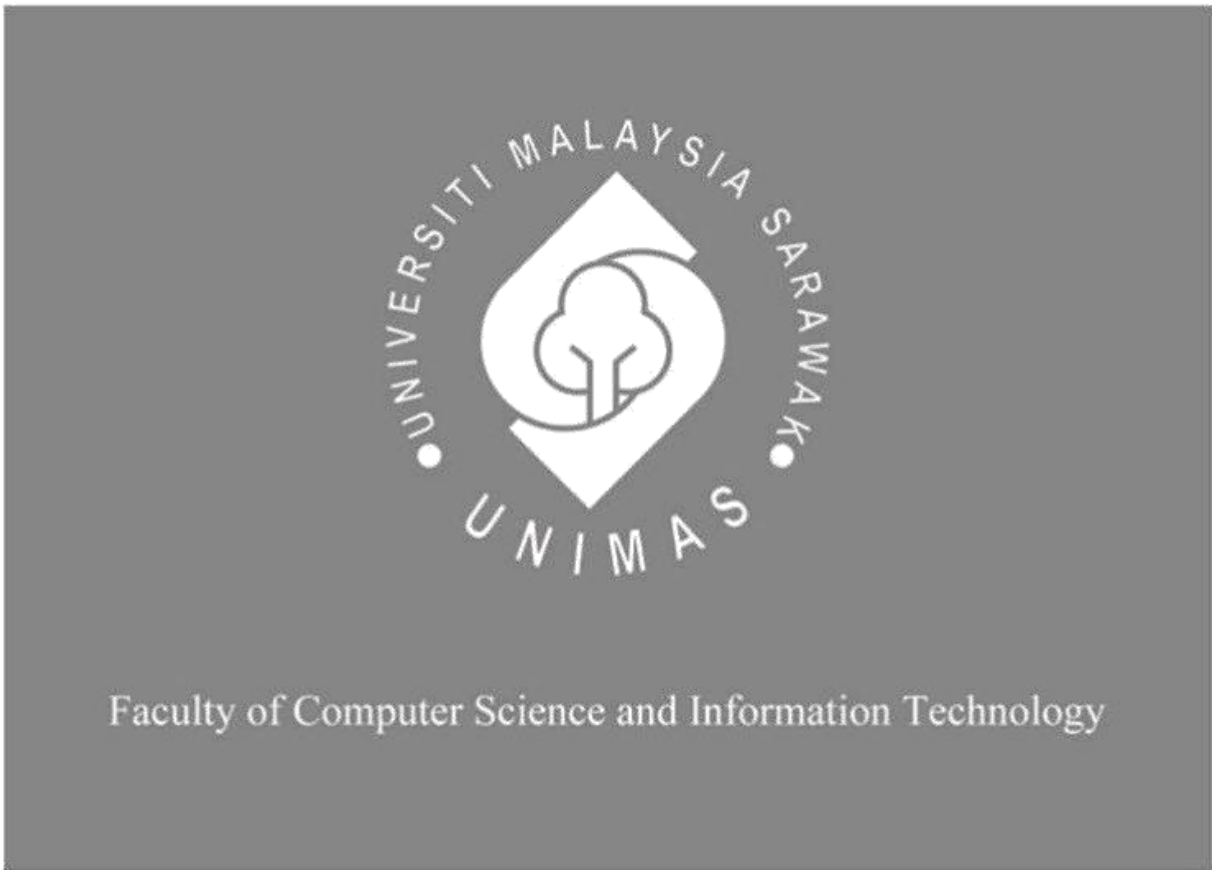


Automated Essay Grading System

Alexander Anak Adrian

Bachelor of Software Engineering with Honours

2024

UNIVERSITI MALAYSIA SARAWAK

THESIS STATUS ENDORSEMENT FORM

TITLE AUTOMATED ESSAY GRADING SYSTEM

ACADEMIC SESSION: 2024/2025

ALEXANDER ANAK ADRIAN
(CAPITAL LETTERS)

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

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

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

(AUTHOR'S SIGNATURE)

Permanent Address

Rh. Berendam, Ulu Durin Kiba,
96000, Sibu, Sarawak

Date: 25 July 2025

Validated by

(SUPERVISOR'S SIGNATURE)

Dr. Tan Ping Ping

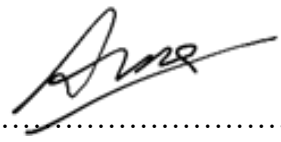
Date: 25th Jul, 2025

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

Declaration

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

Signed,



…….

ALEXANDER ANAK ADRIAN

Faculty of Computer Science and Information Technology

University Malaysia Sarawak

23 JUNE 2025

Acknowledgement

I would like to sincerely thank my supervisor, Dr. Tan Ping Ping, for her continuous guidance, constructive feedback, and unwavering support throughout the course of my Final Year Project. Her encouragement and expertise have been very helpful to me in the in completing this project. I would also like to extend my appreciation to Miss Feona anak Albert Abell for her willingness to contribute her time and insights during the testing phase of my project. Her valuable input helped ensure the system remains practical and relevant to MUET-based academic use. My gratitude also goes to the Final Year Project coordinator, Professor Dr. Wang Yin Chai, for providing clear guidelines and organizing the FYP structure in a way that kept us aligned and on track throughout both semesters. I am thankful to Universiti Malaysia Sarawak (UNIMAS) and the Faculty of Computer Science and Information Technology (FCSIT) for providing a conducive learning environment, useful resources, and the opportunity to grow both technically and academically. I also wish to acknowledge my friends and classmates, who have offered constant support, shared their knowledge, and stood by me during difficult parts of the project. Last but not least, I owe my deepest gratitude to my parents and family for their unconditional love, patience, and motivation. Their support has been the foundation of my perseverance and success throughout my studies.

Table of Contents

List Of Tables

List Of Figures

List Of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
CSS	Cascading Style Sheets
ERD	Entity Relationship Diagram
FDD	Feature-Driven Development
FELC	Faculty of Education, Language, and Communication
FYP	Final Year Project
HTML	HyperText Markup Language
JS	JavaScript
MUET	Malaysian University English Test
NLP	Natural Language Processing
OCR	Optical Character Recognition
ORM	Object-Relational Mapping
SDK	Software Development Kit
SQL	Structured Query Language
UI	User Interface
UNIMAS	Universiti Malaysia Sarawak
UAT	User Acceptance Testing
URL	Uniform Resource Locator
VS CODE	Visual Studio Code

Abstract

Grading an essay for pre-university students is not an easy task particularly for student who takes Malaysian University English Test (MUET). The lecturers at Universiti Malaysia Sarawak (UNIMAS) faces significant challenges due to its labor-intensive nature and potential inconsistencies in manual grading. This project proposes an Automated Essay Grading System that leverages Natural Language Processing (NLP) and Artificial Intelligence (AI) technologies to streamline the grading process. The system is developed using Feature-Driven Development (FDD) methodology, incorporating requirements gathered through interviews with a FELC lecturers. The proposed system includes key features such as automated grading based on MUET-specific rubrics, detailed feedback generation, and a user-friendly interface for digital essay submission and result review. Moreover, the analysis of existing systems revealed limitations in current solutions, particularly in providing detailed feedback and maintaining consistency with specific examination rubrics. A survey is conducted, including both lecturers and students, indicated positive reception towards AI integration in essay grading, while highlighting concerns about accuracy and bias. The proposed system can offer a platform for the FELC lecturers to quickly analyse students' essays, provide more personalized feedback, and also supporting students' progress in developing their writing skills. Based on testing with a MUET language lecturer and pilot tester, the system was found to be functionally reliable, and the AI-generated feedback was considered useful for formative learning.

Abstrak

Proses penggredan esei untuk pelajar pra-universiti bukanlah satu tugas yang mudah terutamanya bagi pelajar yang mengambil Malaysian University English Test (MUET). Para pensyarah di Universiti Malaysia Sarawak (UNIMAS) menghadapi cabaran yang besar berikutan sifat intensif buruh dan berpotensi tidak konsisten dalam penggredan secara manual. Projek ini mencadangkan Sistem Penggredan Esei Automatik yang memanfaatkan teknologi Pemprosesan Bahasa Semulajadi (NLP) dan Kecerdasan Buatan (AI) untuk menyelaraskan proses penggredan. Sistem ini dibangunkan menggunakan metodologi Feature-Driven Development (FDD), menggabungkan keperluan yang dikumpul melalui temu bual dengan pensyarah FELC. Sistem yang dicadangkan termasuk ciri utama seperti penggredan automatik berdasarkan rubrik khusus MUET, penjanaan maklum balas terperinci dan antara muka pengguna yang mesra pengguna untuk proses penghantaran esei digital dan semakan keputusan. Selain itu, analisis sistem sedia ada mendedahkan batasan dalam solusi penyelesaian yang wujud sekarang, terutamanya dalam menyediakan maklum balas terperinci dan mengekalkan konsistensi dengan rubrik peperiksaan tertentu. Tinjauan dijalankan, termasuk seorang pensyarah dan 4 orang pelajar, menunjukkan penerimaan positif terhadap integrasi AI dalam penggredan esei, sambil menunjukkan kebimbangan terhadap ketepatan dan potensi berat sebelah. Sistem yang dicadangkan dapat mewujudkan platform kepada pensyarah FELC untuk menganalisis esei pelajar dengan cepat, memberikan maklum balas yang lebih peribadi, dan juga menyokong kemajuan pelajar dalam membangunkan kemahiran menulis mereka. Berdasarkan ujian yang dilakukan dengan pensyarah bahasa untuk MUET dan penguji rintis, sistem ini didapati berfungsi dengan baik, dan maklumbalas yang dijana oleh kecerdasan buatan (AI) dianggap amat berguna untuk pembelajaran formatif.

Chapter 1: Introduction

1.1 Introduction

The Faculty of Education, Language and Communication (FELC) at Universiti Malaysia Sarawak (UNIMAS) accommodates a growing number of Pre-University students preparing for the Malaysian University English Test (MUET). The students are required to develop proficient English language skills, particularly in writing essay. The grading process of the essay must be done by experienced lecturers who specialize in marking the MUET examination essay to carefully evaluate each student's work based on specific rubrics to ensure consistency and fairness. The lecturers must minimize discrepancies between individual assessments to maintain grading accuracy and need to provide detailed, timely feedback to the students.

Integrating Natural Language Processing (NLP) technology into the grading process can enhance the teaching and learning experience by allowing for more efficient, consistent, and objective grading. NLP can automate aspects of the evaluation process, helping lecturers quickly analyze student essays which not only reduces lecturer workload but also allows for more personalized feedback, supporting students' progress in developing their writing skills.

1.2 Problem Statement

The lecturers of Faculty of Education, Language and Communication (FELC) in Universiti Malaysia Sarawak (UNIMAS) face challenges in grading essays for Pre-University students preparing for the Malaysian University English Test (MUET). The current manual grading process is both labor-intensive and time-consuming, given the large number of students compare to the limited number of lecturers. As lecturers work to keep up with the grading load, the work can lead to fatigue, affecting grading quality and consistency, which ideally should not vary more than 5% across lecturers. Moreover, the manual grading process limits lecturers' ability to provide detailed, personalized feedback to students on their essays. Due to time constraints and additional responsibilities, lecturers cannot always address individual student weaknesses or offer guidance for improvement. Therefore, this situation hinders students' progress in developing their writing skills.

1.3 Objectives

The objectives of the project are:

1. To develop a web application system that integrates OCR to accurately scan essays.
2. To apply artificial intelligence models to analyze and score essays based on predefined rubrics and provide detailed justification for each assigned score.
3. To evaluate the user satisfaction of the automated essay grading system.

1.4 Methodology

The project will employ the Feature-Driven Development (FDD) methodology, a type of Agile approach known for its flexibility, iterative, and incremental structure, which is particularly suited for developing AI-based web applications that require the systematic delivery of functional features. As shown in Figure 1.1, FDD focuses on iterative development and continuous feedback, enabling quick adjustments to meet user needs and adapt to changing project objectives (Beck et al., 2001). This methodology is especially suitable for the Automated Essay Grading System, as it allows a focused, feature-based approach where specific functions are prioritized and developed incrementally.

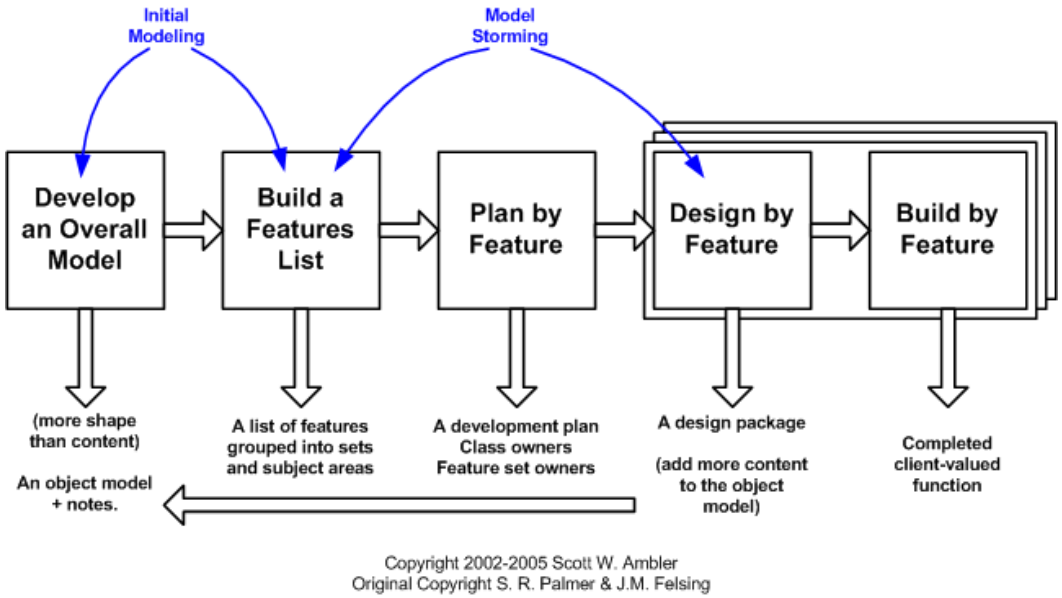


Figure 1.1: Five processes of FDD (Ambler, 2023)

Each feature represents a distinct function of the grading system, from evaluating essays to providing feedback. This approach supports continuous refinement through testing and feedback, a crucial advantage for AI-driven projects. The Feature-Driven Development methodology consists of the following steps:

i. Develop the Overall Model

The initial step involves determining the project's overall scope by creating models that represent the key functionalities of the grading system. The project scope will be outlined with a focus on how AI can assist in grading and providing feedback to students. Several models for different functionalities will be proposed, reviewed and merged to create an overall model that represents the structure and behavior of the final system.

ii. Build the Features List

A comprehensive list of features created based on the project requirements which focus on specific functionalities that will deliver immediate value to lecturers using the system. Each feature is designed to be a small, manageable task that can be completed within a short timeframe, ensuring steady progress throughout development.

iii. Plan by Feature

Each feature from the list will be evaluated, prioritized, and ordered according to its importance and dependency on other features. This ensures essential features are completed first and lays out a roadmap for feature implementation. The features will be assigned to specific developers or team members with relevant expertise based on the priority which allows more efficient use of resources.

iv. Design by Feature

In this phase, a subset of features will be selected for design and implementation. A design package will be created for each selected feature, outlining the technical specifications and user interface elements while ensuring the design package aligns with project goals.

v. Build by Feature

The developer starts building selected features in this phase. Each feature will be coded, integrated into the system, and thoroughly tested to ensure it functions as expected. After initial testing, the feature will be refined based on feedback, and any issues will be resolved before it is integrated into the final system. Testing will include both functionality testing and usability testing.

1.5 Scope

The scope of this project is to address the issues identified in the problem statement by developing an Automated Essay Grading System specifically for UNIMAS language lecturers who teach pre-university students. The system aims to reduce the workload of lecturers by automating the grading of MUET essays and ensuring grading consistency by creating a web-based AI application that evaluates MUET essays and generates detailed feedback aligned with specific grading rubrics. The system will include AI-based grading features capable of assessing various essay components, and a feedback function to highlight areas for improvement, allowing lecturers to offer targeted guidance. The application's user interface (UI) will prioritize usability, providing lecturers with an intuitive experience for grading and feedback generation. The primary users, UNIMAS language lecturers, will access features for grading and generating feedback on student essays. Additionally, user authentication will ensure secure access, enabling lecturers to manage and review system-generated feedback for submitted essays. The Feature-Driven Development (FDD) methodology will guide incremental feature development, emphasizing continuous feedback and refinement.

1.6 Significant of Project

The Automated Essay Grading System aims to ease the workload and improve grading consistency for UNIMAS lecturers that assessing pre-university students' essays as the system provides grading that lecturers can use as a guideline for delivering personalized feedback. This approach minimizes human error and bias, ensuring more objective and consistent grading. Additionally, the system's detailed feedback helps lecturers identify students' understanding levels and areas for improvement, enhancing their exam preparation for tests like MUET. Beyond its practical application, this project contributes to AI research in educational tools, providing insights that could support further developments in automated grading systems.

1.7 Project Schedule

The project schedule for the Automated Essay Grading System is graphically represented in a Gantt chart (refer to Appendix A), detailing all scheduled tasks for planning, organizing, and monitoring each phase of system development. The chart covers the timeline from the initial research and requirement gathering to the final implementation and testing phases.

1.8 Expected Outcome

The expected outcomes of this project include:

- A fully functional web-based automated essay grading system that provides consistent, rubric-based grading.
- A user-friendly interface enabling lecturers to upload essays, view scores, and understand grading justifications.
- An AI-driven system that significantly reduces the grading workload, allowing lecturers to focus on teaching and providing tailored feedback for students.

1.9 Project Outline

Chapter 1: Introduction

This chapter gives an overview of the project. The problem statement, objectives, project scope, methodology, expected outcome, significance of the project, project schedule and project outlines are part of the chapter content.

Chapter 2: Literature Review

Chapter 2 discusses similar research or review of previous research on the system. This section compares existing systems to assess their strengths and weaknesses which is useful for establishing the foundation for the project design.

Chapter 3: Requirement Analysis and Design

This chapter will outline the necessary system requirements, design factors, and architectural structure of the project. This section also discusses the overall process, functions, features and interface of the project which serve as guidelines.

Chapter 4: Implementation and Testing

Chapter 4 discusses in detail the project's technical implementation and testing procedure. This section also focuses on system functionality and user experience.

Chapter 5: Conclusion & Further Work

This chapter summarize the project outcomes, limitations, and recommendations for future enhancements.

Chapter 2: Introduction

2.1 Introduction

Essay grading is a crucial aspect of teaching and learning in pre-university education, particularly for students preparing for the Malaysian University English Test (MUET) at Universiti Malaysia Sarawak (UNIMAS). The lecturers of Faculty of Education, Language and Communication (FELC) in UNIMAS, manually grade essays to evaluate student performance based on predefined rubrics. However, this process is labor-intensive and time-consuming, to avoid inconsistencies due to some degree of variations in grading among lecturers. Thus, Automated Essay Grading System is developed to address these challenges and assist lecturers by leveraging advancements in Natural Language Processing (NLP) and Artificial Intelligence (AI) technologies.

This chapter provides a comprehensive review of existing systems and methods related to essay grading, including the conventional manual grading approach and three automated systems sourced from academic and practical applications such as GitHub repositories. Each reviewed system is analyzed for its objectives, features, strengths, and weaknesses. The systems are compared to identify areas where the proposed Automated Essay Grading System can offer improvements, such as enhanced grading consistency, detailed feedback generation, and a user-friendly interface tailored to UNIMAS language lecturers' needs.

The review includes a comparison table summarizing the specifications and key features of the systems, providing insights into their strengths and limitations. This literature review will serve as a foundation for designing a robust, scalable, and effective essay grading system, aligning with the project’s objectives.

2.2 Review on Similar Existing Systems

This section evaluates four systems relevant to the proposed Automated Essay Grading System, including the conventional manual grading approach and three automated systems sourced from GitHub repositories. The chosen existing system is analyzed for its features, strengths, and limitations to identify gaps that the proposed system aims to address.

2.2.1 Conventional Manual-Based System

Currently, essay grading for pre-university students in Universiti Malaysia Sarawak (UNIMAS) is performed manually by lecturers, following a traditional approach. This process involves lecturers reviewing each essay individually and assigning scores based on predefined rubrics, such as those used for the Malaysian University English Test (MUET). The flowchart in Figure 2.1 illustrates the general workflow for manual essay grading in FELC UNIMAS, starting with the collection of essays from students, followed by individual essay reviews and manual scoring by lecturers. The process includes comparing scores between lecturers to ensure consistency before finalizing grades and providing feedback to students.

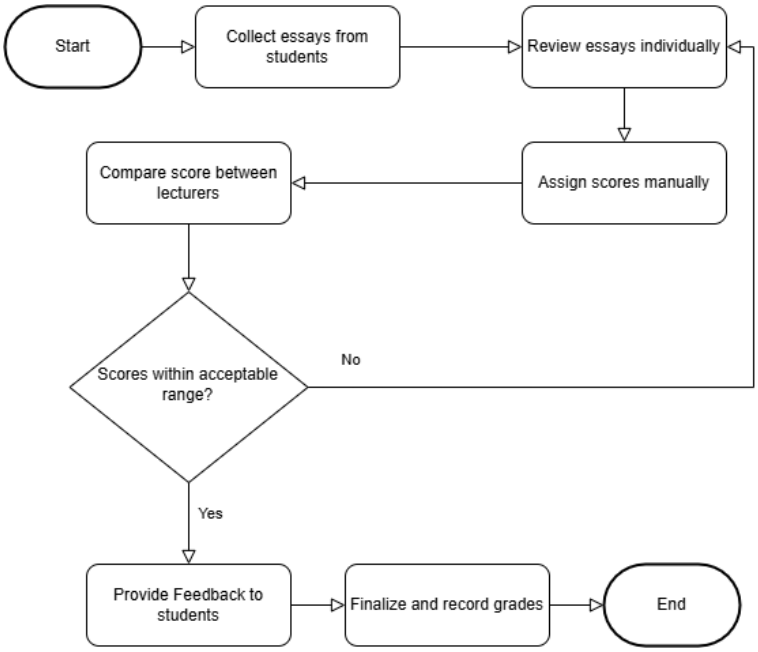


Figure 2.1: Flowchart of conventional manual-based system

In this system, students submit their essays either in physical format or as digital documents via the university submission portal called Unimas eLeaP (E-Learning Enrichment and Advancement Platform). Once collected, lecturers begin the grading process, which includes reading and analyzing the content of each essay, assessing its structure, grammar, vocabulary, and coherence according to the predefined rubric criteria. Scores are manually assigned and compared among lecturers. If the scores provided by different lecturers vary significantly, the essays are reviewed again to resolve inconsistencies. After finalizing the scores, it is then recorded for academic purposes.

The method of the system is a very time-consuming process as grading essays manually is labor-intensive, especially for large class sizes. It also introduces inconsistent grading and prone to errors due to fatigue from grading large volumes of essays and can lead to discrepancies in student scores. Besides, students are given limited feedback as providing detailed and personalized feedback is difficult due to lecturers' time constraints. The system also lack scalability which may become less feasible as student numbers grow, making it unsustainable for larger cohorts.

2.2.2 EssayScore_FYP by tingwei3931 on GitHub

The system was built by Ling Ting Wei associated with the University of Lincoln as a final year project. According to Wei (2019), the EssayScore_FYP system is a web-based application designed to alleviate the grading workload for teachers while offering students instant feedback on their essays. This system is built using Keras with TensorFlow as its backend for deep learning model training, while Flask serves as the web application framework. The users of the system are students. There are four modules in the system which are login module, essay topic selection list module, essay submission module, and essay result detail module.

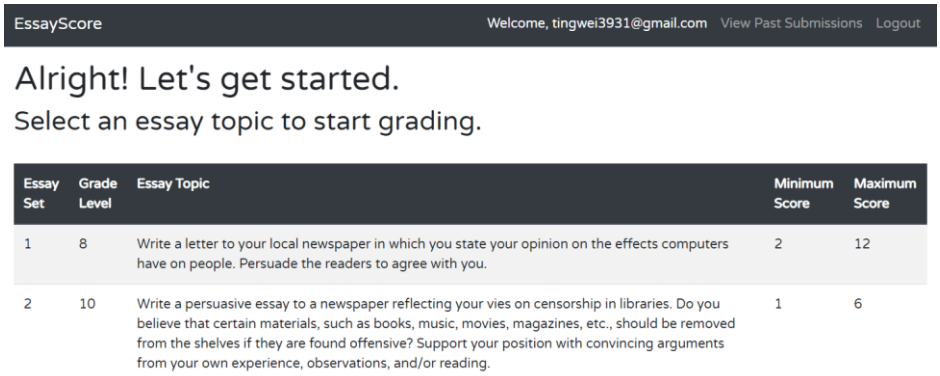


Figure 2.2: Essay Topic Selection Interface

The system EssayScore_FYP includes an essay topic selection user interface to streamline the grading process. This interface allows users to view all of essay information such as Essay List, Grade List, Essay Topic, Minimum Score, Maximum Score. As shown in Figure 2.2, the interface provides a table-based format, enabling lecturers to view and select topics efficiently.

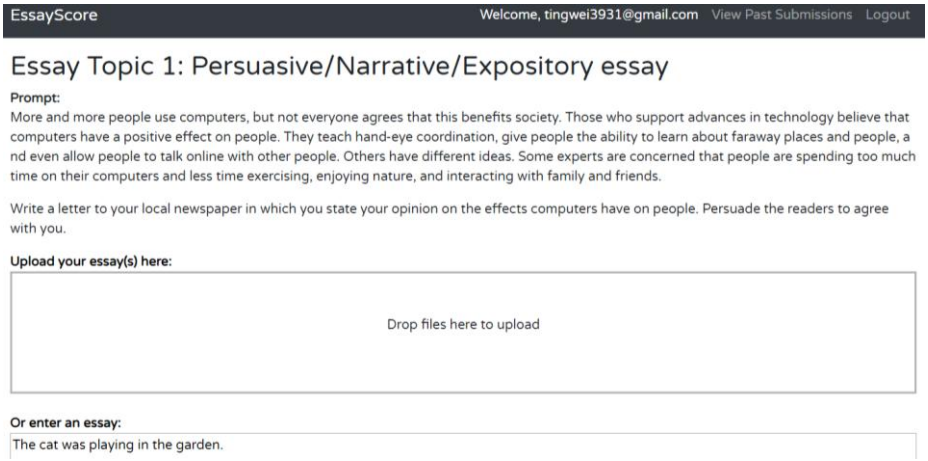


Figure 2.3: Essay Submission Interface

The system includes an essay submission user interface designed for students as shown in Figure 2.3. This interface provides two primary options for essay submission: uploading a digital essay file by dragging and dropping it into the rectangular upload area or typing the essay directly into the provided textbox. The interface accommodates diverse submission preferences, ensuring accessibility and convenience for all users. Additionally, the page shows the essay topic and the question prompt to guide students before submission, enhancing clarity and user experience.

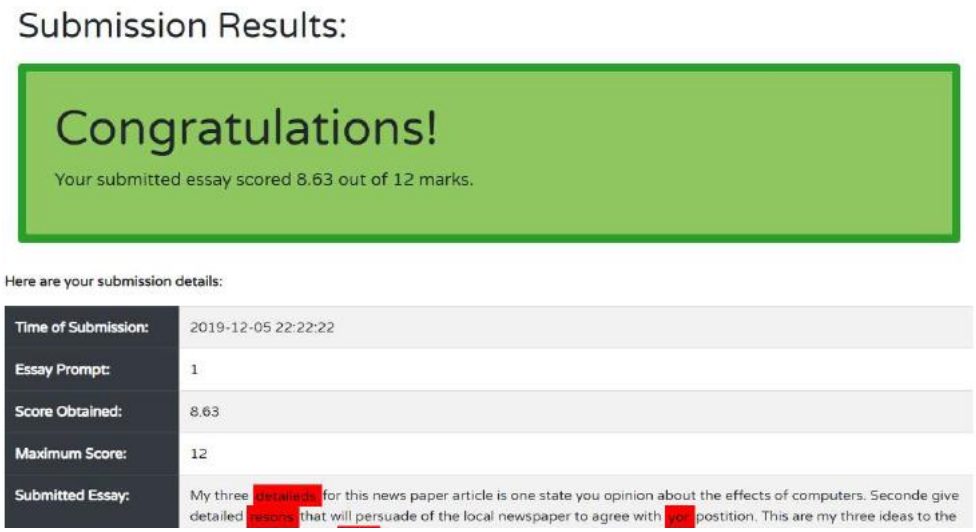


Figure 2.4: Essay Result Detail Interface

The system also includes a submission results detail user interface, which provides detailed feedback to students after submitting their essays. As shown in Figure 2.4, at the top of the page, a congratulatory message is prominently displayed in a green box, showcasing the student's score. The page shows the essay score at the top of the page in a rectangular box, and below the box, it shows the submission details in a table that consists of *Time of Submission, Essay Prompt, Score Obtained, Maximum Score, Submitted Essay.*

The advantages of the system are it allows user to upload in digital format or type in the essay inside a textbox. The system interface also straight forward to use although some of the interface can be improve more to make it more user friendly. The disadvantages of the system are it cannot add new essay topic or essay prompt into the system apart from the predefined essay topic which limits the system customization. The system also does not provide a history page list of submitted essays which makes it challenging for users to revisit previous submissions. While the system provides scores, it lacks detailed explanations or constructive feedback, which are critical for helping students improve specific aspects of their writing.

2.2.3 Automatic-Essay-Scoring by sankalpjain99 on GitHub

The Automatic-Essay-Scoring system, developed by Sankalp Jain, is a tool designed to automate essay evaluation and grading (Jain & Thakkar, 2020). The system processes essays using Word2Vec embeddings and LSTM layers to predict scores based on textual features. It is built using TensorFlow and Keras for model training, and Flask as the web application framework. This project aims to automate the assessment process of essay grading for educators and learners to encourage iterative improvement for students' writings.

The system includes four modules which are data preprocessing, machine learning, neural network application, and the web app. Preprocessing steps involve cleaning essays, normalizing data, and extracting additional features such as word counts, average word lengths, and misspellings. Machine learning algorithms like Random Forest and Support Vector Regression were initially applied, but the best results were achieved using a deep learning approach with LSTM layers. The web application allows users to submit essays through a Flask interface, which then returns predicted scores based on the trained model.

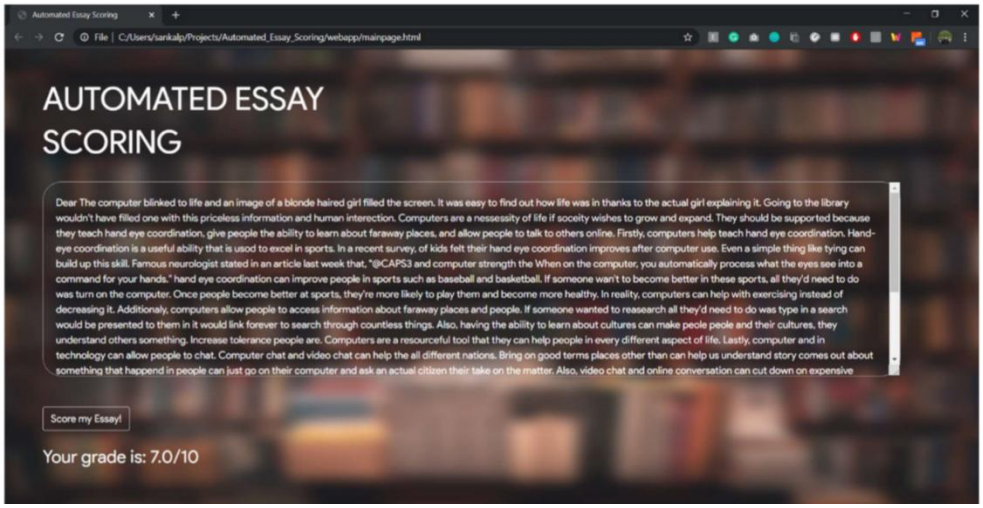


Figure 2.5: Automated Essay Scoring Interface

Figure 2.5 shows the user interface of the system, which prominently features a textbox where users can type or paste their essays for evaluation. Below the textbox is a button labeled "*Score My Essay!*", which initiates the scoring process. Once the essay is submitted, the system automatically assigns a score, which is displayed clearly below the button.

The advantages of the system are it provides immediate scoring. The system interface is very simple and easy to use. The disadvantages of the system are it does not provide a history page list of submitted essays which makes it challenging for users to revisit previous submissions. The system feedback is restricted to a numeric score, without detailed explanations or actionable insights for students to address weaknesses.

2.2.4 Automated-Essay-Grading-with-NLP by AlexEBall on GitHub

The Automated-Essay-Grading-with-NLP system is a machine learning-based project by Alex Edward Ball that is designed to help in aiding teachers with grading student's essays (Ball, 2019). The project experiment with multiple models and uses two trained models in which the performance of one of the models is bad while the other is good.

The system contains five modules which are phrase module, essay module, data preprocessing module, grammatical module, and topic modelling module. These modules collectively enable detailed linguistic analysis and essay scoring. The system integrates sophisticated feature, including linguistic analysis with SpaCy, which extracts grammatical constructs, phrase structures, and vocabulary usage, while Latent Dirichlet Allocation (LDA) is employed for topic modeling to assess thematic consistency in essays.

The advantages of the system include the implementation of multiple machine learning models, such as Random Forest, Support Vector Machines (SVMs), and Linear Classifiers. This allows for the comparison of different algorithms' performance and enables fine-tuning for enhanced scoring accuracy. However, the system has some disadvantages, such as lacking a dedicated user interface for end-users, which limits its accessibility and focuses primarily on backend experimentation. Additionally, the absence of a feedback module restricts its ability to provide explanations or constructive feedback alongside scores to the teachers.

2.3 Comparison of Reviewed Systems

The comparison evaluates the reviewed systems against the proposed Automated Essay Grading System. The benchmark criteria ensure an objective and comprehensive evaluation.

Table 1: Comparison Reviewed System Table

Criteria	Conventional Manual-Based System	EssayScore_FYP	Automatic-Essay-Scoring	Automated-Essay-Grading-with-NLP	Proposed System
Target Users	Lecturers	Students	Students and educators	Educators	MUET language teacher
Essay Submission	Physical or digital upload	Digital upload or textbox	Textbox	None	Digital upload with OCR or textbox
Grading Process	Manual (rubric-based)	Automated using neural network	Automated using machine learning and neural network	Automated using natural language processing	Natural language processing with GPT-4o mini via prompt engineering

Feedback Mechanism	Limited, manual feedback	Numeric score only	Numeric score only	None	Score with rubric-aligned justification (AI-generated)
Customizability	Flexible	Fixed essay topic	Limited	Supports multiple trained models	Fixed to MUET rubric
Scalability	Low	Moderate	High	Moderate	High
Rubric Alignment	MUET's rubrics	Generic	Generic	Undefine	MUET aligned
Usability	No interface	Simple user interface	Basic user interface	Lack user interface	User friendly
Strengths	Personalized evalution	Flexible submission format, fast scoring, provide interface for spelling error detection and correction recommendation	Simple interface, fast scoring	Custom trained model	Detailed feedback, supports images, scalable, educator-centric

Limitations	Labor intensive, inconsistent grading	No submission history list, no feedback justification	No submission history list, no feedback justification, no interface for showing essay mistake	No user interface, Focus on experimentation	Currently limited to MUET essays only

2.4 Brief Overview of the Proposed System

The Automated Essay Grading System is designed to address the challenges faced by UNIMAS language lecturers in evaluating essays for pre-university students preparing for the Malaysian University English Test (MUET). By leveraging advanced technologies, the system aims to automate essay grading while providing detailed and rubric-aligned feedback. The proposed system will be web-based and tailored specifically for UNIMAS lecturers, focusing on enhancing grading efficiency, maintaining consistency, and improving the overall learning process for students.

There are several key features of the proposed system which are automate grading by utilizing Natural Language Processing (NLP) and Artificial Intelligence (AI) to evaluate essays against MUET-specific rubrics. The system can give detailed feedback generation for lecturers by providing rubric-aligned feedback to help lecturers highlight strengths and weaknesses in student writing. The system is also designed to have a user-friendly interface, enables lecturers to upload essays digitally and review results thorough an intuitive web application. The system builds for scalability in mind to handle large number of essay submissions, ensuring it is suitable for larger cohorts.

2.5 Technological Review

The proposed system will leverage widely used and reliable technologies to ensure robustness, scalability, and efficiency of the system. This will ensure the system is easy to maintain and does not use out-of-date technologies. There are several technologies that are planned to be used inside the proposed system.

a) OCR for Essay Digitization
    i. Tesseract OCR is used to convert scanned essay images into text format for analysis. It is also able to handle handwritten or printed essays and can be integrated with Django as it is lightweight.
    ii. Azure AI Vision also offers OCR services to extract printed and handwritten text from images.

b) Natural Language Processing for Text Analysis
    i. SpaCY is used as the NLP for text analysis as it offers more efficient and accurate tokenization, POS tagging, dependency parsing, and Named Entity Recognition (NER).
    ii. Azure AI Document Intelligence also can be used to extract text, key-value pairs, tables, and structures from documents for text analysis.

c) Artificial Intelligence for Essay Scoring
    i. OpenAI API can be used to evaluate essays based on the rubric criteria and assign scores. It also can be used to provide insight into semantic and contextual coherence.

d) Web Framework

   i. The system will use Django for developing a robust web application. It provides built-in support for creating APIs, managing databases, and building user interfaces, ensuring seamless integration with other components like MySQL and external APIs

   ii. Azure App Service also can be used to build and host web apps, mobile backends, and RESTful APIs in any programming language without managing infrastructure.

e) Database

   i. The system will use MySQL as Django supports MySQL as a backend database, making it a suitable choice for storing essay data, user information, and grading results. MySQL is also widely used for web applications.

   ii. Alternatively, Azure SQL Database can be used to fully manage relational databases and is compatible with MySQL.

f) Hosting

   i. Hostinger is a commonly used hosting platform as it provides a reliable and cost-effective hosting environment, supporting Python-based web frameworks like Django. It has VPS or Cloud Hosting plans that are ideal for hosting Django applications and connecting with MySQL databases.

   ii. Azure App Service also can be used to provide hosting capabilities.

2.6 Summary

The literature review examined four systems relevant to the proposed Automated Essay Grading System which are the conventional manual-based grading process, and three automated essay grading systems sourced from GitHub repositories. The review analyzed these systems based on their features, strengths, and limitations to identify gaps and opportunities for improvement.

The conventional manual-based system, while flexible and personalized, suffers from scalability issues, inconsistent grading, and the inability to provide timely, detailed feedback due to its labor-intensive nature. It highlights the need for automation in grading processes to reduce workload and improve consistency.

The EssayScore_FYP system simplifies essay submission and scoring for students using a deep learning model. However, it lacks customization options, historical tracking of submitted essays, and detailed feedback, limiting its utility for academic grading purposes.

The Automatic-Essay-Scoring system incorporates advanced NLP and neural networks, offering immediate scoring and a scalable backend. However, its feedback is limited to numeric scores, and the absence of submission tracking reduces its practical usability for lecturers or institutions.

The Automated-Essay-Grading-with-NLP system demonstrates advanced feature extraction, supporting multiple machine learning models for experimentation. However, it focuses primarily on backend development and lacks a user-friendly interface or detailed feedback mechanisms for students.

The insights from these systems underscore the gaps that the proposed Automated Essay Grading System aims to address, particularly in providing detailed, rubric-aligned feedback, enhancing usability for lecturers, and ensuring scalability while maintaining grading consistency. This review forms a foundation for designing a robust, effective solution targeted to the needs of UNIMAS lecturers.

Chapter 3: Requirement Analysis and Design

3.1 Introduction

This chapter provides a detailed analysis and design process used for the project creation, focusing on defining the requirements and creating a detailed system documentation. It includes the methodologies applied, the system and user requirements, and the design elements such as activity, use case, sequence, and class diagrams. The chapter also describes the hardware and software specifications necessary for implementation and introduces the proposed system prototype.

3.2 Develop an Overall Model Process

This process is the initial step in the FDD methodology, where the project scope and system functionality are outlined through comprehensive modeling. For this project, the Automated Essay Grading System, this phase focuses on gathering and analyzing requirements to design a high-level model of the grading system.

3.2.1 Analysis on Proposed System

The method used to gather requirements for the purposed system is conducting interview. The interview is conducted with one of the Faculty of Education, Language and Communication (FELC) lecturers at Universiti Malaysia Sarawak (UNIMAS), Miss Feona Anak Albert, who is one of the examiners had graded Pre-University students' Malaysian University English Test (MUET).

Based on the interview conducted with Miss Feona, the user requirements and functionalities for the proposed system can be summarized as below:

- Good user interface
- Can upload students' essay easily
- Able to adapt to new rubrics
- Give score on students' essay based on predefined rubrics criteria
- Able to give justification on the given score.
- View the list of submitted essays sent by the lecturer.

In addition, a survey was distributed to both lecturers and students via Google Forms to gather insights on the proposed Automated Essay Grading System. The survey was structured into four sections, covering demographic information, awareness and perception of AI in essay grading, prototype feedback, and adoption/improvement suggestions. A total of eight respondents participated, and their feedback which attached in the appendices, provided valuable information for the development of the system.

The key insights from the survey can be summarized as follows:

1. **Demographic**

   Many respondents were either lecturers or students, providing a balanced perspective from both educators and learners. The respondents included individuals with varying years of experience in education, with 62.5% working in the education sector for less than 5 years, and 37.5% working in the education sector for 5-10 years, ensuring diverse viewpoints.

2. **Awareness and Perception of AI in Essay Grading**

   25% of the respondents knows the existence of automated essay grading systems, though specific tools were not widely recognized. In general, the opinions on integrating AI into MUET essay grading were positive, with many believing it could improve consistency and efficiency, but concerns were raised about fairness, accuracy, and potential biases in AI grading systems.

3. **Prototype Feedback**

   The prototype was generally rated as easy to navigate, with some respondents highlighting features such as score justification and rubric-based grading as most useful. Most respondents found the system user-friendly, though some suggestions for

improvement included better transparency of grading criteria and integration of additional features.

4. **Adoption and Improvement**

   The respondents expressed a willingness to use the system if fully developed, citing its potential to reduce grading workload and improve efficiency. The concerns were noted about ethical considerations, such as bias, as well as challenges in implementing the system institutionally.

In summary, the respondents highlighted several areas for improvement in the system, including enhancing the user interface for better accessibility and simplifying the navigation experience. They recommended incorporating additional features such as rubric customization to accommodate various grading criteria and detailed breakdowns for clarity. Some concerns were also raised about ensuring the accuracy of AI-generated scores. Overall, the feedback emphasized the importance of maintaining transparency and providing actionable feedback to enhance the utility and reliability of the Automated Essay Grading System

3.2.2 Design of Proposed System

In this section, the proposed system design will be shown in logical and physical design. The logical design will be represented by an activity diagram, use case diagram and use case specification, class diagram and sequence diagram. The physical design will be represented in prototype frame design using Figma.

3.2.2.1 Activity Diagram

The figures 3.1 shows the flow of activity and the dynamic behavior of the purposed system. First, the lecturer needs to login into the system or sign up if they do not have an account. They will be greeted with the system dashboard after login, which shows the recently graded essay score, total graded essay numbers, the average score of all graded essays and the pending task. The user which is lecturer can upload a new essay to be graded by the system. After the process is done, the user can view the checked essay lists and view the submitted essay result followed by the score breakdown. The users can manage their account whether they want to update or delete the account.

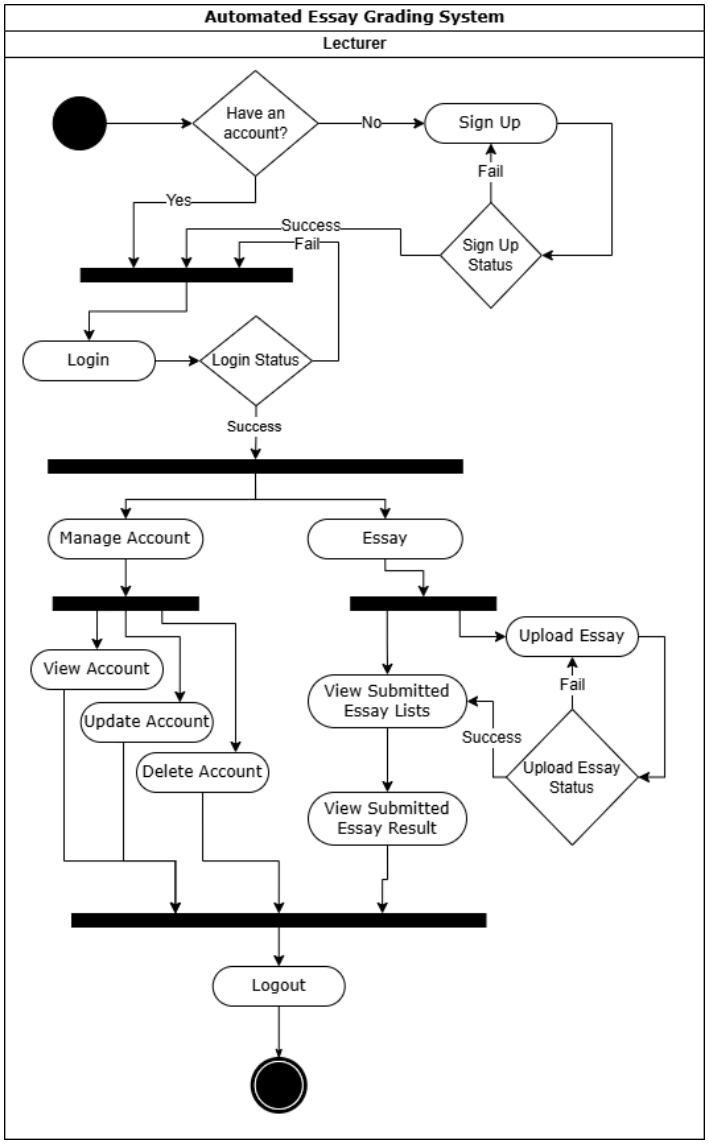


Figure 3.1: Automated Essay Grading System Activity Diagram

3.2.2.2 Use Case Diagram

Figure 3.2 shows the use case diagram for the proposed system. It consists of one actor which is lecturer. There are 11 number of use cases identified, and the details of the use cases are explained in use case specifications below.

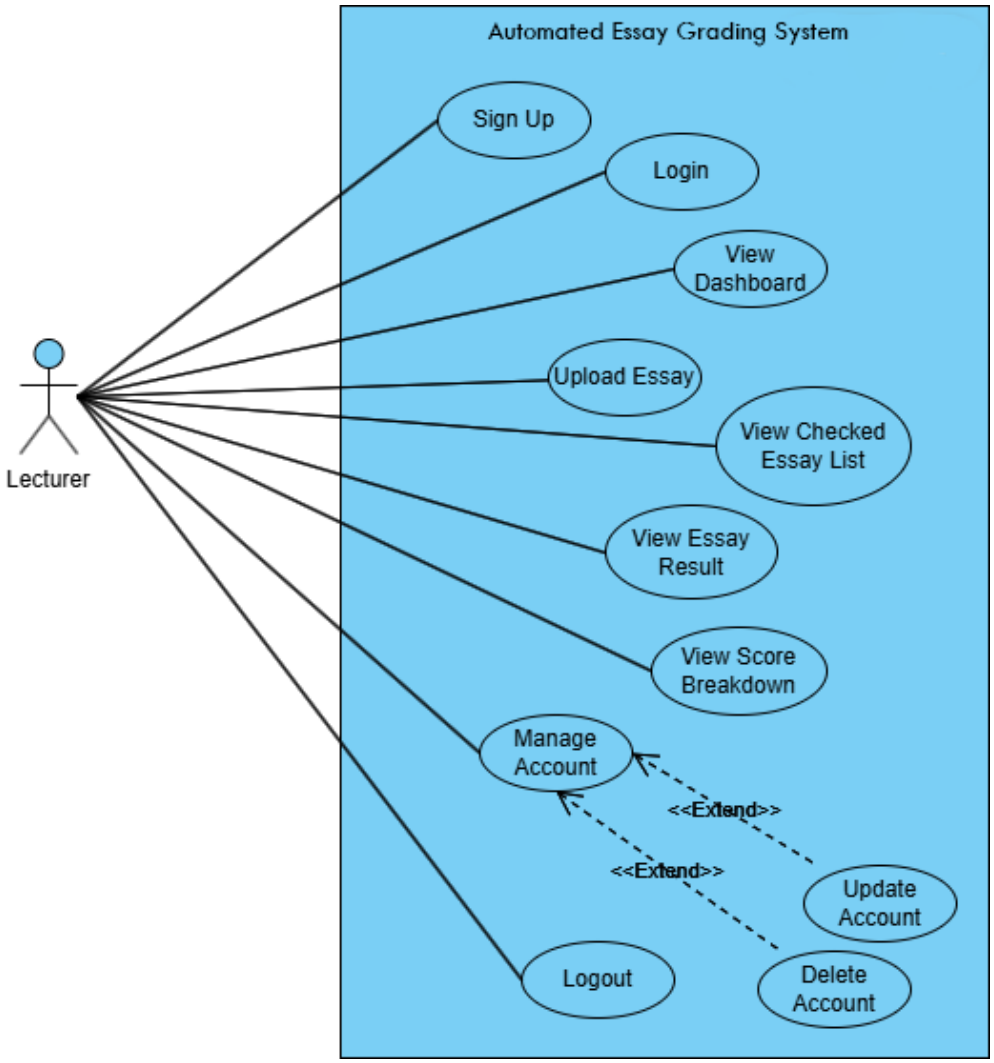


Figure 3.2: Automated Essay Grading System Use Case Diagram

3.2.2.3 Use Case Specification

Table 2: Use Case Specification for Sign Up

Use Case No	AEGS_UC1
Use Case Name	Sign Up
Actor	Lecturer
Brief Description	Allow new lecturers to create an account in the Automated Essay Grading System
Pre-Condition	1. Lecturer has access to the system. 2. Lecturer provides valid registrations details.
Main Flow	1. Lecturer on the system landing page 2. Lecturer clicks the "Start" button (A1). 3. System take lecturer to "Create Account" page 4. Lecturer fills in personal details (E1). 5. Lecturer submits the registration by clicking the "Sign Up" button. 6. System validates the details and confirm the registrations.
Exceptional Flow	**E1: Lecturer cancels sign up** 1. Lecturer chooses not to further sign up for the system. 2. Lecturer clicks the "X" button at the top of the "Create Account" frame. 3. System redirects to the landing page.
Alternative Flow	**A1: Admin registers the lecturer.** 1. Admin enters lecturer details into the system.

	2. System sends an email to the lecturer to set their password and activate the account.
Post-Condition	A new account is created and activated

Table 3: Use Case Specification for Login

Use Case No	AEGS_UC2
Use Case Name	Login
Actor	Lecturer
Brief Description	Allow lecturer to log in into the system using their credentials
Pre-Condition	1. Lecturer has valid account.
Main Flow	1. Lecturer on the system landing page 2. Lecturer clicks the “Login” button 3. System take lecturer to “Login” page 4. Lecturer enters login credentials and click “Sign In” button (E1). 5. Lecturer is redirected to the dashboard.
Exceptional Flow	**E1: Lecturer enters incorrect credentials.** 1. System displays a login error message. 2. Lecturer cannot log in and is asked to enter valid credentials.
Post-Condition	Lecturer gains access to the system dashboard.

Table 4: Use Case Specification for View Dashboard

Use Case No	AEGS_UC3
Use Case Name	View Dashboard
Actor	Lecturer
Brief Description	Display an overview of relevant information of essay grading activities.
Pre-Condition	1. Lecturer is logged in.
Main Flow	1. Lecturer navigates to the dashboard. 2. System display information on recent graded essay score, total graded essay number, average essay score, pending graded essay and a graph of total graded essay by month.
Post-Condition	Lecturer can access the system dashboard.

Table 5: Use Case Specification for Upload Essay

Use Case No	AEGS_UC4
Use Case Name	Upload Essay
Actor	Lecturer
Brief Description	Allow lecturer to upload essay for grading.
Pre-Condition	1. Lecturer is logged in. 2. Essay file is ready for upload or type into system.
Main Flow	1. Lecturer clicks on “Upload Essay” from the sidebar. 2. System take lecturer to “Upload Essay” page. 3. Lecturer uploads the essay file into the provided box (A1). 4. System confirms the successful upload.

Alternative Flow	**A1: Lecturer type in the essay.** 1. Lecturer clicks on “Upload Essay” from the sidebar. 2. System take lecturer to “Upload Essay” page. 3. Lecturer type in the essay into the provided box. 4. System confirms the successful upload.
Post-Condition	The essay is successfully uploaded to the system.

Table 6: Use Case Specification for View Checked Essay List

Use Case No	AEGS_UC5
Use Case Name	View Checked Essay List
Actor	Lecturer
Brief Description	Allow lecturer to view a list of essays that have been graded.
Pre-Condition	1. Lecturer is logged in.
Main Flow	1. Lecturer clicks on “Essay List” from the sidebar. 2. System take lecturer to “Essay List” page. 3. System displays all graded essays (E1).
Exception Flow	**E1: No essays graded yet** 1. System displays a message indicating no graded essays.
Post-Condition	Lecturer accesses the list of graded essays.

Table 7: Use Case Specification for View Essay Result

Use Case No	AEGS_UC6
Use Case Name	View Essay Result
Actor	Lecturer
Brief Description	Allow lecturer to view the grading result for an essay.
Pre-Condition	1. Lecturer is logged in. 2. Essay has been graded by the system. 3. Lecturer is on “Essay List” page.
Main Flow	1. Lecturer clicks on a submitted essay (E1). 2. System take lecturer to “Essay Result” page of the essay. 3. System displays all the information of essay, including the students name, remarks, submission date, essay mark, and the essay content.
Exceptional Flow	**E1: No information being display** 1. System displays error message. 2. Lecturer cannot view essay result. 3. Lecturer can retry accessing the essay list or contact support for assistance.
Post-Condition	Lecturer accesses the grading result.

Table 8: Use Case Specification for View Score Breakdown

Use Case No	AEGS_UC7
Use Case Name	View Score Breakdown
Actor	Lecturer
Brief Description	Displays the detailed score breakdown based on rubric criteria with AI-generated justifications on the given score.
Pre-Condition	1. Lecturer is logged in. 2. Essay has been graded by the system. 3. Lecturer has click on a submitted essay. 4. Lecturer is on “Essay Result” page.
Main Flow	1. Lecturer clicks the “Score Breakdown” button. 2. System take lecturer to “Score Breakdown” page 3. System displays information on the essay breakdown, which includes individual scores for the essay criteria and AI-generated justifications for the scores given (E1).
Exceptional Flow	**E1: No information being display** 1. System displays error message. 2. Lecturer cannot view the essays' score breakdown 3. Lecturer can retry accessing the essay list or contact support for assistance.
Post-Condition	Lecturer accesses the essays’ score breakdown.

Table 9: Use Case Specification for Manage Account

Use Case No	AEGS_UC8
Use Case Name	Manage Account
Actor	Lecturer
Brief Description	Allow lecturers to manage their account.
Pre-Condition	1. Lecturer is logged in.
Main Flow	1. Lecturer clicks on “Account” from the sidebar. 2. System take lecturer to “Account” page. 3. System displays the lecturer's account information (E1).
Exceptional Flow	**E1: No information being display** 1. System displays an error message. 2. Lecturer cannot view account details. 3. Lecturer can retry accessing the page or contact support for assistance.
Post-Condition	1. Lecturer accesses the account information page.

Table 10: Use Case Specification for Update Account

Use Case No	AEGS_UC9
Use Case Name	Update Account
Actor	Lecturer
Brief Description	Allow lecturers to update their personal information
Pre-Condition	1. Lecturer is logged in. 2. Lecturer is on “Account” page.
Main Flow	1. Lecturer clicks on “Edit” button. 2. System takes lecturer to “Edit Account” page. 3. System displays editable fields. 4. Lecturer updates the required information (E1). 5. System validates and saves changes.
Exception Flow	**E1: Lecturer enters invalid data** 1. System displays error message to prompt lecturer to correct the error.
Post-Condition	Account information is successfully updated.

Table 11: Use Case Specification for Delete Account

Use Case No	AEGS_UC10
Use Case Name	Delete Account
Actor	Lecturer
Brief Description	Allow lecturers to delete their account permanently.

Pre-Condition	1. Lecturer is logged in. 2. Lecturer is on “Account” page.
Main Flow	1. Lecturer clicks on “Delete Account” button. 2. System prompts for confirmation (A1). 3. Lecturer confirms account deletion. 4. System removes the account and logs the lecturer out.
Alternative Flow	**A1: Lecturer cancels the deletion request.** 1. System returns to the “Account” page.
Post-Condition	Account information is successfully updated.

Table 12: Use Case Specification for Logout

Use Case No	AEGS_UC11
Use Case Name	Logout
Actor	Lecturer
Brief Description	Allow lecturers to log out of the system.
Pre-Condition	1. Lecturer is logged in.
Main Flow	1. Lecturer clicks on “Logout” from the sidebar. 2. System logs the lecturer out. 3. System redirects the lecturer to the landing page.
Post-Condition	Lecturer is logged out and redirect to the landing page.

3.2.2.4 Sequence Diagram

A sequence diagram gives a visual representation of the interactions between system objects as it can capture on how an actor, the objects, and the components interact during the execution of use case. The sequence diagram in vertical dimension represents time and in horizontal represents the different objects. Figure 3.3 show the sequence diagram in which the actor is shown corresponding to the use case scenario in the sequence diagram. The “System Controller” refers to the backend logic layer responsible for managing communication between the user interface, OCR module, AI model, and database. Specifically, the System Controller handles receiving user input (essay text or image uploads), coordinating OCR processing using Azure Document Intelligence, submitting extracted essay content to the GPT-4o mini model, receiving and parsing scoring responses, saving the AI-generated scores and justification to the database, and returning results to the frontend for display

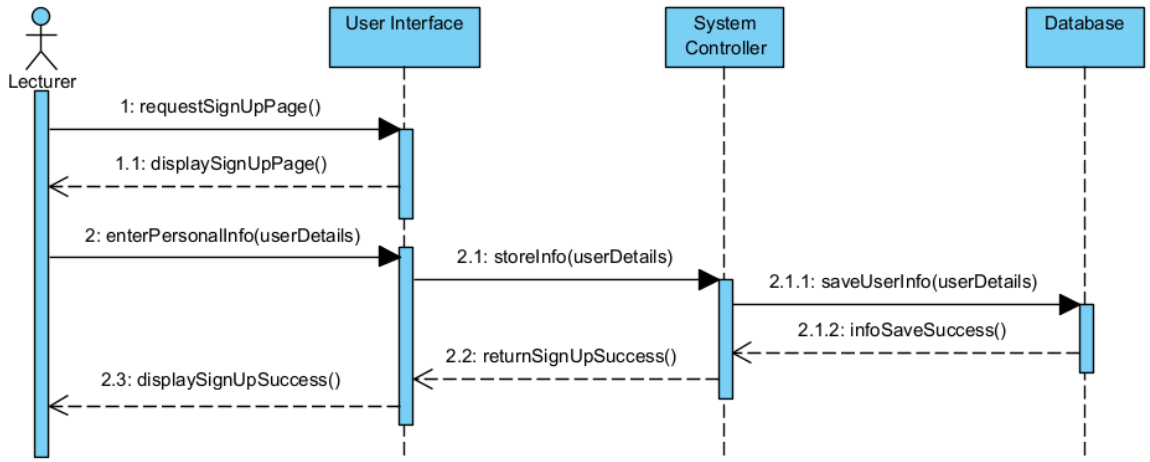


Figure 3.3: Sequence Diagram for Sign Up

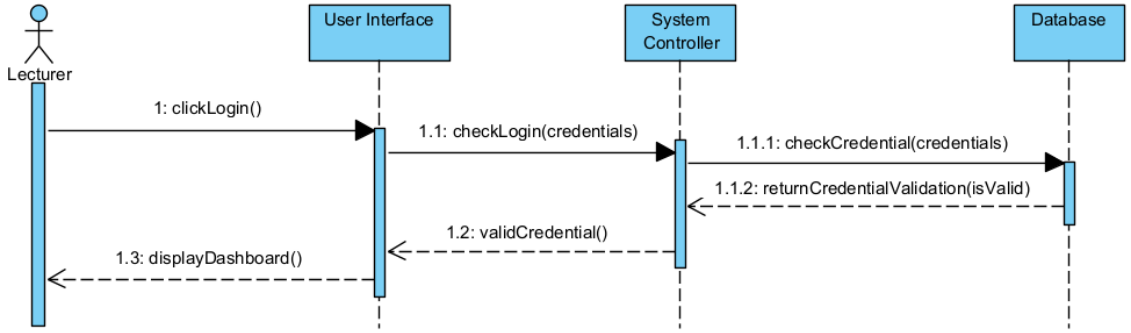


Figure 3.4: Sequence Diagram for Login

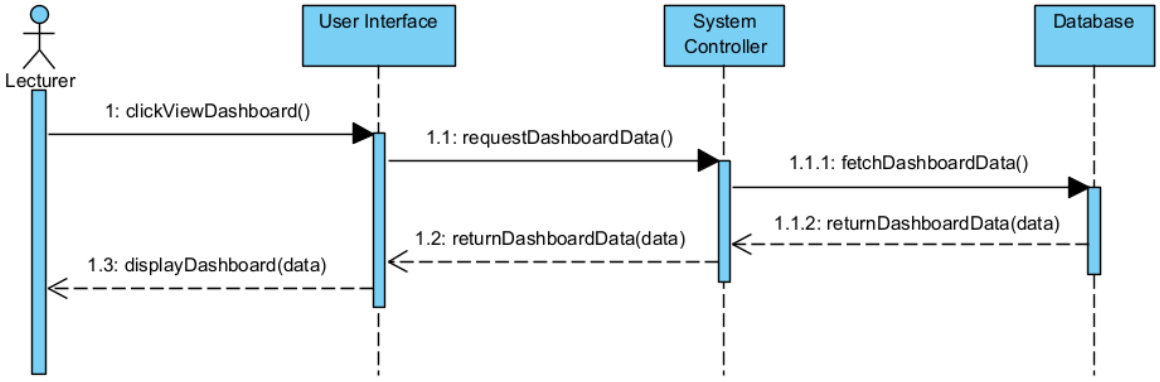


Figure 3.5: Sequence Diagram for View Dashboard

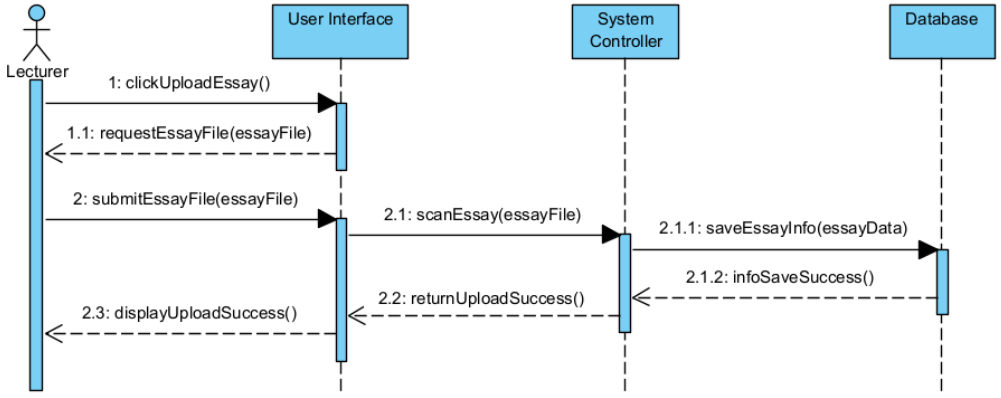


Figure 3.6: Sequence Diagram for Upload Essay

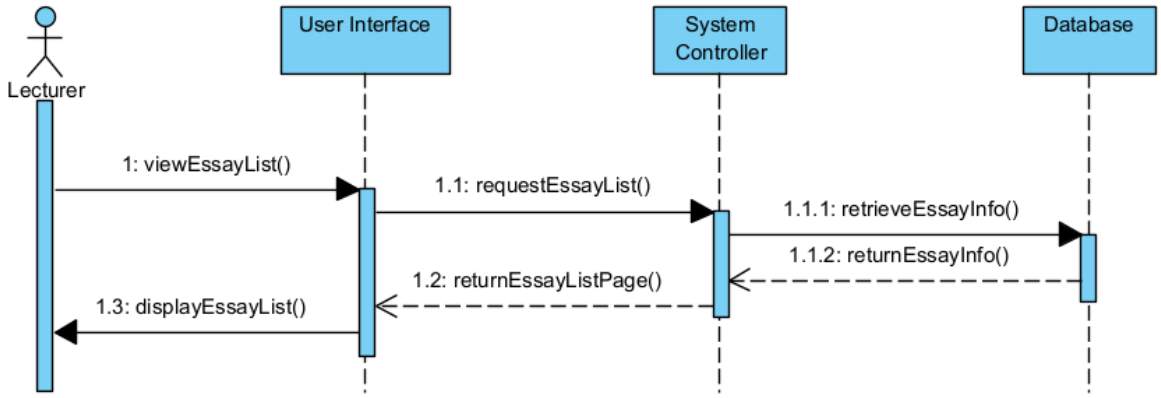


Figure 3.7: Sequence Diagram for View Checked Essay List

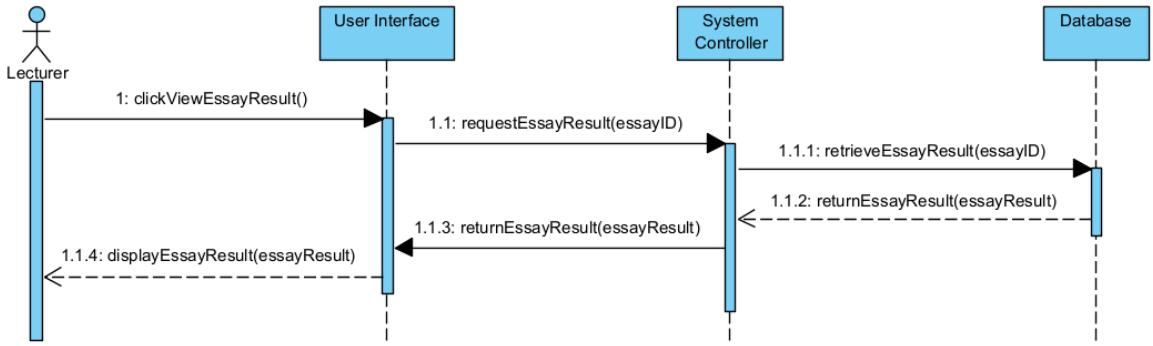


Figure 3.8: Sequence Diagram for View Essay Result

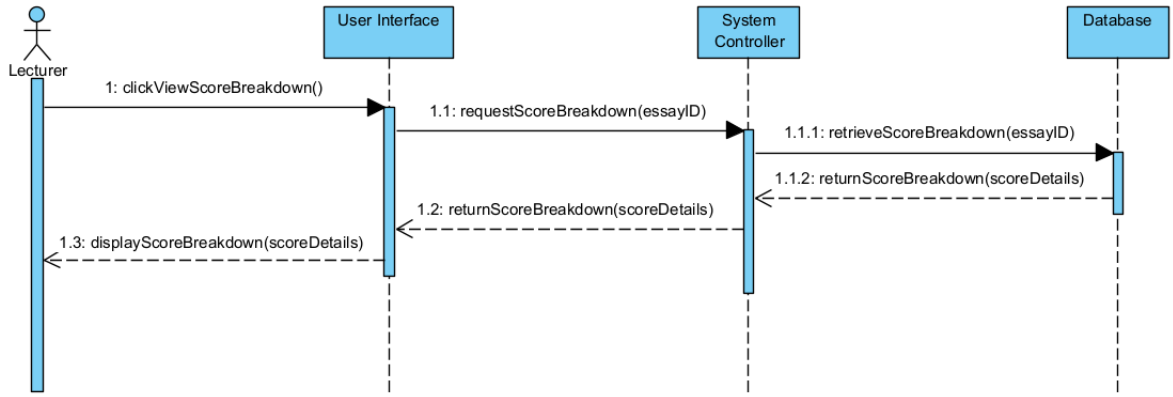


Figure 3.9: Sequence Diagram for View Score Break Down

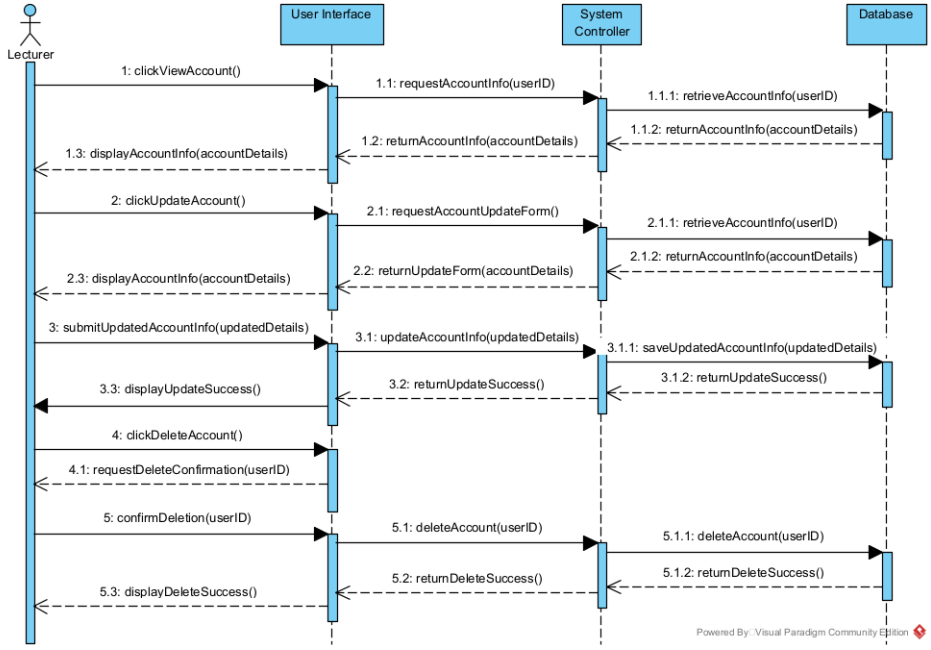


Figure 3.10: Sequence Diagram for Manage Account

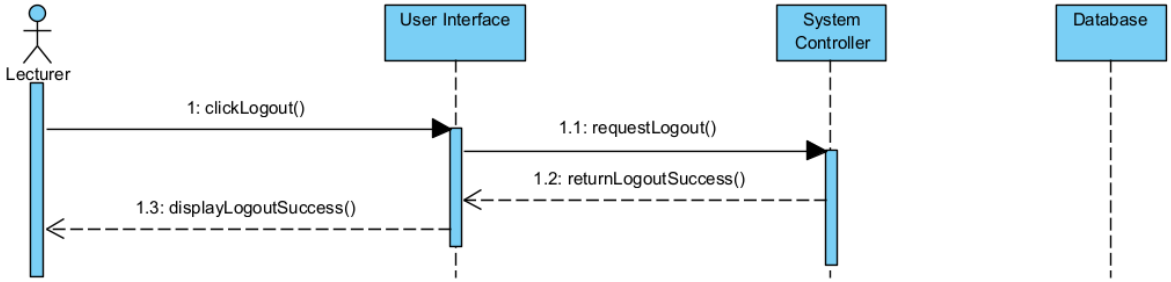


Figure 3.11: Sequence Diagram for Logout

3.2.2.5 Class Diagram

The class diagram below is the representation of the proposed system structure.

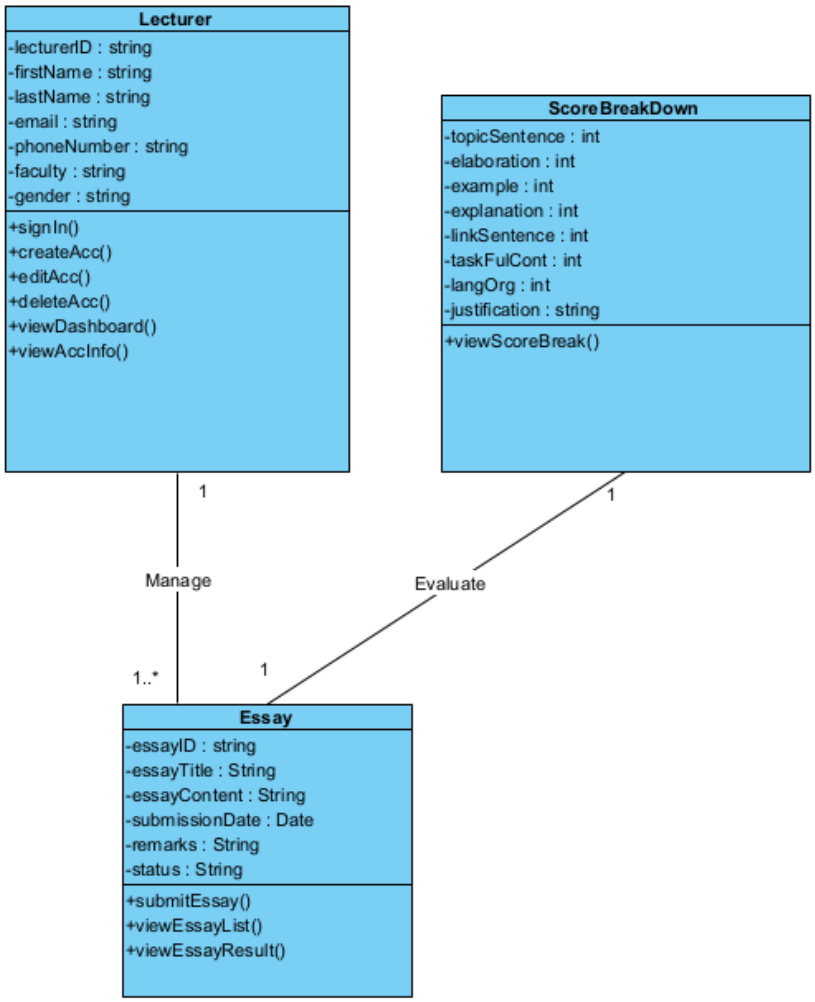


Figure 3.12: Automated Essay Grading System Class Diagram

3.2.2.6 Entity Relationship Diagram (ERD)

The system architecture of the Automated Essay Grading System (AEGS) incorporates a structured relational database designed using Django ORM. This database supports secure and organized storage of lecturer data, essay submissions, OCR images, and AI-generated scoring components. The Entity Relationship Diagram (ERD) illustrates the logical structure and relationships among these core entities, as shown in Figure X. The key entities and their relationships are described as follows:

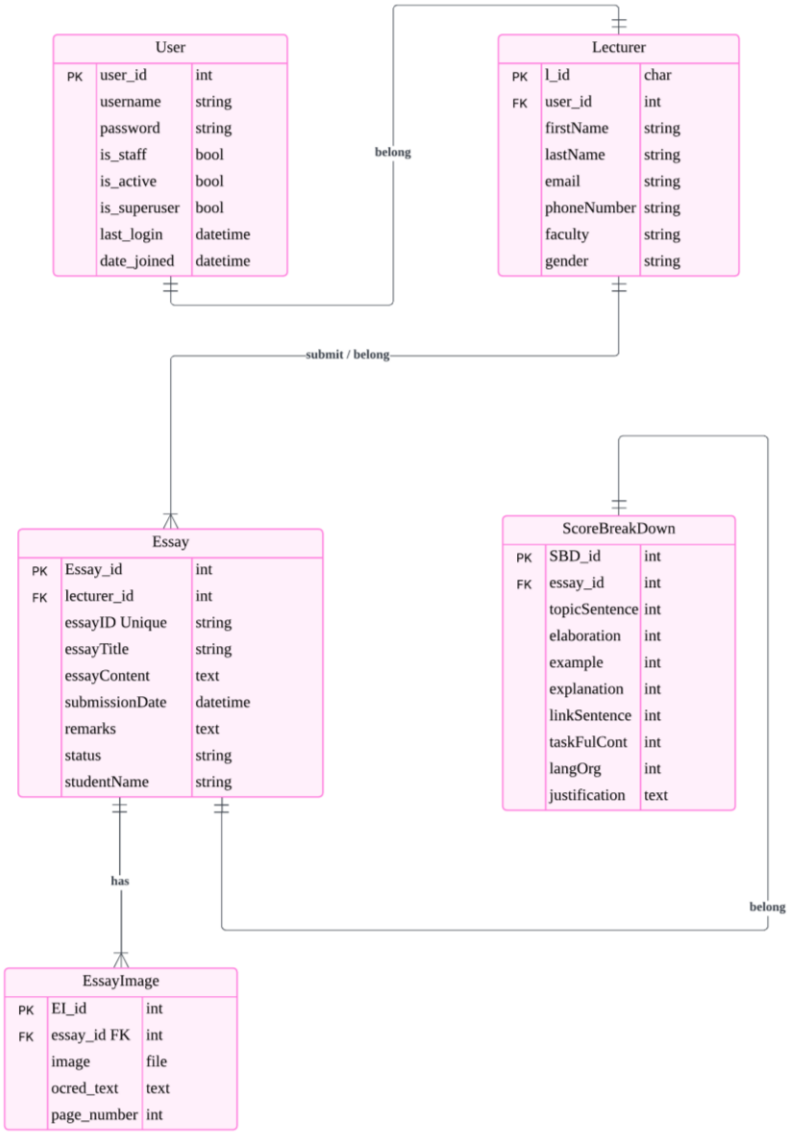


Figure 3.13: Entity Relationship Diagram for Automated Essay Grading System

- Lecturer
    - Each “Lecturer” record is uniquely linked to a “User” account via a one-to-one relationship, ensuring seamless integration with Django’s built-in authentication system.
    - Fields include “lecturerID”, “firstName”, “lastName”, “email”, “phoneNumber”, “faculty”, and “gender”.
    - A “Lecturer” can submit multiple essays, establishing a one-to-many relationship with the “Essay” entity.
- Essay
    - This entity captures individual essay submissions, including fields like “essayID”, “essayTitle”, “essayContent”, “submissionDate”, “remarks”, “status”, and “studentName”.
    - Each “Essay” is linked to a single “Lecturer”.
    - It also maintains a one-to-many relationship with “EssayImage”, and a one-to-one relationship with “ScoreBreakDown”.
- EssayImage
    - This entity stores scanned images of essay pages. Fields include “image”, “ocred_text” (OCR result), and “page_number”.
    - Each image is associated with one “Essay”, and essays can have multiple images.
    - The default ordering by “page_number” ensures correct sequential processing during OCR.

- ScoreBreakDown
    - Represents detailed rubric-based scoring.
    - Contains individual scoring components such as “topicSentence”, “elaboration”, “example”, “explanation”, “linkSentence”, “taskFulCont”, and “langOrg”, along with a “justification” field.
    - This model is one-to-one related to the “Essay”, meaning each essay has exactly one set of scores.

3.2.2.7 Functional Requirements

In this system, there is only one user of the system which is lecturer. The user has their own functionalities which is discussed as below.

Lecturer

- Lecturer is able to log in into the system.
- Lecturer is able to sign up into the system.
- Lecturer is able to upload essay into the system.
- Lecturer is able to view submitted essay list.
- Lecturer is able to view individual essay score and score breakdown.
- Lecturer is able to manage their account.

3.2.2.8 Non-Functional Requirements

Availability

- The system shall be available and accessible anywhere, provided the user has an internet connection.
- The system shall maintain a 99% uptime to ensure high availability during peak hours.

Security

- The system shall restrict access to people other than user that have been sign up in the system.
- Passwords must contain at least 8 characters, including at least one uppercase letter, one lowercase letter, one number, and one special character.

Data Handling

- The system shall validate all inputs to handle invalid data entries gracefully without crashing.
- All errors shall be accompanied by user-friendly error messages that guide users on corrective actions.
- Uploaded essays shall support file formats such as .docx and .pdf

Standard Compliance

The system's graphical user interfaces shall maintain a consistent look and feel across all pages.

3.2.3 Build a Features List

A comprehensive list of features was created based on the user requirements. Each feature is broken into manageable tasks for incremental development. The list of features includes:

1. User Authentication
    a. Login
    b. Sign Up
2. Essay Management
    a. Upload Essay
    b. View Submitted Essay List
    c. View Individual Essay Result
    d. View Essay Score Breakdown
3. Account Management
    a. Manage Lecturer Account

3.2.4 Plan by Features

Each feature is evaluated, prioritized, and scheduled for development based on importance and interdependencies. The planning process ensures critical functionalities are developed first while maintaining efficient resource allocation.

Table 13: Plan by Features

User Authentication	
Feature Name	Login
Description	Allows lecturers to log in to the system using their credentials.
Dependencies	None
Priority	High
Feature Name	Sign Up
Description	Allows new lecturers to create an account in the system.
Dependencies	None
Priority	High
Essay Management	
Feature Name	Upload Essay
Description	Enables lecturers to upload student essays into the system.
Dependencies	User Authentication
Priority	High
Feature Name	View Submitted Essay List
Description	Allows lecturers to view a list of all submitted essays.

Dependencies	Upload Essay
Priority	Medium
Feature Name	View Individual Essay Result
Description	Enables lecturers to view the results of individual essays.
Dependencies	View Submitted Essay List
Priority	Medium
Feature Name	View Essay Score Breakdown
Description	Allows lecturers to see a detailed breakdown of the essay scores.
Dependencies	View Individual Essay Result
Priority	Medium
Account Management	
Feature Name	Manage Account
Description	Allows lecturers to manage their account details.
Dependencies	User Authentication
Priority	Medium

3.2.5 Design by Features

For each prioritized feature, a design package is created, including:

3.2.5.1 User Interface Design

The following is the prototype designed for the proposed system.

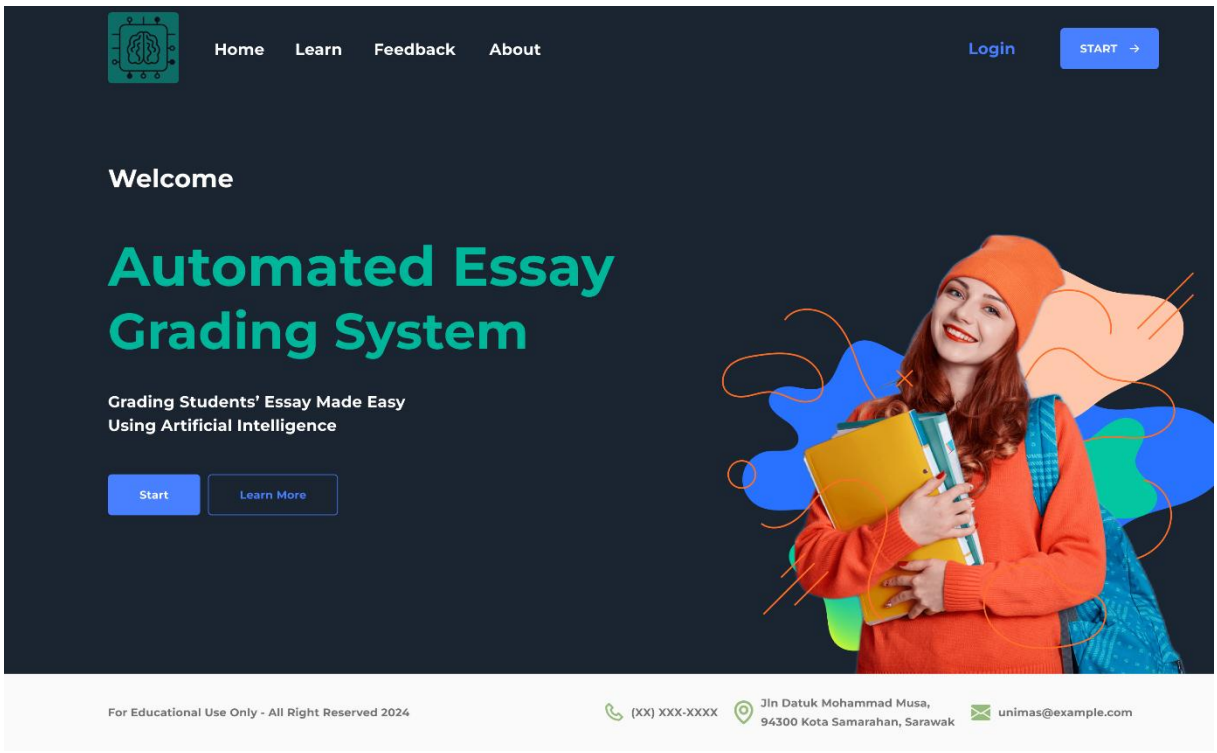


Figure 3.14: Landing page for Automated Essay Grading System

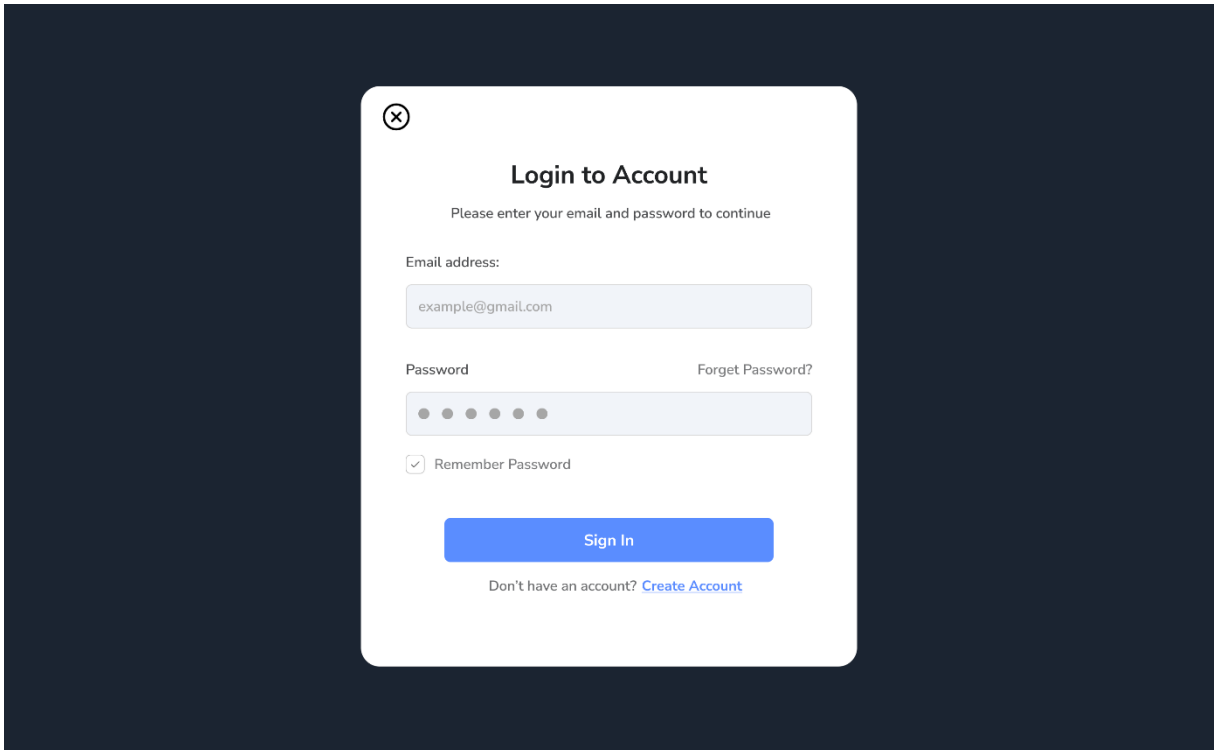


Figure 3.15: Login page for Automated Essay Grading System

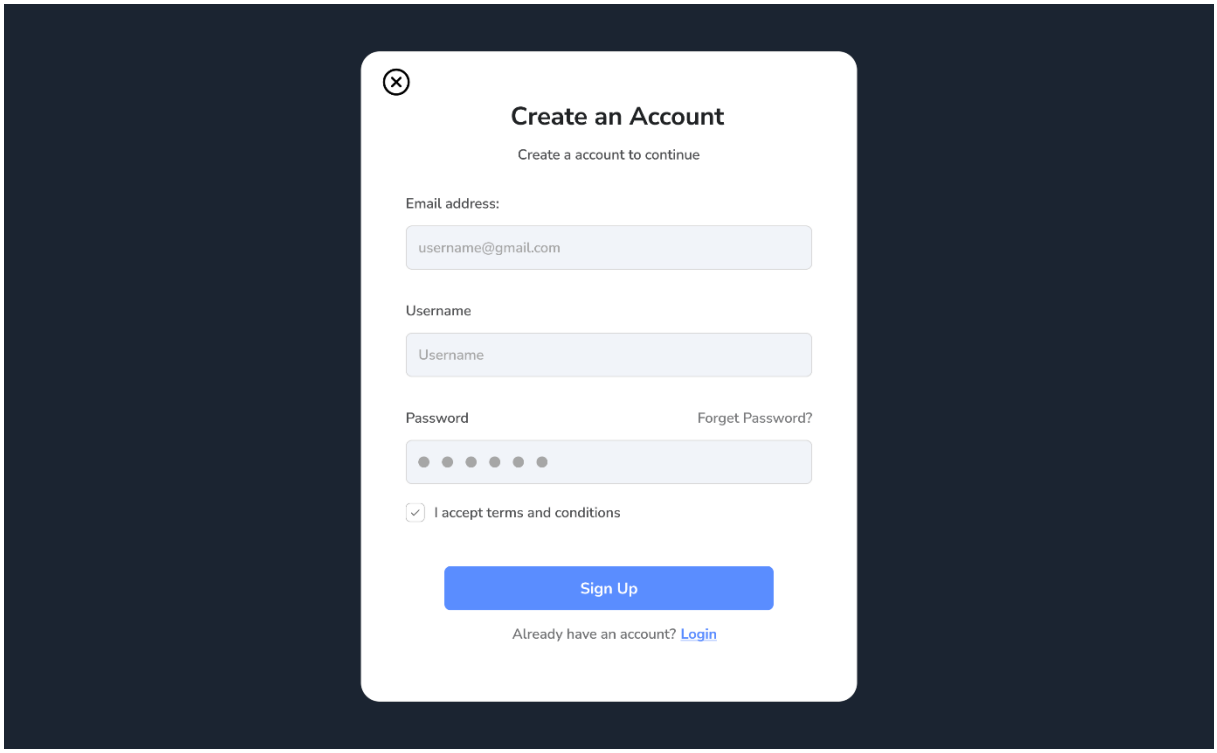


Figure 3.16: Sign Up page for Automated Essay Grading System

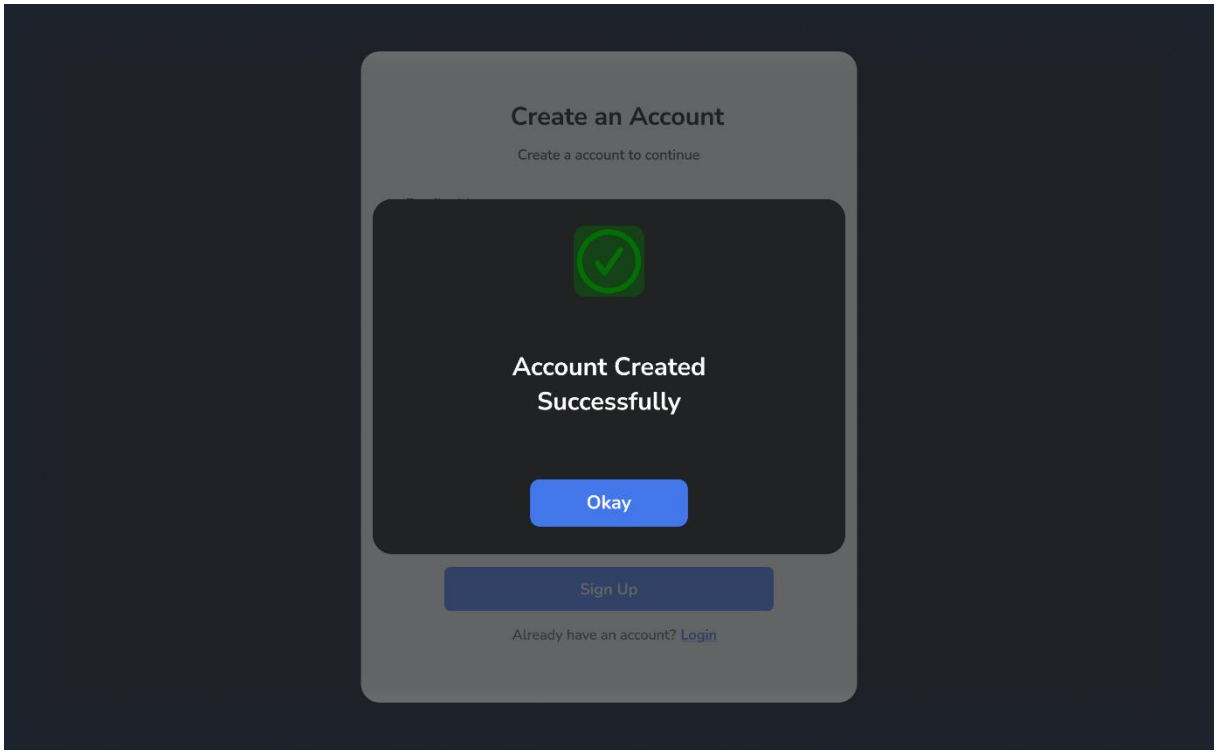


Figure 3.17: System pop up message when successfully signing up.

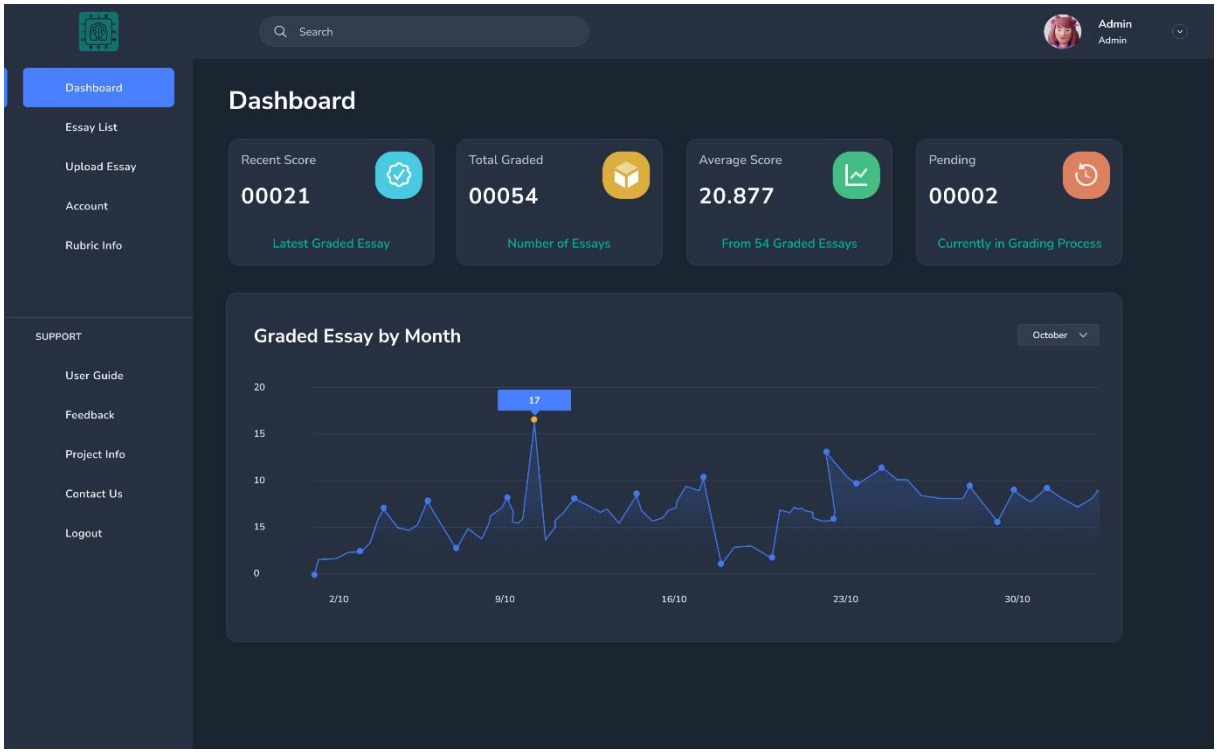


Figure 3.18: Dashboard for Automated Essay Grading System

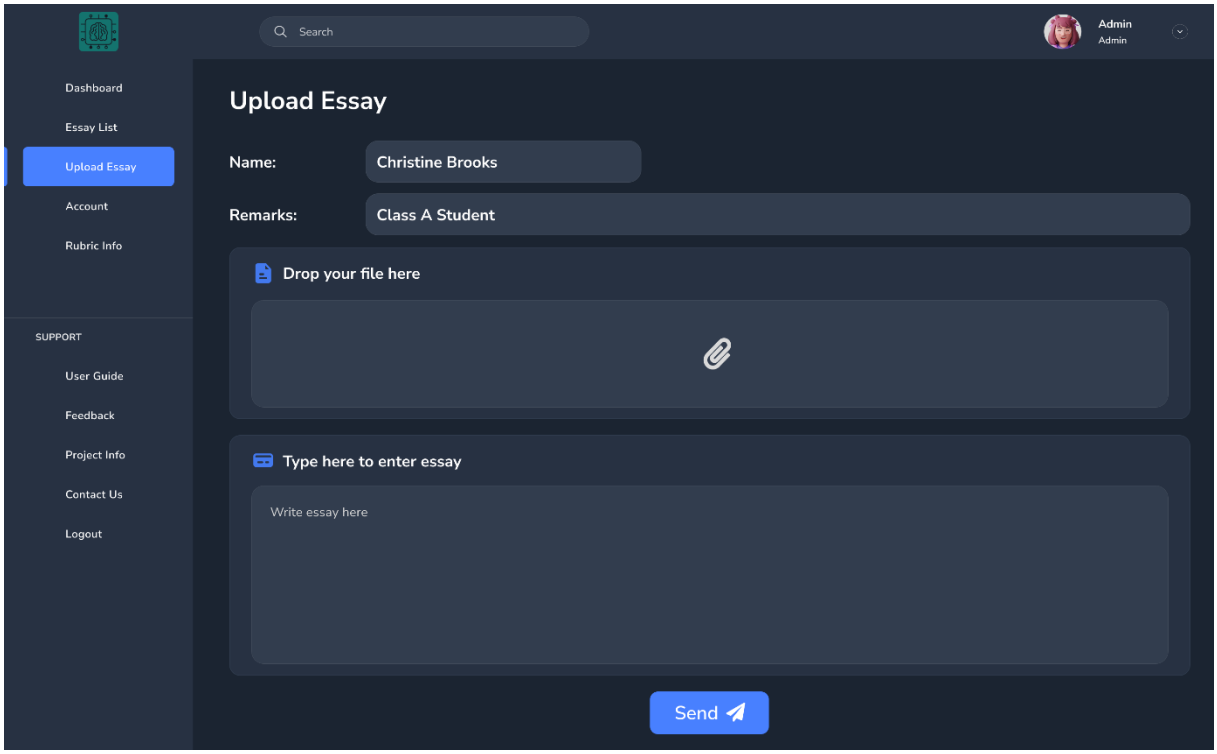


Figure 3.19: Upload Essay page for Automated Essay Grading System

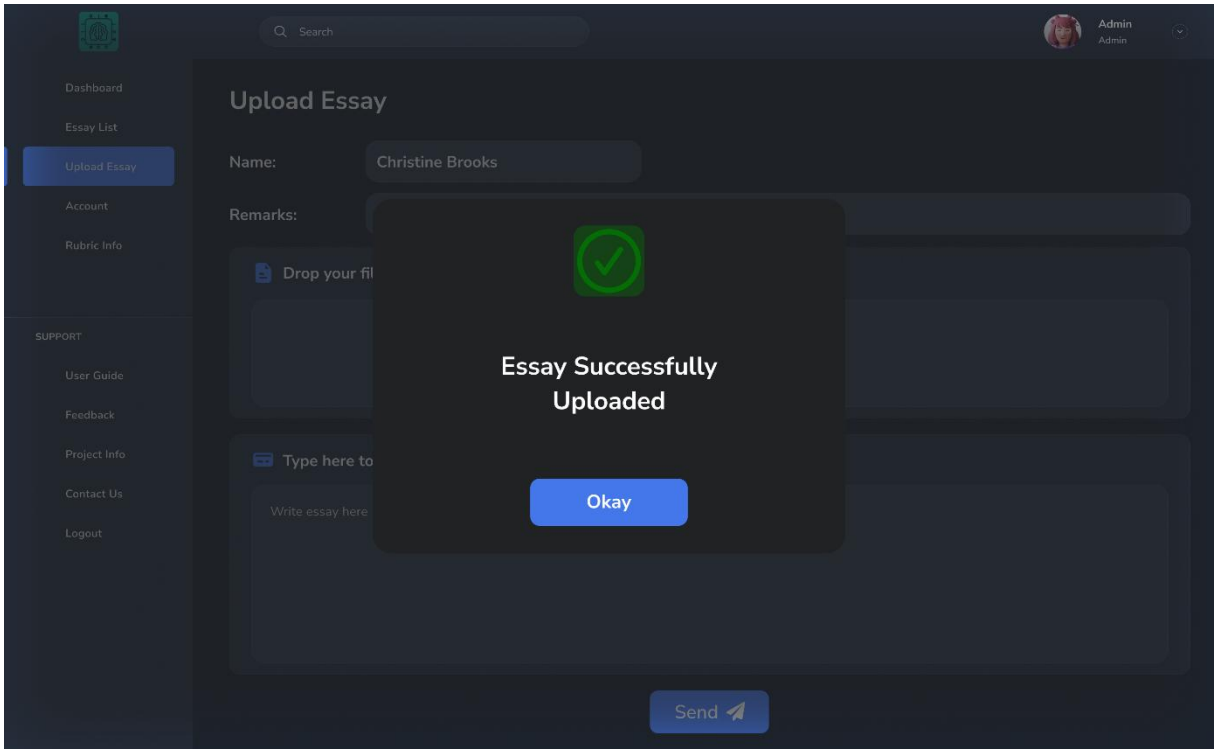


Figure 3.20: System pop up message when essay uploaded successfully

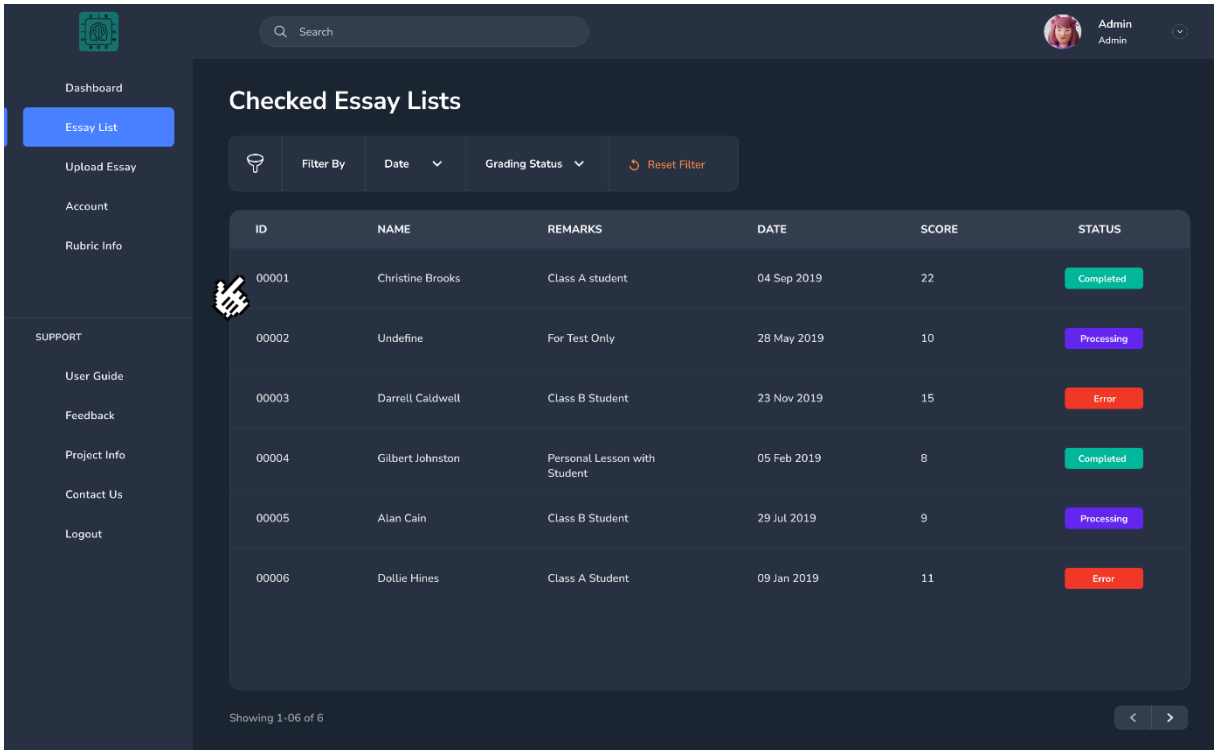


Figure 3.21: Checked Essay List page for Automated Essay Grading System

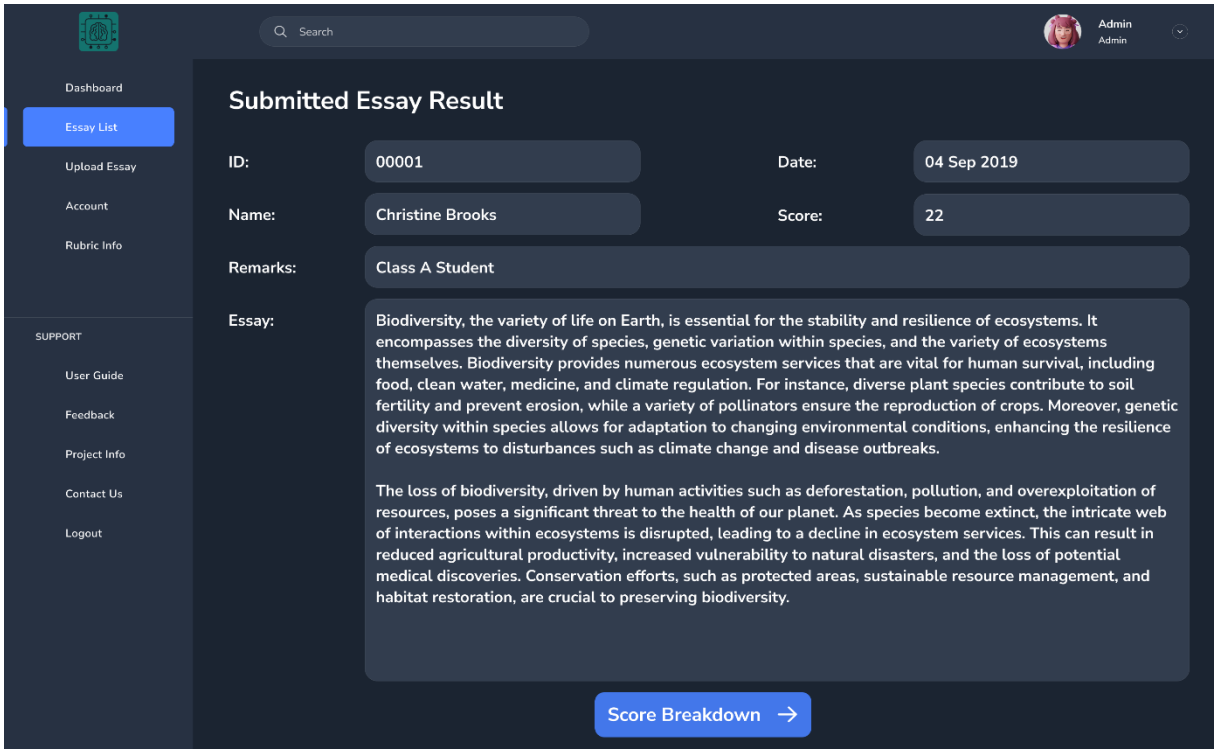


Figure 3.22: Submitted Essay Result page for Automated Essay Grading System

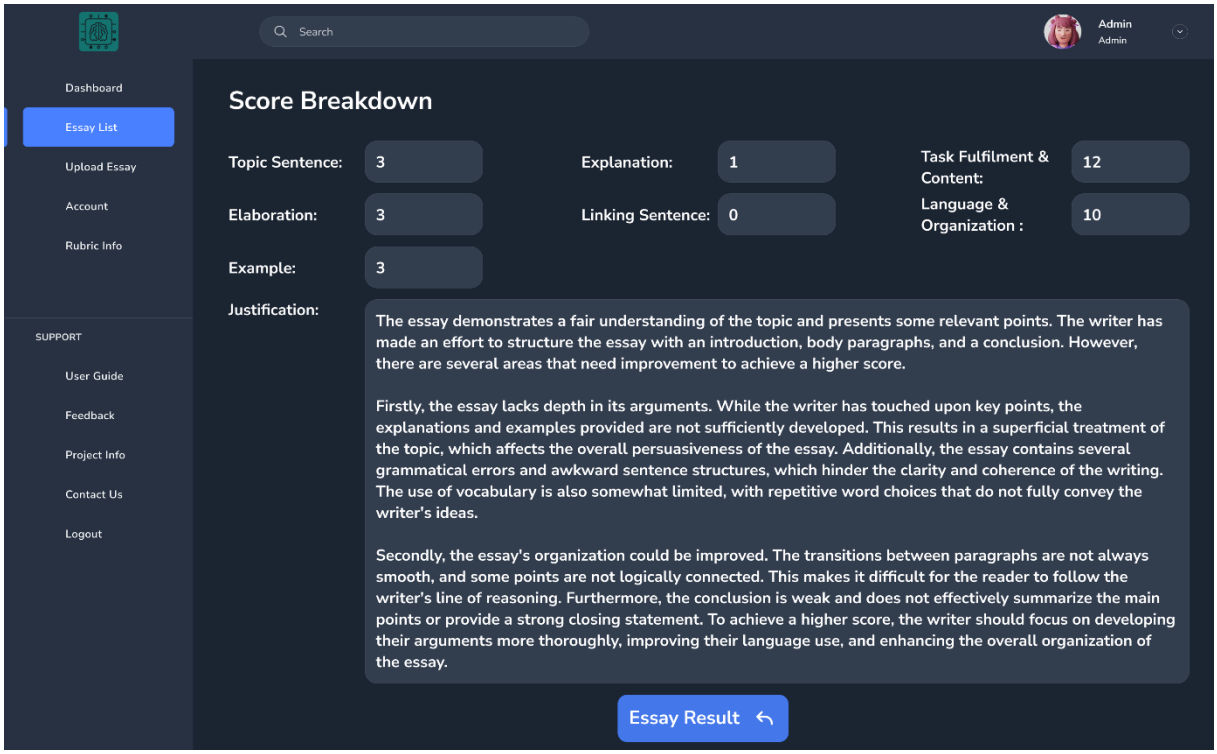


Figure 3.23: Score Breakdown page for Automated Essay Grading System

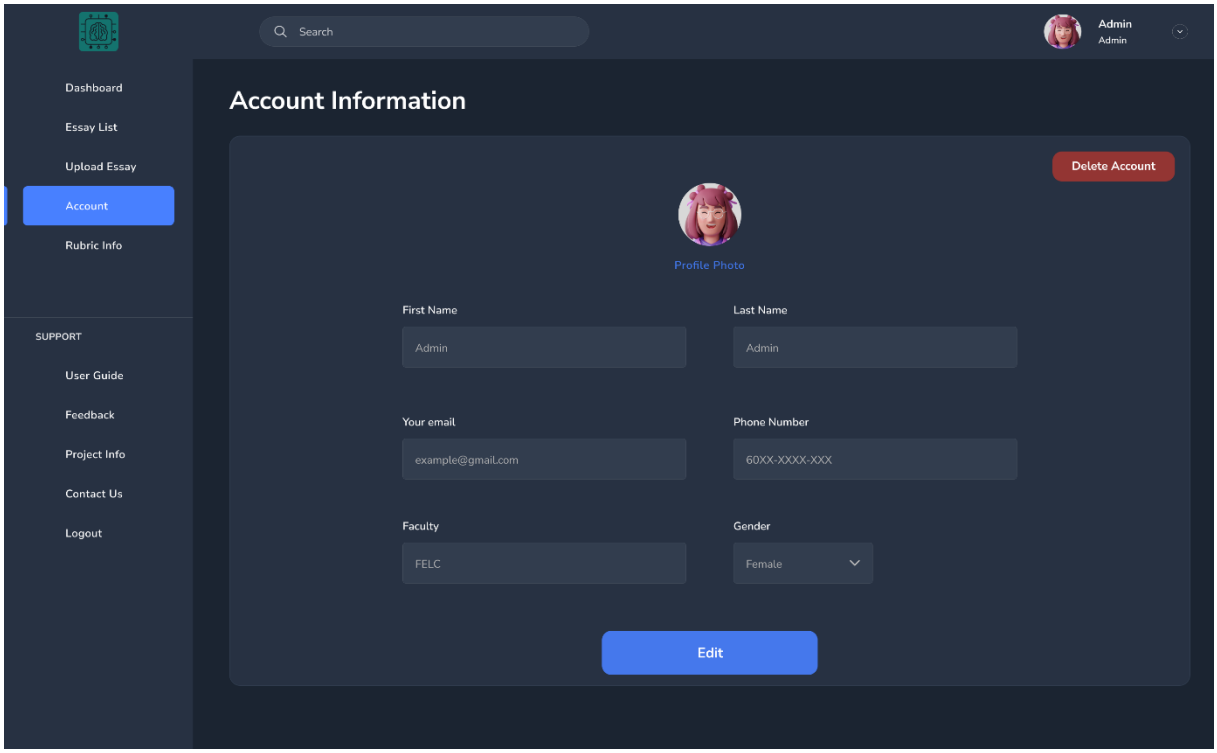


Figure 3.24: Account page for Automated Essay Grading System

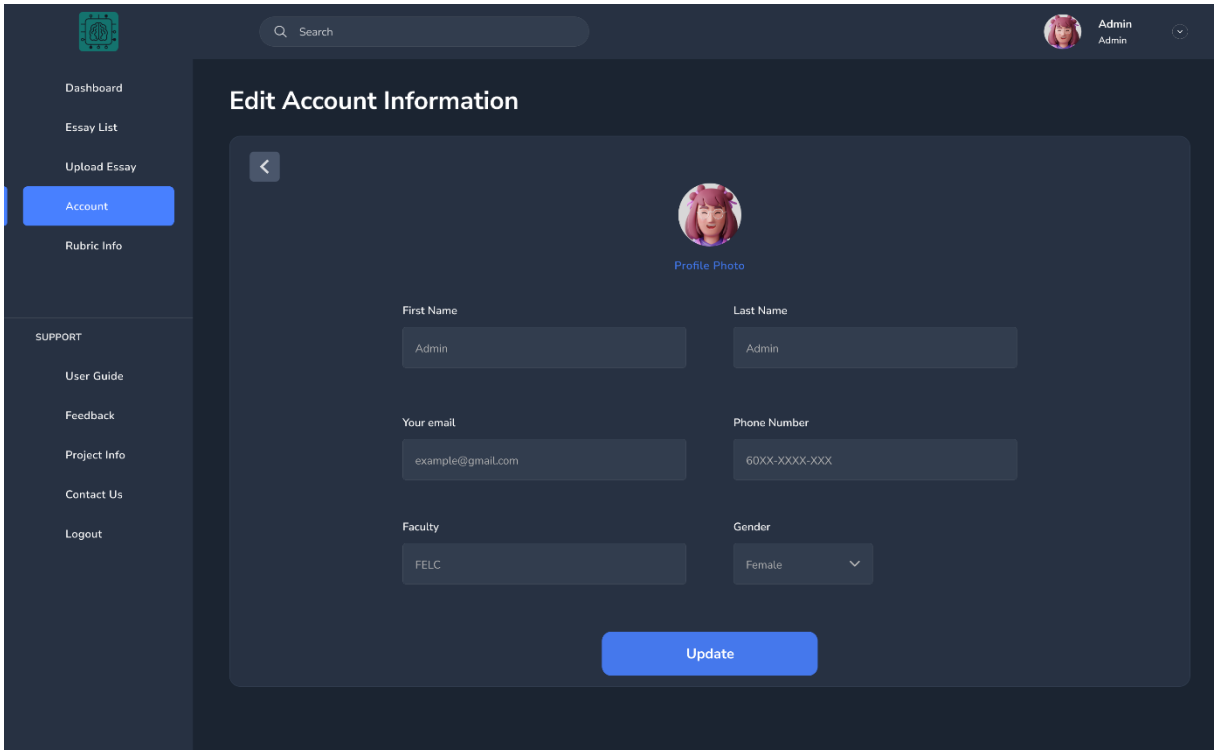


Figure 3.25: Edit Account page for Automated Essay Grading System

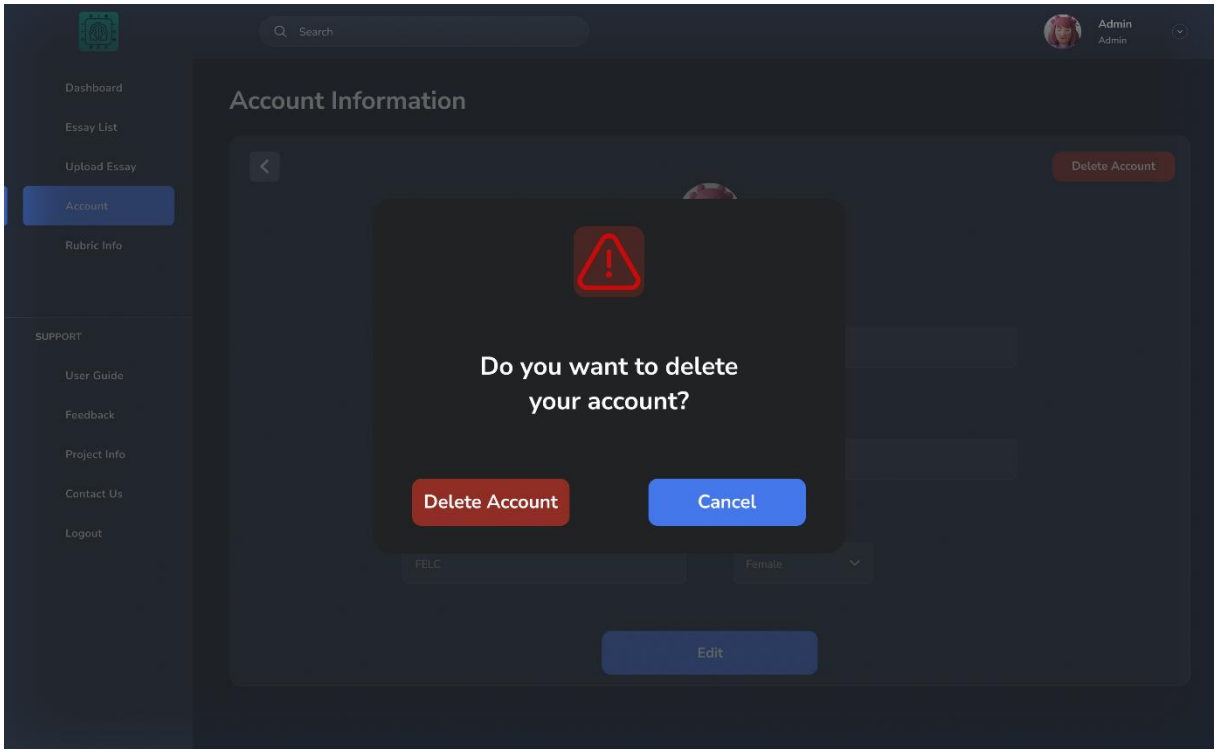


Figure 3.26: Delete Account confirmation prompt

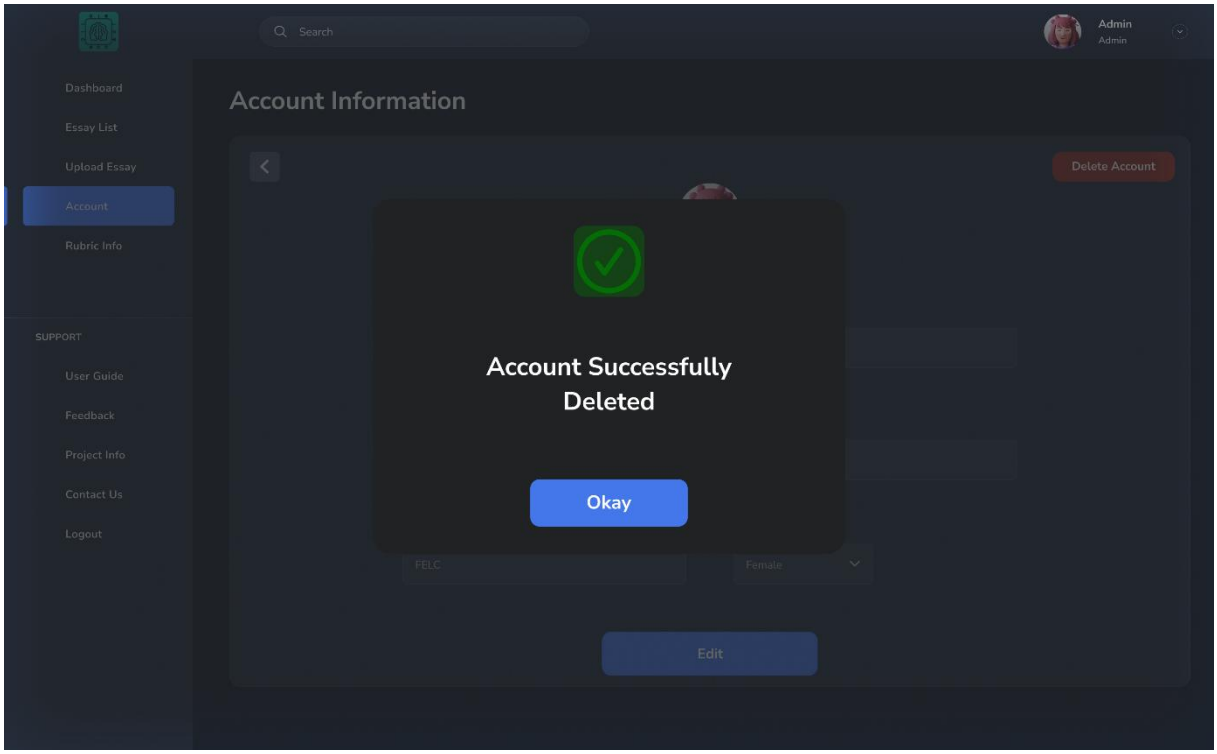


Figure 3.27: Account Deleted Successfully pop-up message

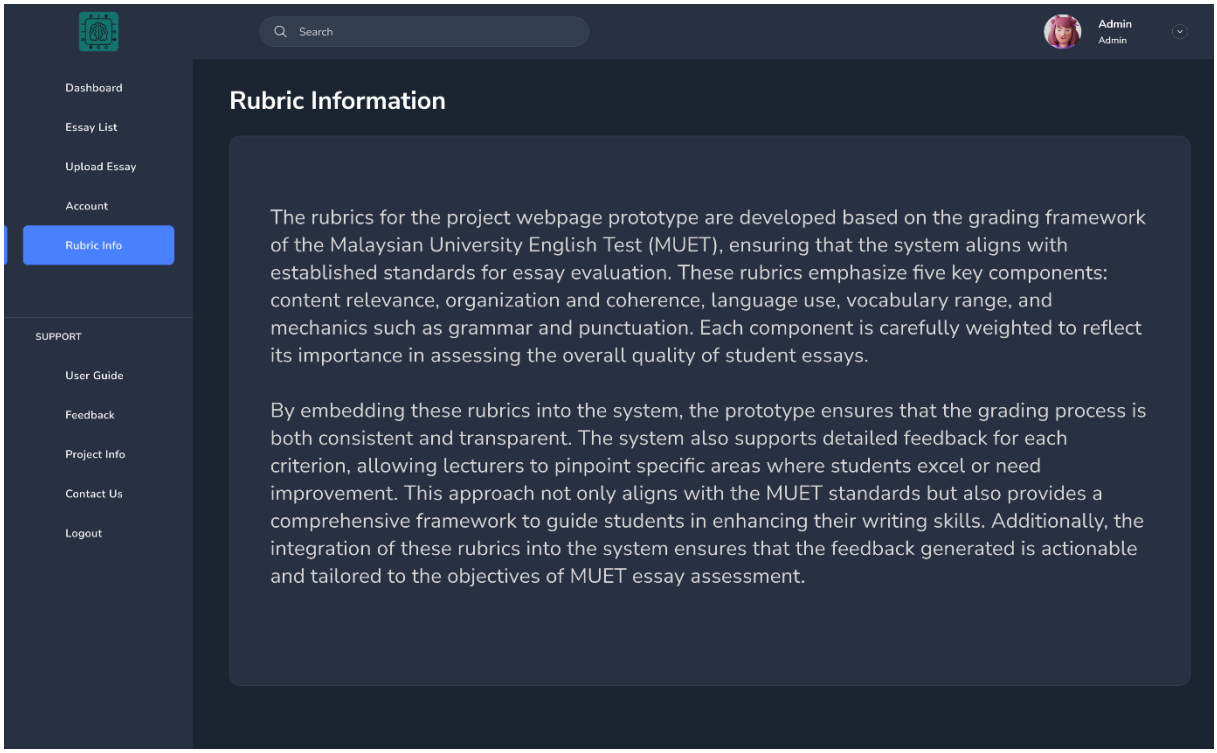


Figure 3.28: Rubric Info page for Automated Essay Grading System

3.2.6 Build by Features

This process will be implemented and refined after completing a detailed requirement analysis and project design phases where each feature will be selected based on their priority before being integrated into the system. Each feature will go through the following stages:

1. **Requirement Analysis**

   Defining the specific functionality to be built, ensuring alignment with project objectives and stakeholder needs.

2. **Feature Design**

   Creating the blueprint for how the feature will work within the system, including user interactions and system integration points.

3. **Implementation**

   Developing the feature using appropriate programming techniques, tools, and AI models tailored to the essay grading system.

4. **Testing and Validation**

   Ensuring the feature functions correctly through rigorous unit, integration, and user acceptance testing, with feedback incorporated for improvements.

5. **Deployment and Feedback Collection**

   Publishing the completed feature to user for evaluation and incorporating any insights into subsequent development cycles.

3.3 Summary

This chapter presented the methodology and outlined the five processes of Feature-Driven Development (FDD) applied to the project. The comprehensive approach ensures iterative development, robust design, and systematic implementation, leading to a functional Automated Essay Grading System

Chapter 4: Implementation

4.1 Introduction

This chapter discusses in more detail about the implementation of the Automated Essay Grading System. The system was developed based on the requirements and design specified in Chapter 3. The implementation phase focuses on both the user interface and the core system functionalities that support the essay grading process using artificial intelligence. The frontend and backend were implemented using HTML, CSS, JavaScript, and Django (Python framework), while Optical Character Recognition (OCR), essay feedback generation and essay scoring features were integrated using Azure AI services.

This chapter is divided into several sections, starting with the installation and configuration of the software and tools used during development. Following that, detailed implementation of each module will be explained, including how the system uploads essay images, extracts text using OCR, and generates personalised detail essay feedback and scores using AI.

4.2 Installation and Configuration of Software/Tools Used

The development of the Automated Essay Grading System is mainly develop using Python and involved the use of several tools and software to facilitate system design, coding, deployment, and integration of artificial intelligence services. This section outlines the key software and services used, along with their setup and configuration during the development process.

4.2.1 Visual Studio Code (VS Code)

Visual Studio Code is a lightweight but powerful source-code editor used throughout the development phase (Microsoft, 2021). Figure 4.2.1.2 shows the welcome page of the VS Code when opening the editor. The editor is chosen because it provides essential features such as syntax highlighting, Git integration, debugging tools, and extension support. It is configured to support Python and Django development through extensions such as Python, Django, and Azure Tools. The software can be downloaded from the official website at https://code.visualstudio.com.

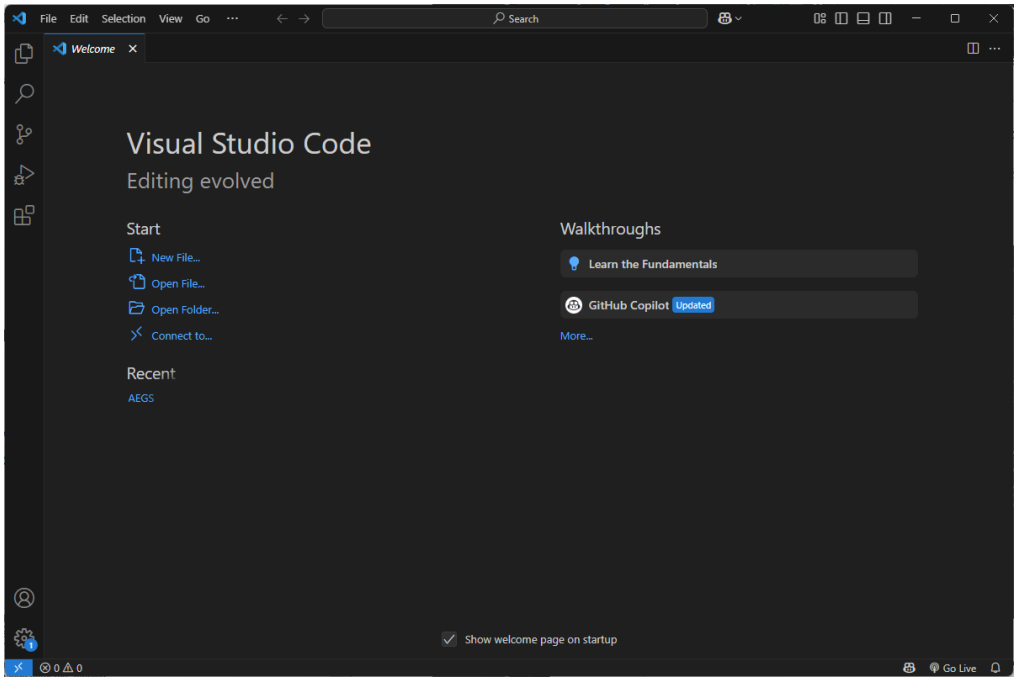


Figure 4.2.1.1: Welcome Page of Visual Studio Code

4.2.2 Django Framework

Django is a high-level Python web framework that is used to build the server-side logic of the system (django, 2025). It handles routing of webpages, database models using SQLite, form submissions, authentication, and API integration. The environment was configured by installing necessary dependencies using “pip” which is a package installer bundled when installing Python. The Figure 4.2.2.1 show the installation of Django using VS Code terminal with the command “python -m pip install Django”.

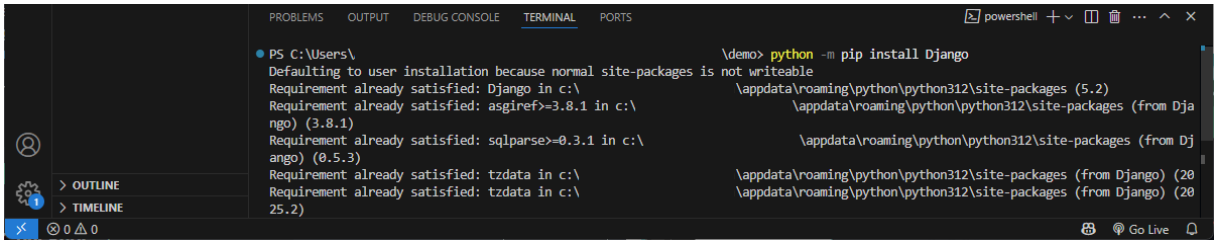


Figure 4.2.2.1: Installation of Django using VS Code Terminal

The framework provides clear and well-structured documentation which make troubleshooting any issues encountered during development easier. The official website is accessible at https://docs.djangoproject.com/en/5.2/. As shown in Figure 4.2.2.2, the documentation provides guidance to helps first-time developers navigate, troubleshoot, and streamline their workflow.

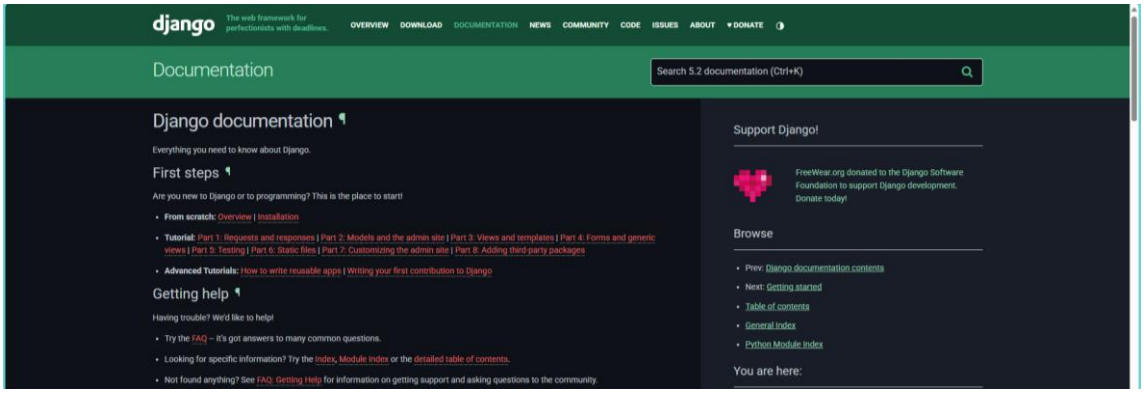


Figure 4.2.2.2: Django Documentation Webpage

In Django framework, the important files that need to be configured carefully are “models.py”, “views.py”, and “urls.py”. The first file is where the database schema for the project is defined, and it contains the model classes that represent tables in the database. As shown in Figure 4.2.2.3, Django uses an Object-Relational Mapping (ORM) system, which allows developers to interact with the database using Python code instead of raw SQL queries. There are four models class define for the project which are class Lecturer, Essay, EssayImage, and ScoreBreakDown. Once defined, Django automatically generates the corresponding SQL tables when “python manage.py makemigrations” and “python manage.py migrate” commands are run.

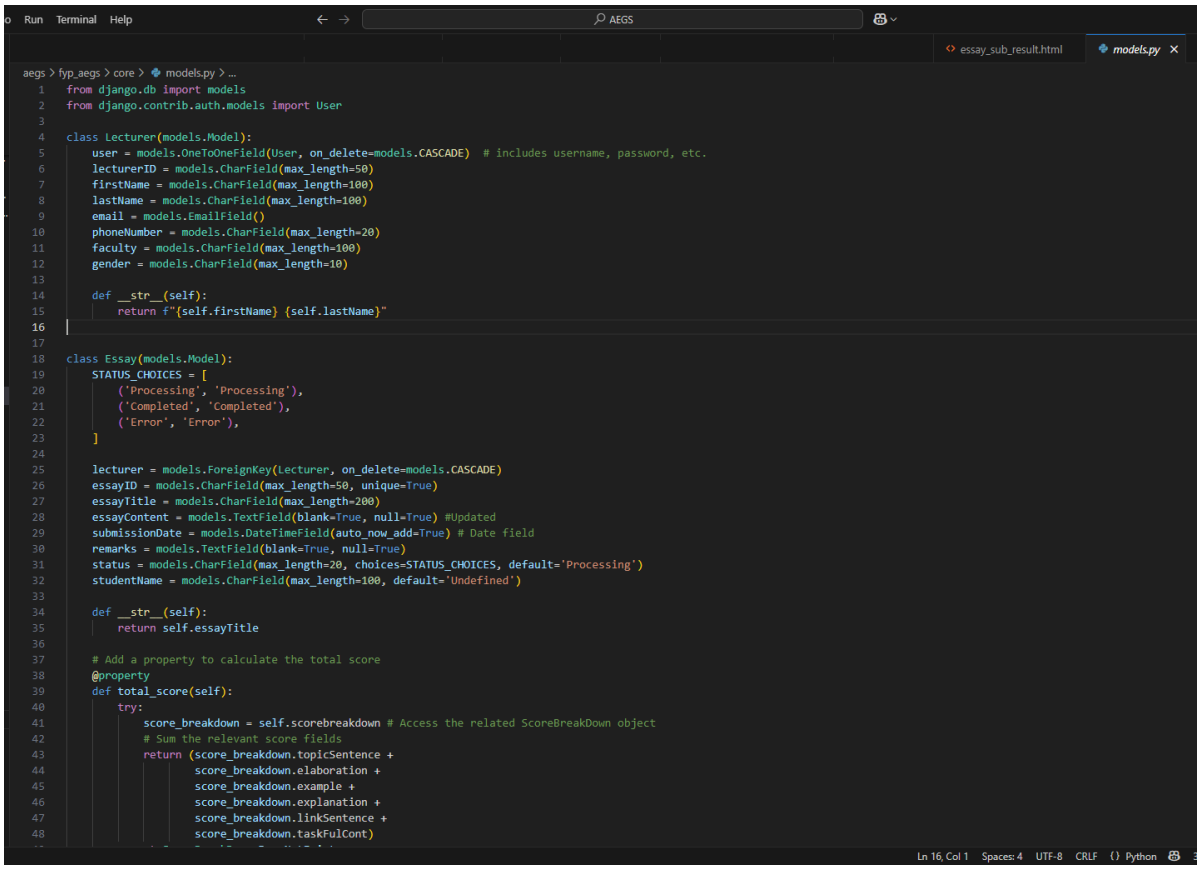


Figure 4.2.2.3: Models of Automated Essay Grading System

The second file which is “views.py”, handles the project logic and respond to user requests. Each of the view function or class receives an HTTP request, processes data or calls an API, and returns an HTTP response or JSON. The configuration in this file which is shown in Figure 4.2.2.4 also handles the upload of the essays and the images, perform Optical Character Recognition (OCR) of the uploaded using Azure AI Document Intelligence, send essay content to Azure OpenAI service for detail feedback generation and display results to lectures.

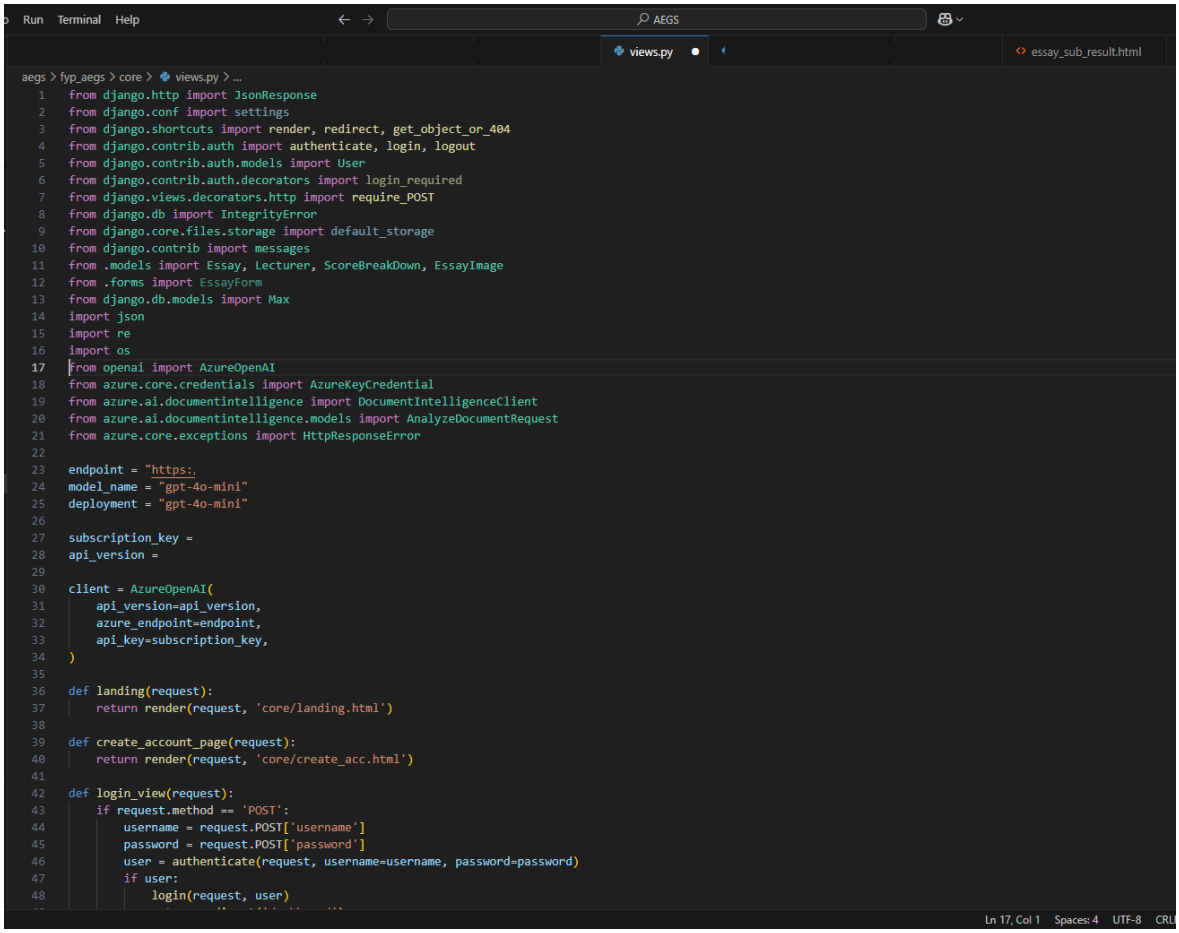


Figure 4.2.2.4: View functions of Automated Essay Grading System

The “urls.py” file maps the URLs to the view functions and acts as the router for the Automated Essay Grading System by directing the HTTP requests to the correct view logic based on the requested path which is shown in Figure 4.2.2.5. Without this file, Django would not know how to connect incoming URLs to the appropriate functionality.

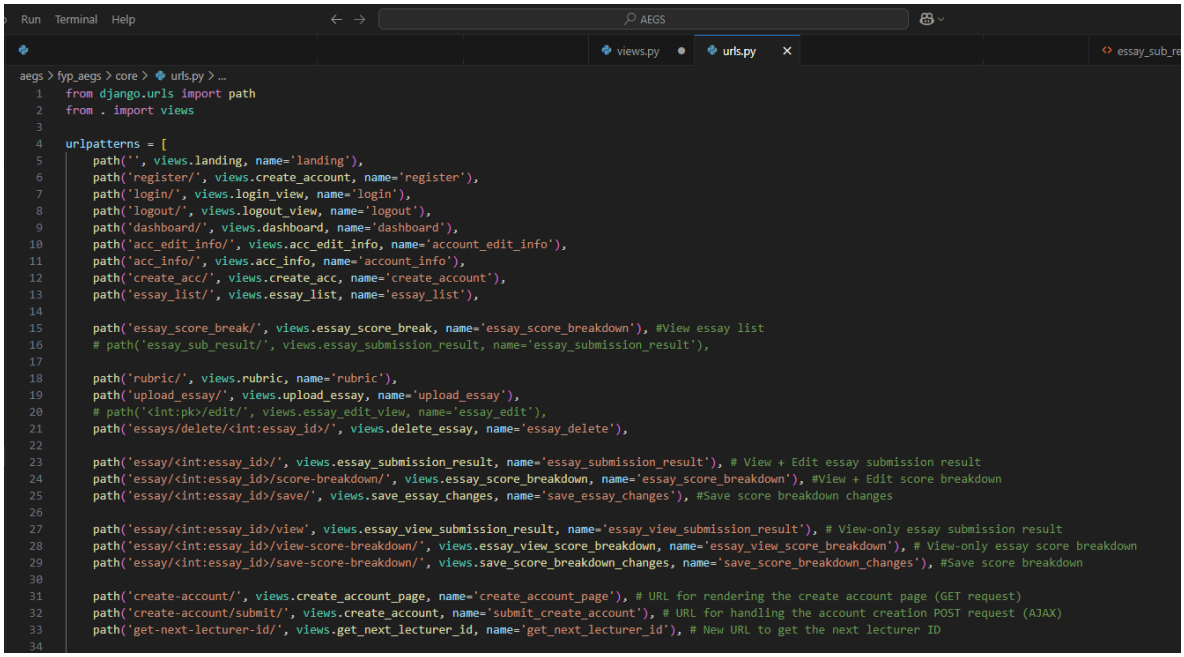


Figure 4.2.2.5: URLs map configuration of Automated Essay Grading System

4.2.3 Azure OpenAI Service

Azure OpenAI service is used to perform AI-driven essay feedback generation. For the project, the gpt-4o-mini deployment model is chosen from the interference task for Chat Completion models to generating essay feedback justifications. Figure 4.2.3.1 show Azure OpenAI to manage the deployment of the service while Figure 4.2.3.2 show Azure AI Foundry, a platform use designed to deploy, setup the deployment model and select the type of inference tasks.

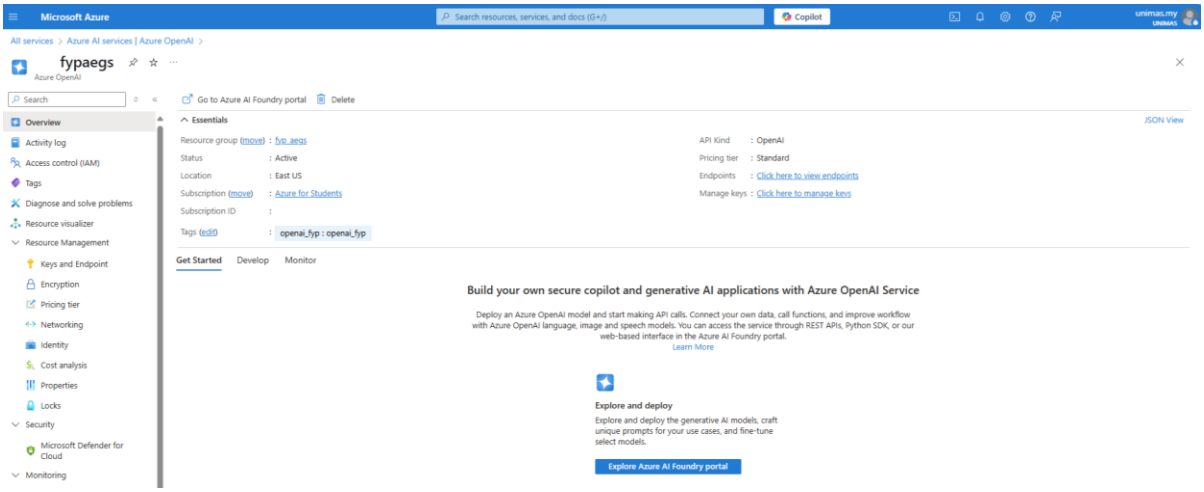


Figure 4.2.3.1: Azure OpenAI resource management page

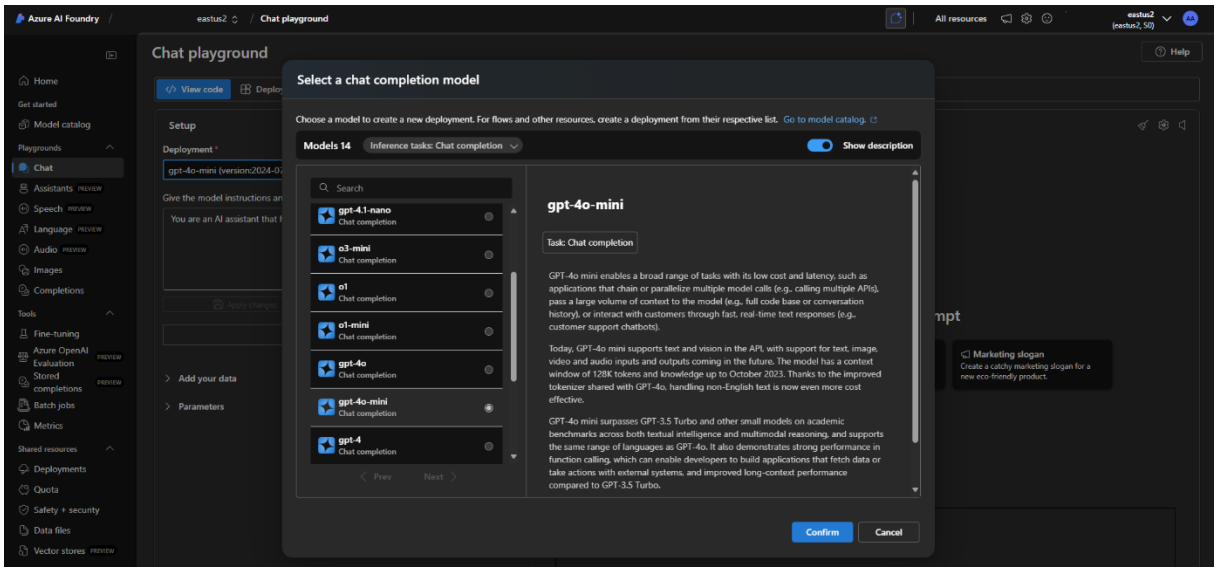


Figure 4.2.3.2: Azure AI Foundry model deployment

In the project, the Azure OpenAI usage is configured in the “views.py” for the function uploading essay as shown in Figure 4.2.3.3. The role of the model is set in the configuration alongside with the prompt specially tailor for the Automated Essay Grading System. The content and the score breakdown of the essay also given included in the prompt to provide AI model with more data and contextual information which promote it to generate detailed feedback tailored to the specific essay rather than offering a general assessment based solely on the score breakdown.

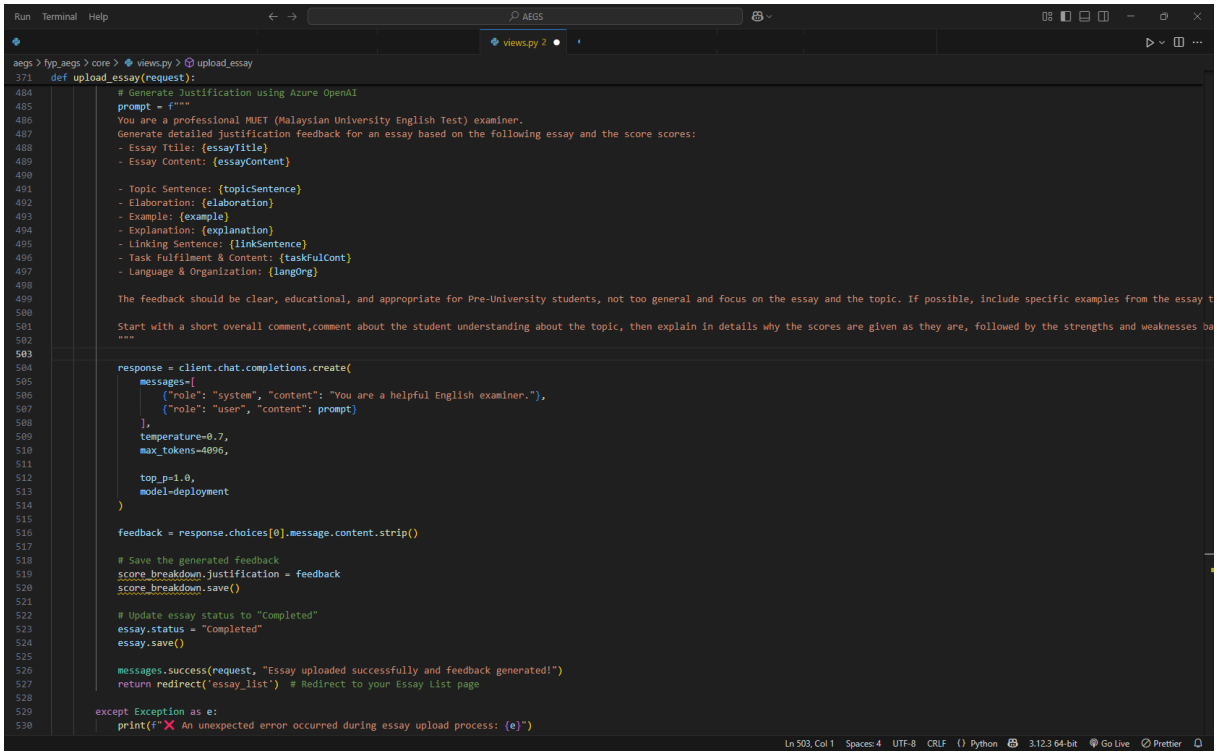


Figure 4.2.3.3: Upload Essay Function to utilize Azure OpenAI

4.2.4 Azure Document Intelligence

Azure AI Document Intelligence is use for OCR (Optical Character Recognition) to extract text from uploaded essay images in the project. This service allows the system to process handwritten or printed essays and convert them into machine-readable text. A Document Intelligence resource was configured in the Azure portal, as illustrated in Figure 4.2.4.1. The API credentials were integrated into the views.py file within the upload essay function, as shown in Figure 4.2.4.2. Image uploads were managed using Django's FileField, and Optical Character Recognition (OCR) results were retrieved via Azure AI Document Intelligence REST API using the synchronous layout model.

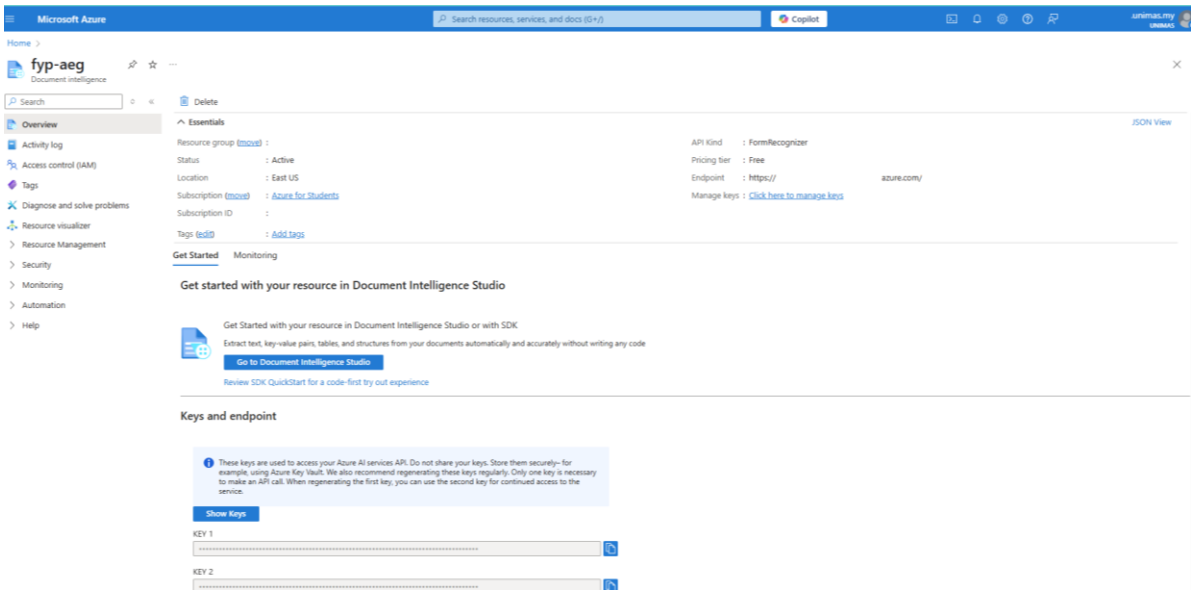


Figure 4.2.4.1: Document Intelligence resource management page

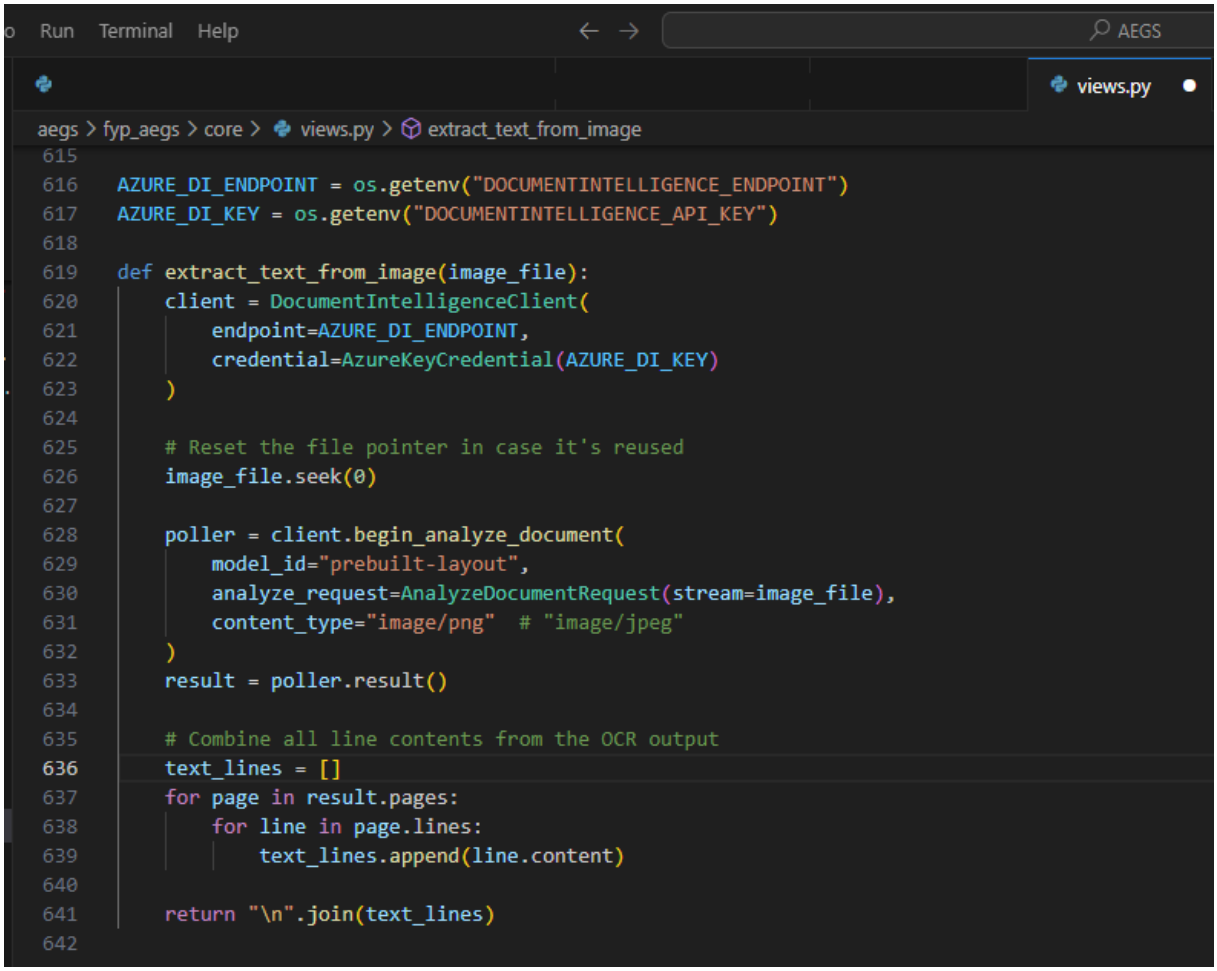


Figure 4.2.4.2: Extract text from image function to handle image upload to Azure Document Intelligence Service

4.3 System Module and Prototype

4.3.1 Login Module

The login module contains login page which allows registered lectures to securely access their accounts. Figure 4.3.1.1 show the login page where it includes fields for emails and password, and implements frontend validation to prevent empty submissions before lecturer login.

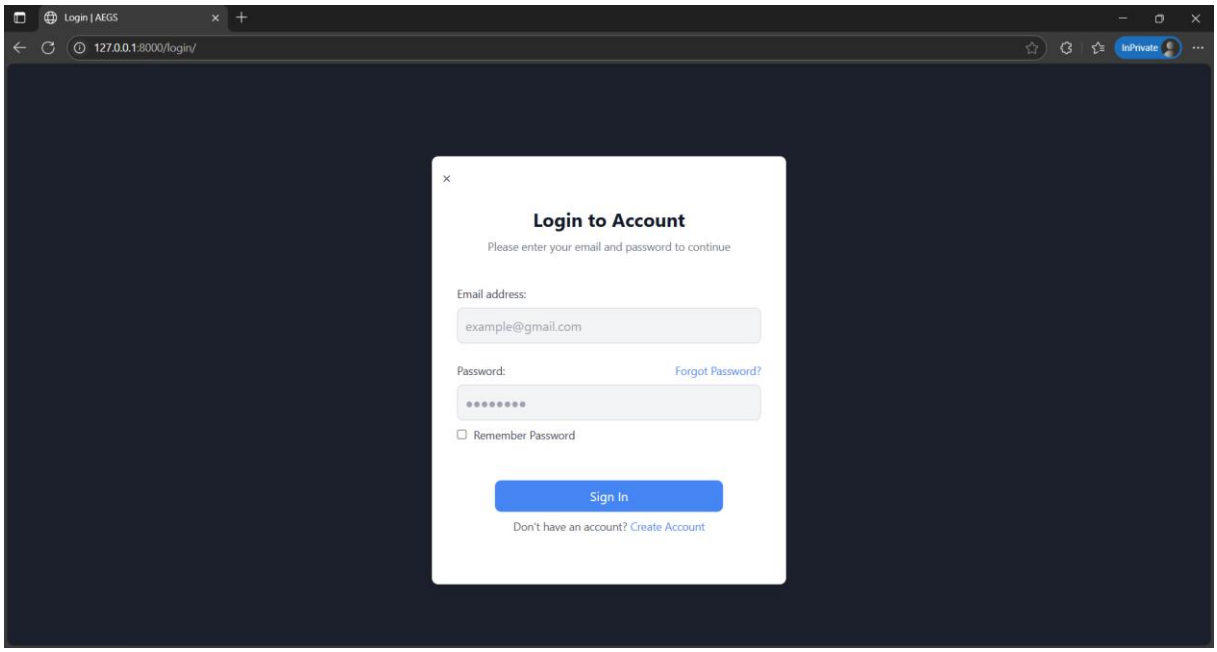


Figure 4.3.1.1: Login Page

4.3.2 Registration Module

The registration module is where new users can create an account. New user needed to provide details such as full name, email, phone number, faculty, gender, username and password as shown in Figure 4.3.2.1. The system validates inputs before submission and confirms successful registration via a popup notification.

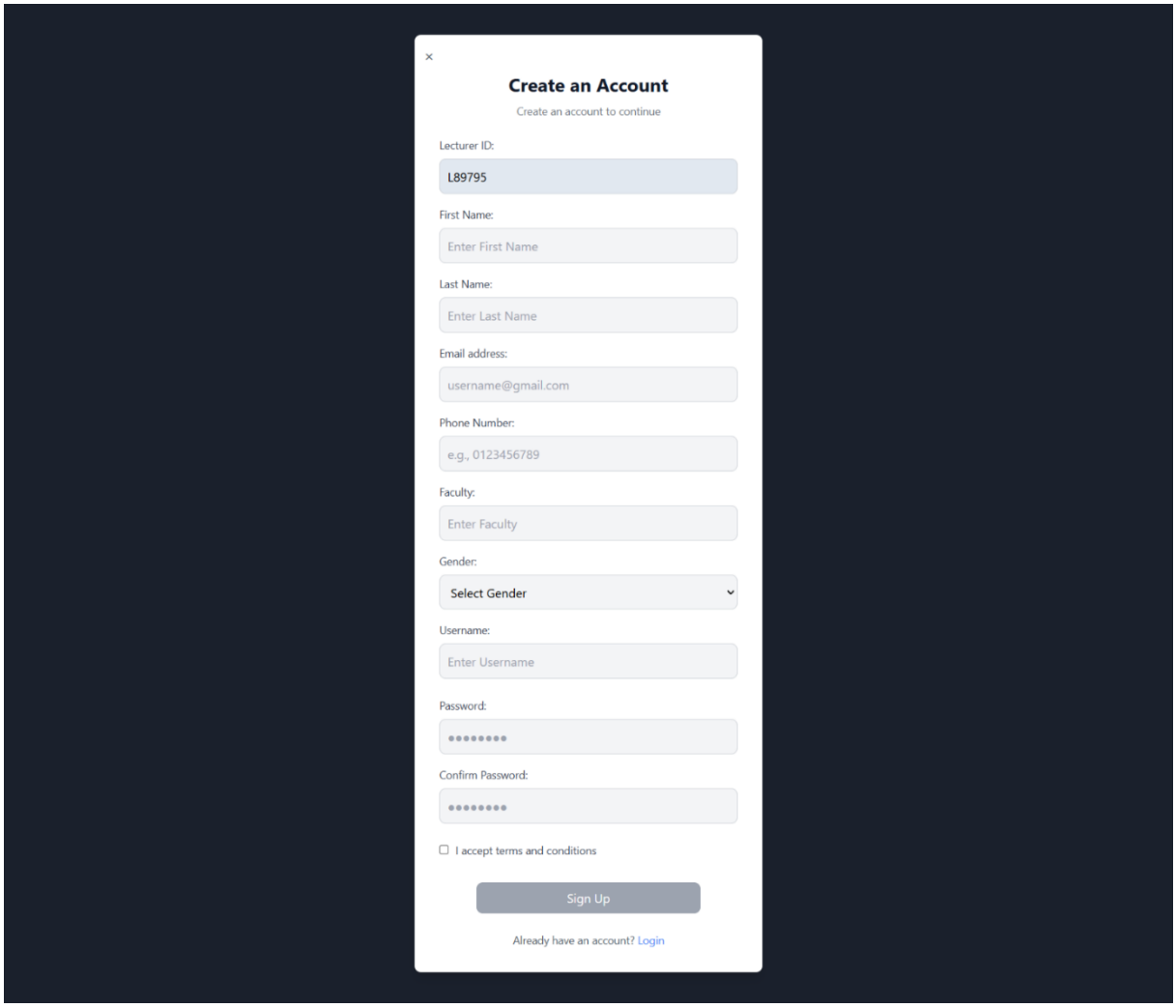


Figure 4.3.2.2: Registration Page

4.3.3 Essay Management Module

The essay management module displays all of submitted essay by the lecturer in a page called "Checked Essay List". It includes functionalities such as sorting, filtering, and status indicators as shown in Figure 4.3.3.1 (Checked Essay List). Lecturers can also view, update, or delete individual entries.

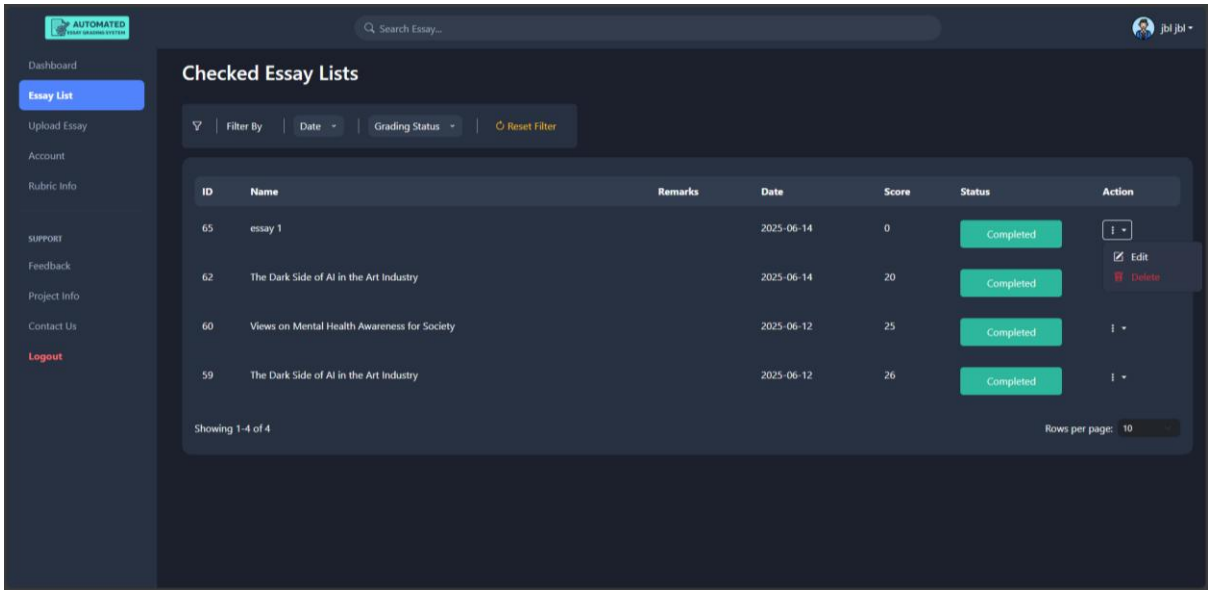


Figure 4.3.3.1: Essay List Page

The module also includes a page called "Upload Essay" where the page allows upload typed essays or images of handwritten essays. The form includes fields for title, essay content (optional if image provided, and image upload (currently support PNG image type only) as shown in Figure 4.3.3.2 (upload essay). Upon submission, the essay is processed for OCR and feedback generation.

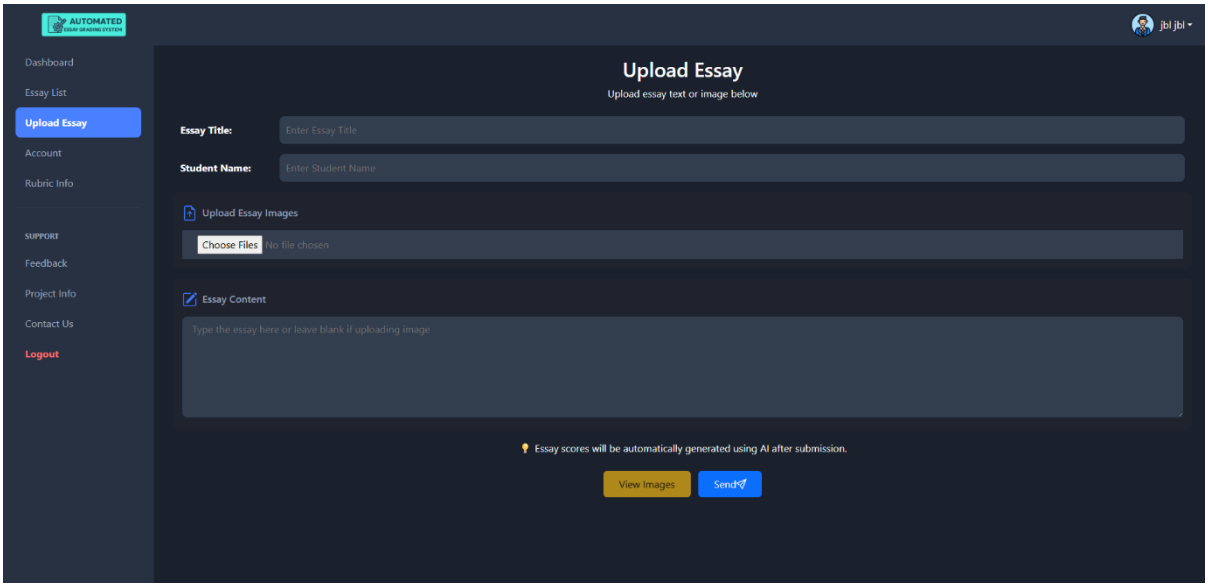


Figure 4.3.4.2: Upload Essay Page

4.3.4 Optical Character Recognition (OCR) Module

The OCR module handle essay image (handwritten or type) conversion into text. When users upload an essay image using "Upload Essay" page shown in Figure 4.3.3.2 above (upload essay), the OCR module extracts text using Azure Document Intelligence. The extracted content is saved into database and use in essay score breakdown module to generate score and essay feedback generation module for personalized feedback. The extracted content and text can be viewed on the "Submitted Essay Result" page, where users can see both the extracted essay text and the uploaded essay image, as shown in Figure 4.3.4.1 (OCR in Submitted Essay).

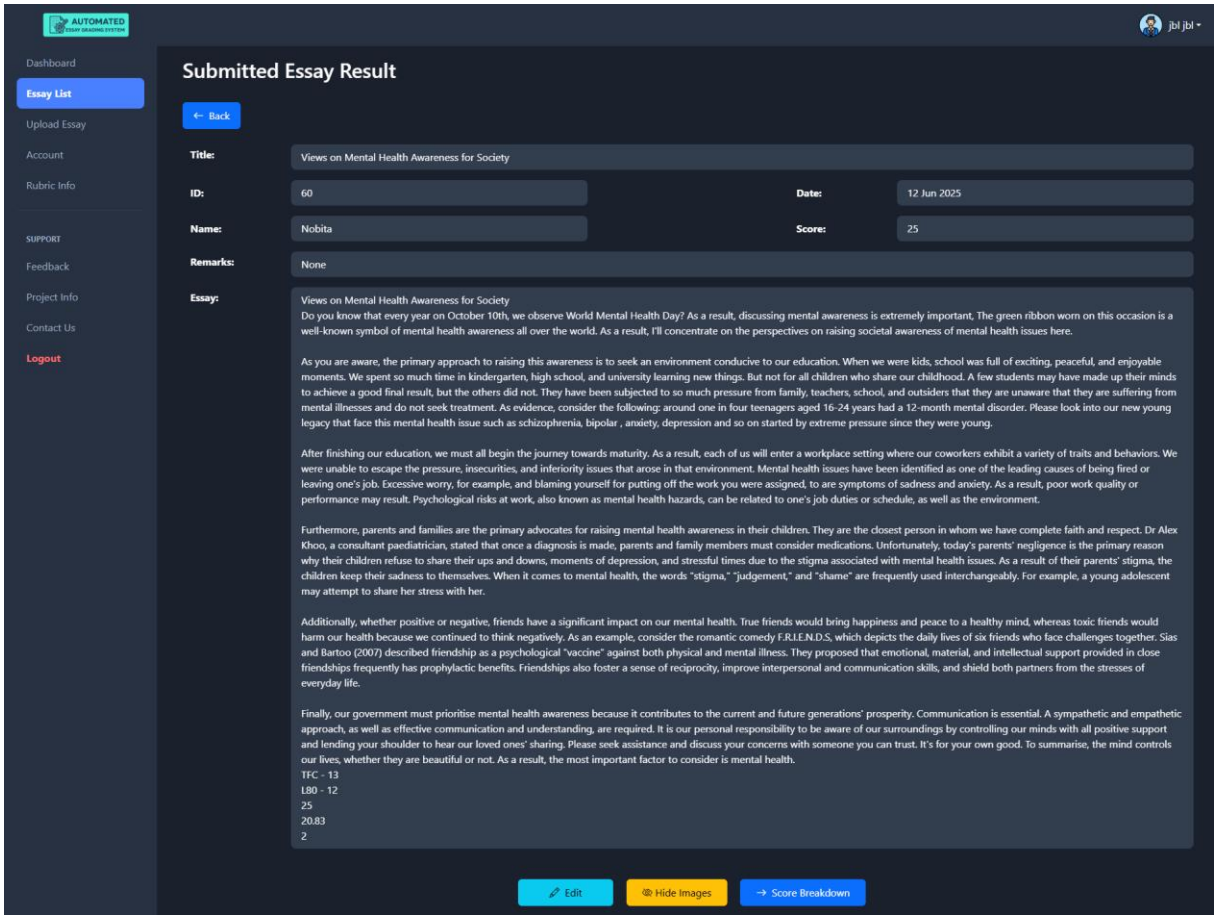


Figure 4.3.5.1: OCR module used in Submitted Essay Result Page when image is uploaded

4.3.5 Essay Feedback Generation Module

After the text is extracted or typed and the score breakdown module already giving the essay score, the text and the score will be send and processed in this module. Azure OpenAI model is used in this module to generate detailed feedback. The AI-generated justification is saved into the database and can be view by the user in "Score Breakdown" page as shown in Figure 4.3.5.1.

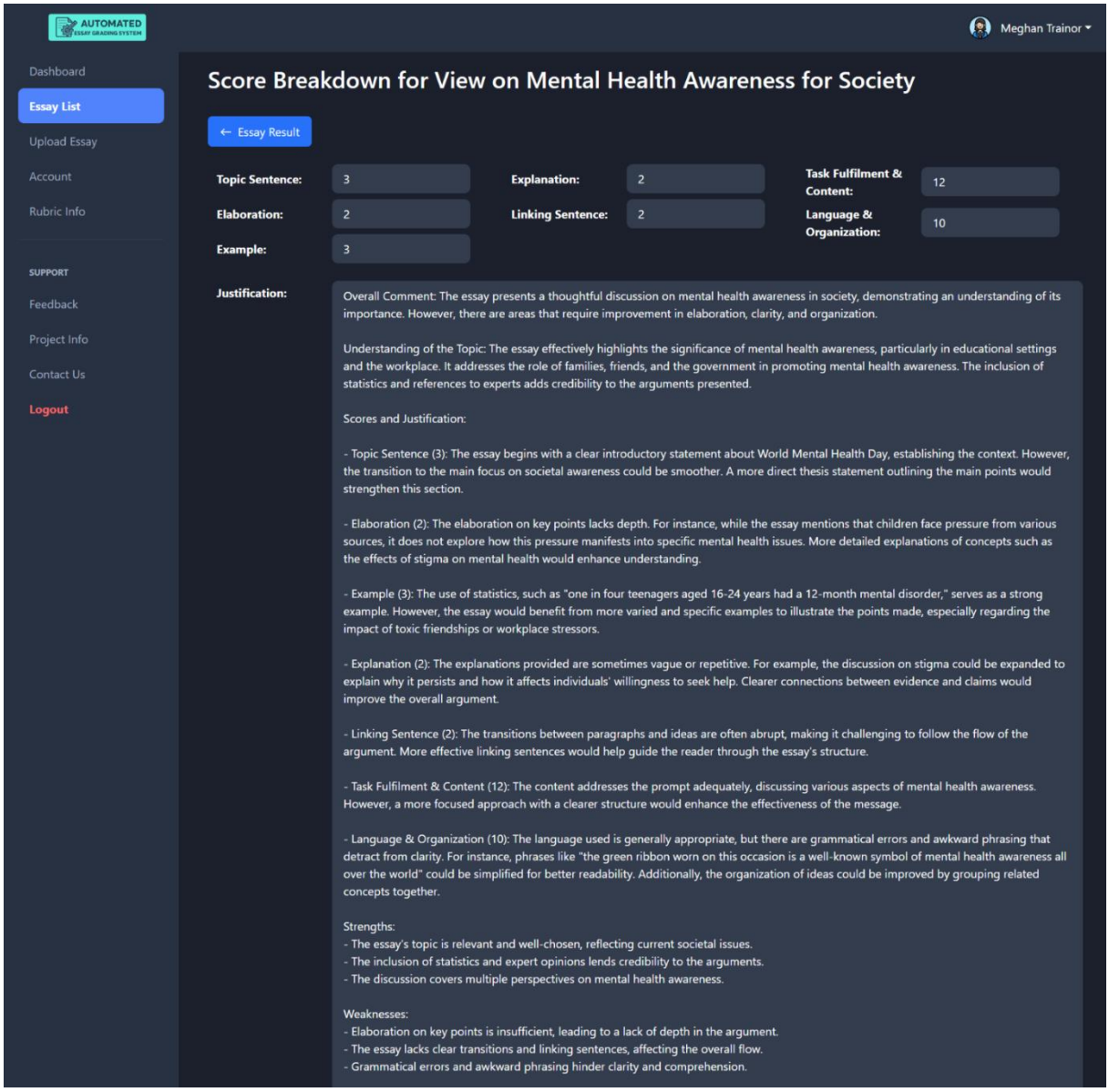


Figure 4.3.6.1: Essay feedback generation module for Score Breakdown for Submitted Essay Page

4.3.6 Essay Score Breakdown Module

In this module, the extracted essay text is processed by a trained AI model, which assigns an appropriate score based on predefined evaluation criteria. The AI model is trained on dataset provided by lecturer of FELC. The assigned scores, categorized according to various rubric components and then displayed on the 'Score Breakdown' page in a structured tabular format for easy interpretation, as illustrated in Figure 4.3.6.1.

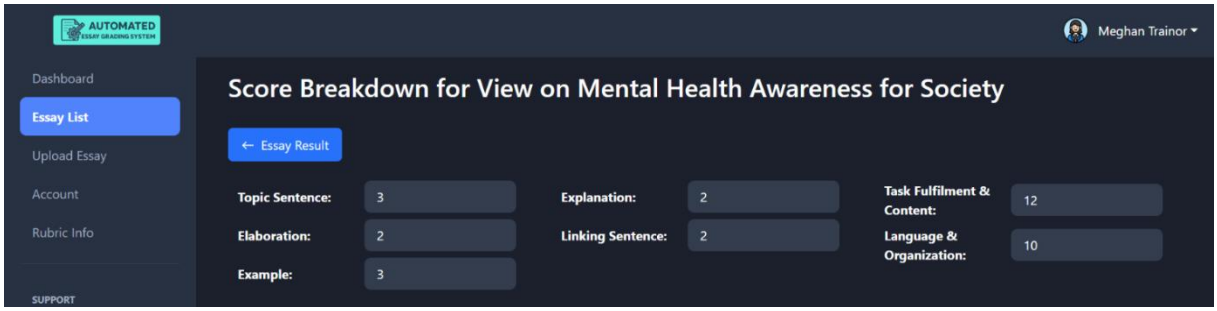


Figure 4.3.7.1: Essay feedback generation module for Score Breakdown for Submitted Essay Page

4.3.7 User Account Module

This module enables lecturers to manage their accounts efficiently. They can view and update their account details, as well as delete their accounts if necessary. These functionalities are illustrated in Figure 4.3.7.1.

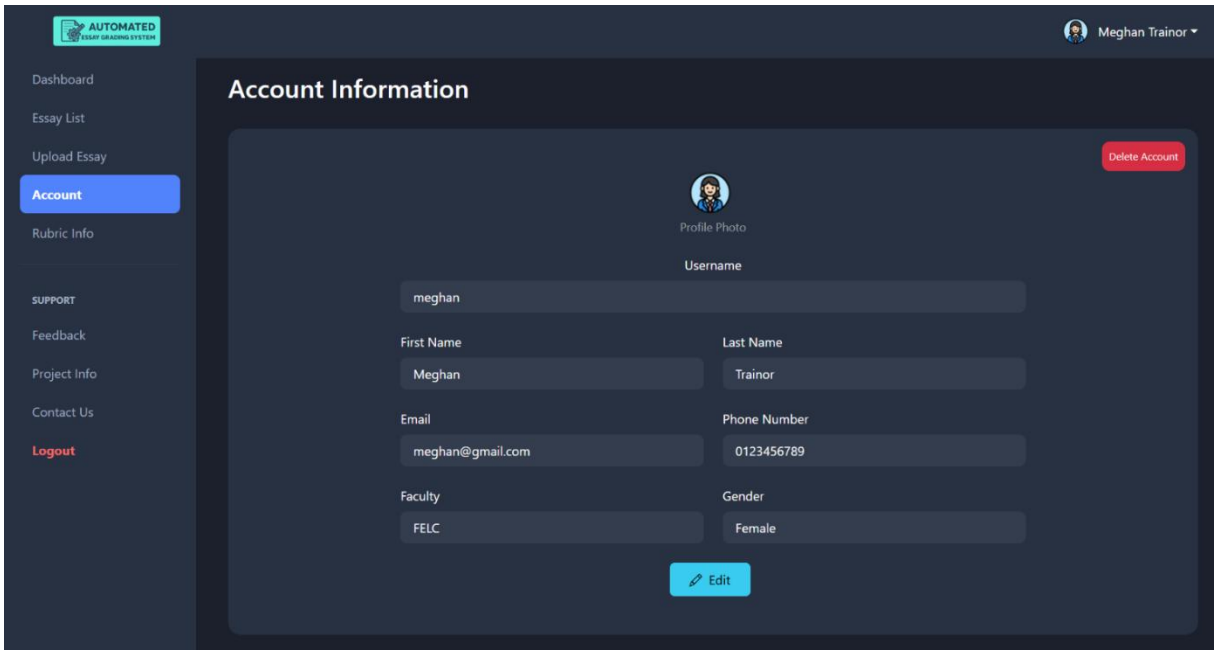


Figure 4.3.8.1: Essay feedback generation module for Score Breakdown for Submitted Essay Page

4.4 Dataset and AI Model Integration

4.4.1. Dataset

The dataset used for evaluation in this study comprises scanned MUET essays provided by a language teacher from UNIMAS. These essays were originally sourced from three PDF documents titled H1B.pdf, H2B.pdf, and H2D.pdf. Each document contains multiple scanned essay submissions that have been scored manually by qualified MUET examiners. A total of 52 essays were extracted from the PDFs. However, only 20 essays were deemed complete and suitable for evaluation, as they contained the full essay content along with clearly written scores for Task Fulfilment & Content, Language & Organization, and the total score. The remaining 32 essays were excluded due to missing titles, incomplete content, or absent handwritten scoring, and were placed into a folder labeled “incomplete”.

To prepare the usable data for processing, each page of the PDF files was first exported into high-resolution PNG image files. These images were then organized into a folder structure under a directory named “input_essay”. Each complete essay was placed in a uniquely named subfolder (e.g., essay_001, essay_002, etc.). A custom Python script was developed to perform OCR (Optical Character Recognition) on each image using Azure Document Intelligence. The extracted text was saved into individual .txt files. Essays with successfully parsed content were then submitted to the Automated Essay Grading System (AEGS) for scoring using the OpenAI GPT-4o mini model, guided by a structured prompt based on the MUET rubric. The AI-generated results comprising essay title and score breakdown were stored in a structured CSV file along with the corresponding lecturer-assigned scores for comparison. The data used to compute evaluation metrics and generate visualizations such as line and bar graphs illustrating the similarity between AI and human-assigned scores.

The dataset, although modest in size, includes a diverse range of essay topics commonly found in MUET exams, such as technology, social issues, and education. This diversity ensures that the AI scoring system is tested across multiple writing styles and thematic areas, providing a realistic and practical evaluation of its performance in academic essay assessment.

4.4.2. Prompt Design & Model Configuration

The essay evaluation module of the proposed system utilizes OpenAI's GPT-4o mini, a state-of-the-art large language model known for its multi-modal processing capabilities and efficiency (OpenAI, 2024). Rather than training a model from scratch, this approach leverages the model's pre-trained capabilities by employing prompt engineering, where task-specific instructions and context are embedded directly in the prompt submitted to the model.

The prompts were carefully crafted to guide the AI in evaluating essays based on predefined MUET rubric components, including Task Fulfilment & Content, and Language & Organization which are detailed further in Table 14 (Majlis Peperiksaan Malaysia, 2015). A breakdown of rubric-based scoring instructions is embedded into the system's request payload to ensure consistency and interpretability.

Table 14: MUET Rubrics Test Specification

Component	**Test Specifications**
Writing	Candidates are assessed on their ability to write various types of text covering a range of rhetorical styles. Assessment will cover the following: **(i) accuracy** • using correct spelling and mechanics

	• using correct grammar • using correct sentence structures **(ii) appropriacy** • using varied vocabulary and expressions • using clear varied sentences • using language appropriate for the intended purpose and audience • observing conventions appropriate to a specific situation or text type **(iii) coherence and cohesion** • developing and organising ideas • using appropriate markers and linking devices • using anaphora appropriately together with other cohesive devices **(iv) use of language functions** • defining, describing, explaining • comparing and contrasting • classifying • giving reasons • giving opinions • expressing relationships • making suggestions and recommendations • expressing agreement and disagreement • persuading • interpreting information from non-linear texts

	• drawing conclusions • stating and justifying points of view • presenting an argument **(v) task fulfilment** • presenting relevant ideas • providing adequate content showing a mature treatment of topic

The model is configured with the following inference parameters:

- Temperature: 0.2 (to reduce randomness and maintain response stability)
- Max Tokens: 1024
- Model: gpt-4o-mini via OpenAI API (June 2025 release) (Microsoft Learn, 2025)
- Input Format: Structured prompt containing either OCR-extracted or user-submitted essay text with MUET rubric.
- Output Format: JSON-formatted response including two main scores and a detailed justification

The AI-generated scores are then mapped to the internal database and compared against lecturer-assigned scores for evaluation and visualization. The system does not involve fine-tuning of the GPT-4o mini model. Instead, it uses prompt engineering to adapt the model for MUET essay scoring. This approach was selected for its scalability, practical integration, and resource efficiency that make it ideal for embedding NLP and AI into a production-ready academic assessment tool. This method ensures standardized and scalable essay assessment without the need for fine-tuning a custom model. Instead, it offers an efficient way to embed

domain knowledge into a general-purpose LLM, balancing performance with simplicity of integration.

4.5 Summary

This chapter covered in details of the implementation process of the Automated Essay Grading System, while also highlighting the tools use in the system development, the setup configurations, and the developed system modules. The backend of the project is build using Django and is integrated with Azure AI services for OCR and essay feedback. Each interface was designed with user experience in mind, ensuring lecturers can easily upload essays, receive detailed AI-generated feedback, and manage their essay submissions. Core functionalities such as login, registration, essay uploading, OCR processing, score breakdown, feedback display, and account management have been successfully implemented, despite encountering some obstacles during development.

Chapter 5: Testing

5.1 Introduction

Testing phase is a crucial phase in software development that ensures the overall system can functions correctly, meets the user expectations, and doesn't have any critical defects. For the Automated Essay Grading System, the testing is conducted to verify its core features such as essay upload, OCR text extraction, and AI-generated feedback, as well as to assess the usability and user acceptance of the platform among lecturers. The chapter also covers the testing process, including functional testing using predefined test cases, usability evaluations to assess user experience, and user acceptance testing (UAT) gathered through structured feedback via Google Forms. The results helped in identifying and fixing bugs, improving the interface, and validating system readiness.

5.2 Testing Setup

The testing was carried out in a controlled development environment, and it incorporated both technical and user-centric components to ensure a comprehensive evaluation of the system's functionality, usability, and performance. The setup included essential tools, platforms, and participant roles to facilitate structured testing and accurate feedback collection. The testing was carried out in the following environment:

Table 15: Testing Setup Environment

Category	Details
Platform	Localhost server using Django development environment
Browser	Microsoft Edge
Operating System	Windows 11, Window 10
Development Tool	Visual Studio Code, Python, Django and Azure SDK libraries
Testers	Lecturer, Pilot Tester
Feedback Collection	Google Form for usability testing and UAT responses
Bug Reporting	Manual notes and screenshots captured during testing sessions

5.3 Functional Testing

Functional testing was conducted to ensure that the AEGS system performs correctly based on the specified requirements and user expectations. This testing focuses on verifying the behavior of each core module, such as user authentication, essay management, AI scoring, and dashboard functionality. All key features were tested through structured test cases that outline input data, expected outcomes, and actual results.

5.3.1 Testing Test Cases

The following test cases were designed to validate the core modules of the system:

Table 16: Test Case for Registration Functionality

Project Name: Automated Essay Grading System (AEGS)			**Test Designed by:** Alexander Anak Adrian		
Module Name: User Authentication – Registration			**Test Designed Date:** 17 June 2025		
Test Title: To verify the registration functionality with valid information					
Test ID	**Test Case**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
RG001	1	Enter valid name, email, password, confirm password. Click 'Register'.	User is registered and redirected to login page.	User successfully registered and redirected.	Pass
Test Title: To verify registration functionality with mismatched passwords					
RG002	2	Valid name, email, password ≠ confirm password. Click 'Register'.	Error shown: “Passwords do not match.”	Error message displayed.	Pass

Test Title: To verify registration functionality with existing email.					
RG003	3	Enter already registered email. Click 'Register'.	Error shown: “This email is already registered.”	System prevents duplicate email.	Pass
Test Title: To verify registration functionality without entering any data					
RG004	4	Leave all fields blank. Click 'Register'.	All fields show validation errors.	Error messages shown as expected.	Pass

Table 17: Test Case for Login Functionality

Project Name: Automated Essay Grading System (AEGS)			**Test Designed by:** Alexander Anak Adrian		
Module Name: User Authentication – Login			**Test Designed Date:** 17 June 2025		
Test Title: To verify login functionality with valid credentials					
Test ID	**Test Case**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
LG001	1	Enter correct email and password. Click 'Login'.	User logged in and redirected to dashboard.	User logged in successfully.	Pass
Test Title: To verify login functionality with invalid credentials					
LG002	2	Wrong email or password. Click 'Login'.	Error message: “Invalid username/email or password.”	Error message displayed.	Pass
Test Title: To verify login functionality with blank input					
LG003	3	Leave both fields empty. Click 'Login'.	Error message asking user to fill out the field.	Error messages appear.	Pass

Table 18: Test Case for Dashboard Graph Functionality

Project Name: Automated Essay Grading System (AEGS)			**Test Designed by:** Alexander Anak Adrian		
Module Name: Dashboard			**Test Designed Date:** 17 June 2025		
Test Title: To verify dashboard functionality by viewing graded essay graph					
Test ID	**Test Case**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
DB001	1	Open dashboard	Line graph shows current month's data	Graph loads and shows monthly total	Pass
Test Title: To verify dashboard functionality by changing graph to different month					
DB002	2	Select another month from filter	Graph updates to reflect data of selected month	Graph displays new data correctly	Pass

Table 19: Test Case for Essay Upload Functionality

Project Name: Automated Essay Grading System (AEGS)			**Test Designed by:** Alexander Anak Adrian		
Module Name: Essay Submission			**Test Designed Date:** 17 June 2025		
Test Title: To verify upload essay functionality with typed content					
Test ID	**Test Case**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
UP001	1	Enter title, student name, paste text. Click 'Submit'.	Essay saved and shown in Essay List.	Essay added successfully.	Pass
Test Title: To verify upload essay functionality via image (OCR)					
UP002	2	Title, student name, upload image. Click 'Submit'.	OCR extracts text, essay saved and listed.	OCR processed and saved.	Pass

Test Title: To verify upload essay functionality with missing fields					
UP003	3	Submit with empty title or name.	Required field warning shown.	Error messages shown.	Pass
Test Title: To verify upload essay functionality with unreadable image.					
UP004	4	Blurry image file. Click 'Submit'.	OCR fails	OCR result produce garbled or nonsensical text.	Pass

Table 20: Test Case for Essay Management

Project Name: Automated Essay Grading System (AEGS)			**Test Designed by:** Alexander Anak Adrian		
Module Name: Essay Management			**Test Designed Date:** 17 June 2025		
Test Title: To verify essay management functionality by filtering essay list by year					
Test ID	**Test Case**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
EM001	1	Select year from dropdown	List updates to show only essays from that year	Year filter works correctly	Pass
Test Title: To verify essay management functionality by filtering essay list by status					
EM002	2	Select "Completed", "Processing", or "Error"	List shows only essays with that status	Status filter applied successfully	Pass
Test Title: To verify essay management functionality by filter with no matching results					
EM003	3	Select filters that return nothing	No essay list display.	No essay shown, message “Showing 0 of 0” displayed	Pass
Test Title: To verify essay management functionality by search by essay title					
EM004	4	Enter full or partial essay title	Matching essay(s) shown in list	Search displays correct result	Pass

Test Title: To verify essay management functionality by viewing essay result					
EM005	5	Click on essay marked 'Completed'.	Complete essay information shown with essay content	All essay information with essay content displayed.	Pass
Test Title: To verify essay management functionality by viewing essay score breakdown					
EM006	6	Click on essay marked 'Completed'. Click on ‘Score Breakdown’ button.	Complete AI essay score breakdown shown with AI score justification	AI essay score breakdown with AI score justification displayed.	Pass
Test Title: To verify the essay result functionality by editing essay result data					
EM007	7	Update essay title or student name. Click 'Save'.	Updated info is saved and displayed.	Essay data updated successfully.	Pass
Test Title: To verify the essay result functionality by editing score breakdown					
EM008	8	Click 'Edit', change scores, then save.	Scores updated and display.	Score changes saved and shown.	Pass
Test Title: To verify the essay result functionality by deleting an essay					
EM009	9	Click 'Delete' on Essay List. Confirm deletion.	Essay removed.	Essay deleted successfully.	Pass

Table 21: Test Case for Account Info Management

Project Name: Automated Essay Grading System (AEGS)	**Test Designed by:** Alexander Anak Adrian
Module Name: User Profile Management	**Test Designed Date:** 17 June 2025
Test Title: To verify account management functionality by viewing account info	

Test ID	Test Case	Input Data	Expected Result	Actual Result	Status
AC001	1	Go to Account Info page.	Name, email displayed.	Info shown correctly.	Pass
Test Title: To verify account management functionality by editing account info					
AC002	2	Change name or email. Click 'Save'.	Changes saved and reflected.	Info updated successfully.	Pass
Test Title: To verify account management functionality by submit blank field					
AC003	3	Clear all fields. Click 'Save'.	Validation error shown.	Errors handled properly.	Pass

Table 22: Test Case for Logout Functionality

Project Name: Automated Essay Grading System (AEGS)			**Test Designed by:** Alexander Anak Adrian		
Module Name: User Authentication - Logout			**Test Designed Date:** 17 June 2025		
Test Title: To verify logout functionality by login out user from system					
Test ID	**Test Case**	**Input Data**	**Expected Result**	**Actual Result**	**Status**
LO001	1	Click on 'Logout' button	User session ends, redirected to landing page	Logout successful, landing page displayed	Pass

In addition to the detailed test cases, a System Module Testing Checklist was prepared to ensure all major features and functionalities of the AEGS system were tested across different modules. This checklist summarizes the key components tested, along with observations and outcomes recorded during the testing sessions. The full checklist is included in Appendix.

5.4 Non-Funtional Testing

Non-functional testing in this project focuses on usability and user acceptance. While the core system features were tested functionally, it was also important to assess how real users (lecturers) interacted with the Automated Essay Grading System (AEGS) and whether they found it intuitive, efficient, and reliable. To achieve this, a Google Form was distributed to language teacher and pilot tester (student) to gather structured feedback. The form included Likert-scale evaluations (1 to 5) and open-ended questions that focused on aspects like ease of use, AI feedback quality, design aesthetics, and overall satisfaction. This allowed for a qualitative and quantitative evaluation of user experience. Only lecturers involved in essay marking and teaching MUET-related courses were selected to participate in the usability evaluation.

5.4.1 User Acceptance Testing (UAT)

A total of four respondents participated in the UAT, one language lecturer and three final-year software engineering students who served as testers was invited to perform real user testing by completing specific tasks such as registering, uploading essays (both typed and image-based), and reviewing results. The objective of the UAT was to validate whether the AEGS platform meets the functional, usability, and user workflow expectations of its target users.

Tasks Given:

- Register a new account
- Login into system
- Upload a typed essay
- Upload an essay using an image (OCR)
- Review the generated feedback (score & justification)
- Edit essay content and information

- Attempt to delete an essay entry
- Navigate the dashboard and view the graph
- Edit score breakdown for an essay
- View account information
- Update account information
- Delete account information

Feedback Summary (Based on Google Form Responses in Appendix):

User Acceptance Testing (UAT) was conducted using Google Forms. The goal was to gather insightful feedback on the proposed Automated Essay Grading System. The survey was divided into four sections: tester information, core module satisfaction ratings, system usability and accuracy, and comparative value with open-ended feedback. A total of four respondents participated one lecturer and three student testers. Their responses, included in the appendices, provided valuable insights that contributed meaningfully to the system's development.

Most testers found the system easy to navigate (avg. score: 4.5/5), with the essay upload process rated smooth and simple (4/5). AI-generated scores and justifications were seen as helpful, though one suggestion was made to support multiple essay uploads and post-edit analysis. Visual tools like the dashboard and filters were praised for effectively summarizing student performance. No major bugs were reported. Overall, the UAT confirmed that the system is functional, user-friendly, and aligns with the marking and grading workflow of language lecturers. Additionally, user satisfaction was notably high, with testers consistently rating the system positively across key areas such as usability, accuracy, and feature relevance.

5.4.2 Accuracy Analysis: AI vs Human Scoring

To evaluate the accuracy and reliability of the AI-generated essay scores, a dataset of 20 complete MUET essays was compared against human-assigned scores provided by a trained MUET examiner. The scoring focused on two main components from the MUET rubric: (1) Task Fulfilment & Content (TFC), and (2) Language & Organization (LO). Each essay was first processed using Azure Document Intelligence for OCR extraction, then submitted to the GPT-4o mini model via a structured prompt designed to replicate the official rubric.

The following performance metrics were used for evaluation:

- Mean Absolute Error (MAE): The average absolute difference between the AI-generated scores and lecturer-assigned scores was 2.25 points. This relatively low MAE indicates that the AI model consistently assigns scores that are close to human evaluations, suggesting reliable performance in automated grading.
- Within ±5 Point Range: 100% of the AI-generated scores fell within a ±5 point range of the lecturer's scores. This perfect alignment suggests that the AI model demonstrates strong agreement with human judgment in terms of general scoring consistency, even if not matching exactly on every essay.
- Standard Deviation: The standard deviation of the score differences was 1.69, reflecting a reasonably tight clustering of discrepancies. A low standard deviation means that most differences between AI and human scores were small and consistent, rather than erratic or widely spread.

As illustrated in Figure 5.4.2.1, the bar chart shows that AI-assigned scores closely follow the trend of lecturer-assigned scores, indicating strong alignment across multiple essays. Meanwhile, Figure 5.4.2.2 visualizes the score difference per essay, where all differences fall

within the acceptable ±5 point tolerance range. This reinforces the reliability of the AI model for practical academic use, especially where slight variation is tolerable. These results suggest that the system performs with high reliability, particularly within practical grading tolerance margins accepted by MUET examiner.

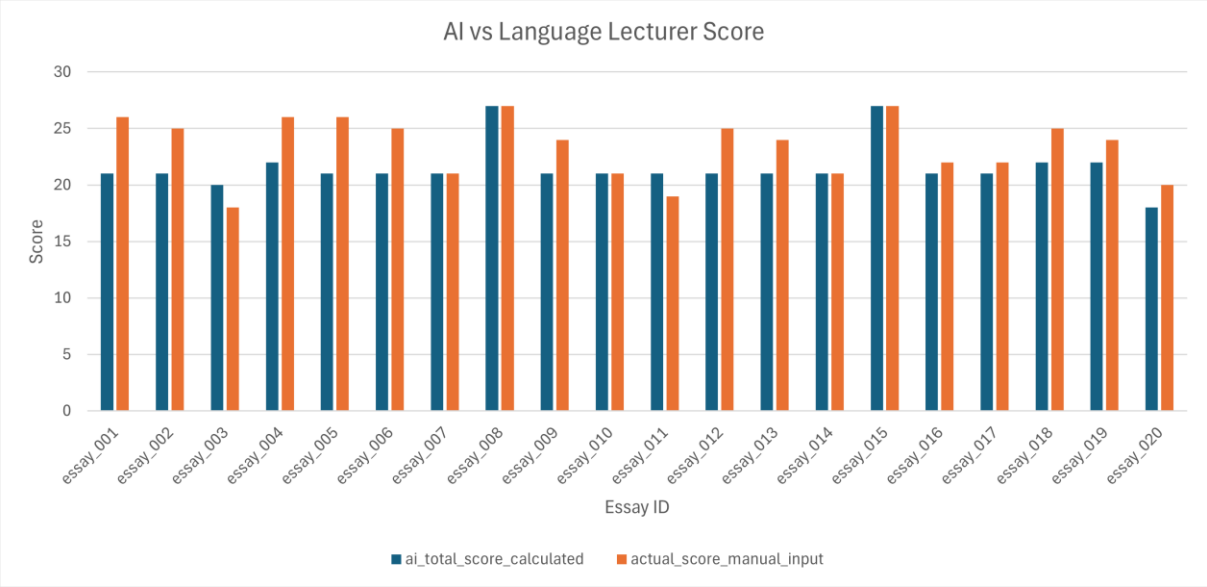


Figure 5.1.2.1: Comparison of AI vs Human Scores

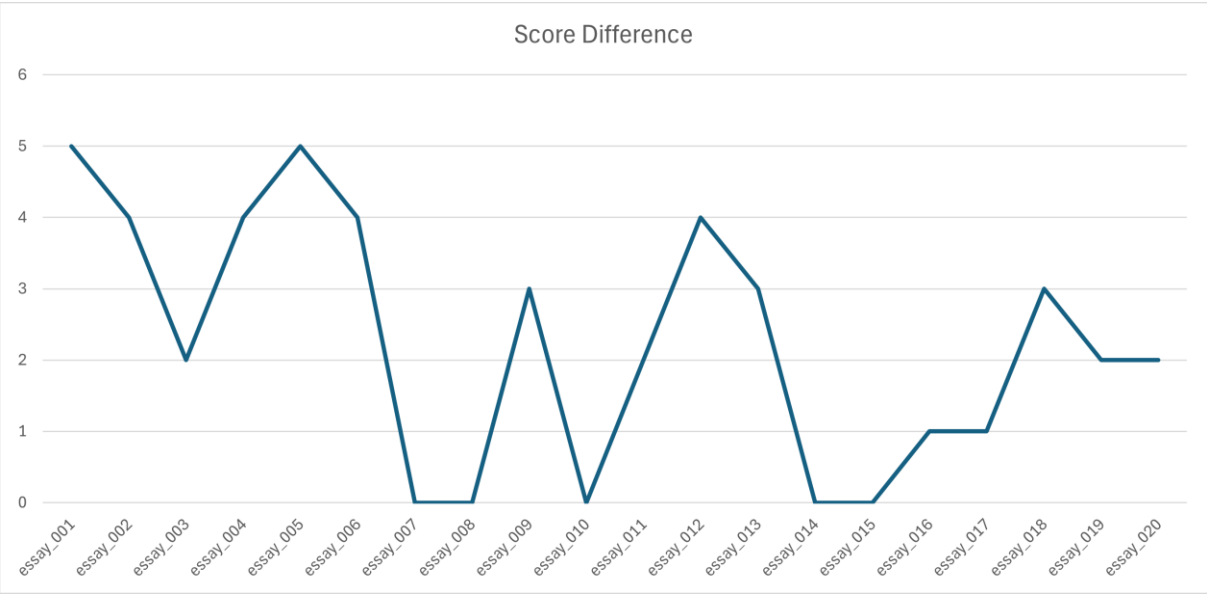


Figure 5.2.2.2: Score Difference between Essay

5.5 Discussion

The testing process for AEGS has demonstrated that the system successfully meets its functional requirements and shows promising performance in usability and user satisfaction. Functional testing confirmed that all major features including user registration, essay upload, AI scoring, and result display perform as expected.

Furthermore, non-functional testing, particularly through usability evaluation and UAT, highlighted the system's strength in providing a clean interface and intuitive interactions. When testing with FELC found the system easy to learn and use, requiring minimal guidance, which is a crucial aspect for adoption in academic settings.

The OCR functionality, powered by Azure Document Intelligence, performed well even with handwritten essays, although its accuracy still depends on image clarity. The integration of AI scoring through OpenAI allowed for instant feedback generation, which testers described as a potential time-saving feature.

The few suggestions and issues reported were mostly related to enhancing system, rather than fixing core system problems. The lecture also requested the system to be able to analyse variety of essay type and not just MUET essay. This indicates a high level of system stability and readiness for future scaling or pilot deployment in MUET-related classrooms.

5.6 Summary

This chapter presented a comprehensive evaluation of the Automated Essay Grading System (AEGS) through both functional and non-functional testing methods. Functional testing using predefined test cases confirmed that all system modules such as registration, login, essay upload, scoring, and account management operate correctly. Non-functional testing, including usability testing and user acceptance testing (UAT), demonstrated that the system is well-received by its intended users. Testers found the interface easy to use and the AI-generated scores meaningful and helpful. Feedback gathered from Google Forms provided constructive insights into improving the system's usability and user confidence. The outcomes of this testing phase validate that AEGS is ready for real-world academic environments, with only minor enhancements recommended for future iterations.

Chapter 6: Conclusion and Future Works

6.1 Introduction

This chapter summarizes the key outcomes of the Automated Essay Grading System (AEGS) project and reflects on how the initial objectives were achieved, identifies the limitations encountered during development, and suggests future enhancements that could further improve the system. The aim is to provide a closing reflection on the work done and to explore possibilities for extending the system's capability beyond its current version.

6.2 Objective & Achievement

Automated Essay Grading System (AEGS) manage to accomplish its objectives and fulfil the requirements which were proposed during the start of the project report. The objective and the accomplishment of the system is shown in Table X below.

Table 23: Objective and Achievement of Automated Essay Grading System

Objective	Achievements
To develop a web application system that integrates OCR to accurately scan essays	AEGS successfully integrates Optical Character Recognition (OCR) using Azure Document Intelligence, enabling the system to extract text from uploaded handwritten or scanned essay images. The extracted content is automatically compiled and displayed in the essay view, reducing manual input and

	supporting flexible essay submission formats.
To apply artificial intelligence models to analyze and score essays based on predefined rubrics and provide detailed justification for each assigned score.	The system utilizes OpenAI's language models to perform essay analysis and scoring. Each essay is evaluated based on predefined rubric components, producing sub-scores and a total score. A justification is also generated for each result, helping lecturers understand how the AI derived each score and supporting transparency in automated grading.
To evaluate the user satisfaction of the automated essay grading system.	User acceptance testing was conducted through structured tasks and a Google Form distributed to lecturers. The results showed that users found the system intuitive, functional, and relevant to their workflow. Feedback confirmed that AEGS offers practical support for academic essay grading and has potential for broader classroom use.

6.3 Limitation

Despite the system's success in meeting its core objectives, several limitations were identified during the testing and development phases:

- Limited AI Scoring Scope

  The AI model was trained and fine-tuned primarily on MUET essay structure and rubrics. As such, it may not generalize well to other writing styles or types of academic writing without additional dataset tuning.

- Single-Section Coverage

  The current version of AEGS only supports grading for the writing component (essay section) of MUET. Other language skills such as listening, reading comprehension, and speaking are not yet supported.

- Basic UI Design

  While the interface is fully functional, it can still be improved in terms of design consistency, user accessibility, and responsiveness across devices.

- Limited Security Measures

  The system lacks advanced and modern security implementations such as multi-factor authentication (MFA), activity logging, and encrypted API endpoints, which may become necessary for real-world deployment.

6.4 Future Works

Several enhancements and extensions can be made to AEGS to increase its usability, reliability, and impact:

1. Improve AI Scoring Accuracy

   Future development can involve training the AI model using a larger and more diverse dataset, including real MUET essays graded by certified markers. Fine-tuning based on feedback loops and rubric alignment will help improve accuracy and reduce scoring bias.

2. Expand to Full MUET Assessment

   The system can be extended to support additional components of the MUET examination, such as grammar quizzes, listening exercises, and even speaking evaluations using speech recognition and NLP sentiment analysis tools.

3. Enhance UI/UX Design

   The interface can be improved further with a mobile-first approach, consistent iconography, better typography, and accessibility features (such as keyboard navigation and colour-blind-friendly themes).

4. Add Advanced Security Features

   For deployment in real academic environments, the system should integrate multi-factor authentication, secure password hashing, HTTPS by default, user activity logs, and role-based access control to protect sensitive data and ensure accountability.

5. Integrate Feedback Mechanism

   A simple rating or comment feature for each graded essay can be added, allowing lecturers to validate or challenge AI-generated feedback and provide corrections that can be logged for future retraining.

6.5 Conclusion

The development of the Automated Essay Grading System (AEGS) has proven that it is possible to leverage artificial intelligence, specifically NLP and OCR technologies, to automate parts of the grading process for MUET essays. The system met all its technical and functional goals, offering a usable and reliable platform for lecturers to evaluate student essays with less manual effort.

Through AI-generated scoring and justification, lecturers gain a helpful reference in their marking decisions, and the platform provides consistent and rapid feedback. Although some limitations remain, particularly in terms of scope, UI design, and system security, the results of this project are promising. With future enhancements, AEGS could evolve into a comprehensive tool for supporting English language instruction at the Pre-University level and beyond.

References

Ambler, S. (2023, November 26). *Feature Driven Development (FDD) and Agile Modeling*. The Agile Modeling (AM) Method - Effective Strategies for Modeling and Documentation. https://agilemodeling.com/essays/fdd.htm

API platform. API Platform | OpenAI. (2024). https://openai.com/api

Azure Ai Document Intelligence. Azure AI Document Intelligence | Microsoft Azure. (2024). https://azure.microsoft.com/en-us/products/ai-services/ai-document-intelligence?msockid=00198826b79e6fda21e59d76b6096e5a

Azure ai vision with OCR and AI: Microsoft Azure. Azure AI Vision with OCR and AI | Microsoft Azure. (2024). https://azure.microsoft.com/en-us/products/ai-services/ai-vision?msockid=00198826b79e6fda21e59d76b6096e5a

Ball, A. E. (2019, September 1). *Automated-Essay-Grading-with-NLP*. GitHub. https://github.com/AlexEBall/Automated-Essay-Grading-with-NLP

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001, February). The agile manifesto.

django. (2025). *Django documentation: Django documentation*. Django Project. https://docs.djangoproject.com/en/5.2/

Gurnov, A. (2024, October 3). *What is Agile Methodology in project management?*. Versatile & Robust Project Management Software. https://www.wrike.com/project-management-guide/faq/what-is-agile-methodology-in-project-management/

How to use an Agile Gantt chart in project management. RSS. (n.d.). https://www.teamgantt.com/blog/how-to-use-gantt-charts-for-your-agile-project#

Is python supported at Hostinger?. Hostinger Help Center. (2024, October). https://support.hostinger.com/en/articles/3648030-is-python-supported-at-hostinger

Jain, S., & Thakkar, K. (2020, March 9). *Automatic-Essay-Scoring (AES)*. GitHub. https://github.com/sankalpjain99/Automatic-Essay-Scoring

Majlis Peperiksaan Malaysia. (2015). *MUET Test Specification | MPM*. Majlis Peperiksaan Malaysia (MPM). http://webmpm1.mpm.edu.my/download_MUET/MUET_Test_Specification_2015VersiPortal.pdf

Microsoft. (2021, November 3). *Visual studio code - code editing. redefined*. RSS. https://code.visualstudio.com/

Microsoft Learn. (2025, February 7). *Azure OpenAi in Azure AI Foundry Models - Azure OpenAI | Microsoft Learn*. Azure OpenAI | Microsoft Learn. https://learn.microsoft.com/en-us/azure/ai-foundry/openai/concepts/models?tabs=global-standard%2Cstandard-chat-completions#gpt-4o-and-gpt-4-turbo

Msangapu-Msft. (2024, September 13). *Overview of Azure App Service - Azure App Service*. Overview of Azure App Service - Azure App Service | Microsoft Learn. https://learn.microsoft.com/en-us/azure/app-service/overview#why-use-app-service

OpenAI. (2024, July 18). *GPT-4O Mini: Advancing cost-efficient intelligence | OpenAi*. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

Palmer, S. R., & Felsing, J. M. (2002). *A practical guide to feature-driven development*. Prentice Hall PTR.

Spacy · industrial-strength natural language processing in python. · Industrial-strength Natural Language Processing in Python. (2024). https://spacy.io/

Tesseract documentation. Tesseract OCR. (2024, December 6). https://tesseract-ocr.github.io/

Wei, L. T. (2019, December 24). *EssayScore: Automated Essay Scoring with Deep Learning (Final Year Project)*. GitHub. https://github.com/tingwei3931/EssayScore_FYP

Zhezherau, A. (2024, June 11). *What is FDD in agile?: Wrike Agile Guide*. Versatile & Robust Project Management Software. https://www.wrike.com/agile-guide/faq/what-is-fdd-in-agile/

Appendixes

Figure A.1: Gantt Chart of Automated Essay Grading System for FYP 1

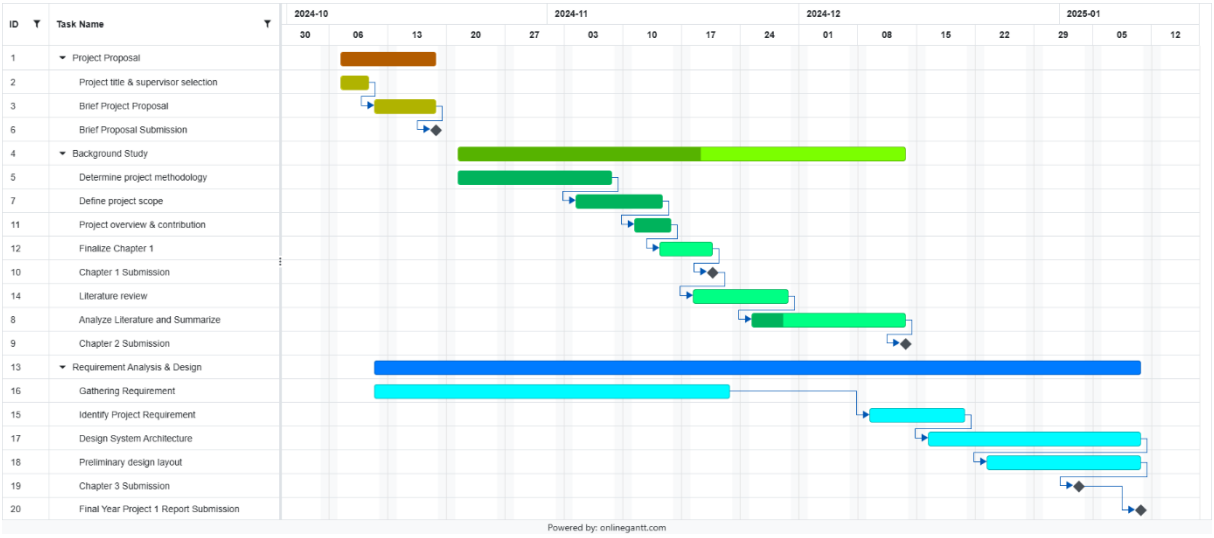


Figure A.2: Gantt Chart of Automated Essay Grading System for FYP 2

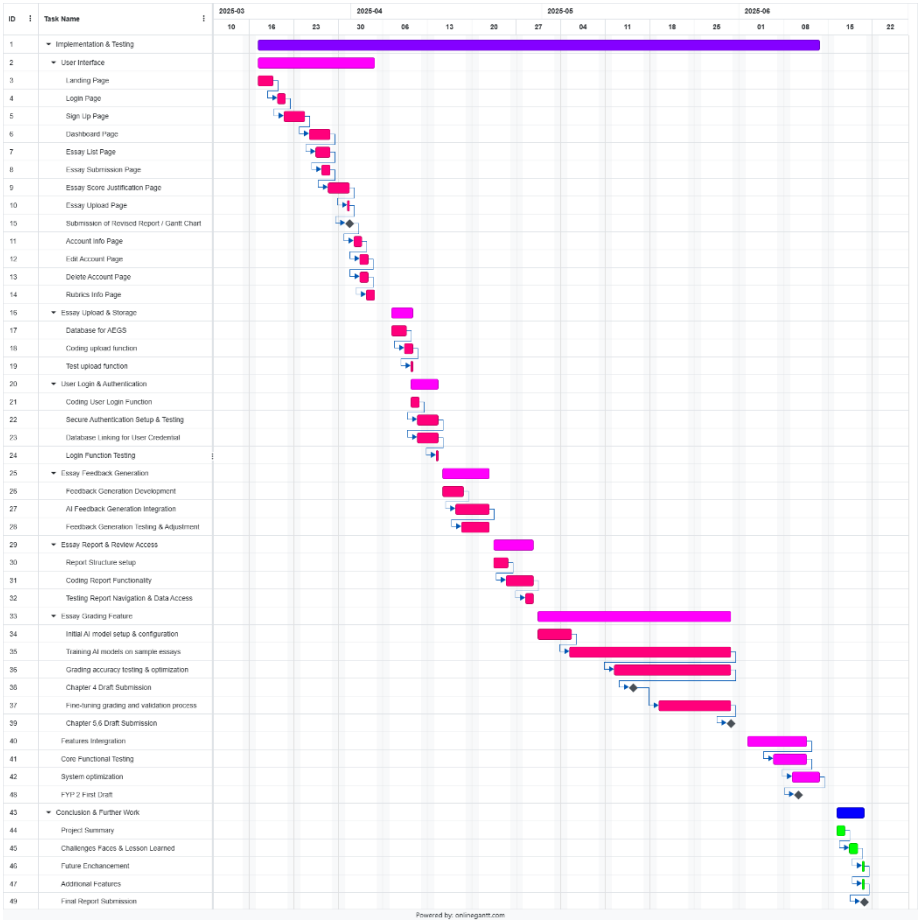


Figure A.3: Survey on Demographic Information via Google Forms

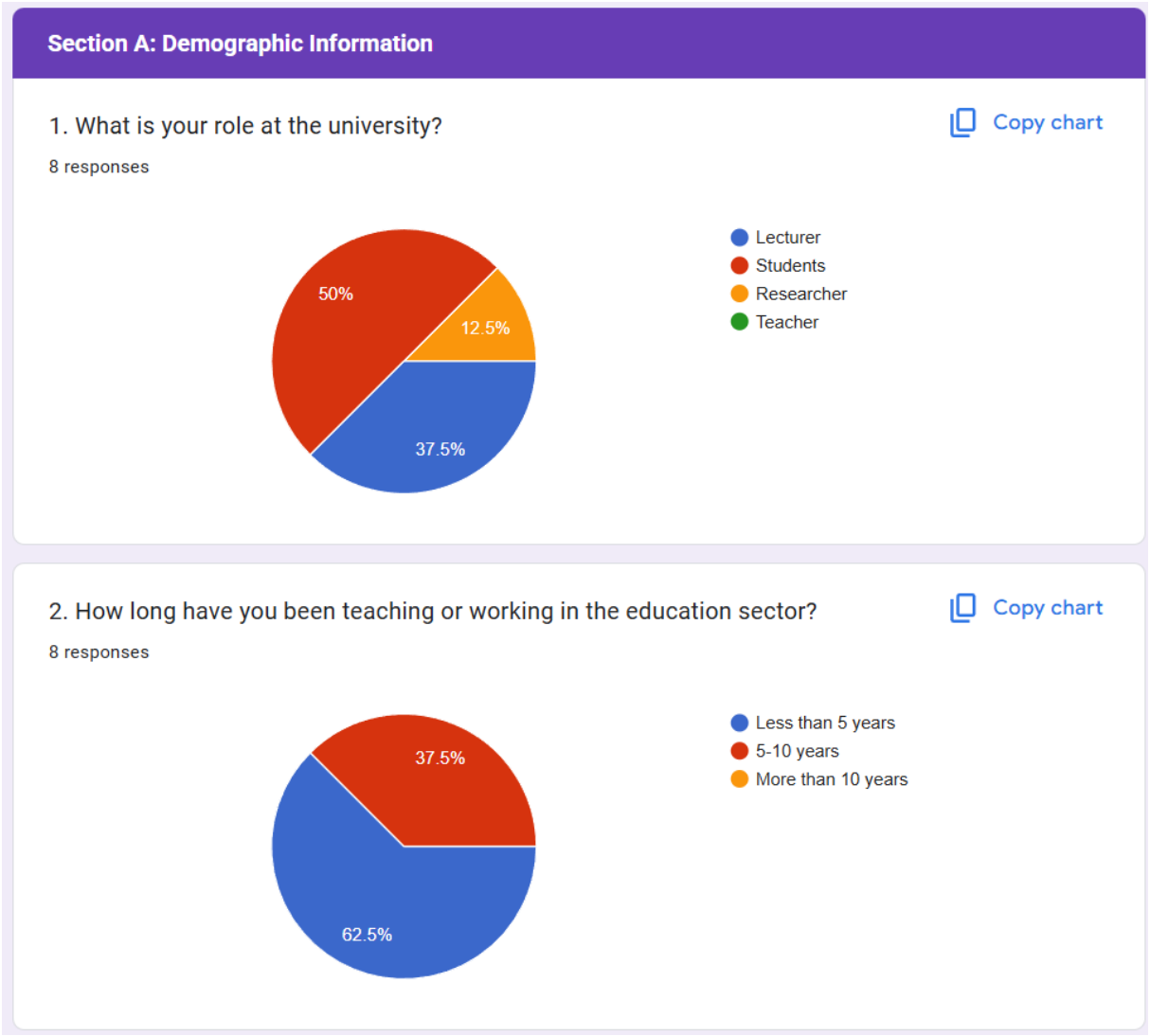


Figure A.4: Survey on Awareness and Perception of AI via Google Forms (1)

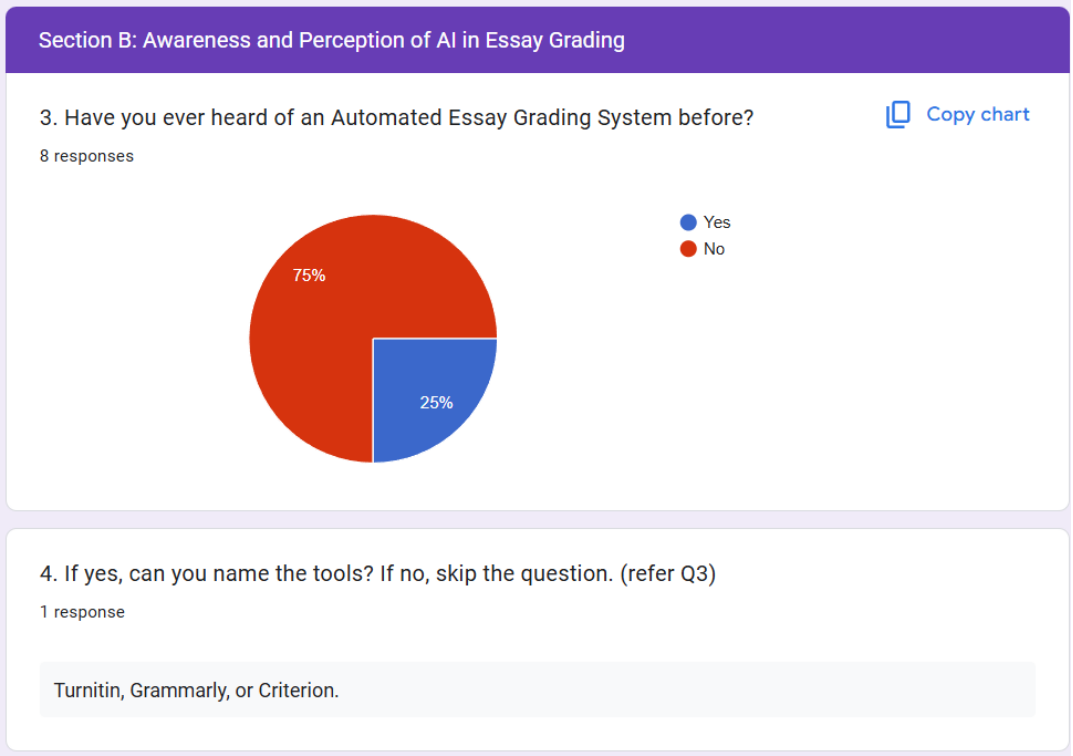


Figure A.5: Survey on Awareness and Perception of AI via Google Forms (2)

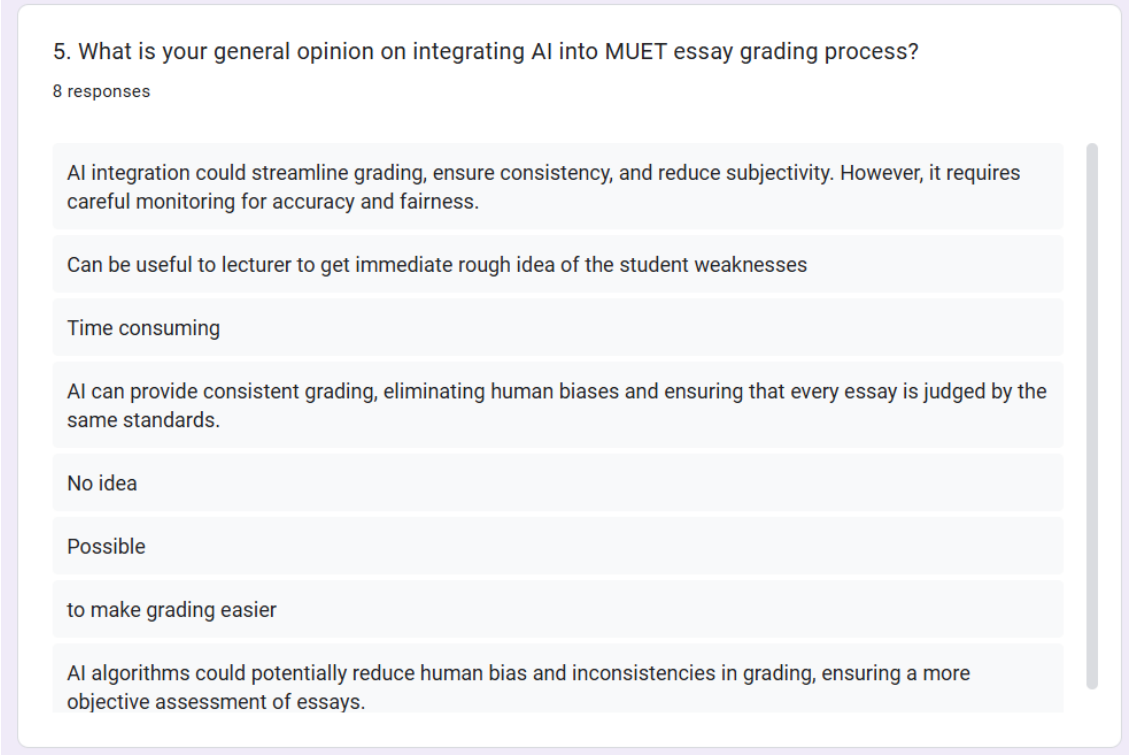


Figure A.6: Survey on Awareness and Perception of AI via Google Forms (3)

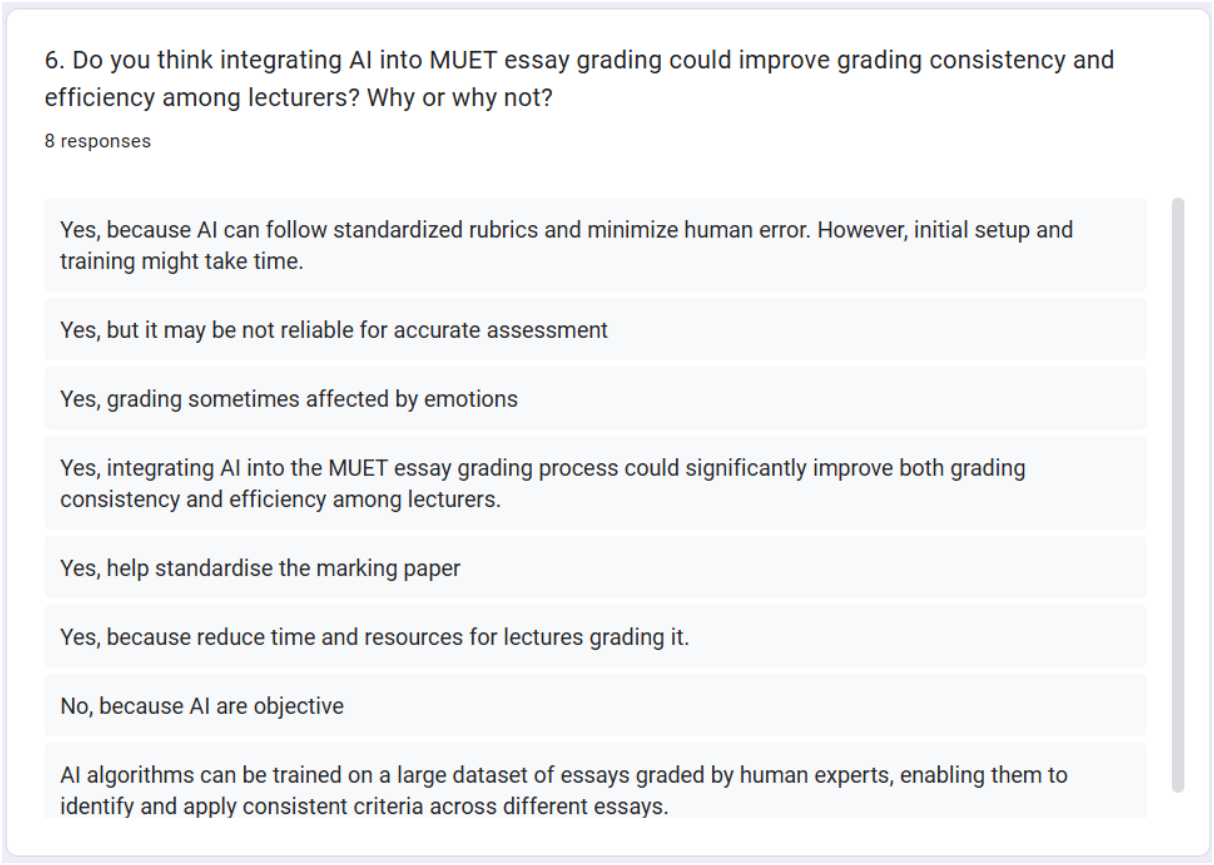


Figure A.7: Survey on Awareness and Perception of AI via Google Forms (4)

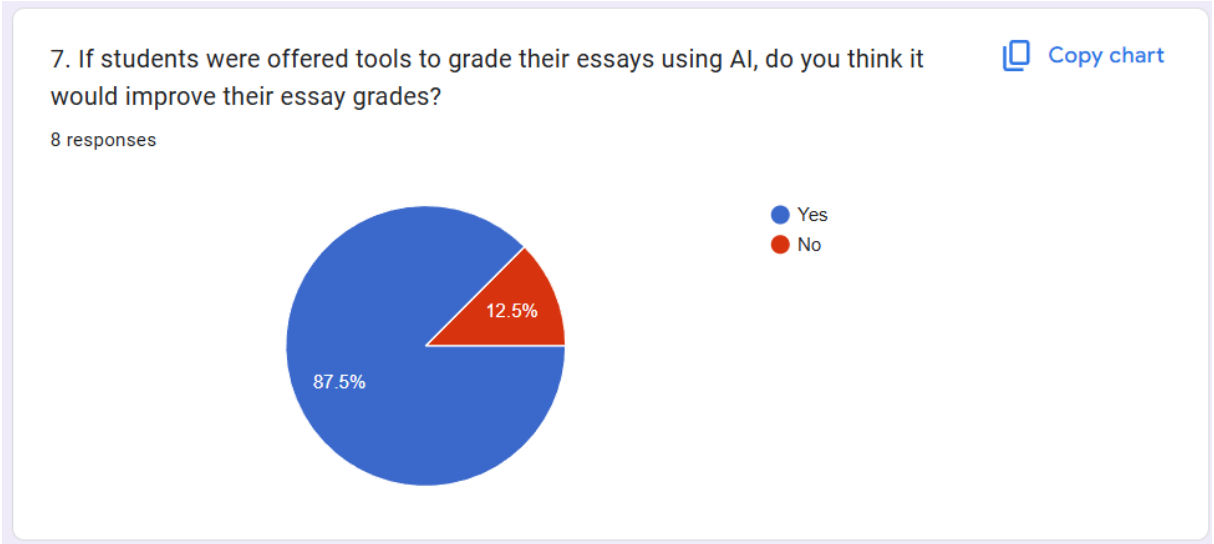


Figure A.8: Survey on Awareness and Perception of AI via Google Forms (5)

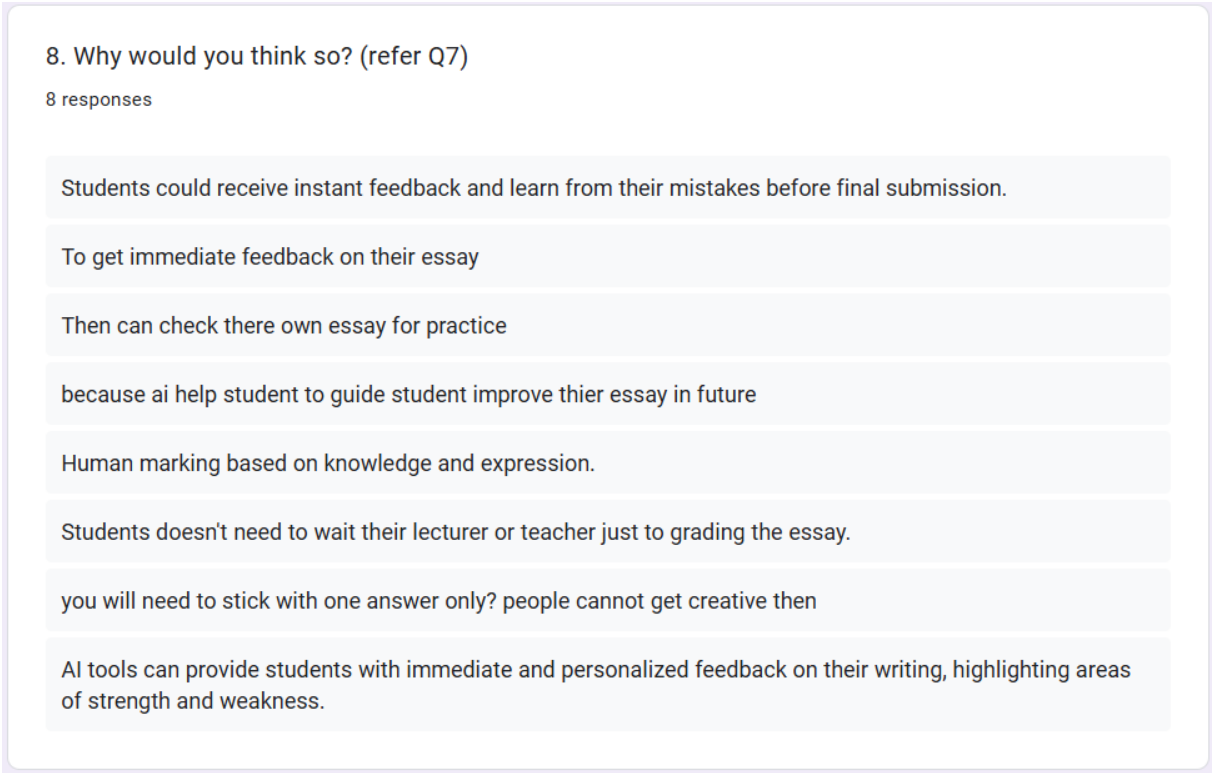


Figure A.9: Survey on Awareness and Perception of AI via Google Forms (6)

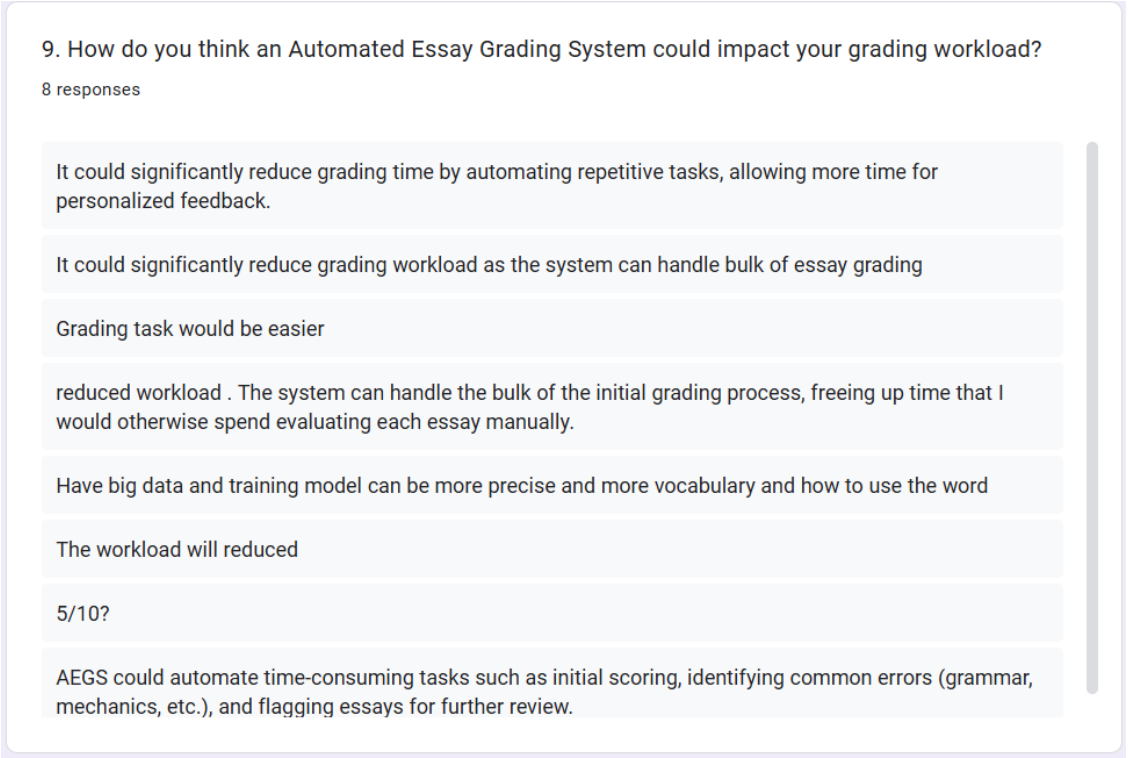
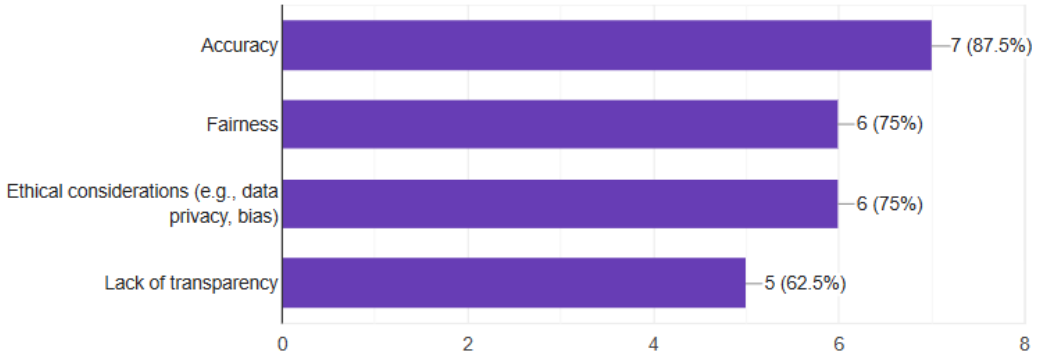


Figure A.10: Survey on Awareness and Perception of AI via Google Forms (7)

10. What are your main concerns about using AI for essay grading?

Copy chart

8 responses



11. What challenges or concerns do you foresee with implementing such a system?

8 responses

-

Ensuring fairness when evaluating student essays

The system might not give a complex evaluation like a real person do

Accuracy and Reliability: Ensuring the AI system consistently provides accurate and reliable grades is crucial. Inconsistent grading could undermine trust in the system.

The source of data , how the big the data, what model been training and how it much data been train

The time for the user to get used to it.

Evaluating essays often involves assessing complex arguments, identifying logical fallacies, and evaluating the depth of critical thinking. AI algorithms may not be sophisticated enough to perform these tasks reliably.

Figure A.11: Survey on Prototype Feedbacck via Google Forms (1)

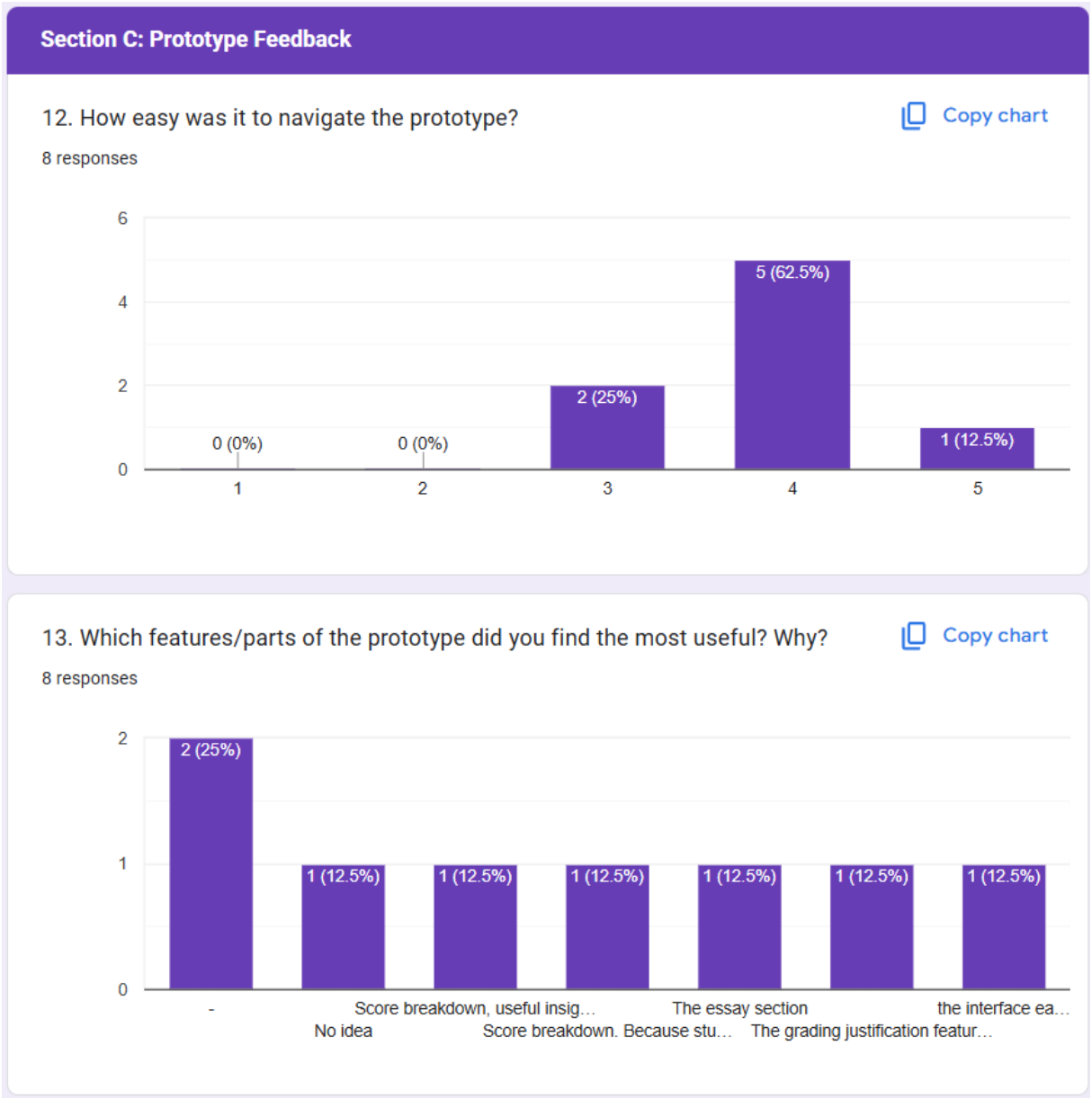


Figure A.12: Survey on Prototype Feedbacck via Google Forms (2)

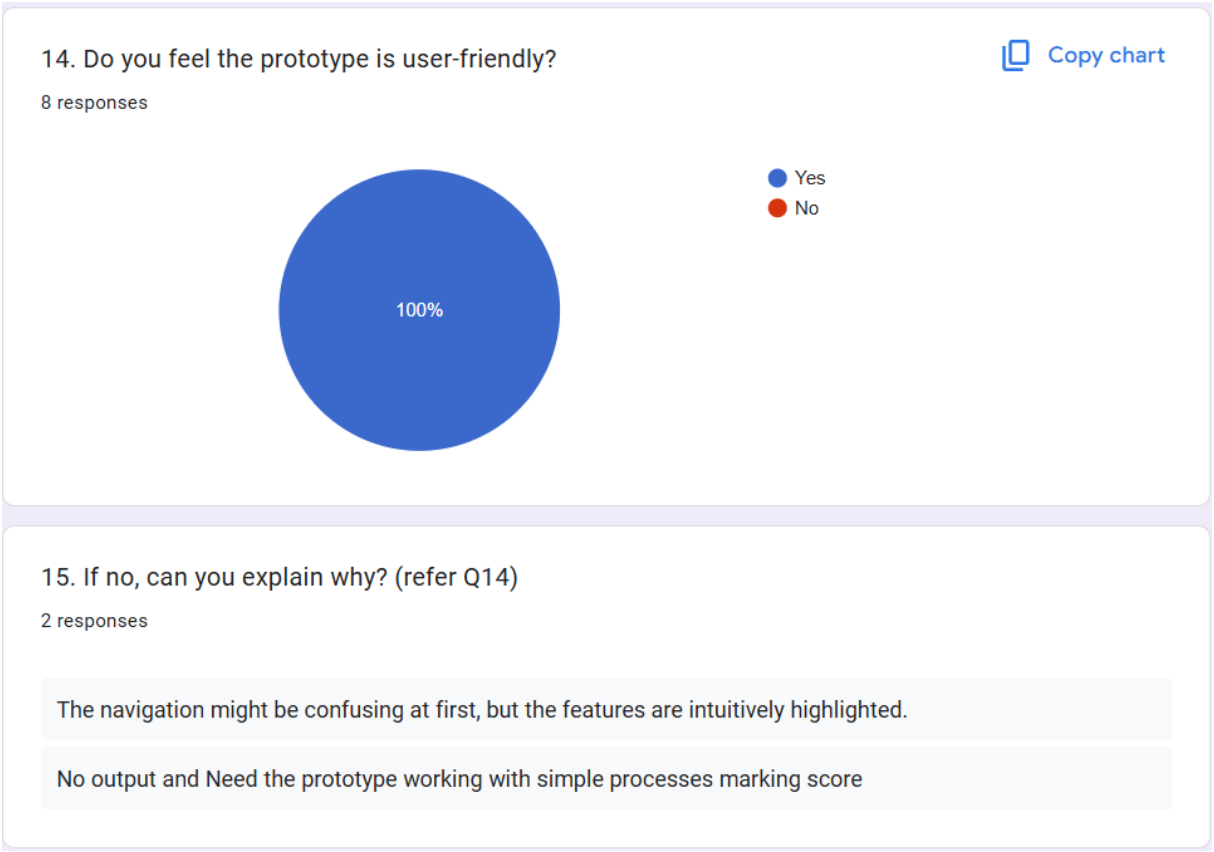
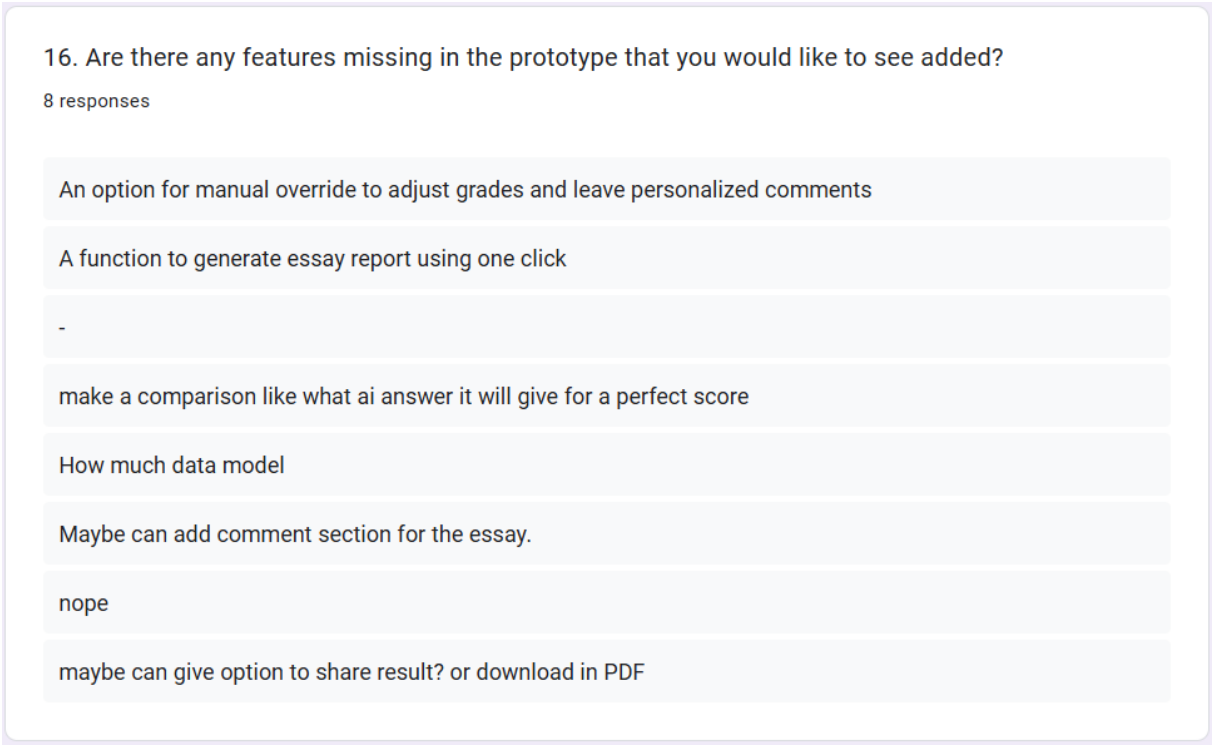


Figure A.13: Survey on Prototype Feedbacck via Google Forms (3)


16. Are there any features missing in the prototype that you would like to see added?

8 responses

An option for manual override to adjust grades and leave personalized comments

A function to generate essay report using one click

-

make a comparison like what ai answer it will give for a perfect score

How much data model

Maybe can add comment section for the essay.

nope

maybe can give option to share result? or download in PDF

Figure A.14: Survey on Adoption and Improvement via Google Forms (1)

Figure A.15: Survey on Adoption and Improvement via Google Forms (2)

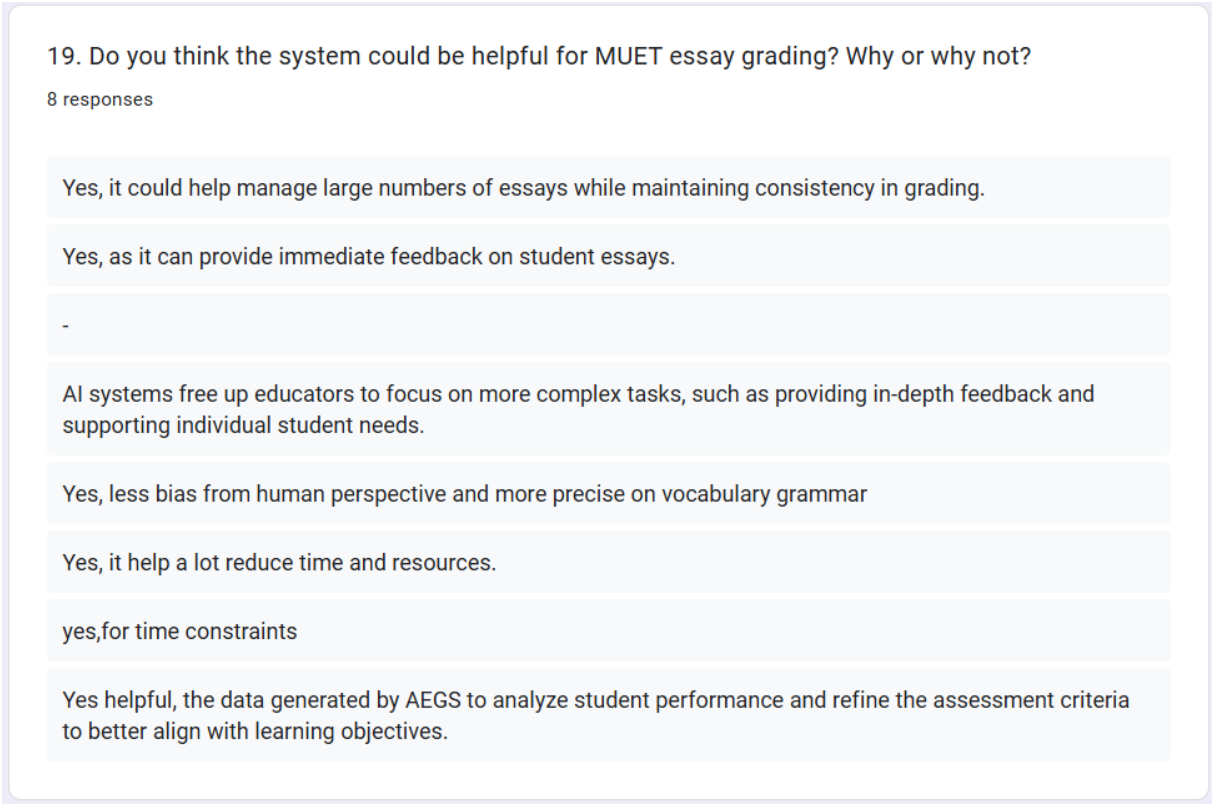


Figure A.16: Survey on Adoption and Improvement via Google Forms (3)

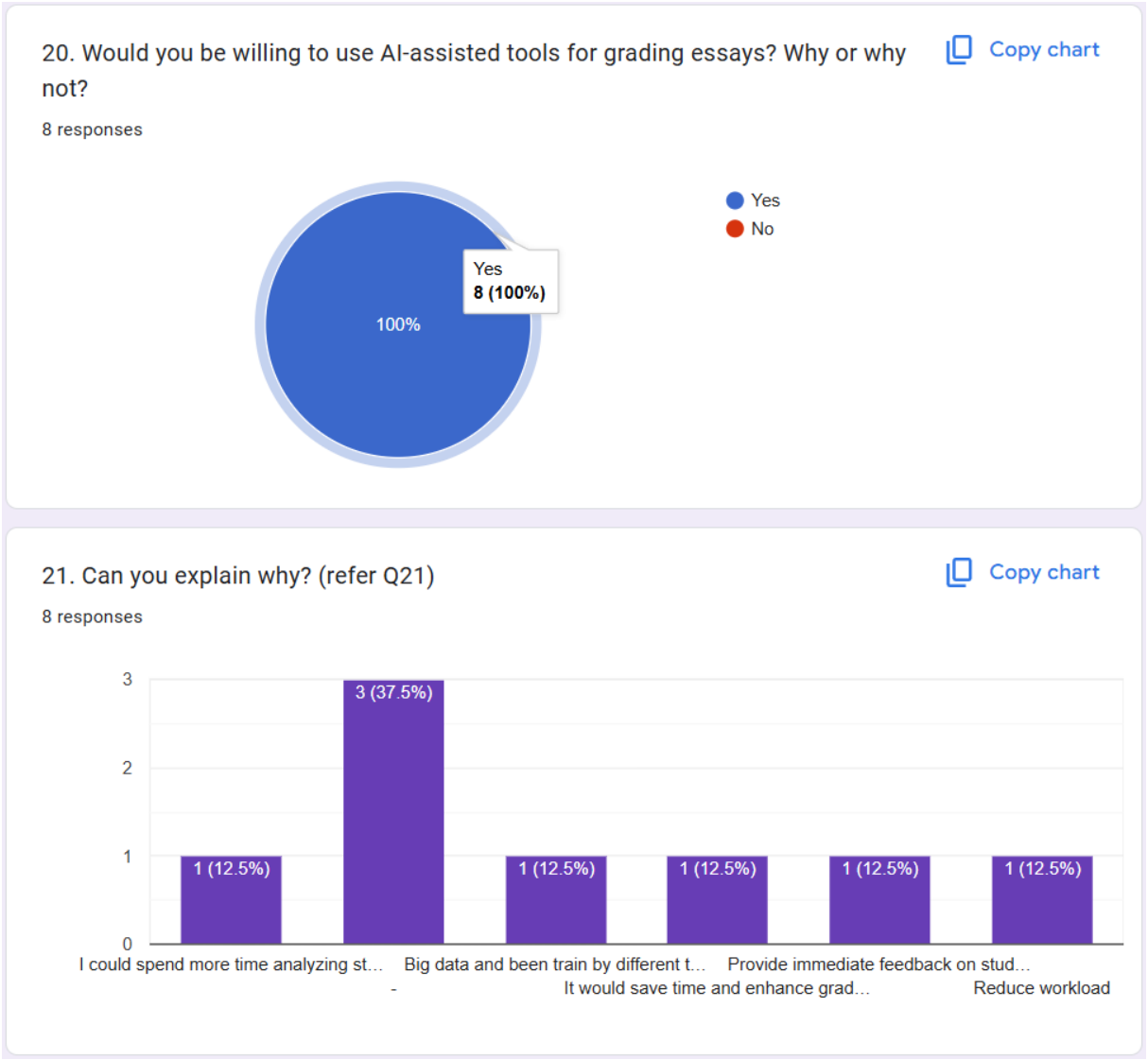


Figure A.17: Survey on Adoption and Improvement via Google Forms (4)

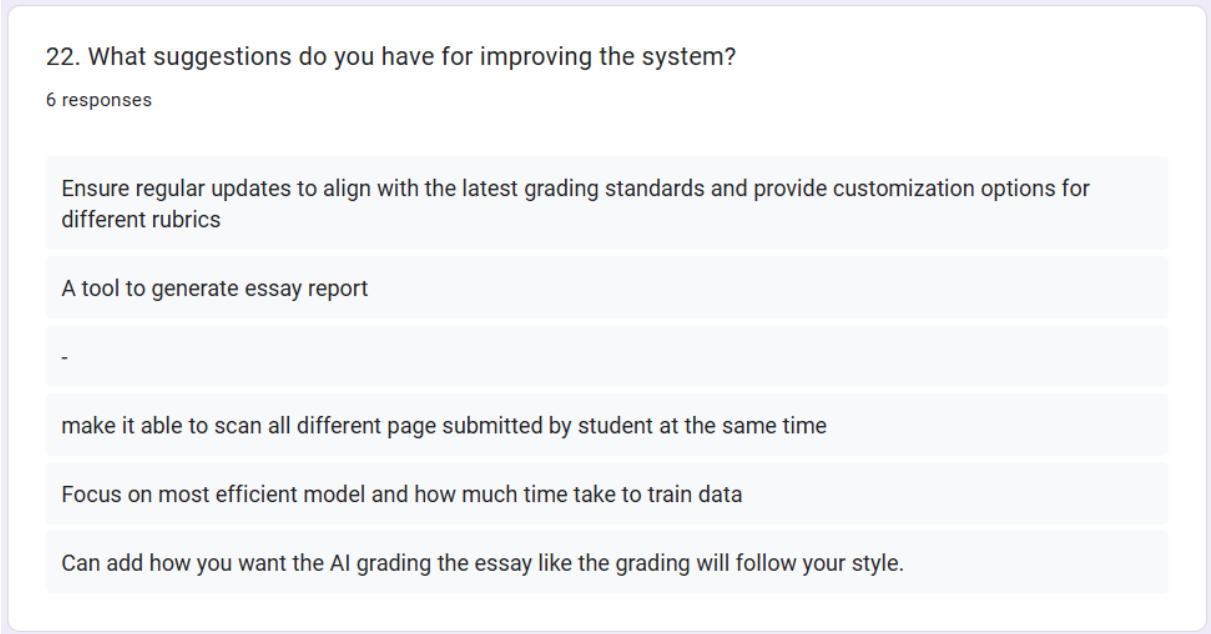


Figure A.18: AEGS User Acceptance Testing (UAT) survey (1)

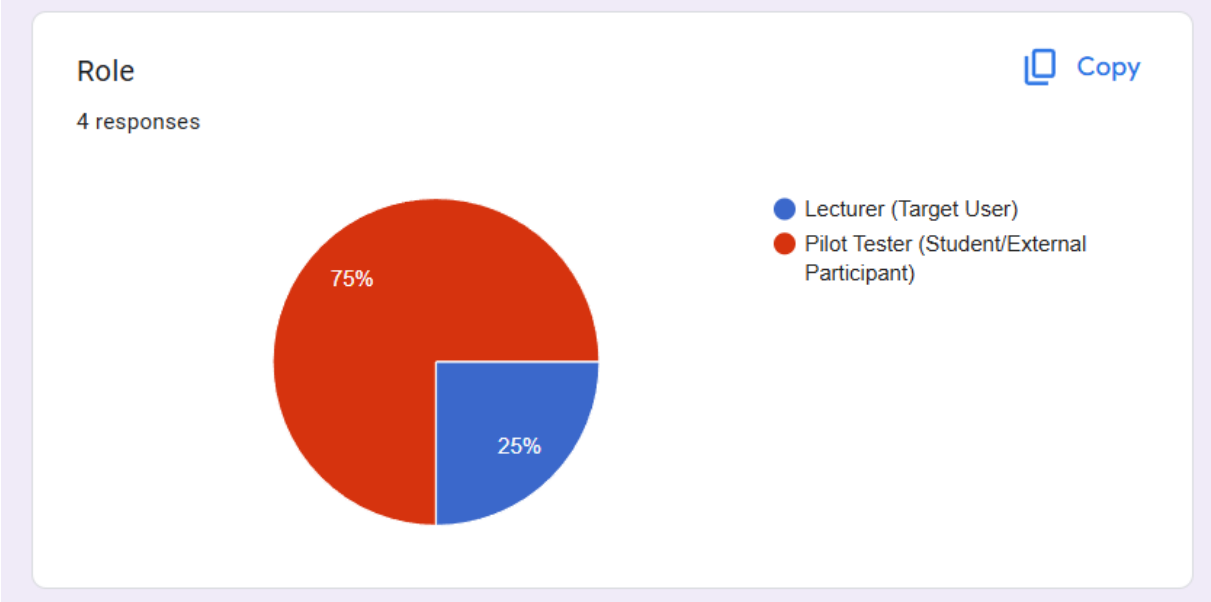


Figure A.19: AEGS User Acceptance Testing (UAT) survey (2)

Figure A.20: AEGS User Acceptance Testing (UAT) survey (3)

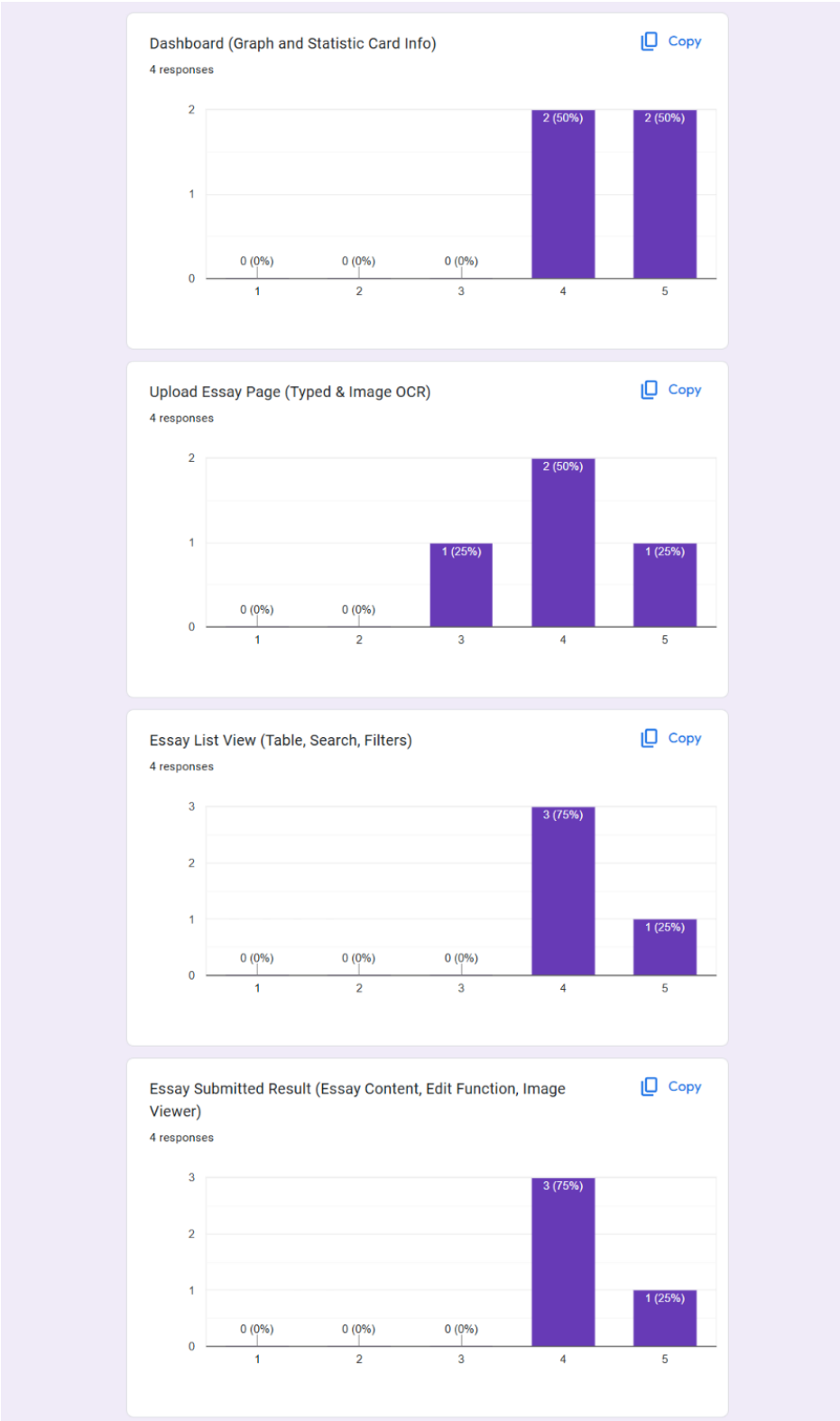


Figure A.21: AEGS User Acceptance Testing (UAT) survey (4)

Figure A.22: AEGS User Acceptance Testing (UAT) survey (5)

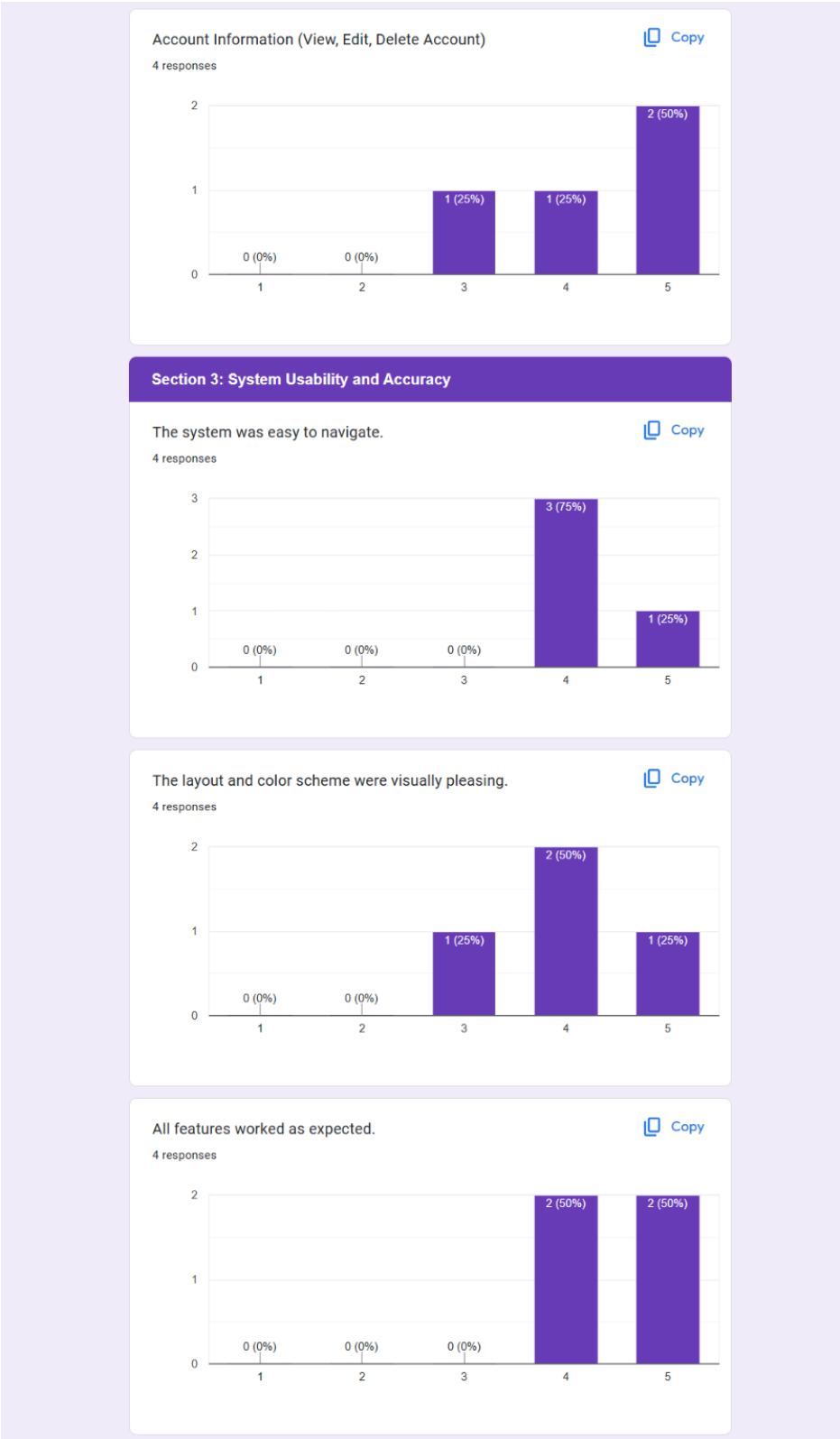


Figure A.23: AEGS User Acceptance Testing (UAT) survey (6)

Figure A.24: AEGS User Acceptance Testing (UAT) survey (7)

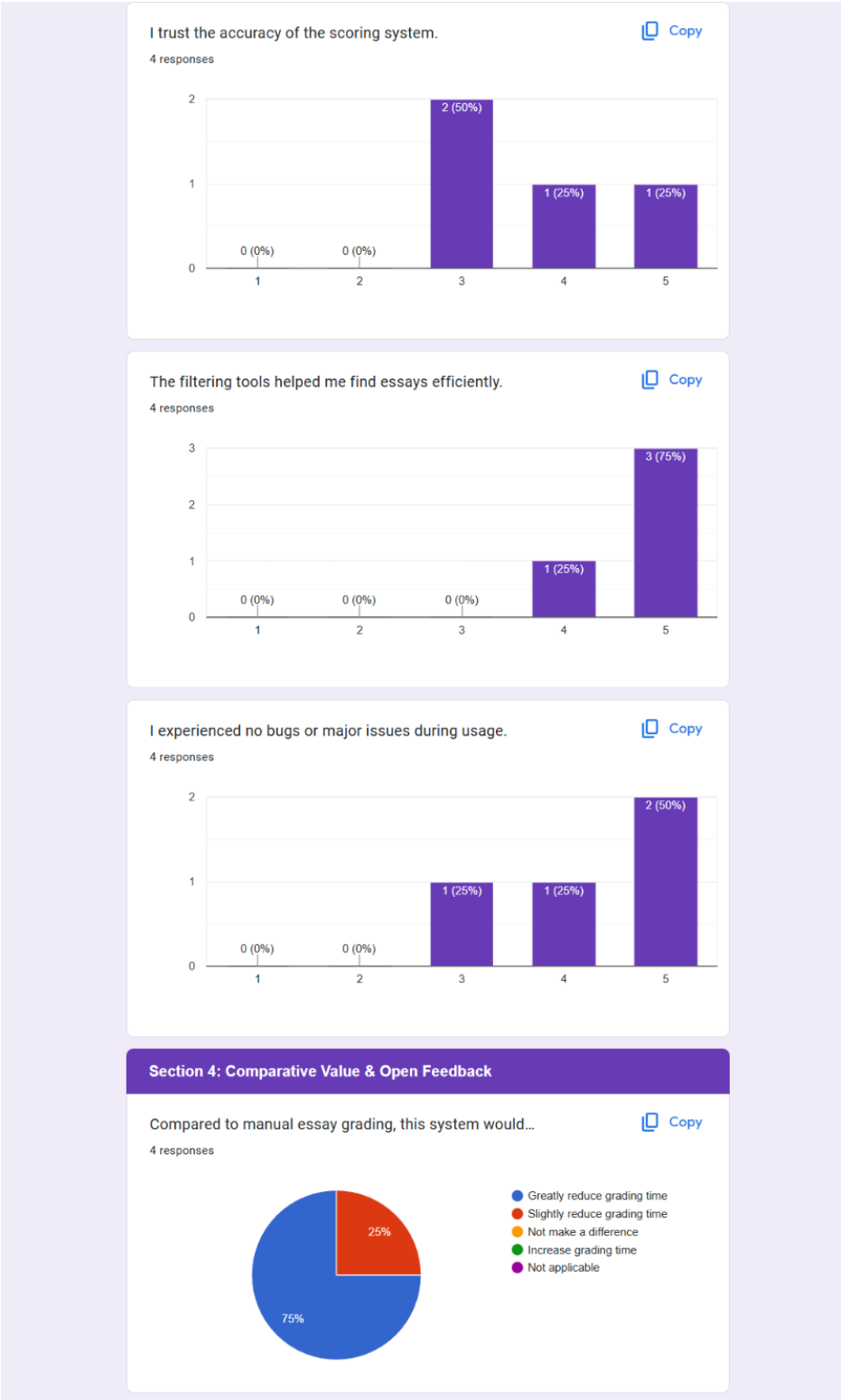


Figure A.25: AEGS User Acceptance Testing (UAT) survey (8)

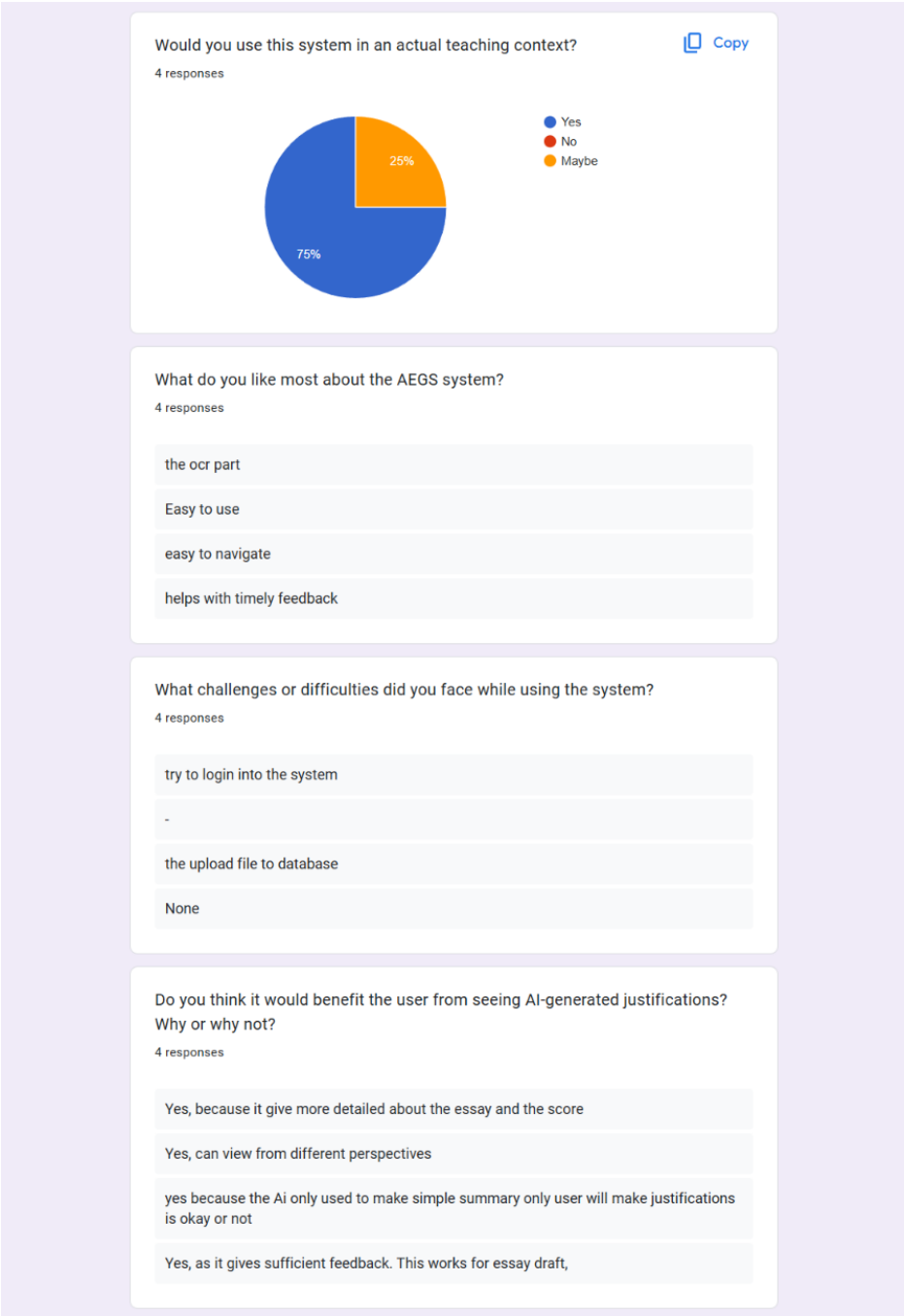
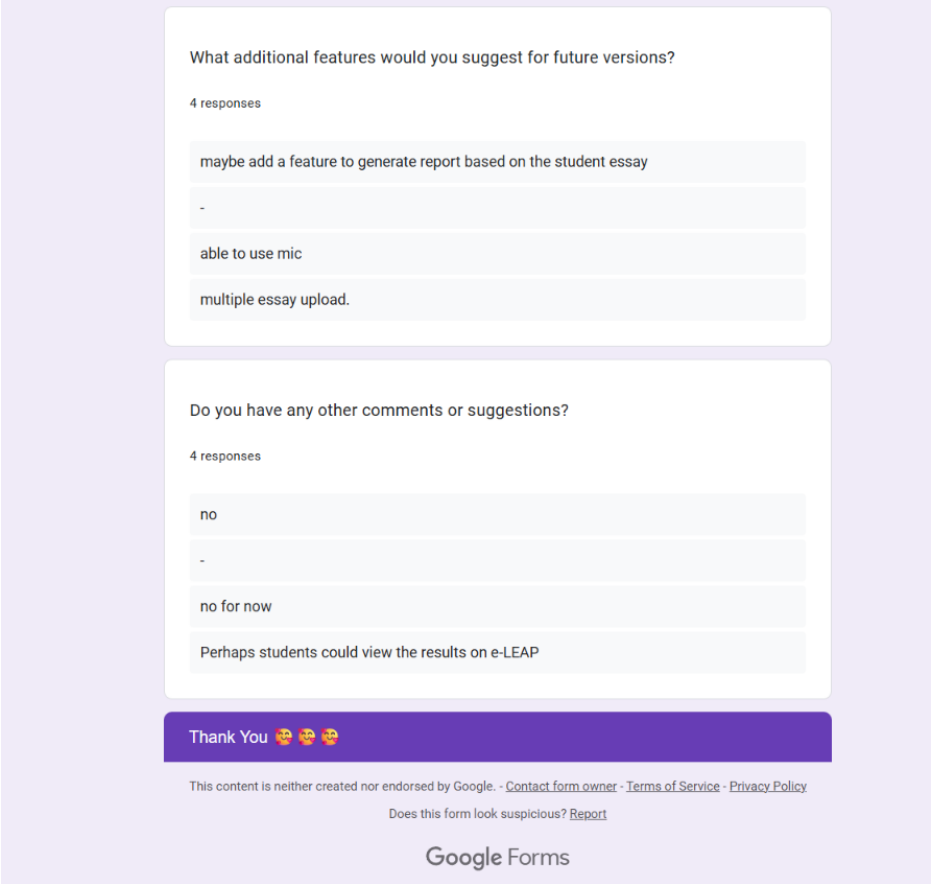


Figure A.26: AEGS User Acceptance Testing (UAT) survey (9)



System Module Testing Checklist - Automated Essay Grading System (AEGS)

Project Name: Automated Essay Grading System (AEGS)

Tested By: Alexander Anak Adrian

Test Date: 17–18 June 2025

Table 24: System Module Testing Checklist of Automated Essay Grading System

Module / Feature		Tested (✓)	Comments / Notes
Authentication			
1	User Registration	✓	Tested with valid data, mismatched passwords, and duplicate email. Validation works as expected.
2	User Login	✓	Valid and invalid credentials tested. Error messages appear properly.
3	Logout	✓	Successfully ends session and redirects to landing page. Session cleared.
Account Management			
1	View Account Information	✓	Displays name and email correctly. Layout consistent with dark theme.
2	Edit Account Details	✓	Successfully updates user info. Edit/save toggle works as intended.
3	Validation on Blank Edit Fields	✓	Validation triggers with empty input. User is prevented from saving.
Essay Submission			
1	Upload Typed Essay	✓	Title, student name, and essay content saved successfully.
2	Upload Essay via Image (OCR)	✓	OCR works well with clear images. Blurry image produces garbled text.
3	Submit Essay with Missing Info	✓	Required field warning appears. Submit button disabled.
4	Submit Poor Quality Image	✓	OCR handled gracefully and warning about essay content being garbled is display on the essay justification
Essay List & Result			
1	View Essay List	✓	Table displays all submitted essays with ID and status correctly.
2	Filter Essays by Year	✓	Dropdown filters essays based on year.
3	Filter Essays by Grading Status	✓	Filters “Completed”, “Processing”, “Error” work correctly.
4	Search Essays by Title	✓	Partial and full matches displayed

Essay Scoring & Justification (AI)			
1	AI-Generated Score Breakdown	✓	Scores generated using GPT-4o-mini based on rubric criteria.
2	AI-Generated Justification	✓	Clear explanation provided based on score. Matches rubric logic.
3	View Essay Result Page	✓	Displays essay content, scores, and AI justification accurately.
Score & Essay Management			
1	Edit Essay Result Info (title, name)	✓	Editable fields work. Changes saved and reflected in table.
2	Edit Score Breakdown (manual override)	✓	Lecturers can update scores. Save button confirms changes.
3	Delete Essay	✓	Confirmation popup appears. Essay removed from list upon confirmation.
Dashboard			
1	View Monthly Graph	✓	Graph shows current month by default with correct values.
2	Change Graph by Month	✓	Graph updates based on selection. Dropdown functional.
1	Consistent Theme UI	✓	Constant theme applied across all pages.
2	Responsive Design	✓	Layout adapts properly on different screen sizes (Windows 11).
3	Success/Error Popups	✓	Popups shown for delete, update, and submit actions.
4	Validation Feedback	✓	All form fields have visible validation messages when errors occur.
Security Checks			
1	Password Masking and Validation	✓	Password input hidden; errors shown for short/empty passwords.
2	Unauthorized Access Blocked	✓	Error message shown with forbidden access warning.