



Faculty of Computer Science and Information Technology

Smart Grading: A Mobile OMR Scanner App for Lecturers in UNIMAS

Aiman Harith bin Abdul Halek

**Bachelor of Software Engineering with Honours
2024**

Smart Grading: A Mobile OMR Scanner App for Lecturer in UNIMAS

Aiman Harith bin Abdul Halek

A thesis submitted

In fulfilment of the requirements for the degree of Bachelor of Computer Science with
Honours (Software Engineering)

Faculty of Computer Science and Information Techn

UNIVERSITI MALAYSIA SARAWAK

2024

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE Smart Grading: A Mobile OMR Scanner App for Lecturers In
UNIMAS

ACADEMIC SESSION: 2024/2025

AIMAN HARITH BIN ABDUL HALEK

(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)

Validated by



(SUPERVISOR'S SIGNATURE)

Permanent Address

No. 59, Lot 1049,
Jalan Arang,
93250 Kuching, Sarawak

Date: 24/7/2025

Date: 28/7/2025

Note * Thesis refers to PhD, Master, and Bachelor Degree

** For Confidential or Restricted materials, please attach relevant documents from relevant organizations / authorities

DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



.....

Signature

Name: Aiman Harith bin Abdul Halek

Matric No.: 77761

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date: 23/6/2025

ACKNOWLEDGEMENT

I would like to express my gratitude to my FYP supervisor, Dr. Azman bin Bujang Masli, for his guidance, support, supervision, and encouragement through my final year project. His ideas inspired this final year project. He had introduced me to the concepts of Image Processing which has proven to be an interesting field of study computer science as it has many uses and contributions to recent scientific studies. I would also like to extend my gratitude to the FYP Coordinator, Professor Dr. Wang Yin Chai, for his determination in making sure every final year student understands what needs to be done for their final year project. I would also like to extend my gratitude to Associate Professor Ts. Dr Hamimah binti Ujir for her efforts in examining my FYP report.

I would also like to thank my family for their support and encouragement when I'm feeling unwell whether it be physically or mentally throughout this project. I would also like to give my gratitude towards my friends and peers for always being there for me especially my loving partner Dayangku Faridatul Fakhirah binti Awang Mohamad, who have always given me the support, motivation and opinions to aid me in my final year project. Without their support, I would surely be at a loss.

ABSTRACT

The proposed project aims to simplify and modernise the grading of Optical Mark Recognition (OMR) answer sheets through a mobile application tailored for lecturers. Traditional methods of grading OMR papers often rely on manual processes or expensive hardware, which are time-consuming and inefficient. *Smart Grading: A Mobile OMR Scanner App for Lecturers* addresses this issue by leveraging mobile technology and image processing techniques to automate the grading process. The system is capable of scanning answer papers, detecting and correcting alignment, and evaluating results in real time. Developed using the Agile methodology, the application includes core features such as quiz creation, answer key setup, and performance analysis.

The system was implemented using FlutterFlow for the mobile frontend, Python with OpenCV for backend image processing, and Firebase for data handling and authentication. Extensive functional testing was conducted using detailed test cases to validate individual features, while integration and image processing tests ensured the reliability of the grading workflow. User Acceptance Testing (UAT) was also carried out through SUS, UEQ, and open-ended feedback to assess overall usability. The project concludes with recommendations for future improvements such as full-paper scanning, offline grading, and extended support for subjective questions. Overall, the Smart Grading system offers an efficient, accessible, and scalable solution for automating academic assessments.

Keywords: Optical Mark Recognition (OMR), mobile application, image processing, automated grading, OpenCV, Agile Methodology

ABSTRAK

Projek yang dicadangkan bertujuan untuk mempermudah dan memodenkan proses pemarkahan lembaran jawapan Optical Mark Recognition (OMR) melalui aplikasi mudah alih yang direka khas untuk pensyarah. Kaedah tradisional pemarkahan OMR selalunya bergantung kepada proses manual atau perkakasan yang mahal, yang memerlukan masa yang lama dan kurang cekap. *Smart Grading: A Mobile OMR Scanner App for Lecturers* menangani isu ini dengan memanfaatkan teknologi mudah alih dan teknik pemprosesan imej untuk mengautomasikan proses pemarkahan. Sistem ini mampu mengimbas kertas jawapan, mengesan dan membetulkan penjajaran, serta menilai keputusan secara masa nyata. Aplikasi ini dibangunkan menggunakan metodologi Agile dan merangkumi ciri-ciri utama seperti penciptaan kuiz, tetapan kunci jawapan, dan analisis prestasi.

Sistem ini dibangunkan menggunakan FlutterFlow untuk antaramuka mudah alih, Python dengan OpenCV untuk pemprosesan imej di bahagian backend, serta Firebase untuk pengurusan data dan pengesahan pengguna. Ujian fungsi dijalankan secara menyeluruh dengan menggunakan set kes ujian terperinci bagi mengesahkan ciri-ciri individu, manakala ujian integrasi dan pemprosesan imej memastikan ketepatan aliran kerja pemarkahan. Ujian Penerimaan Pengguna (UAT) turut dilaksanakan melalui SUS, UEQ dan soalan terbuka untuk menilai kebolehgunaan keseluruhan. Projek ini diakhiri dengan cadangan penambahbaikan seperti pengimbasan keseluruhan kertas, mod pemarkahan luar talian, dan sokongan tambahan bagi soalan subjektif. Secara keseluruhannya, sistem Smart Grading menawarkan penyelesaian pemarkahan akademik yang cekap, mudah diakses dan berskala.

Kata kunci: Optical Mark Recognition (OMR), aplikasi mudah alih, pemprosesan imej, pemarkahan automatik, OpenCV, Metodologi Agile

TABLE OF CONTENTS

	Page
DECLARATION.....	iv
ACKNOWLEDGEMENT.....	v
ABSTRACT	vi
ABSTRAK	vii
TABLE OF CONTENTS	viii
LIST OF FIGURES.....	xiii
LIST OF TABLES	xvi
LIST OF ABBREVIATIONS	xviii
CHAPTER 1 INTRODUCTION.....	19
1.1 Introduction	19
1.2 Problem Statement.....	20
1.3 Scope	20
1.4 Aim and Objectives	21
1.5 Brief Methodology	21
1.6 Significant of Project.....	22
1.7 Project Schedule	23
1.8 Expected Outcome.....	23
1.9 Chapter Summary.....	24

CHAPTER 2	Literature Review	26
2.1	Overview	26
2.2	Background of Study	26
2.3	Image Processing Techniques in OMR Systems	28
2.4	Review on Existing Mobile OMR Scanner Applications.....	29
2.4.1	EvalBee.....	30
2.4.2	ZipGrade.....	34
2.4.3	Exam Reader	38
2.5	Comparison Between Existing Application	41
2.6	Chapter Summary	43
CHAPTER 3	Requirement Analysis & Design	44
3.1	Overview	44
3.2	Methodology.....	44
3.3	Requirement Elicitation.....	46
3.3.1	Interview	47
3.4	Requirement Analysis	49
3.4.1	Use Case Diagram	49
3.4.2	Sequence Diagram.....	54
3.4.3	Class Diagram	61
3.5	System Design	62
3.5.1	Image Processing Approach	62

3.5.2	Activity Diagram	64
3.5.3	System User Interface Design	70
3.5.4	System Database Design	79
3.6	Chapter Summary	80
CHAPTER 4	Implementation	81
4.1	Overview	81
4.2	Technologies and Tools Overview	82
4.2.1	Technology Stack	82
4.3	Justification for Tool Selection	83
4.4	Core Application Modules	84
4.4.1	User Authentication Module	84
4.4.2	Exam Management Module.....	85
4.4.3	Course Management Module.....	88
4.4.4	Answer Key Configuration Module	91
4.4.5	Scanning and Grading Module	92
4.4.6	Exporting Results Module	95
4.5	Image Processing Workflow	96
4.5.1	Image Acquisition and Preprocessing	96
4.5.2	Contour Detection and Perspective Warping	97
4.5.3	Bubble Segmentation.....	98
4.5.4	Answer Detection and Grading	99

4.5.5	Upload and Response Delivery	100
4.6	Chapter Summary	100
CHAPTER 5	Testing	102
5.1	Overview	102
5.2	Testing Strategy	102
5.3	Test Case	104
5.4	Real-World Testing Scenarios	117
5.5	User Acceptance Testing (UAT)	118
5.5.1	Participant Profile	118
5.5.2	System Usability Scale (SUS)	119
5.5.3	User Experience Questionnaire (UEQ)	120
5.5.4	Open Ended Questions	120
5.6	Limitations and Known Issues	121
5.7	Chapter Summary	122
CHAPTER 6	Conclusion and Future Work.....	123
6.1	Overview	123
6.2	Future Work.....	123
6.2.1	Enable User to Take Pictures of the Whole Paper.....	123
6.2.2	Offline Grading Mode	124
6.3	Conclusion	124
6.3.1	Conclusion Based on Objective 1	125

6.3.2	Conclusion Based on Objective 2.....	125
6.3.3	Conclusion Based on Objective 3.....	126
6.4	Chapter Summary	126
REFERENCES		127
APPENDICES.....		129

LIST OF FIGURES

Figure 2.1:	Process of OMR scanning using scanner	27
Figure 2.2:	Process of OMR scanning using Mobile App Scanner	27
Figure 2.3:	Creating a new exam sheet in EvalBee	31
Figure 2.4:	Subscription plans in EvalBee	33
Figure 2.5:	Creating a new answer sheet on ZipGrade	35
Figure 2.6:	Cloud Integration in ZipGrade	36
Figure 2.7:	Creating a new answer sheet in Exam Reader.....	38
Figure 2.8:	Generated answer sheet from Exam Reader.....	40
Figure 2.9:	Pop-up after a successful scan in Exam Reader	41
Figure 3.1:	Agile Methodology	44
Figure 3.2:	Smart Grading Use Case Diagram.....	49
Figure 3.3:	Sequence Diagram for Lecturer Authentication	55
Figure 3.4:	Sequence Diagram for Manage Exam	56
Figure 3.5:	Sequence Diagram for Manage Course	57
Figure 3.6:	Sequence Diagram for Set Answer Key	58
Figure 3.7:	Sequence Diagram for Scan Answer Papers	59
Figure 3.8:	Sequence Diagram for Analysis and Export Data	60
Figure 3.9:	Smart Grading Class Diagram.....	61
Figure 3.10:	Activity Diagram for Exam Management	65
Figure 3.11:	Activity Diagram for Course Management	66
Figure 3.12:	Activity Diagram for Grading Student's Answer Paper	67
Figure 3.13:	Activity Diagram for Exam's Analysis	68
Figure 3.14:	Activity Diagram for Exporting Data.....	69

Figure 3.15:	Landing Page / Exam List	70
Figure 3.16:	Exam Management	71
Figure 3.17:	Course List.....	72
Figure 3.18:	Course Management	73
Figure 3.19:	Answer Key Page	74
Figure 3.20:	Scanning Student Answer Paper	75
Figure 3.21:	Exam Analysis	76
Figure 3.22:	Export to Excel	77
Figure 3.23:	Error Handling	78
Figure 3.24:	Entity Relationship Diagram	79
Figure 4.1:	API Call to Flask Backend via FlutterFlow	83
Figure 4.2:	Login and Sign-Up Page	84
Figure 4.3:	Exam Dashboard	85
Figure 4.4:	Exam Details Page.....	86
Figure 4.5:	Add and Edit Exam Dialogs	87
Figure 4.6:	Course List Interface	88
Figure 4.7:	Add New Course Dialog	89
Figure 4.8:	Course Details Page - Linked Exams	90
Figure 4.9:	Creating and Scanning the Answer Key.....	91
Figure 4.10:	Student Matric Entry and Scan Page for Answer Papers	92
Figure 4.11:	Confirmation screen after answer box is successfully detected.	93
Figure 4.12:	Error screen when no answer box is detected.	94
Figure 4.13:	Exporting exam data report.	95
Figure 4.14:	Image reading and preprocessing	96

Figure 4.15:	Code for contour detection and perspective warp	97
Figure 4.16:	Slicing the warped image into rows and answer options.....	98
Figure 4.17:	Counting pixels and determining selected answers.....	99
Figure 4.18:	Uploading the marked image and returning results.....	100
Figure 5.1:	Grading Accuracy Distribution	117
Figure 5.2:	Error Breakdown by Testing Condition	117

LIST OF TABLES

Table 2.1:	Comparison between Pre-existing App	42
Table 3.1:	Use Case for View Exam Details	50
Table 3.2:	Use Case for Edit Exam Details	50
Table 3.3:	Use Case for Delete Exam.....	50
Table 3.4:	Use Case for Add New Exam.....	51
Table 3.5:	Use Case for View Course Details	51
Table 3.6:	Use Case for Edit Course	51
Table 3.7:	Use Case for Delete Course.....	52
Table 3.8:	Use Case for Add New Course.....	52
Table 3.9:	Use Case for Set Answer Key	52
Table 3.10:	Use Case for Scan Student's Answer Paper.....	53
Table 3.11:	Use Case for Validate Scanned Answer Papers	53
Table 3.12:	Delete Scanned Answer Papers	53
Table 3.13:	Use Case for View Exam's Analysis.....	54
Table 3.14:	Use Case for Export Data	54
Table 4.1:	Core Technologies Used in Smart Grading.....	82
Table 5.1:	User Authentication Test Cases.....	104
Table 5.2:	Exam Management Module Test Cases	106
Table 5.3:	Course Management Module Test Cases	109
Table 5.4:	Answer Key Configuration Module Test Cases	111
Table 5.5:	Scanning and Grading Module Test Cases.....	113
Table 5.6:	Exporting Data Module Test Cases	115
Table 5.7:	SUS Table.....	119

Table 5.8: UEQ Summary Table 120

LIST OF ABBREVIATIONS

API	: Application Programming Interface
CSV	: Comma-Separated Values
ERD	: Entity Relationship Diagram
FYP	: Final Year Project
JSON	: JavaScript Object Notation
MCQ	: Multiple Choice Questions
NEET	: National Eligibility-cum-Entrance Test
OCR	: Optical Character Recognition
OMR	: Optical Mark Recognition
PDF	: Portable Document Format
SDLC	: Software Development Life Cycle
SDK	: Software Development Kit
SUS	: System Usability Scale
UAT	: User Acceptance Testing
UEQ	: User Experience Questionnaire
UI/UX	: User Interface / User Experience
UNIMAS	: Universiti Malaysia Sarawak
URL	: Uniform Resource Locator

CHAPTER 1

INTRODUCTION

1.1 Introduction

Optical Mark Recognition (OMR) is a widely used technology for collecting data from users by identifying marked responses on paper (Dharmik et al., 2024). This technology has proven valuable in various contexts, such as examinations, surveys, feedback forms, and aptitude tests, due to its efficiency and accuracy in processing large volumes of data quickly. Traditionally, OMR scanning requires specialized hardware, which can be costly, cumbersome, and accessible only in specific locations.

In response to these limitations, this project introduces Smart Grading: A Mobile OMR Scanner App for Lecturers in UNIMAS, which seeks to bring the power of OMR technology to mobile devices. By allowing lecturers to grade exams and process OMR papers directly from their smartphones, this app provides a flexible, accessible, and efficient solution that is especially beneficial in educational environments where resources may be limited or where quick grading is essential. The integration of advanced image processing techniques in mobile applications has been shown to enhance the accuracy of OMR systems significantly (AbhiSubrahmanyam et al., 2022). With this app, lecturers can streamline the grading process, reduce dependence on traditional OMR hardware, and improve overall efficiency in academic assessment.

This project aims to demonstrate the practicality of mobile-based OMR technology for educational use. The app will integrate advanced image recognition techniques to ensure user-friendly interfaces to support efficient grading workflows and provide secure data management features to safeguard student information. Recent evaluations have also

indicated that mobile OMR systems can achieve accuracy rates of up to 99.7% when recognizing marked responses on answer papers (De Elias et al., 2021).

1.2 Problem Statement

In today's academic environment, efficient and accurate grading is crucial for timely feedback and effective learning. However, the grading process, especially when using Optical Mark Recognition (OMR) technology, faces several challenges:

Traditional OMR scanning systems rely on specialized hardware that can be costly and prone to breakdowns. When an OMR scanner malfunctions or becomes unavailable, lecturers lack an alternative solution, causing interruptions and delays in the assessment process. Furthermore, without access to OMR scanners, lecturers may be forced to grade OMR papers manually, which is laborious and time-consuming. This manual process not only increases the workload but also heightens the risk of grading errors, leading to potential discrepancies in student evaluations.

These challenges highlight the need for an accessible, reliable alternative to traditional OMR scanners and manual grading that empowers lecturers to efficiently grade exams without specialized hardware.

1.3 Scope

The scope of this proposed project is divided into two main components: mobile application development and Optical Mark Recognition (OMR) functionality integration.

The mobile application development focuses on designing and developing a user-friendly application for lecturers. This app will include essential features such as a camera interface for scanning answer papers, result storage, and a grading interface that displays

scanned results for easy review. Ensuring a seamless and intuitive user experience is a key priority in this phase.

The OMR functionality integration involves implementing image processing techniques to detect and recognise marked answers on scanned papers. By leveraging OpenCV, the system will develop algorithms capable of handling various conditions, such as different lighting and paper orientations, to improve accuracy. This component also includes real-time grading generation and summary reports, allowing lecturers to obtain immediate grading results.

This project will be implemented within a structured timeline, ensuring that both the application interface and the OMR recognition algorithm undergo thorough testing to guarantee usability in real-world academic settings.

1.4 Aim and Objectives

The aim for this project is to create a specialized mobile application that could grade OMR answer papers using image processing and the camera of the device.

- i. Develop a mobile app that allows UNIMAS lecturers to scan OMR answer papers using their phone's camera.
- ii. Able to integrate OMR technology onto the mobile app to detect and process marked answer on OMR papers
- iii. Automate the grading of OMR answer papers to help UNIMAS lecturers save time and reduce grading errors.

1.5 Brief Methodology

The development of the Smart Grading mobile app will follow the Agile Methodology, which is an iterative approach that emphasizes flexibility, collaboration, and ongoing feedback. Agile is a methodology where continuous iterations and testing take place

during the entire Software Development Life Cycle (SDLC) of a product (Srivastava et al., 2017). Agile is well-suited for this project, allowing for incremental improvements and adjustments based on user feedback throughout the development process.

The project will be divided into multiple sprints where each sprint will focus on a specific set of features or tasks, including design, coding, testing, and review stages, to ensure rapid, continuous progress. At the end of each sprint, a functional part of the app will be delivered, allowing for user testing and feedback that can guide the next stages of development.

Sprint structure consists of:

1. Planning: Define sprint goals, including features or tasks to be completed.
2. Design: Create the UI/UX elements for the assigned tasks.
3. Development: Implement and code the features based on design specifications.
4. Testing: Perform unit and integration testing to ensure functionality.
5. Review and Retrospective: Gather feedback, evaluate performance, and refine the approach for the next sprint.

Using Agile allows the Smart Grading app to evolve based on real-time input and adapt to any changes needed, ensuring that the final product aligns closely with user needs and expectations.

1.6 Significant of Project

The Smart Grading mobile app aims to significantly improve the grading process within UNIMAS by providing a modern and accessible solution for Optical Mark Recognition (OMR). This project has the potential to bring several key benefits.

One of the main advantages is increased accessibility and affordability. Traditional OMR scanners are often costly and limited to specific locations, making them inaccessible in certain situations. By eliminating the need for specialised hardware, the Smart Grading app provides a more affordable alternative that requires only a smartphone, making grading more convenient for lecturers.

The app also enhances efficiency in grading processes by reducing the time and effort required for manual grading. Handling large volumes of exams manually can be tedious and error-prone, but with the integration of image recognition algorithms, the system automates the process, minimising reliance on manual efforts and ensuring greater accuracy.

Additionally, the app offers flexibility and convenience by allowing lecturers to grade exams from anywhere. This improves workflow efficiency, enabling them to manage grading tasks on their schedules and complete assessments more quickly. By integrating these features, the Smart Grading app provides an effective and practical solution to streamline the grading process at UNIMAS.

1.7 Project Schedule

The project schedule was organised based on the timeline outlined in the TMF4913 Final Year Project (FYP) course. Each phase of the system development was planned to follow the course milestones closely, helping ensure that all reports, deliverables, and chapters were completed on time. The schedule covers the entire span of both FYP 1 and FYP 2. A full breakdown can be found in the Gantt Chart shown in Appendix A.

1.8 Expected Outcome

The development of the Smart Grading: A Mobile OMR Scanner App for lecturers in UNIMAS is expected to deliver several positive outcomes.

First, it will provide a fully functional mobile application that allows lecturers to scan OMR answer papers using their smartphone cameras. Designed with a user-friendly interface, the app will streamline the grading process and improve efficiency.

Additionally, the app serves as a cost-effective and portable solution by eliminating the need for traditional OMR hardware. Instead of relying on expensive and location-dependent scanning equipment, lecturers can conduct grading using a standard smartphone, making the process more accessible and affordable.

Beyond functionality and cost savings, the app is expected to make an impactful contribution to academic settings by enhancing grading accessibility, affordability, and efficiency. By automating the grading process, the system improves assessment workflows, allowing lecturers to provide timely feedback while reducing manual workload.

Overall, the Smart Grading app aims to revolutionise the grading process by offering a cost-effective, accurate, and convenient solution that empowers lecturers to conduct assessments efficiently and effectively, ultimately transforming academic evaluation methods at UNIMAS.

1.9 Chapter Summary

Chapter 1 introduces the development of the Smart Grading: A Mobile OMR Scanner App for Lecturers. This project aims to provide a mobile application that enables UNIMAS lecturers to scan and grade OMR answer papers using their smartphone cameras. The project addresses the challenges of hardware dependence and time-consuming manual grading by offering a cost-effective, portable, and efficient alternative.

The app will utilize advanced image processing techniques with OpenCV to detect marked answers under various conditions and provide real-time grading results. The Agile

Methodology is chosen for the project, ensuring iterative development and continuous feedback to refine the app.

The success of this project is expected to significantly improve the grading process by increasing accessibility, efficiency, and accuracy, addressing the limitations outlined in the problem statement.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

Chapter 2 reviews the literature on Optical Mark Recognition (OMR) technology, highlighting its evolution from traditional systems reliant on costly specialized hardware to more accessible and economical mobile-based solutions. Derived from a comprehensive evaluation of academic articles and research papers sourced from reliable platforms such as IEEE Xplore and Google Scholar (2019–2020), it explores key image processing techniques like edge detection, alignment correction, and error detection that enhance the accuracy and flexibility of mobile OMR systems. The chapter also examines existing mobile OMR applications, including EvalBee, ZipGrade, and Exam Reader, analyzing their features, limitations, and suitability for educational use. While these apps offer functionalities like customizable templates and cloud integration, they reveal gaps such as reliance on proprietary designs and limited customization. These findings underscore the need for the proposed "Smart Grading" application, which aims to address these shortcomings by providing a customizable, user-friendly, and cost-effective solution tailored for UNIMAS educators.

2.2 Background of Study

Optical Mark Recognition (OMR) is a technique that gathers data from individuals by detecting marked replies on specifically designed paper forms, including tests, surveys, and feedback sheets. White-Dzuro et al. (2021) conducted research integrating Optical Mark Recognition (OMR) with Optical Character Recognition (OCR), establishing its efficacy by

processing an average of 108 documents daily, with each document requiring around 157 seconds for completion.

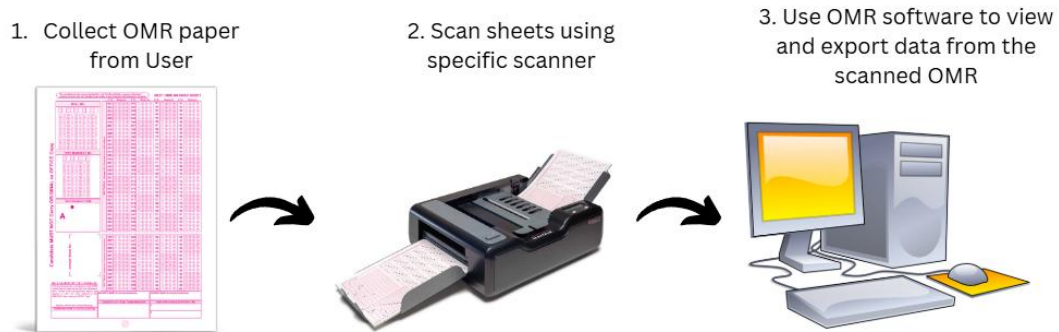


Figure 2.1: Process of OMR scanning using scanner

Conventional OMR systems, as shown in the figure 2.1 depend on specialized hardware scanners and software, facilitating great precision in data processing but frequently necessitating expensive and resource-intensive configurations. Although successful in controlled settings, classic OMR systems exhibit limitations in accessibility and flexibility due to their reliance on equipment and stringent criteria, including perfect paper alignment and high-quality printing.

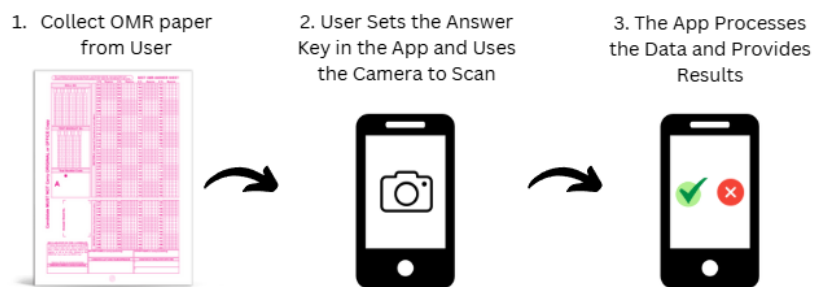


Figure 2.2: Process of OMR scanning using Mobile App Scanner

Mobile devices have significantly improved the accessibility of OMR technology, building upon recent developments. These solutions utilize the capabilities of smartphones and libraries such as OpenCV to replicate the functions of specialized OMR scanners. Methods include edge detection, thresholding, and contour analysis are utilized to identify and analyse marked replies, even in difficult settings such as inconsistent illumination or misaligned answer sheets. This advancement has rendered OMR technology more accessible, economical, and portable, especially for educational institutions and small enterprises.

2.3 Image Processing Techniques in OMR Systems

Optical Mark Recognition (OMR) systems, particularly those implemented on mobile platforms, rely heavily on sophisticated image processing algorithms to ensure the precise and reliable identification of marked answers. These techniques address common challenges such as inconsistent lighting, image distortion, and variations in marking styles, ultimately enhancing the flexibility and functionality of OMR systems.

To improve image quality before processing, preprocessing techniques are applied. One crucial step is grayscale conversion, which simplifies the image by retaining only the deepest black and the brightest white, making it easier to analyse. Additionally, noise reduction techniques, such as Gaussian blur, help eliminate unwanted artifacts caused by inconsistent lighting or low-resolution images, improving the clarity of detected marks (Qiao, 2024).

Once preprocessing is complete, edge detection plays a key role in identifying and distinguishing different areas within the image. The widely used Canny edge detection

algorithm is employed to detect sharp intensity variations, which helps define the boundaries of objects within the scanned sheet (Rangkuti et al., 2023).

To further enhance accuracy, alignment and skew correction methods are used to rectify misaligned or rotated answer sheets. The Hough Transform algorithm detects skewed lines and corrects them, ensuring that answer sheets are properly aligned for processing. Research has shown that this technique can reduce alignment error rates from 7.22% to as low as 0.44% (Hafeez et al., 2022).

The final stage of processing involves mark detection and classification, where the system analyses pixel intensity within answer areas to determine whether a bubble has been marked. A predefined intensity threshold is used to classify responses, ensuring that only intentionally marked answers are recorded.

To further improve accuracy, error detection algorithms are implemented to identify and correct common issues such as incomplete markings or overlapping answers. Statistical validation techniques help distinguish genuine marks from artifacts, ensuring the final grading results are reliable and consistent.

In conclusion, the integration of these image processing techniques significantly enhances the accuracy and efficiency of mobile-based OMR systems. By overcoming the limitations of conventional OMR scanners, these advancements expand the applicability of OMR technology across various educational and administrative settings.

2.4 Review on Existing Mobile OMR Scanner Applications

There are 3 related mobile OMR scanner applications that are being analysed and reviewed in this section.

2.4.1 EvalBee

EvalBee is a widely utilized OMR scanner application that is available on Android, iOS and as a web application. It was developed to enable the scanning and grading of OMR answer sheets via camera or uploaded image. The application allows users to utilize smartphone cameras to scan answer sheets and automatically provide comprehensive results. EvalBee accommodates several examination forms, rendering it a flexible option for many educational environments. Its user-friendliness and accessibility serve educators requiring fast grading tools without dependence on costly technology.

Cuerdo et al. (2021) conducted research to investigate the efficacy of EvalBee as a substitute for manual evaluation of OMR papers in grading test results. Their findings revealed a distinct disparity in efficiency, with the automated OMR procedure markedly decreasing the time needed for evaluation relative to manual techniques. Nevertheless, the investigation indicated no substantial discrepancies in accuracy and dependability between the two methodologies. These findings further validate the reliability and feasibility of OMR mobile applications in educational contexts as dependable substitutes for conventional manual grading techniques.

Key Features:

The screenshot shows the EvalBee app interface on a mobile device. At the top, a status bar displays the time 10:06, signal strength, and 50% battery. The app screen has a dark blue header with 'Subject 1'. Below this is a white box labeled 'Section1'. A 'Number of question' dropdown menu is set to '225'. A 'Question type' dropdown menu is open, showing options: 'TrueOrFalse', '3 option', '4 option', '5 option', '6 option', '8 option', '10 option', 'Matrix', and 'Numerical'. Below the dropdowns are labels for 'Marks for correct', 'Marks for incorrect', 'Allow partial marks', and 'Allow optional attempts'. At the bottom, there are 'Previous' and 'Next' buttons.

Figure 2.3: Creating a new exam sheet in EvalBee

EvalBee is designed to accommodate a wide range of exam formats by allowing the creation of templates with up to 225 questions per section and a maximum of 10 sections, as illustrated in Figure 2.3. This capacity ensures that EvalBee can support standardized tests such as the National Eligibility-cum-Entrance Test (NEET), which consists of 200 questions (Pushparaj & Vidya, 2024).

To enhance versatility, EvalBee supports multiple question types, including multiple-choice questions (MCQs) with 3–10 answer options, true/false questions, matrix-based questions, and numerical responses. This extensive range of supported formats makes it suitable for various assessment styles, catering to different evaluation needs.

Additionally, EvalBee offers highly customizable templates, enabling educators to tailor them to specific exam formats and institutional requirements. Users can modify the layout to efficiently organize questions and fields, edit labels to align with grading schemes or instructional needs, and adjust headers to incorporate course details or institutional branding. This adaptability ensures that the templates maintain a professional and clear structure for diverse assessment scenarios.

To further streamline the grading process, EvalBee allows manual entry of correct answers into the answer key. This functionality provides flexibility in handling various question formats, such as multiple-choice, true/false, or numerical answers, and allows educators to update or correct answers as needed. By simplifying answer key input, EvalBee enhances efficiency in exam evaluation.

Limitations:

EvalBee relies on a custom layout for processing scanned images, meaning that institutions must use the application's predefined format. If an institution already has its own OMR sheets, EvalBee will be unable to process the images, limiting its compatibility with existing systems.

For data management, EvalBee allows users to export exam results to Excel and share them via email or SMS, facilitating easy distribution and record-keeping. However, cloud synchronization for data storage is only available through a premium subscription, restricting free users to local data management options.

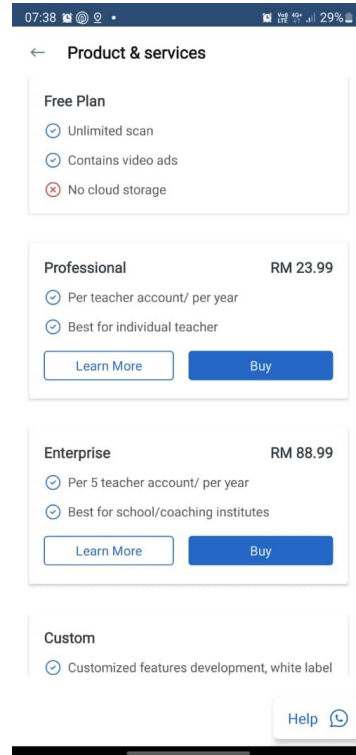


Figure 2.4: Subscription plans in EvalBee

Regarding cost structure, the premium subscription may reach an annual cost of RM88.99, as shown in Figure 2.4. This subscription unlocks additional features, including cloud synchronization, an ad-free experience, and an increased number of scanning attempts. However, the presence of advertisements in the free version may lead to user fatigue, unresponsiveness, or frustration, as highlighted by Guo and Jiang (2024). Additionally, the pricing may deter educators who prefer a completely free solution, making cost a potential barrier for some users.

2.4.2 ZipGrade

ZipGrade is another OMR application that is available on Android, iOS and as a web application. It was designed for educators to optimize the grading of multiple-choice examinations with OMR technology. The application allows educators to use their device's camera to scan answer sheets or upload scanned sheets from a printer or other hardware via the web application, providing immediate feedback on student performance. Educators may obtain the printed answer sheets in three basic formats, accommodating examinations with up to 20, 50, or 100 questions. Furthermore, ZipGrade facilitates the entry of answer keys either manually or by the scanning of a pre-completed page, therefore streamlining the quiz preparation process.

Significant research conducted by Calaguas and Consunji (2022) assessed the accuracy of ZipGrade as a substitute for human grading in OMR paper evaluations. Their investigation demonstrated no substantial disparity between the precision of human grading and ZipGrade's automated scanning, therefore validating the application's reliability as a grading instrument. This discovery underscores the application's capacity to conserve time and diminish mistakes while maintaining grading quality.

Key Features

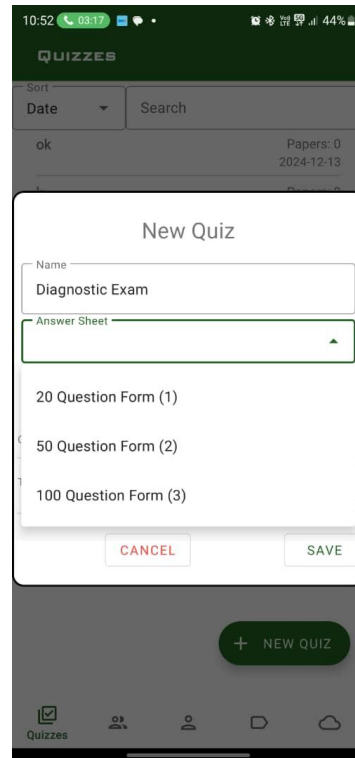


Figure 2.5: Creating a new answer sheet on ZipGrade

ZipGrade offers limited customizability in its templates, as it features a fixed header that includes fields for the student's name, date, and period. Educators have the option to select the number of questions per form either 20, 50, or 100 but the system is restricted to multiple-choice questions (MCQs) with only five answer choices. This lack of flexibility may be a limitation for institutions that require more diverse question formats or custom header modifications, as illustrated in Figure 2.5.

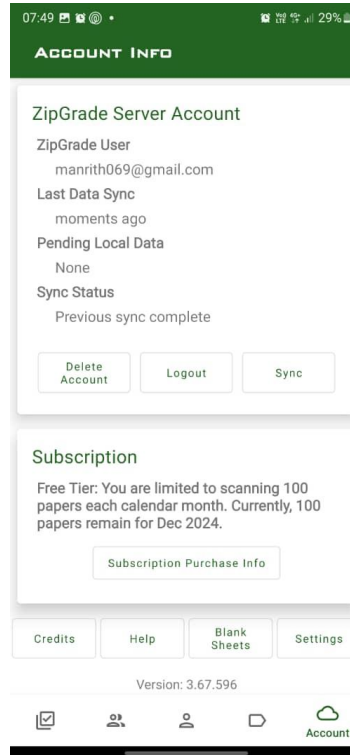


Figure 2.6: Cloud Integration in ZipGrade

ZipGrade is designed to function offline for scanning and grading, making it highly suitable for environments with limited or no internet access. Once an internet connection is available, the application synchronizes data with its web platform, allowing users to store, analyze, and export reports in PDF or CSV formats for further review or documentation, as illustrated in Figure 2.6.

For answer key input, ZipGrade offers a flexible approach by enabling educators to either manually enter the correct answers or scan them from pre-completed sheets. This dual-option feature accommodates different grading workflows, providing both the convenience of quick setup with scanned answer sheets and the precision of manual entry for adjustments or verification.

Limitations

ZipGrade has a limited question capacity, as its standard templates support a maximum of 100 questions. This restriction may be insufficient for extensive examinations, requiring educators to split exams exceeding this limit into multiple response sheets. Such partitioning can complicate the grading process, increasing both administrative effort and the potential for errors.

Additionally, ZipGrade depends on its designated response sheet templates for accurate processing, making it incompatible with pre-existing OMR sheets used by some institutions. This requirement forces educators to adopt ZipGrade's format, potentially disrupting established examination procedures and necessitating the rewriting of large numbers of answer sheets.

Regarding its cost structure, ZipGrade's free plan allows only 100 scans per month, which may not be adequate for educators handling large classes or conducting frequent assessments. To unlock unlimited scans, users must subscribe to a premium plan costing RM31.99 per year. While this is relatively affordable, it may still be a limitation for those seeking a completely free solution.

2.4.3 Exam Reader

Exam Reader is a mobile application intended for educators to facilitate the grading of multiple-choice assessments and quizzes using smartphone cameras. The application is compatible with both Android and iOS devices and accommodates many answer sheet types. It supports evaluations consisting of 5 to over 200 questions, with options ranging from 2 to 7 choices per question. Exam Reader further has functionalities such as scanning horizontal and vertical answer sheets and interpreting erased marks, rendering it especially efficient for unclear solutions.

Key Features

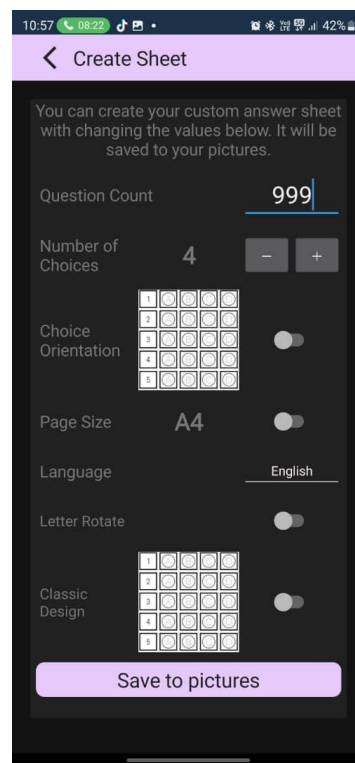


Figure 2.7: Creating a new answer sheet in Exam Reader

Exam Reader offers a high degree of customizability, allowing educators to design or modify answer sheets to fit specific examination formats. Users can adjust elements such as orientation and page size, as illustrated in Figure 2.7. This flexibility ensures compatibility with various exam structures, making Exam Reader a suitable solution for diverse educational environments.

In terms of question capacity, Exam Reader supports assessments with over 200 questions and accommodates between 2 to 7 answer choices per question. This extensive capacity makes it ideal for quizzes, standardized tests, and more comprehensive evaluations that require a broader range of answer options.

For answer key input, Exam Reader provides a dual approach by supporting both manual entry and scanning of answer keys from pre-completed sheets. This feature allows educators to quickly configure grading using scanned answer sheets while also offering the precision of manual input for more customized assessments.

Additionally, Exam Reader simplifies data management by offering cloud storage for synchronizing and storing scanned results. This functionality enhances data accessibility, enabling educators to retrieve exam results from multiple devices and export data for further analysis and documentation.

Limitations

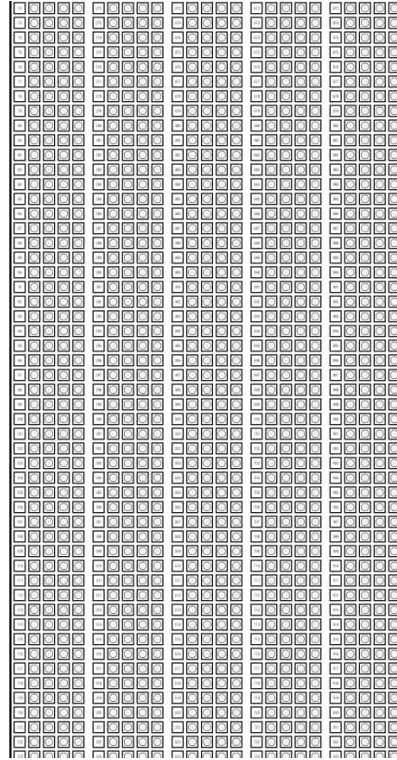


Figure 2.8: Generated answer sheet from Exam Reader

Exam Reader's performance with large assessments presents a potential limitation, as educators can add more questions than a single OMR sheet can accommodate. This may result in an incomplete answer sheet on a single page, as shown in Figure 2.8, which could complicate the examination and grading process by requiring additional pages or adjustments.

Furthermore, Exam Reader relies on its proprietary response sheet formats, which may pose challenges for universities that already use pre-existing OMR sheets. This dependence could necessitate additional configuration or redesigns to align with the application's requirements, potentially disrupting established workflows and increasing administrative workload.

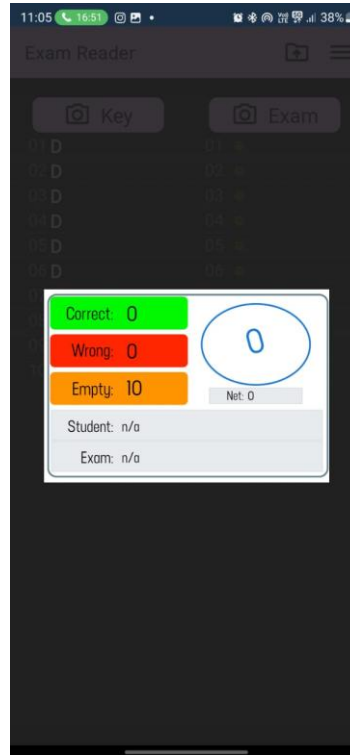


Figure 2.9: Pop-up after a successful scan in Exam Reader

Exam Reader exhibits inconsistencies in its user interface (UI) and user experience (UX) design, which may affect usability. As shown in Figure 2.9, the pop-up window displaying scanning results features a different color scheme and layout compared to the primary interface, disrupting design coherence. This lack of visual consistency can make navigation less intuitive for users. Additionally, the app's overall design appears somewhat basic, which may reduce user engagement and impact the overall experience, making it feel less polished compared to more visually refined alternatives.

2.5 Comparison Between Existing Application

Mobile-based Optical Mark Recognition (OMR) applications have made grading easier for educators, but they differ in features, usability, and adaptability. This section compares three popular applications which is EvalBee, ZipGrade, and Exam Reader

highlighting their strengths and limitations. A table has been created to clearly highlight the differences between each app.

Table 2.1: Comparison between Pre-existing App

Feature	EvalBee	ZipGrade	Exam Reader
Target Audience	Eligible for all educators	Eligible for all educators	Eligible for all educators
Question Types Supported	• MCQ (3 - 10 options) • True/False • Matrix • Numerical	• MCQ (5 options only)	• MCQ (2 – 10 options)
Number of Questions	Up to 225 questions per section with a maximum of 10 sections	20, 50 and 100 questions per template	200+ questions but excessive number of questions may overlimit the page
Customisability Of Templates	Able to change the layout, label and header of the template	Fixed templates	Able to change the orientation, page size (A4 & Letter) and design
Answer key	Allows manual entry of correct answers for grading.	Supports manual entry or scanning of answer keys from pre-completed sheets.	Supports manual entry or scanning of answer keys from pre-completed sheets.
Data Management	• Can be export to Excel and shares via email • Require subscription for cloud sync	Free cloud sync for data storage	Offers cloud integration for syncing and analytics
Platform	• Android, iOS and web application	• Android, iOS and web application	• Android and iOS • Web application for viewing detailed reports
Cost Structure	Free with ads Premium plans up to RM88.99	Free for 100 scans/month RM31.99 /year for unlimited	Free

2.6 Chapter Summary

This chapter has offered a thorough examination of the current literature about Optical Mark Recognition (OMR) systems and mobile applications, highlighting their development, technological progress, and practical uses. It underscored the constraints of conventional OMR systems, including reliance on costly hardware, and demonstrated how mobile-based solutions are mitigating these issues by providing economical and accessible alternatives.

Additionally, this chapter analysed many image processing techniques, such as edge detection, alignment correction, and error detection, which are essential for maintaining the accuracy and dependability of mobile OMR systems. An in-depth analysis was conducted on three mobile OMR scanner applications which are EvalBee, ZipGrade, and Exam Reader to evaluate their features, limits, and appropriateness for educational settings.

This chapter ultimately revealed deficiencies in current solutions, including restricted customization choices and incompatibility with existing institutional templates, which the proposed "Smart Grading" program seeks to rectify. The results of the literature research provide a basis for developing a customized mobile OMR solution for lecturers at UNIMAS, consistent with the project's goals.

CHAPTER 3

REQUIREMENT ANALYSIS & DESIGN

3.1 Overview

This chapter will highlight the methodology used to develop the OMR mobile scanner application. The methodology chosen for this project is Agile Methodology which is a recognized approach to software development. This chapter is important as it guides the development of the project.

3.2 Methodology

The development of the Smart Grading: A Mobile OMR Scanner App for Lecturers adopts the Agile Methodology, which emphasizes iterative progress, user feedback, and adaptability. This approach ensures the system meets the dynamic needs of lecturers and supports rapid adjustments based on stakeholder input.

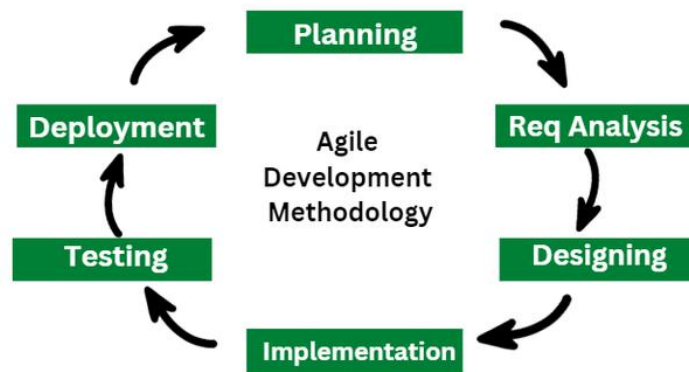


Figure 3.1: Agile Methodology

Source: *What is Agile Methodology?* (2024, September 23). GeeksforGeeks.

<https://www.geeksforgeeks.org/what-is-agile-methodology/>

Agile is particularly suitable for this project as it allows for incremental development, enabling the continuous integration of functionalities such as quiz creation, answer paper scanning, and report generation. Each iteration brings the app closer to meeting user expectations while addressing any challenges that arise during development.

The Agile methodology for this project is structured into six key phases, each ensuring a smooth and iterative development process. The Planning Phase began with interviews conducted with lecturers to gain insights into their grading workflows and identify common pain points. Based on these discussions, essential system features such as quiz creation, OMR scanning, and reporting were defined and prioritized. The primary objective of this phase was to gather clear requirements and establish a structured development roadmap.

Following this, the Requirements Analysis Phase focused on breaking down the gathered requirements into actionable development tasks. Various system modeling techniques, including use case diagrams, sequence diagrams, and class diagrams, were employed to visualize system interactions and features. These diagrams provided a structured framework for understanding user needs, ensuring a well-defined foundation for subsequent phases.

In the Design Phase, wireframes and prototypes were created to outline the app's interface and user interactions. Additionally, a database schema was designed to support data storage for courses, exams, and scanned papers. The objective of this phase was to establish both a user-friendly interface and a strong technical foundation, ensuring seamless functionality and ease of use.

The Development Phase involved implementing core features incrementally, such as courses and quiz management, OMR scanning, and reporting functionalities. Regular

feedback loops allowed for continuous refinement and enhancement of the system. The modular approach ensured flexibility, making it easier to integrate and expand features as needed.

Once development was complete, the Testing Phase was carried out through multiple levels of validation. Unit testing was performed to verify the functionality of individual modules, integration testing ensured smooth interactions between different components, and user acceptance testing (UAT) was conducted with lecturers to validate the app's usability. The goal of this phase was to identify and resolve any issues, ensuring the system met user expectations and functioned optimally under various conditions.

Finally, the Deployment Phase marked the completion of the development cycle. The app was finalized and deployed for use by lecturers, accompanied by training sessions to familiarize them with its features. Feedback was actively collected post-deployment to identify potential areas for future improvements. The primary objective of this phase was to ensure a smooth transition into practical use, delivering a reliable and user-friendly application that effectively supports educators in their grading processes.

3.3 Requirement Elicitation

Requirement elicitation is the process of gathering and defining the needs and expectations of stakeholders to ensure the system meets its intended purpose. For this project, elicitation was essential to identify the specific requirements for functionalities such as quiz management, answer paper scanning, and report generation. The primary technique used was interviews to gather insights into the lecturers' workflows and challenges.

3.3.1 Interview

The interview was conducted with Madam Wee Bui Lin, a lecturer at UNIMAS, to gather insights into the grading processes and challenges associated with OMR answer papers.

Interview Questions and Responses

1. Can you describe your current process for grading OMR answer papers?
 - **Response:** Grading is done manually by mapping the answers to a mold answer paper.
2. What are the most significant challenges you face when grading OMR papers?
 - **Response:** Corrections are a major challenge, especially when a wrong rubric is used or there is an error in marking the mold answer paper. The process of rechecking and mapping answers is time-consuming to ensure accuracy.
3. What tools or systems do you currently rely on for grading, and what are their limitations?
 - **Response:** None. The process is entirely manual.
4. Do you think this app would reduce your grading workload significantly? Why or why not?
 - **Response:** Yes, it would significantly reduce the workload. The app would eliminate the need to manually count correct answers, map answers, or recount, making the process easier and faster.
5. What concerns or reservations would you have about used a mobile OMR scanner app?

- **Response:** The primary concerns are ensuring flexibility in image capture angles and maintaining accuracy if answers are not properly erased, or multiple answers are detected due to unclear markings. Addressing these issues would enhance the reliability of the app.

6. Are there any specific features or functions that you think would improve this app?

- **Response:** Features like keeping records of all checked answer papers, providing error warnings during image capture, and auto-generating Excel grading reports would greatly improve usability and efficiency.

Key Findings

The interview provided valuable insights into the challenges and expectations surrounding the grading process, highlighting key areas for improvement. One of the primary concerns was the labor-intensive nature of manual grading, which is not only time-consuming but also prone to human errors. Educators emphasized the need for an automated solution that could streamline the grading process and minimize inaccuracies.

A mobile OMR scanner app emerged as a potential game-changer, offering the ability to significantly reduce workload while improving overall efficiency. However, for the app to be effective, certain usability concerns needed to be addressed. Flexibility in image capturing and ensuring clear, high-quality scans were identified as critical factors that would enhance the app's practicality in real-world use.

Additionally, educators highlighted the importance of incorporating essential features to improve the overall user experience. These included robust record-keeping capabilities, error detection warnings to flag potential scanning mistakes, and automated Excel report generation for seamless result management. By integrating these functionalities,

the app would not only simplify grading but also provide educators with a more reliable and efficient assessment tool.

3.4 Requirement Analysis

Requirement analysis is a critical step in the development of the Smart Grading: A Mobile OMR Scanner App for Lecturers. This phase translates the elicited requirements into detailed representations of the system's functionality and behaviour. To achieve this, several diagrams were developed to model the system's features and interactions, ensuring a clear understanding of user needs and technical feasibility.

3.4.1 Use Case Diagram

The use case diagram demonstrated the interactions between the system and its primary user, the lecturer.

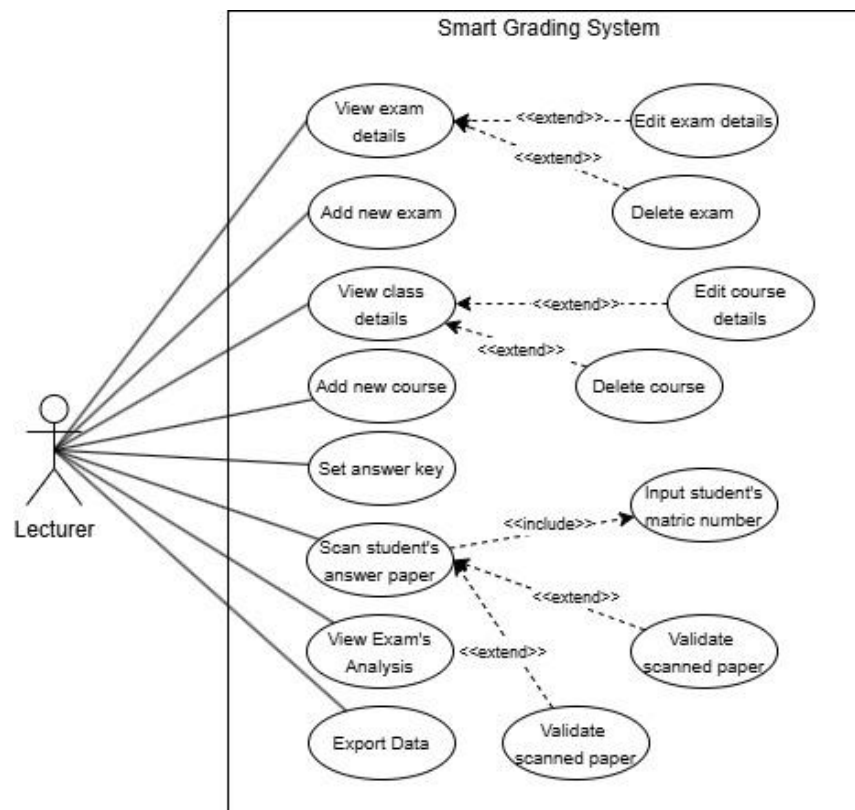


Figure 3.2: Smart Grading Use Case Diagram

Use Case Specification

Table 3.1: Use Case for View Exam Details

Use Case Name	View Exam Details
Actor	Lecturer
Description Summary	The lecturer views details about a selected exam.
Pre-Condition	The exam exists in the system.
Post Condition	The system displays detailed information about the exam.
Basic Path	1. Navigate to the "Exam" tab. 2. Select an exam from the list. 3. View details such as name, number of questions, options and if there is course link to the exam.
Alternative Path	If the exam does not exist, the system displays an error: "Exam not found."

Table 3.2: Use Case for Edit Exam Details

Use Case Name	Edit Exam Details
Actor	Lecturer
Description Summary	The lecturer updates details for a specific exam.
Pre-Condition	The exam exists in the system.
Post Condition	The updated exam details are saved.
Basic Path	1. Navigate to "Exams" tab. 2. Select an exam. 3. Tap edit icon. 4. Edit details like name, number of questions, options and course 4. Save changes.
Alternative Path	If the save operation fails, an error message is displayed: "Failed to save changes."

Table 3.3: Use Case for Delete Exam

Use Case Name	Delete Exam
Actor	Lecturer
Description Summary	The lecturer removes a specific exam.
Pre-Condition	The exam exists in the system.
Post Condition	The exam is deleted from the system.
Basic Path	1. Navigate to "Exams" tab. 2. Select an exam. 3. Tap delete icon. 4. Confirm deletion. 5. The system removes the exam.
Alternative Path	If the delete fails, the system displays: "Failed to delete exam."

Table 3.4: Use Case for Add New Exam

Use Case Name	Add New Exam
Actor	Lecturer
Description Summary	The lecturer creates a new exam by entering required details such as the name, number of questions, and options. The exam may optionally be associated with a course.
Pre-Condition	The system is operational.
Post Condition	A new exam is created and saved in the system.
Basic Path	1. Navigate to the "Exams" tab. 2. Tap "+ New Exam" 3. Enter exam details, such as name, number of questions, and options. 4. (Optional) Select an associated course from the list. 5. Save the exam. 6. The system confirms the creation of the new exam.
Alternative Path	If required fields are missing, the system displays an error: "Please complete all required fields."

Table 3.5: Use Case for View Course Details

Use Case Name	View Course Details
Actor	Lecturer
Description Summary	The lecturer views the details of a specific course.
Pre-Condition	The course exists in the system.
Post Condition	Course details are displayed and associated exams are listed.
Basic Path	1. Navigate to "Courses" tab. 2. Select a course. 3. View the course name and associated exams.
Alternative Path	If no courses exist, the system displays: "No courses available. Please create a new course"

Table 3.6: Use Case for Edit Course

Use Case Name	Edit Course
Actor	Lecturer
Description Summary	The lecturer updates the course details.
Pre-Condition	The course exists in the system.
Post Condition	The updated course details are saved.
Basic Path	1. Navigate to "Courses" tab. 2. Select a course. 3. Edit the course name. 4. Edit the short name 5. Save changes.
Alternative Path	If the course name or short name is empty or invalid, the system displays an error: "Course name and short name must be valid and non-empty."

Table 3.7: Use Case for Delete Course

Use Case Name	Delete Course
Actor	Lecturer
Description Summary	The lecturer deletes a course.
Pre-Condition	The course exists in the system.
Post Condition	The selected course is deleted from the system, and all associated exams are reassigned to "No Course."
Basic Path	1. Navigate to "Courses" tab. 2. Select a course. 3. Tap the delete icon. 4. Confirm deletion.
Alternative Path	If the course cannot be deleted, the system displays: "Failed to delete course."

Table 3.8: Use Case for Add New Course

Use Case Name	Add New Course
Actor	Lecturer
Description Summary	The lecturer creates a new course by entering the course name.
Pre-Condition	The system is operational.
Post Condition	The new course is added to the system.
Basic Path	1. Navigate to the "Courses" tab. 2. Tap "Add New Course." 3. Enter the course name. 4. Enter the short name. 5. Save the course. 6. The system confirms the creation of the new course.
Alternative Path	If the course name or short name is empty, the system displays an error: "Course name and short name cannot be empty."

Table 3.9: Use Case for Set Answer Key

Use Case Name	Set Answer Key
Actor	Lecturer
Description Summary	The lecturer sets the correct answers for an exam. This can be done by manually entering answers or scanning a pre-completed answer paper.
Pre-Condition	The exam exists in the system.
Post Condition	The answer key is saved in the system.
Basic Path	1. Select an exam. 2. Navigate to the "Answer Key" option. 3. Choose to either manually input answers or scan a completed answer paper. 4. Save the answer key.

Table 3.9 continued

Alternative Path	If the answer key is incomplete or the scanned paper is invalid, the system displays: "Please complete or retry the answer key process."
------------------	--

Table 3.10: Use Case for Scan Student's Answer Paper

Use Case Name	Scan Student's Answer Paper
Actor	Lecturer
Description Summary	The lecturer scans a student's answer paper to automatically grade it based on the predefined answer key.
Pre-Condition	The answer key for the selected exam has been set.
Post Condition	The student's graded results are saved in the system.
Basic Path	1. Navigate to the "Exams" tab. 2. Select an exam. 3. Tap "Scan Student's Answer Paper." 4. Use the camera or upload picture to scan the paper. 5. Enter the student's matric number. 6. The app validates and grades the answers based on the answer key. 7. The results are displayed and saved.
Alternative Path	If the scanned paper is unclear, the system displays: "Scan failed. Please retry."

Table 3.11: Use Case for Validate Scanned Answer Papers

Use Case Name	Validate Scanned Answer Papers
Actor	Lecturer
Description Summary	The lecturer reviews and corrects errors in scanned answer papers.
Pre-Condition	Scanned answer papers exist for the selected exam.
Post Condition	Corrected answers are saved.
Basic Path	1. Tap the edit icon in the selected answer paper page. 2. Review and correct errors. 3. Save the validated results.
Alternative Path	If no changes are made or the lecturer cancels the editing, the system discards any unsaved modifications and returns to the answer paper view without updating the data.

Table 3.12: Delete Scanned Answer Papers

Use Case Name	Delete Scanned Answer Papers
Actor	Lecturer
Description Summary	The lecturer deletes specific scanned answer papers.
Pre-Condition	Scanned papers exist in the system.
Post Condition	The selected papers are deleted.

Table 3.12 continued

Basic Path	1. Select the paper to delete. 2. Tap the delete icon. 3. Confirm deletion. 4. The system removes the paper.
Alternative Path	If the lecturer attempts to proceed without selecting any papers, the system prompts: "Please select at least one paper to delete."

Table 3.13: Use Case for View Exam's Analysis

Use Case Name	View Exam's Analysis
Actor	Lecturer
Description Summary	The lecturer views performance analysis, including graphs and question statistics.
Pre-Condition	At least one answered paper has been scanned.
Post Condition	Analysis data is displayed.
Basic Path	1. Select an exam. 2. Tap "Analysis." 3. The system displays a graph of marks and a question performance table.
Alternative Path	If no student answer paper has been scanned, the system displays: "No data available for analysis."

Table 3.14: Use Case for Export Data

Use Case Name	Export Data
Actor	Lecturer
Description Summary	The lecturer exports exam results to a CSV file.
Pre-Condition	At least one answered paper has been scanned.
Post Condition	A CSV file containing the results is generated and made available for download.
Basic Path	1. Select an exam. 2. Tap "Export Data" 3. The system compiles the results. 4. The CSV file is ready for download.
Alternative Path	If no results exist, the system displays: "No results available to export."

3.4.2 Sequence Diagram

The sequence diagrams for the Smart Grading represent the interaction flows between the system and the lecturer for various functionalities. To reduce complexity and

improve clarity, the diagrams are categorised into five main use cases: Manage Exam, Manage Course, Set Answer Key, Scan Answer Papers, and Analysis and Export Data.

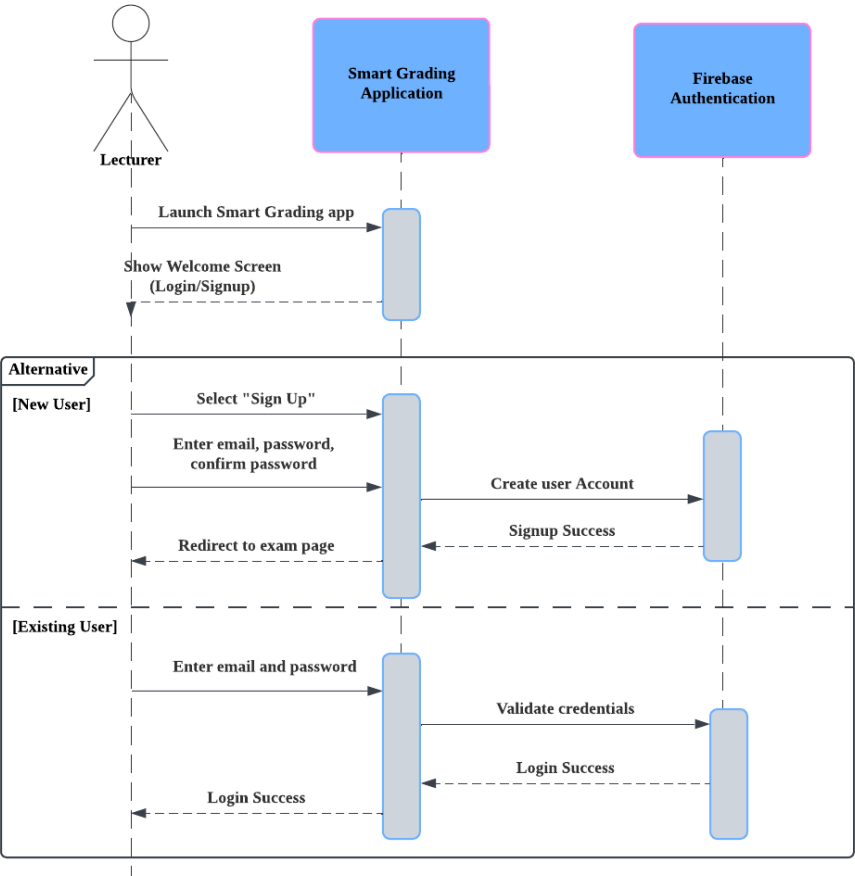


Figure 3.3: Sequence Diagram for Lecturer Authentication

Figure 3.3 illustrates the authentication process for lecturers in the Smart Grading system. The flow begins when the lecturer launches the app and is presented with a welcome screen offering login and signup options. For new users, the lecturer selects "Sign Up", enters the required credentials (email, password, and confirm password), and the system communicates with Firebase Authentication to create a new user account. Upon successful registration, the lecturer is redirected to the exam dashboard. For existing users, the lecturer simply enters their email and password, which the system verifies through Firebase. If the

credentials are valid, the lecturer is granted access to the system. This diagram demonstrates how Firebase Authentication is seamlessly integrated into the mobile app to manage user identity and access, ensuring a secure and user-friendly onboarding experience for lecturers.

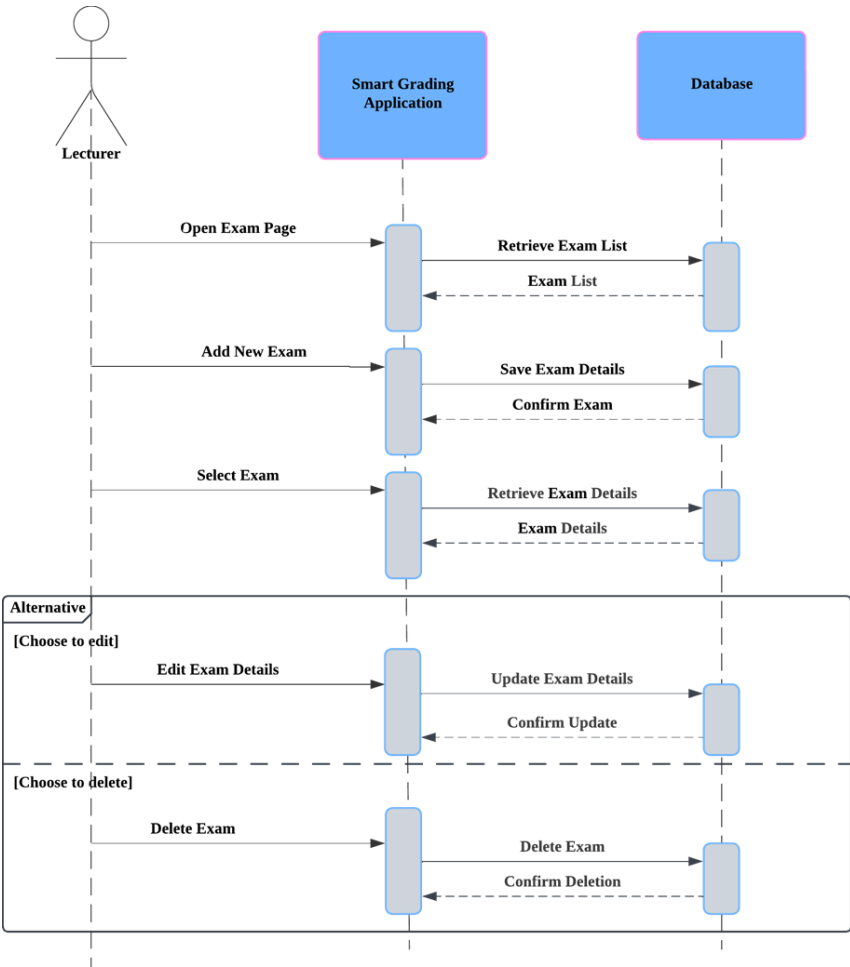


Figure 3.4: Sequence Diagram for Manage Exam

This sequence diagram, figure 3.4 demonstrates how the lecturer interacts with the Smart Grading Application to manage exams. It shows the processes for retrieving a list of exams, adding a new exam, viewing specific exam details, editing exam information, and deleting exams. The interactions between the application and the database ensure seamless

data retrieval, updates, and deletions. Alternative flows, such as confirming updates or deletions, ensure that user actions are intentional and recorded properly.

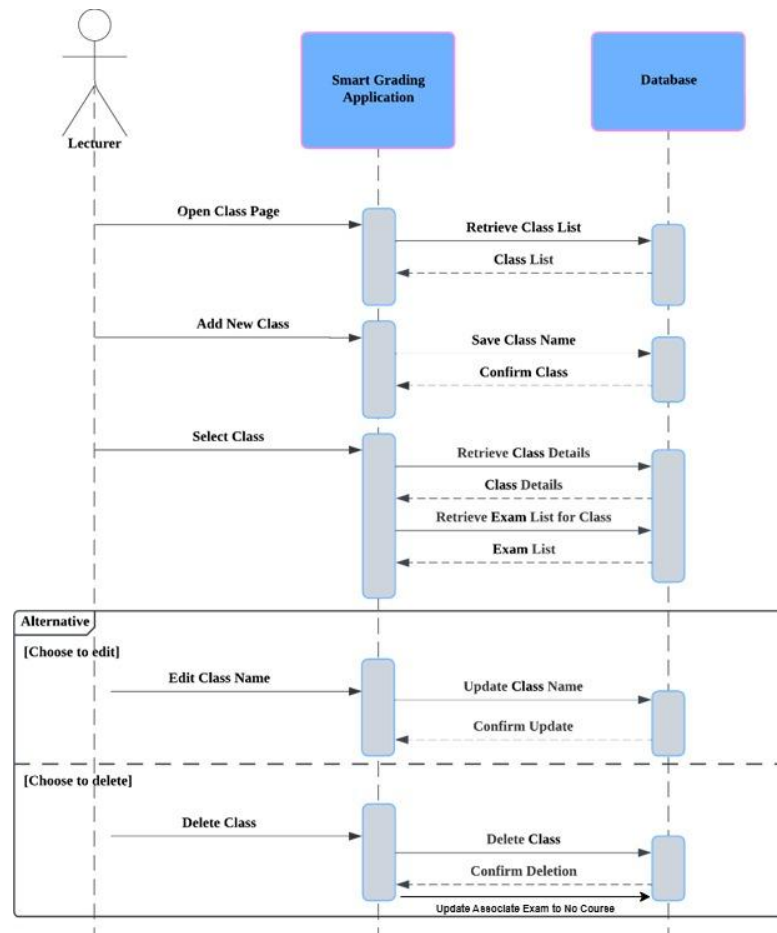


Figure 3.5: Sequence Diagram for Manage Course

The sequence diagram, figure 3.5 for managing courses illustrates how the lecturer handles course-related operations within the Smart Grading Application. It includes retrieving a list of courses, adding a new course, viewing course details, editing course names, and deleting courses. The diagram highlights interactions with the database to fetch, update, and remove course data. Alternative flows ensure that the system confirms critical actions, such as updates or deletions, maintaining data integrity.

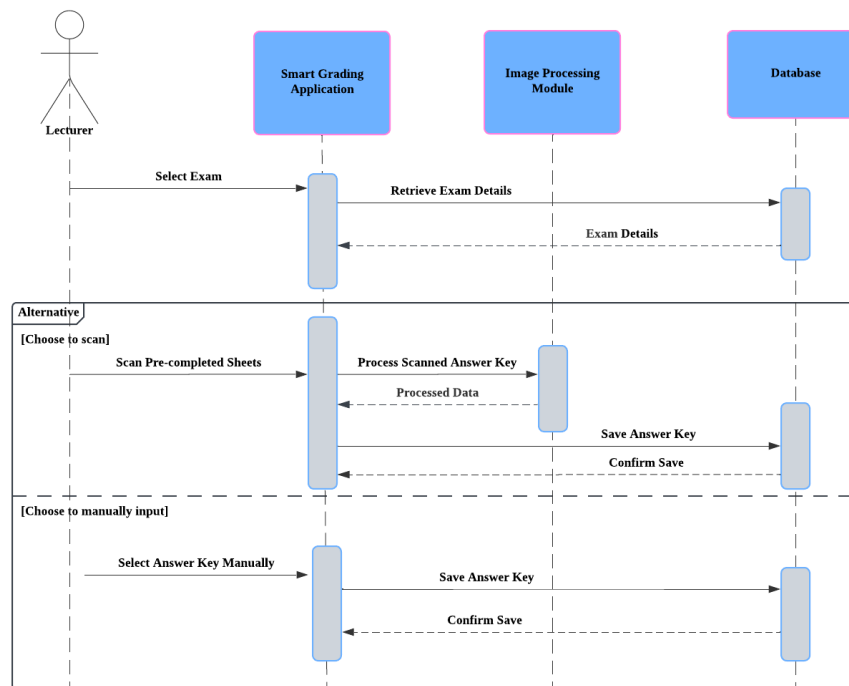


Figure 3.6: Sequence Diagram for Set Answer Key

This sequence diagram, figure 3.6 details the process of setting an answer key for an exam, either manually or by scanning a pre-filled answer paper. The lecturer initiates the process by selecting an exam, and the system provides options for inputting the answer key. If scanning is chosen, the Image Processing Module validates the scanned data before saving it in the database. Error-handling mechanisms ensure the lecturer is notified of invalid scans, allowing them to retry or manually input the data.

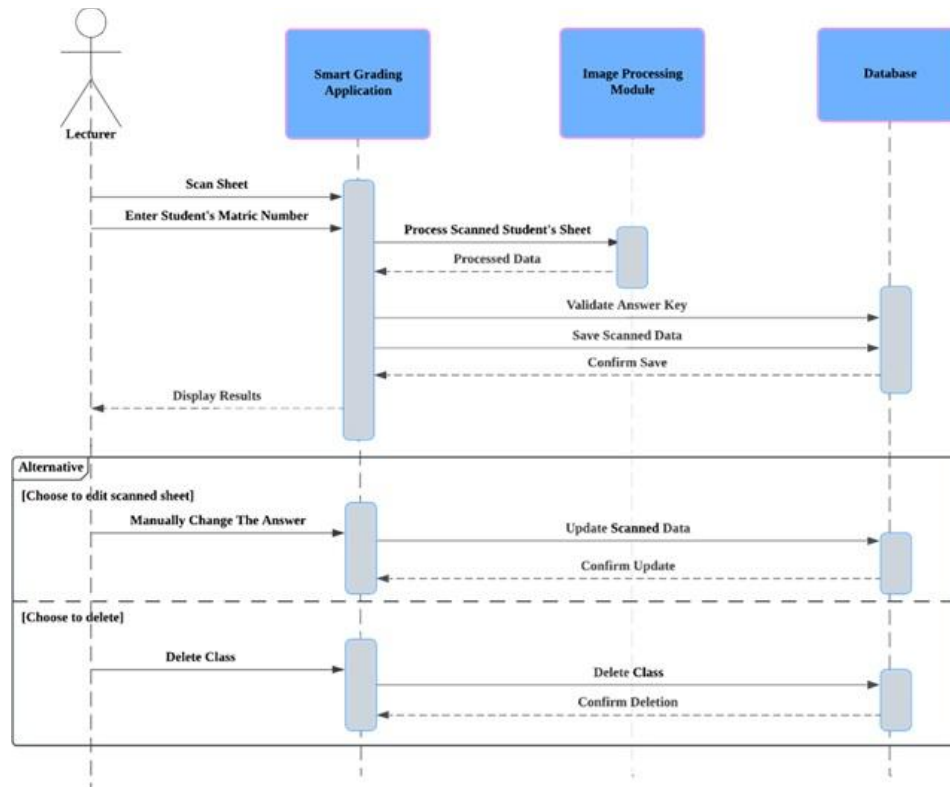


Figure 3.7: Sequence Diagram for Scan Answer Papers

The scan answer papers sequence diagram explains the process of grading student answer papers as shown in the figure 3.7. The lecturer scans the answer papers, and the Image Processing Module validates the scanned data against the predefined answer key, calculates scores, and saves the results in the database. The lecturer can view the results and repeat the scanning process if needed. Alternative flows include manual corrections of errors or deletion of invalid scans, ensuring flexibility and accuracy.

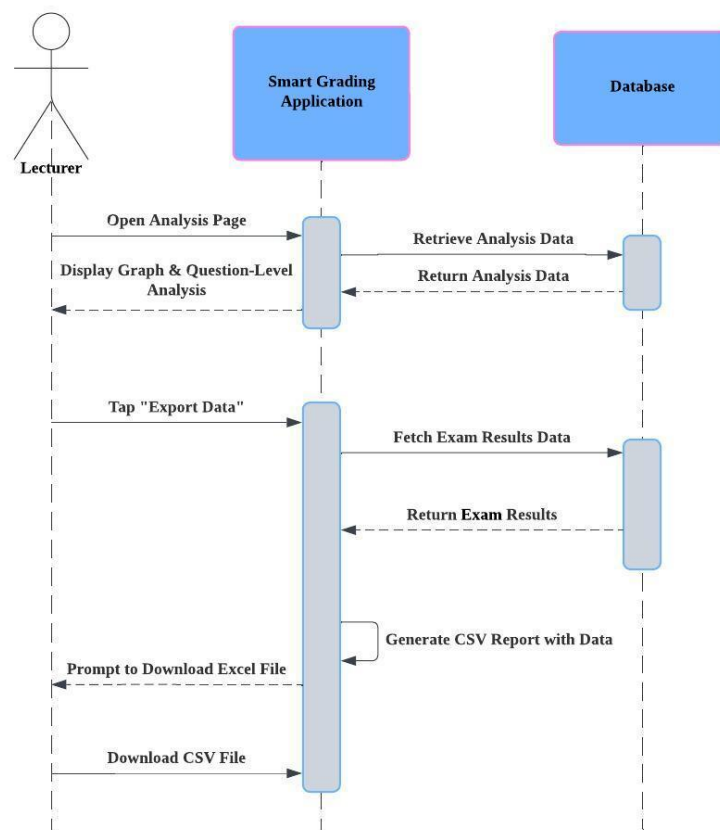


Figure 3.8: Sequence Diagram for Analysis and Export Data

Figure 3.8 illustrates the sequence of actions involved when a lecturer analyses exam results and exports them. The lecturer begins by accessing the analysis page, prompting the application to retrieve and display performance data such as score distribution and question-level insights from the database. When the lecturer taps the "Export Data" option, the system compiles the grading results and directly generates a CSV file for download. If no results are available or if the generation process encounters an issue, appropriate error messages are shown to inform the lecturer.

3.4.3 Class Diagram

The Class Diagram is a static structure diagram that represents the relationships, attributes, and operations of the classes in the Smart Grading: A Mobile OMR Scanner App for Lecturers system.

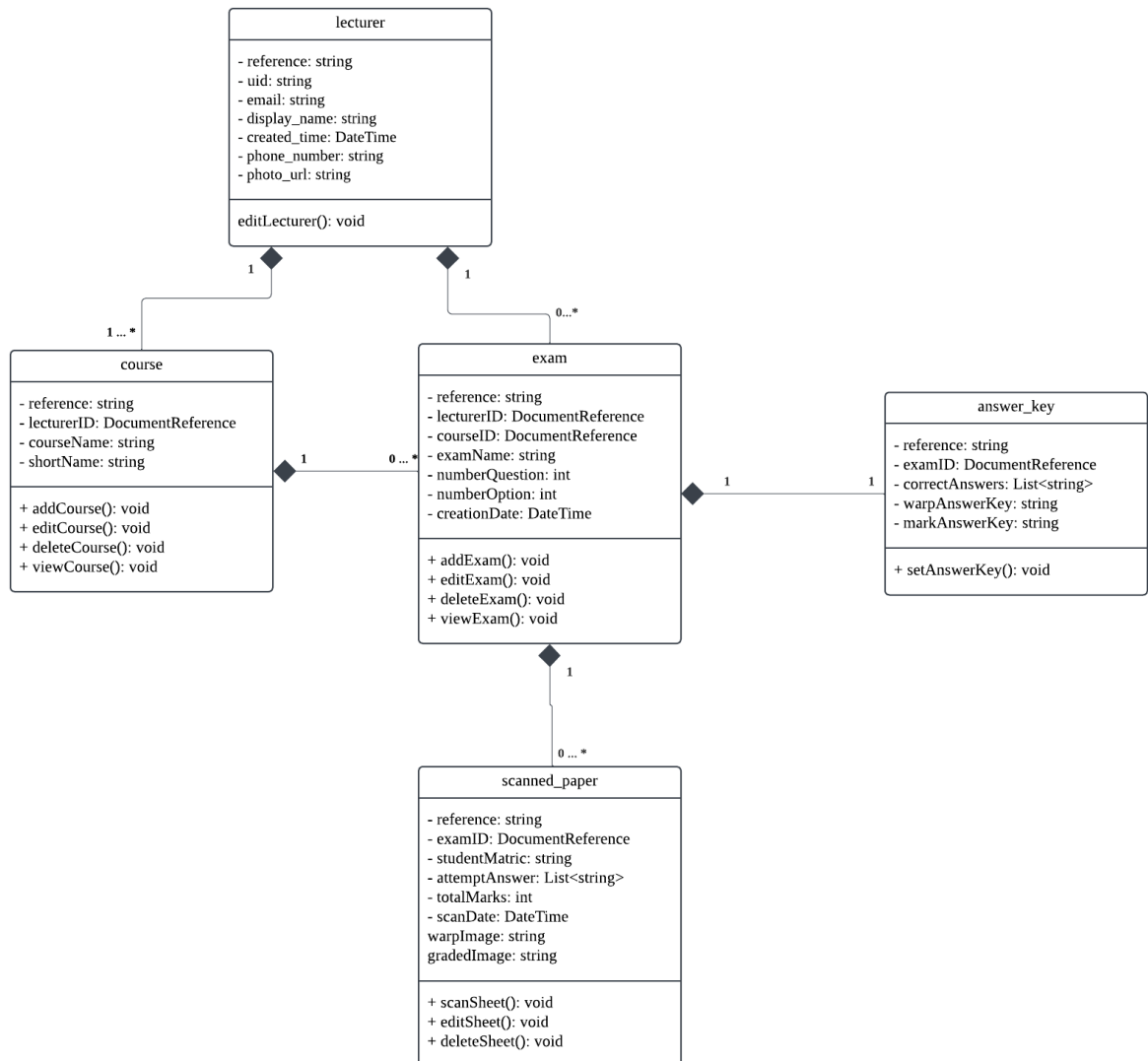


Figure 3.9: Smart Grading Class Diagram

The class diagram in Figure 3.9 presents the updated architecture of the Smart Grading System, reflecting its modular components and their tightly coupled relationships.

At the core of the system is the lecturer class, which owns both the course and exam classes through composition, ensuring that these entities cannot exist independently without a lecturer. Each course may include multiple exam documents, representing a one-to-many relationship. The exam class plays a central role by linking to both `answer_key` and `scanned_paper`. The `answer_key` class maintains a list of correct answers and is directly and exclusively bound to a single exam, supporting consistent grading. Meanwhile, the `scanned_paper` class stores student submissions, including their matric number, detected answers, total marks, and processed images, all associated with a particular exam.

All relationships are designed with data integrity and Firebase's document-reference structure in mind. This design supports scalable grading operations, secure data separation by lecturer ownership, and a clear representation of ownership between classes especially via strong composition where applicable.

3.5 System Design

System design focuses on translating the requirements and analysis into a structured architecture that ensures the system meets its intended functionality, performance, and usability. This chapter outlines the high-level and detailed designs that form the foundation of the Smart Grading Application. Each section presents visual representations and descriptions of how the components interact and function within the system.

3.5.1 Image Processing Approach

The Smart Grading application implements a structured image processing workflow to accurately detect and evaluate Optical Mark Recognition (OMR) answer papers. By leveraging OpenCV, the system processes captured images and extracts marked responses, ensuring efficient and reliable grading.

The process begins with image acquisition, where lecturers capture the OMR answer paper using their mobile device's camera. To improve accuracy, alignment guides are provided on-screen to assist in capturing a properly positioned image. Once captured, the image undergoes preprocessing, where it is converted to grayscale to simplify analysis while preserving key features. Following grayscale conversion, edge detection is applied using the Canny Edge Detection method, which highlights the boundaries of the answer paper and its components. The system then extracts contours from the image, identifying distinct regions that could correspond to answer bubbles. To ensure that only the relevant sections are processed, the algorithm detects and isolates the answer rectangle, which contains the student's responses.

Once the answer area is identified, the system applies perspective transformation to achieve a bird's-eye view of the paper, correcting distortions caused by camera angles. This step ensures that the detected marks are accurately aligned for processing. A thresholding technique is then applied to convert the image into a binary format, where marked answers appear distinctly as black regions while unmarked areas remain white. With the thresholded image, the system proceeds to mark detection by analysing pixel intensities within the designated answer regions. The presence of a filled-in bubble is determined by comparing its darkness against a predefined threshold. If multiple bubbles are marked in a single question, the system flags the response for manual review.

After extracting the student's answers, the system compares them against the predefined answer key. The total score is then calculated based on correct responses, and the results are stored in the system. For enhanced accuracy, additional validation checks are performed to detect potential errors, such as faintly erased marks or unclear markings. To improve user experience, the system incorporates error handling mechanisms. If the image

is blurry or poorly aligned, the application prompts the lecturer to retake the scan. In cases where the system detects incomplete or ambiguous responses, it allows for manual verification and correction before finalising the grading.

By integrating these image processing techniques, the Smart Grading application ensures efficient, accurate, and automated grading of OMR answer papers, reducing human effort while maintaining high reliability.

3.5.2 Activity Diagram

The activity diagrams illustrate the workflows for key functionalities in the **Smart Grading System**, providing a clear visual representation of processes and decision points. To reduce complexity and improve understanding, the workflows have been divided into five detailed diagrams, each focusing on a specific functionality. This modular approach ensures that each diagram is easier to interpret while maintaining clarity in the system's overall behaviour.

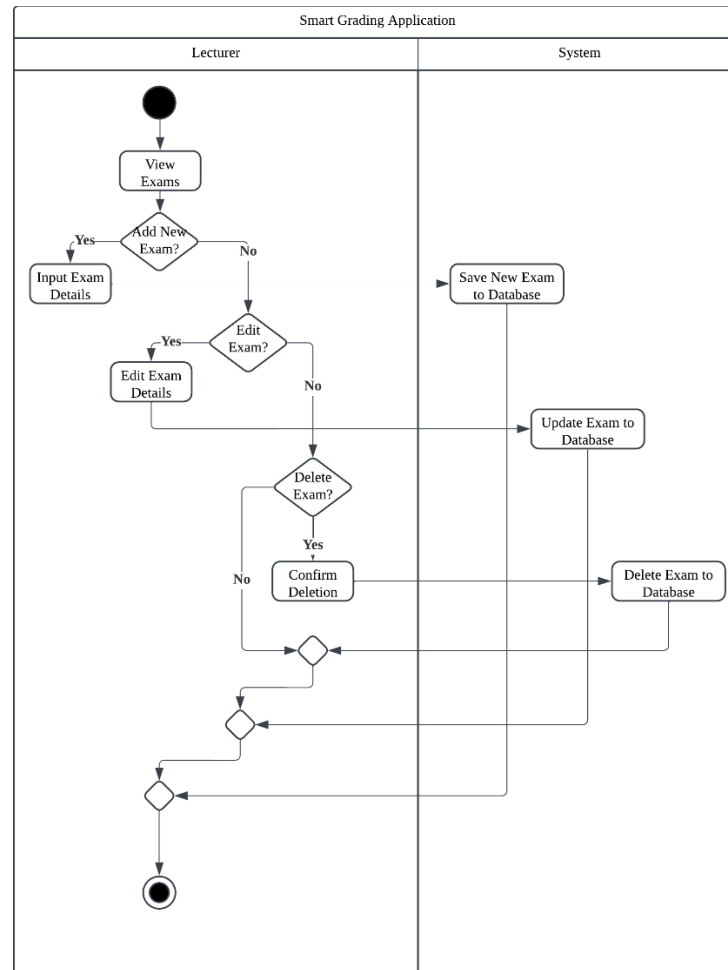


Figure 3.10: Activity Diagram for Exam Management

The activity diagram for exam management outlines the workflow for handling exams in the Smart Grading System as shown in the figure 3.10. It showcases the lecturer's ability to view a list of exams, add new exams by inputting details, edit existing exam information, and delete exams after confirming the action. The system ensures data integrity by saving new or updated exam data and securely deleting exams from the database. Decision points like "Add New Exam?" and "Delete Exam?" ensure that the workflow is flexible and responsive to the lecturer's actions. The diagram separates manual tasks performed by the lecturer from automated system actions for clarity.

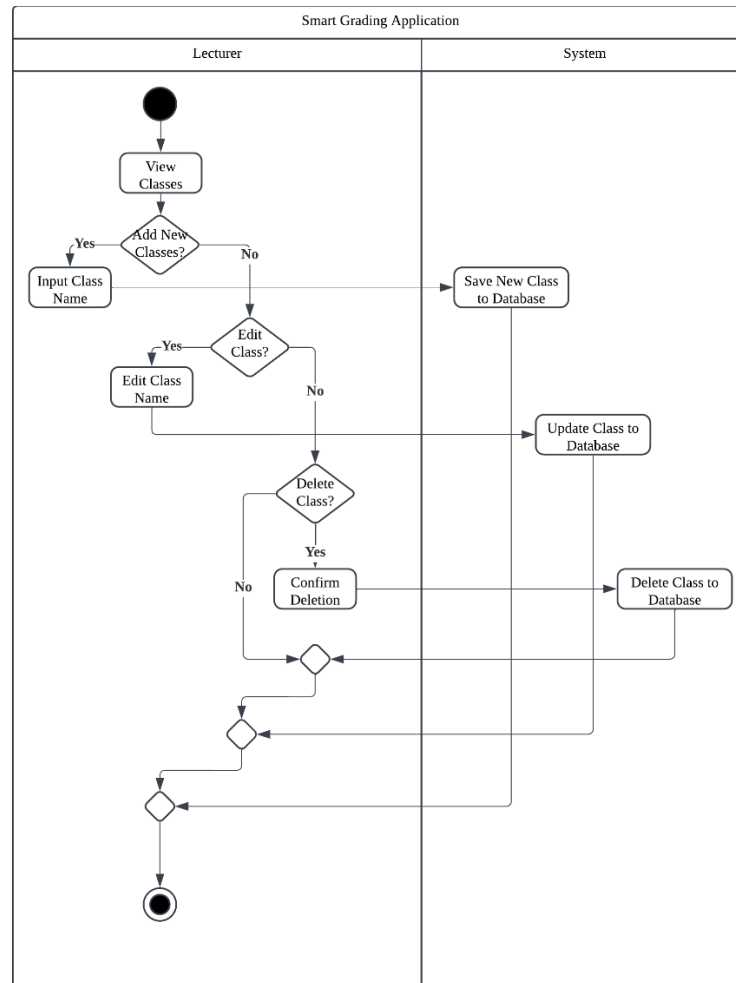


Figure 3.11: Activity Diagram for Course Management

This diagram depicts the workflow for managing courses in the system as shown in the figure 3.11. It highlights the lecturer's role in viewing courses, adding a new course by entering the course name, editing existing course names, and deleting courses after confirmation. The system performs back-end operations such as saving, updating, and deleting course data in the database. Decision points like "Add New Course?" and "Delete Course?" guide the flow of operations, ensuring accurate and efficient data management. The clear distinction between the lecturer's actions and the system's automated processes ensures an intuitive and streamlined workflow.

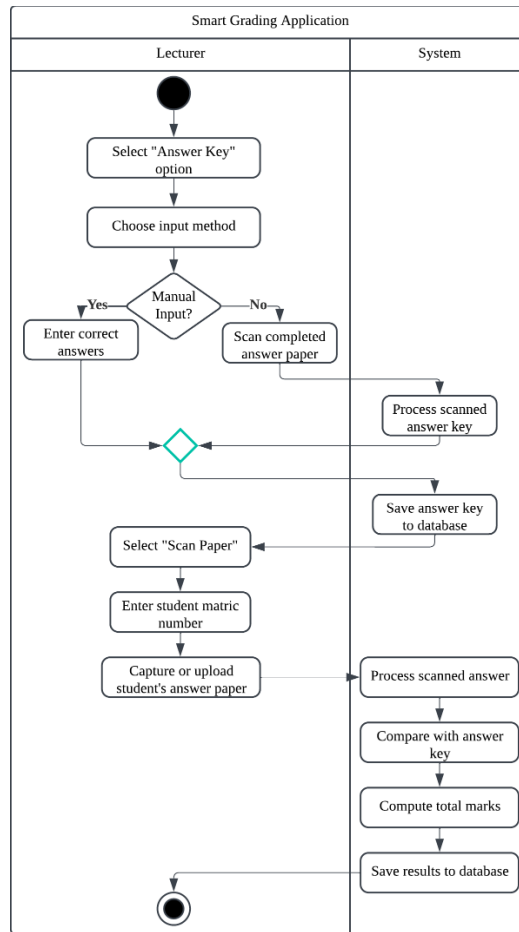


Figure 3.12: Activity Diagram for Grading Student's Answer Paper

Figure 3.12 illustrates the grading workflow involved in scanning and assessing student answer papers. The process begins with the lecturer inputting the answer key either manually or by scanning a pre-filled sheet. Next, the lecturer scans the student's answer paper and enters the student's matric number. The system then processes the scanned image, compares the extracted answers against the predefined answer key, calculates the total score, and stores the results in the database. This workflow highlights the seamless interaction between the lecturer's input and the system's automated grading, ensuring efficiency, accuracy, and reduced manual workload.

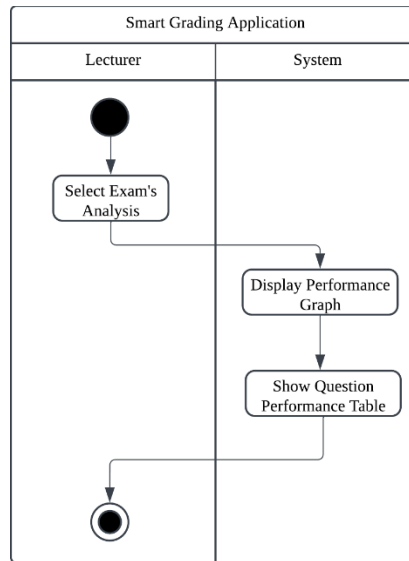


Figure 3.13: Activity Diagram for Exam's Analysis

This diagram illustrates the workflow for viewing the analysis of an exam's performance as shown in the figure 3.13. The lecturer selects the "Exam's Analysis" option, and the system retrieves data to generate a performance graph and a detailed question-level performance table. The graph visually represents student marks distribution, while the table provides insights into how students performed on individual questions. The separation of lecturer actions and system processes ensures a clear representation of the steps involved in accessing and displaying meaningful analysis data.

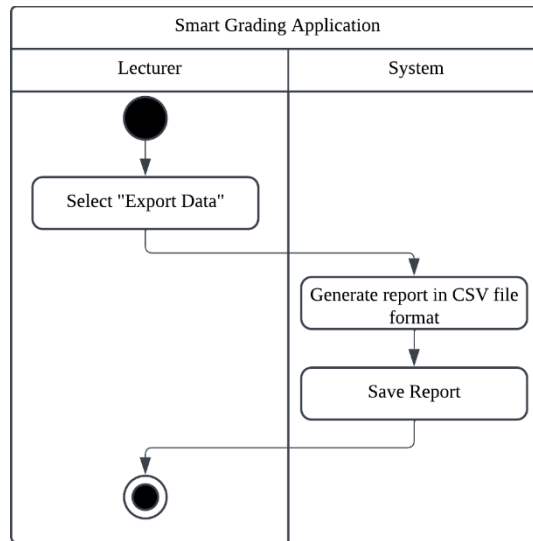


Figure 3.14: Activity Diagram for Exporting Data

The exporting workflow diagram highlights the process of generating and saving an CSV report for exam results as shown in the figure 3.14. The lecturer initiates the export process by selecting the "Export Data" option. The system gathers the relevant data, processes it to generate an CSV file, and saves the report for download. This diagram showcases the system's ability to automate report generation, reducing manual effort while ensuring that data is presented in an organised and accessible format for offline use.

3.5.3 System User Interface Design

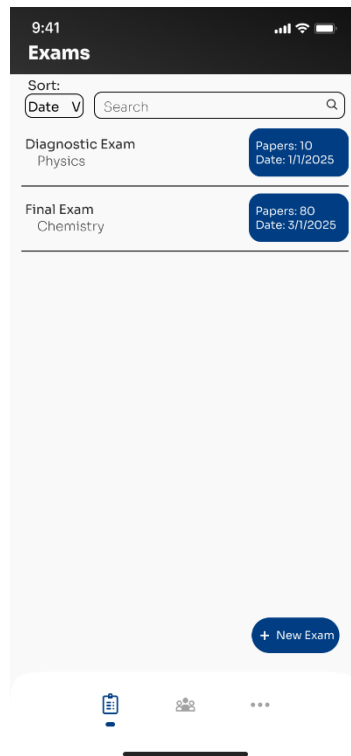


Figure 3.15: Landing Page / Exam List

Figure 3.15 shows a page that serves as the main landing page for the application and displays a list of all exams created by the lecturer. From here, users can view exam details, add new exams, edit existing ones, or delete exams.

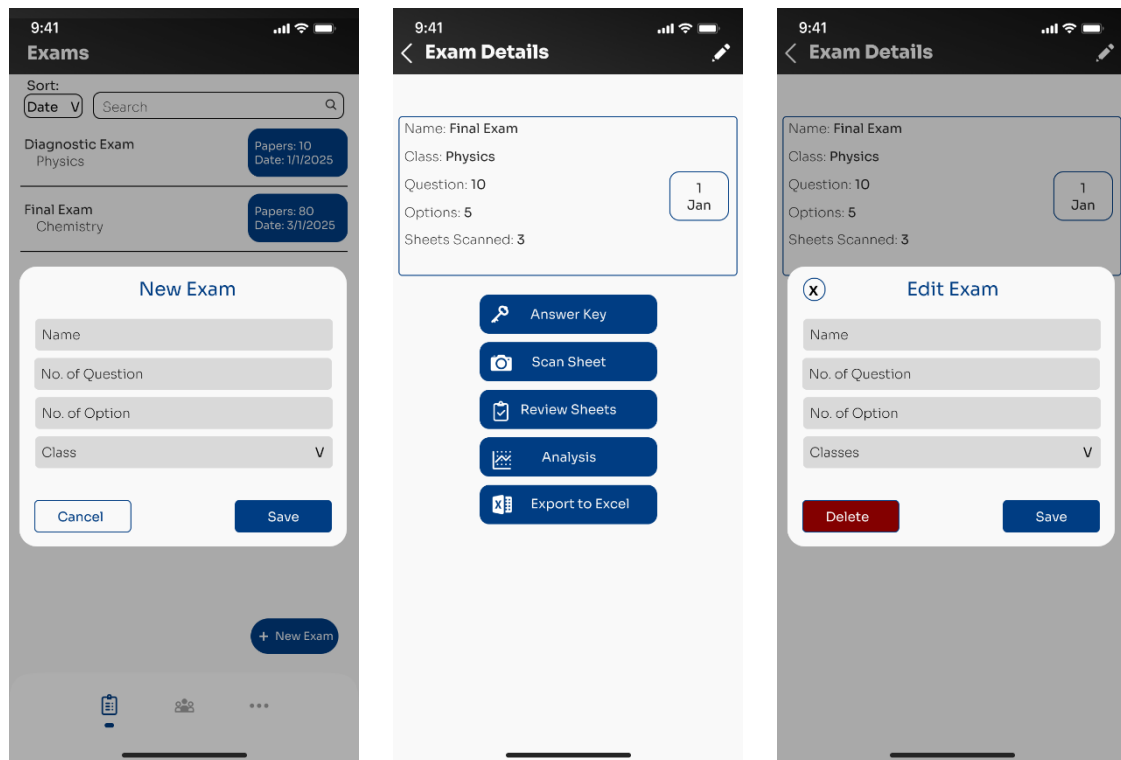


Figure 3.16: Exam Management

This figure 3.16 illustrates the core functionalities for managing exams. The first image shows the pop-up for creating a new exam, where lecturers can input exam details. The second image displays the exam details page, providing information about the exam and five action buttons for further operations. The third image shows the pop-up for editing exam details, allowing lecturers to update or delete the exam.

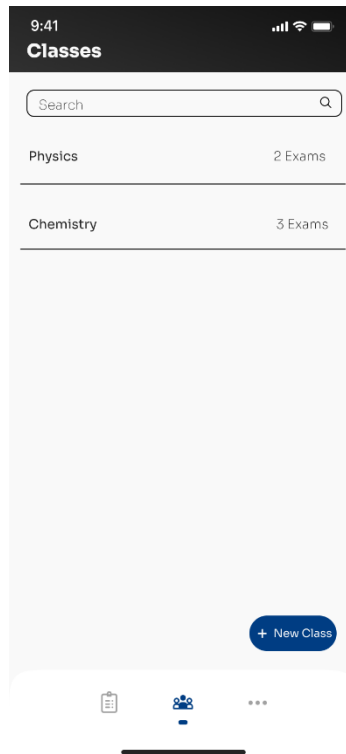


Figure 3.17: Course List

This figure 3.17 shows the interface for managing courses. When the user taps on the "Courses" tab, they are presented with a list of courses created by the lecturer. Each course entry provides an overview and allows users to select a course for further actions, such as viewing its details or managing associated exams.

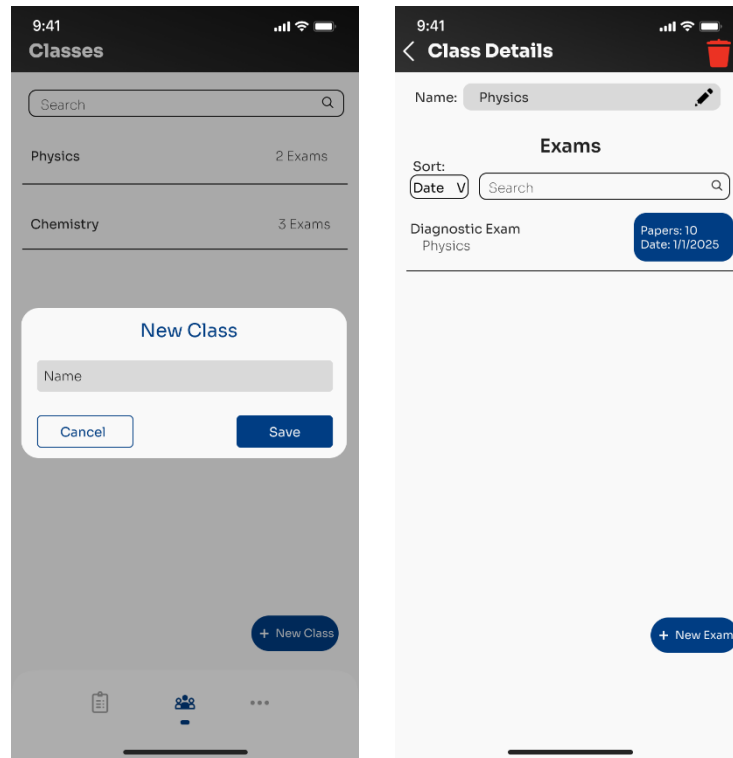


Figure 3.18: Course Management

This figure 3.18 demonstrates the course management workflow in the system. The first image shows the pop-up interface for adding a new course, where the lecturer inputs the course name. The second image displays the course details page, which lists exams associated with the selected course. In this view, the lecturer can edit the course name, delete the course, or create a new exam. Additionally, tapping on an exam redirects the lecturer to the exam details page for further actions.

9:41
< Answer Key

Name: Final Exam

SCAN FOR KEY

1:	A	B	C	D	E
2:	A	B	C	D	E
3:	A	B	C	D	E
4:	A	B	C	D	E
5:	A	B	C	D	E
6:	A	B	C	D	E
7:	A	B	C	D	E
8:	A	B	C	D	E
9:	A	B	C	D	E
10:	A	B	C	D	E

Reset Save

Figure 3.19: Answer Key Page

This figure 3.19 illustrates the interface for setting the answer key in the system. It shows a list of answers corresponding to the number of questions in the selected exam. The lecturer can manually set the answer key by selecting options for each question. Alternatively, if a pre-completed answer paper is available, the lecturer can scan it, and the system will automatically select the answer key based on the scanned data. Once the answer key is set, the lecturer can save the changes.

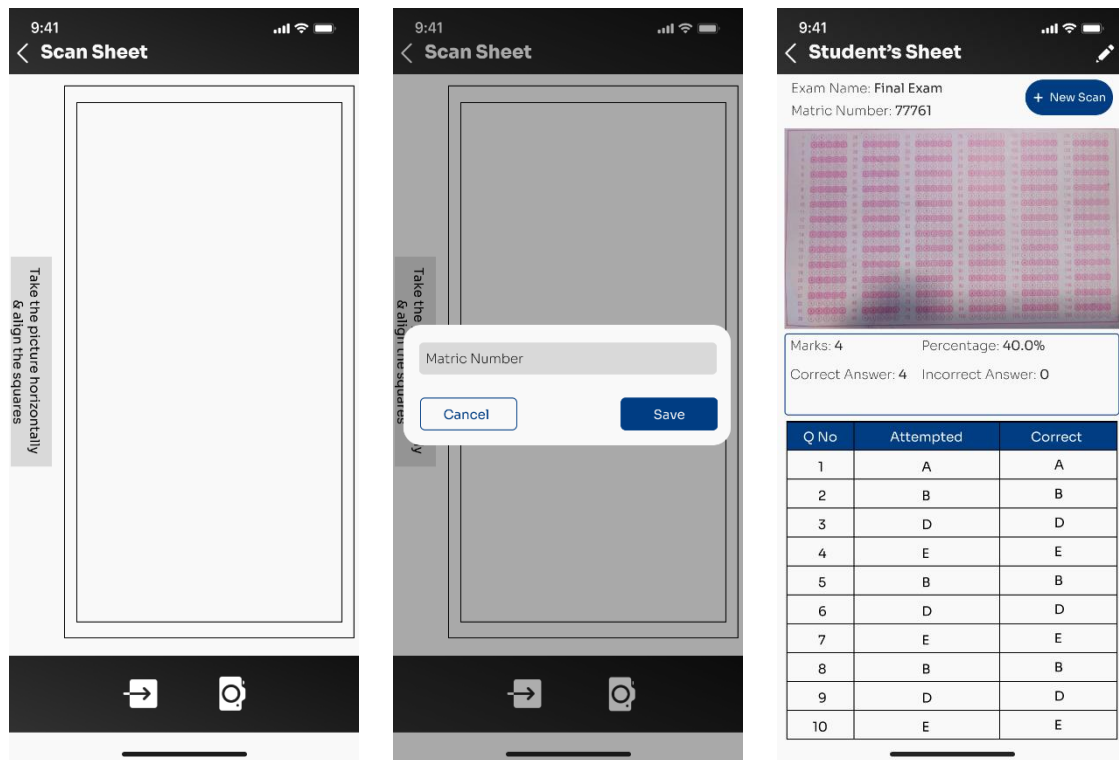


Figure 3.20: Scanning Student Answer Paper

This figure 3.20 illustrates the process of scanning and managing student answer papers. The first image shows the interface for scanning, where the lecturer must align the paper horizontally with on-screen guidelines or upload a photo of the answer paper. The second image displays a popup prompting the lecturer to input the student's matric number after a successful scan. The third image shows the detailed results on the student's paper page, including exam details, marks, percentage, correct and incorrect answers, and a table of question numbers with attempted and correct answers. The lecturer can edit the scanned answers if corrections are needed.

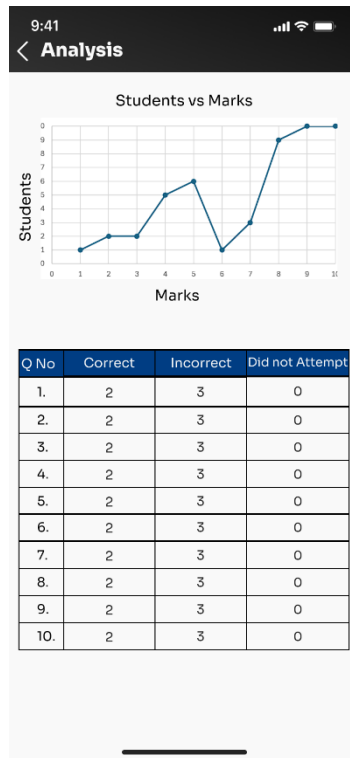


Figure 3.21: Exam Analysis

This figure 3.21 shows the exam analysis interface, where the system provides insights into student performance. The graph at the top visualises the distribution of students' marks, offering an overview of overall performance. Below the graph is a detailed table that displays the number of students who answered each question correctly, incorrectly, or did not attempt it, providing a question-by-question breakdown for deeper analysis.

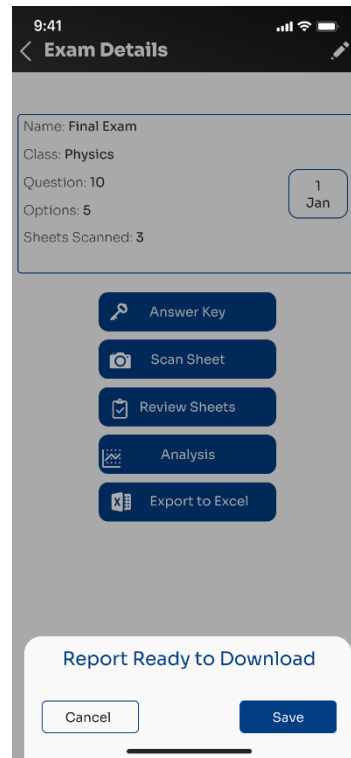


Figure 3.22: Export to Excel

This figure 3.22 illustrates the process of exporting exam results to an Excel file. When the lecturer taps on the "Export to Excel" option, the system generates the Excel file containing detailed exam results and analysis. Once the file is ready, a popup appears prompting the lecturer to either download the file or cancel the operation. This ensures an easy and efficient way to save and access exam data in a shareable format.

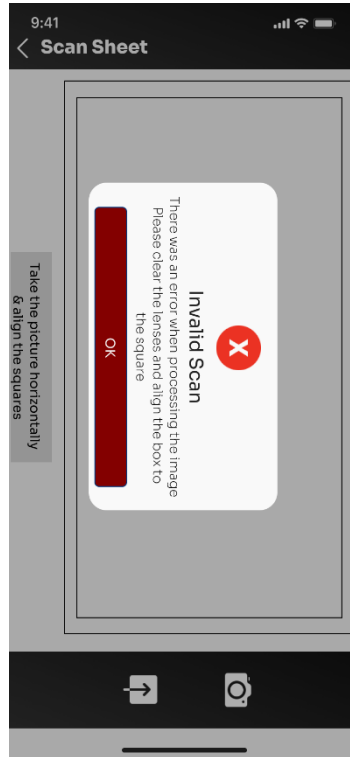


Figure 3.23: Error Handling

This figure 3.23 demonstrates the error handling mechanism in the system. When an invalid scan occurs, a popup appears notifying the lecturer of the issue. The message provides details about the error and offers instructions to resolve it, such as ensuring proper alignment of the paper or improving image clarity. This feature enhances user experience by guiding lecturers to correct errors and successfully complete the scanning process.

3.5.4 System Database Design

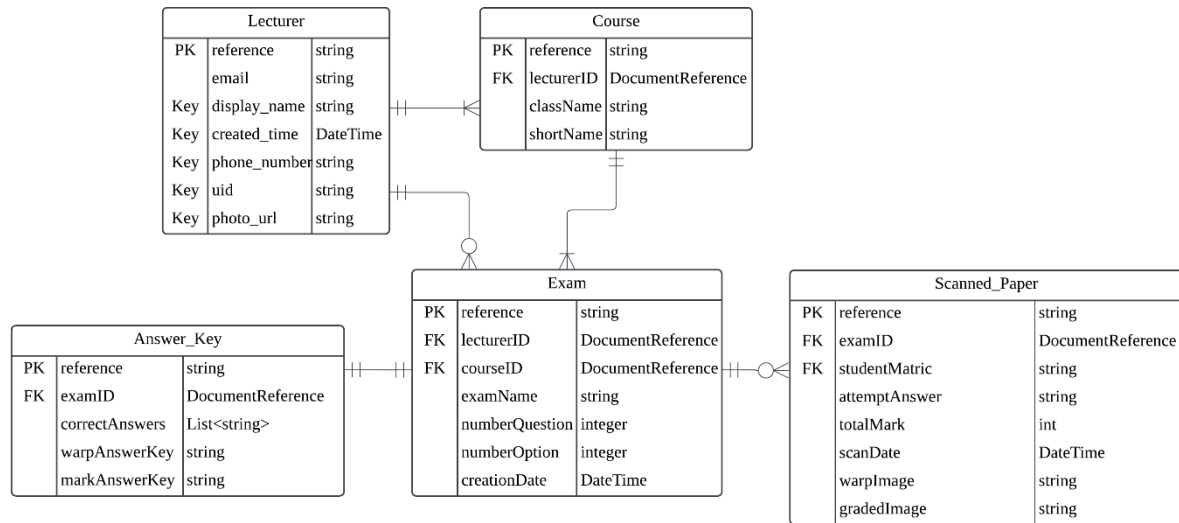


Figure 3.24: Entity Relationship Diagram

The Entity-Relationship Diagram (ERD) in Figure 3.24 illustrates the database structure of the Smart Grading System, designed to align with Firebase’s document-based model. Each entity represents a Firestore collection, while inter-entity connections are implemented through DocumentReference fields to reflect logical relationships.

The Lecturer entity stores vital user information such as email, UID, and contact details. A lecturer must have at least one course, forming a 1 to 1..N relationship with the Course entity. In addition, a lecturer may manage zero or more exams, establishing a 1 to 0..N link to the Exam entity. Each Course contains metadata like courseName and shortName and is uniquely linked to one lecturer via lecturerID. A course must contain at least one exam, creating a 1 to 1..N relationship with the Exam entity. The Exam entity includes attributes such as examName, numberQuestion, numberOption, and creationDate, and references both the lecturerID and courseID. Every exam is required to have exactly one answer key, forming a 1 to 1 relationship with the Answer_Key entity. This entity also stores

grading aids like correctAnswers, warpAnswerKey, and markAnswerKey. Student responses are managed in the Scanned_Paper entity, each linked to one exam, and an exam may have zero or many scanned papers a 1 to 0..N relationship. Each record captures studentMatric, attemptAnswer, totalMark, scanDate, and associated image paths (warpImage, gradedImage) to support grading, auditability, and validation.

This ERD effectively models hierarchical relationships using Firebase's NoSQL principles, ensuring scalability and support for Smart Grading's core operations including multi-course management, automated grading, and visual answer validation.

3.6 Chapter Summary

This chapter outlined the methodology and design considerations for developing the Smart Grading: A Mobile OMR Scanner App for Lecturers. It began by discussing the Agile methodology adopted to ensure iterative development and continuous feedback throughout the project lifecycle. Requirement elicitation was conducted through interviews to gather insights into the grading processes and challenges faced by lecturers. The requirement analysis was supported by visual models, including use case, sequence, and class diagrams, which provided a structured view of the system's features and interactions.

The system design was then detailed through activity diagrams, illustrating workflows for functionalities such as managing exams, managing courses, grading answer papers, analysing exam performance, and exporting results. User interface designs were presented to showcase the app's usability and intuitive navigation. Lastly, the database design was represented through an ERD, ensuring a robust and scalable framework for data management. Together, these elements establish a solid foundation for implementing a functional and efficient grading solution.

CHAPTER 4

IMPLEMENTATION

4.1 Overview

This chapter presents the implementation phase of the Smart Grading mobile application. Following the design and planning stages outlined in Chapter 3, the system was developed in accordance with the Agile methodology, incorporating continuous feedback and iterative refinement. The focus of this chapter is to describe the actual construction of the system, including both frontend and backend components, as well as the integration of the Optical Mark Recognition (OMR) image processing workflow.

The implementation was divided into several key components, each responsible for fulfilling specific functionalities within the application. These include the mobile app interface developed using Flutter, Firebase for authentication and real-time database management, and a Python-based backend that handles image processing using OpenCV. The development process ensured that each module functioned independently while maintaining seamless interaction across the overall system.

Throughout this chapter, the technologies, tools, and frameworks used in the implementation are discussed, followed by detailed explanations of the development of core features such as exam management, answer key configuration, OMR answer paper scanning, grading logic, and data export functionalities. Each implementation step aims to ensure accuracy, usability, and performance, ultimately delivering a practical and reliable grading tool for lecturers at UNIMAS.

4.2 Technologies and Tools Overview

This section outlines the main technologies and platforms used to build the Smart Grading system. The project combined a mobile frontend (FlutterFlow), a Python backend (Flask + OpenCV), and Firebase services for authentication, data storage, and image hosting. Together, these components enabled real-time, accurate grading using Optical Mark Recognition (OMR).

4.2.1 Technology Stack

The Smart Grading system was developed using a combination of mobile development platforms, backend frameworks, cloud services, and essential development tools, each chosen for their specific contribution to functionality, efficiency, and integration.

Table 4.1: Core Technologies Used in Smart Grading

Layer	Technology/Platform	Purpose
Frontend	FlutterFlow	UI design and mobile app logic
Backend	Python + Flask	Image processing and API serving
Backend Services	Firebase (Auth, Firestore, Storage)	Handles login, stores exam data, and hosts answer paper images
Backend Hosting	Hugging Face Spaces	Hosting Flask server and grading API
Development Tools	Android Studio	Device testing and release build generation
	Postman	API testing and validation during backend development
	Postman	Backend development and debugging of Python + OpenCV logic
	GitHub	Version control, backup, and team collaboration

As shown in Table 4.1, the integration of these technologies and tools enabled seamless communication between the mobile app and backend services, ensuring efficient and accurate real-time grading.

4.3 Justification for Tool Selection

Each tool was selected based on its suitability for mobile-first grading applications. The following sections briefly justify their usage in this project:

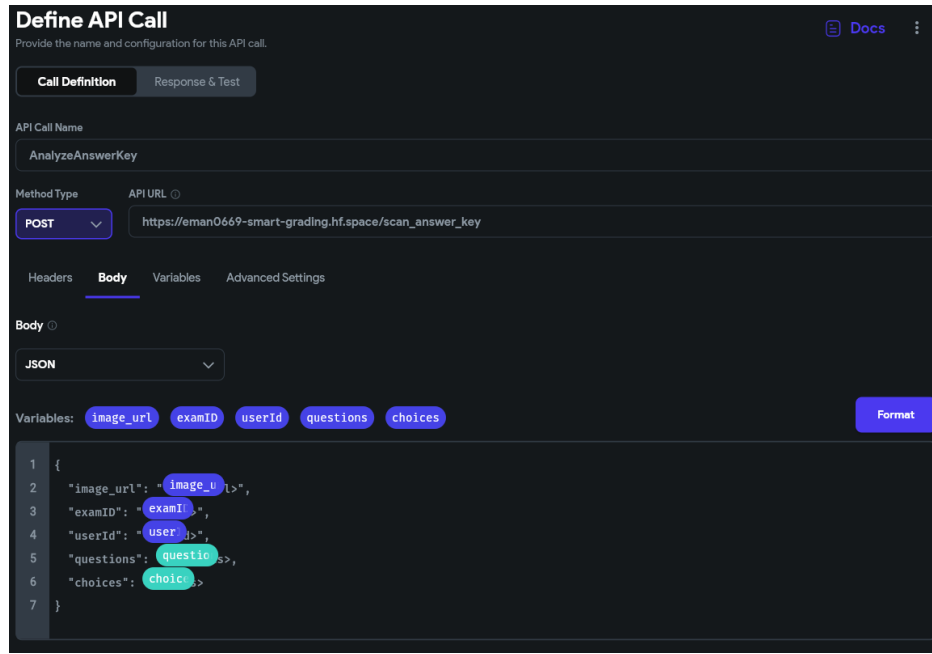


Figure 4.1: API Call to Flask Backend via FlutterFlow

- **FlutterFlow:** Enabled rapid UI development and integration with Firebase and custom APIs. Allowed flexible frontend logic and API call setup as shown in the Figure 4.1.
- **Firebase:** Provided secure login, real-time database, and image storage. Its seamless integration with FlutterFlow made it ideal for mobile deployment.
- **Python + Flask:** Allowed fast API development for answer grading using image processing techniques.
- **OpenCV:** Handled all core image analysis tasks like contour detection, warping, and bubble recognition with high accuracy.
- **Hugging Face Spaces:** Offered a lightweight, free hosting platform for the Flask backend, suitable for development and testing.

4.4 Core Application Modules

This section outlines the main functional modules of the Smart Grading mobile application. Each module represents a core feature that enables lecturers to manage the grading process ranging from user authentication to course creation, exam setup, answer key configuration, answer paper scanning, and result export. These modules are designed to integrate seamlessly, providing a streamlined and efficient user experience.

4.4.1 User Authentication Module

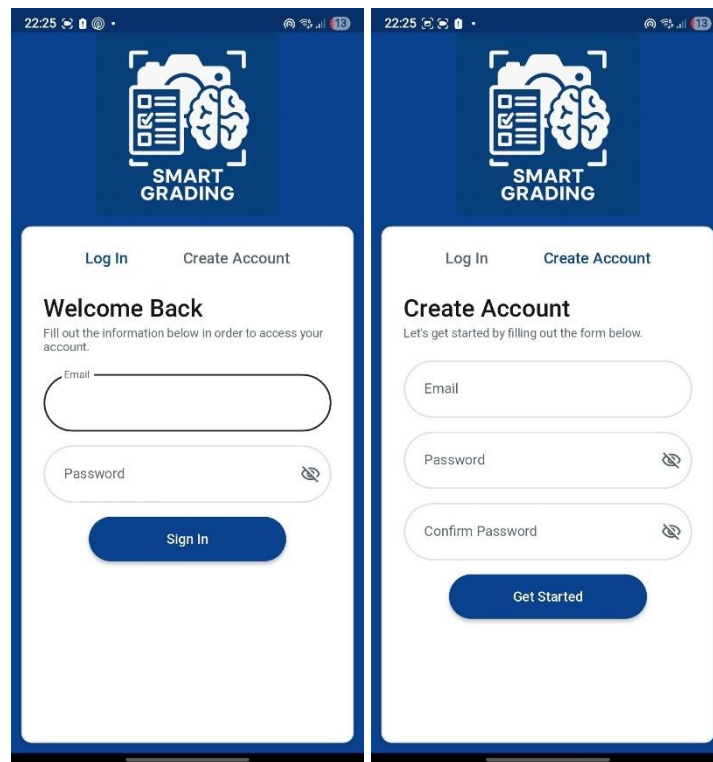


Figure 4.2: Login and Sign-Up Page

When users open the Smart Grading app, they are taken to an authentication screen with two options: login or sign-up as shown in the Figure 4.2. Lecturers can log in using their registered email and password, while new users can create an account with a valid email and password. The interface includes password visibility toggles for ease of use. All login data

is securely handled through Firebase Authentication. After successful login or registration, users are redirected to the exam dashboard.

4.4.2 Exam Management Module

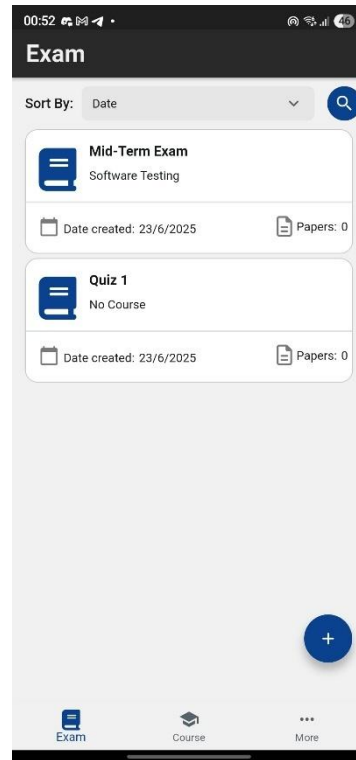


Figure 4.3: Exam Dashboard

Upon logging in, users will see a list of all exams they have created, along with their details, as shown in Figure 4.3. Here, users can sort exams by date or name and can also search for specific exams. If no exams are found, a placeholder message will appear prompting the user to add a new exam using the + button.

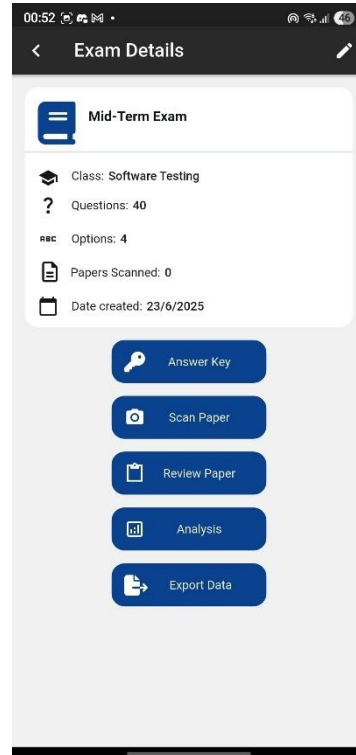


Figure 4.4: Exam Details Page

As shown in Figure 4.4, this screen displays key metadata of a selected exam, including its name, course, number of questions, number of options, and creation date. Below, lecturers can access features such as setting the answer key, scanning papers, reviewing results, viewing analytics, and exporting data.

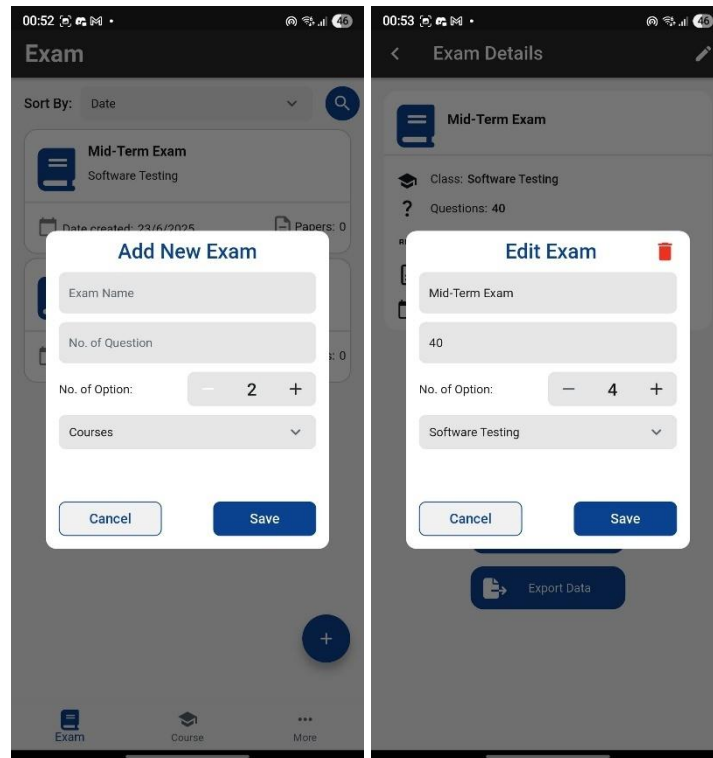


Figure 4.5: Add and Edit Exam Dialogs

As illustrated in Figure 4.5, these interfaces allow lecturers to create or modify exams efficiently. The "Add New Exam" dialog is accessed via the "+" button and enables users to input the exam name, number of questions, answer options, and course. The "Edit Exam" dialog, shown after tapping the pencil icon, allows users to update existing exam details. Additionally, a red trash icon is available in the edit view, enabling users to permanently delete the selected exam. This setup offers both flexibility and control in managing assessments.

4.4.3 Course Management Module

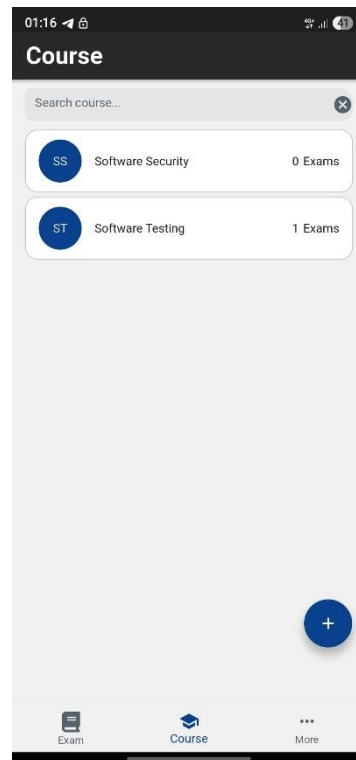


Figure 4.6: Course List Interface

As shown in Figure 4.6, users can view all the available courses they have created, along with the number of exams linked to each course. A search bar is provided at the top for easy filtering, and new courses can be added by tapping the + button in the bottom-right corner.

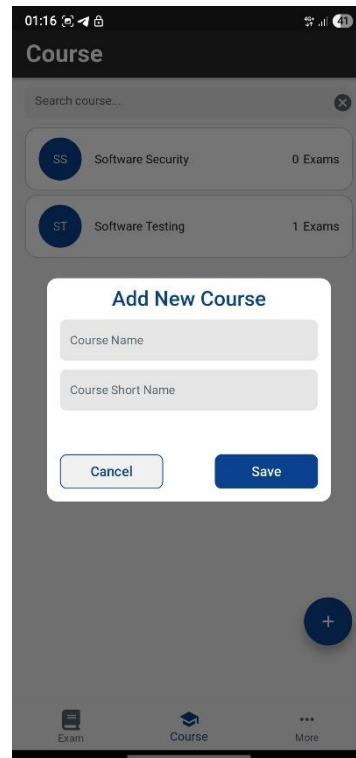


Figure 4.7: Add New Course Dialog

As illustrated in **Figure 4.7**, this pop-up allows users to add a new course by providing its full name and a short name. The short name is used solely to display a two-letter abbreviation in the course UI card for visual identification. Once saved, the course becomes available for exam linking.

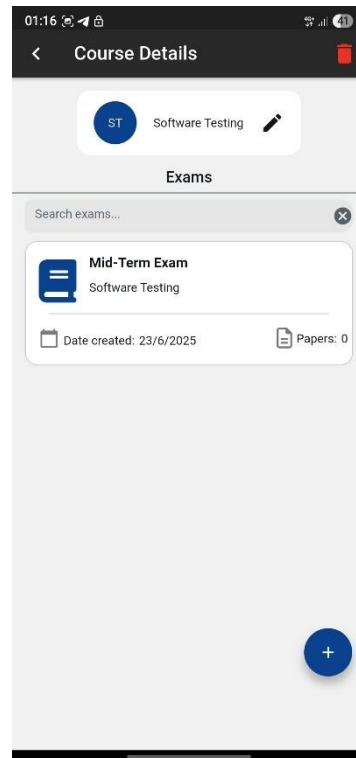


Figure 4.8: Course Details Page - Linked Exams

The course details interface, presented in **Figure 4.8**, lists all exams linked to a selected course. It allows users to browse and search within the course's exams, while also offering quick actions to edit or delete the course. New exams can be added directly from this view using the + button, streamlining the management of course-specific assessments.

4.4.4 Answer Key Configuration Module

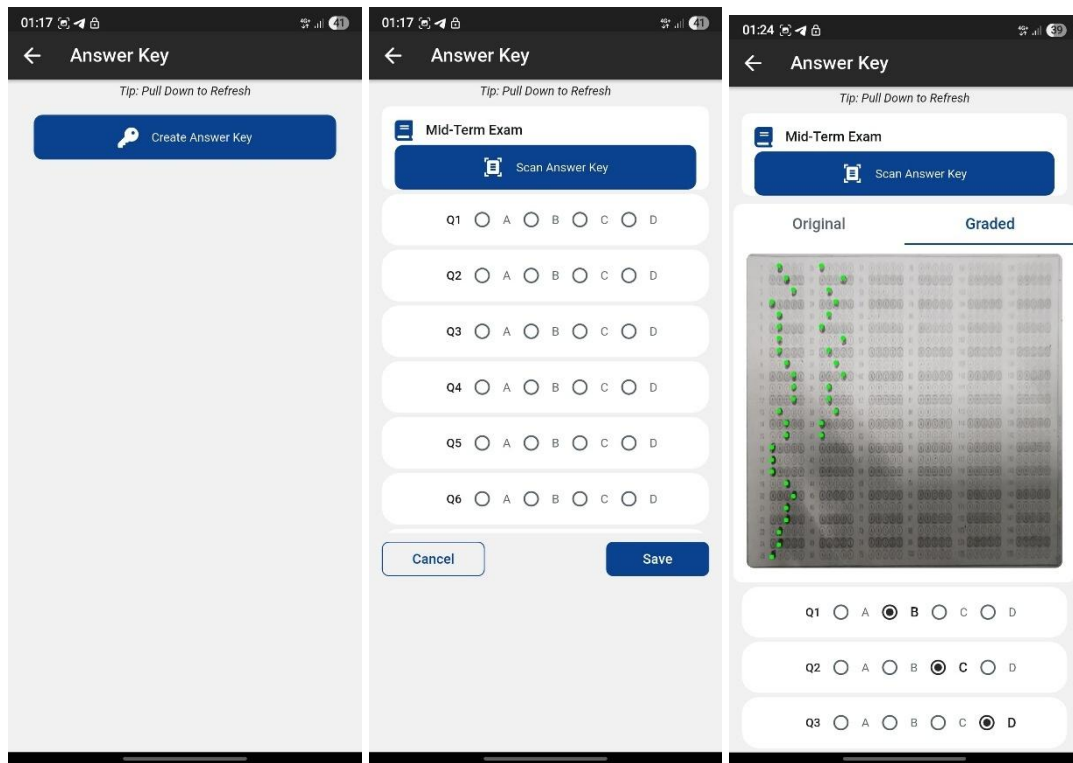


Figure 4.9: Creating and Scanning the Answer Key

When accessing the Answer Key module for the first time as shown in the first image of the figure 4.9, users will be prompted with a "Create Answer Key" button. Tapping this opens a manual entry interface allowing lecturers to specify the correct option for each question. Alternatively, users can scan a printed answer paper using the Scan Answer Key feature, which automatically fills in answers and provides a visual confirmation of the detected selections. This speeds up the process while ensuring accuracy.

4.4.5 Scanning and Grading Module

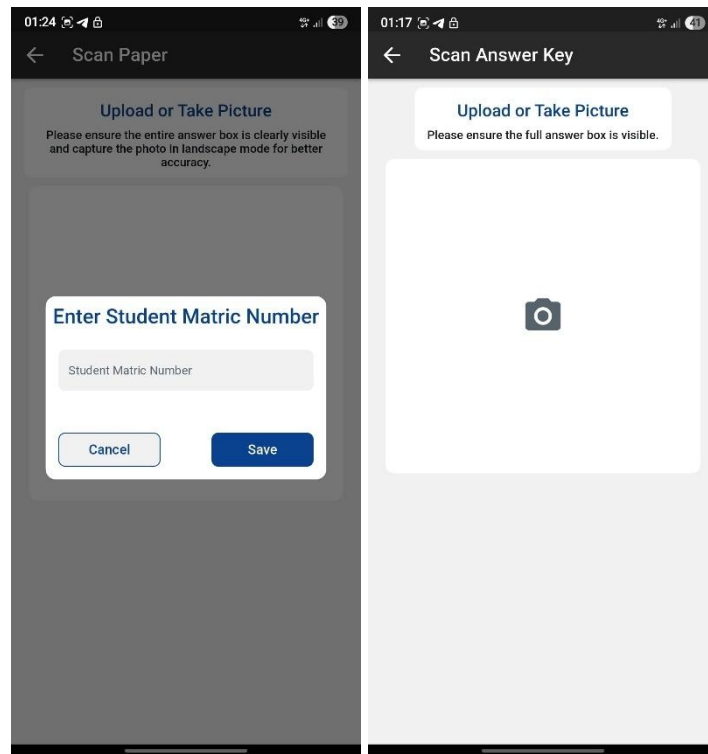


Figure 4.10: Student Matric Entry and Scan Page for Answer Papers

Figure 4.10 illustrates the student matric entry and scan interface. When users tap on the Scan Paper option, a prompt appears for entering the student's matric number ensuring the scan is properly linked. After confirming, the interface transitions to an upload screen with visual guidance at the top, reminding users to capture the entire answer box for accurate processing.

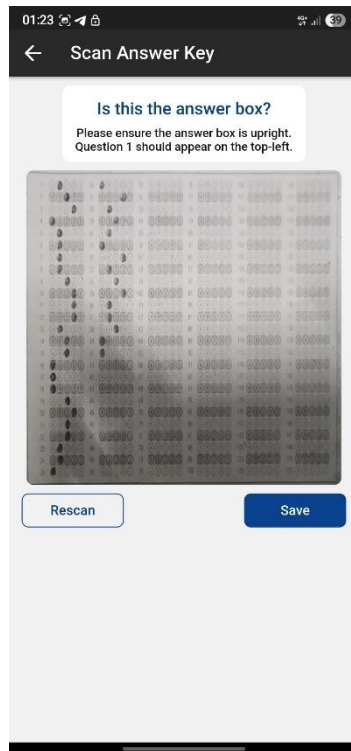


Figure 4.11: Confirmation screen after answer box is successfully detected.

After the system detects an answer box, a confirmation screen is shown in Figure 4.11, presenting a preview of the scanned paper. A helpful tip at the top reminds users to check that Question 1 is positioned in the top left. Users can either rescan the paper or proceed by tapping “Save” to finalise the grading.

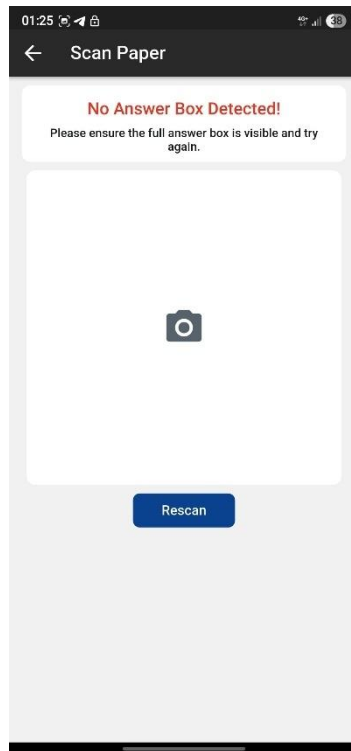


Figure 4.12: Error screen when no answer box is detected.

If the system is unable to detect an answer box, it displays an error screen as shown in Figure 4.12. A red warning message appears along with a prompt encouraging the user to try again. This helps ensure that the captured image includes the complete answer area, maintaining accuracy during scanning.

4.4.6 Exporting Results Module

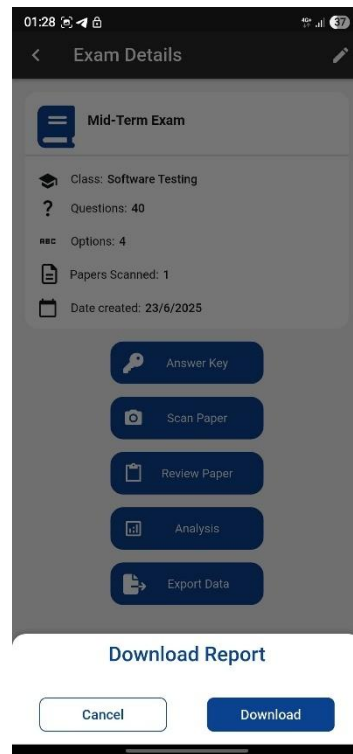


Figure 4.13: Exporting exam data report.

As illustrated in Figure 4.13, tapping the Export Data button from the Exam Details page opens a confirmation dialog prompting the user to download the report. The exported file summarises scanned exam data and serves as a useful resource for documentation and record keeping. Users can choose to cancel or proceed with the download as needed.

4.5 Image Processing Workflow

This section describes how scanned answer papers are processed using Python and OpenCV to detect filled bubbles, compare them to the answer key, and return grading results. The process consists of several sequential steps.

4.5.1 Image Acquisition and Preprocessing

```
#Read image
img = cv2.imread(local_path)

widthImg, heightImg = 700, 700
questions = len(answerKey)
img = cv2.resize(img, dsize=(widthImg, heightImg))

#Preprocessing
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imgBlur = cv2.GaussianBlur(imgGray, ksize=(5, 5), sigmaX: 1)
```

Figure 4.14: Image reading and preprocessing

As shown in Figure 4.14, once the image is received from the app, it is downloaded and resized to a fixed resolution for consistency. It is then converted to grayscale and processed using Gaussian blur to reduce noise.

4.5.2 Contour Detection and Perspective Warping

```
#Contour detection
found, biggestContour = False, None
for t1 in range(10, 100, 5):
    for t2 in range(t1 + 10, 150, 5):
        imgCanny = cv2.Canny(imgBlur, t1, t2)
        contours, _ = cv2.findContours(imgCanny, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        rectCon = []
        for c in contours:
            peri = cv2.arcLength(c, closed=True)
            approx = cv2.approxPolyDP(c, 0.02 * peri, closed=True)
            if len(approx) == 4 and cv2.contourArea(c) > 50000:
                rectCon.append(approx)

        if len(rectCon) > 0:
            biggestContour = rectCon[0]
            found = True
            break
    if found:
        break

if not found:
    os.remove(local_path)
    raise RuntimeError("No answer box detected.")

biggestContour = reorder(biggestContour)
pts1 = np.float32(biggestContour)
pts2 = np.float32([[0, 0], [widthImg, 0], [0, heightImg], [widthImg, heightImg]])
matrix = cv2.getPerspectiveTransform(pts1, pts2)
imgWarpColored = cv2.warpPerspective(img, matrix, dsize=(widthImg, heightImg))

#Threshold
imgWarpGray = cv2.cvtColor(imgWarpColored, cv2.COLOR_BGR2GRAY)
imgThresh = cv2.adaptiveThreshold(
    imgWarpGray, maxValue=255,
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
    cv2.THRESH_BINARY_INV,
    blockSize=23, C=13
)
```

Figure 4.15: Code for contour detection and perspective warp

In this step, Canny edge detection and dilation are used to highlight contours in the image, as illustrated in Figure 4.15. A contour refers to the outline of a shape, and the largest rectangular contour is assumed to be the answer box. After detecting and reordering its corners, a perspective warp is applied to produce a clean top-down view. The warped image is then thresholded using adaptive thresholding, turning filled bubbles into white regions for easier detection.

4.5.3 Bubble Segmentation

```
# Columns (start, end)
columns = [
    (47, 127),
    (159, 239),
    (270, 350),
    (381, 461),
    (494, 574),
    (605, 685)
]

# Rows (y_start, height)
row_y_ranges = [
    (17, 24), (43, 24), (70, 24), (97, 24), (124, 24),
    (150, 24), (177, 24), (204, 24), (231, 24), (258, 24),
    (285, 24), (311, 24), (338, 24), (365, 24), (392, 24),
    (419, 24), (446, 24), (473, 24), (499, 24), (526, 24),
    (554, 24), (580, 24), (607, 24), (634, 24), (661, 24)
]

all_boxes = []

for col_idx, (x_start, x_end) in enumerate(columns):
    col_img = imgThresh[:, x_start:x_end]
    col_rows = []
    for y_start, height in row_y_ranges:
        y_end = y_start + height
        row_img = col_img[y_start:y_end, :]
        col_rows.append(row_img)
    all_boxes.append(col_rows)

# Split into options for each question
all_answers = []
for col_rows in all_boxes:
    for box in col_rows:
        w = box.shape[1]
        option_width = w // 5
        options = [
            box[:, 0:option_width],
            box[:, option_width:option_width * 2],
            box[:, option_width * 2:option_width * 3],
            box[:, option_width * 3:option_width * 4],
            box[:, option_width * 4:w]
        ]
        all_answers.append(options[:choices])
```

Figure 4.16: Slicing the warped image into rows and answer options

Figure 4.16 demonstrates how the thresholded image is manually sliced into six columns, each containing 25 rows. These rows are further divided into sections representing answer options (A–E), preparing the image for the analysis stage.

4.5.4 Answer Detection and Grading

```
#Count pixels
myPixelVal = np.zeros(shape=(questions, choices), dtype=np.int32)
for q_index, option_imgs in enumerate(all_answers[:questions]):
    for c_index, imgAnswer in enumerate(option_imgs):
        myPixelVal[q_index][c_index] = cv2.countNonZero(imgAnswer)

#Get selected options
myIndex = []
for x in range(questions):
    arr = myPixelVal[x]
    selected = np.argmax(arr)
    myIndex.append(int(selected))

#Map selected indices
index_to_letter = {i: chr(65 + i) for i in range(choices)}
answers_as_letters = [index_to_letter.get(i, '?') for i in myIndex]

grading = []
for x in range(questions):
    if answerKey[x] == myIndex[x]:
        grading.append(1)
    else:
        grading.append(0)
totalMarks = (sum(grading))
```

Figure 4.17: Counting pixels and determining selected answers

As depicted in Figure 4.17, the number of white pixels in each bubble is calculated using `cv2.countNonZero()`. The bubble with the highest count is selected as the answer. These are then compared to the answer key to compute the student's score. Correct and incorrect answers are highlighted on the image using coloured circles.

4.5.5 Upload and Response Delivery

```
#Draw result
imgResult = imgWarpColored.copy()
showAnswers(imgResult, myIndex, answerKey, grading, questions, all_boxes)
print("Answer circles drawn.")

#Save and upload graded image
graded_temp = tempfile.NamedTemporaryFile(delete=False, suffix=".jpg")
cv2.imwrite(graded_temp.name, imgResult)
print("Replace img.")
graded_temp.close()

graded_path = f"users/{userId}/GradedImages/{examID}_{studentMatric}_{unique_id}.jpg"
graded_url = upload_image_to_firebase(graded_temp.name, graded_path)

os.remove(local_path)
os.remove(graded_temp.name)
delete_firebase_image_by_url(image_url)

return {
    "gradedImageUrl": graded_url,
    "totalMark": totalMarks,
    "answersSelected": myIndex,
    "answersSelectedLetters": answers_as_letters,
    "grading": grading
}
```

Figure 4.18: Uploading the marked image and returning results

Finally, once grading is complete, the marked image is saved and uploaded to Firebase Storage. As shown in Figure 4.18, the backend returns a JSON response containing the student's total mark, selected answers, and a URL to the graded image. This enables instant result display on the mobile app.

4.6 Chapter Summary

This chapter described the full implementation of the Smart Grading mobile application, including the tools, technologies, and modular components used. The system was developed using FlutterFlow for the frontend, Python with Flask and OpenCV for the backend, and Firebase for authentication, data storage, and image hosting. Each core module such as exam and course management, answer key setup, paper scanning, and result exporting was explained in detail, supported by visual references and backend logic.

The image processing workflow was also presented, outlining how scanned answer papers are preprocessed, segmented, and graded in real time using OpenCV techniques. These implementations ensured a functional, accurate, and efficient grading experience tailored to academic use.

With the system fully developed, the next chapter focuses on evaluating its performance and usability through systematic testing and user feedback. This helps verify whether the Smart Grading system meets its intended objectives in practical scenarios.

CHAPTER 5

TESTING

5.1 Overview

This chapter outlines the testing activities conducted to verify the functionality and usability of the Smart Grading application. A combination of functional, black-box, and end-to-end testing was used to ensure each module worked as expected. Real Android devices were used to validate both system performance and user interaction. User Acceptance Testing (UAT) was also carried out, combining SUS, UEQ, and open-ended feedback. Overall, the app performed reliably, with a few noted limitations that will guide future improvements.

5.2 Testing Strategy

The testing strategy for Smart Grading focused on ensuring that each module of the application functioned as intended, provided accurate results, and maintained a smooth user experience across devices. A combination of functional testing, black-box testing, and end-to-end scenario testing was used to validate the system from both technical and user perspectives.

All core modules such as user authentication, exam management, course linking, answer key configuration, scanning and grading, and result exporting were tested independently and collectively to confirm their correctness. Testing was conducted using real Android devices to ensure compatibility and proper UI rendering. This allowed the developer to validate both backend API responses and frontend user interactions.

During development, incremental testing was applied after each major implementation phase. This approach ensured that bugs could be detected and fixed early

without affecting other parts of the system. While formal regression testing was not adopted, previously tested modules were manually re-tested after new features were introduced or bugs were resolved, helping to prevent unintended side effects.

In summary, the system was tested thoroughly across all modules, simulating realistic lecturer workflows to confirm that the grading process from login to report export operated reliably under expected usage scenarios

5.3 Test Case

Table 5.1: User Authentication Test Cases

Module Name: User Authentication • Test Case Description: This set of test cases evaluates the login and registration features to ensure secure access control for lecturers using the Smart Grading app. • Testing Objective: To verify that only authorised users can successfully register, log in, and access protected areas of the app, and that appropriate error messages are shown for invalid inputs.								
Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Severity Failure	Notes
TC-A01	Valid user registration	1. Launch the Smart Grading app 2. Tap "Sign Up" 3. Enter valid email and password 4. Confirm password 5. Tap "Register"	• Email: harith0669@gmail.com • Password: 12345678	Account is created successfully, and user is redirected to the Exam page	Works as expected	Pass	Medium	
TC-A02	Registration invalid email format	1. Launch the Smart Grading app 2. Tap "Sign Up" 3. Enter invalid email and valid password 4. Confirm password 5. Tap "Register"	• Email: harith0669 • Password: 12345678	Snackbar appear: "Error: The email address is badly formatted"	Works as expected	Pass	High	

TC-A03	Registration confirms password mismatch	1. Launch the Smart Grading app 2. Tap "Sign Up" 3. Enter valid email and password 4. Enter password that does not match the password field in the Confirm Password 5. Tap "Register"	- Email: harith0669@gmail.com - Password: 12345678 - Confirm Password: 123456	Snackbar appear: "Password don't match!"	Works as expected	Pass	High	
TC-A04	Valid login	1. Launch the Smart Grading app 2. Enter registered email and password 3. Tap "Login" button	- Email: emanarit0@gmail.com - Password: 12345678	User is authenticated and redirected to the Exam Dashboard page	Works as expected	Pass	Medium	
TC-A05	Invalid account credentials	1. Launch the Smart Grading app 2. Enter wrong email or password 3. Tap "Login" button	- Email: emanarit0@gmail.com - Password: 123	Snackbar appear: "Error: The supplied auth credential is incorrect, malformed or has expired"	Work as expected	Pass	High	

TC-A06	Logout Functionality	- Go to “More” page - Tap “Logout” button	User: emanarit0@gmail.com	User is logged out and is directed to the login & sign-up page	Work as expected	Pass	Medium	
--------	----------------------	--	---	--	------------------	------	--------	--

Table 5.2: Exam Management Module Test Cases

Module Name: Exam Management Test Case Description: Verifies that lecturers can create, view, edit, delete, and manage exams. Testing Objective: To ensure exam creation and management features function correctly, and user interactions trigger the expected system responses.								
Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Severity Failure	Notes
TC-E01	Add new exam with valid input	- Tap “+” button in Exam page - Fill all fields - Tap “Create”	- Name: Quiz 1 - Questions: 10 - Options: 5 - Course: No Course	New exam appears in the exam page list	Works as expected	Pass	Medium	
TC-E02	Add new exam with missing name	- Tap “+” button in Exam page - Leave exam name blank - Fill other fields - Tap “Create”	- Name: - Questions: 10 - Options: 5 - Course: No Course	Error message: “Exam name is required”	Works as expected	Pass	Medium	
TC-E03	Add exam with invalid question count	- Tap “+” button in Exam page - Fill other fields - Enter negative number or zero for number of questions field	- Name: Quiz - Questions: 0 - Options: 4 - Course: Math	Error message: “Number of questions must be between 1 - 150”	Works as expected	Pass	High	

		4. Tap “Create”						
TC-E04	Add exam with missing course	1. Tap “+” button in Exam page 2. Fill other fields but do not select a course 3. Tap “Create”	• Name: Quiz • Questions: 10 • Options: 4 • Course:	Error message: “Please select a course or just select No Course”	Works as expected	Pass	Medium	
TC-E06	View exam details	1. Tap on an existing exam card	N/A	Redirected to exam details page	Works as expected	Pass	Low	
TC-E07	Valid edit exam details	1. Open existing exam 2. Tap edit icon 3. Change exam name, number of questions, number of options and course 4. Tap “Save”	• New name: "Mid-Term" • New number of questions: 20 • New Options: 5 • Course: Software Testing	Exam details are updated successfully	Works as expected	Pass	Low	
TC-E08	Edit exam with missing name	1. Open existing exam 2. Tap edit icon 3. Remove exam name 4. Tap “Save”	• Name: • Questions: 10 • Options: 5 • Course: No Course	Error message: “Exam name is required”	Works as expected	Pass	Medium	

TC-E09	Edit exam with invalid question count	1. Open existing exam 2. Tap edit icon 3. Enter large number 4. Tap "Save"	• Name: Quiz • Questions: 500 • Options: 4 • Course: Math	Error message: "Number of questions must be between 1 - 150"	Works as expected	Pass	High	
TC-E10	Delete an exam	1. Open existing exam 2. Tap edit icon 3. Tap trash icon 4. Confirm deletion	N/A	Redirected the user to Exam page and the exam is removed from the list	Works as expected	Pass	Medium	
TC-E11	Search for an exam	1. Tap search icon 2. Enter keyword in search bar	Keyword: Quiz	Matching exams are displayed	Works as expected	Pass	Low	
TC-E12	No exam fallback view	1. Log in with a new account 2. Navigate to exam page with no exams	N/A	Text prompts user to add a new exam.	Works as expected	Pass	Low	

Table 5.3: Course Management Module Test Cases

Module Name: Course Management • Test Case Description: Verifies that lecturers can create, view, edit, delete, and manage courses and associate exam data. • Testing Objective: To ensure that all course-related features function as intended, including adding new courses, updating existing ones, deleting unused courses, and linking exams correctly.								
Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Severity Failure	Notes
TC-C01	Add new course with valid input	1. Tap “+” button in Course page 2. Fill all fields 3. Tap “Create”	• Name: “Software Testing” • Short Name: “ST”	New course appears in the course page list	Works as expected	Pass	Medium	
TC-E02	Add new course with missing name	1. Tap “+” button in course page 2. Leave course name blank 3. Fill short name field 4. Tap “Create”	• Name: • Short Name: “ST”	Error message: “Course name is required”	Works as expected	Pass	Medium	
TC-E03	Add course with invalid short name	1. Tap “+” button in course page 2. Fill course name field 3. Leave short name blank 4. Tap “Create”	• Name: “Software Testing” • Short Name:	Error message: “Course short name is required”	Works as expected	Pass	High	
TC-E06	View course details	1. Tap on an existing course card	N/A	Redirected to course page details page where all exams under the course will be shown	Works as expected	Pass	Low	

TC-E07	Valid edit course details	1. Open existing course 2. Tap edit icon 3. Change course name and short name 4. Tap "Save"	• New name: "Software Security" • New short name: "SS"	Course details are updated successfully and every exam under the course will change its course name	Works as expected	Pass	Low	
TC-E08	Edit course with missing name	1. Open existing course 2. Tap edit icon 3. Remove course name 4. Tap "Save"	• Name: • Short name: "SS"	Error message: "Course name is required"	Works as expected	Pass	Medium	
TC-E09	Edit course with missing short name	1. Open existing exam 2. Tap edit icon 3. Remove short name 4. Tap "Save"	• Name: "Software Security" • Short name:	Error message: "Course name is required"	Works as expected	Pass	High	
TC-E09	Edit course with more than 2 letters short name	1. Open existing exam 2. Tap edit icon 3. Fill in the short name with new data 4. Tap "Save"	• Name: "Software Security" • Short name: "SSE"	Limit user input to 2 letters only	Works as expected	Pass	High	
TC-E10	Delete a course	5. Open existing course 6. Tap trash icon 7. Confirm deletion	N/A	Redirected the user to the course page, the course is removed from the list and every exam linked with	Works as expected	Pass	Medium	

				the course will be No Course				
TC-E11	Search for an exam	1. Enter keyword in search bar	Keyword: Software Testing	Matching exams are displayed	Works as expected	Pass	Low	

Table 5.4: Answer Key Configuration Module Test Cases

Module Name: Answer Key Configuration • Test Case Description: Verifies that lecturers can create, scan, view, and edit the answer key for each exam, whether manually or through image-based scanning. • Testing Objective: To ensure that the answer key configuration process works correctly, including manual input, automated scanning, visual feedback, and data persistence.								
Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Severity Failure	Notes
TC-K01	Create answer key manually with valid data	1. Tap “Create Answer Key” 2. Enter correct options for all questions 3. Tap 'Save'.	• Q1: A • Q2: B • Q3: D • Q4: B • Q5: A	Answer key saved	Works as expected	Pass	Low	
TC-K02	Create answer key with missing answers	1. Tap “Create Answer Key” 2. Leave some questions unanswered 3. Tap 'Save'.	• Q1: A • Q2: • Q3: D • Q4: B • Q5:	Answer key saved but when user trying to go back a popup will appear alerting the user to fill in all the answer key	Works as expected	Pass	High	

TC-K03	Scan answer key from image with valid format	1. Tap “Scan Answer Key” 2. Select a clear image of the printed answer key.	• Picture of answer key with visible and aligned answer box	Answers automatically detected and save to the database. User can refresh the answer key page to see the new updated answer key	Works as expected	Pass	Medium	
TC-K04	Scan answer key with poor image quality	1. Tap “Scan Answer Key” 2. Select a blurry or incomplete image.	• Blurry image, partial answer box, poor lighting	Error message shown: ‘No answer box detected!’ and rescan button will appear	Works as expected	Pass	High	
TC-K05	Overwrite existing answer key	3. Tap “Answer Key” option 4. Re-enter answers for an existing exam 5. Tap “Save”	• Q1: B • Q2: A • Q3: C • Q4: D • Q5: B	Old answer key replaced with new entries	Works as expected	Pass	Medium	

Table 5.5: Scanning and Grading Module Test Cases

Module Name: Scanning and Grading • Test Case Description: Verifies that lecturers can scan student answer papers, confirm detected answer boxes, and store grading results accurately. • Testing Objective: To ensure the scanning and grading workflow including image upload, answer box detection, user confirmation, and database storage operates correctly and handles errors gracefully.								
Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Severity Failure	Notes
TC-S01	Scan paper with valid image	1. Tap “Scan Paper” 2. Enter valid matric number 3. Take pictures or upload the answer paper 4. Tap 'Save'.	• Student Matric Number: 77761 • Image: Clear answer paper with answer box in frame and in landscape mode	After warping the image, the app asks the user to confirm the answer box. If confirmed, the data is saved to the database and redirected to student details page.	Works as expected	Pass	High	
TC-S02	Scan paper without entering matric	1. Tap “Scan Paper” 2. Skip matric number 3. Tap “Save” on the popup	• Student Matric Number:	Error: “Student Matric Number is required”	Works as expected	Pass	Medium	
TC-S03	Upload blurry or partial answer paper	1. Tap “Scan Paper” 2. Enter matric number 3. Upload image with cut-off or blurry answer box	• Matric: 77761 • Image: Blurry / incomplete	Warning shown: “No Answer Box Detected!” and Rescan button appear	Works as expected	Pass	High	

TC-S04	Confirm wrong answer box	1. Tap “Scan Paper” 2. Enter matric number 3. Upload valid image 4. Tap “Rescan”	• Matric: 77761 • Image: Clear image with correct alignment	Image is discarded and user prompted to rescan	Works as expected	Pass	High	
TC-S05	Cancel scanning process	1. Tap “Scan Paper” 2. Enter matric number 3. Upload valid image 4. Tap back button before uploading	• Matric: 77761 • Image: Clear image with correct alignment	Snackbar appear: “Image processing failed. Please try again”	Works as expected	Pass	Low	
	Upload duplicate image for same student	1. Tap “Scan Paper” 2. Enter same matric as before 3. Upload same answer image again	• Matric: 77761 • Image: Previously uploaded answer paper	Previous entry is updated, or user is prompted to confirm overwrite	New entry created	Fail	Medium	Duplicate entries may appear in the list, potentially causing confusion despite grading data remaining accurate.
TC-S07	Upload valid image in portrait mode	1. Tap “Scan Paper” 2. Enter matric	• Matric: 77761 • Image: Full answer box but in portrait orientation	User is asked “Is this the answer box?” with a guide tip: “Q1 should appear at top-left”	Works as expected	Pass	Medium	

		3. Upload portrait-oriented image						
TC-S08	Upload corrupted or unsupported file	1. Tap “Scan Paper” 2. Enter matric 3. Upload corrupted file	• Matric: 77761 • Image: Corrupted or unsupported file format	Snackbar appear: “Image processing failed. Please try again”	Works as expected	Pass	High	
TC-S09	Manually edit scanned answers	4. Go to Exam Details 5. Tap a student record 6. Edit one or more detected answers 7. Save changes	• Edited answers	The modified answers are saved and reflected in the student details page and export report	Works as expected	Pass	Medium	

Table 5.6: Exporting Data Module Test Cases

Module Name: Scanning and Grading • Test Case Description: Verifies that lecturers can create, view, edit, delete, and manage exams. • Testing Objective: To ensure exam creation and management features function correctly, and user interactions trigger the expected system responses.								
Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Status	Severity Failure	Notes
TC-R01	Export exam data report	8. Go to Exam Details page 9. Tap “Export Data” button	• Existing graded data	A report file in CSV is generated and downloaded containing all scanned results for the selected exam	Works as expected	Pass	Low	Useful for documentation purposes

		10. Confirm download in the popup						
--	--	---	--	--	--	--	--	--

5.4 Real-World Testing Scenarios

In addition to structured module testing and user acceptance evaluations, Smart Grading was tested under various real-world conditions to assess its robustness and accuracy. These tests involved a total of 270 multiple-choice questions graded under different environmental and technical scenarios.

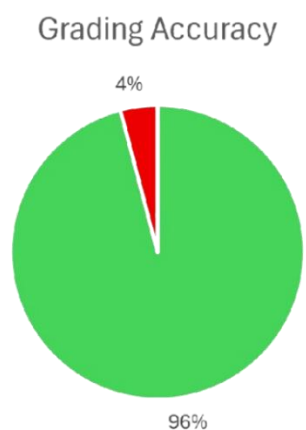


Figure 5.1: Grading Accuracy Distribution

As illustrated in Figure 5.1, the system achieved an overall accuracy rate of 96%, with only 4% errors. This demonstrates the reliability of the mobile-based OMR grading system in practical use. To further examine the system’s behaviour under challenging conditions, the app was tested using angled image captures, extreme lighting, long-format exams (150 questions), and across different devices.

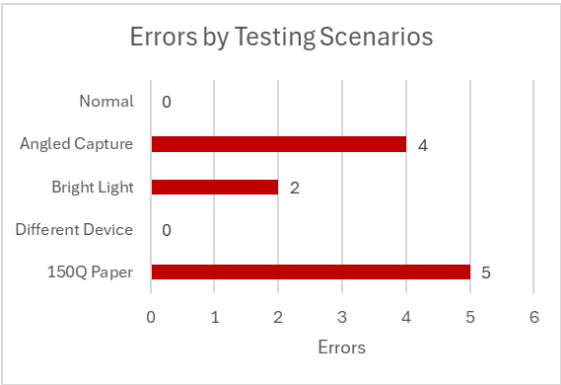


Figure 5.2: Error Breakdown by Testing Condition

The results are summarised in Figure 5.2: Error Breakdown by Testing Condition. The system showed zero errors under normal use and when operated on a different device. Minor errors occurred under angled capture (4 errors), bright lighting (2 errors), and during the 150-question paper test (5 errors). These conditions introduced slight deviations in image detection and marking accuracy.

Overall, the system maintained strong performance across varied testing conditions. The few errors recorded were linked to external factors such as lighting and image alignment rather than system flaws, reinforcing the viability of Smart Grading as a dependable academic grading tool.

5.5 User Acceptance Testing (UAT)

User Acceptance Testing (UAT) was carried out to gather feedback on the usability, functionality, and overall effectiveness of the Smart Grading application from a user's point of view. The goal was to determine whether the system is intuitive, reliable, and practical for real-world usage. Test participants explored key features such as scanning answer sheets, managing exams, and exporting results. Their responses provided valuable insights into how well the application supports grading tasks and whether it aligns with user expectations.

5.5.1 Participant Profile

As shown in Figure 5.1, a total of 24 participants took part in the User Acceptance Testing (UAT) for the Smart Grading application. Among them, 3 participants (12.5%) identified themselves as lecturers or educators, while the remaining 21 participants (87.5%) were random testers or volunteers. Additionally, 3 participants reported having prior experience with OMR grading tools either software-based or hardware-based systems. This

participant mix provided a balanced perspective between experienced users and new testers, offering valuable feedback on the system's usability and practicality.

5.5.2 System Usability Scale (SUS)

The System Usability Scale (SUS) was used to evaluate the perceived usability of the Smart Grading application. Participants were asked to rate ten usability statements using a five-point Likert scale. The results, summarised in Table 5.7, provide insight into the strengths and minor areas of improvement within the system.

Table 5.7: SUS Table

Statement	Mean Score
I think I would like to use this system frequently.	4.04
I found the system simple and easy to use.	4.29
I felt confident using the system.	4.29
The functions in this system were well integrated.	4.21
I think most people could learn to use this system quickly.	4.25
I found the system intuitive and user-friendly.	4.21
I could complete tasks in a straightforward way.	4.13
The system was responsive and performed as expected.	4.21
I could navigate the system without any confusion.	4.04
Overall, I am satisfied with the system's usability.	4.25

In summary, the SUS results show that the Smart Grading app is well-received by its target audience, offering a usable, accessible, and efficient grading solution. While overall satisfaction is high, minor adjustments to improve task flow and navigational support could further elevate the system's usability for a broader range of users.

5.5.3 User Experience Questionnaire (UEQ)

Table 5.8: UEQ Summary Table

Dimension	Mean Score
Attractiveness	3.96
Clarity	4.08
Efficiency	4.13
Dependability	4.04
Stimulation	4.04
Novelty	4.00

In conclusion, the UEQ results shown in the table 5.8 indicate that Smart Grading successfully delivers a user-friendly, efficient, and engaging experience. With all dimensions scoring above average, the system demonstrates strong potential for real-world use, particularly in academic environments where ease of use and task efficiency are critical. Minor enhancements in visual appeal and interactivity could further elevate the overall user experience.

5.5.4 Open Ended Questions

i. What did you like most about Smart Grading?

Users praised its ease of use, speed, reliable OMR scanning, and clear interface. Overall, they found it efficient and intuitive for grading tasks.

ii. What problems or bugs did you encounter?

Most users reported no issues, though some noted limitations such as dependence on precise answers, lack of offline support, and occasional connectivity concerns.

iii. Any features you'd like to suggest or improve?

Suggestions included smarter grading with contextual feedback, support for subjective questions, improved rural device compatibility, and flexible answer key formats.

iv. Other comments or feedback

Feedback was largely positive, with comments like “easy and useful,” “nicely done,” and “no issues,” reflecting strong user satisfaction.

5.6 Limitations and Known Issues

Despite positive test results and encouraging user feedback, several limitations and known issues were identified during system testing and User Acceptance Testing. One key limitation is the system’s strict reliance on exact answer matching. It lacks contextual understanding and flexible interpretation, which can lead to grading inaccuracies when bubbles are only partially filled or shaded imprecisely.

The system is also dependent on internet connectivity. Users in low-connectivity areas reported reduced usability, particularly when accessing support features or exporting data. Image quality presents another challenge. Blurry, cropped, or corrupted images may cause scanning failures or inaccurate detection results.

Currently, the system cannot identify unanswered questions accurately. If a student leaves a question blank, the app still selects the darkest bubble, even if none were deliberately marked. A possible workaround was considered: setting a threshold where selections below 50 detected pixels would be classified as unanswered. Similarly, the app does not handle multiple marked answers per question. A logic was planned to flag answers as invalid if two or more options exceeded 150 pixels in density, but time constraints prevented its implementation.

Additionally, the system allows the same answer paper to be submitted multiple times without any warning. While grading accuracy remains intact, this may lead to duplicate

records and confusion in result tracking. Lastly, although visual guidance is available when scanning in portrait mode, some users still found the process unclear. This suggests improvements are needed in the instructional design to enhance scanning confidence. These limitations do not impact the system's core functionality but highlight areas for improvement to increase flexibility, accuracy, and overall user satisfaction.

5.7 Chapter Summary

This chapter detailed the testing process for the Smart Grading application, covering internal system testing, real-world scenario testing, and user-based evaluations. All core modules were tested successfully using a combination of functional, black-box, and end-to-end methods to ensure reliability, accuracy, and a smooth user experience.

In addition to standard test cases, experimental testing was conducted under varied conditions such as angled captures, bright lighting, long-format exams, and different devices. These tests demonstrated the system's robustness and confirmed its ability to maintain high grading accuracy across real-world scenarios.

The User Acceptance Testing further validated the system's usability, with strong scores recorded in both the System Usability Scale (SUS) and User Experience Questionnaire (UEQ). While the system met its core objectives, several limitations were identified, including strict answer matching, lack of support for subjective questions, and limited offline flexibility. These insights provide a strong foundation for future improvements to increase flexibility, accuracy, and overall effectiveness.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Overview

This chapter provides the conclusion of the Smart Grading project, reflecting on the achievements, limitations, and future enhancements. It reviews how each objective was met, outlines key areas for improvement, and suggests potential features to expand the system's capabilities. The chapter also summarises the project's overall contribution to simplifying the grading process for UNIMAS lecturers through mobile and image processing technologies.

6.2 Future Work

Although the Smart Grading system has met its intended objectives, several areas for improvement have been identified. Future work may focus on enhancing user convenience and system flexibility. This includes enabling users to capture the entire answer paper without manually aligning the answer box, automating the slicing of answer areas to reduce setup complexity, and introducing an offline grading mode to support usage in areas with limited internet connectivity. These enhancements aim to make the system more efficient, adaptable, and accessible for a wider range of academic environments.

6.2.1 Enable User to Take Pictures of the Whole Paper

Currently, the Smart Grading system requires users to manually align and capture only the answer box section of the answer paper. While this approach works, it may be unintuitive or challenging for some users, especially in fast-paced or inconsistent scanning

conditions. To enhance usability, a future improvement would be to allow users to simply take a photo of the entire answer paper without needing precise alignment.

With this feature, the system would automatically detect the location of the answer box within the full image using computer vision techniques such as contour detection or trained shape classifiers. Once identified, the answer box could then be warped, cropped, and processed without requiring manual input. This enhancement would simplify the scanning process, reduce human error, and improve consistency across varied lighting or orientation conditions. It would also help first-time users or those unfamiliar with camera alignment to achieve accurate scans with minimal effort.

6.2.2 Offline Grading Mode

The current version of Smart Grading relies on a stable internet connection to upload images, process scans, and store grading results in the cloud. While this works well in connected environments, it poses limitations for users in rural or low-connectivity areas. To address this, future development could introduce an offline grading mode that allows lecturers to scan answer papers and temporarily store grading data locally on their device.

Once connectivity is restored, the system could automatically sync the stored data with the cloud database. This feature would enhance flexibility and ensure uninterrupted use of the app during on-site grading sessions, fieldwork, or examination halls without reliable Wi-Fi. Implementing offline support would also improve accessibility, especially in educational institutions where internet infrastructure is inconsistent.

6.3 Conclusion

Smart Grading successfully fulfilled all three project objectives: developing a mobile application that enables UNIMAS lecturers to scan OMR answer papers using their phone's

camera, integrating OMR technology to accurately detect and process marked answers, and automating the grading process to reduce errors and save time. The system leverages OpenCV for image processing and Firebase for seamless data management, resulting in a reliable and efficient grading workflow. The app's user interface was designed to support lecturers with minimal technical steps, allowing for straightforward exam setup, real-time scanning, and easy result exporting. These results highlight the effectiveness of combining mobile and image recognition technologies to enhance assessment processes in academic environments.

6.3.1 Conclusion Based on Objective 1

The first objective was to develop a mobile app that allows UNIMAS lecturers to scan OMR answer papers using their phone's camera. This objective was successfully achieved through the implementation of a fully functional Android application that supports both capturing and uploading answer paper images. The app provides an intuitive scanning interface with alignment guides to assist lecturers in accurately capturing answer boxes. Testing across various Android devices confirmed that the system performs consistently and efficiently, making it a practical tool for on-the-go grading without the need for specialised hardware.

6.3.2 Conclusion Based on Objective 2

The second objective was to integrate OMR technology into the mobile app to detect, and process marked answers on OMR papers. This goal was successfully met by leveraging OpenCV-based image processing techniques such as contour detection, warping, thresholding, and pixel analysis. The system is capable of identifying the answer box, extracting selected answers, and handling variations in image quality and orientation. Both manual and scanned answer key configurations were supported, ensuring flexibility for

lecturers. The integration proved reliable during testing, confirming that the app can process OMR papers accurately using only a mobile device.

6.3.3 Conclusion Based on Objective 3

The third objective was to automate the grading of OMR answer papers to help UNIMAS lecturers save time and reduce grading errors. This objective was successfully accomplished through the implementation of a grading pipeline that compares scanned student responses against a predefined answer key and calculates marks automatically. The system stores results in a structured database and provides options to review, edit, and export grading data in CSV format. User Acceptance Testing indicated that the automated grading process significantly reduces manual workload and minimises human error, supporting the overall goal of improving efficiency in academic assessment.

6.4 Chapter Summary

This chapter concluded the development and evaluation of the Smart Grading application. It outlined possible future improvements, such as enabling full-paper scanning, automatic answer box slicing, and offline grading support. The project successfully achieved all three objectives developing a mobile scanning app, integrating OMR technology, and automating grading to improve efficiency. The results confirm the feasibility and practicality of using mobile-based image processing to support lecturers in academic grading tasks.

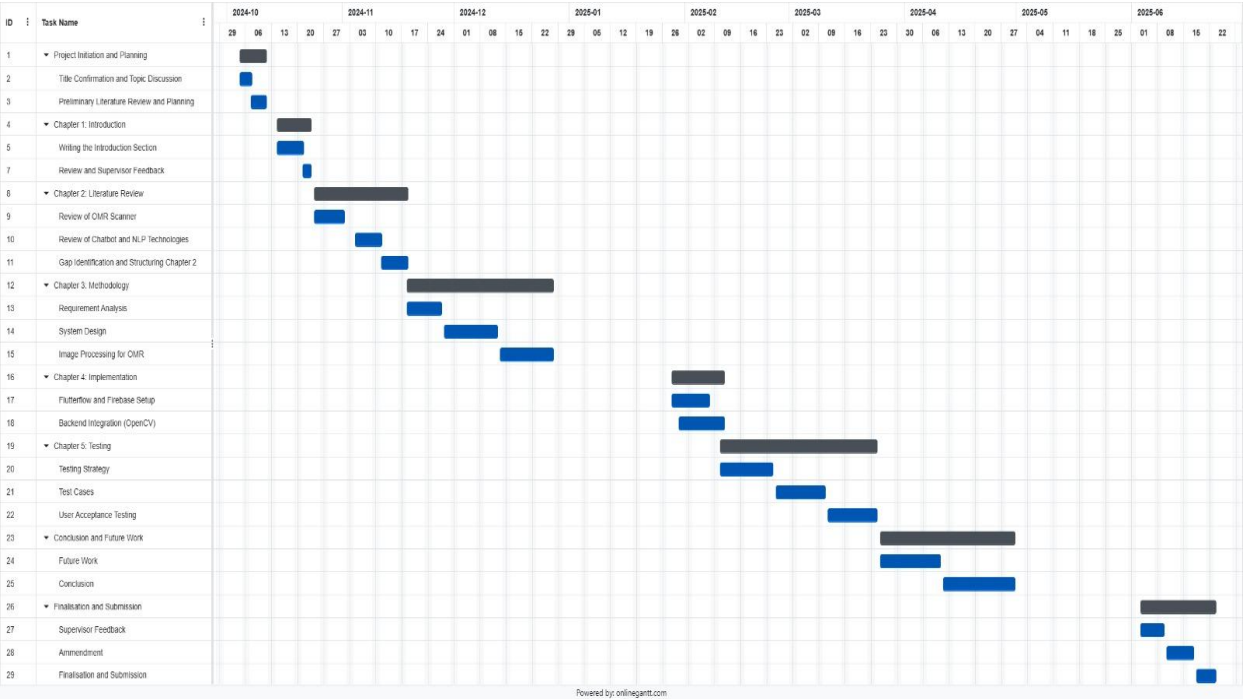
REFERENCES

- AbhiSubrahmanyam, K., Vineeth, M. S., Suhith, Y. M. S., & PrabhathBandi, K. (2022). *An Optical Mark Recognition and Evaluation System Based on Image Processing*, 7(7), ISSN: 2456-3315. <https://ijrti.org/papers/IJRTI2207124.pdf>
- Calaguas, N. P. C. & Consunji, P. M. P. (2022). Mobile Optical Mark Recognition Application As A Non-Inferior Alternative To Manual Marking For Lower- And Middle-Income Countries. *Journal of Advanced Studies*, 5(1), 10-19pp
- Cuerdo, R., Ison, M. J. B., & Oñate, C. D. T. (2021). Effectiveness of Automation in Evaluating Test Results Using EvalBee as an Alternative Optical Mark Recognition (OMR): A Quantitative-Evaluative Approach from a Philippine Public School. *International Journal of Theory and Application in Elementary and Secondary School Education*, 3(2), 61–75. <https://doi.org/10.31098/ijtaese.v3i2.661>
- Dharmik, R. C., Rangari, S., Jain, S., Nilawar, A., Deshmukh, G., & Yeole, A. (2024, January 7). *Optical Mark Recognition Evaluation System using Dual-Component Approach*. <https://ijisae.org/index.php/IJISAE/article/view/4383>
- Guo, R., & Jiang, Z. (2024). Optimal dynamic advertising policy considering consumer ad fatigue. *Decision Support Systems*, 187, 114323. <https://doi.org/10.1016/j.dss.2024.114323>
- Hafeez, Q., Aslam, W., Lali, M. I., Ahmad, S., Alqahtani, M., & Shafiq, M. (2022). Fault tolerant optical mark recognition. *Computers, Materials & Continua/Computers, Materials & Continua (Print)*, 74(2), 3829–3847. <https://doi.org/10.32604/cmc.2023.026422>

- Himabindu, G., Reeta, A., Srinivas Manikanta, A., & Manogna, S. (2023). *Evaluation of Optical Mark Recognition (OMR) Sheet Using Computer Vision*, 5(4).
https://www.irjmets.com/uploadedfiles/paper/issue_4_april_2023/35551/final/fin_irjmets1681035482.pdf
- Pushparaj, N. P., & Vidya., N. (2024). ACADEMIC STRESS AMONG NEET ASPIRANTS CASE STUDY. EPRA International Journal of Multidisciplinary Research (IJMR), 174–177. <https://doi.org/10.36713/epra17319>
- Qiao, Q. (2022). Image processing technology based on machine learning. *IEEE Consumer Electronics Magazine*, 13(4), 90–99. <https://doi.org/10.1109/mce.2022.3150659>
- Rangkuti, A. H., Athalia, Z., Thalia, T. E., Wiharja, C. E., & Rakhmansyah, A. (2023, July 16). *Economical and Efficient Multiple-Choice Question Grading System using Image Processing Technique*.
<https://www.ijisae.org/index.php/IJISAE/article/view/3159>
- Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM model for agile methodology. *In 2017 International Conference on Computing, Communication and Automation (ICCCA)*, 864–869. <https://doi.org/10.1109/ccaa.2017.8229928>
- What is Agile Methodology?* (2024, September 23). GeeksforGeeks.
<https://www.geeksforgeeks.org/what-is-agile-methodology/>
- White-Dzuro, C. G., Schultz, J. D., Ye, C., Coco, J. R., Myers, J. M., Shackelford, C., Rosenbloom, S. T., & Fabbri, D. (2021). Extracting Medical Information from Paper COVID-19 Assessment Forms. *Applied Clinical Informatics*, 12(01), 170–178.
<https://doi.org/10.1055/s-0041-1723024>

APPENDICES

Appendix A: Gantt Chart



Appendix B: User Acceptance Testing (UAT) Google Form Questions

Section 1: Tester Information

1. Full Name *

Your answer

2. Email Address *

Your answer

3. Role *

☐ Lecturer / Educator

☐ Random Tester / Volunteer

4. Have you used an OMR grading tool before? *

☐ Yes

☐ No

Section 2: Task Confirmation

Please download and use the images below to perform the testing tasks. These are required to ensure consistency in your feedback.

[Smart Grading UAT Kit](#)

5. Which of the following tasks did you test? *

☐ Upload Answer Key

☐ Scan Student Paper (Valid)

☐ Scan Student Paper (Invalid)

☐ View Grading Results

☐ Export Data

Section 3: System Usability Scale (SUS)

Rate the following statements from 1 (Strongly Disagree) to 5 (Strongly Agree)

6. I think I would like to use this system frequently. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

7. I found the system simple and easy to use. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

8. I felt confident using the system. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

9. The functions in this system were well integrated. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

10. I think most people could learn to use this system quickly. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

11. I found the system intuitive and user-friendly. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

12. I could complete tasks in a straightforward way. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

13. The system was responsive and performed as expected. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

14. I could navigate the system without any confusion. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

15. Overall, I am satisfied with the system's usability. *

	1	2	3	4	5	
Strongly Disagree	☐	☐	☐	☐	☐	Strongly Agree

Section 4: User Experience Questionnaire

Attractiveness *

	1	2	3	4	5	
Unpleasant	☐	☐	☐	☐	☐	Pleasant

Clarity *

	1	2	3	4	5	
Confusing	☐	☐	☐	☐	☐	Clear

Efficiency *

	1	2	3	4	5	
Inefficient	☐	☐	☐	☐	☐	Efficient

Dependability *

	1	2	3	4	5	
Unpredictable	☐	☐	☐	☐	☐	Predictable

Stimulation *

	1	2	3	4	5	
Boring	☐	☐	☐	☐	☐	Exciting

Novelty *

	1	2	3	4	5	
Conventional	☐	☐	☐	☐	☐	Innovative

Section 5: Open Feedback

16. What did you like most about Smart Grading? *

Your answer

17. What problems or bugs did you encounter?

Your answer

18. Any features you'd like to suggest or improve? *

Your answer

19. Other comments or feedback

Your answer

Smart Grading UAT Kit:

