A logo of a company

Description automatically generated

**Machine-Generated Text Detection with Exam Grading System**

**A senior project submitted in partial fulfillment of the requirements for the degree of Bachelor of Computers and Artificial Intelligence**

**AI Department**

**Project Team**

1. **Ziad Mostafa Mahmoud Abdelhamed**
2. **Saher Mohamed Mohamed Ahmed**
3. **Omar Ayman Atia Mahmoud**
4. **Zyad Hesham Mohammed Alsharnobi**
5. **Osama Ayman Abdelfattah Mohammed**

**Under Supervision of**

**Dr. Sara Sweidan**

**Eng. Hossam Fakher**

**Benha, February 2025**

Acknowledgment

The development of this AI-powered exam grading and AI text detection platform has been a collaborative effort, drawing on the expertise and contributions of numerous individuals and resources. We would like to extend our gratitude to **Dr. Sara Sweidan**: We are deeply grateful for Dr. Sweidan's invaluable guidance, insights, and support throughout the development of this project. Her expertise in the field has been instrumental in shaping the platform's design and functionality.

DECLARATION

We hereby certify that this material, which we now submit for assessment on the program of study leading to the award of Bachelor of Computers and Artificial Intelligence in *(Artificial Intelligence)* is entirely our own work, that we have exercised reasonable care to ensure that the work is original, and does not to the best of our knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of our work.

**Signed:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ziad Mostafa

Ziad Mostafa

Ziad Mostafa

Ziad Mostafa

Ziad Mostafa

Ziad Mostafa

Ziad Mostafa

Ziad Mostafa

**Date:** 8, February 2025

ABSTRACT

This report presents a comprehensive overview of the AI-powered exam grading and AI text detection platform, designed to automate the grading of exams and detect machine-generated text in student submissions. The platform leverages advanced technologies, including Optical Character Recognition (OCR), Vision-Language Models (VLMs), and Natural Language Processing (NLP), to handle both typed and handwritten exams efficiently. The report details the system's key features, requirements, and architecture, highlighting the integration of user roles and role-based access control to ensure secure and efficient interactions. It also provides a detailed methodology for the grading and AI detection modules, along with visual representations. The report aims to provide a clear understanding of the platform's design, functionality, and benefits, making it a valuable resource for educators, students, and developers interested in enhancing educational assessment through AI.

**Table of Contents**

[Chapter 1 : Introduction 11](#_Toc190021446)

[1.1 Purpose of the Document 11](#_Toc190021447)

[1.2 Project Overview 12](#_Toc190021448)

[1.3 Audience 14](#_Toc190021449)

[1.4 Scope 16](#_Toc190021450)

[Chapter 2 : Literature Review 20](#_Toc190021451)

[2.1 Background Research 20](#_Toc190021452)

[2.1.1 Automated Grading Systems 20](#_Toc190021453)

[2.1.2 Optical Character Recognition (OCR) and Vision-Language Models (VLM) 21](#_Toc190021454)

[2.1.3 Natural Language Processing (NLP) for Grading 22](#_Toc190021455)

[2.1.4 AI Text Detection 22](#_Toc190021456)

[2.1.5 Challenges and Innovations 23](#_Toc190021457)

[2.2 Limitations of Current Solutions 23](#_Toc190021458)

[2.2.1 Lack of Support for Handwritten Exams 24](#_Toc190021459)

[2.2.2 Inconsistent Accuracy in Detecting AI-Generated Text 25](#_Toc190021460)

[2.2.3 Limited Transparency and Customization Options for Educators 25](#_Toc190021461)

[2.2.4 Scalability Issues 26](#_Toc190021462)

[2.2.5 Ethical and Privacy Concerns 26](#_Toc190021463)

[2.2.6 Limited Integration and Accessibility 27](#_Toc190021464)

[2.3 Relevant Technologies 27](#_Toc190021465)

[2.3.1 Optical Character Recognition (OCR) 27](#_Toc190021466)

[2.3.2 Vision-Language Models (VLM) 28](#_Toc190021467)

[2.3.3 Natural Language Processing (NLP) 28](#_Toc190021468)

[2.3.4 AI Text Detection 29](#_Toc190021469)

[2.4 Gaps Addressed by This Project 29](#_Toc190021470)

[2.4.1 Enhanced Support for Handwritten Exams 30](#_Toc190021471)

[2.4.2 Improved Accuracy in Detecting AI-Generated Text 30](#_Toc190021472)

[2.4.3 Transparent and Adjustable Detection 30](#_Toc190021473)

[2.4.4 Customizable Feedback 30](#_Toc190021474)

[Chapter 3 : Dataset Design 32](#_Toc190021475)

[3.1 Dataset Requirements 32](#_Toc190021476)

[3.2 Data Collection Process 33](#_Toc190021477)

[3.3 Data Preprocessing 35](#_Toc190021478)

[3.4 Dataset Size 38](#_Toc190021479)

[Chapter 4 : Methodology 40](#_Toc190021480)

[4.1 Grading Module Workflow 40](#_Toc190021481)

[4.1.1 Typed Exams (Form-Based) 41](#_Toc190021482)

[4.1.2 Handwritten Exams 42](#_Toc190021483)

[4.2 AI Text Detection Workflow 44](#_Toc190021484)

[4.3 Integration with User Roles 45](#_Toc190021485)

[4.3.1 Professors 45](#_Toc190021486)

[4.3.2 Students 46](#_Toc190021487)

[4.3.3 Admins 46](#_Toc190021488)

[4.4 Role-Based Access Control 47](#_Toc190021489)

[4.4.1 JWT Tokens 47](#_Toc190021490)

[4.4.2 Role-Specific Permissions 48](#_Toc190021491)

[Chapter 5 : System Design 49](#_Toc190021492)

[5.1 Key Features 49](#_Toc190021493)

[5.2 Requirements 50](#_Toc190021494)

[5.2.1 Assumptions and Dependencies 50](#_Toc190021495)

[5.2.2 Functional Requirements 51](#_Toc190021496)

[5.2.3 Non-Functional Requirements 52](#_Toc190021497)

[5.3 Use Case Diagram 54](#_Toc190021498)

[5.4 Framework Diagram 57](#_Toc190021499)

[5.5 Entity-Relationship Diagram (ERD) 60](#_Toc190021500)

[5.6 Data Flow Diagrams 66](#_Toc190021501)

[Context DFD (Level 0) 66](#_Toc190021502)

[Level 1 DFD 69](#_Toc190021503)

[5.7 Class Diagram 72](#_Toc190021504)

[5.8 Activity Diagrams 80](#_Toc190021505)

[5.9 Sequence Diagram 87](#_Toc190021506)

[5.10 Flowchart Diagram 93](#_Toc190021507)

[5.11 System Workflow 97](#_Toc190021508)

[Chapter 6 : Web Application 102](#_Toc190021509)

[6.1 Registration Screen 102](#_Toc190021510)

[6.2 Login Screen 102](#_Toc190021511)

[6.3 Professor Dashboard 103](#_Toc190021512)

[6.4 Student Dashboard 103](#_Toc190021513)

[6.5 About Screen 104](#_Toc190021514)

[Chapter 7 : Conclusion 105](#_Toc190021515)

[References 106](#_Toc190021516)

**Table of Figures**

[Figure ‎1:1 Project Workflow Diagram 12](#_Toc190021517)

[Figure ‎1:2 User Roles and Responsibilities 14](#_Toc190021518)

[Figure ‎1:3 Project Scope and Exclusions 16](#_Toc190021519)

[Figure ‎2:1 Comparison of Existing Automated Grading Systems 20](#_Toc190021520)

[Figure ‎2:2 Limitations Pie Chart 24](#_Toc190021521)

[Figure ‎2:3 Gaps and Solutions 29](#_Toc190021522)

[Figure ‎4:1 Grading Module Workflow Diagram 40](#_Toc190021523)

[Figure ‎4:2 AI Text Detection Workflow 44](#_Toc190021524)

[Figure ‎4:3 Role-Based Access Control Diagram 47](#_Toc190021525)

[Figure ‎4:4: Use Case Diagram 54](#_Toc190021526)

[Figure ‎4:5 Framework Diagram 57](#_Toc190021527)

[Figure ‎4:6 Entity-Relationship Diagram 61](#_Toc190021528)

[Figure ‎4:7 Context Data Flow Diagram 67](#_Toc190021529)

[Figure ‎4:8 Level 1 Data Flow Diagram 69](#_Toc190021530)

[Figure ‎4:9 Class Diagram 73](#_Toc190021531)

[Figure ‎4:10 User Registration Activity Diagram 80](#_Toc190021532)

[Figure ‎4:11 User Login Activity Diagram 81](#_Toc190021533)

[Figure ‎4:12 Exam Creation Activity Diagram 82](#_Toc190021534)

[Figure ‎4:13 Exam Upload Activity Diagram 83](#_Toc190021535)

[Figure ‎4:14 OCR Processing Activity Diagram 84](#_Toc190021536)

[Figure ‎4:15 Grading Activity Diagram 84](#_Toc190021537)

[Figure ‎4:16 AI Text Detection Activity Diagram 85](#_Toc190021538)

[Figure ‎4:17 Results Dashboard Activity Diagram 86](#_Toc190021539)

[Figure ‎4:18 Settings Customization Activity Diagram 86](#_Toc190021540)

[Figure ‎4:19 Admin Management Activity Diagram 87](#_Toc190021541)

[Figure ‎4:20 Flowchart Diagram 93](#_Toc190021542)

[Figure ‎4:21 System Workflow 97](#_Toc190021543)

[Figure ‎6:1 Registration Screen 102](#_Toc190021544)

[Figure ‎6:2 Login Screen 102](#_Toc190021545)

[Figure ‎6:3 Professor Dashboard 103](#_Toc190021546)

[Figure ‎6:4 Student Dashboard 103](#_Toc190021547)

[Figure ‎6:5 About Screen 104](#_Toc190021548)

List of Acronyms/Abbreviations

|  |  |
| --- | --- |
| AI | Artificial Intelligence |
| OCR | Optical Character Recognition |
| VLM | Vision-Language Model |
| NLP | Natural Language Processing |
| MCQ | Multiple Choice Question |
| T/F | True/False |
| JWT | JSON Web Token |
| API | Application Programming Interface |
| LLM | Large Language Model |
| MCQ\_TF | Multiple Choice and True/False Questions |
| RoBERTa | Robustly Optimized BERT Pretraining Approach |
| DeBERTa | Decoding-enhanced BERT with Electra-style Attention |
| Qwen2.5-VL | Qwen Vision-Language Model Version 2.5 |
| Gemini 2.0 | Vision-Language Model |
| API | Application Programming Interface |

# : Introduction

## 1.1 Purpose of the Document

This document serves as a comprehensive guide to the AI-powered exam grading and AI text detection platform. It is designed to provide a detailed overview of the project, its objectives, key features, and the target audience. The primary goal of this documentation is to equip educators, students, developers, and stakeholders with the necessary information to understand, utilize, and contribute to the platform effectively.

**Objectives of the Document**

1. **Project Overview**: To offer a clear and concise summary of the project's goals, scope, and key features. This includes an explanation of how the platform automates exam grading and detects machine-generated text in student submissions.
2. **Technical Insights**: To delve into the technical aspects of the platform, including the system architecture, data flow, and the AI models used for grading and text detection. This section is particularly useful for developers and technical stakeholders.
3. **User Guidance**: To provide step-by-step instructions and guidelines for different user roles (professors, students, and admins) on how to interact with the platform. This includes uploading exams, viewing results, and managing user accounts.
4. **Evaluation and Performance**: To present the methodologies used for evaluating the platform's performance, including accuracy metrics for grading and AI text detection. This section also covers user feedback and testing results.
5. **Future Development**: To outline potential upgrades and improvements that can be made to the platform, such as multilingual support and integration with learning management systems (LMS).

## 1.2 Project Overview

A close-up of a diagram

AI-generated content may be incorrect.

Figure ‎1:1 Project Workflow Diagram

**Goal**

The primary goal of this project is to develop an AI-powered platform that automates the grading of exams and detects machine-generated text in student submissions. This platform aims to enhance the efficiency and accuracy of the grading process while providing educators with tools to identify and address the use of AI-generated content in academic assessments.

**Key Features**

1. **Automated Exam Grading**:
   * The platform supports the grading of both typed and handwritten exams using advanced Optical Character Recognition (OCR) and Vision-Language Models (VLM) for text extraction and interpretation.
   * Natural Language Processing (NLP) models, such as RoBERTa, are employed to compare student answers with model answers provided by professors, ensuring accurate and consistent grading.
2. **AI Text Detection**:
   * The platform includes a sophisticated AI text detection module that identifies machine-generated text in student submissions.
   * This module utilizes a combination of transformer models, like DeBERTa, and statistical features, such as perplexity and burstiness, to determine the likelihood that a text was generated by an AI.
3. **Educator Dashboard**:
   * Professors have access to a comprehensive dashboard where they can upload exams, manage student submissions, and view grading results.
   * The dashboard allows educators to adjust grading weights and AI detection sensitivity, providing them with control over the assessment process.
   * Educators can generate reports and export results for further analysis or record-keeping.
4. **Student Access**:
   * Students can securely view their grades and feedback through a dedicated dashboard.
   * The platform ensures that students have access only to their own results, maintaining privacy and security.
   * Feedback is presented in an interactive format, allowing students to understand their strengths and areas for improvement.
5. **User Roles and Access Control**:
   * The platform supports multiple user roles, including professors, students, and admins, each with specific permissions and access levels.
   * Role-based access control is implemented using JWT tokens, ensuring that users can only access features and data relevant to their role.

## 1.3 Audience

A diagram of a diagram

AI-generated content may be incorrect.

Figure ‎1:2 User Roles and Responsibilities

The target audience for this documentation includes a diverse range of stakeholders who will interact with or contribute to the AI-powered exam grading and AI text detection platform. The primary audiences are:

1. **Educators**:
   * **Professors and Teachers**: These users will utilize the platform to upload exams, manage student submissions, and view grading results. They will benefit from the automated grading and AI text detection features, which aim to enhance their efficiency and accuracy in assessing student performance.
   * **Instructional Designers**: Individuals involved in designing and structuring educational content may use the platform to ensure that assessments are fair and reliable.
2. **Students**:
   * **Examinees**: Students who submit exams through the platform will access their grades and feedback. The platform provides a secure and transparent way for students to understand their performance and areas for improvement.
3. **Developers**:
   * **Software Engineers**: Technical personnel responsible for the development, maintenance, and enhancement of the platform. They will use this documentation to understand the system architecture, data flow, and integration of AI models.
   * **Data Scientists**: Individuals involved in training and fine-tuning the AI models used for grading and text detection.
4. **Stakeholders**:
   * **Administrators**: School or university administrators who oversee the implementation and use of the platform within their institutions. They will ensure that the platform meets educational standards and policies.
   * **Investors and Funders**: Individuals or organizations that support the project financially. They will be interested in the project's goals, progress, and potential impact on education.
   * **Researchers**: Academics and researchers studying the application of AI in education. They may use the platform's data and insights for further research and publications.
5. **Technical Support**:
   * **IT Professionals**: Individuals responsible for the deployment, security, and troubleshooting of the platform. They will use this documentation to understand the technical requirements and ensure smooth operation.

## 1.4 Scope

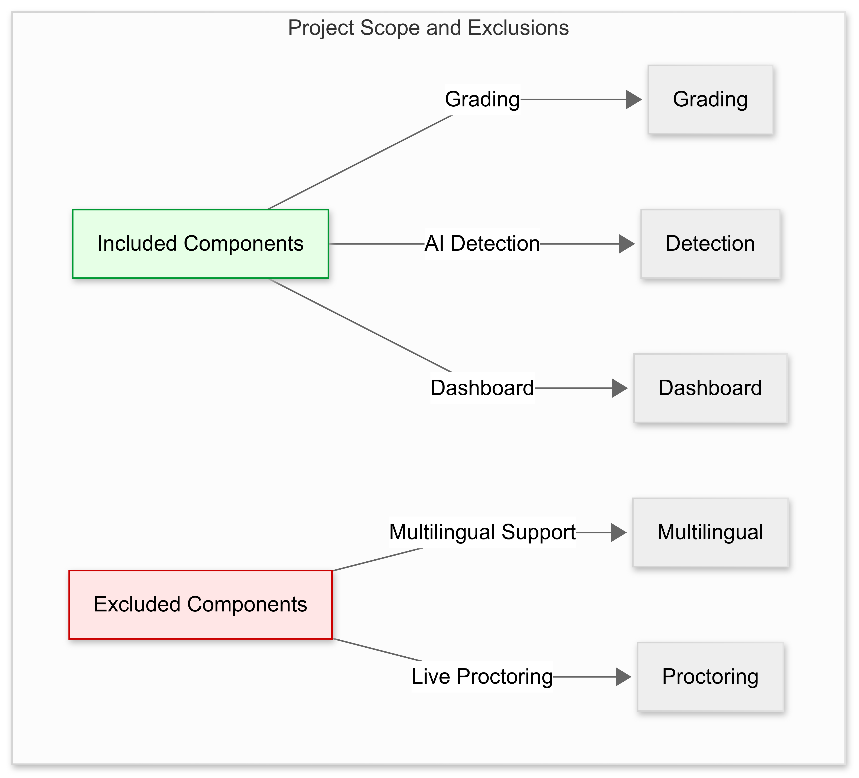


Figure ‎1:3 Project Scope and Exclusions

The scope of this project encompasses the development, deployment, and initial evaluation of the AI-powered exam grading and AI text detection platform. The following components are included within the project's scope:

1. **Technical Implementation**:
   * **Frontend Development**: Design and implementation of user interfaces for educators, students, and admins using React.js, Framer Motion, and Three.js.
   * **Backend Development**: Development of API endpoints, database architecture, and server-side logic using Node.js/Express, Flask, and PostgreSQL.
   * **AI Model Integration**: Implementation of OCR/VLM for text extraction, NLP models for semantic analysis, and transformer models for AI text detection.
2. **User Interface and Experience**:
   * **Dashboard Design**: Creation of intuitive and user-friendly dashboards for educators and students, including features for uploading exams, viewing results, and adjusting settings.
   * **Access Control**: Implementation of role-based access control using JWT tokens to ensure secure and appropriate access to platform features.
3. **Performance Evaluation**:
   * **Accuracy Testing**: Assessment of the platform's grading and AI text detection accuracy using benchmark datasets and human-graded samples.
   * **User Feedback**: Collection and analysis of user feedback to refine the platform's features and usability.
4. **Security Measures**:
   * **Data Protection**: Implementation of HTTPS encryption, data anonymization, and regular audits to ensure the security and privacy of user data.
   * **Compliance**: Adherence to relevant data protection regulations and educational standards.
5. **Documentation and Support**:
   * **User Guides**: Provision of step-by-step instructions for educators and students on how to use the platform.
   * **API Documentation**: Detailed descriptions of all API endpoints for developers.
   * **Technical Whitepaper**: In-depth explanation of the AI models, algorithms, and system architecture.

**Exclusions**

The following components are outside the initial scope of this project and may be considered for future development:

* **External Integrations**: Integration with other educational platforms or learning management systems (LMS) is not included in the initial scope.
* **Multilingual Support**: The platform will initially support English, with additional languages considered for future upgrades.
* **Live Proctoring**: Real-time monitoring of exams using AI webcam proctoring is not included in the initial release.
* **Extensive Customization**: Beyond the initial settings provided, extensive customization options for educators are not included in the initial scope.

By defining the scope clearly, the project aims to focus on delivering a robust and functional platform that meets the core needs of its users while allowing for future enhancements and expansions.

# : Literature Review

## 2.1 Background Research

The field of automated exam grading and AI text detection has seen significant advancements in recent years, driven by the increasing sophistication of machine learning and natural language processing (NLP) technologies. This section provides an overview of existing tools, platforms, and methodologies, highlighting key challenges and innovations that have shaped the current landscape.

### 2.1.1 Automated Grading Systems

Several tools and platforms have been developed to automate the grading of exams, particularly for multiple-choice and short-answer questions. However, the challenge of accurately grading handwritten and essay-type questions remains a focal point for innovation.

A graph of blue and red rectangles

AI-generated content may be incorrect.

Figure ‎2:1 Comparison of Existing Automated Grading Systems

* **Turnitin**: Primarily known for plagiarism detection, Turnitin also offers automated grading capabilities for multiple-choice and short-answer questions. Its strengths lie in its vast database of previously submitted work, which aids in identifying similarities and inconsistencies.
* **Gradescope**: Designed for higher education, Gradescope supports grading of both handwritten and digital exams. It allows instructors to define grading rubrics and distribute scores based on predefined criteria. Gradescope excels in handling complex, multi-part questions and provides detailed analytics for performance tracking.
* **GPTZero**: Specialized in detecting AI-generated text, GPTZero uses statistical features like perplexity and burstiness to identify machine-generated content. While effective for certain types of text, it lacks the nuanced understanding required for complex, context-dependent analyses.
* **QuillBot**: Known for paraphrasing and summarization, QuillBot also offers AI-based grading for essays. It evaluates writing quality based on grammar, coherence, and style, providing feedback on areas for improvement.

### 2.1.2 Optical Character Recognition (OCR) and Vision-Language Models (VLM)

OCR and VLM technologies play a crucial role in converting handwritten text into digital formats, enabling further processing and analysis.

* **OCR**: Traditional OCR systems like Tesseract and OpenCV have made significant strides in recognizing printed text. However, they struggle with complex, cursive handwriting, especially in educational settings where handwriting styles can vary widely.
* **Vision-Language Models (VLM)**: Recent advancements in VLMs have improved the interpretation of handwritten text and diagrams. These models integrate visual and textual information, making them more effective in educational contexts where diagrams and charts are common.

### 2.1.3 Natural Language Processing (NLP) for Grading

NLP models have become increasingly sophisticated, capable of understanding and analyzing the semantic meaning of text. Key models and techniques include:

* **RoBERTa**: A pre-trained transformer-based model that excels in semantic analysis. RoBERTa is fine-tuned on large corpora and can be adapted for specific domains, such as educational assessments. It is particularly useful for comparing student answers against model answers and providing detailed feedback.
* **DeBERTa**: Another transformer-based model that enhances contextual understanding through advanced techniques like relative position embeddings and multi-head attention. DeBERTa is used in this project for detecting AI-generated text, leveraging its ability to capture subtle nuances in language.

### 2.1.4 AI Text Detection

Detecting AI-generated text is a challenging task due to the evolving nature of AI models and the increasing sophistication of generated content. Current approaches include:

* **Statistical Approaches**: Methods like GPTZero rely on statistical features to identify patterns indicative of AI-generated text. These methods are fast but may miss nuanced differences in highly coherent AI-generated text.
* **Transformer Models**: Advanced transformer-based models like DeBERTa offer deeper contextual understanding and can better distinguish between human and AI-generated text. They are trained on large datasets and fine-tuned for specific tasks, providing more accurate and reliable results.
* **Ensemble Methods**: Combining multiple detection techniques, such as statistical features and transformer models, can significantly improve detection accuracy and reduce false positives.

### 2.1.5 Challenges and Innovations

Despite the advancements, several challenges remain:

* **Accuracy**: Ensuring high accuracy in grading and AI text detection, especially for complex, subjective questions.
* **Scalability**: Handling large volumes of exams and submissions efficiently, particularly in high-stakes educational environments.
* **Transparency**: Providing transparent feedback to students and educators, without exposing underlying model logic.
* **Bias Mitigation**: Ensuring fairness across different demographic groups and avoiding over-reliance on AI-generated probabilities.

Innovations in OCR, VLM, NLP, and AI text detection have laid the groundwork for more advanced and reliable systems. This project builds upon these advancements, addressing key challenges through the integration of hybrid models, adjustable detection thresholds, and robust infrastructure.

## 2.2 Limitations of Current Solutions

Despite the advancements in automated exam grading and AI text detection, existing solutions face several limitations that hinder their effectiveness and widespread adoption. This section highlights these limitations, providing context for the gaps that our platform aims to address.

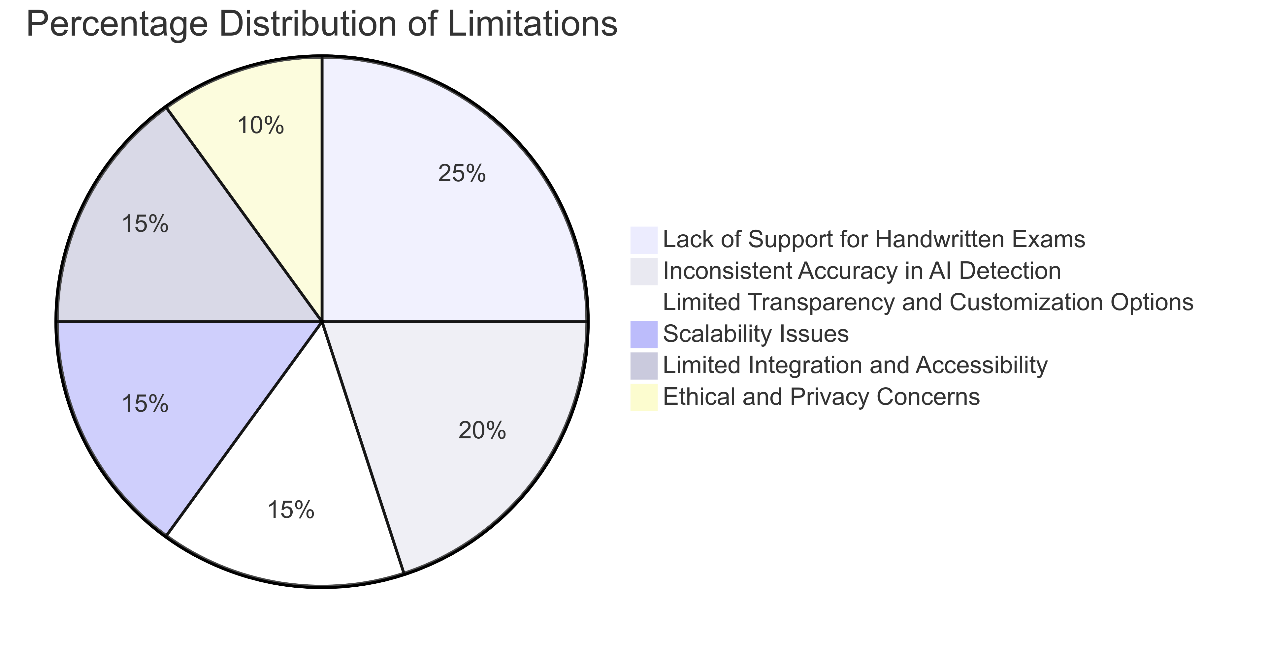


Figure ‎2:2 Limitations Pie Chart

### 2.2.1 Lack of Support for Handwritten Exams

While some platforms like Gradescope offer limited support for handwritten exams, they often struggle with accuracy and efficiency, particularly for complex, cursive handwriting. Key issues include:

* **Poor OCR Accuracy**: Traditional OCR systems like Tesseract and OpenCV excel at recognizing printed text but falter with handwritten text, especially in educational settings where handwriting styles can vary widely.
* **Limited Contextual Understanding**: OCR alone cannot interpret diagrams, graphs, or complex mathematical equations, which are common in educational exams.
* **Manual Post-Processing**: Even when OCR is partially successful, manual post-processing is often required to correct errors, adding significant time and effort for educators.

### 2.2.2 Inconsistent Accuracy in Detecting AI-Generated Text

Current AI text detection tools, while useful, often struggle with accuracy, especially when dealing with highly coherent and contextually appropriate AI-generated content. Key limitations include:

* **Statistical Approaches**: Tools like GPTZero rely heavily on statistical features (e.g., perplexity, burstiness) to detect AI-generated text. While effective for basic text, they may fail to identify more sophisticated AI-generated content that mimics human writing styles closely.
* **Contextual Insensitivity**: Many statistical methods lack the ability to understand the broader context of the text, leading to false positives or negatives in complex scenarios.
* **Overfitting**: Some models may overfit to specific datasets, performing well in controlled environments but poorly in real-world scenarios where text diversity is high.

### 2.2.3 Limited Transparency and Customization Options for Educators

Educators require transparency and flexibility in grading and AI detection processes. Current solutions often fall short in these areas:

* **Opaque Detection Logic**: Many AI text detection tools do not provide detailed explanations of how they arrived at their conclusions, leaving educators unsure about the basis of their findings.
* **Fixed Sensitivity Settings**: Detection tools often come with fixed sensitivity settings, limiting educators' ability to fine-tune the tool to their specific needs. This can lead to either too many false positives or missed detections.
* **Lack of Customizable Feedback**: Educators need the ability to provide personalized feedback to students, but many platforms offer limited customization options, reducing the value of the feedback provided.

### 2.2.4 Scalability Issues

Handling large volumes of exams and submissions efficiently is a significant challenge for many existing solutions:

* **Resource Intensive**: Many grading and detection systems require substantial computational resources, making them impractical for institutions with limited infrastructure.
* **Manual Intervention**: Despite automation, many platforms still require significant manual intervention, particularly for complex or ambiguous questions, which can slow down the grading process.
* **Limited Load Balancing**: Without robust load balancing and auto-scaling capabilities, systems can struggle during peak usage periods, leading to delays and reduced performance.

### 2.2.5 Ethical and Privacy Concerns

Ensuring ethical and privacy considerations is paramount in educational settings:

* **Data Privacy**: Many existing platforms do not adequately anonymize student data, raising concerns about privacy and data security.
* **Bias and Fairness**: Without regular audits and bias mitigation strategies, AI models can perpetuate or exacerbate existing biases, leading to unfair grading outcomes.
* **Transparency**: Lack of transparency in AI decision-making can erode trust among educators and students, making it difficult to adopt and integrate these tools into existing workflows.

### 2.2.6 Limited Integration and Accessibility

Many platforms lack seamless integration with existing learning management systems (LMS) and other educational tools:

* **Isolated Solutions**: Platforms often operate as standalone systems, requiring educators and students to switch between multiple tools, disrupting workflow efficiency.
* **Accessibility**: Some tools may not be accessible to all users, particularly those with limited technical expertise or resources.

## 2.3 Relevant Technologies

The development of our AI-powered exam grading and AI text detection platform leverages a range of advanced technologies to address the limitations identified in current solutions. These technologies encompass Optical Character Recognition (OCR), Vision-Language Models (VLM), Natural Language Processing (NLP), and AI text detection methodologies.

### 2.3.1 Optical Character Recognition (OCR)

OCR technologies are essential for converting handwritten text into digital format, enabling further processing and analysis.

* **Tesseract**: An open-source OCR engine that supports multiple languages and can recognize printed text. It is widely used for its accuracy and ease of integration.
* **OpenCV**: A library for computer vision that can assist in preprocessing images, making them more suitable for OCR. It is particularly useful for enhancing image quality and preparing images for recognition.

### 2.3.2 Vision-Language Models (VLM)

VLMs are crucial for interpreting handwritten text and diagrams, providing a more comprehensive understanding of complex educational content.

* **Qwen2.5-VL**: A state-of-the-art VLM developed by Alibaba Cloud. Qwen2.5-VL is designed to understand multimodal data, combining visual and textual information to provide context-rich interpretations. It is particularly effective in handling complex diagrams and handwritten text in educational settings.
* **Gemini 2.0**: Developed by Google, Gemini 2.0 is a powerful multimodal model that excels in understanding both text and images. It can interpret handwritten text, diagrams, and graphs, offering a deeper level of contextual understanding.

### 2.3.3 Natural Language Processing (NLP)

NLP models are central to the semantic analysis of student answers and the comparison against model solutions.

* **RoBERTa**: A pre-trained transformer-based model that excels in semantic analysis. RoBERTa is fine-tuned on large corpora and can be adapted for specific domains, such as educational assessments. It is particularly useful for comparing student answers against model answers and providing detailed feedback.
* **DeBERTa**: Another transformer-based model that enhances contextual understanding through advanced techniques like relative position embeddings and multi-head attention. DeBERTa is used in this project for detecting AI-generated text, leveraging its ability to capture subtle nuances in language.

### 2.3.4 AI Text Detection

AI text detection involves identifying machine-generated content using a combination of statistical features and advanced transformer models.

* **Perplexity and Burstiness**: Statistical features used to quantify the complexity and variability of text. These features help in distinguishing between human-written and AI-generated content.
* **Transformer Models**: Advanced transformer-based models like DeBERTa are used to analyze text in depth, capturing contextual nuances and providing more accurate detection of AI-generated text.

## 2.4 Gaps Addressed by This Project

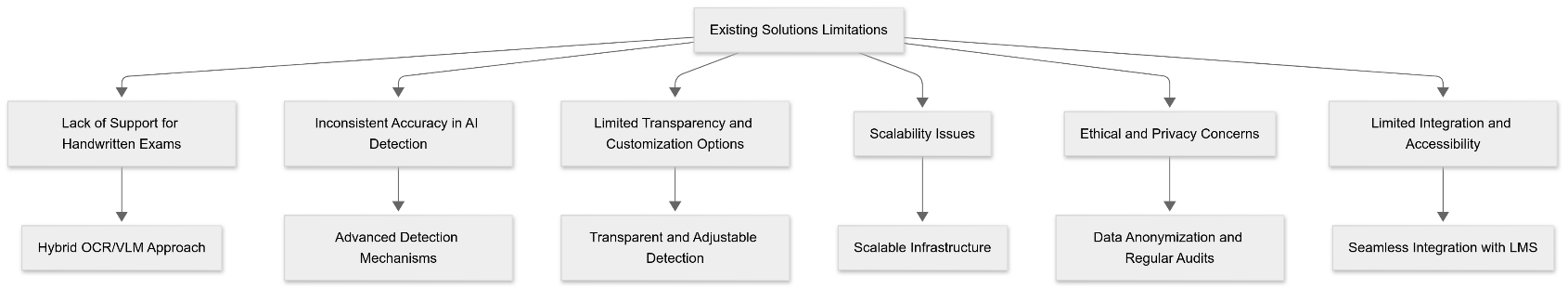
Our platform addresses the key limitations and gaps identified in current automated exam grading and AI text detection solutions. By leveraging advanced technologies and thoughtful design, we aim to provide a more accurate, efficient, and user-friendly solution.

Figure ‎2:3 Gaps and Solutions

### 2.4.1 Enhanced Support for Handwritten Exams

* **Hybrid OCR/VLM Approach**: Combining OCR (Tesseract/OpenCV) with VLMs (Qwen2.5-VL, Gemini 2.0) to improve accuracy in recognizing handwritten text.
* **Contextual Interpretation**: VLMs like Qwen2.5-VL and Gemini 2.0 can interpret complex diagrams, graphs, and mathematical equations, providing a more holistic understanding of handwritten content.

### 2.4.2 Improved Accuracy in Detecting AI-Generated Text

* **Advanced Detection Mechanisms**: Using transformer-based models like DeBERTa combined with statistical features (perplexity, burstiness) to enhance detection accuracy and reduce false positives.
* **Multimodal Understanding**: VLMs like Qwen2.5-VL and Gemini 2.0 can analyze both text and visual elements, providing a more nuanced understanding of student responses.

### 2.4.3 Transparent and Adjustable Detection

* **Detailed AI Detection Scores**: Providing educators with detailed AI detection scores and explanations for flagged phrases.
* **Adjustable Sensitivity**: Allowing educators to adjust sensitivity settings to fine-tune AI detection results based on their specific needs.

### 2.4.4 Customizable Feedback

* **Personalized Feedback**: Allowing educators to provide customized feedback to students, enhancing the learning experience.
* **Interactive Feedback Sections**: Offering expandable sections with detailed explanations and highlighted areas of strength and improvement.

By addressing these gaps and leveraging advanced technologies, our platform aims to provide a more robust, accurate, and user-friendly solution for automated exam grading and AI text detection, benefiting both educators and students. This approach ensures that the platform not only meets but exceeds the needs of its users, fostering a more efficient and transparent educational environment.

# : Dataset Design

## 3.1 Dataset Requirements

To build an effective AI-powered exam grading and AI text detection platform, it is essential to design and curate datasets that meet specific requirements for both the Vision-Language Models (VLMs) and AI text detection modules. These datasets must be comprehensive, balanced, and representative of the diverse content and scenarios encountered in educational settings. Below are the detailed requirements for each dataset component.

3.1.1 VLM Pre-Training Datasets

* **Diversity**: Cover a wide range of visual and textual data to ensure the model can generalize across different domains and languages.
* **Quality**: High-quality images and captions to ensure accurate and meaningful learning.
* **Size**: Large-scale datasets to enable deep learning models to learn robust representations.
* **Annotations**: Well-annotated datasets with bounding boxes, object labels, and captions for both images and text.

3.1.2 AI Detection Datasets

* **Balance**: Ensure a balanced mix of human-written and machine-generated text to prevent bias in model training.
* **Diversity**: Include a wide range of topics, genres, and writing styles to improve generalization.
* **Labels**: Accurate binary labels (human-written or machine-generated) for each text sample.
* **Size**: Large enough to train and evaluate models effectively, with sufficient samples for both training and testing.

3.1.3 Semantic Analysis Dataset

* **Structured Format**: Well-structured CSV files containing context, questions, answers, and labels.
* **Diversity**: Rich diversity in the context, questions, and answers to cover a wide range of topics.
* **Quality**: High-quality and well-labeled data to ensure accurate training and evaluation.
* **Size**: Sufficient size to train and evaluate models effectively, with a balanced distribution of labels.

## 3.2 Data Collection Process

3.2.1 VLM Pre-Training Datasets

* **Publicly Available Datasets**: Utilize large-scale, publicly available datasets such as MSCOCO, Visual Genome, Conceptual Captions, SBU Captions, and CC3M.
* **Data Aggregation**: Aggregate data from multiple sources to ensure diversity and coverage.
* **Collaboration**: Partner with educational institutions to collect additional datasets, ensuring relevance and quality.

3.2.2 AI Detection Datasets

* **Training and Development Sets**:
  + **HC3 (Huge Controversial Conversation Corpus)**: Contains 57,811 machine-generated texts and 18,671 human-written texts.
  + **M4GT-Bench (Machine-Generated Text Generation Benchmark)**: Contains 267,863 machine-generated texts and 181,081 human-written texts.
  + **MAGE (Multi-Angle Generative Evaluation)**: Contains 285,093 machine-generated texts and 182,093 human-written texts.
  + **Combine and Filter**: Merge these datasets, removing duplicates and ensuring a balanced distribution of human-written and machine-generated texts.
  + **Random Splitting**: Randomly split the combined dataset into training (70%) and development (20%) sets.
* **Dev-Test Set**:
  + **RAID (Realistic AI Detection)**: Contains 13,371 human-written texts.
  + **LLMDetectAIve**: Contains 19,186 machine-generated texts, categorized into three types: fully machine-generated, human-written and machine-polished, and machine-generated and machine-humanized.
  + **Random Sampling**: Sample texts from RAID and LLMDetectAIve to create a balanced dev-test set.
* **Test Set**:
  + **MixSet**: A dataset containing a mix of human-written and machine-generated texts.
  + **CUDRT (Controversial and Uncontroversial Debate Rounds Transcripts)**: Contains 22,978 texts, with 12,287 machine-generated texts and 10,691 human-written texts.
  + **IELTS**: Contains 24,700 texts, with 11,382 machine-generated texts and 13,318 human-written texts.
  + **NLPeer and PeerSum**: Used to generate academic paper peer reviews.
  + **Additional Datasets**: Collected by the team, involving 27 different corpora and spanning 15 languages, with six of them not seen in the training, development, or dev-test sets.
  + **Random Sampling**: Sample texts from these datasets to create a balanced test set.

3.2.3 Semantic Analysis Dataset

* **Publicly Available Dataset**: Utilize the "Expect True or False by Similarity EN" dataset from Kaggle.
* **Data Collection**: Download the dataset and ensure it is properly formatted for training and validation.

## 3.3 Data Preprocessing

3.3.1 VLM Pre-Training Datasets

* **Image Processing**:
  + **Resizing**: Resize images to a consistent resolution (e.g., 224x224 pixels).
  + **Normalization**: Normalize pixel values to a range of [0, 1].
  + **Augmentation**: Apply data augmentation techniques (e.g., rotation, flipping, color jittering) to increase diversity.
* **Text Processing**:
  + **Tokenization**: Split text into tokens using a tokenizer (e.g., BERT tokenizer).
  + **Lowercasing**: Convert all text to lowercase to ensure uniformity.
  + **Special Characters Removal**: Remove special characters and punctuation marks.
  + **Stopword Removal**: Remove common stopwords to reduce noise.
  + **Spell Correction**: Correct spelling errors using a spell-checking library.
* **Annotation**:
  + **Bounding Boxes**: Ensure high-quality bounding box annotations for objects in images.
  + **Object Labels**: Verify and correct object labels if necessary.

3.3.2 AI Detection Datasets

* **Text Cleaning**:
  + **Duplicate Removal**: Remove duplicate texts.
  + **Spelling Correction**: Correct spelling errors.
  + **Formatting**: Normalize text formatting (e.g., consistent line breaks).
* **Tokenization**:
  + **Splitting**: Split text into tokens using a tokenizer (e.g., SentencePiece or BERT tokenizer).
* **Labeling**:
  + **Binary Labels**: Assign binary labels (human-written or machine-generated) to each text sample.
  + **Verification**: Verify labels for accuracy and consistency.
* **Data Augmentation**:
  + **Synthetic Generation**: Generate synthetic data to increase the diversity of the dataset.

3.3.3 Semantic Analysis Dataset

* **Cleaning**:
  + **Duplicate Removal**: Remove duplicate entries.
  + **Spelling Correction**: Correct spelling errors.
  + **Formatting**: Normalize text formatting (e.g., consistent line breaks).
* **Tokenization**:
  + **Splitting**: Split text into tokens using a tokenizer (e.g., BERT tokenizer).
* **Labeling**:
  + **Binning**: Ensure labels are correctly binned into ranges.
  + **Verification**: Verify labels for accuracy and consistency.

## 3.4 Dataset Size

3.4.1 VLM Pre-Training Datasets

* **MSCOCO**: Contains over 328,000 images with 5 captions each, totaling over 1.5 million image-caption pairs.
* **Visual Genome**: Includes over 108,000 images with detailed annotations, including objects, attributes, and relationships.
* **Conceptual Captions**: Comprises over 3 million image-caption pairs, providing extensive coverage for multimodal learning.
* **SBU Captions**: Contains over 1 million images with associated captions, offering additional training data.
* **CC3M**: Features 2.3 million images with captions, providing extensive coverage for pre-training VLMs.

3.4.2 AI Detection Datasets

* **Training and Development Sets**:
  + **HC3**: 57,811 machine-generated texts + 18,671 human-written texts = 76,482 total.
  + **M4GT-Bench**: 267,863 machine-generated texts + 181,081 human-written texts = 448,944 total.
  + **MAGE**: 285,093 machine-generated texts + 182,093 human-written texts = 467,186 total.
  + **Combined**: 76,482 + 448,944 + 467,186 = 992,612 total.
  + **Split**: 70% training (694,828), 20% development (198,522), 5% dev-test (49,631), 5% test (49,631).
* **Dev-Test Set**:
  + **RAID**: 13,371 human-written texts.
  + **LLMDetectAIve**: 19,186 machine-generated texts.
  + **Combined**: 13,371 + 19,186 = 32,557 total.
* **Test Set**:
  + **MixSet**: 10,702 texts.
  + **CUDRT**: 22,978 texts.
  + **IELTS**: 24,700 texts.
  + **NLPeer and PeerSum**: 12,075 texts.
  + **Additional Datasets**: 151,425 texts.
  + **Combined**: 10,702 + 22,978 + 24,700 + 12,075 + 151,425 = 221,880 total.

3.4.3 Semantic Analysis Dataset

* **Training Data**: train.csv (approximately 185.64 MB), containing 69,000 rows.
* **Validation Data**: valid.csv, containing 5,000 rows.

By carefully curating datasets that meet these requirements, the platform ensures that the VLMs and AI text detection models are trained and evaluated on high-quality, diverse, and representative data, leading to improved accuracy and reliability in both grading and AI text detection tasks.

# : Methodology

This section outlines the detailed methodologies for the grading module, AI text detection module, integration with user roles, and role-based access control. These methodologies ensure accurate and efficient grading, robust AI text detection, and secure user interactions.

## 4.1 Grading Module Workflow

The grading module is designed to handle both typed and handwritten exams, ensuring accurate and efficient grading. The workflow varies slightly depending on the format of the exam.

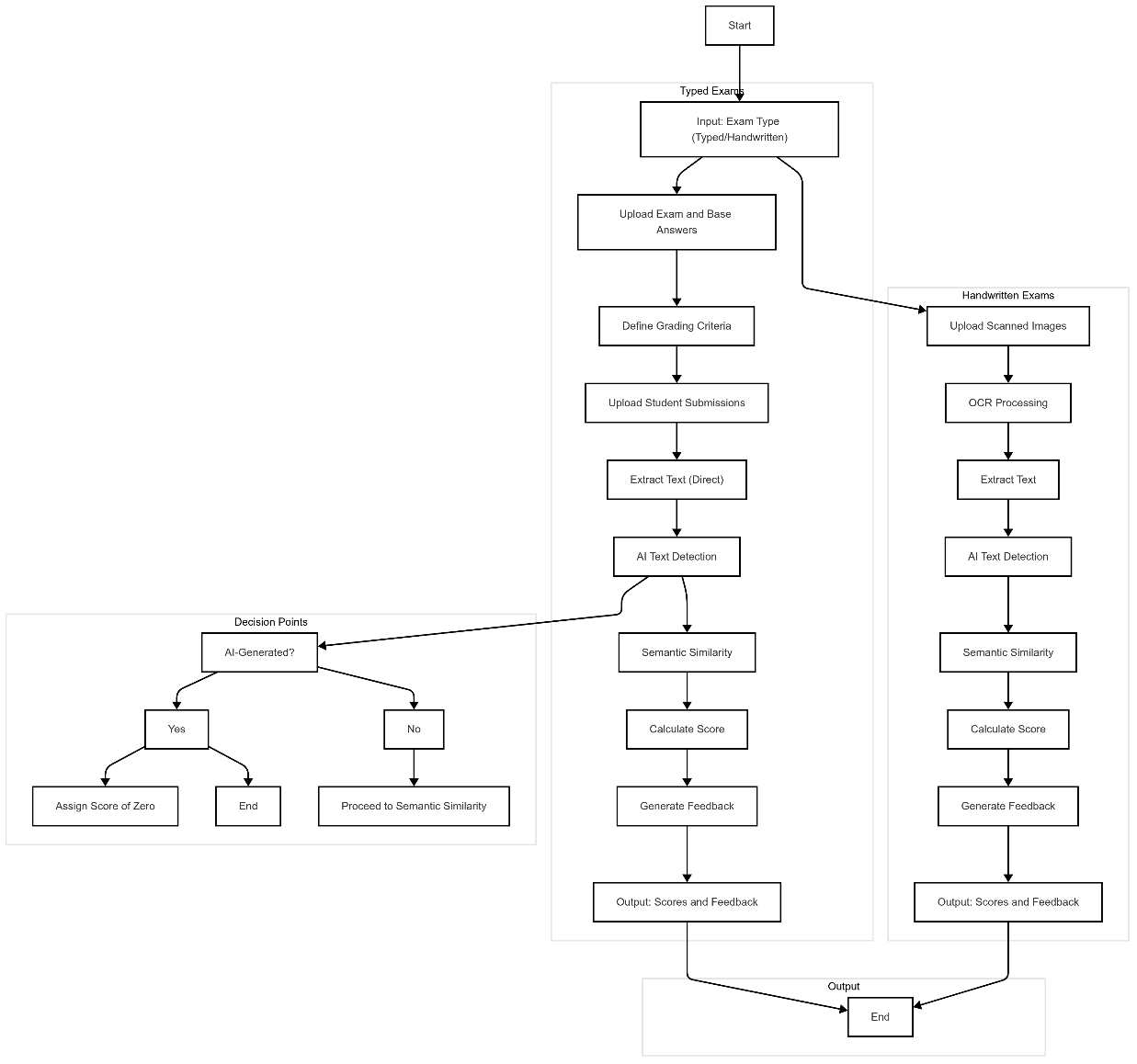


Figure ‎4:1 Grading Module Workflow Diagram

### 4.1.1 Typed Exams (Form-Based)

1. **Input**:
   * **MCQ and T/F Questions**: Student answers are directly matched against the teacher's base answers.
   * **Essay Questions**: Student answers are processed by the model for grading.
2. **Processing**:
   * **MCQ and T/F Questions**:
     + **Matching**: Automatically compare student answers with the teacher's base answers.
     + **Scoring**: Assign points based on the correctness of the answers.
   * **Essay Questions**:
     + **AI Detection**: Use a transformer-based sequence classification model (e.g., DeBERTa) to detect if the essay is AI-generated.
     + **Decision**:
       - If AI-generated, assign a score of zero and do not proceed to the semantic similarity model.
       - If not AI-generated, proceed to the semantic similarity model.
     + **Semantic Similarity**:
       - Compare the student's essay with the teacher's model answer using a semantic similarity model (e.g., RoBERTa).
       - Calculate a similarity score and assign a grade based on the score.
3. **Output**:
   * **Score**: Numerical score for each question.
   * **Feedback**: Detailed feedback highlighting areas of strength and improvement.

### 4.1.2 Handwritten Exams

1. **Input**:
   * Scanned images (PNG/JPG) of handwritten exams.
2. **Preprocessing**:
   * **Image Processing**: Use OCR (Tesseract/OpenCV) to pre-process the images and extract text.
   * **VLM Extraction**: Use Vision-Language Models (VLMs) like Qwen2.5-VL or Gemini 2.0 to interpret and extract text and diagrams.
3. **Processing**:
   * **Text Extraction**: Extract the question and student answer using VLMs.
   * **Structure Matching**: Ensure the extracted structure matches the format of typed exams (e.g., MCQ, T/F, Essay).
4. **Grading**:
   * **MCQ and T/F Questions**: Match student answers against the teacher's base answers.
   * **Essay Questions**:
     + **AI Detection**: Use a transformer-based sequence classification model (e.g., DeBERTa) to detect if the essay is AI-generated.
     + **Decision**:
       - If AI-generated, assign a score of zero and do not proceed to the semantic similarity model.
       - If not AI-generated, proceed to the semantic similarity model.
     + **Semantic Similarity**:
       - Compare the student's essay with the teacher's model answer using a semantic similarity model (e.g., RoBERTa).
       - Calculate a similarity score and assign a grade based on the score.
5. **Output**:
   * **Score**: Numerical score for each question.
   * **Feedback**: Detailed feedback highlighting areas of strength and improvement.

## 4.2 AI Text Detection Workflow

The AI text detection module uses advanced transformer-based models to detect AI-generated text in student submissions.

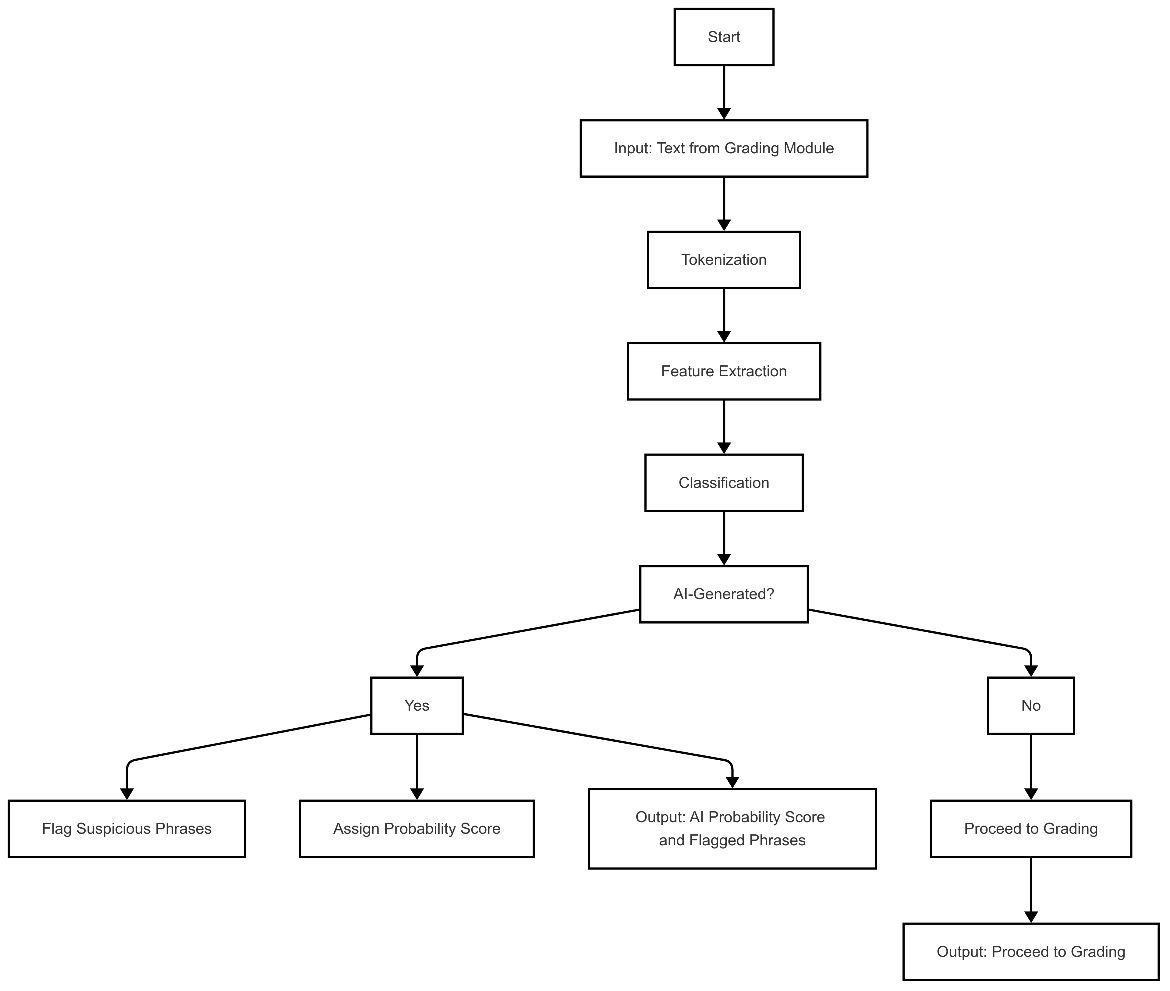


Figure ‎4:2 AI Text Detection Workflow

1. **Input**:
   * Text from the grading module, which could be either typed or handwritten and processed by VLMs.
2. **Processing**:
   * **Tokenization**: Tokenize the input text using a tokenizer (e.g., BERT tokenizer).
   * **Feature Extraction**: Extract features using a transformer-based sequence classification model (e.g., DeBERTa).
   * **Classification**: Use the model to classify the text as either human-written or AI-generated.
3. **Decision**:
   * **AI-Generated**: If the model predicts that the text is AI-generated, flag it and assign a score of zero for the corresponding question.
   * **Human-Written**: If the model predicts that the text is human-written, proceed to the semantic similarity model for grading.
4. **Output**:
   * **AI Probability Score**: Probability score indicating the likelihood of the text being AI-generated.
   * **Flagged Phrases**: Highlighted suspicious sentences if the text is flagged as AI-generated.

## 4.3 Integration with User Roles

The platform integrates seamlessly with different user roles, ensuring that each role has access to the appropriate features and functionalities.

### 4.3.1 Professors

* **Exam Management**:
  + **Upload Exams**: Upload typed or handwritten exams.
  + **Set Grading Criteria**: Define grading rules and scoring schemes.
  + **Adjust Detection Settings**: Fine-tune AI text detection sensitivity.
* **View Grading Results**:
  + **Heatmaps**: Visualize class-wide performance.
  + **Detailed Reports**: View individual student results and feedback.
* **Generate Reports**:
  + Export grading results and reports for further analysis.

### 4.3.2 Students

* **View Personal Results**:
  + **Grade Card**: View scores and feedback for each question.
  + **AI Detection Scores**: See AI probability scores and flagged phrases.
  + **Interactive Feedback**: Expandable sections with detailed explanations and highlighted areas of strength and improvement.

### 4.3.3 Admins

* **User Management**:
  + **Manage Accounts**: Add, edit, and delete user accounts.
  + **Monitor System Health**: Check system performance and troubleshoot issues.
* **System Administration**:
  + **Update Models**: Fine-tune and update AI models with new data.
  + **Perform Audits**: Conduct regular audits to ensure fairness and compliance.

## 4.4 Role-Based Access Control

Role-based access control (RBAC) ensures that each user role has access to appropriate features and functionalities, maintaining security and privacy.

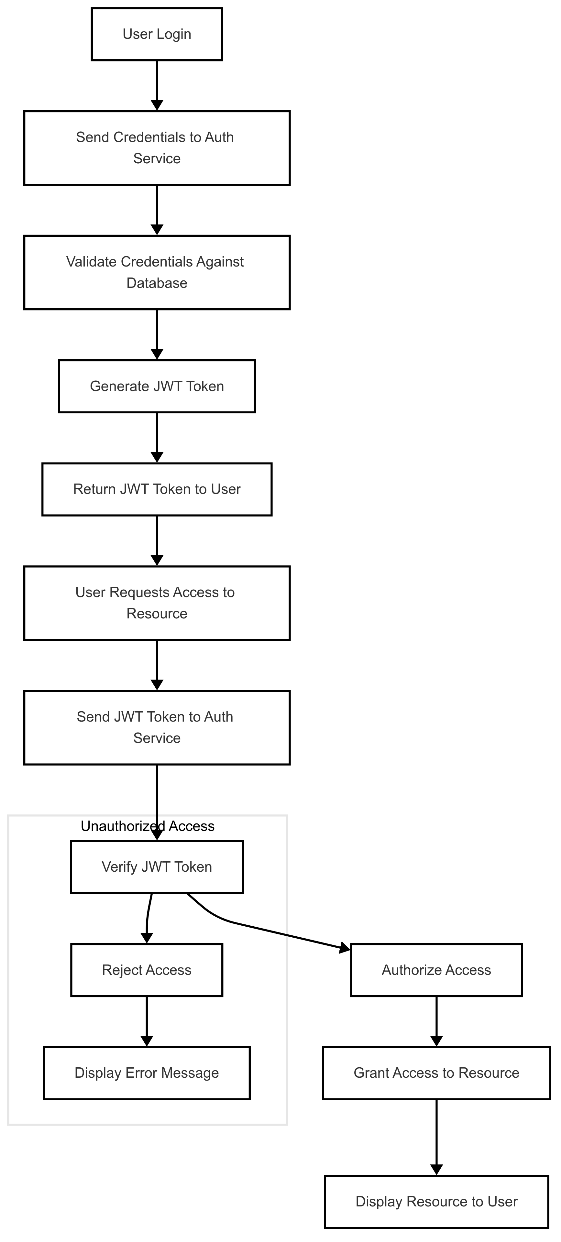


Figure ‎4:3 Role-Based Access Control Diagram

### 4.4.1 JWT Tokens

* **Authentication**: Use JSON Web Tokens (JWT) to authenticate users and verify their roles.
* **Authorization**: Ensure that users can only access features and data relevant to their roles.

### 4.4.2 Role-Specific Permissions

* **Professors**:
  + **Permissions**: Upload exams, set grading criteria, adjust detection settings, view and export grading results.
  + **Restrictions**: Cannot view or modify other professors' exams or results.
* **Students**:
  + **Permissions**: View personal grades and feedback.
  + **Restrictions**: Cannot upload exams or view others' results.
* **Admins**:
  + **Permissions**: Manage user accounts, monitor system health, update models, and perform audits.
  + **Restrictions**: Cannot view individual student grades or results.

By implementing these methodologies, the platform ensures accurate and efficient grading, robust AI text detection, and secure user interactions, enhancing the overall educational experience for both educators and students.

: System Design

## 5.1 Key Features

The AI-powered exam grading and AI text detection platform is designed with several key features to ensure accurate grading, robust AI text detection, and seamless user interaction. The following are the core features of the system:

1. **Automated Grading**:
   * Supports grading for both typed and handwritten exams using advanced OCR and Vision-Language Models (VLMs) like Qwen2.5-VL and Gemini 2.0.
   * Utilizes Natural Language Processing (NLP) models like RoBERTa for semantic analysis and comparison with model answers.
2. **AI Text Detection**:
   * Employs transformer-based models like DeBERTa to detect AI-generated text in student submissions.
   * Provides AI probability scores and highlights flagged phrases for detailed analysis.
3. **User Role Management**:
   * Role-based access control (RBAC) with distinct permissions for professors, students, and admins.
   * Secure authentication and authorization using JSON Web Tokens (JWT).
4. **Interactive Dashboards**:
   * Professors have access to a comprehensive dashboard for managing exams, viewing grading results, and generating reports.
   * Students can view their grades and feedback through a personalized dashboard.
5. **Customizable Settings**:
   * Professors can adjust grading criteria and AI detection sensitivity to suit their specific needs.
   * Admins can update AI models and perform regular audits to ensure system fairness and compliance.
6. **Scalability and Performance**:
   * Designed to handle large volumes of exams and submissions efficiently.
   * Optimized for performance with robust load balancing and auto-scaling capabilities.
7. **Security and Privacy**:
   * Implements data anonymization and encryption to protect student data and ensure privacy.
   * Regular audits and compliance checks to maintain ethical standards and data integrity.

## 5.2 Requirements

### 5.2.1 Assumptions and Dependencies

* **Assumptions**:
  + Users have basic technical proficiency to interact with the platform.
  + The platform will be accessed via standard web browsers on desktops, laptops, and mobile devices.
  + Educators will provide model answers for comparison during the grading process.
* **Dependencies**:
  + Relies on trained models like RoBERTa, DeBERTa, Qwen2.5-VL, and Gemini 2.0 for NLP and VLM functionalities.
  + Requires a stable internet connection for accessing the platform and uploading exams.
  + Dependent on third-party libraries and frameworks

### 5.2.2 Functional Requirements

1. **User Authentication and Authorization**:
   * Implement secure login and role-based access control using JWT.
   * Ensure users can only access features and data relevant to their roles.
2. **Exam Management**:
   * Professors can upload typed or handwritten exams and define grading criteria.
   * The system should support various question types, including MCQ, T/F, and essay questions.
3. **Grading Process**:
   * Automatically grade typed exams by matching student answers with model answers.
   * Use OCR and VLMs to extract and grade handwritten exams accurately.
   * Provide detailed feedback and scores for each question.
4. **AI Text Detection**:
   * Detect AI-generated text in student submissions using transformer-based models.
   * Flag suspicious phrases and provide AI probability scores.
5. **Reporting and Analytics**:
   * Generate detailed reports on class performance and individual student results.
   * Provide visualizations such as heatmaps to help educators identify trends and areas for improvement.
6. **User Interface**:
   * Develop intuitive and user-friendly dashboards for professors, students, and admins.
   * Ensure the interface is responsive and accessible on various devices.

### 5.2.3 Non-Functional Requirements

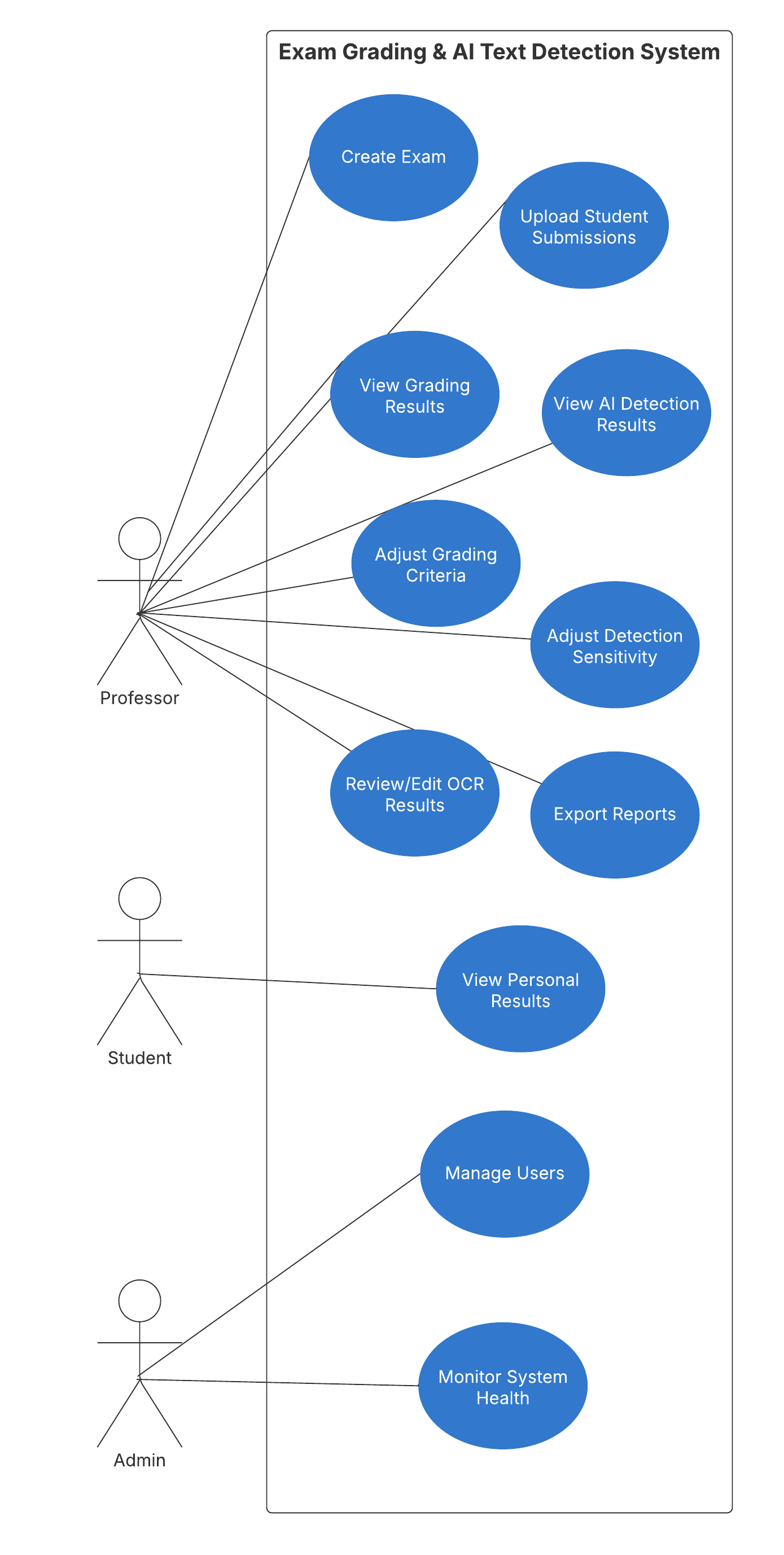
1. **Performance**:
   * The system should handle high volumes of exams and submissions efficiently.
   * Ensure fast response times and minimal latency during peak usage periods.
2. **Scalability**:
   * Design the system to scale horizontally to accommodate increasing user loads.
   * Implement load balancing and auto-scaling to manage varying levels of traffic.
3. **Security**:
   * Protect user data with encryption and secure storage practices.
   * Implement regular security audits and compliance checks.
4. **Usability**:
   * Ensure the platform is easy to navigate and use for all user roles.
   * Provide clear instructions and help documentation for users.
5. **Reliability**:
   * Maintain high system uptime and availability.
   * Implement backup and recovery processes to prevent data loss.
6. **Compliance**:
   * Adhere to data protection regulations such as GDPR.
   * Ensure ethical use of AI and transparency in decision-making processes.

By outlining these key features and requirements, the system design ensures that the platform is robust, secure, and user-friendly, meeting the needs of educators, students, and administrators.

## 5.3 Use Case Diagram

The Use Case Diagram visually represents the interactions between the different actors (Professor, Student, Admin) and the system's functionalities. It provides a clear overview of how each actor interacts with the system to achieve their goals.

Figure ‎4:4: Use Case Diagram

* 1. ****

**Diagram Components**

1. **Actors**:
   * **Professor**: Represents educators who manage exams, view results, and adjust settings.
   * **Student**: Represents learners who view their personal grades and feedback.
   * **Admin**: Represents administrators who manage users, monitor system health, and ensure compliance.
2. **Use Cases**:
   * **Create Exam**: Allows professors to create new exams with defined questions and answer keys.
   * **Upload Student Submissions**: Enables professors to upload student exam submissions for grading.
   * **View Grading Results**: Allows professors to view detailed grading results for individual students and the entire class.
   * **View AI Detection Results**: Provides professors with AI detection results, including probability scores and flagged phrases.
   * **Adjust Grading Criteria**: Allows professors to modify the criteria used for grading exams.
   * **Adjust Detection Sensitivity**: Enables professors to fine-tune the sensitivity of the AI text detection model.
   * **Review/Edit OCR Results**: Allows professors to review and edit OCR results for handwritten exams to ensure accuracy.
   * **Export Reports**: Enables professors to export grading results and reports for further analysis.
   * **View Personal Results**: Allows students to view their grades and feedback.
   * **Manage Users**: Enables admins to add, edit, and delete user accounts.
   * **Monitor System Health**: Allows admins to monitor the system's performance and troubleshoot issues.
   * **Update Models**: Enables admins to update AI models with new data to improve accuracy and performance.
   * **Perform Audits**: Allows admins to perform regular audits to ensure fairness and compliance.

**Relationships**

* **Professor** is associated with the following use cases:
  + Create Exam
  + Upload Student Submissions
  + View Grading Results
  + View AI Detection Results
  + Adjust Grading Criteria
  + Adjust Detection Sensitivity
  + Review/Edit OCR Results
  + Export Reports
* **Student** is associated with the use case:
  + View Personal Results
* **Admin** is associated with the use cases:
  + Manage Users
  + Monitor System Health
  + Perform Audits

## 5.4 Framework Diagram

The System Architecture Diagram provides a visual representation of the platform's structure, highlighting the interactions between different layers and components. The architecture is divided into four main layers: User Layer, Application Layer, Processing Layer, and Data Layer.

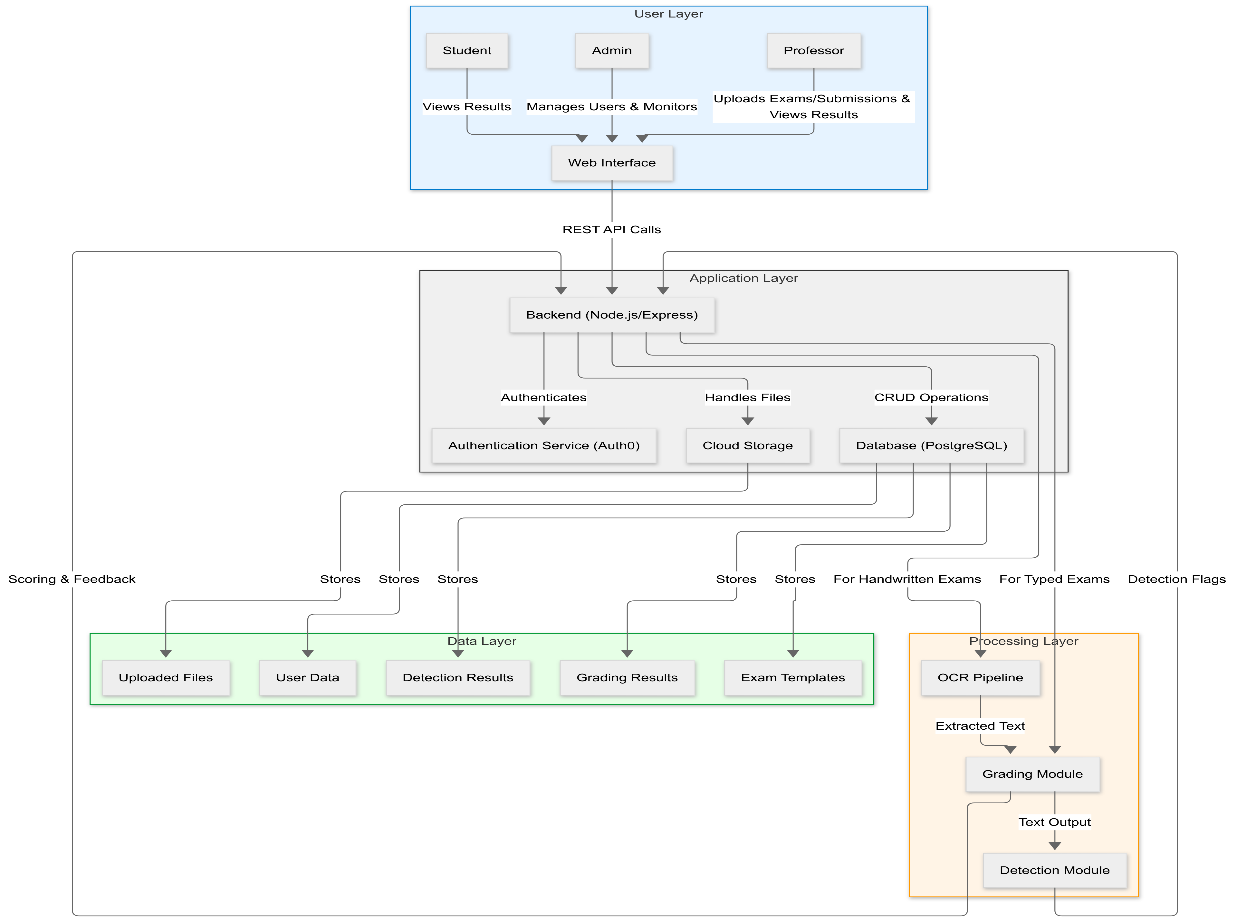


Figure ‎4:5 Framework Diagram

**User Layer**

* **Components**:
  + **Professor**: Interacts with the platform to upload exams and student submissions, view grading results, and adjust settings.
  + **Admin**: Manages user accounts, monitors system health, and performs administrative tasks.
  + **Students**: Interacts with the platform to view exam grading results
  + **Web Interface**: The primary interface through which users interact with the platform.
* **Interactions**:
  + Professors, Students, and admins access the platform through the web interface, which facilitates interactions with the backend services via REST API calls.

**Application Layer**

* **Components**:
  + **Backend (Node.js/Express)**: Handles API requests, manages user authentication, and performs CRUD operations on the database.
  + **Authentication Service (Auth0)**: Manages user authentication and authorization, ensuring secure access to the platform.
  + **Cloud Storage**: Stores uploaded files, including exam templates and student submissions.
* **Interactions**:
  + The backend communicates with the authentication service to verify user credentials and manage sessions.
  + CRUD operations are performed on the database to manage user data, exam templates, grading results, and detection results.
  + File handling operations are managed through cloud storage.

**Processing Layer**

* **Components**:
  + **OCR Pipeline**: Processes handwritten exams to extract text using Optical Character Recognition (OCR) techniques.
  + **Grading Module**: Scores exams based on predefined criteria and provides feedback.
  + **Detection Module**: Analyzes text to detect AI-generated content and flags suspicious phrases.
* **Interactions**:
  + Handwritten exams are processed through the OCR pipeline to extract text, which is then sent to the grading module.
  + Typed exams are preprocessed and sent directly to the grading module.
  + The grading module generates scores and feedback, which are sent back to the backend.
  + The detection module analyzes the text output from the grading module to identify AI-generated content and sends detection flags back to the backend.

**Data Layer**

* **Components**:
  + **Database (PostgreSQL)**: Stores all persistent data, including user data, exam templates, grading results, and detection results.
  + **Cloud Storage**: Stores uploaded files, such as exam templates and student submissions.
* **Interactions**:
  + The database stores structured data related to users, exams, grading results, and detection results.
  + Cloud storage handles the storage of unstructured data, such as uploaded files.

This architecture ensures a modular and scalable design, facilitating efficient data flow, secure user interactions, and robust performance. It provides a clear separation of concerns, allowing each layer to focus on specific functionalities, thus enhancing maintainability and extensibility.

## 5.5 Entity-Relationship Diagram (ERD)

The Entity-Relationship Diagram (ERD) illustrates the relationships between the key entities within the AI-powered exam grading and AI text detection platform. It provides a visual representation of how data is structured and interconnected within the system.

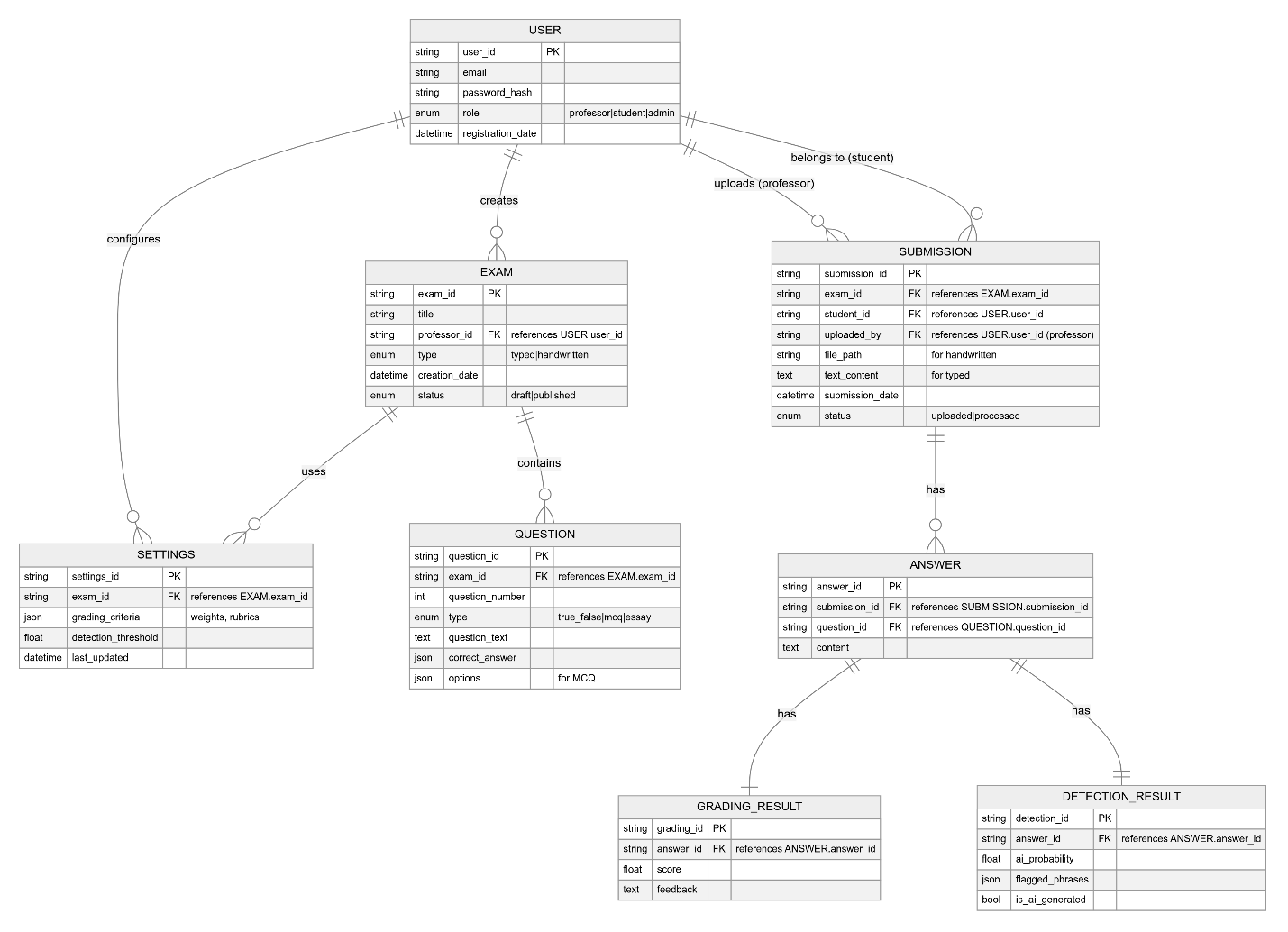


Figure ‎4:6 Entity-Relationship Diagram

**Entities and Relationships**

1. **USER**
   * **Attributes**:
     + user\_id (Primary Key): Unique identifier for each user.
     + email: User's email address.
     + password\_hash: Hashed password for security.
     + role: Enum indicating the user's role (professor, student, admin).
     + registration\_date: Date when the user registered.
   * **Relationships**:
     + A USER can create multiple EXAMs.
     + A USER (professor) can upload multiple SUBMISSIONs.
     + A USER (student) owns their SUBMISSIONs.
     + A USER can configure multiple SETTINGS.
2. **EXAM**
   * **Attributes**:
     + exam\_id (Primary Key): Unique identifier for each exam.
     + title: Title of the exam.
     + professor\_id (Foreign Key): References the user\_id of the professor who created the exam.
     + type: Enum indicating whether the exam is typed or handwritten.
     + creation\_date: Date when the exam was created.
     + status: Enum indicating the exam status (draft or published).
   * **Relationships**:
     + An EXAM contains multiple QUESTIONs.
     + An EXAM uses a set of SETTINGS.
3. **QUESTION**
   * **Attributes**:
     + question\_id (Primary Key): Unique identifier for each question.
     + exam\_id (Foreign Key): References the exam\_id of the exam it belongs to.
     + question\_number: Sequential number of the question within the exam.
     + type: Enum indicating the type of question (true\_false, mcq, essay).
     + question\_text: Text of the question.
     + correct\_answer: JSON object containing the correct answer.
     + options: JSON object containing options for MCQs.
   * **Relationships**:
     + A QUESTION is contained within an EXAM.
4. **SUBMISSION**
   * **Attributes**:
     + submission\_id (Primary Key): Unique identifier for each submission.
     + exam\_id (Foreign Key): References the exam\_id of the exam it belongs to.
     + student\_id (Foreign Key): References the user\_id of the student who submitted it.
     + uploaded\_by (Foreign Key): References the user\_id of the professor who uploaded it.
     + file\_path: Path to the uploaded file for handwritten exams.
     + text\_content: Text content for typed exams.
     + submission\_date: Date when the submission was made.
     + status: Enum indicating the submission status (uploaded or processed).
   * **Relationships**:
     + A SUBMISSION has multiple ANSWERs.
     + A SUBMISSION belongs to a USER (student).
5. **ANSWER**
   * **Attributes**:
     + answer\_id (Primary Key): Unique identifier for each answer.
     + submission\_id (Foreign Key): References the submission\_id of the submission it belongs to.
     + question\_id (Foreign Key): References the question\_id of the question it answers.
     + content: Text content of the answer.
   * **Relationships**:
     + An ANSWER has a GRADING\_RESULT.
     + An ANSWER has a DETECTION\_RESULT.
6. **GRADING\_RESULT**
   * **Attributes**:
     + grading\_id (Primary Key): Unique identifier for each grading result.
     + answer\_id (Foreign Key): References the answer\_id of the answer it grades.
     + score: Numerical score for the answer.
     + feedback: Text feedback for the answer.
   * **Relationships**:
     + A GRADING\_RESULT is associated with an ANSWER.
7. **DETECTION\_RESULT**
   * **Attributes**:
     + detection\_id (Primary Key): Unique identifier for each detection result.
     + answer\_id (Foreign Key): References the answer\_id of the answer it analyzes.
     + ai\_probability: Probability score indicating the likelihood of AI-generated text.
     + flagged\_phrases: JSON object containing flagged phrases.
     + is\_ai\_generated: Boolean indicating if the text is AI-generated.
   * **Relationships**:
     + A DETECTION\_RESULT is associated with an ANSWER.
8. **SETTINGS**
   * **Attributes**:
     + settings\_id (Primary Key): Unique identifier for each settings configuration.
     + exam\_id (Foreign Key): References the exam\_id of the exam it applies to.
     + detection\_threshold: Threshold for AI text detection sensitivity.
     + last\_updated: Date when the settings were last updated.
   * **Relationships**:
     + SETTINGS are used by an EXAM.
     + SETTINGS are configured by a USER.

## 5.6 Data Flow Diagrams

### Context DFD (Level 0)

The Context DFD provides a high-level overview of the Exam Grading and Text Detection System, illustrating how it interacts with external entities.

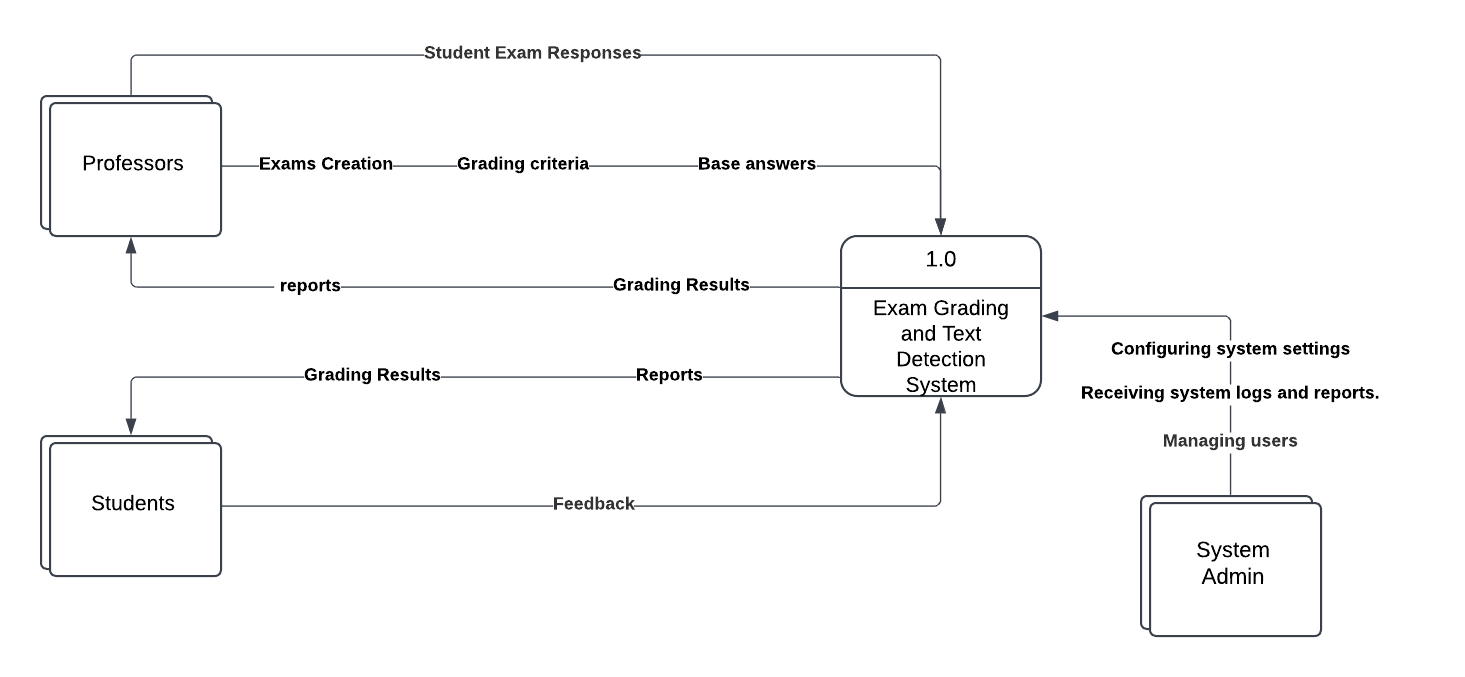


Figure ‎4:7 Context Data Flow Diagram

**Central Process**

* **1.0 Exam Grading and Text Detection System**: This is the core system being modeled, encompassing all functionalities related to exam grading and AI text detection.

**External Entities**

1. **Professors**:
   * **Inputs**:
     + Provide exam creation details.
     + Set grading criteria.
     + Upload base answers.
   * **Outputs**:
     + Receive reports and grading results from the system.
2. **Students**:
   * **Inputs**:
     + Provide student exam responses.
   * **Outputs**:
     + Receive grading results and feedback from the system.
3. **System Admin**:
   * **Inputs**:
     + Configure system settings.
   * **Outputs**:
     + Receive system logs and reports.
     + Manage users.

**Data Flows**

* **Exam Creation**: From professors to the system.
* **Grading Criteria**: From professors to the system.
* **Base Answers**: From professors to the system.
* **Student Exam Responses**: From students to the system.
* **Reports and Grading Results**: From the system to professors and students.
* **System Logs and Reports**: From the system to the admin.
* **User Management**: From the admin to the system.

The Context DFD provides a high-level view of the interactions between the system and its external entities, showing what data flows into and out of the system.

### Level 1 DFD

The Level 1 DFD expands on the Context DFD by breaking down the main process into several sub-processes and data stores, providing a more detailed view of the system's internal workings.

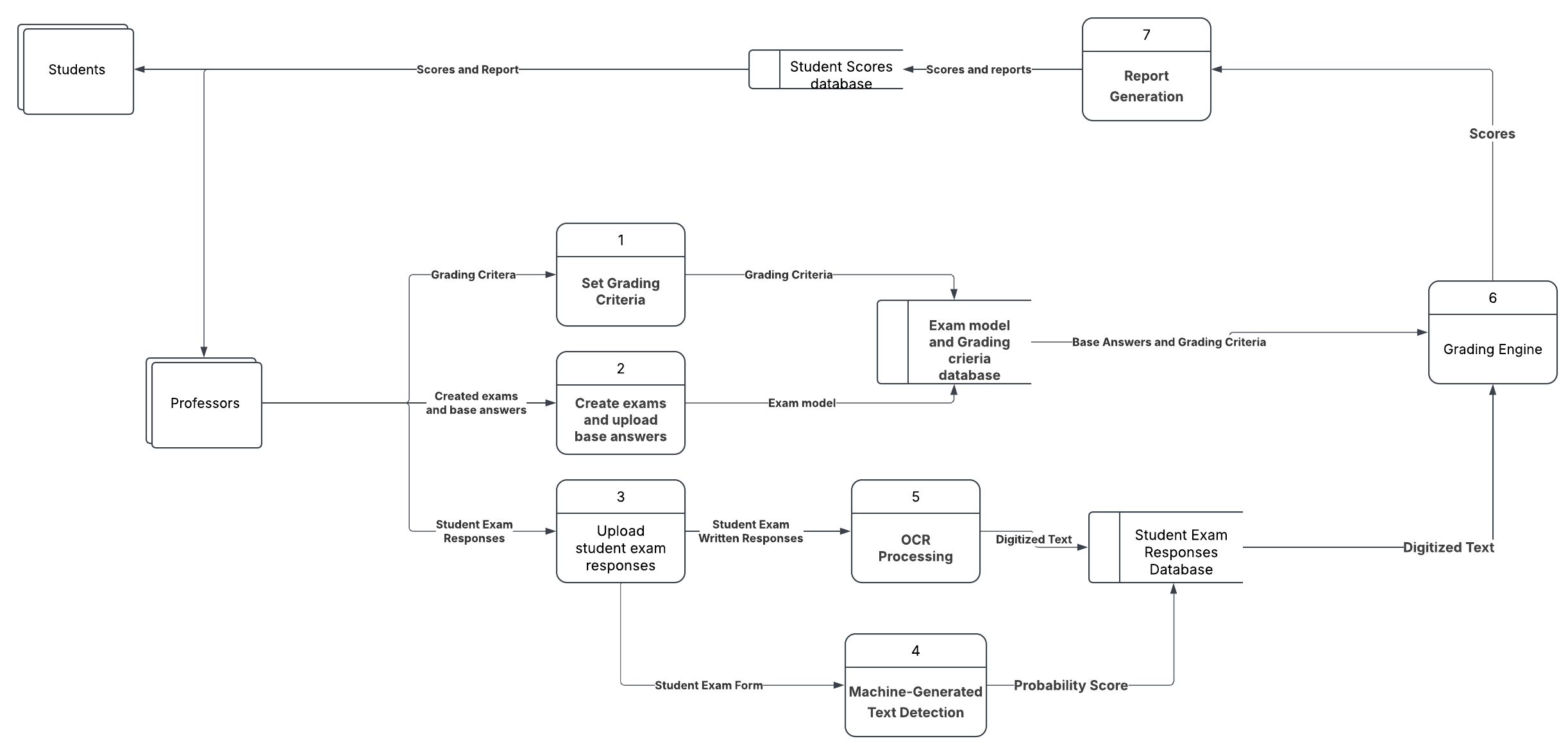


Figure ‎4:8 Level 1 Data Flow Diagram

**External Entities**

1. **Students**:
   * Send "Student Exam Responses" to the system.
   * Receive "Scores and Reports" from the system.
2. **Professors**:
   * Set grading criteria.
   * Create exams and upload base answers.
   * Upload student exam responses.

**Processes**

1. **Set Grading Criteria**:
   * **Input**: Grading criteria from professors.
   * **Output**: Stored in the Exam Model and Grading Criteria Database.
2. **Create Exams and Upload Base Answers**:
   * **Input**: Exam details and base answers from professors.
   * **Output**: Stored as "Exam Model" in the database.
3. **Upload Student Exam Responses**:
   * **Input**: Student exam responses from professors.
   * **Output**: "Student Exam Written Responses" sent for OCR processing.
4. **Machine-Generated Text Detection**:
   * **Input**: Student exam responses.
   * **Output**: Probability score indicating AI-generated text, stored in the Student Exam Responses Database.
5. **OCR Processing**:
   * **Input**: Student exam written responses.
   * **Output**: Digitized text sent to the Student Exam Responses Database.
6. **Grading Engine**:
   * **Input**: Digitized text and base answers.
   * **Output**: Scores sent to the Student Scores Database.
7. **Report Generation**:
   * **Input**: Scores from the Student Scores Database.
   * **Output**: Reports sent to students and professors.

**Data Stores**

1. **Exam Model and Grading Criteria Database**:
   * Stores grading criteria, exam structure, and correct answers provided by professors.
2. **Student Exam Responses Database**:
   * Stores digitized student responses and probability scores from the text detection process.
3. **Student Scores Database**:
   * Stores scores and reports generated by the grading engine.

**Data Flows**

* **Grading Criteria**: From "Set Grading Criteria" process to the Exam Model and Grading Criteria Database.
* **Exam Model**: From "Create Exams and Upload Base Answers" process to the Exam Model and Grading Criteria Database.
* **Student Exam Written Responses**: From "Upload Student Exam Responses" process to OCR Processing.
* **Digitized Text**: From OCR Processing to the Student Exam Responses Database.
* **Probability Score**: From Machine-Generated Text Detection to the Student Exam Responses Database.
* **Scores**: From Grading Engine to the Student Scores Database.
* **Reports**: From Report Generation to students and professors.

The Level 1 DFD provides a detailed view of the internal workings of the Exam Grading and Text Detection System, clarifying the steps involved in processing exams from setting criteria and uploading responses to grading and generating reports. It highlights the system's capabilities for handling handwritten responses and identifying AI-generated content.

## 5.7 Class Diagram

The Class Diagram represents the structure of the system by showing the system's classes, their attributes, methods, and the relationships between them. This diagram is crucial for understanding the design and interactions within the system.

A diagram of a company

AI-generated content may be incorrect.

Figure ‎4:9 Class Diagram

**Classes and Attributes**

1. **User (Abstract Class)**
   * **Attributes**:
     + user\_id: Unique identifier for each user.
     + email: User's email address.
     + password\_hash: Hashed password for security.
     + registration\_date: Date when the user registered.
   * **Methods**:
     + login(): Authenticates the user.
     + logout(): Logs the user out of the system.
   * **Relationships**:
     + Inherited by Student, Professor, and Admin.
2. **Student**
   * **Attributes**:
     + student\_id: Unique identifier for each student.
     + name: Name of the student.
   * **Methods**:
     + viewResults(): Allows the student to view their exam results and feedback.
   * **Relationships**:
     + Inherits from User.
3. **Professor**
   * **Attributes**:
     + professor\_id: Unique identifier for each professor.
     + department: Department the professor belongs to.
   * **Methods**:
     + createExam(): Creates a new exam.
     + uploadSubmissions(): Uploads student submissions for grading.
     + generateReport(): Generates reports on exam results.
   * **Relationships**:
     + Inherits from User.
     + Creates multiple Exam objects.
     + Uploads multiple Submission objects.
     + Configures Settings.
4. **Admin**
   * **Attributes**:
     + admin\_id: Unique identifier for each admin.
   * **Methods**:
     + manageUsers(): Manages user accounts.
     + monitorSystem(): Monitors system health and performance.
   * **Relationships**:
     + Inherits from User.
5. **Exam**
   * **Attributes**:
     + exam\_id: Unique identifier for each exam.
     + title: Title of the exam.
     + type: Enum indicating whether the exam is typed or handwritten.
     + creation\_date: Date when the exam was created.
     + status: Enum indicating the exam status (draft or published).
   * **Methods**:
     + storeToDatabase(): Stores the exam details in the database.
   * **Relationships**:
     + Contains multiple Question objects.
     + Stored in the Database.
6. **Question**
   * **Attributes**:
     + question\_id: Unique identifier for each question.
     + question\_number: Sequential number of the question within the exam.
     + type: Enum indicating the type of question (true\_false, mcq, essay).
     + question\_text: Text of the question.
     + correct\_answer: JSON object containing the correct answer.
     + options: JSON object containing options for MCQs.
   * **Relationships**:
     + Belongs to an Exam.
7. **Submission**
   * **Attributes**:
     + submission\_id: Unique identifier for each submission.
     + file\_path: Path to the uploaded file for handwritten exams.
     + text\_content: Text content for typed exams.
     + submission\_date: Date when the submission was made.
     + status: Enum indicating the submission status (uploaded or processed).
   * **Methods**:
     + processOCR(): Processes handwritten submissions using OCR.
   * **Relationships**:
     + Has a GradingResult.
     + Has a DetectionResult.
     + Stored in the Database.
8. **GradingResult**
   * **Attributes**:
     + grading\_id: Unique identifier for each grading result.
     + score: Numerical score for the answer.
     + feedback: Text feedback for the answer.
     + rubric\_details: JSON object containing rubric details.
   * **Methods**:
     + calculateSimilarity(): Calculates the similarity score between the student's answer and the model answer.
   * **Relationships**:
     + Associated with a Submission.
     + Stored in the Database.
9. **DetectionResult**
   * **Attributes**:
     + detection\_id: Unique identifier for each detection result.
     + ai\_probability: Probability score indicating the likelihood of AI-generated text.
     + flagged\_phrases: JSON object containing flagged phrases.
     + is\_ai\_generated: Boolean indicating if the text is AI-generated.
   * **Methods**:
     + analyzeContent(): Analyzes the text to detect AI-generated content.
   * **Relationships**:
     + Associated with a Submission.
     + Stored in the Database.
10. **Settings**
    * **Attributes**:
      + settings\_id: Unique identifier for each settings configuration.
      + grading\_criteria: JSON object containing grading criteria (weights, rubrics).
      + detection\_threshold: Threshold for AI text detection sensitivity.
    * **Methods**:
      + updateSettings(): Updates the settings configuration.
    * **Relationships**:
      + Configured by a Professor.
11. **Database**
    * **Methods**:
      + storeExam(): Stores exam details.
      + fetchExam(): Fetches exam details.
      + storeSubmission(): Stores submission details.
      + fetchGrades(): Fetches grading results.
      + storeDetectionResults(): Stores detection results.
    * **Relationships**:
      + Stores Exam, Submission, GradingResult, and DetectionResult.
12. **AuthService**
    * **Methods**:
      + validateCredentials(): Validates user credentials.
      + generateToken(): Generates authentication tokens.
    * **Relationships**:
      + Authenticates User.

## 5.8 Activity Diagrams

Activity Diagrams illustrate the workflows and processes within the system, highlighting the sequence of actions, decisions, and outcomes for various activities.

**1. User Registration Activity Diagram**

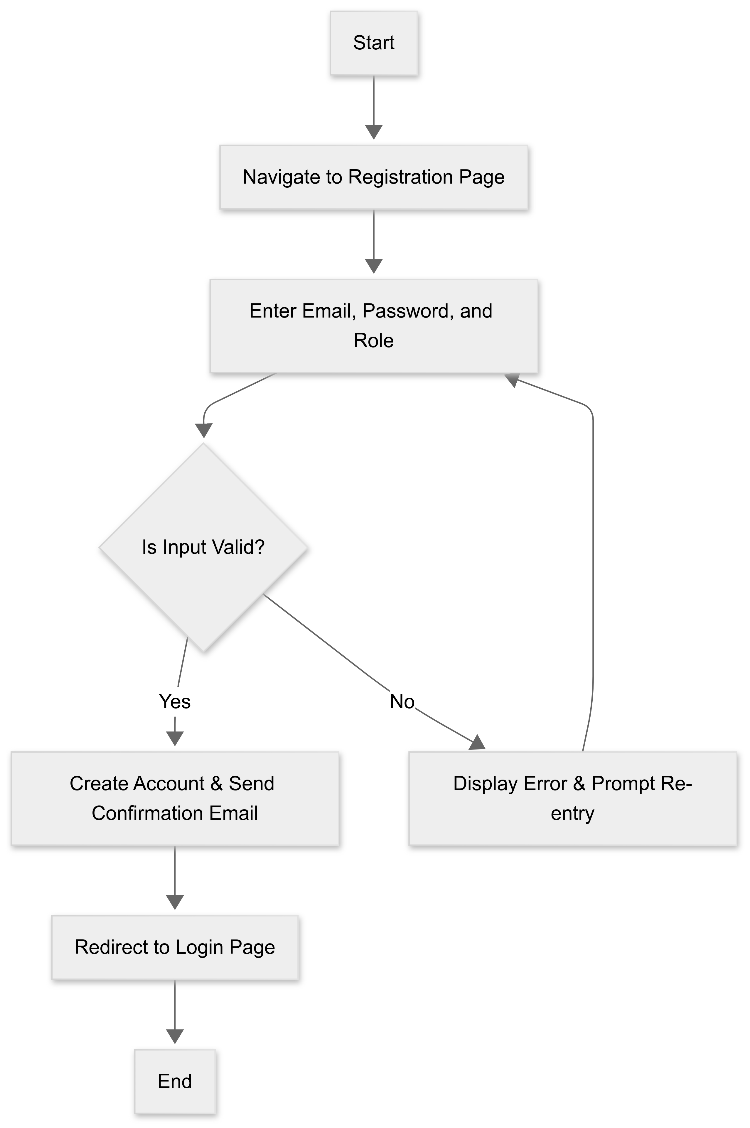


Figure ‎4:10 User Registration Activity Diagram

**2. User Login Activity Diagram**

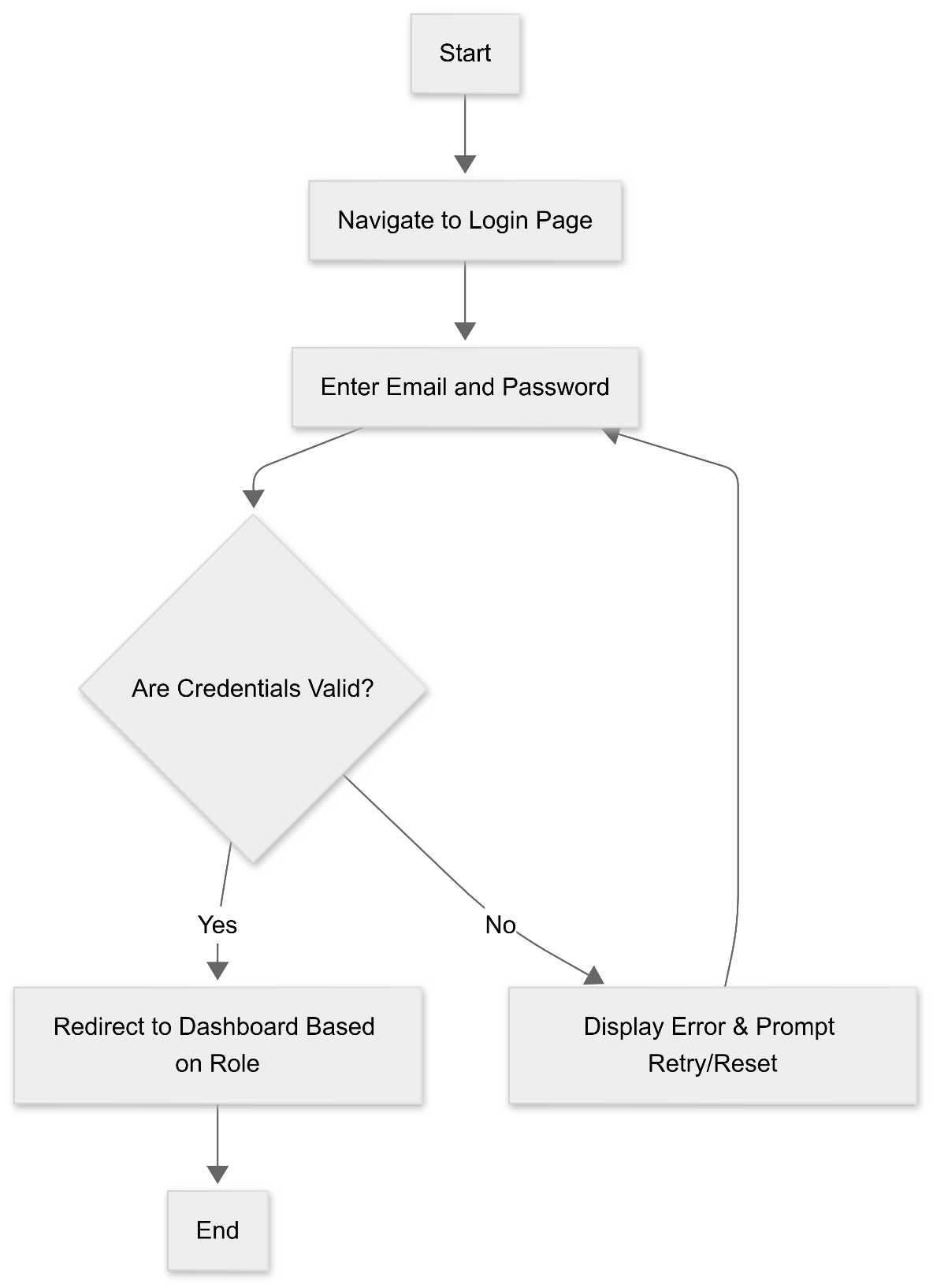


Figure ‎4:11 User Login Activity Diagram

A diagram of a flowchart

AI-generated content may be incorrect.**3. Exam Creation Activity Diagram**

Figure ‎4:12 Exam Creation Activity Diagram

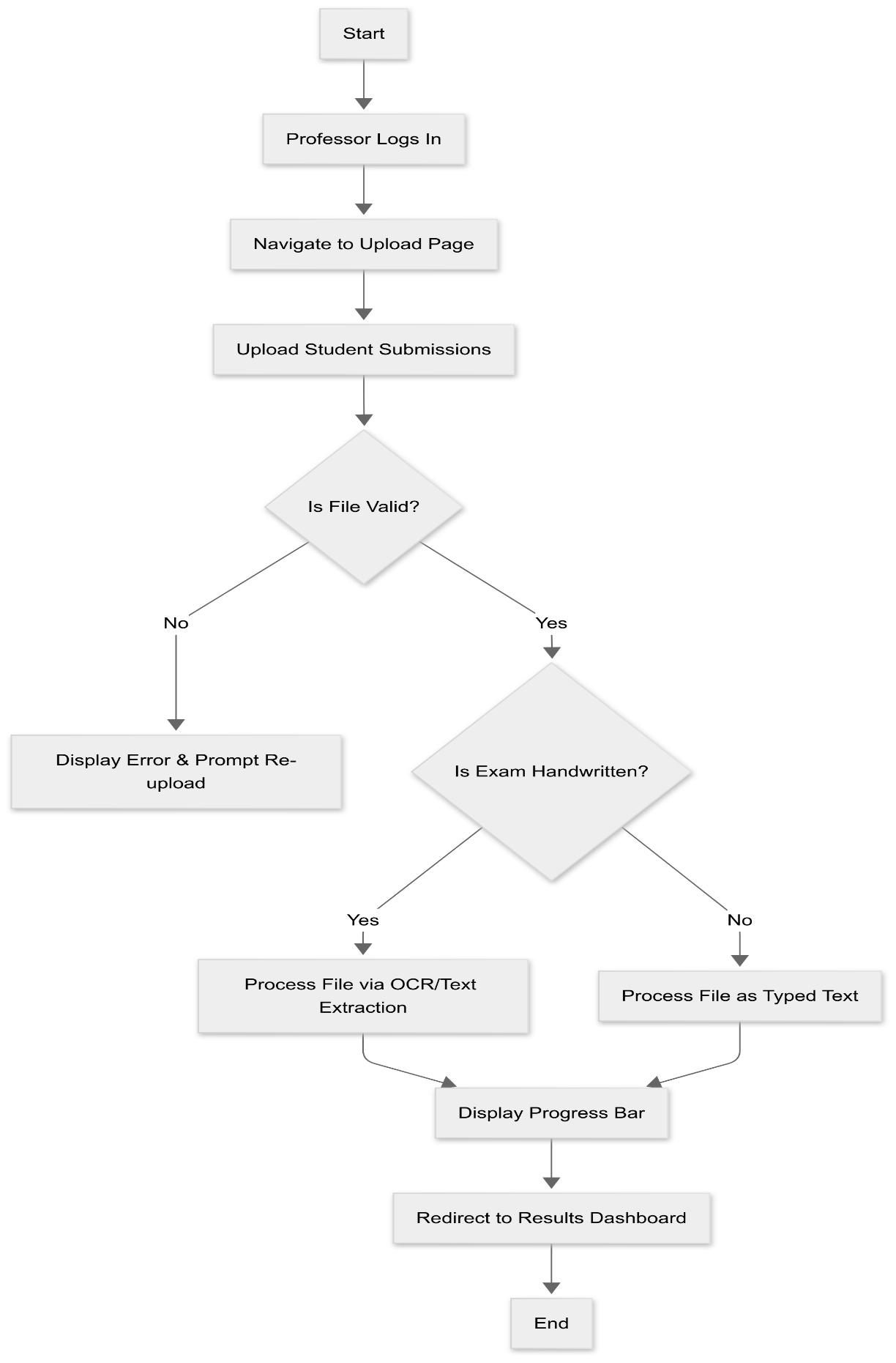
**4. Exam Upload Activity Diagram**

Figure ‎4:13 Exam Upload Activity Diagram

**5. OCR Processing Activity Diagram**

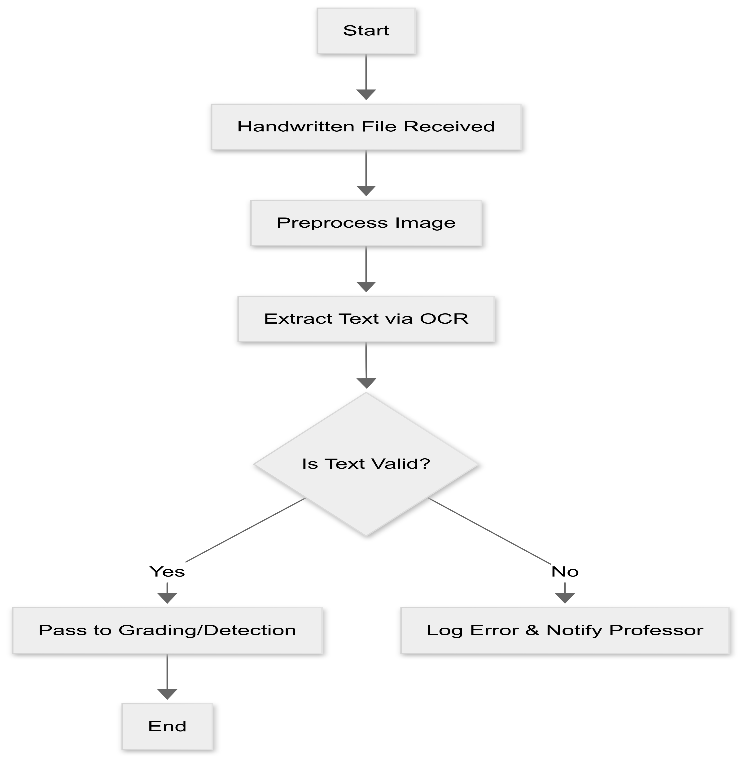


Figure ‎4:14 OCR Processing Activity Diagram

**6. Grading Activity Diagram**

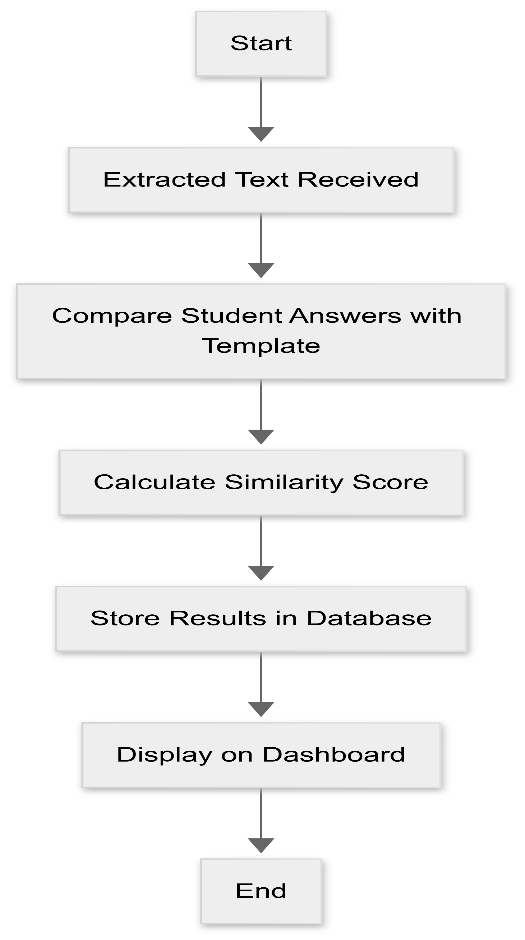


Figure ‎4:15 Grading Activity Diagram

**7. AI Text Detection Activity Diagram**

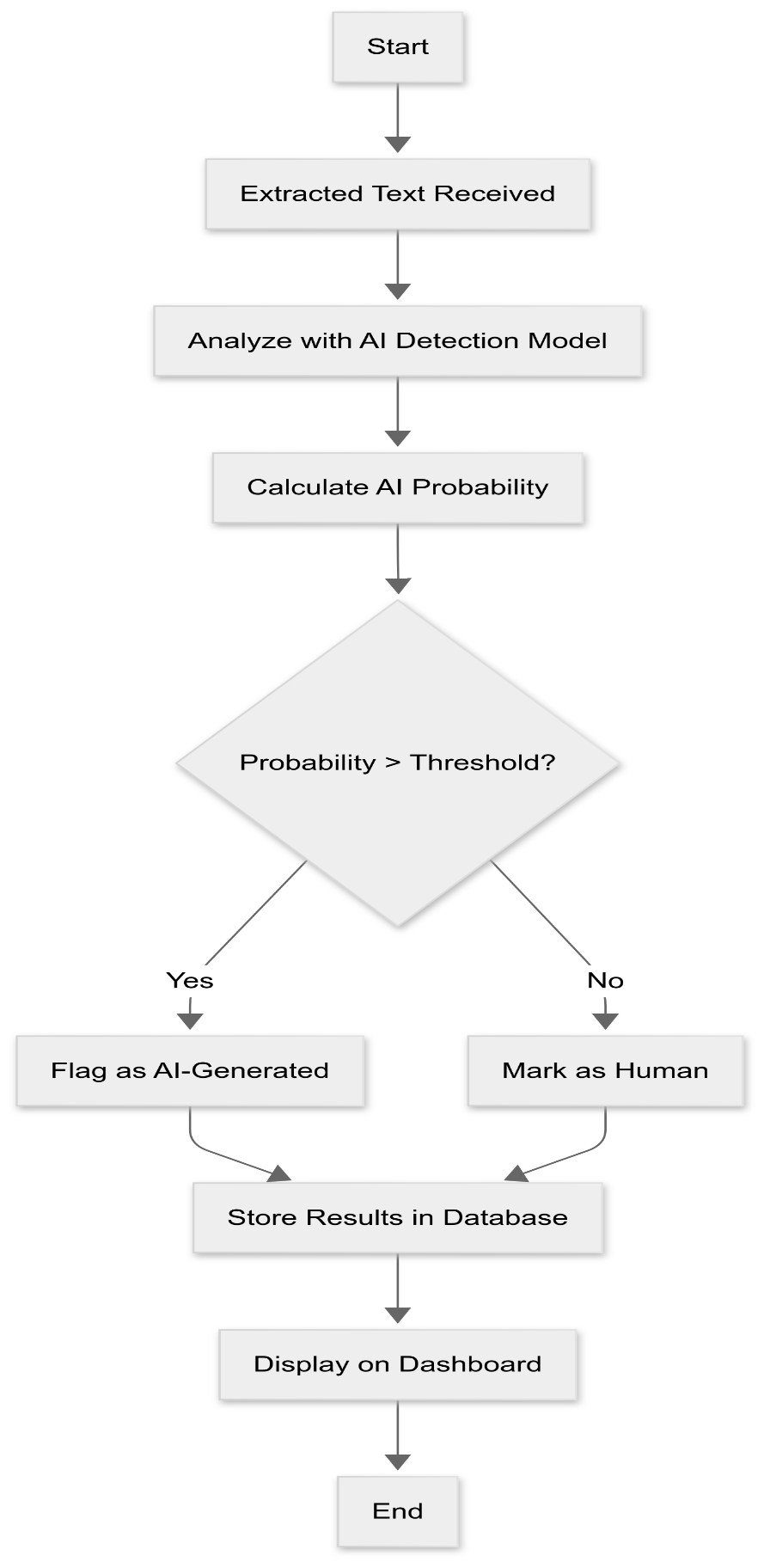


Figure ‎4:16 AI Text Detection Activity Diagram

**8. Results Dashboard Activity Diagram**

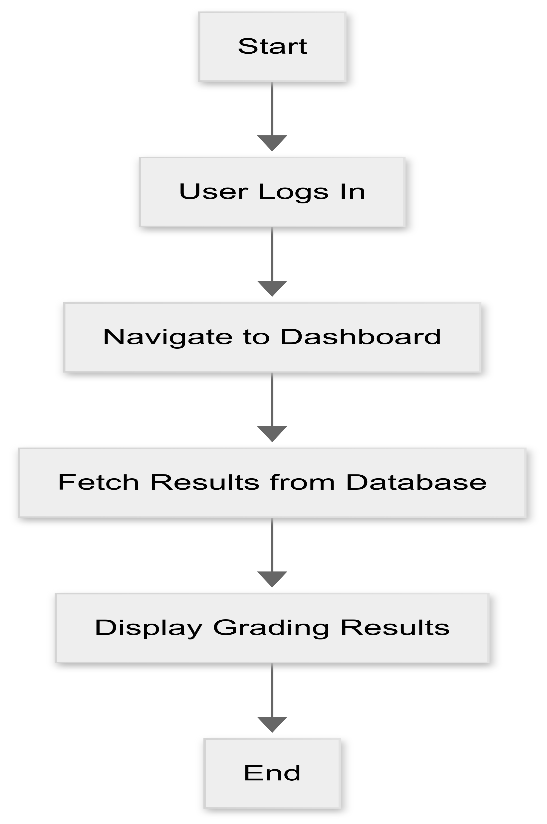


Figure ‎4:17 Results Dashboard Activity Diagram

**9. Settings Customization Activity Diagram**

A diagram of a flowchart

AI-generated content may be incorrect.

Figure ‎4:18 Settings Customization Activity Diagram

**10. Admin Management Activity Diagram**

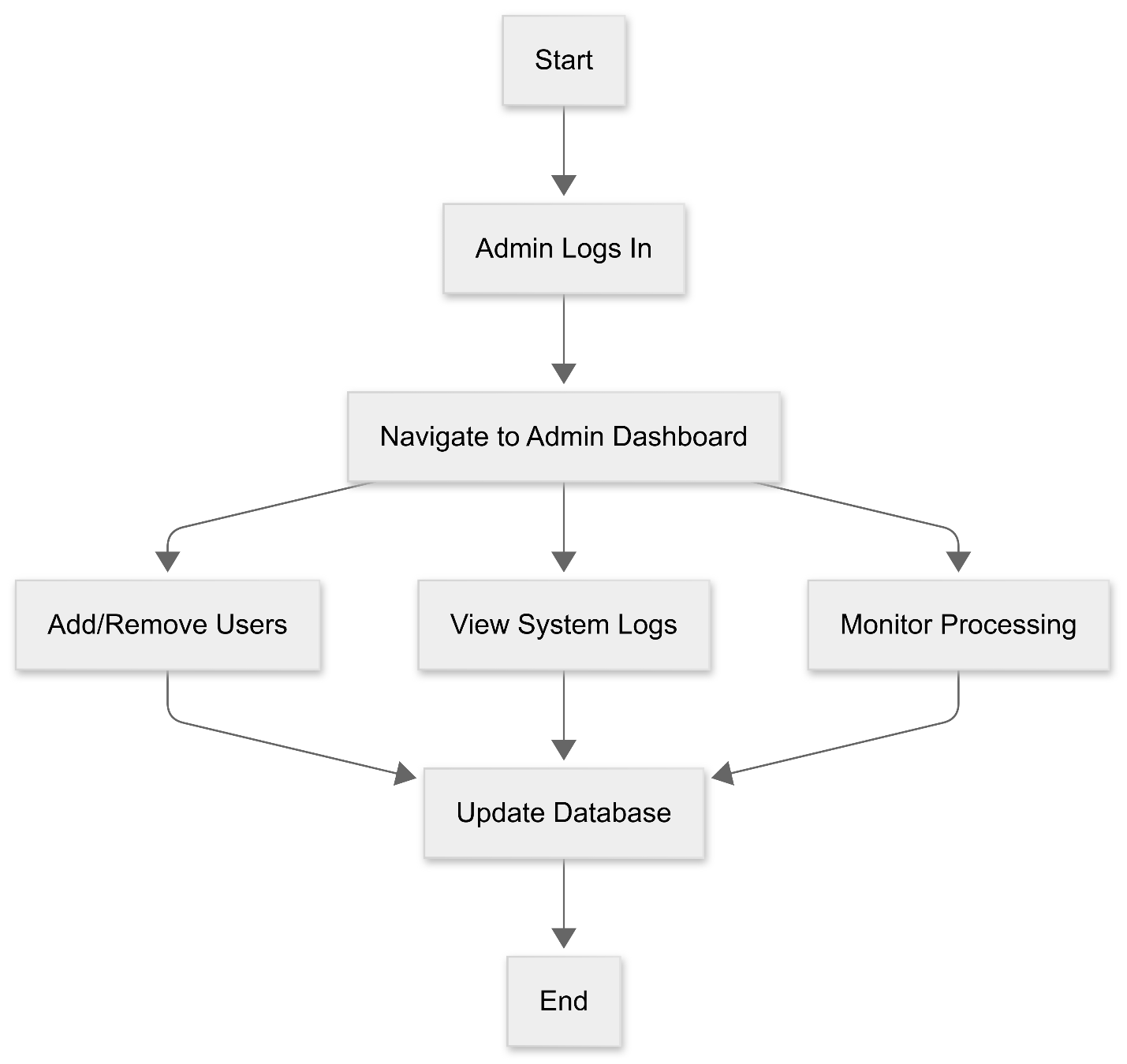
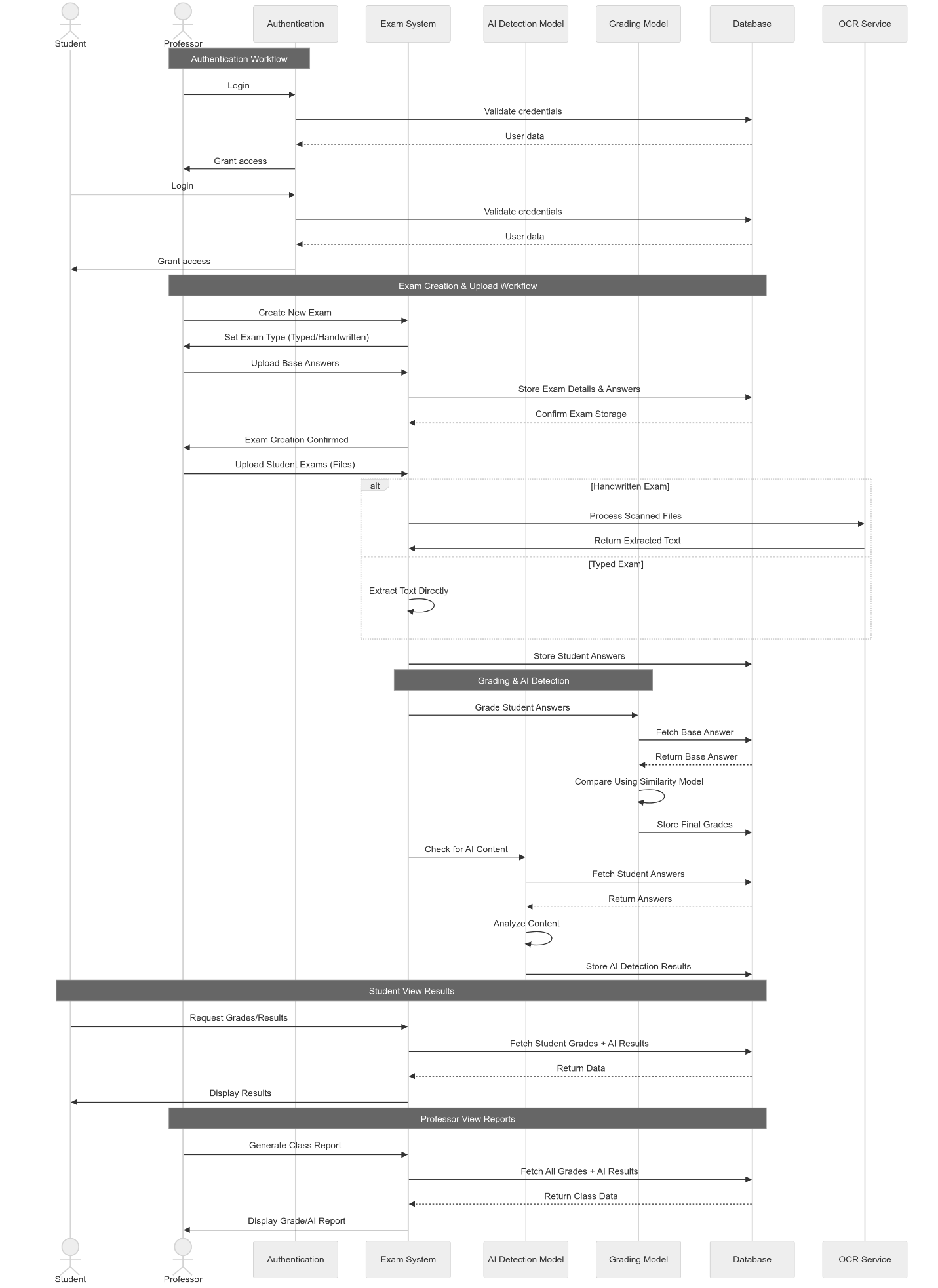


Figure ‎4:19 Admin Management Activity Diagram

## 5.9 Sequence Diagram

The Sequence Diagram illustrates the interactions between actors (Professor, Student) and the system's components over time, highlighting the workflows for authentication, exam creation, grading, AI detection, and results viewing.



**Actors**

1. **Professor**: Interacts with the system to create exams, upload submissions, and view reports.
2. **Student**: Interacts with the system to view their grades and feedback.

**Participants**

1. **Auth (Authentication)**: Manages user authentication and authorization.
2. **ExamSys (Exam System)**: Core component handling exam creation, uploads, and grading workflows.
3. **AIModel (AI Detection Model)**: Analyzes text to detect AI-generated content.
4. **Grading (Grading Model)**: Grades student answers based on similarity to model answers.
5. **DB (Database)**: Stores exam details, student answers, grades, and AI detection results.
6. **OCR (OCR Service)**: Processes scanned files to extract text for handwritten exams.

**Workflows**

1. **Authentication Workflow**
   * **Professor**:
     + Logs in by sending credentials to Auth.
     + Auth validates credentials with DB and grants access upon successful validation.
   * **Student**:
     + Logs in by sending credentials to Auth.
     + Auth validates credentials with DB and grants access upon successful validation.
2. **Exam Creation & Upload Workflow**
   * **Professor**:
     + Creates a new exam by specifying the exam type (typed/handwritten) and uploading base answers to ExamSys.
     + ExamSys stores exam details and answers in DB.
     + Confirms exam creation to the professor.
   * **Professor**:
     + Uploads student exams (files) to ExamSys.
     + If the exam is handwritten, ExamSys sends scanned files to OCR for processing and receives extracted text.
     + If the exam is typed, ExamSys extracts text directly.
     + ExamSys stores student answers in DB.
3. **Grading & AI Detection Workflow**
   * **ExamSys**:
     + Sends student answers to Grading for grading.
     + Grading fetches base answers from DB, compares them using a similarity model, and stores final grades in DB.
   * **ExamSys**:
     + Sends student answers to AIModel for AI content detection.
     + AIModel fetches student answers from DB, analyzes the content, and stores AI detection results in DB.
4. **Student View Results Workflow**
   * **Student**:
     + Requests grades/results from ExamSys.
     + ExamSys fetches student grades and AI results from DB and displays them to the student.
5. **Professor View Reports Workflow**
   * **Professor**:
     + Requests to generate a class report from ExamSys.
     + ExamSys fetches all grades and AI results from DB and displays the report to the professor.

**Notes**

* **Authentication Workflow**: Ensures secure access to the system for both professors and students.
* **Exam Creation & Upload Workflow**: Handles the creation and storage of exams, including processing handwritten exams using OCR.
* **Grading & AI Detection Workflow**: Manages the grading of student answers and detection of AI-generated content, storing results in the database.
* **Student View Results Workflow**: Allows students to view their grades and AI detection results.
* **Professor View Reports Workflow**: Enables professors to generate and view class-wide reports on grades and AI detection results.

This Sequence Diagram provides a clear and structured view of the interactions and workflows within the system, highlighting the key processes and data flows involved in exam grading and AI text detection. It is essential for understanding the dynamic behavior of the system and ensuring that all components work together seamlessly.

## 5.10 Flowchart Diagram

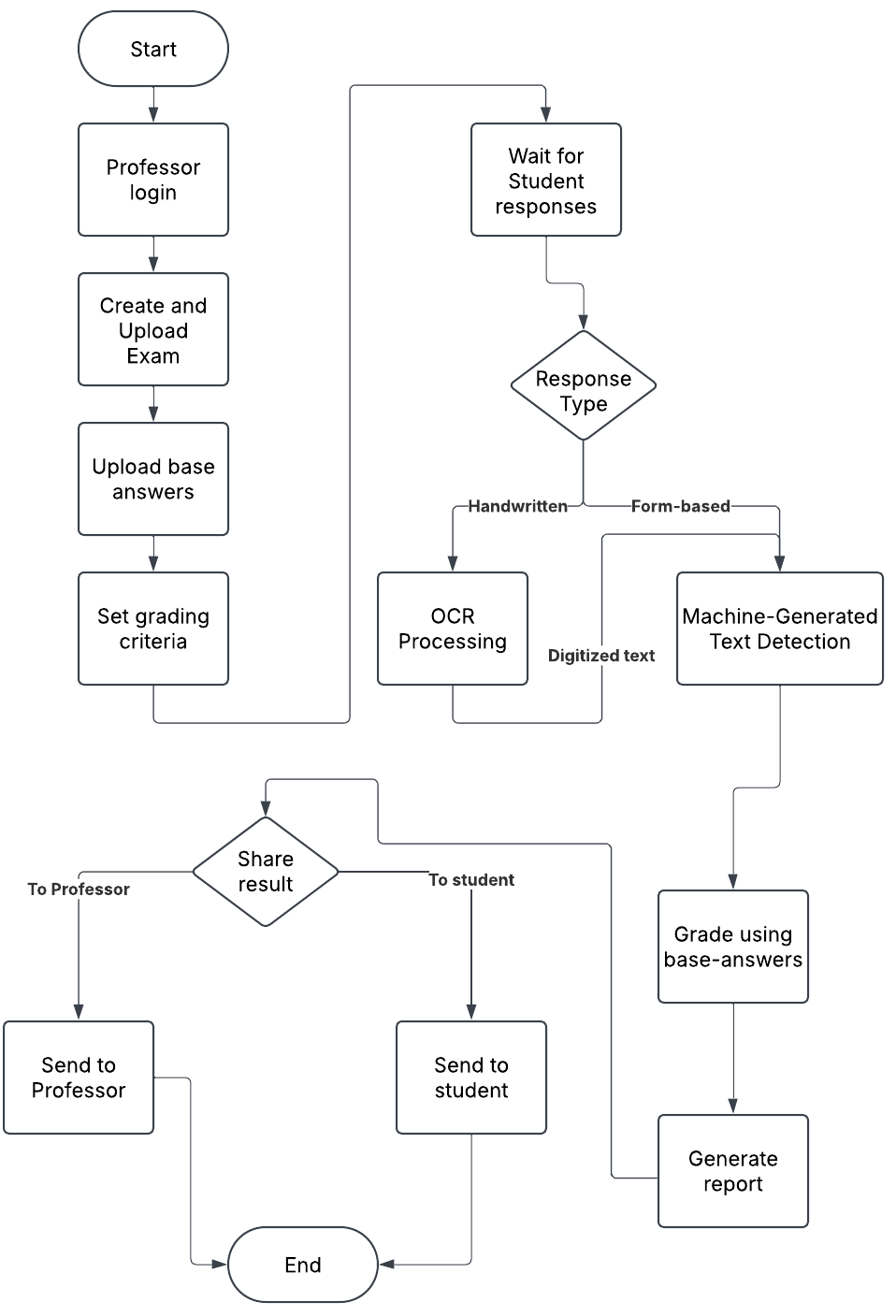
The Flowchart Diagram illustrates the step-by-step process of the Exam Grading and Text Detection System, highlighting the key actions, decisions, and workflows involved in managing exams from creation to grading and reporting.

Figure ‎4:20 Flowchart Diagram

**Flowchart Elements**

1. **Start/End**: Rounded rectangles indicate the beginning and end of the process.
2. **Process**: Rectangles represent actions or steps in the process.
3. **Decision**: Diamonds represent points where a decision is made, with different paths based on the outcome.
4. **Flow Lines**: Arrows show the direction of the process flow.

**Process Flow**

1. **Start**: The process begins.
2. **Professor Login**: The professor logs into the system.
3. **Create and Upload Exam**: The professor creates the exam and uploads it to the system.
4. **Upload Base Answers**: The professor uploads the correct answers (answer key) for the exam.
5. **Set Grading Criteria**: The professor defines the criteria for grading, such as point values for each question.
6. **Wait for Student Responses**: The system waits for students to submit their answers.
7. **Response Type (Decision)**: The system determines the type of response:
   * **Handwritten**: If handwritten, the response goes through "OCR Processing" to convert it to "Digitized Text."
   * **Form-based**: If form-based (e.g., multiple-choice, online form), it goes to "Machine-Generated Text Detection."
8. **OCR Processing**: Handwritten responses are processed using Optical Character Recognition (OCR) to convert them into digital text.
9. **Machine-Generated Text Detection**: Form-based responses are checked for potential AI-generated content or plagiarism.
10. **Grade Using Base Answers**: Both digitized text and the results from the Machine-Generated Text Detection go to this process, where they are compared to the base answers, and grades are calculated.
11. **Generate Report**: A report is generated with the grading results.
12. **Share Result (Decision)**: A decision is made on whether the score will be shared with the professor, the student, or both.
13. **Send to Professor/Send to Student**: Share the grade depending on the decision.
14. **End**: The process concludes.

**Key Features Highlighted**

* **Professor-Centric**: The flowchart primarily focuses on the professor's actions in setting up and managing the exam.
* **Automated Grading**: The system automates the grading process by comparing student responses to the uploaded base answers.
* **Handling Different Response Types**: The system can handle both handwritten and form-based (digital) responses, with OCR for handwritten submissions.
* **AI Text Detection**: The system includes a feature to detect potentially machine-generated text, addressing academic integrity concerns.

**Report Generation**: The system provides a summary report of the grading results.

## 5.11 System Workflow

A diagram of a flowchart

AI-generated content may be incorrect.The System Workflow diagram illustrates the sequence of operations and interactions within the Exam Grading and Text Detection System, from authentication to results reporting. It is divided into several subgraphs to highlight different stages of the workflow.

Figure ‎4:21 System Workflow

**Authentication**

1. **Login**:
   * **Actors**: Professor/Student
   * **Process**:
     + Users (Professor/Student) log in through the Authentication Service.
     + The service validates credentials with the Database.
     + Upon successful validation, access is granted to the System Dashboard.

**Professor Workflow**

1. **Create Exam**:
   * **Process**:
     + The professor creates a new exam via the System Dashboard.
     + The exam type is determined (Typed or Handwritten).
2. **Upload Base Answers**:
   * **Typed Exam**:
     + Base answers are uploaded in form format and stored in CSV, then converted to JSON.
   * **Handwritten Exam**:
     + Base answers are uploaded as scanned images and processed using OCR (Tesseract/OpenCV) before being stored in JSON format.
   * **Storage**:
     + Both formats are stored in the Database under "Exam Templates."
3. **Upload Student Exams**:
   * **Process**:
     + The professor uploads student exams to the system.
     + The exam type is determined (Typed or Handwritten).
   * **Typed Exam**:
     + Text is extracted directly and stored in the Database.
   * **Handwritten Exam**:
     + Text is processed using OCR and then stored in the Database.

**System Processing**

1. **Grading Module**:
   * **Process**:
     + Fetches base answers from the Database.
     + Calculates similarity scores using a model like RoBERTa.
     + Stores grading results in the Database.
2. **AI Detection Module**:
   * **Process**:
     + Analyzes text using models like DeBERTa along with statistical features.
     + Stores AI detection results in the Database.

**Results & Reporting**

1. **View Personal Grades**:
   * **Actors**: Student
   * **Process**:
     + Students view their personal grades and AI detection results from the Database.
2. **Generate Class Report**:
   * **Actors**: Professor
   * **Process**:
     + The professor generates a class report based on grading and AI detection results.
     + The report can be exported as PDF/CSV.

**Key Features Highlighted**

* **Authentication**: Ensures secure access to the system for both professors and students.
* **Exam Creation and Upload**: Handles both typed and handwritten exams, with OCR processing for handwritten submissions.
* **Grading and AI Detection**: Automates the grading process and detects AI-generated content, storing results in the database.
* **Results Reporting**: Provides personalized grade views for students and comprehensive class reports for professors.

# : Web Application

The web application for the AI-powered exam grading and AI text detection platform is designed to provide seamless and intuitive user experience for both professors and students. The application consists of five primary screens, each serving a specific purpose in the workflow of exam creation, grading, and results viewing. Below is all the screens:

## 6.1 Registration Screen

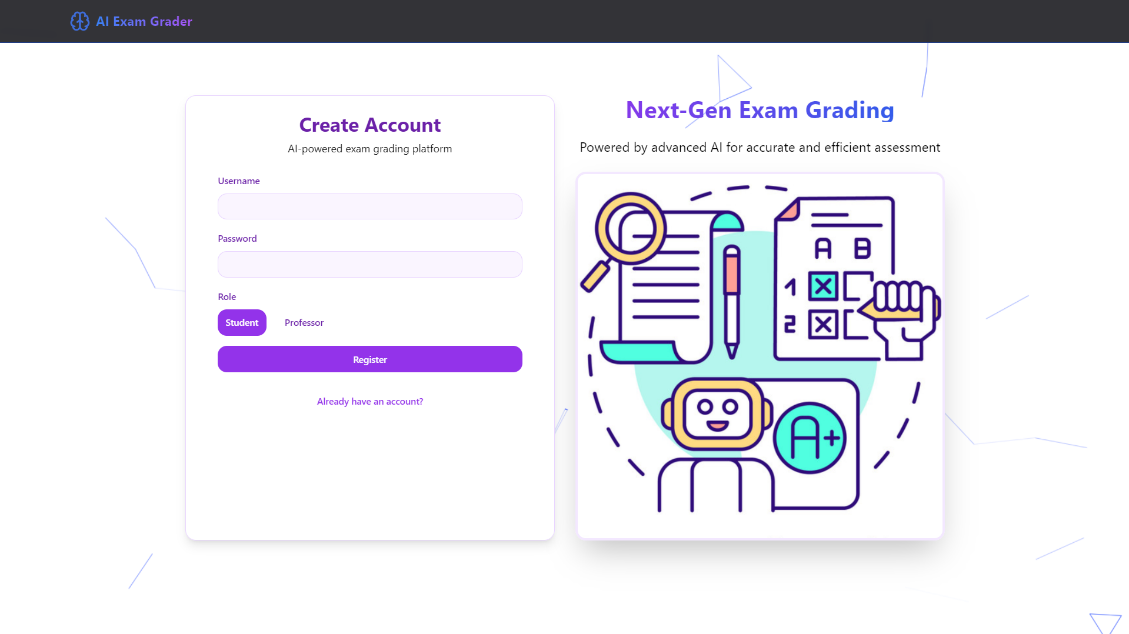


Figure ‎6:1 Registration Screen

## 6.2 Login Screen

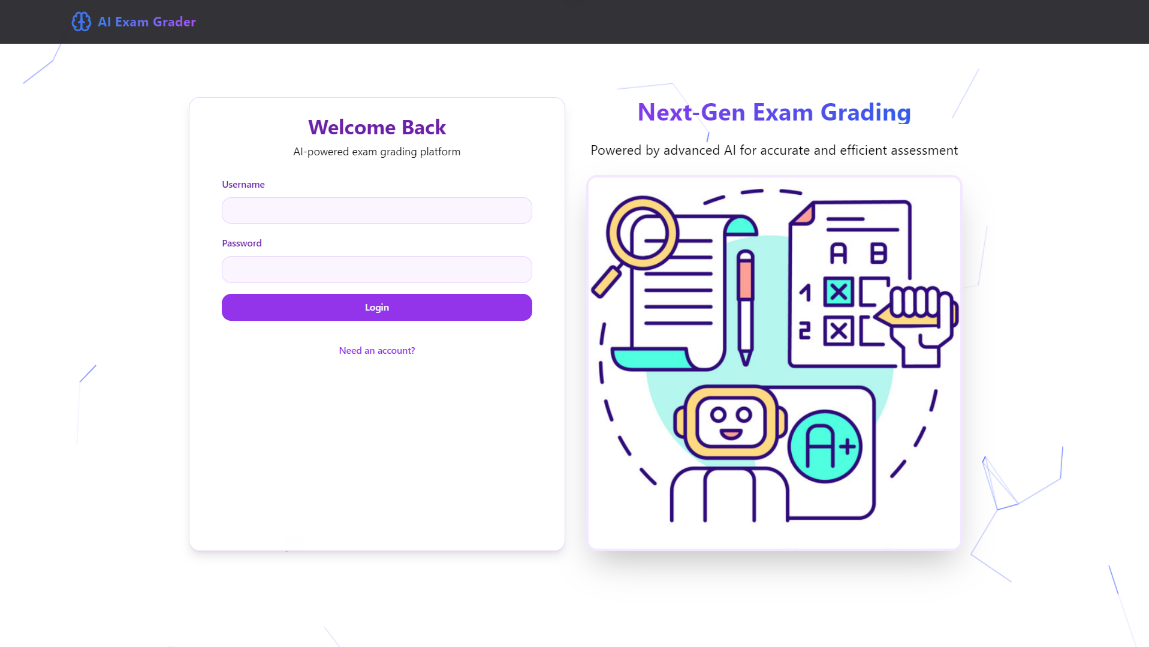


Figure ‎6:2 Login Screen

## 6.3 Professor Dashboard

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‎6:3 Professor Dashboard

## 6.4 Student Dashboard

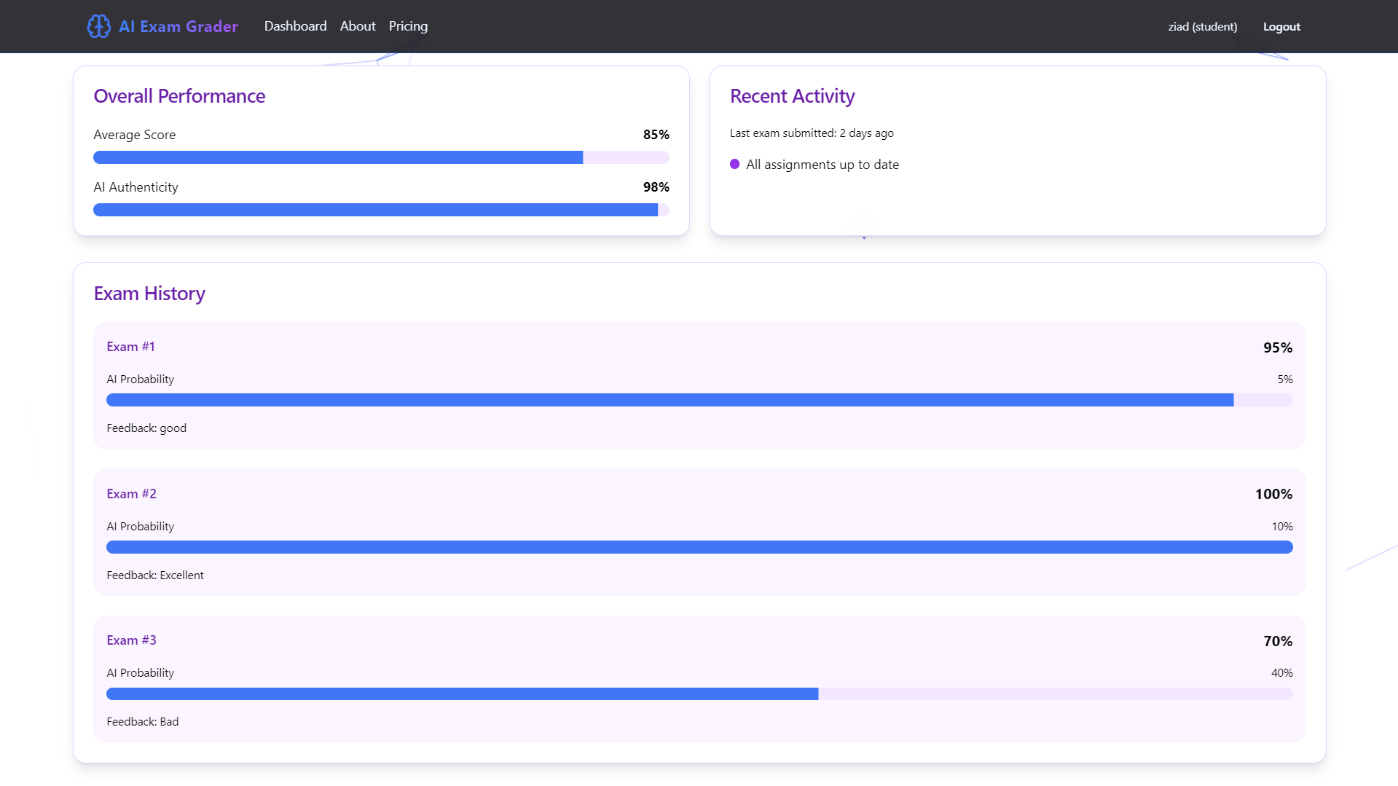


Figure ‎6:4 Student Dashboard

## 6.5 About Screen

A screenshot of a computer

AI-generated content may be incorrect.

Figure ‎6:5 About Screen

# : Conclusion

The AI-powered exam grading and AI text detection platform represents a significant advancement in educational technology, offering a comprehensive solution for automated and efficient exam assessment. By leveraging advanced AI technologies, including Optical Character Recognition (OCR), Vision-Language Models (VLMs), and Natural Language Processing (NLP), the platform successfully addresses the challenges of grading both typed and handwritten exams, while also ensuring academic integrity through AI text detection.

Throughout this report, we have detailed the system's architecture, workflows, and functionalities, providing a clear understanding of how the platform operates and interacts with its users. The inclusion of various diagrams, such as use case diagrams, data flow diagrams, and sequence diagrams, has helped illustrate the system's processes and interactions, ensuring clarity and coherence in its design.

The web application, with its intuitive interface and dedicated screens for registration, login, and dashboards for both professors and students, provides a user-friendly experience that streamlines the exam management process. The platform's ability to handle different response types, automate grading, and detect AI-generated content positions it as a valuable tool for educators seeking to enhance their assessment methods.

Moreover, the acknowledgment of contributions from various stakeholders, including educators, technical experts, and researchers, underscores the collaborative nature of this project. Their insights and support have been instrumental in developing a platform that not only meets the technical requirements but also addresses the practical needs of the educational community.

In conclusion, the AI-powered exam grading and AI text detection platform demonstrates the potential of AI to revolutionize educational assessment. By providing accurate, efficient, and secure grading solutions, the platform paves the way for future innovations in educational technology, ultimately enhancing the learning experience for both students and educators.

# References

[1] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805.

[2] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692.

[3] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. IEEE International Conference on Computer Vision (ICCV).

[4] Smith, L. N. (2017). Cyclical Learning Rates for Training Neural Networks. 2017 IEEE Winter Conference on Applications of Computer Vision (WACV).

[5] Tesseract OCR Documentation. (n.d.). Retrieved from https://github.com/tesseract-ocr/tesseract

[6] Hugging Face Transformers Library. (n.d.). Retrieved from https://huggingface.co/transformers/

[7] PyTorch Documentation. (n.d.). Retrieved from https://pytorch.org/

[8] COLING 2025 Workshop on MGT Detection Task 1. (n.d.). Retrieved from https://github.com/mbzuai-nlp/COLING-2025-Workshop-on-MGT-Detection-Task1