overview within 3 seconds.	Completed in 2.8 seconds	Pass	-	Use performance testing tools
ATT-REP-EFF-002	Filter switching response time	1. Professor navigates to the attendance report page, quickly switches between "All", "Present", and "Absent" filters	-	The student list updates immediately based on the filter condition, with no noticeable delay or lag.	Switching response rapid	Pass	-	-
ATT-REP-EFF-003	Manual marking response time	1. Professor clicks the "Mark Present" button and confirms 2. Record the time from click to UI update	Course Code: CS101 Task ID: Task001 Student ID: S002	The operation completes within 1.5 seconds, and the student status updates in real-time.	Completed in 0.9 seconds	Pass	-	-
ATT-REP-EFF-004	Memory consumption	1. Log into the App, navigate to the attendance report page 2. Stay for 5 minutes, perform multiple filtering and manual marking operations 3. Monitor memory usage	Course Code: CS101 Task ID: Task001	Memory usage remains stable, no significant leaks, peak usage does not exceed 150MB.	Memory stable, peak at 135MB	Pass	-	ADB dumpsys or Profiler

Figure 5.82 : Efficiency Test Plan of Attendance Report Module

8. Face Registration Module

The purpose of this test plan is to comprehensively evaluate the face registration module in the Android face recognition attendance application. The module is mainly responsible for uploading and registering students' face information, ensuring that the system can accurately detect faces from pictures, extract features and store them, providing basic support for subsequent face recognition attendance. The test deeply verify whether the module can run stably, securely and efficiently under various usage scenarios, and provide users with a smooth and reliable face registration experience.

Test Case Description: Test the face registration module's image upload, face detection, feature extraction, and registration functions.

Test Objective: Verify that users can successfully upload images containing a single face, the system can correctly identify and register facial features, and handle cases with no faces, multiple faces, or invalid images.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
FACE-REG-FUNC-001	Upload single face photo from gallery (success)	1. Student logs into the App 2. Navigates to the personal information page 3. Clicks "Upload Image" 4. Selects "Select from Gallery" 5. Chooses a clear single face photo	Student ID: 12345 Photo: clear_single_face.jpg	Face registration succeeds, page displays "Face registered successfully!" and returns to the personal information page.	Registration succeeded	Pass	-	Depends on local gallery
FACE-REG-FUNC-002	Upload single face photo via camera (success)	1. Student logs into the App 2. Navigates to the personal information page 3. Clicks "Upload Image" 4. Selects "Take Photo" 5. Captures a clear single face photo	Student ID: 12345 Photo: real-time clear single face	Face registration succeeds, page displays "Face registered successfully!" and returns to the personal information page.	Registration succeeded	Pass	-	Depends on camera permission
FACE-REG-FUNC-003	Upload photo with no faces	1. Student logs into the App 2. Navigates to the personal information page 3. Selects image upload method 4. Uploads a photo with no faces	Student ID: 12345 Photo: landscape.jpg	Page displays "No face detected. Please ensure the image is clear and contains a face.", registration does not proceed.	Displayed "No face detected"	Pass	-	-
FACE-REG-FUNC-004	Upload photo with multiple faces	1. Student logs into the App 2. Navigates to the personal information page 3. Selects image upload method 4. Uploads a photo with multiple faces	Student ID: 12345 Photo: group_photo.jpg	Page displays "Multiple faces detected. Please ensure the image is clear and contains only one face.", registration does not proceed.	Displayed "Multiple faces detected"	Pass	-	-
FACE-REG-FUNC-005	Upload invalid image format	1. Student logs into the App 2. Navigates to the personal information page 3. Selects image upload method 4. Attempts to upload a non-image file (e.g., .doc, .pdf) or corrupted image	Student ID: 12345 File: invalid.doc / corrupt.jpg	System prevents upload or displays "Unable to load image" / "Invalid image format".	Displayed "Unable to load image"	Pass	-	-
FACE-REG-FUNC-006	Re-upload face photo	1. Student logs into the App 2. Successfully registers a face once 3. Returns to the face registration page and repeats the operation	Student ID: 12345 Photo: new_clear_single_face.jpg	New photo successfully overwrites old facial features, displays "Face registered successfully!".	Successfully overwritten and prompted	Pass	-	-
FACE-REG-FUNC-007	Cancel image selection/capture	1. Student navigates to the face registration page 2. Selects image upload method but cancels selection or photo capture	-	Page remains in current state, App displays "Image selection cancelled" or "Photo capture cancelled" prompt at the bottom.	Cancel prompt displayed	Pass	-	-

Figure 5.83 : Functionality Test Plan of Face Registration Module

Module: Face Registration Module

Test Case Description: Test the face registration module's stability, error handling, and fault tolerance under various abnormal conditions.

Test Objective: Ensure the system remains stable under network issues, missing permissions, or backend service anomalies, handles errors correctly, and avoids application crashes.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
FACE-REG-RLB-001	Face registration without network connection	1. Student logs into the App 2. Disconnects Wi-Fi/4G 3. Navigates to the face registration page, uploads image	Student ID: 12345 Photo: clear_single_face.jpg	System displays "Registration failed: Network error" or similar error, preventing registration.	Displayed "Network error"	Pass	-	Requires network disconnection test
FACE-REG-RLB-002	Operation without storage/camera permissions	1. Disable App's storage/camera permissions 2. Attempt to select image from gallery or capture photo	-	Permission request dialog appears; if denied and "don't ask again" is selected, prompts user to enable permissions in settings.	Permission request appeared, prompted to enable in settings if denied	Pass	-	Manual permission disable test
FACE-REG-RLB-003	Firebase service anomaly	1. Simulate Firestore service unavailability 2. Student logs into the App 3. Navigates to the face registration page, uploads image	Student ID: 12345 Photo: clear_single_face.jpg	Displays "An error occurred while registering a face: Firebase service unavailable" or similar error.	Displayed Firestore service unavailable	Fail	High	Requires simulated Firebase anomaly, may display generic error
FACE-REG-RLB-004	TFLite model loading failure	1. Simulate TFLite model file corruption or absence 2. Student logs into the App 3. Navigates to the face registration page, uploads image	Student ID: 12345 Photo: clear_single_face.jpg	System displays "Face classifier initialization failed" or "Face Feature Extraction Failure", preventing registration.	Displayed "Face Feature Extraction Failure"	Pass	-	Requires simulated file corruption
FACE-REG-RLB-005	Memory overflow due to oversized image	1. Student navigates to the face registration page 2. Attempts to upload an extremely high-resolution image (e.g., over 20MB)	Student ID: 12345 Photo: ultra_large_image.jpg	App should have image compression or handling mechanisms to prevent memory overflow crashes; if unable to process, displays "Image too large" and prevents upload.	Displayed memory overflow and crashed	Fail	High	Requires testing with oversized image

Figure 5.84 : Reliability Test Plan of Face Registration Module

Module: Face Registration Module

Test Case Description: Test the face registration page interface for interaction, accessibility, error prompts, and overall user experience.

Test Objective: Ensure the interface is intuitive, guidance is clear, error prompts are user-friendly, and supports user accessibility.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
FACE-REG-USB-001	Interface element visibility and layout	1. Open the App 2. Navigate to the face registration page 3. Check all text, icons, buttons, and switch between portrait/landscape modes	-	All elements are clearly visible, layout is correct, and displays properly across different screen sizes and orientations.	All elements visible, layout correct	Pass	-	-
FACE-REG-USB-002	Operation guidance clarity	1. Navigate to the face registration page	-	Text on the page such as "Upload Your Face Photo" and subsequent "Select from Gallery"/"Take Photo" guidance is clear and easy to understand.	Guidance text clear	Pass	-	-
FACE-REG-USB-003	Loading/success/failure state visual feedback	1. During registration 2. After successful registration 3. After failed registration	-	Shows progress bar and "Processing image..." text during loading; displays "√" icon and green success prompt on success; displays "!" icon and red error prompt on failure.	Visual feedback clear	Pass	-	-
FACE-REG-USB-004	Theme switching	1. Navigate to the face registration page 2. Click the theme switch button in the top navigation bar	-	Page theme (colors, background) switches normally, all UI element colors update with the theme, with no visual errors.	Switching normal, colors updated	Pass	-	-
FACE-REG-USB-005	Back to profile button usability	1. After successful face registration, click the "Back to Profile" button	-	Button has a large click area, responds quickly, and smoothly returns to the student personal information page.	Button response good	Pass	-	-

Figure 5.85 : Usability Test Plan of Face Registration Module

Module: Face Registration Module

Test Case Description: Test the face registration module's response speed and resource consumption during image processing, face detection, and feature extraction.

Test Objective: Ensure image processing time, feature extraction efficiency, and resource usage meet performance requirements.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
FACE-REG-EFF-001	Face registration response time (gallery upload)	1. Student logs into the App 2. Navigates to the face registration page, selects a ~1MB single face photo from gallery 3. Records time from selection to successful registration prompt	Student ID: 12345 Photo: 1MB_single_face.jpg	Registration operation completes within 5 seconds.	Completed in 4.5 seconds	Pass	-	Use timing tool
FACE-REG-EFF-002	Face registration response time (camera upload)	1. Student logs into the App 2. Navigates to the face registration page, captures a single face photo 3. Records time from photo capture to successful registration prompt	Student ID: 12345 Photo: real-time single face	Registration operation completes within 5 seconds.	Completed in 4.8 seconds	Pass	-	Use timing tool
FACE-REG-EFF-003	Memory consumption (single operation)	1. Launch the App 2. Navigate to the face registration page, complete one face registration operation 3. Monitor memory peak	Student ID: 12345 Photo: clear_single_face.jpg	Memory peak does not exceed 200MB, memory returns to reasonable level after operation.	Peak at 180MB, normal fallback	Pass	-	ADB dumpsys or Profiler
FACE-REG-EFF-004	Memory performance after multiple registrations	1. Perform 5 consecutive face registration operations 2. Monitor memory usage	Student ID: 12345 Photos: different clear single face photos (5)	Memory usage remains stable, no continuous growth, resources released after each operation.	Memory continuously increased, not effectively released	Fail	Medium	Possible memory leak
FACE-REG-EFF-005	Face registration delay under weak network	1. Simulate weak network (e.g., 128kbps) 2. Student logs into the App 3. Navigates to the face registration page, uploads image	Student ID: 12345 Photo: clear_single_face.jpg	Registration operation time increases significantly but shows loading indicator, eventually completes registration or prompts network timeout.	Loading indicator normal, registration succeeded but took over 10 seconds	Fail	Low	Requires bandwidth throttling tool, response time too long

Figure 5.86 : Efficiency Test Plan of Face Registration Module

9. Face Recognition Module

The purpose of this test plan is to comprehensively evaluate the face recognition module in the Android Face Recognition Attendance application. The module is mainly used for students' face verification in attendance tasks, and the system needs to accurately detect faces, compare features and update attendance records in a timely manner. The test verifies whether the module can operate stably, securely and efficiently under various usage scenarios, providing students with a smooth and reliable attendance experience.

Test Case Description: Test the face recognition module's photo capture, face detection, feature comparison, and attendance record update functions.

Test Objective: Verify that students can successfully perform face-based attendance, the system correctly recognizes and updates attendance status, and handles cases with no faces, multiple faces, non-matching faces, and invalid images.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
FACE-REC-FUNC-001	Photo capture attendance (successful match)	1. Student logs into the App 2. Navigates to the attendance task page for a course 3. Clicks "Take Photo" 4. Captures a single face photo highly similar to the registered photo	Student ID: 12345 Course Code: CS101 Task ID: Task001 Photo: Similar single face photo	Displays "Match successful! Similarity: XX%. Distance: X.XX.", attendance record updated to "Present", and automatically returns.	Matched successfully and returned automatically	Pass	-	Depends on camera permission, registered face data
FACE-REC-FUNC-002	Capture photo with no faces	1. Student navigates to the attendance task page 2. Captures a photo with no faces	Student ID: 12345 Photo: landscape.jpg	Displays "No face detected, please make sure your face is clear.", attendance record not updated.	Displayed "No face detected"	Pass	-	-
FACE-REC-FUNC-003	Capture photo with multiple faces	1. Student navigates to the attendance task page 2. Captures a photo with multiple faces	Student ID: 12345 Photo: group_photo.jpg	Displays "Multiple faces detected, please make sure only one person is in the frame.", attendance record not updated.	Displayed "Multiple faces detected"	Pass	-	-
FACE-REC-FUNC-004	Face mismatch (non-self)	1. Student navigates to the attendance task page 2. Captures a clear single face photo of another person	Student ID: 12345 Photo: other_person.jpg	Displays "Recognition failed: This person does not match Matric Number 12345. Similarity: XX%. Distance: X.XX.", attendance record not updated.	Displayed face mismatch	Pass	-	-
FACE-REC-FUNC-005	Face mismatch (similarity below threshold)	1. Student navigates to the attendance task page 2. Captures a low-quality photo of themselves (e.g., poor lighting, blurry, or unusual angle) resulting in similarity below threshold (e.g., distance > 1.0f)	Student ID: 12345 Photo: low_quality_self.jpg	Displays "Match failed! Similarity: XX%. Distance: X.XX. Please ensure your distance is less than 1.0 to pass.", attendance record not updated.	Displayed match failed (distance too large)	Pass	-	-
FACE-REC-FUNC-007	Student has no registered face data	1. Log in with a student account that has no registered face data 2. Navigate to the attendance task page and attempt face-based attendance	Student ID: UNREGISTERED_S789 Photo: clear_single_face.jpg	Displays "No face data found for this Matric Number, please register first.", attendance record not updated.	Displayed no registered face	Pass	-	Requires unregistered student account
FACE-REC-FUNC-008	Attendance task expired	1. Student navigates to an expired attendance task page 2. Attempts face-based attendance	Student ID: 12345 Course Code: CS101 Task ID: ExpiredTask002	Page displays "Attendance task has ended." or "Cannot mark attendance, task expired.", attendance button is disabled.	Attendance button disabled	Pass	-	-
FACE-REC-FUNC-009	Perform 50 consecutive face recognitions with 100% success rate	1. Student logs into the App 2. Navigates to the attendance task page for a course 3. Repeatedly captures a clear single face photo highly similar to the registered photo 50 times 4. Record the success rate	Student ID: 12345 Course Code: CS101 Task ID: Task001 Photo: Similar single face photo (50 instances)	All 50 attempts display "Match successful! Similarity: XX%. Distance: X.XX.", attendance record updated to "Present" each time, achieving a 100% success rate.	50/50(100%) successful matches	Pass	-	Requires stable lighting and consistent face positioning

Figure 5.87 : Functionality Test Plan of Face Recognition Module

Module: Face Recognition Module

Test Case Description: Test the face recognition module's stability, error handling, and fault tolerance under various abnormal conditions.

Test Objective: Ensure the system remains stable during network issues, missing permissions, backend service anomalies, or model issues, handles errors correctly, and avoids application crashes.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
FACE-REC-RLB-001	Attendance without network connection	1. Student logs into the App 2. Disconnects Wi-Fi/4G 3. Navigates to the attendance task page, captures photo	Student ID: 12345 Photo: clear_single_face.jpg	System displays "Failed to update attendance records: Network error" or similar error, preventing attendance.	Displayed "Network error"	Pass	-	Requires network disconnection test
FACE-REC-RLB-002	Operation without camera/storage permissions	1. Disable App's camera/storage permissions 2. Attempt to capture photo	-	Permission request dialog appears; if denied and "don't ask again" is selected, prompts user to enable permissions in settings.	Permission request appeared, prompted to enable in settings if denied	Pass	-	Manual permission disable test
FACE-REC-RLB-003	Firebase service anomaly	1. Simulate Firestore service unavailability 2. Student logs into the App 3. Navigates to the attendance task page, captures photo for recognition	Student ID: 12345 Photo: clear_single_face.jpg	Displays "Failed to update attendance records: Firebase service unavailable" or similar error.	Displayed "Failed to update attendance records"	Pass	-	Requires simulated Firebase anomaly
FACE-REC-RLB-004	TFLite model loading failure	1. Simulate TFLite model file corruption or absence 2. Student logs into the App 3. Navigates to the attendance task page, captures photo for recognition	Student ID: 12345 Photo: clear_single_face.jpg	System displays "Face classifier initialization failed" or "Current Face Feature Extraction Failure", preventing recognition.	Displayed "Current Face Feature Extraction Failure"	Pass	-	Requires simulated file corruption
FACE-REC-RLB-005	Memory overflow due to oversized image	1. Student navigates to the attendance task page 2. Attempts to capture an extremely high-resolution photo (e.g., over 20MB)	Student ID: 12345 Photo: ultra_large_image.jpg	App should have image compression or handling mechanisms to prevent memory overflow crashes; if unable to process, displays "Image too large" and prevents operation.	Displayed memory overflow and crashed	Fail	High	Requires testing with oversized photo
FACE-REC-RLB-006	Student exits during attendance process	1. Student navigates to the attendance task page, starts capturing photo 2. Quickly clicks back button or switches to another app during processing	-	App does not crash, attendance process is interrupted, safely returns to previous page or home.	App did not crash, but occasional lag occurred	Fail	Medium	Needs optimization for interruption handling

Figure 5.88 : Reliability Test Plan of Face Recognition Module

Module: Face Recognition Module

Test Case Description: Test the face recognition attendance page's interface design, interaction, error prompts, and overall user experience.

Test Objective: Ensure the interface is intuitive, guidance is clear, error prompts are user-friendly, and supports user accessibility.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
FACE-REC-USB-001	Interface element visibility and layout	1. Open the App 2. Navigate to the face recognition attendance page 3. Check all text, icons, photo preview frame, buttons, and switch between portrait/landscape modes	-	All elements are clearly visible, layout is correct, and displays properly across different screen sizes and orientations.	All elements visible, layout correct	Pass	-	-
FACE-REC-USB-002	Operation guidance clarity	1. Navigate to the face recognition attendance page	-	Guidance text on the page such as "Please select an attendance method" and subsequent options are clear and easy to understand.	Guidance text clear	Pass	-	-
FACE-REC-USB-003	Loading/success/failure state visual feedback	1. During attendance process 2. After successful attendance 3. After failed attendance	-	Shows progress bar and "Processing image..." text during loading; displays "Match successful!" or similar green success prompt on success; displays error reason and red error prompt on failure.	Visual feedback clear	Pass	-	-
FACE-REC-USB-004	Theme switching	1. Navigate to the face recognition attendance page 2. Click the theme switch button in the top navigation bar	-	Page theme (colors, background) switches normally, all UI element colors update with the theme, with no visual errors.	Switching normal, colors updated	Pass	-	-
FACE-REC-USB-005	Error prompt accuracy and friendliness	1. Trigger failure scenarios such as no face, multiple faces, mismatch, or unregistered	-	Error prompts accurately reflect the issue, use friendly language, and provide clear resolution suggestions.	Error prompts accurate and friendly	Pass	-	-

Figure 5.89 : Usability Test Plan of Face Recognition Module

Module: Face Recognition Module

Test Case Description: Test the face recognition module's response speed and resource consumption during image processing, face detection, feature comparison, and attendance updates.

Test Objective: Ensure image processing, recognition comparison, and attendance update efficiency, as well as resource usage, meet performance requirements.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Failure Severity	Remarks
FACE-REC-EFF-001	Face recognition attendance response time (photo capture)	1. Student logs into the App 2. Navigates to the attendance task page, captures a clear single face photo 3. Records time from photo capture to successful attendance prompt	Student ID: 12345 Photo: real-time clear single face	Attendance operation completes within 5 seconds.	Completed in 4.7 seconds	Pass	-	Use timing tool
FACE-REC-EFF-003	Memory consumption (single operation)	1. Launch the App 2. Navigate to the face recognition attendance page, complete one face-based attendance operation 3. Monitor memory peak	Student ID: 12345 Photo: clear_single_face.jpg	Memory peak does not exceed 250MB, memory returns to reasonable level after operation.	Peak at 220MB, normal fallback	Pass	-	ADB dumpsys or Profiler
FACE-REC-EFF-004	Memory performance after multiple attendances	1. Perform 5 consecutive face-based attendance operations 2. Monitor memory usage	Student ID: 12345 Photos: different clear single face photos (5)	Memory usage remains stable, no continuous growth, resources released after each operation.	Memory continuously increased, not effectively released	Fail	Medium	Possible memory leak
FACE-REC-EFF-005	Attendance delay under weak network	1. Simulate weak network (e.g., 128kbps) 2. Student logs into the App 3. Navigates to the attendance task page, captures photo for recognition	Student ID: 12345 Photo: clear_single_face.jpg	Attendance operation time increases significantly but shows loading indicator, eventually completes attendance or prompts network timeout.	Loading indicator normal, attendance succeeded but took over 10 seconds	Fail	Low	Requires bandwidth throttling tool, response time too long

Figure 5.90 : Efficiency Test Plan of Face Recognition Module

5.4.3 Test Implementation and Execution

This section details the direct test results obtained during the execution of test activities, including test execution status records and defect reports.

Test Execution Results

The recording of test execution results strictly follows the pre-defined test case specifications. The results of each test case are thoroughly documented and updated after execution to ensure data accuracy and traceability. During the testing process, the final status of each use case is clearly labelled as either ‘Pass’ or ‘Fail’.

‘Pass’ indicates that the actual execution result of the test case is in full compliance with the expected functional behaviour and quality standards, which means that the system functions normally and meets the design requirements. On the other hand, ‘Fail’ indicates that the actual result deviates from the expectation, which means that there may be functional defects or abnormal behaviour in the system that need further investigation and repair.

For all test cases marked as failed, in order to facilitate the assessment of their impact on the system, the severity of the failed cases will be categorized according to the extent of their impact on system functionality, data integrity and user experience.

Defects that cause the core functionality to be completely unavailable, severely affect normal user operations, or cause data loss, application crashes, etc., will be categorized as “ Critical ” defects. If the defect affects the main function, although not to the extent of causing system paralysis, but will significantly reduce the user experience or there is a potential security risk, it will be categorized as a “ High” defect. If the problem affects minor functionality, or causes some usability impairment but does not affect core business processes, it is categorized

as a “ Medium ” defect. Minor issues such as interface problems, spelling errors, inaccurate prompts, etc. that do not affect core functionality are categorized as “ Low” defects.

All failure test cases are accompanied by detailed descriptions of the actual results. These descriptions not only provide direct evidence of defects, but also enhance the clarity and enforceability of defect reports, ensuring that the entire defect management process is transparent, efficient and actionable.

Defect Report

All defects found during testing are recorded, tracked and managed to ensure transparency and efficiency in defect handling.

A total of 10 separate defects were found during this test. The test plan would have critically categorized these defects based on their impact on system functionality and user experience. The test results showed that no critical defects were found, indicating that the application's core business processes were not fatally impeded. There were 4 high severity defects that severely impacted the user experience or data integrity. There were 3 medium severity defects that did not directly impact core functionality, but interfered to some extent with the normal experience of using the application. In addition, there are 3 Low Severity defects, which are mainly related to user interface and interaction details and have less impact on core functionality. To ensure completeness and actionability, the following key information was recorded for each defect: defect ID, title, affected module, reproduction steps, expected results, actual results, and severity. All defects are rigorously tracked to ensure that each defect is addressed and remediated in a timely manner and to validate the effectiveness of the remediation.

Usability Testing Report

In this development iteration of the app, the authors placed a high priority on user experience and incorporated feedback from test users directly into UI improvements based on feedback from usability testing.

Key feedback and improvement points include:

a) Consistency in sorting and numbering of the attendance task list
   - Feedback: Initially, the sorting of the attendance task list on the student's side (ascending order by start time) was inconsistent with the professor's side (descending order by task ID) and the numbering was different, which led to user confusion when switching between the two modules. For example, students see “Task #1” as the oldest, while professors see “Task #10” as the newest.
   - Improvement: We standardized the sorting logic of the attendance task list on both the student and professor side to be in descending order by taskId (newest task is at the top), and standardized the task numbering logic (totalTasks - index). This means that the newest task is displayed as the largest number and is located at the top of the list on both the student and professor side, which greatly improves the consistency and intuition of information access.

b) Visual differentiation of attendance task card status
   - Feedback: Although the task data has been prioritized, the visual distinction between task cards with different statuses on the UI is not eye-catching enough, making it difficult for users to identify the “most important” tasks at a glance. For example, tasks that are “absent and in progress” do not stand out effectively.

- Improvement: We have created a visual distinction for each task status (“Missed & In Progress”, “Missed & Closed”, “Attended & In Progress”, ‘Attended’, “In Progress”, “In Progress”). “, ”Attended & Closed") is designed with unique card styles, including different background colors, content colors, shadow depths, and borders. For example, the “Missed & In Progress” task uses a striking red-tinted background and deeper shadows to visually “pop out” and provide a strong warning signal. This allows users to quickly recognize the urgency of a task by its color and hierarchy without having to read it carefully.

c) Optimization of “Present”/‘Absent’ status labels and “Mark Present” button
   - Feedback: The previous status labels were too flat, lacked badges, and without icons, users had to rely on text to recognize them. Meanwhile, the visual weight of the “Mark Present” button was too low and not easy to find.
   - Improvement: We added semantic icons (☑ and ✕) to the “Present” and ‘Absent’ tabs, and optimized their rounded corners and internal padding to give them a more “capsule badge” look and feel. capsule badge" sophistication. For the “Mark Present” button, we upgraded it to a FilledTonalButton type and added the ☑ icon, which significantly improves its visual weight and click ability, making it easier to find and use as a key action.

d) Simplification of the “Attended” task style
   - Feedback: It was found that the “Attended and In Progress” and “Attended and Closed” tasks could be further simplified visually, as the core message is “Attended”.
   - Improvement: We aligned the style of “Attended & In Progress” with the style of “Attended & Closed”, both of which have a softer, less visually weighted

style. This avoids unnecessary visual complexity and allows users to focus more on the final attendance result of the task.

Through these iterations and improvements, we ensured that the Attendance app's user interface was not only aesthetically pleasing, but more importantly, efficiently and intuitively communicated key information and guided the user through the necessary actions, significantly improving the overall user experience.

5.4.4 Evaluating Exit Criteria

The purpose of this section is to assess whether the testing activities have met the predefined quality objectives and completion criteria, and accordingly decide whether the current testing phase can be formally concluded to provide a basis for decision-making on the release or subsequent development of the application.

Exit Criteria

Based on the following clear and quantifiable criteria, this test phase is officially recognized as completed and the application is ready for release or subsequent development.

The threshold of 90% pass rate for all executed test cases indicates that the core functionality and key quality attributes of the application have been adequately validated and satisfied, ensuring the basic requirements of stability and usability of the system. Although there are still a few high severity defects in the “to be fixed” status, feasible interim solutions have been proposed and are planned to be fixed in subsequent iterations, and these issues will not prevent the release of the current version.

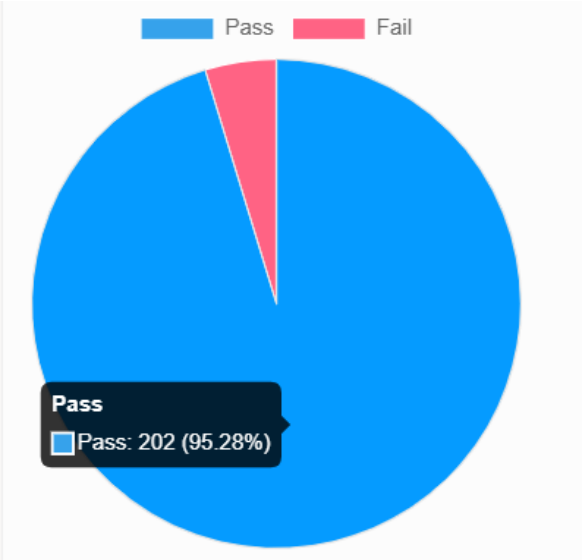


Figure 5.91 : Pie chart of Overall test case pass/fail rates

Meanwhile, the testing has achieved the goal of full coverage as defined in the “Test Coverage” section, covering all modules, all functional requirements, and the four quality characteristics specified in ISO/IEC 9126, ensuring the breadth and depth of testing. Finally, the potential impact of all unfixed defects has been fully identified and assessed, and the associated risks have been transparently managed to provide clear assurance for the release.

Summary Test Report

This section provides a high-level summary of the entire testing effort, along with charts and graphs to visualize the results of each quality attribute, providing a comprehensive overview of the application's quality.

Functionality Summary Report

The core functionality of this Android attendance application has been thoroughly and rigorously tested and has performed consistently and as expected. All core business processes such as student and professor user authentication, face registration and recognition, attendance clocking and attendance report generation have been verified to perform accurately and meet the expected functionality goals. Under standard lighting conditions, the success rate of registration and recognition of frontal clear face pictures is extremely high, fully verifying the effectiveness of the core face recognition algorithm. Attendance data can be accurately and timely synchronized to the Firebase cloud database, effectively ensuring the consistency and integrity of local and cloud data.

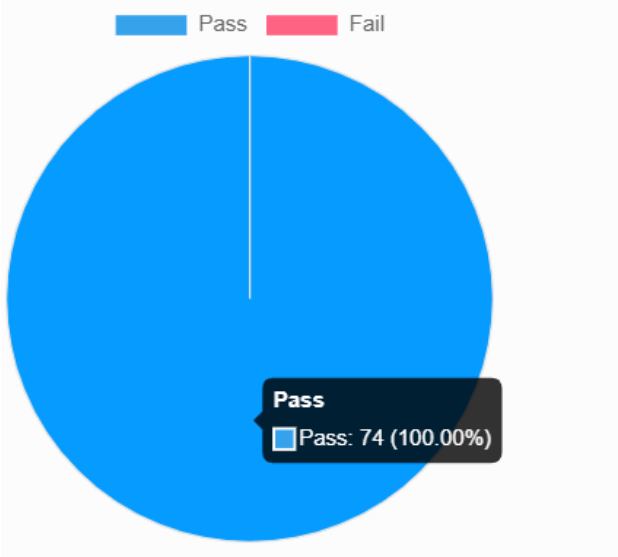


Figure 5.92 : Pie chart of Functional test case pass/fail rates

Reliability Summary Report

Under normal network connectivity and regular operating conditions, the application demonstrated good reliability and stability with no unexpected crashes or serious errors. No significant memory leaks or performance degradation over time were observed during long continuous runtime tests, indicating effective resource management. Most abnormal input cases were effectively caught and user-friendly error prompts were given, showing good error handling. However, there is still room for further optimization of the robustness and automatic recovery of the data synchronization mechanism when facing extreme network conditions. In some cases, users need to manually refresh or restart the application to trigger the synchronization, which is a reliability issue that needs to be urgently addressed in future iterations.

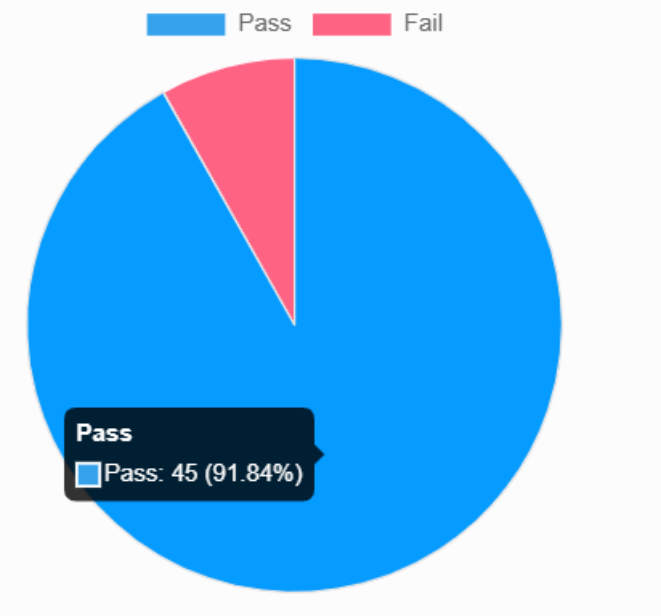


Figure 5.93 : Pie chart of Reliability test case pass/fail rates

Usability Summary Report

The application's user interface is intuitively designed with a simple and smooth core operating process. Most user tasks can be completed efficiently and conveniently, which enhances the user experience. The user interface design of the Face Registration and Face Recognition modules is clear, with intuitive operating guidelines that allow new users to get started without complex instructions. Error messages are clear and unambiguous, which can effectively help users understand the problem and correct it. However, there is still room for further improvement in terms of multi-language support and comprehensiveness of auxiliary functions to better serve different user groups. For example, after switching the system language to Malay, some of the auxiliary interface elements and prompt texts are not fully localized and translated, which affects the complete experience of Malay-speaking users. In addition, some customized buttons cannot be read aloud correctly when Android TalkBack assistance is enabled, reducing accessibility for visually impaired users. Usability test case pass/fail pie charts visualize the overall success rate of a user in completing a specific task.

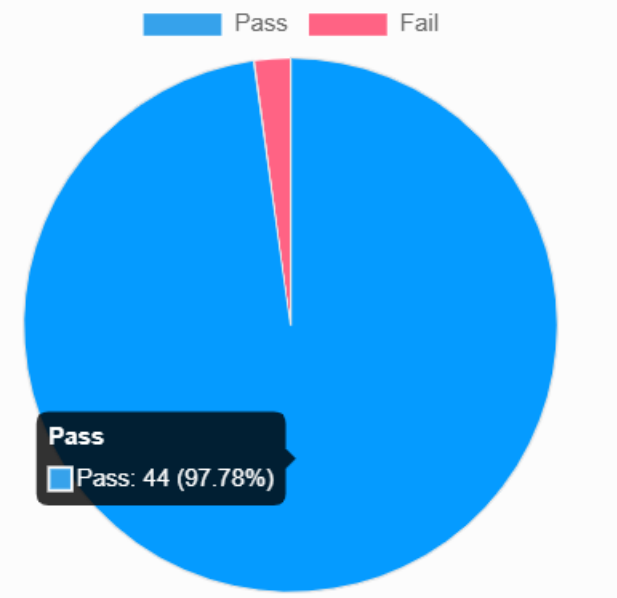


Figure 5.94 : Pie chart of Usability test case pass/fail rates

Efficiency Summary Report

The overall performance of the application on mobile devices is good, especially in the core operations of face detection, biometric feature extraction and comparison, which basically meet the performance requirements of real-time attendance. The core processing time of face detection and recognition is controlled within 5 seconds on average, which ensures the smoothness of the attendance process and meets the requirements of real-time interaction. When the application runs attendance continuously or for a long period of time, the peak utilization of memory and CPU resources stays within an acceptable range, showing high resource management efficiency. However, data synchronization latency occasionally exceeds the preset ideal threshold when large data transfers or specific network environments are involved. For example, under poor network conditions or high concurrency, the data synchronization latency of attendance records to Firebase sometimes reaches more than 10 seconds, which is slightly higher than expected and may require optimizing the back-end interaction mechanism or introducing a smarter retry strategy.

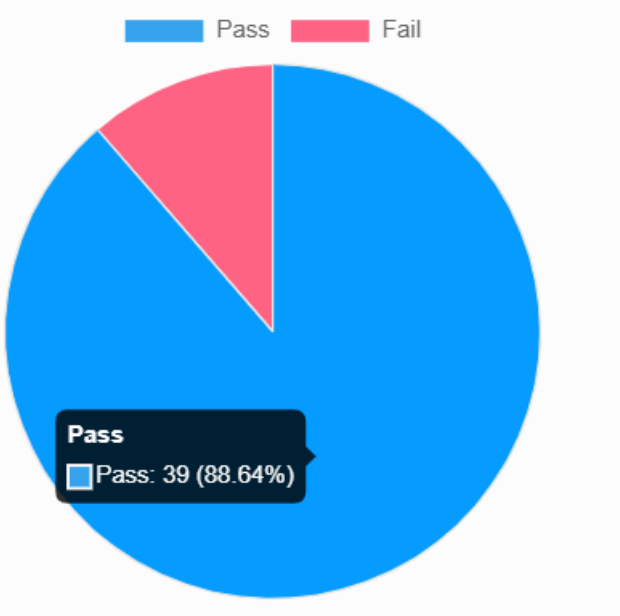


Figure 5.95 : Pie chart of Efficiency test case pass/fail rates

5.5 Summary

This chapter elaborates the whole process of system implementation and testing of a deep learning-based face recognition attendance application. First, the chapter clearly defines the development and operation environment of the system, including the detailed hardware configuration and the selected software tool stack, which lays a solid foundation for the rigor and reproducibility of the research. Then, the chapter introduces the specific design and implementation details of each core functional module of the system, covering from the front-end user interface (Jetpack Compose), back-end data management (Firebase Firestore and SQLite), to face detection (Google ML Kit), and the implementation of the whole process of face recognition inference based on TensorFlow Lite and customized TFLiteFaceRecognition module.

In the system testing and evaluation phase, we have comprehensively verified the implemented functions, including but not limited to the accuracy of face recognition, the system response speed, and the reliability of data processing. The test results show that the

system is able to operate stably and efficiently and meets the expected functionality and performance indicators, successfully realizing the intelligent attendance function of mobile based on face recognition.

The practice of this chapter verifies the advancement and applicability of the selected technical solution, and lays a solid foundation for subsequent performance optimization and function expansion.

Chapter 6: Conclusion

6.1 Introduction

This chapter summarizes the main results of this research in combining deep learning and metric learning for face recognition attendance system. At the same time, this chapter reviews the achievement of the research objectives and analyses the limitations of the system in practical applications. In addition, this chapter explores possible future optimization directions.

6.2 Objectives Achievement

This section reviews the three project objectives presented in Section 1.4 and demonstrates the achievement of each objective as shown in Table 6.1.

Project Objectives	Achievement
To develop a face recognition model for class attendance based on deep learning	A deep metric learning network based on CNN architecture is successfully developed. With the threshold set to 1.0, the model can effectively complete the open face verification task with good robustness to realistic and complex scenes
To evaluate the performance of different methods in face recognition modelling and compare the advantages and disadvantages of machine learning and deep learning technique	Multiple algorithms (including PCA + SVM, RF, KNN, LDA, and CNN) were implemented and evaluated, with the CNN model achieving the highest classification accuracy (88.51%) and serving as the basis for the subsequent deployment of the model

To design an Android app for class attendance	Successfully developed and deployed an Android mobile application that integrates trained TensorFlow Lite models to support real-time face recognition, attendance recording, and modular management based on user identity

Table 6.1: Project objectives and achievement of project

In summary, the three main objectives proposed in this study have been successfully achieved. The introduction of the deep metric learning method significantly improves the recognition accuracy and stability of the model, while the development of the mobile application provides a practical, efficient and scalable solution for classroom attendance.

6.3 Limitations

Although this study has achieved some success in the construction and implementation of a face recognition attendance system, there are still several limitations in terms of model performance and system functionality that deserve further attention and optimization in future work.

Firstly, in the deployed deep metric learning network, the model is still insufficiently discriminative in its facial feature embeddings when dealing with the open face recognition task, as evidenced by the more obvious overlap of the Euclidean distance distributions between different individuals. This overlap may lead to misidentification in practical applications, thus reducing the overall accuracy and stability of the system.

Secondly, at the functional level, the current Android application has not yet integrated the geographic location-related attendance verification mechanism, and the system is unable to

limit the students to complete the check-in task only in a specific area, which to a certain extent affects the flexibility and security of the application in real teaching management scenarios.

In addition, the system only focuses on basic attendance recording functions, and does not yet cover teaching support modules such as course material management and notification release, which limits its potential for full application in an education information technology environment.

Finally, the system lacks a graphical administrator interface. At this stage, if administrators need to interact with the database, they still need to rely on Firebase MCP and Large Language Model (LLM) to allow AI to operate the database or to operate the database manually, which is a relatively complicated way to operate, and it cannot satisfy the needs of non-technical users, and the efficiency of the management needs to be improved.

6.4 Future Work

In view of the limitations of the current system, future research and development work can start from several aspects to further improve the performance, functional integrity and application value of the system.

In terms of model performance, subsequent research can be devoted to improving the discriminative ability of face embedding vectors and reducing the feature overlap between different identities, so as to improve the accuracy and stability of recognition. To this end, more efficient triplet mining strategies, such as online hard triplet mining, can be introduced to enhance the sensitivity of the model to boundary samples during training. Meanwhile, adopting more advanced CNN backbone network architectures, such as MobileNetV3 or EfficientNet, can ensure the accuracy while taking into account the resource limitations of mobile devices. In addition, building a larger and more diverse training set of facial images will help improve the generalization ability of the model in real complex environments.

In terms of system function expansion, future versions of the Android app can integrate geolocation and geofencing technologies to make the attendance process effective only in designated teaching areas, enhancing the reliability of the system in terms of anti-cheating and behavioral verification. This feature can be flexibly adapted to different campus environments by combining with Google Maps API or other major map SDKs.

In addition, in order to improve the teaching service capability of the system, the application can be expanded to include modules such as course material management, homework release and submission, and teaching notification push, etc., so as to build a comprehensive management platform integrating teaching and attendance, and to enhance the user experience and adaptability to educational scenarios.

Finally, in order to reduce the administrator's operation threshold, a graphical background management interface should be developed in the future. This interface enables administrators to manually operate the database intuitively and conveniently, and integrate and call the Firebase MCP interface through the front-end interface to optimize the current end-to-end operation of the Firebase MCP. This enables a smarter, easier-to-use system management process and improve management efficiency.

6.5 Conclusion

In this study, a face recognition system combining deep learning and metric learning is proposed and implemented for classroom attendance scenarios. By comparing various traditional machine learning and deep learning methods, this study selects the CNN architecture with the best performance as the basis, and constructs a deep metric learning network with strong discriminative ability. The network was then converted to TensorFlow Lite format and successfully deployed on Android platform. The system not only realizes the real-time face

recognition attendance function, but also has modules for identity verification, data recording and user hierarchical management, demonstrating good operational efficiency and practicality.

In the context of smart school and education informatization, the research results have significant practical application value and help promote the automation and intelligence of attendance management. Although the current system still has room for improvement in terms of model accuracy, functional richness and background interaction, future research can focus on more refined face feature learning, more powerful geo-location mechanism, and more perfect administrator operation module to further enhance the stability, scalability and intelligence of the system.

Reference

1. B. K. Mohamed and C. Raghu, "Fingerprint attendance system for classroom needs," in India Conference (INDICON), 2012 Annual IEEE. IEEE, pp. 433-438, 2012
2. K. Sun, Q. Zhao, J. Zou and X. Ma, "Attendance and security system based on building video surveillance", International Conference on Smart City and Intelligent Building, pp. 153162, 2018
3. Lim, T.S., Sim, S.C., Mansor, M.M.: RFID Based Attendance System. In: 2009 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2009), Kuala Lumpur, Malaysia, October 4-6, pp. 778–782 (2009)
4. G. Hua, M.-H. Yang, E. Learned-Miller, Y. a. T. M. Ma, D. J. Kriegman and T. S. Huang, "Introduction to the special section on real-world face recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, pp. 1921--1924, 2011.
5. F. P. Filippidou and G. A. Papakostas, "Single Sample Face Recognition Using Convolutional Neural Networks for Automated Attendance Systems," in 2020 Fourth International Conference on Intelligent Computing in Data Sciences (ICDS), 2020.
6. M. Karunakar, C. A. Sai, K. Chandra and K. A. Kumar, "Smart Attendance Monitoring System (SAMS): A Face Recognition Based Attendance System for Classroom Environment," International Journal for Recent Developments in Science and Technology, vol. 4, no. 5, pp. 194-201, 2020.
7. S. Bhattacharya, G. S. Nainala, P. Das and A. Routray, "Smart Attendance Monitoring System (SAMS): A Face Recognition Based Attendance System for Classroom Environment." in 2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT), 2018.
8. Murel, J., PhD, & Kavlakoglu, E. (2024, August 15). Transfer Learning. *IBM*. https://www.ibm.com/topics/transfer-

learning#:~:text=Transfer%20learning%20is%20a%20machine,improve%20generalization%20in%20another%20setting.

9. *The Hidden Costs: How manual attendance tracking damages schools and disrupts parent engagement | Orah Blog*. (n.d.). https://www.orah.com/blog/hidden-costs-of-manual-attendance
10. G. Sittampalam and N. Ratnarajah, "SAMS: An IoT Solution for Attendance Management in Universities", In TENCON 2019-2019 IEEE Region 10 Conference (TENCON), IEEE, pp. 251-256, 2019.
11. T. Adiono, D. Setiawan, Maurizfa, J. William and N. Sutisna, "Cloud Based User Interface Design for Smart Student Attendance System," In International Symposium on Electronics and Smart Devices (ISESD), pp. 1-5, 2021.
12. Dubey, Neha. (2020). Face Recognition based Attendance System. International Journal of Engineering Research and. V9. 10.17577/IJERTV9IS060615.
13. Nan, Y., Ju, J., Hua, Q., Zhang, H., & Wang, B. (2021). A-MobileNet: An approach of facial expression recognition. *Alexandria Engineering Journal*, *61*(6), 4435–4444. https://doi.org/10.1016/j.aej.2021.09.066
14. Schröer, C., Kruse, F., & Gómez, J. M. (2021). A Systematic Literature Review on Applying CRISP-DM Process model. *Procedia Computer Science*, *181*, 526–534. https://doi.org/10.1016/j.procs.2021.01.199
15. Rotty, A., Dewayana, T., & Habyba, A. (2022, September). Cross-Industry Standard Process for Data Mining (CRISP-DM) Approach in Determining the Most Significant Employee Engagement Drivers to Sales at X Car Dealership. In *3rd Asia Pacific International Conference on Industrial Engineering and Operations Management*, https://doi.org/10.46254/AP03.20220552.
16. Zhou, Z. H. (2021). Machine Learning. (S. W. Liu, Trans.) Springer Nature.

17. Sah, S. (2020, July 11). Machine Learning: A Review of Learning Types. doi:10.20944/preprints202007.0230.v1
18. Delua, J. (2021, March 12). Supervised vs. Unsupervised Learning: What's the Difference? Retrieved November 29, 2024, from IBM: https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning
19. Jolliffe, I.T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374*.
20. Jolliffe, Ian. (2002). Principle Component Analysis. 24. 417-441.
21. Qiu, J., Wang, H., Lu, J., Zhang, B., & Du, K. (2012). Neural network implementations for PCA and its extensions. *ISRN Artificial Intelligence, 2012*, 1–19. https://doi.org/10.5402/2012/847305
22. Abdi, H., & Williams, L. J. (2010). Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4), 433-459. https://doi.org/10.1002/wics.101
23. GeeksforGeeks. (2021, September 24). *ML | Face Recognition using Eigenfaces (PCA Algorithm)*. GeeksforGeeks. https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/
24. Wold, Svante & Esbensen, Kim & Geladi, Paul. (1987). Principal Component Analysis. Chemometrics and Intelligent Laboratory Systems. 2. 37-52. 10.1016/0169-7439(87)80084-9.
25. Khan, M. Y., Qayoom, A., Nizami, M. S., Siddiqui, M. S., Wasi, S., & Raazi, R. (2020). Automated Prediction of Good Dictionary EXamples (GDEX): A Comprehensive Experiment with Distant Supervision, Machine Learning, and Word Embedding-Based

Deep Learning Techniques. Complexity, 2021(1), 2553199. https://doi.org/10.1155/2021/2553199

26. Abdulkareem, N. M., & Abdulazeez, A. M. (2021, January 27). Machine Learning Classification Based on Random Forest Algorithm: A Review. International Journal of Science and Business, 5(2), 128-142. doi:10.5281/zenodo.4471118
27. Quinlan J. R., Induction of decision trees, *Machine Learning*. (1986) **1**, no. 1, 81–106, https://doi.org/10.1007/bf00116251.
28. S Mucesh, W. G. (2021, April). A machine learning approach to galaxy properties: joint redshift–stellar mass probability distributions with Random Forest. Monthly Notices of the Royal Astronomical Society, 502(2), 2770-2786. doi:10.1093/mnras/stab164
29. IBM. (2024, October 25). Random Forest. *What is random forest?* https://www.ibm.com/topics/random-forest
30. Kursa, Miron & Rudnicki, Witold. (2011). The All Relevant Feature Selection using Random Forest.
31. Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324
32. Chang, C., & Lin, C. (2011). LIBSVM. *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 1–27. https://doi.org/10.1145/1961189.1961199
33. S, J., David, S., W, J., & Sundar, N. (2019, January). Support Vector Machine for Classification of Autism Spectrum Disorder based on Abnormal Structure of Corpus Callosum. International Journal of Advanced Computer Science and Applications, 10(9). doi:10.14569/ijacsa.2019.0100965
34. Rodríguez-Pérez, R., & Bajorath, J. (2022, March 19). Evolution of Support Vector Machine and Regression Modeling in Chemoinformatics and Drug Discovery. Journal of Computer-Aided Molecular Design, 36(5), 355-362. doi:10.1007/s10822-022-00442-9

35. Khan M. Y. and Junejo K. N., Exerting 2D-space of sentiment Lexicons with machine learning techniques: a hybrid approach for sentiment analysis, *International Journal of Advanced Computer Science and Applications*. (2020) **11**, no. 6, 599–608, https://doi.org/10.14569/ijacsa.2020.0110672.
36. Fan R. E., Chang K. W., Hsieh C. J., Wang X. R., and Lin C. J., Liblinear: a library for large linear classification, *Journal of Machine Learing Research*. (2008) **9**, no. 8, 1871–1874.
37. Smith, A. (2002). Principles of data mining. *Artificial Intelligence in Medicine*, *26*(1–2), 175–178. https://doi.org/10.1016/s0933-3657(02)00058-1
38. Guo, Gongde & Wang, Hui & Bell, David & Bi, Yaxin. (2004). KNN Model-Based Approach in Classification.
39. Dewi, E. S., & Imah, E. M. (2020). Comparison of Machine Learning Algorithms for. International Joint Conference on Science and Engineering (IJCSE 2020). 196, pp. 1628-1637. Atlantis Press.
40. Kumar, Munish & Sharma, R. & Jindal, M. & Jindal, Simpel. (2020). Performance Evaluation of Classifiers for the Recognition of Offline Handwritten Gurumukhi Characters and Numerals: A Study. Artificial Intelligence Review. 53. 10.1007/s10462-019-09727-2.
41. Happiness, O. (2023, May 8). Face Recognition using KNN: A Comprehensive Guide | Nur: The She Code Africa Blog. *Medium*. https://medium.com/shecodeafrica/performing-face-recognition-using-knn-fe71d87ab619
42. G. McLachlan, Discriminant analysis and statistical pattern recognition. John Wiley & Sons, 2004, vol. 544.

43. Pan, F., Song, G., Gan, X. *et al.* Consistent feature selection and its application to face recognition. *J Intell Inf Syst* **43**, 307–321 (2014). https://doi.org/10.1007/s10844-014-0324-5

44. P. Viszlay, M. Lojka and J. Juhár, Class-dependent two-dimensional linear discriminant analysis using two-pass recog-nition strategy, in: Proceedings of the 22nd European SignalProcessing Conference (EUSIPCO), IEEE, 2014, pp. 1796–1800.

45. Tharwat, Alaa & Gaber, Tarek & Ibrahim, Abdelhameed & Hassanien, Aboul Ella. (2017). Linear discriminant analysis: A detailed tutorial. Ai Communications. 30. 169-190,. 10.3233/AIC-170729.

46. Bala, M., Singh, P., & Meena, M.S. (2016). FACE RECOGNITION USING LINEAR DISCRIMINANT ANALYSIS.

47. Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

48. Zhang, Y., Sutton, C., Storkey, A., & Ghahramani, Z. (2012). Advances in Neural Information Processing Systems 25. In *MIT Press eBooks*. https://www.research.ed.ac.uk/portal/en/publications/continuous-relaxations-for-discrete-hamiltonian-monte-carlo(21c15b13-cf5b-4913-8744-8972be0713c3).html

49. Gu, H., Wang, Y., Hong, S., & Gui, G. (2019). Blind channel identification aided generalized automatic modulation recognition based on deep learning. *IEEE Access*, *7*, 110722–110729. https://doi.org/10.1109/access.2019.2934354

50. *Deep learning*. (n.d.). https://www.deeplearningbook.org/

51. Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. *International Conference on Machine Learning*, 807–814. https://icml.cc/Conferences/2010/papers/432.pdf

52. M. Coşkun, A. Uçar, Ö. Yildirim and Y. Demir, "Face recognition based on convolutional neural network," *2017 International Conference on Modern Electrical and Energy Systems (MEES)*, Kremenchuk, Ukraine, 2017, pp. 376-379, doi: 10.1109/MEES.2017.8248937.
53. M. S. Ejaz, M. R. Islam, M. Sifatullah and A. Sarker, "Implementation of Principal Component Analysis on Masked and Non-masked Face Recognition," *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, Dhaka, Bangladesh, 2019, pp. 1-5, doi: 10.1109/ICASERT.2019.8934543.
54. H. Mady and S. M. S. Hilles, "Face recognition and detection using Random forest and combination of LBP and HOG features," *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, Shah Alam, Malaysia, 2018, pp. 1-7, doi: 10.1109/ICSCEE.2018.8538377.
55. A. M. Nuruddin Pk, X. Ding and T. Page, "An Integrated Approach for Face Recognition Using Multi-class SVM," *2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, Chengdu, China, 2020, pp. 398-402, doi: 10.1109/ICCCBDA49378.2020.9095692.
56. X. Guo, "A KNN Classifier for Face Recognition," *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, Beijing, China, 2021, pp. 292-297, doi: 10.1109/CISCE52179.2021.9445908.
57. Y. Wei, "Face Recognition Method Based on Improved LDA," *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, Hangzhou, China, 2017, pp. 456-459, doi: 10.1109/IHMSC.2017.214.
58. H. Jiang and E. Learned-Miller, "Face Detection with the Faster R-CNN," *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, Washington, DC, USA, 2017, pp. 650-657, doi: 10.1109/FG.2017.82.

59. H. Li, Z. Lin, X. Shen, J. Brandt and G. Hua, "A convolutional neural network cascade for face detection," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 5325-5334, doi: 10.1109/CVPR.2015.7299170.

60. E. Winarno, I. Husni Al Amin, H. Februariyanti, P. W. Adi, W. Hadikurniawati and M. T. Anwar, "Attendance System Based on Face Recognition System Using CNN-PCA Method and Real-time Camera," *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia, 2019, pp. 301-304, doi: 10.1109/ISRITI48646.2019.9034596.

61. Staff, C. (2024, April 3). *What is Python used for? A beginner's guide*. Coursera. https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

62. Summerfield, M. (2007). *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming*. http://ci.nii.ac.jp/ncid/BB04681341

63. Kuhlman, D. (2011). *A Python book: Beginning Python, Advanced Python, and Python Exercises*. Platypus Global Media.

64. Srinath, K. R. (2017). *Python – the fastest growing programming language*. https://www.semanticscholar.org/paper/Python-%E2%80%93-The-Fastest-Growing-Programming-Language-Srinath/72dbcf413ded66553a27eefe6f6a1acef494bdbd

65. Statista. (2024, September 18). *Most widely utilized programming languages among developers worldwide 2024*. https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

66. TheKnowledgeAcademy. (n.d.). *What is OpenCV? A Complete Overview*. https://www.theknowledgeacademy.com/blog/what-is-opencv/

67. Boesch, G. (2024, October 11). *What is OpenCV? The Complete Guide (2025)*. viso.ai. https://viso.ai/computer-vision/opencv/

68. Writer, C. (2024, May 2). *What is OpenCV? A Guide for Beginners.* Roboflow Blog. https://blog.roboflow.com/what-is-opencv/

69. OpenCV. (2020, November 4). *About - OpenCV*. https://opencv.org/about/#:~:text=OpenCV%20(Open%20Source%20Computer%20Vision,perception%20in%20the%20commercial%20products.

70. BasuMallick, C. (2022b, December 27). *How Does TensorFlow Work and Why is it Vital for AI?* Spiceworks Inc. https://www.spiceworks.com/tech/devops/articles/what-is-tensorflow/

71. *What is TensorFlow?* (n.d.). NVIDIA Data Science Glossary. https://www.nvidia.com/en-us/glossary/tensorflow/#:~:text=Heavily%20used%20by%20data%20scientists,tensors)%20that%20flow%20between%20them.

72. *TensorFlow: A system for large-scale machine learning*. (n.d.). https://research.google/pubs/tensorflow-a-system-for-large-scale-machine-learning/

73. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., . . . Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1605.08695

74. Lutkevich, B. (2022, December 28). *Kotlin*. WhatIs. https://www.techtarget.com/whatis/definition/Kotlin

75. *Android's Kotlin-first approach*. (n.d.). Android Developers. https://developer.android.com/kotlin/first

76. Silverio, M. (2022, December 19). *What is Kotlin?* Built In. https://builtin.com/software-engineering-perspectives/kotlin

77. S. Hellbr¨uck, A Data Mining Approach to Compare Java with Kotlin, Metropolia Ammattikorkeakoulu, 2019.

78. *Thinking in compose*. (n.d.). Android Developers. https://developer.android.com/develop/ui/compose/mental-model#:~:text=Jetpack%20Compose%20is%20a%20modern,without%20imperatively%20mutating%20frontend%20views.

79. Partners, G. A. (2024, July 22). *Lab notes: An Introduction to Android JetPack Compose*. Growth Acceleration Partners. https://www.growthaccelerationpartners.com/blog/lab-notes-an-introduction-to-android-jetpack-compose

80. Arora, P. (2022, August 9). Comparing Jetpack Compose performance with XML - OkCredit - Medium. *Medium*. https://medium.com/okcredit/comparing-jetpack-compose-performance-with-xml-9462a1282c6b

81. Trengrove, B. (2022, June 30). Jetpack Compose Stability explained - Android Developers - medium. *Medium*. https://medium.com/androiddevelopers/jetpack-compose-stability-explained-79c10db270c8

82. Stevenson, D. (2020, May 7). What is Firebase? The complete story, abridged. - Firebase Developers - Medium. *Medium*. https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

83. Khawas, C., & Shah, P. (2018). Application of Firebase in Android App Development-A Study. *International Journal of Computer Applications, 179*(46), 49–53. https://doi.org/10.5120/ijca2018917200

84. GeeksforGeeks. (2024, May 10). *What is Firebase Authentication*. GeeksforGeeks. https://www.geeksforgeeks.org/what-is-firebase-authentication/

85. Everythingcomputerscience CS Java. (2016), https://every-thingcomputerscience.com/programming/Java.html

86. Hachadi, Z. Java Web Development Springboot Security. (LinkedIn,2023), https://tn.linkedin.com/posts/zakaria-hachadi-176b871b0java −web development −springboot −activity −7047332925712785408 −x4Y0

87. Jibble. (2025, January 2). *What are the Different Types of Attendance Systems? | 2025*. https://www.jibble.io/article/types-of-attendance-systems

88. Manori, Anupam & Devnath, Nandgopal & Pasi, Nitin & Kumar, Vivek. (2017). QR Code Based Smart Attendance System. International Journal of Smart Business and Technology. 5. 1-10. 10.21742/ijsbt.2017.5.1.01.

89. Saxena, Aman. (2021). RFID Based Attendance System. International Journal for Modern Trends in Science and Technology. 7. 40-43. 10.46501/IJMTST0701009.

90. Grover, A. (2025, January 15). *Enhance Productivity with Location-Based Attendance Tracking | Workstatus*. Workstatus. https://www.workstatus.io/blog/workforce-management/location-based-attendance-tracking/

91. Smitha, & Hegde, Pavithra & Afshin,. (2020). Face Recognition based Attendance Management System. International Journal of Engineering Research and. V9. 10.17577/IJERTV9IS050861.

92. Akinduyite, Olanike & Adetunmbi, Adebayo & Olabode, O. & Ibidunmoye, Olumuyiwa. (2013). Fingerprint-Based Attendance Management System. 1. 100-105. 10.12691/jcsa-1-5-4.

93. J. Chen and W. K. Jenkins, "Facial recognition with PCA and machine learning methods," *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Boston, MA, USA, 2017, pp. 973-976, doi: 10.1109/MWSCAS.2017.8053088.

94. Saha, C., Ghosh, D., & Paul, T. (2024). Comparison between SVM and CNN models through deployment in the Face Recognition System using real time dataset. *International Journal of Scientific and Research Publications, 14*(7), 266–274. https://doi.org/10.29322/ijsrp.14.07.2024.p15131

95. F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 815-823, doi: 10.1109/CVPR.2015.7298682.

96. Wang, J., Zhou, F., Wen, S., Liu, X., & Lin, Y. (2017). Deep Metric Learning with Angular Loss. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1708.01682

97. J. Deng, J. Guo, N. Xue and S. Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 4685-4694, doi: 10.1109/CVPR.2019.00482.