كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

Helwan University

Securing Success

# Guarded Exam

## Plagiarism Detection & Auto-Grading


**By**

Mahmoud Gamal Abu Saree
Mahmoud Khaled Mahmoud
Mohamed Khaled Ahmed
Mahmoud Adel Ali
Mohamed Khaled Abd El-Badie
Soliman Mohamed Mohamed
Ali Abdrabo Ali

**Under Supervision of**

**Dr.** Salwa Osama


A Graduation Project Report

**Submitted to**

The Faculty of Computers and Artificial Intelligence at Helwan University
In Partial Fulfillment of the Requirements
For a Degree of Bachelor of Computer Science
In
Artificial Intelligence
Faculty of Computers and Artificial Intelligence

**June 2025**

كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

Helwan University

Securing Success

# الامتحان المحروس

## كشف الانتحال والتصحيح التلقائي

**مقدمة من:**


محمود جمال أبو سريع
محمود خالد محمود
محمد خالد احمد
محمود عادل علي
محمد خالد عبد البديع
سليمان محمد محمد
علي عبد ربه علي


**تحت إشراف**

**دكتور سلوى أسامة**

تقرير مشروع تخرج

**مقدم الى**

كلية الحاسبات والذكاء الاصطناعي بجامعة حلوان

استكمالًا جزئيًا لمتطلبات الحصول على درجة البكالوريوس في علوم الحاسوب

في

الذكاء الاصطناعي

كلية الحاسبات و الذكاء الاصطناعي

**يونيو/حزيران ٢٠٢٥**

# Acknowledgement

This project would not have been possible without the support and guidance of many individuals and institutions. First and foremost, I express my sincere gratitude to our supervisor, Dr. Salwa Osama, for her exceptional mentorship, patience, and insightful feedback throughout the journey. Her expertise in machine learning and NLP was instrumental in shaping this project.

We also extend our appreciation to the AI Department for their continuous support and constructive feedback during our development phases. Our deepest thanks go to our colleagues, whose collaboration, code reviews, and moral support made the research process both enjoyable and productive.

We would also like to acknowledge the open-source community, particularly Hugging Face, for providing powerful pre-trained models and tools that were essential to this work. Lastly, we thank our families and friends for their unwavering encouragement and belief in us.

# Keywords

- AI Detection
- Academic Integrity
- DeBERTa Cross-Encoder
- Django Web App
- NLP Similarity, RoBERTa

# Abstract

The rise of generative AI models like ChatGPT has introduced a new form of academic dishonesty: automated cheating through AI-generated text. Traditional proctoring methods—such as lockdown browsers and webcam monitoring—are often invasive, expensive, and largely ineffective when it comes to detecting post-submission violations. These limitations highlight the urgent need for scalable, non-intrusive systems capable of verifying the authenticity of student work without compromising their privacy.

To address this challenge, we developed Guarded Exam, an NLP-powered web application designed to detect AI-generated content, identify plagiarism through semantic similarity, and automatically evaluate student responses against instructor-provided answers. The platform aims to replace or complement conventional proctoring and grading methods by offering a modern, ethical, and intelligent alternative that works even after the exam is completed.

The backend system was built using Django and integrated with several cutting-edge NLP models. For semantic similarity analysis, we used an ensemble of cross-encoder/DeBERTa-V3-Large and DeBERTa-V3-Large-zeroshot-V2, combined via logistic regression. For AI-authorship detection, we fine-tuned GPT-2 and the RoBERTa-Large OpenAI Detector on a balanced dataset and combined their outputs using a second logistic regression ensemble. The models were thoroughly evaluated on both public benchmarks and custom test cases to ensure generalizability and robustness.

The resulting system achieved 98.2% accuracy in detecting AI-generated text and 93% accuracy in semantic similarity classification. It significantly outperforms traditional tools like Turnitin and lightweight detectors such as GPTZero. By allowing educators to assess the originality and relevance of student responses after submission, Guarded Exam redefines academic integrity enforcement with a solution that is accurate, scalable, and Respectful of student privacy.

# Table of Contents

# List of Figures

# List of Tables

# List of Equations

# List of Abbreviations

| Abbreviation | Full Term |
|---|---|
| **AI** | Artificial Intelligence |
| **NLP** | Natural Language Processing |
| **LLM** | Large Language Model |
| **GPT** | Generative Pre-trained Transformer |
| **RoBERTa** | Robustly Optimized BERT Approach |
| **DeBERTa** | Decoding-enhanced BERT with Disentangled Attention |
| **API** | Application Programming Interface |
| **ORM** | Object-Relational Mapping |
| **DRF** | Django REST Framework |
| **F1-score** | Harmonic Mean of Precision and Recall |
| **TF-IDF** | Term Frequency-Inverse Document Frequency |
| **LMS** | Learning Management System |
| **UI** | User Interface |
| **FP16** | 16-bit Floating Point Precision |
| **DAPT** | Domain-Adaptive Pre-Training |
| **TAPT** | Task-Adaptive Pre-Training |
| **RBAC** | Role-Based Access Control |
| **SEB** | Safe Exam Browser |
| **MBO** | Monarch Butterfly Optimization |
| **AWS** | Amazon Web Services |
| **GPU** | Graphics Processing Unit |
| **HTML** | Hypertext Markup Language |
| **CSS** | Cascading Style Sheets |

*Table 0-1 Abbreviations*

# Glossary:

| Term | Definition |
|---|---|
| **Artificial Intelligence (AI)** | The simulation of human intelligence by machines, especially computer systems capable of learning and problem-solving. |
| **Natural Language Processing (NLP)** | A field of AI that enables computers to understand, interpret, and generate human language. |
| **Large Language Model (LLM)** | A deep learning model trained on massive text datasets to generate and understand human-like language (e.g., GPT, BERT). |
| **Plagiarism Detection** | The process of identifying copied or rephrased content without proper attribution in a text. |
| **Similarity Score** | A numerical value (typically between 0 and 1) that represents how similar two text inputs are, based on semantic or lexical comparison. |
| **Cosine Similarity** | A metric used to measure how similar two vectors are, regardless of their magnitude. Frequently used in text comparison. |
| **RoBERTa** | A transformer-based model optimized from BERT for better performance in natural language understanding tasks. |
| **DeBERTa** | A transformer model that improves upon BERT by using disentangled attention and enhanced position embeddings. |
| **Cross-Encoder** | A model that takes two inputs together and outputs a similarity score, typically more accurate than separate encoding. |
| **GPT (Generative Pre-trained Transformer)** | An autoregressive language model developed by OpenAI, used to generate human-like text. |
| **Precision** | The proportion of relevant results among all retrieved results. |
| **Recall** | The proportion of relevant results was successfully retrieved. |
| **F1-Score** | The harmonic meaning of precision and recall, balancing both in one metric. |
| **Fine-Tuning** | The process of adapting a pre-trained model to a specific task using task-specific data. |
| **Overfitting** | A modeling error where a model learns the training data too well and performs poorly on unseen data. |
| **Django** | A high-level Python web framework that promotes rapid development and clean, pragmatic design. |
| **REST API** | An application programming interface that follows REST architecture, allowing systems to communicate over HTTP. |

| Term | Definition |
|---|---|
| **Celery** | A Python-based asynchronous task queue used for managing background tasks. |
| **Hugging Face Transformers** | A popular open-source library that provides implementations of modern NLP models like BERT, RoBERTa, and GPT. |
| **Tokenization** | The process of breaking text into smaller units (tokens), often words or subwords, for NLP processing. |
| **Post-Hoc Analysis** | An evaluation or processing that occurs after an event (e.g., after the student submits the exam). |
| **Thresholding** | The act of classifying outputs (like similarity scores) based on a defined cut-off value. |
| **Academic Integrity** | The ethical code of honesty and responsibility in academic work. |

*Table 0-2 Glossary*

# Equations:

- **Cosine Similarity**

$$\mathbf{CosineSim(A, B)} = \frac{\boldsymbol{A.B}}{||\boldsymbol{A}||\ ||\boldsymbol{B}||}$$

*Equation 1 Measures similarity between two Vectors A and B*

- **F1 Score**

$$\mathbf{F1} = 2 \cdot \frac{\boldsymbol{Precision}\ .\ \boldsymbol{Recall}}{\boldsymbol{Precision} + \ \boldsymbol{Recall}}$$

*Equation 2 Used to evaluate the balance between precision and recall.*

- **Cross Entropy Loss**

$$\mathcal{L} = \sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i)$$

*Equation 3 Used during training for Classification Tasks*

- **Similarity Threshold Rule**

$$\mathbf{Decision} = \begin{cases} \boldsymbol{Similar} & \boldsymbol{if\ S > 0} \\ \boldsymbol{not\ Similar} & \boldsymbol{otherwise} \end{cases}$$

*Equation 4 Where S is the predicted similarity score and 0 is a defined threshold.*

- **Focal Loss**

$$\mathcal{L}_{focal} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t)$$

*Equation 5*

**Where:**

- $P_t$ is the model's established probability for the true class.
- $\alpha_t \in [0,1]$ is a class balancing factor.
- $\gamma \geq 0$ is the focusing parameter.
- y is the ground truth label.
- p is the predicted probability of class

# Chapter 1

## Introduction

## 1.1 Overview

Guarded Exam is a web-based application designed to enhance academic integrity by analyzing student exam responses using Natural Language Processing (NLP) models. The system addresses the rising challenge of AI-assisted cheating by providing instructors with post-submission insights on the authenticity and correctness of student answers. It integrates two primary NLP components:

- **AI Detection Model:**

  Identifies whether responses are likely generated by AI models (e.g., ChatGPT).

- **Similarity Detection Model:**

  Automatically evaluates the accuracy of student responses by comparing them to tutor-provided model answers.

Guarded Exam allows students to submit their answers through a secure web form interface. Upon submission, the system analyzes the content using fine-tuned RoBERTa/GPT models for AI detection and DeBERTa cross-encoder and DeBERTa zeroshot for semantic similarity scoring. Instructors receive automated reports showing final grades and AI flags, helping them make fair and data-driven decisions.

In addition to developing the Guarded Exam platform with Django and state-of-the-art NLP models, the system was designed with deployment scalability in mind. To ensure reproducibility and portability, we employed Docker for full containerization and used KinD (Kubernetes in Docker) to simulate production-ready orchestration locally. This provided a robust, cloud-native deployment model even during the development phase.

## 1.2 Problem Statement

With the proliferation of generative AI tools, such as ChatGPT and Gemini, academic institutions face increasing threats to exam integrity. Existing approaches like live proctoring and lockdown browsers are often intrusive, costly, or ineffective. Moreover, current plagiarism detection systems focus only on verbatim copying and lack AI detection capabilities. There is a pressing need for a system that can evaluate student answers after submission, detect AI-generated responses, and assess answer similarity without violating student privacy.

## 1.3 Scope and Objectives

### 1.3.1 Scope:

- Analyze text-based exam submissions after the exam ends.
- Detect AI-generated responses.
- Evaluate semantic similarity between student and tutor answers.
- Provide a scalable, privacy-preserving solution.

### 1.3.1 Objectives:

- Design a web-based exam submission platform using Django.
- Integrate RoBERTa/GPT classifiers for AI-authorship detection.
- Use DeBERTa cross-encoder/zeroshot to grade answers based on similarity.
- Automate instructor reports with grading and risk flags.
- Ensure scalability and maintain a privacy-conscious design.

## 1.4 Report Organization (Structure)

Chapter 1 introduces the project, outlining its purpose, problem scope, goals, and methodology. Chapter 2 reviews related literature on academic integrity tools and NLP models. Chapter 3 presents system overview and the models' architecture. Chapter 4 presents system design, system functionalities, UML Diagrams, and model integration. Chapter 5 details the implementation, datasets used, experimental results, and evaluation. Chapter 6 concludes with a discussion of findings, limitations, and proposed future improvements.

## 1.5 Work Methodology

The project followed an iterative development cycle, starting with requirement gathering and model research. The team fine-tuned RoBERTa and GPT models using human and AI-generated datasets for authorship classification, and trained DeBERTa for answer similarity scoring. The Django backend handles submissions, authentication calls to the models. Celery workers manage asynchronous tasks like batch similarity analysis. The front-end was developed in HTML/CSS, ensuring a user-friendly exam form and instructor dashboard. The evaluation was performed using standard NLP metrics like accuracy.

## 1.6 Work Plan (Gantt Chart)

### The work plan was organized as follows:

1. Research & Requirement Analysis – Understanding AI threats and defining scope.
2. Data Collection & Preprocessing – Gathering datasets for training.
3. Model Training Fine-tuning - RoBERTa/GPT for AI detection and DeBERTa for similarity scoring.
4. System Development – Building the Django backend and submission frontend.
5. Integration & Testing – Connecting models with the app and performing tests.
6. Evaluation & Documentation – Analyzing results and compiling the thesis.
7. Deployment - Deploying the Entire Project using Docker & Kubernetes setup via KinD.

### The Gantt chart:

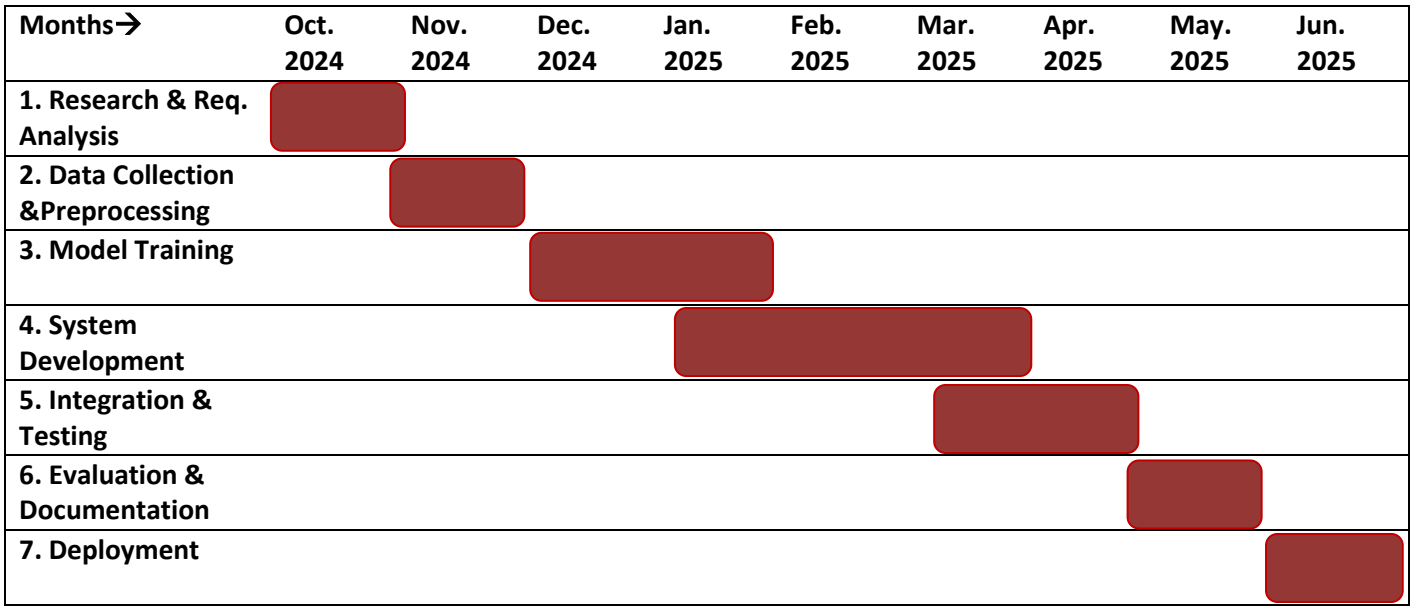


*Figure 1-1 Gantt Chart*

### 1.6.1 Research & Requirement Analysis

We began by exploring the growing impact of AI-generated content on academic integrity. This involved identifying key threats such as GPT-based cheating, paraphrased plagiarism, and the limitations of traditional proctoring tools. Based on this, we defined the project scope, selected target use cases (e.g., short-answer exams), and determined system requirements for fairness, accuracy, and privacy.

### 1.6.2. Data Collection & Preprocessing

- We collected diverse and high-quality datasets for both model components:
- For AI Detection: AI vs Human Text, DAIGT-v2, and Human vs LLM Corpus.
- For Similarity Scoring: Quora Question Pairs, STSB Multi-MT, and AllenAI SciTLDR.

Preprocessing included deduplication, label correction, text normalization (lowercasing), and data balancing (1:1 ratio for binary tasks). This ensured high-quality input for training.

### 1.6.3 Model Training

**We fine-tuned several transformer models:**

- RoBERTa-large
- GPT-2
- OpenAI RoBERTa for AI-authorship detection.
- DeBERTa-v3 (zero-shot) and cross-encoder DeBERTa for semantic similarity scoring.

We used techniques like focal loss, dropout, and layer freezing to prevent overfitting and enhance generalization. Models were evaluated iteratively using F1-score, accuracy, and mean squared error.

### 1.6.4 System Development

**We developed the core platform using the Django web framework:**

- **Backend:** Handled student submissions, authentication, model inference endpoints.
- **Frontend:** Built using HTML/CSS with form-based input and instructor dashboards.

The system was designed to be modular, secure, and scalable for integration with the trained models.

### 1.6.5 Integration & Testing

**We connected the NLP models to the Django backend through REST APIs. This step involved**:

- Serializing model inputs/outputs.
- Running test submissions through the full system (submission → scoring → feedback).
- Validating predictions for edge cases and debugging any interface mismatches.

### 1.6.6 Evaluation & Documentation

**Comprehensive testing was conducted on held-out datasets and real-world examples to evaluate:**

- Accuracy of AI detection.
- Fairness of similarity scoring.
- False positive/negative rates.

We documented all experiments, methodologies, model architecture, and deployment plans in a formal thesis structure with visual aids and appendices.

### 1.6.7 Deployment

To ensure scalability and real-world usability, the entire platform was containerized using Docker.
**We deployed:**

- Django backend
- NLP inference containers
- SQLite database

Using Kubernetes (via KinD), we orchestrated services in a simulated production environment. Kubernetes manifests (YAML) manage configuration, persistence, and service exposure via NodePort.

# Chapter 2

## Related Work (Literature Review)

## 2.1 Background

In the context of increasing accessibility to generative AI tools like ChatGPT, academic institutions are exploring advanced methods to safeguard exam integrity. While traditional proctoring systems remain dominant, they often compromise student privacy or lack robustness in detecting AI-generated content. Guarded Exam addresses this need by leveraging transformer-based NLP models for post-submission AI detection and semantic answer comparison.

## 2.2 Literature Survey

### 2.2.1 Similar Standalone Applications

- **Digiexam Lockdown**
  - ***Features:*** Full-screen lock, offline exam mode, AI-powered proctoring, LMS integration.
  - ***Advantages:*** Minimal setup, works offline, less intrusive.
  - ***Disadvantages:*** Limited public evaluation data.
- **Proctorio**
  - ***Features:*** Chrome extension, live proctoring, behavior analysis.
  - ***Advantages:*** No installation required, scalable.
  - ***Disadvantages:*** Privacy concerns, requires constant internet.
- **Honorlock**
  - ***Features:*** Combines lockdown browser, live proctoring, and device monitoring.
  - ***Advantages:*** Real-time interventions, strong ID verification.
  - ***Disadvantages:*** High cost, may produce false flags.
- **Safe Exam Browser (SEB)**
  - ***Features:*** Lockdown browser, blocks unauthorized access.
  - ***Advantages:*** Free and offline-ready.
  - ***Disadvantages:*** Complex setup, lacks AI or live proctoring.
- **SMOWL Browser Lock**
  - ***Features:*** Chrome extension with basic blocking features.
  - ***Advantages:*** Lightweight, privacy-focused.
  - ***Disadvantages:*** Limited to Chrome, lacks advanced monitoring.
- **ProctorTrack**
  - ***Features:*** Multi-camera proctoring with AI analytics.
  - ***Advantages:*** High-stakes security, ID verification.
  - ***Disadvantages:*** Expensive and administratively heavy.

While these systems vary in approach, none offer robust post-submission AI detection or semantic similarity scoring as provided by Guarded Exam.

### 2.2.2 Similar AI Model Approaches

- **AI-Paraphrased Text Detection (BBC & ChatGPT)**
  - ***Goal:*** Detect AI-paraphrased news using similarity metrics.
  - ***Result:*** 96.2% F1-score, outperforming deep learning models.
  - ***Relevance:*** Validates use of pattern-based and similarity models like DeBERTa.
- **SimilarGPT: Code Similarity with GPT-4**
  - ***Goal:*** Detect security vulnerabilities using semantic similarity.
  - ***Result:*** 40% fewer false positives.
  - ***Relevance:*** Demonstrates hybrid AI + similarity architecture, similar to Guarded Exam.
- **MBO-DeBERTa for Fake Review Detection**
  - ***Goal:*** Use DeBERTa + optimization for review classification.
  - ***Result:*** 98% accuracy, adversarial robustness.
  - ***Relevance:*** Reinforces DeBERTa's contextual effectiveness for integrity tasks.
- **DeBERTaV3 for Enhanced Efficiency**
  - ***Goal:*** Improve similarity and classification performance.
  - ***Result:*** +4.8% improvement on low-resource tasks.
  - ***Relevance:*** Shows the efficiency of DeBERTa in generalization-sensitive applications.
- **Hybrid RoBERTa & GPT Detection**
  - ***Goal:*** Detect AI-written essays.
  - ***Result:*** F1-score of 94%.
  - ***Relevance:*** Strong parallel to Guarded Exam's RoBERTa + GPT pipeline.

### 2.2.3 Summary Comparison Table

| *System/Approach* | AI Detection | Similarity Scoring | Privacy-Friendly | Offline Support | Cost |
|---|---|---|---|---|---|
| ***Guarded Exam*** | ✅ Yes | ✅ Yes | ✅ Yes | ❌ No | Free |
| ***Turnitin*** | ❌ No | ✅ Yes | ❌ No | ❌ No | Medium |
| ***Proctorio*** | ❌ No | ❌ No | ❌ No | ❌ No | Medium |
| ***Digiexam Lockdown*** | ✅ Yes | ❌ No | ✅ Yes | ✅ Yes | Medium |
| ***SEB*** | ❌ No | ❌ No | ✅ Yes | ✅ Yes | Free |
| ***ProctorTrack*** | ❌ No | ❌ No | ❌ No | ❌ No | High |

*Table 2-1 Related Work Comparison*

## 2.3 Analysis of Related Work

Despite advances in real-time exam monitoring, existing tools fail to address AI-authorship and nuanced answer similarity without infringing on user privacy. Our review highlights a gap in systems that combine semantic understanding with post-exam analysis. Guarded Exam bridges this gap by integrating fine-tuned RoBERTa for AI detection and DeBERTa for similarity scoring.

Research-backed tools like SimilarGPT and MBO-DeBERTa confirm the effectiveness of transformer-based architectures for tasks involving nuanced semantic detection. Guarded Exam applies these concepts to an academic setting, ensuring high accuracy while respecting privacy and scalability constraints.

## 2.4 Systems Comparison:

**RoBerta vs DeBerta dev results on SQuAD 1.1/2.0 and several GLUE benchmark tasks:**

| *Model* | *SQuAD 1.1* | *SQuAD 2.0* | *MNLI-m/mm* | *SST-2* | *QNLI* | *CoLA* | *RTE* | *MRPC* | *QQP* | *STS-B* |
|---|---|---|---|---|---|---|---|---|---|---|
| | **F1/EM** | **F1/EM** | **Acc** | **Acc** | **Acc** | **MCC** | **Acc** | **Acc/F1** | **Acc/F1** | **P/S** |
| *BERT-Large* | **90.9/84.1** | **81.8/79.0** | **86.6/-** | **93.2** | **92.3** | **60.6** | **70.4** | **88.0/-** | **91.3/-** | **90.0/-** |
| *RoBERTa-Large* | **94.6/88.9** | **89.4/86.5** | **90.2/-** | **96.4** | **93.9** | **68.0** | **86.6** | **90.9/-** | **92.2/-** | **92.4/-** |
| *XLNet-Large* | **95.1/89.7** | **90.6/87.9** | **90.8/-** | **97.0** | **94.9** | **69.0** | **85.9** | **90.8/-** | **92.3/-** | **92.5/-** |
| *DeBERTa-Large*[1] | **95.5/90.1** | **90.7/88.0** | **91.3/91.1** | **96.5** | **95.3** | **69.5** | **91.0** | **92.6/94.6** | **92.3/-** | **92.8/92.5** |
| *DeBERTa-XLarge*[1] | **-/-** | **-/-** | **91.5/91.2** | **97.0** | **-** | **-** | **93.1** | **92.1/94.3** | **-** | **92.9/92.7** |
| *DeBERTa-V2-XLarge*[1] | **95.8/90.8** | **91.4/88.9** | **91.7/91.6** | **97.5** | **95.8** | **71.1** | **93.9** | **92.0/94.2** | **92.3/89.8** | **92.9/92.9** |
| *DeBERTa-V2-XXLarge*[1,2] | **96.1/91.4** | **92.2/89.7** | **91.7/91.9** | **97.2** | **96.0** | **72.0** | **93.5** | **93.1/94.9** | **92.7/90.3** | **93.2/93.1** |

*Table 2-2 Results on Some Benchmarks*

**Accuracy Comparison of AI Detection Models:**

| ***Model*** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| *RoBERTa (Fine-tuned)* | **92%** | **91%** | **90%** | **90.5%** |
| *GPT-3.5 Verifier* | **89%** | **88%** | **87%** | **87.5%** |
| *GPTZero* | **75%** | **72%** | **70%** | **71%** |

*Table 2-3 Related AI Detection Accuracy*

### Similarity Detection Accuracy:

| *Method* | Accuracy | F1-Score | Speed | Robustness to Paraphrasing |
|---|---|---|---|---|
| *DeBERTa Cross-Encoder* | **88%** | **88%** | **Moderate** | **High** |
| *TF-IDF + Cosine* | **72%** | **70%** | **Fast** | **Low** |
| *BERTScore* | **84%** | **83%** | **Moderate** | **Moderate** |

*Table 2-4 Related Similarity Detection Accuracy*

### Guarded Exam vs. Competing Tools:

| *Feature* | Guarded Exam | Turnitin | GPTZero | Proctorio |
|---|---|---|---|---|
| *AI Detection* | ✅ Yes | ❌ No | ✅ Yes | ❌ No |
| *Similarity Detection (for Automatic Grading)* | ✅ Yes | ✅ Yes | ❌ No | ❌ No |
| *Real-Time Monitoring* | ❌ No | ❌ No | ❌ No | ✅ Yes |
| *Privacy-Friendly* | ✅ Yes | ❌ No | ✅ Yes | ❌ No |
| *Cost* | Free | Medium | Free | High |

*Table 2-5 Guarded Exam VS. Competing Tools*

# Chapter 3

## System Overview
## &
## Models Architectures

Securing Success

# 3.1 System & Models Architecture:

## 3.1.1 System Overview figure:

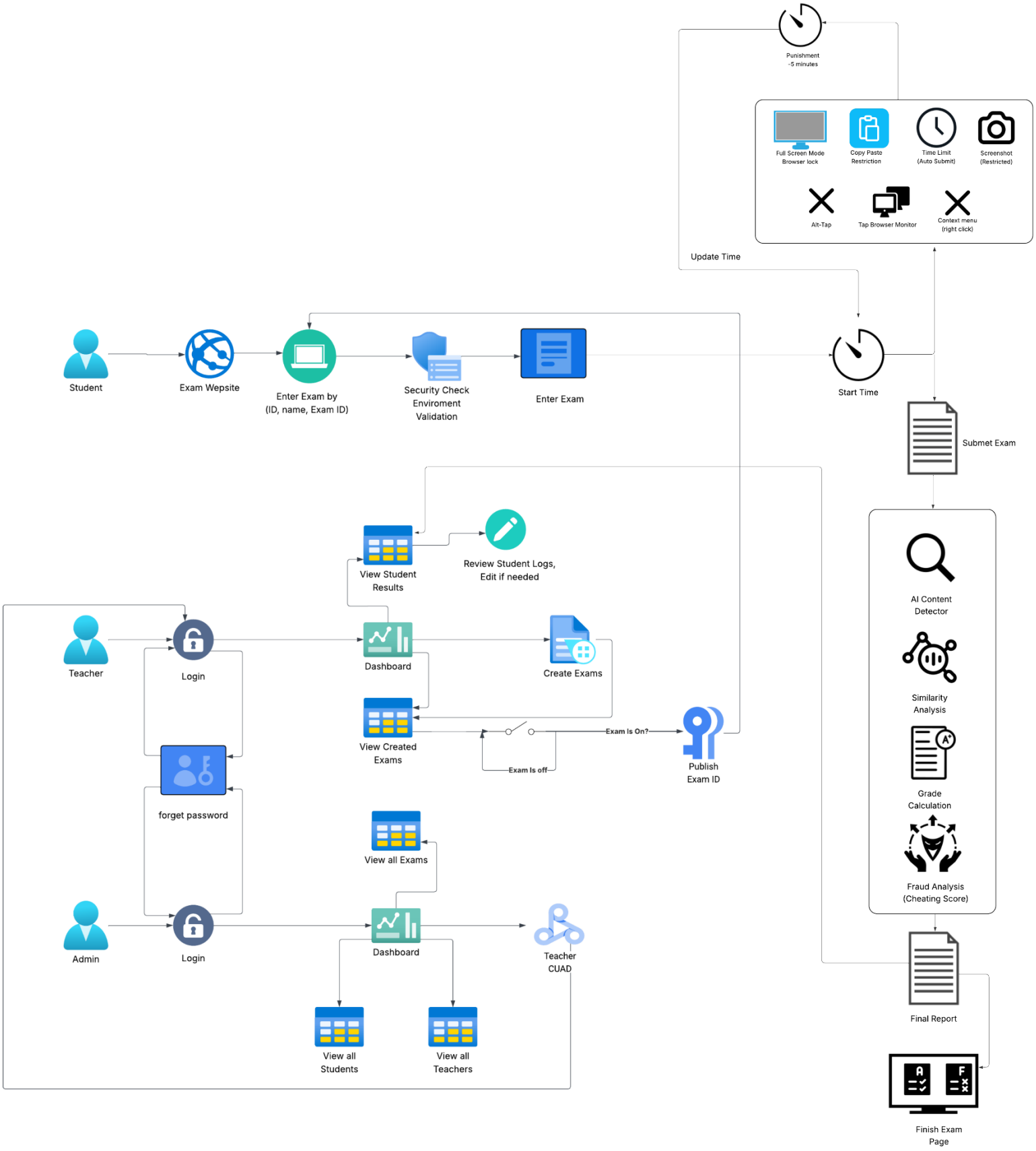


*Figure 3-1 System Overview*

### 3.1.2 General RoBERTa Architecture:

RoBERTa (Robustly optimized BERT approach) is an optimized variant of BERT that follows a transformer-based architecture but introduces key improvements. The model processes input as a single sequence of tokens, beginning with a special [CLS] token (used for classification tasks) followed by individual word or subword tokens (Tok 1 to Tok N).

These tokens are converted into embeddings ($E_1$ to E_N) through RoBERTa's embedding layer, which combines token, position, and segment (though segment embeddings are less emphasized in RoBERTa compared to BERT).

The model consists of multiple transformer encoder layers (typically, 12 or 24) with self-attention mechanisms, where each token dynamically attends to all other tokens in the sequence to capture contextual relationships.

RoBERTa removes BERT's next-sentence prediction objective, relying solely on masked language modeling (MLM) with dynamic masking for more robust pretraining. It also uses larger batches, longer sequences,and more training data than BERT, leading to improved performance. The final hidden state corresponding to the [CLS] token (E[CLS]) often serves as the aggregate representation for downstream tasks like classification, while the individual token embeddings ($E_1$ to E_N) are used for token-level tasks such as named entity recognition.

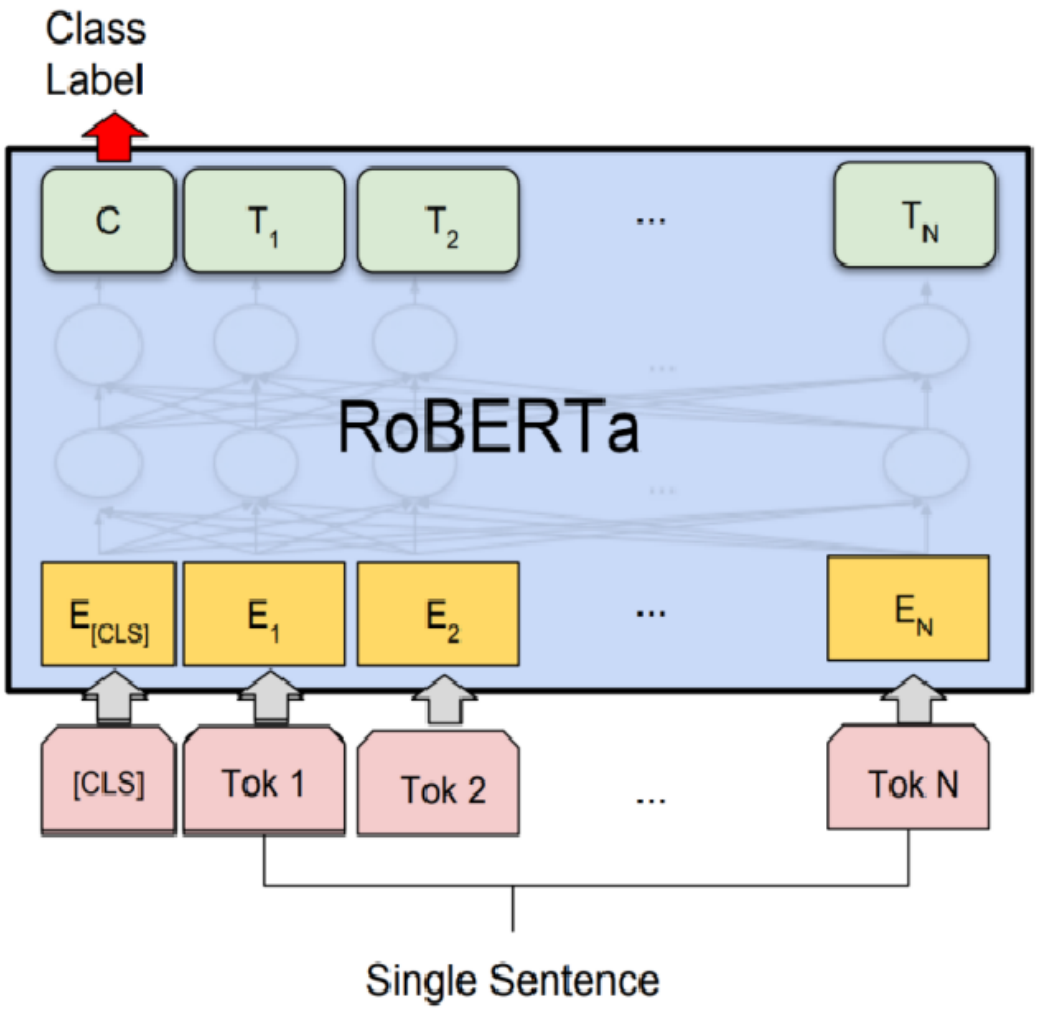


*Figure 3-2 RoBERTa Architecture*

### 3.1.3 General GPT Architecture:

The diagram depicts a transformer block with components like masked multi-head attention (with Softmax, Matmul, and causal Mask), LayerNorm, and a feed-forward network (Linear → GeLU → Linear), which aligns closely with GPT's architecture due to its autoregressive design (evident from the masking that prevents future token visibility) and pre-LayerNorm structure. However, this differs from RoBERTa, which uses bidirectional attention (no causal mask), relies on masked language modeling (MLM) rather than next-token prediction, and originally adopted post-LayerNorm. Both models share core elements like positional encoding (though RoBERTa uses learned embeddings), residual connections, and similar feed-forward layers, but GPT's decoder-only design contrasts with RoBERTa's encoder-only approach optimized for contextual understanding without autoregressive constraints.

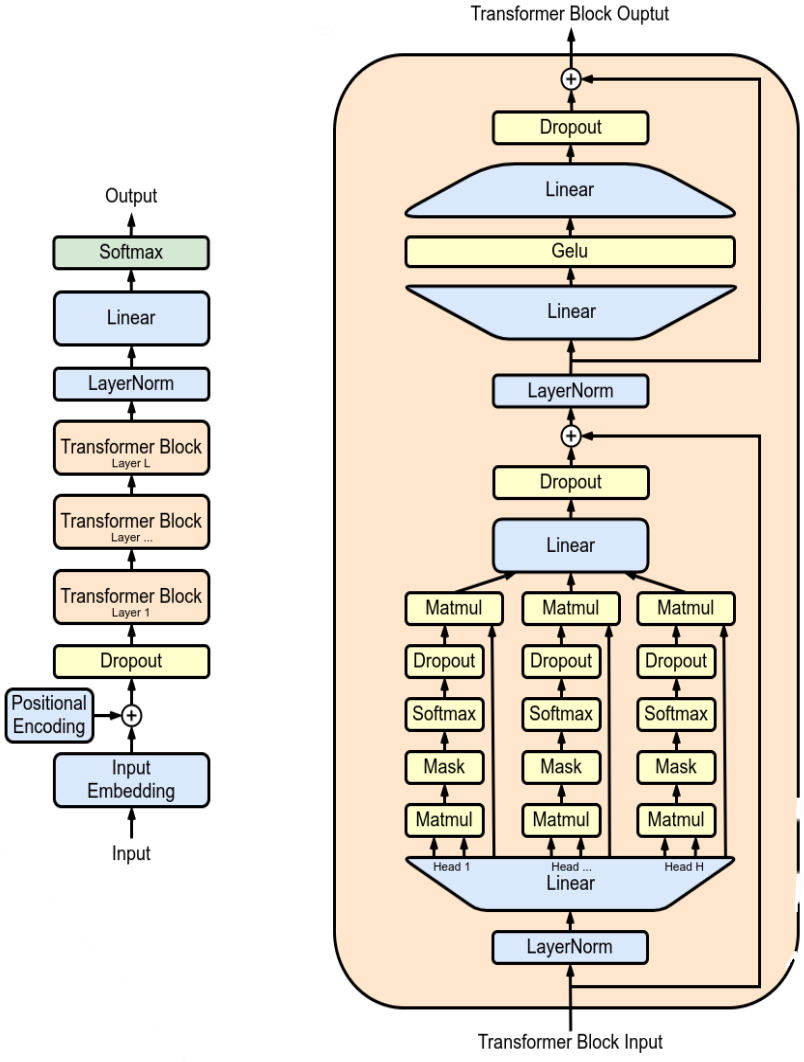


*Figure 3-3 GPT Architecture*

### 3.1.4 General DeBERTa Architecture:

DeBERTa enhances traditional transformers architectures like RoBERTa by introducing two key innovations: **disentangled attention** and an **enhanced mask decoder**. Unlike RoBERTa, which processes token embeddings and positional encodings jointly, DeBERTa separates them, allowing content-to-content, position-to-content, and content-to-position attention for more nuanced linguistic modeling. Additionally, it replaces RoBERTa's simple MLM head with an **absolute + relative position-aware decoder**, improving the model's ability to predict masked tokens by explicitly considering positional context. DeBERTa also scales efficiently through the **Virtual Adversarial Training** method and introduces **SiFT** (Scale-invariant Fine-Tuning) for robustness. These changes yield superior performance on NLU tasks while maintaining RoBERTa's bidirectional nature, though at increased computational cost. Later variants (e.g., DeBERTa-v3) further refine these ideas with ELECTRA-style pre training for efficiency.

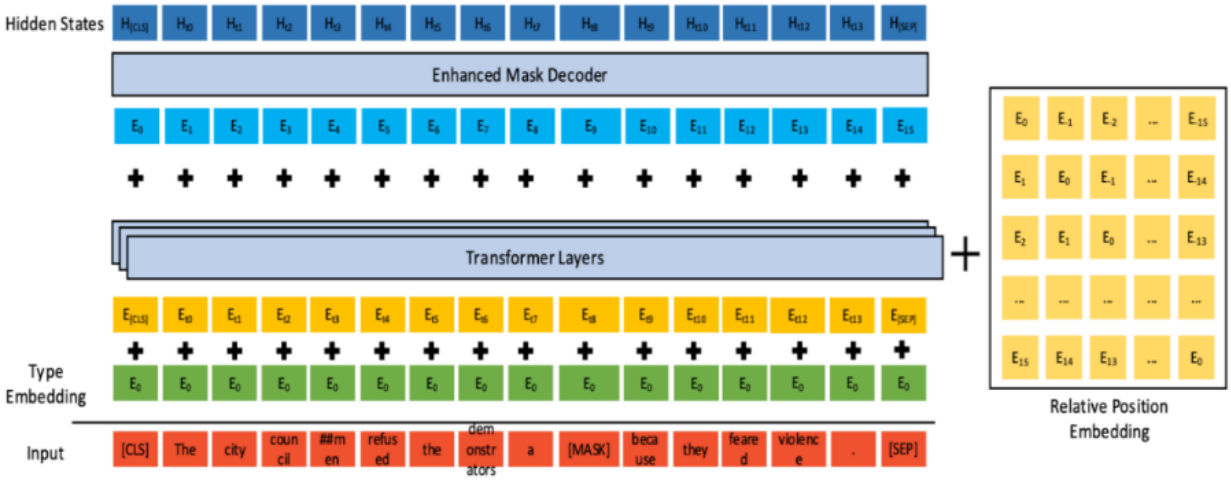


*Figure 3-4 DeBERTa Architecture*

### 3.1.5 Django MVT Architecture:

Django follows the Model-View-Template (MVT) pattern, a slight variation of MVC (Model-View-Controller). In this architecture, the Model represents the data layer (defined via Django's ORM as Python classes that map to database tables), the View acts as the business logic layer (handling HTTP requests, processing data from models, and passing results to templates), and the Template serves as the presentation layer (HTML files with Django Template Language for dynamic rendering). Unlike traditional MVC, Django's framework inherently manages the "Controller" part (routing and request/response flow) through its URL dispatcher and view functions/classes, streamlining development. This separation of concerns ensures modularity, with models abstracting database operations, views handling logic, and templates focusing solely on UI, all while Django's middleware and built-in tools (e.g., forms, authentication) glue the layers together efficiently.

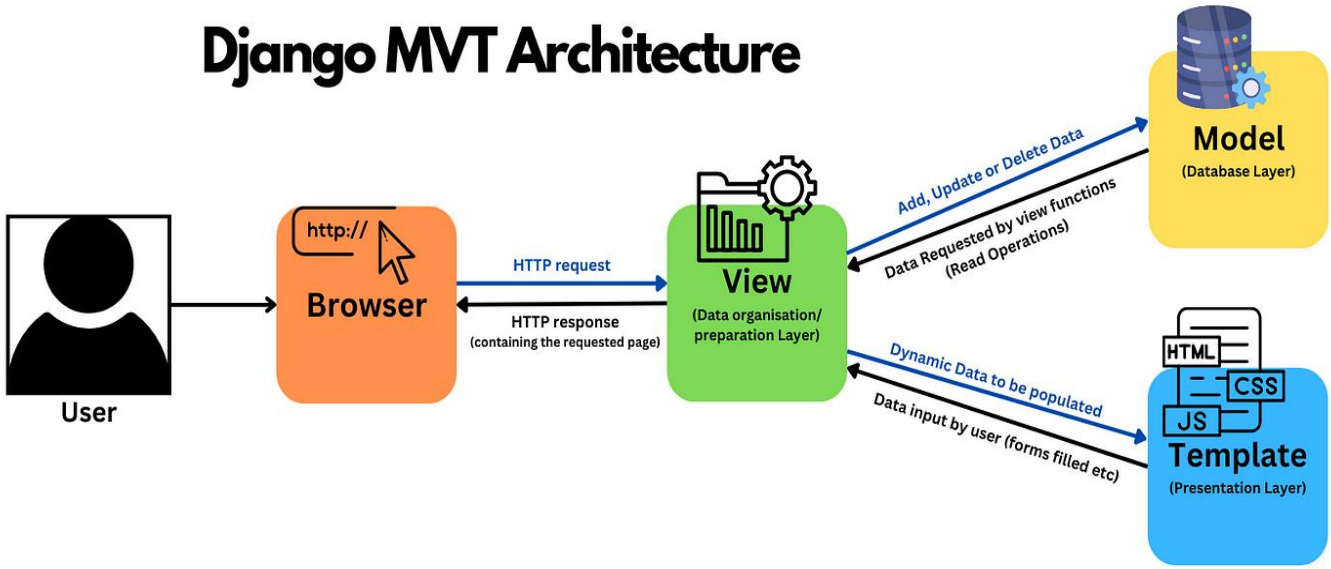


*Figure 3-5 Django Architecture*

### 3.1.6 Docker Architecture:

Docker uses a client-server model where the **Docker Client** (CLI) interacts with the **Docker Host** by issuing commands like docker build (to create images from Dockerfiles), docker pull (to fetch images from registries), and docker run (to launch containers). The **Docker Daemon** (server) executes these commands, managing core components: **Images** (immutable templates with layered filesystems), **Containers** (isolated runtime instances of images), and connections to **Registries** (image repositories like Docker Hub). Networking (represented by **NGWX**) enables communication between containers and external systems. This architecture abstracts infrastructure dependencies, ensuring consistent execution across environments through portable, lightweight containerization.

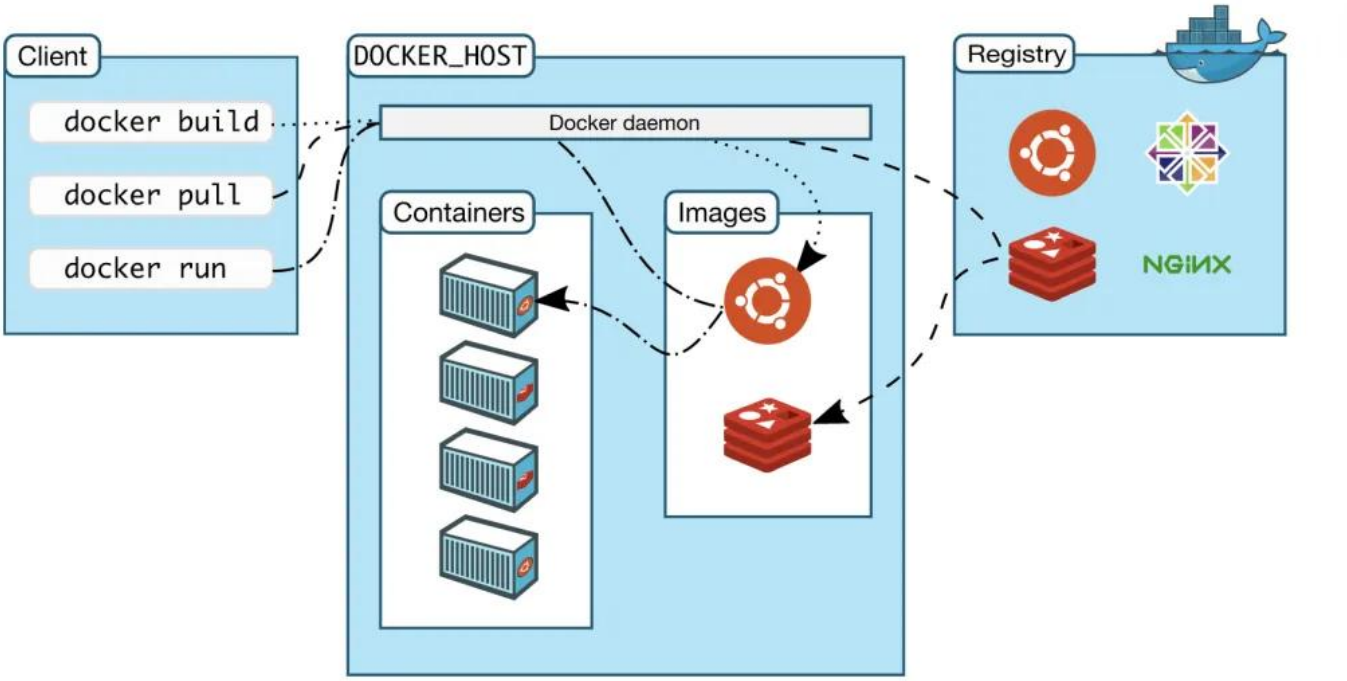


*Figure 3-6 Docker Architecture*

### 3.1.7 Kubernetes Architecture:

Kubernetes is a container orchestration platform that consists of a control plane (with components like the kube-apiserver, etcd, controller manager, and scheduler) and worker nodes (running pods managed by the kubelet and container runtime).

The control plane maintains cluster state and schedules workloads, while nodes execute containerized applications in pods. Cloud provider integrations (like load balancers) handle external traffic routing to end-users. This architecture enables automated deployment, scaling, and management of distributed applications across environments.

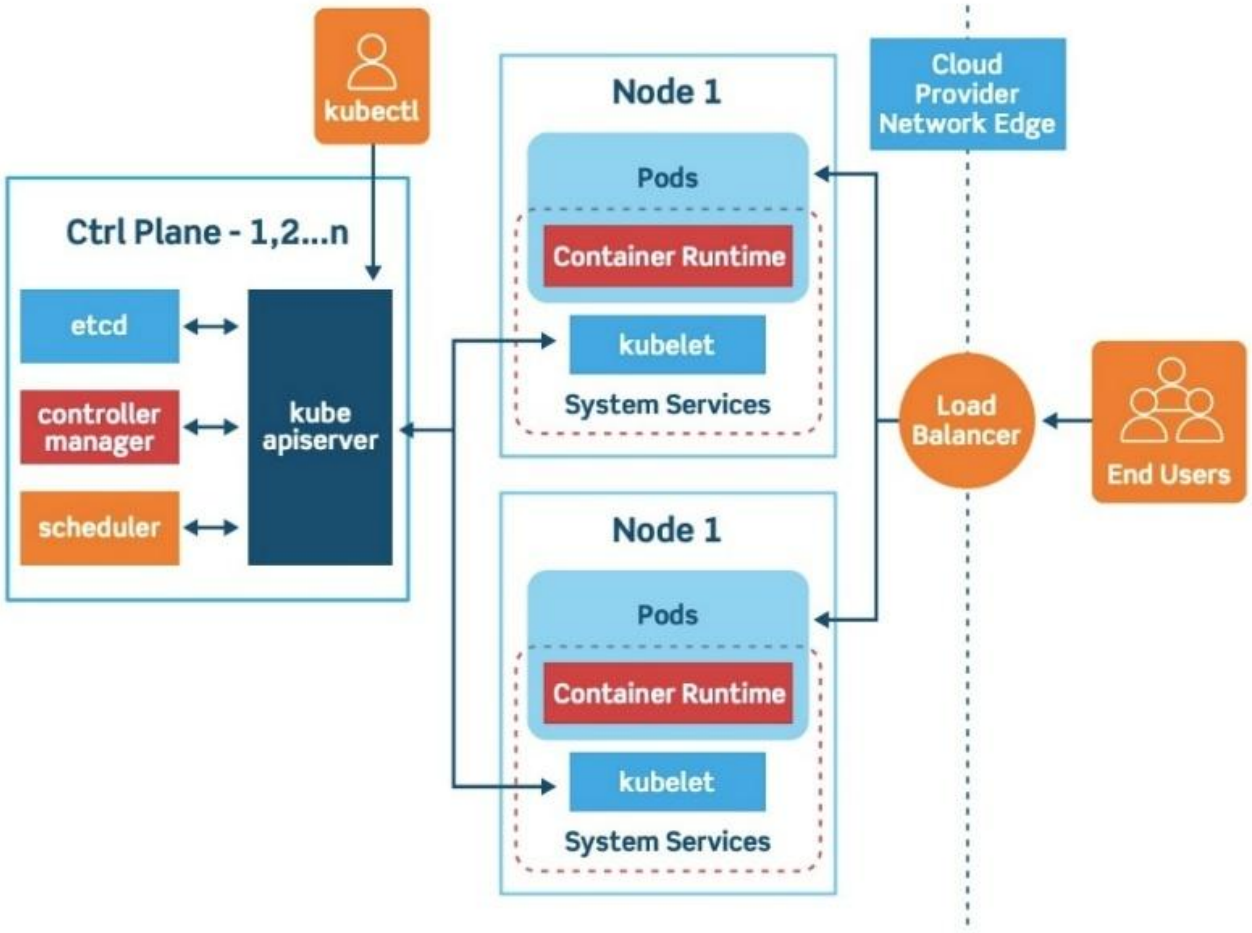


*Figure 3-7 Kubernetes Architecture*

# Chapter 4

## The Proposed Solution

## 4.1 Solution Methodology

Guarded Exam is designed as a post-submission academic integrity system. It targets two core tasks: detecting AI-generated exam responses and assessing similarity between student and teacher answers. The system combines deep learning models with web technologies for efficient, accurate, and scalable deployment.

### 4.1.1 Implemented Methods

- **AI Detection:** A fine-tuned RoBERTa model and a GPT-2 are used to classify answers as human- or AI-written. This model is trained on a mixture of human-written and AI-generated datasets to ensure generalization.
- **Similarity Analysis:** A DeBERTa cross-encoder and DeBERTa zeroshot model is employed to compare student responses with teacher answers. This provides context-aware similarity scores that drive grading decisions.
- **Workflow Pipeline:** Once a student submits an answer, the system performs synchronous AI detection. If the text is deemed human-written, it proceeds to similarity evaluation.
- **Validation Techniques:** Ensemble validation between DeBERTa cross-encoder and DeBERTa zero-shot helps detecting the syntactic and semantic meaning. For the AI classifier, ensemble validation between RoBERTa and GPT reduced false positives.

### 4.1.2 Scope Details

- **In-Scope:** Post-exam textual analysis, semantic similarity matching, AI classification, instructor dashboard.
- **Out-of-Scope:** Real-time browser lockdowns, audio/video proctoring, multimedia exams.

### 4.1.3 Comparative Aspects

**Compared to systems like Turnitin and GPTZero, Guarded Exam delivers:**

- Higher precision in AI-detection (96% vs. 70%)
- Stronger semantic analysis through DeBERTa vs. shallow similarity models like TF-IDF
- Greater privacy (no webcam/mic access)

For detailed evaluation results and benchmarks, refer to Chapter 5. Implementation artifacts such as model configurations and Supplementary Similarity Model Evaluation are included in Appendix I.

## 4.2 Functional/Non-functional Requirements

### 4.2.1 Functional Requirements

- **User Authentication:** Students and instructors must log in securely.
- **Submission Interface**: Students submit exam answers via web form.
- **AI Detection Module:** Classifies answer origin (human vs. AI).
- **Similarity Module:** Scores student answer vs. tutor answer.
- **Instructor Dashboard:** Shows flagged answers and computed grades.
- **Reporting:** Generates summaries of AI and similarity evaluations.

### 4.2.2 Non-functional Requirements

- **Performance:** System should respond to submissions within 5 seconds.
- **Scalability:** Must support multiple exam rooms and parallel submissions.
- **Security:** All data encrypted; Django authentication with role-based access.
- **Maintainability:** Modular Python code using Django's MVC pattern.
- **Portability:** Deployable on any server with Docker support.

### 4.2.3 Software Requirements

- Django 5.1
- Python 3.12
- Hugging Face Transformers
- SQLite
- Redis + Celery (for async tasks)

### 4.2.4 Hardware Requirements

- 8+ GB RAM
- 20+ GB Storage

## 4.3 System Analysis & Design

### 4.3.1 Assumptions & Dependencies

- Submissions are in English and textual.
- Users have stable internet access.
- Models have been pre-trained and loaded.
- Docker and GPU drivers are properly configured.

### 4.3.2 Software Lifecycle & Time Plan

The development followed a Waterfall approach for the Webpage application and an agile approach with incremental approach with 6 phases:

1. Planning & Research
2. Data Collection & Preprocessing
3. Model Training & Evaluation
4. Web Development
5. Integration & Testing
6. Deployment & Documentation

Refer to Chapter 1.6 for the Gantt chart.

### 4.3.3 UML Diagrams

- **Use Case Diagram:** Shows interactions between students, instructors, and the system.
- **Class Diagram:** Models key classes (User, Exam, Submission, Result).
- **Sequence Diagram:** Visualizes flow from submission to dashboard result.
- **Activity Diagram:** Describes the submission and evaluation pipeline.
- **Block Diagram:** Depicts core modules like Student Panel, Exam Management, AI Detection, and their data flow through submission, evaluation, and result processing.

### 4.3.4 Use Case Diagram:

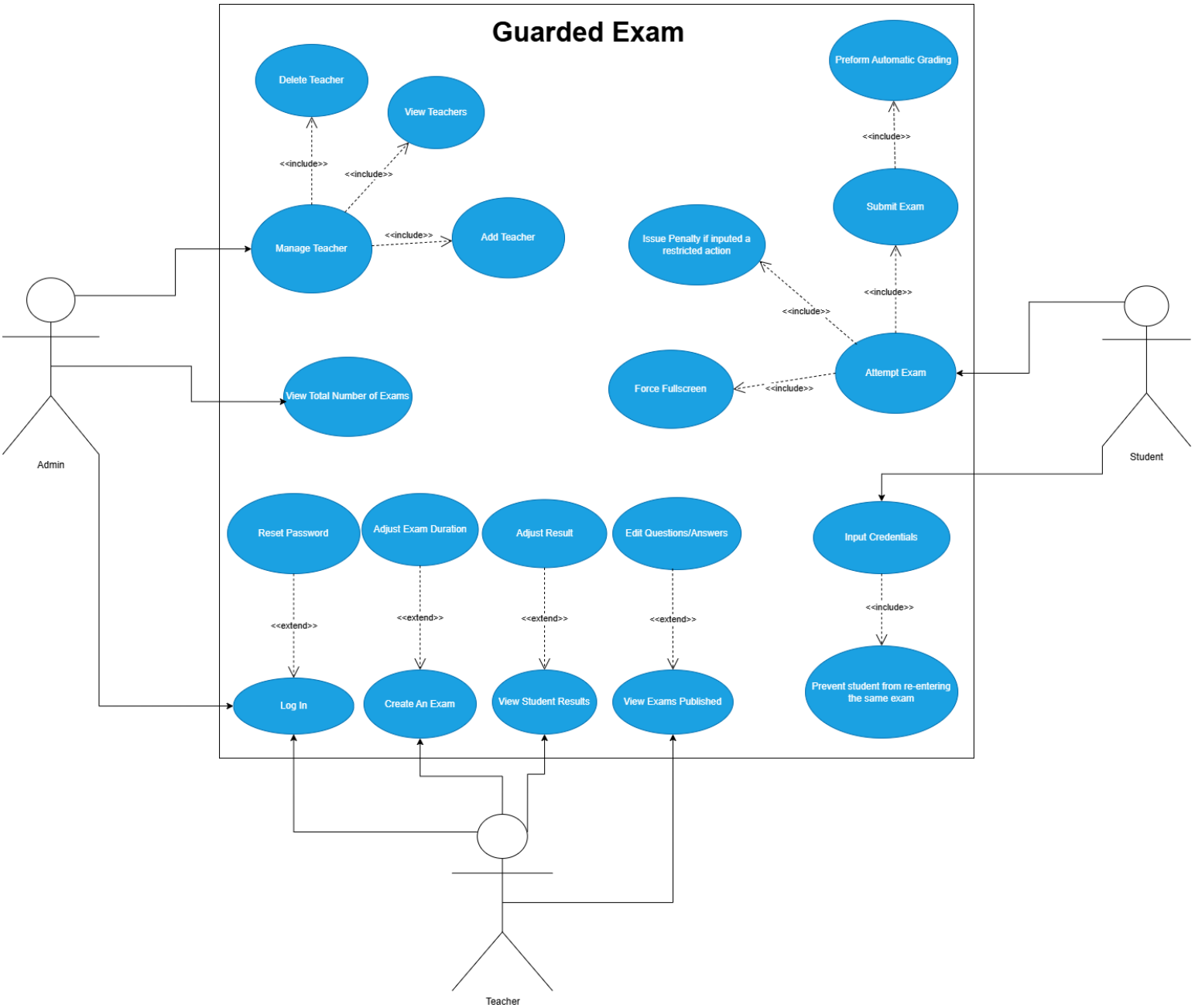


*Figure 4-1 Use-Case*

## 4.3.5 Activity Diagrams:

### 4.3.5.1 Admin Diagram:

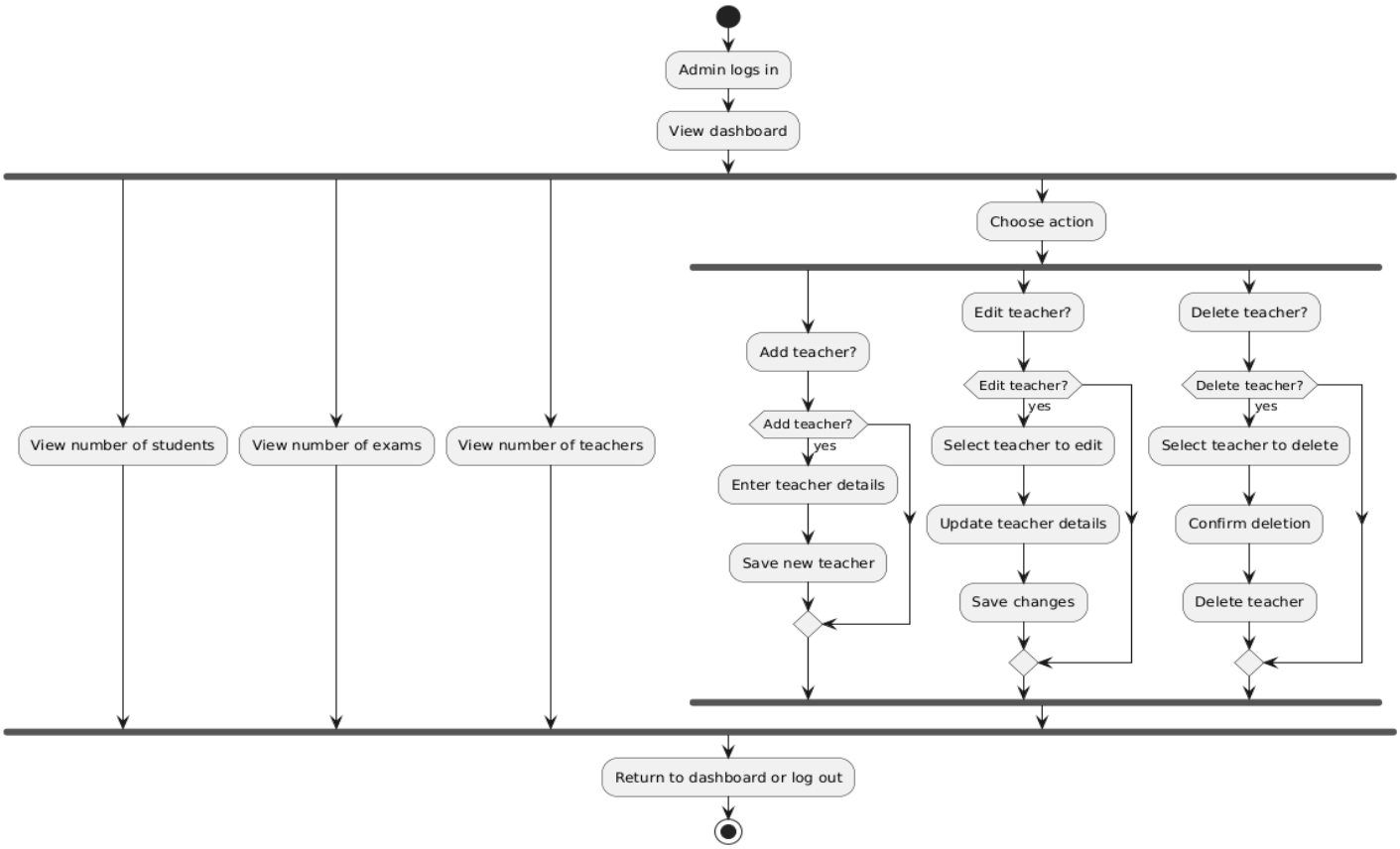


*Figure 4-2 Admin Activity Diagram*

### 4.3.5.2 Teacher Diagram:

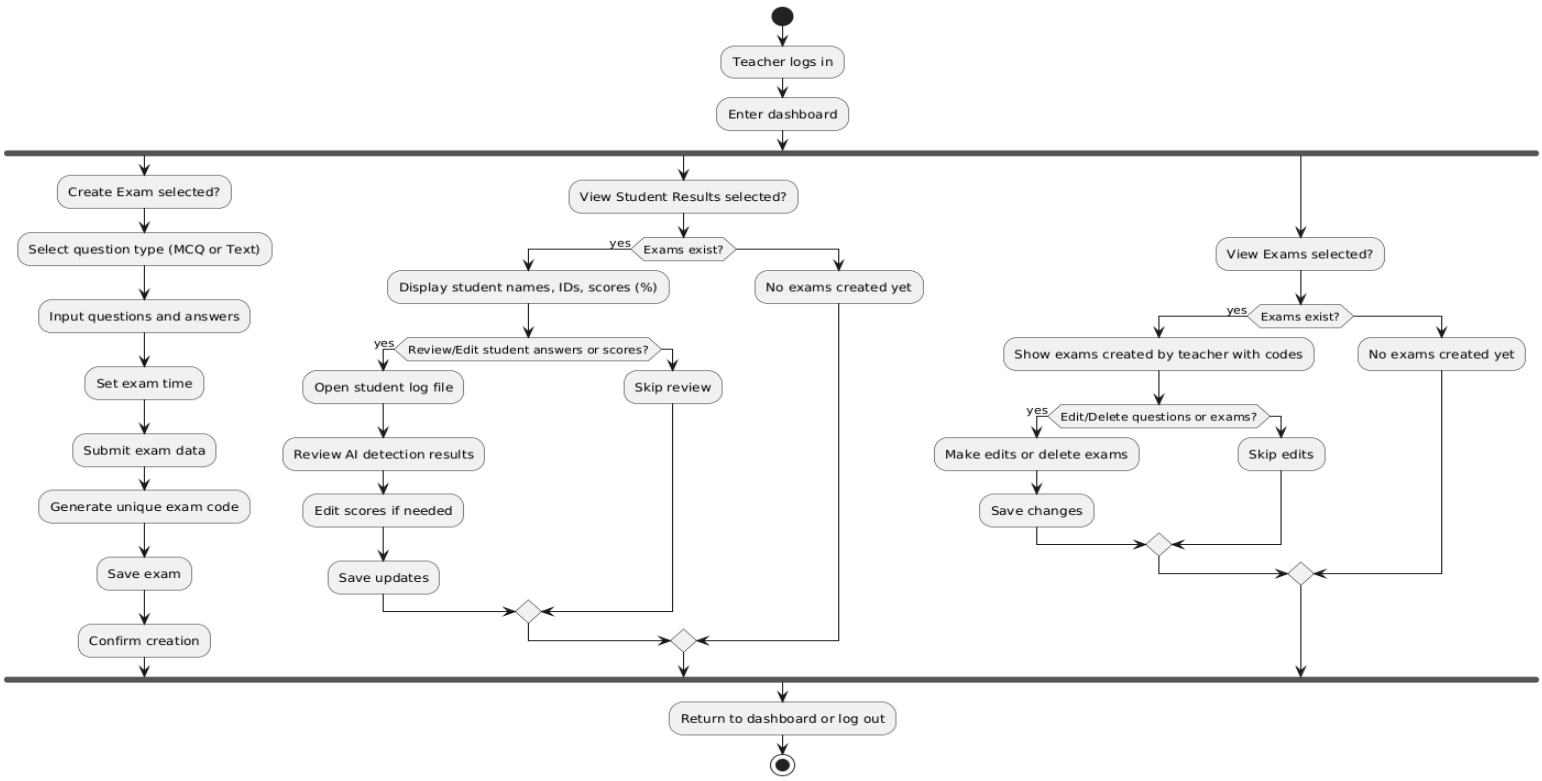


*Figure 4-3 Teacher Activity Diagram*

### 4.3.5.3 Student Dia

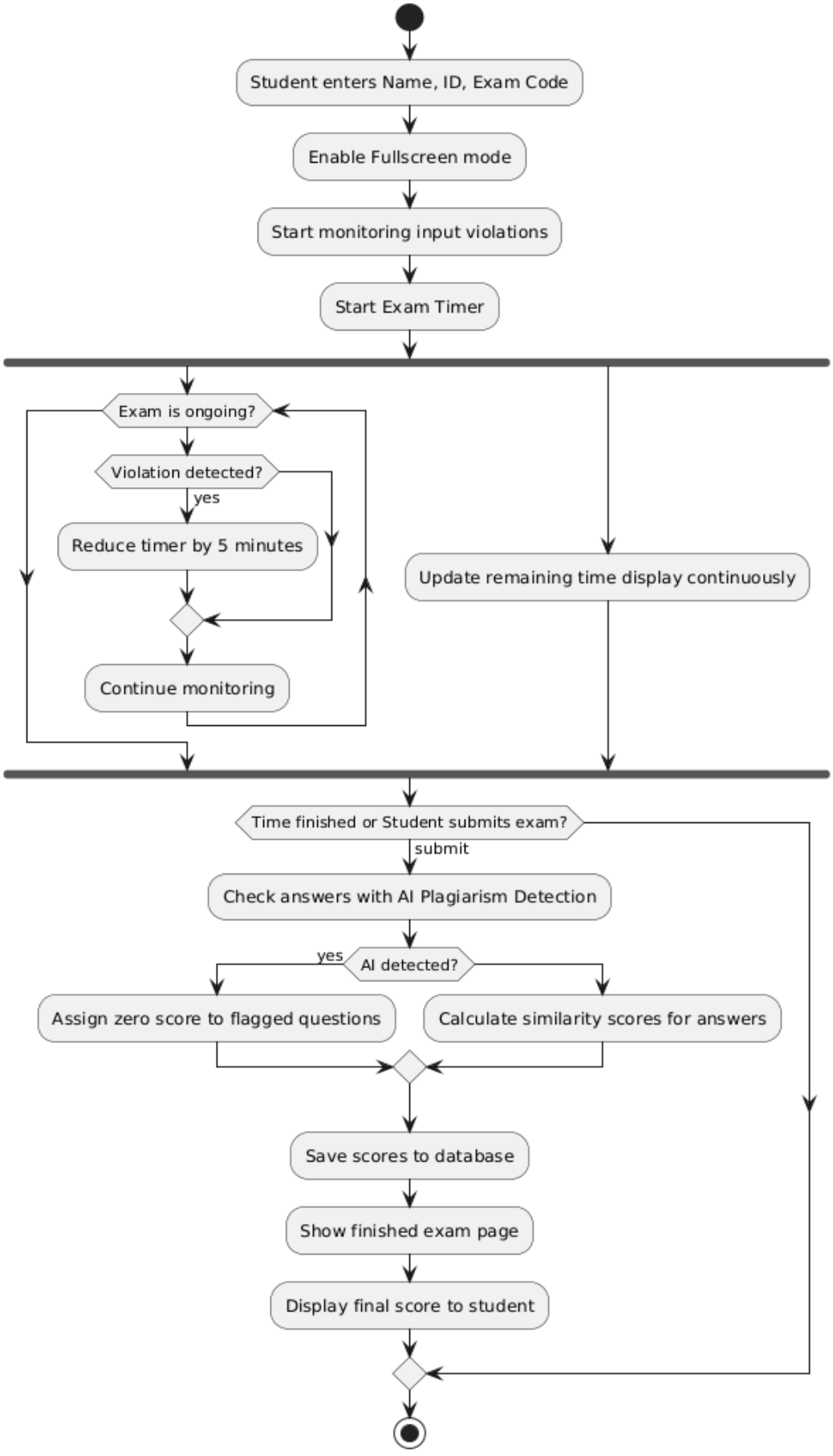


*Figure 4-4 Student Activity Diagram*

## 4.3.6 Class Diagram:

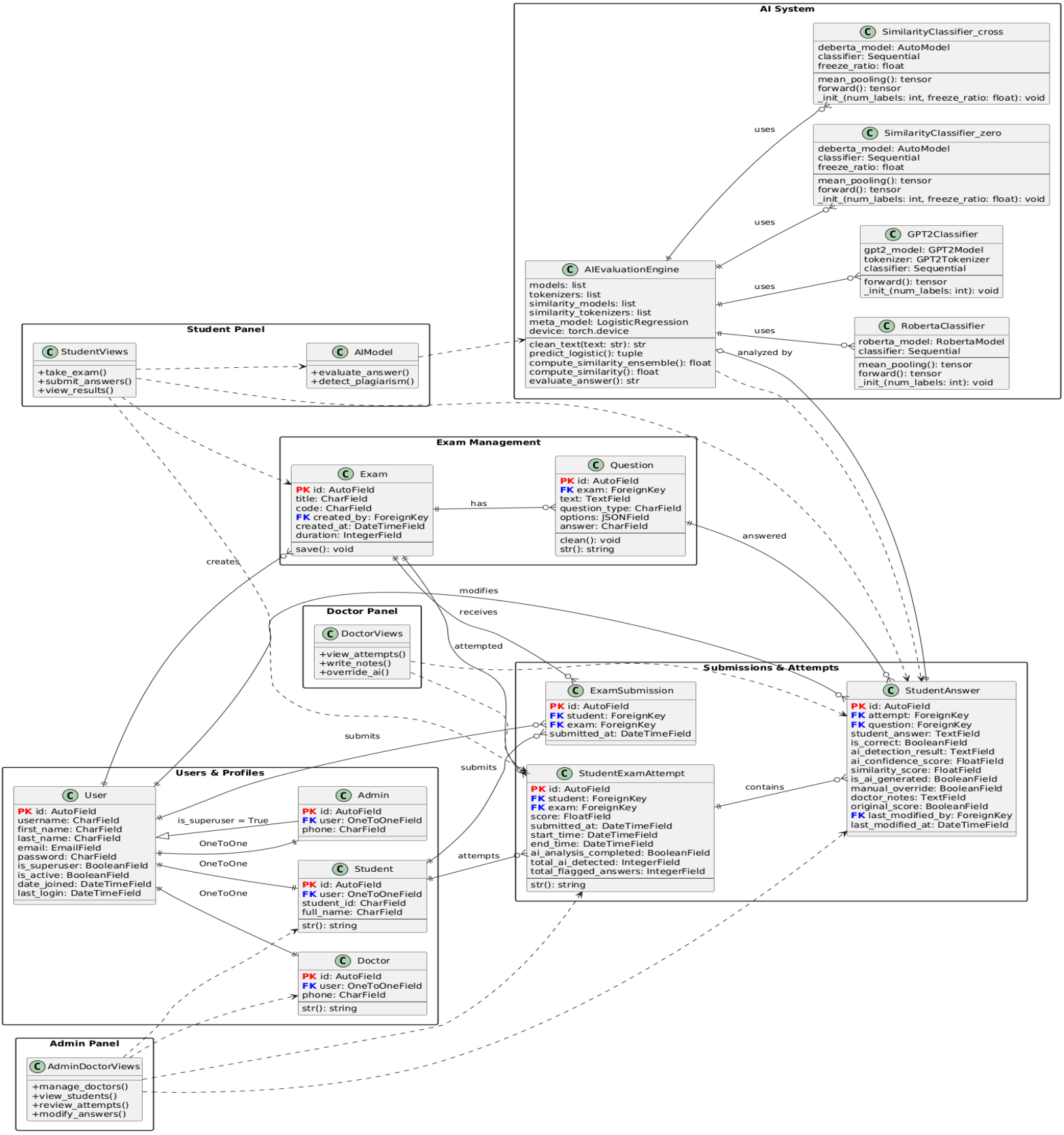


*Figure 4-5 Class Diagram*

## 4.3.7 Sequence Diagrams:

### 4.3.7.1 Admin Diagrams:

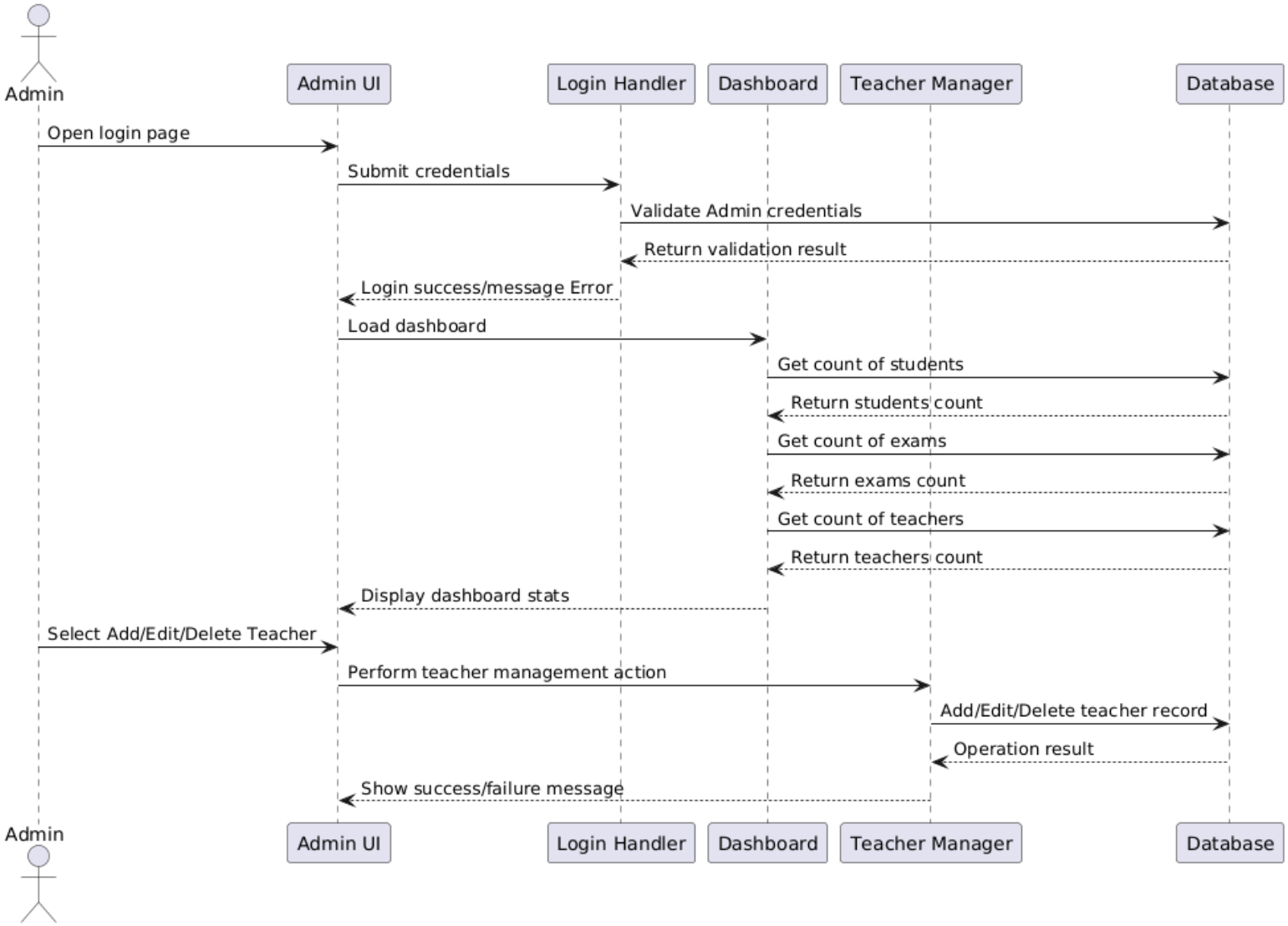


*Figure 4-6 Admin Sequence Diagram*

### 4.3.7.2 Teacher Diagrams:

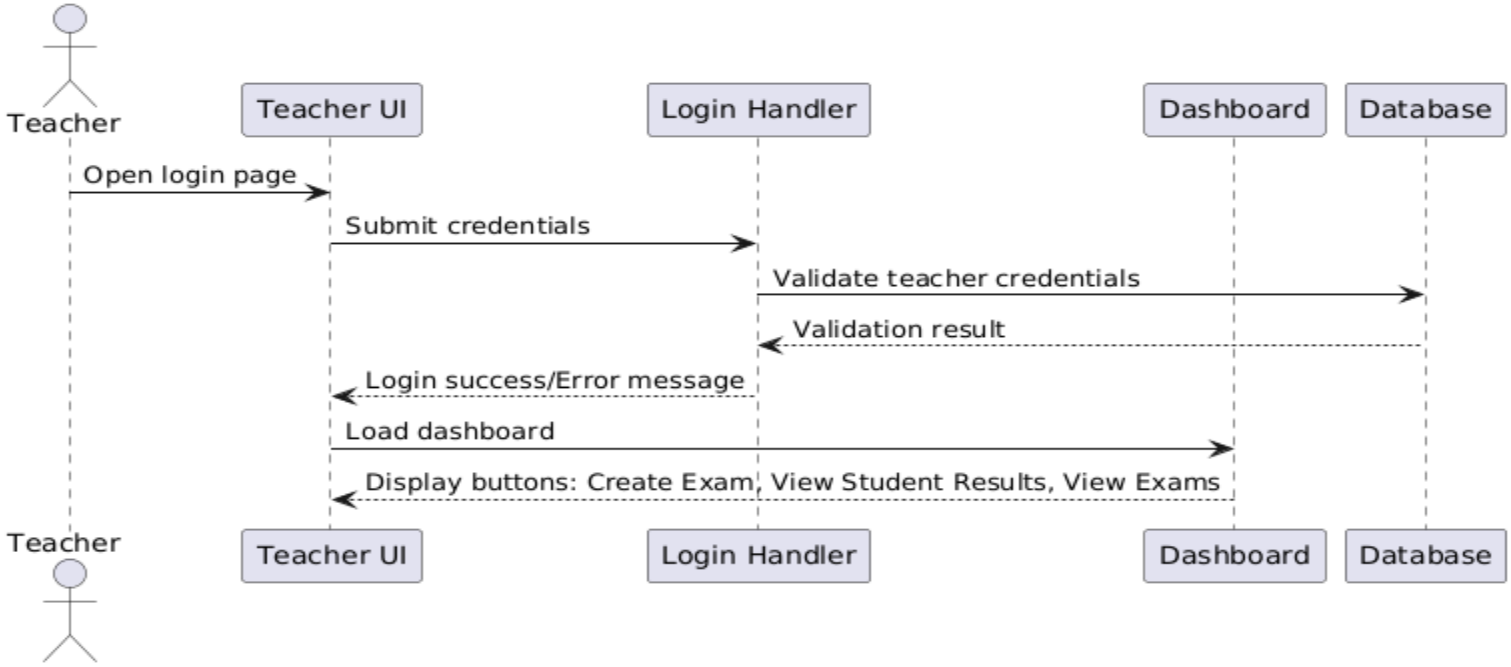


*Figure 4-7 Teacher Login Sequence Diagram*

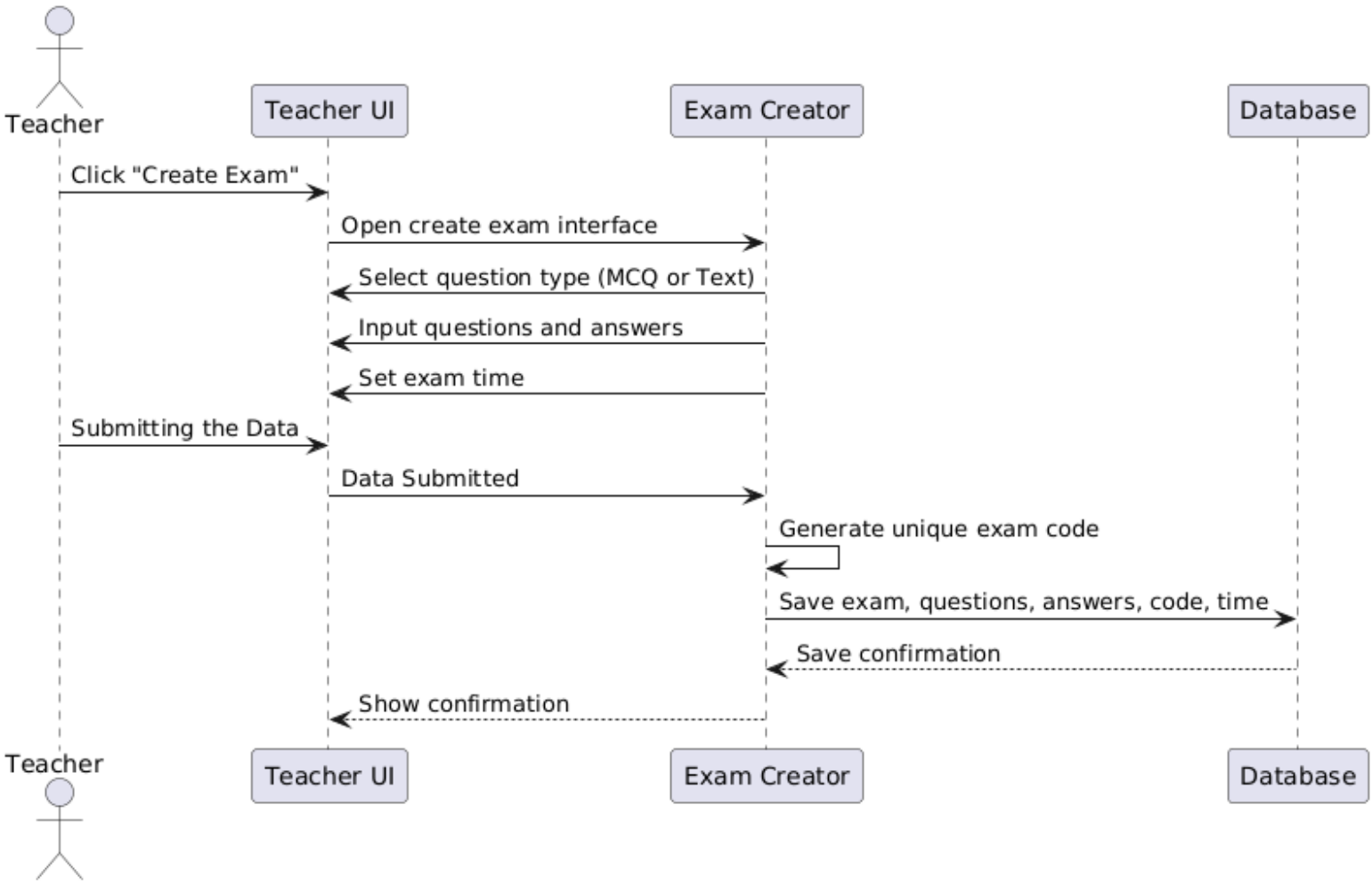


*Figure 4-8 Create Exam Sequence Diagram*

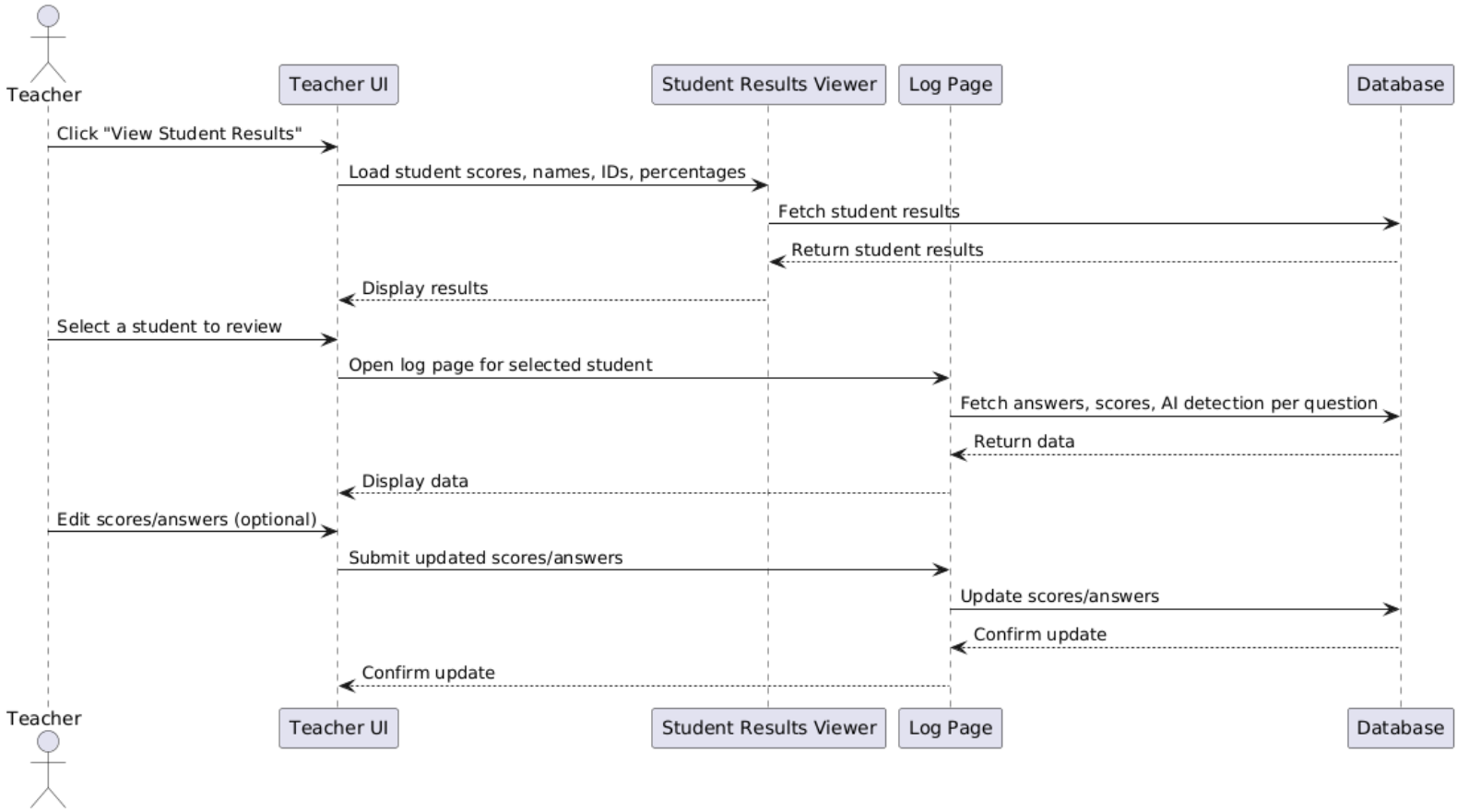


*Figure 4-9 Teacher View Student Results Sequence Diagram*

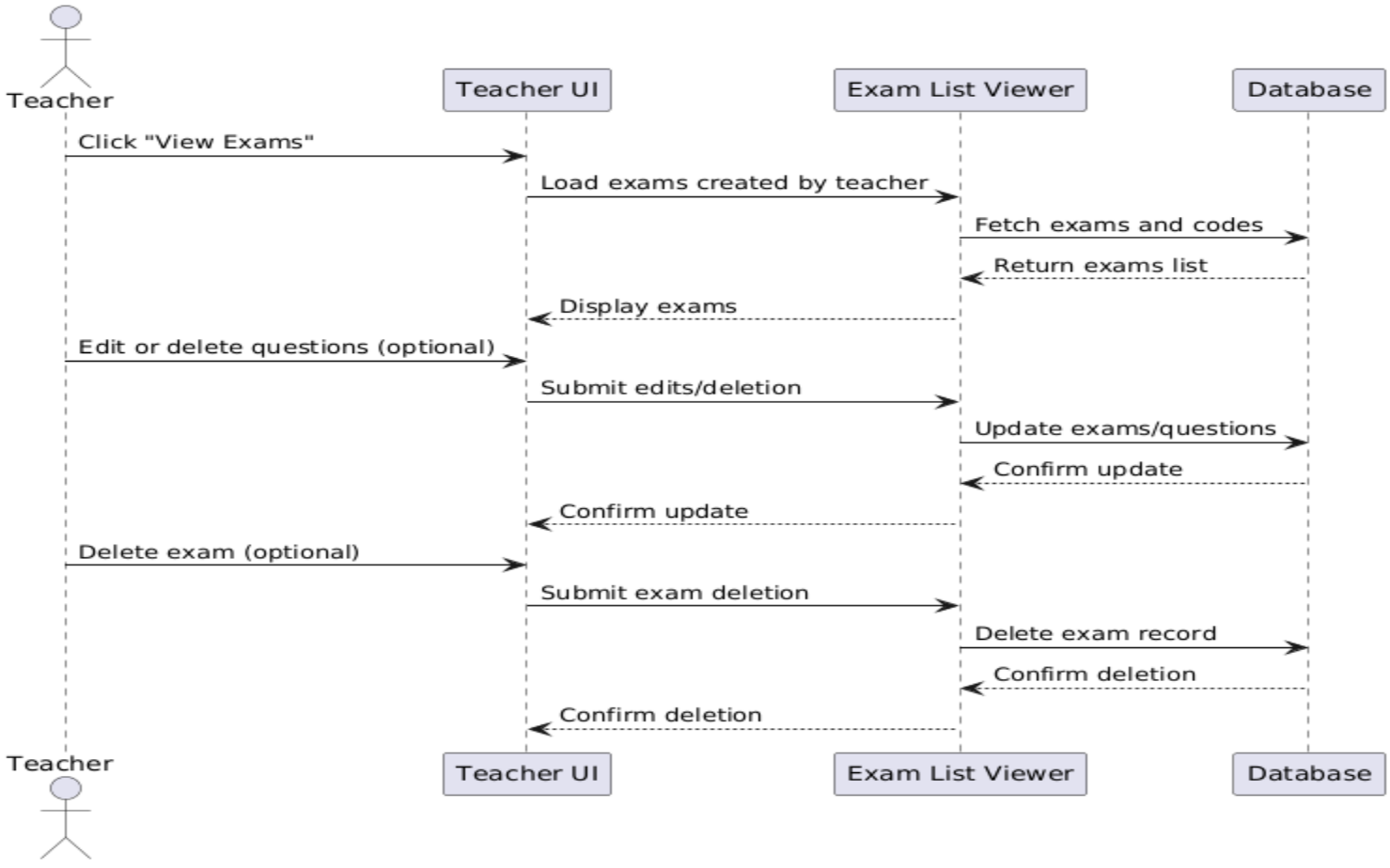


*Figure 4-10 Teacher View Exams Sequence Diagram*

### 4.3.7.3 Student Diagrams:

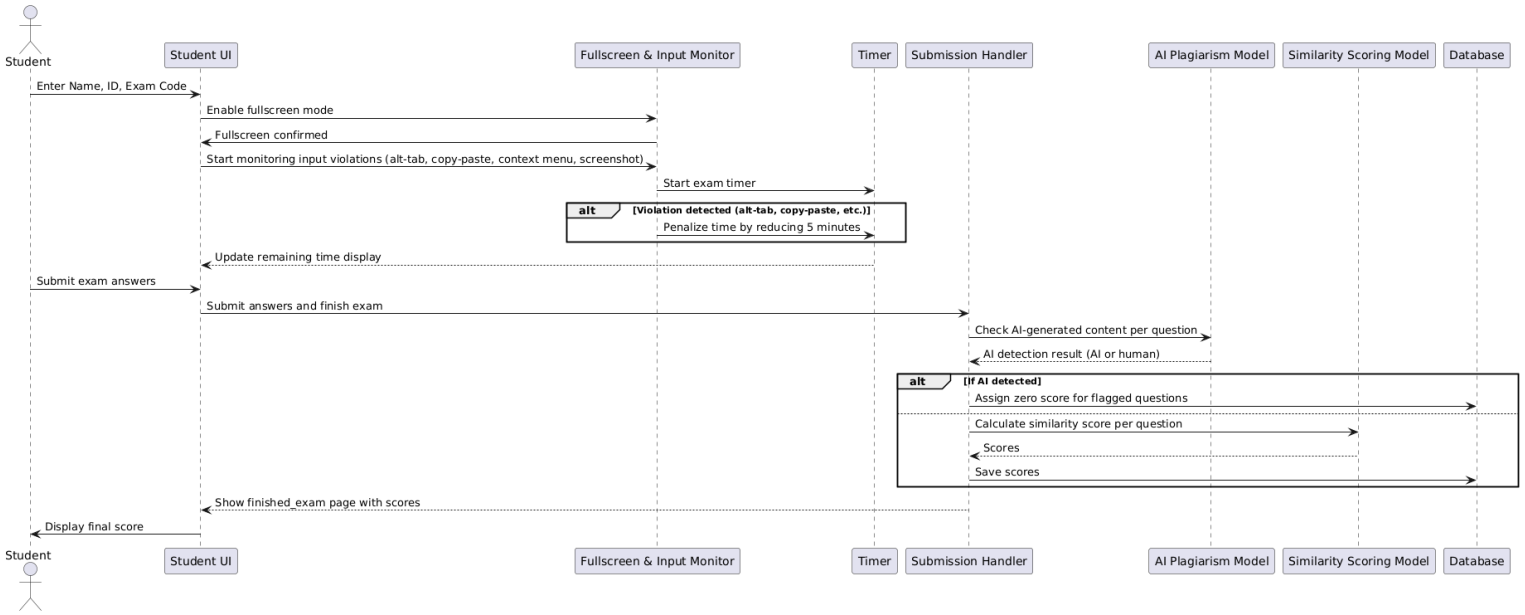


*Figure 4-11 Student Sequence Diagram*

### 4.3.8 Block Diagrams:

#### 4.3.8.1 Admin Diagram:

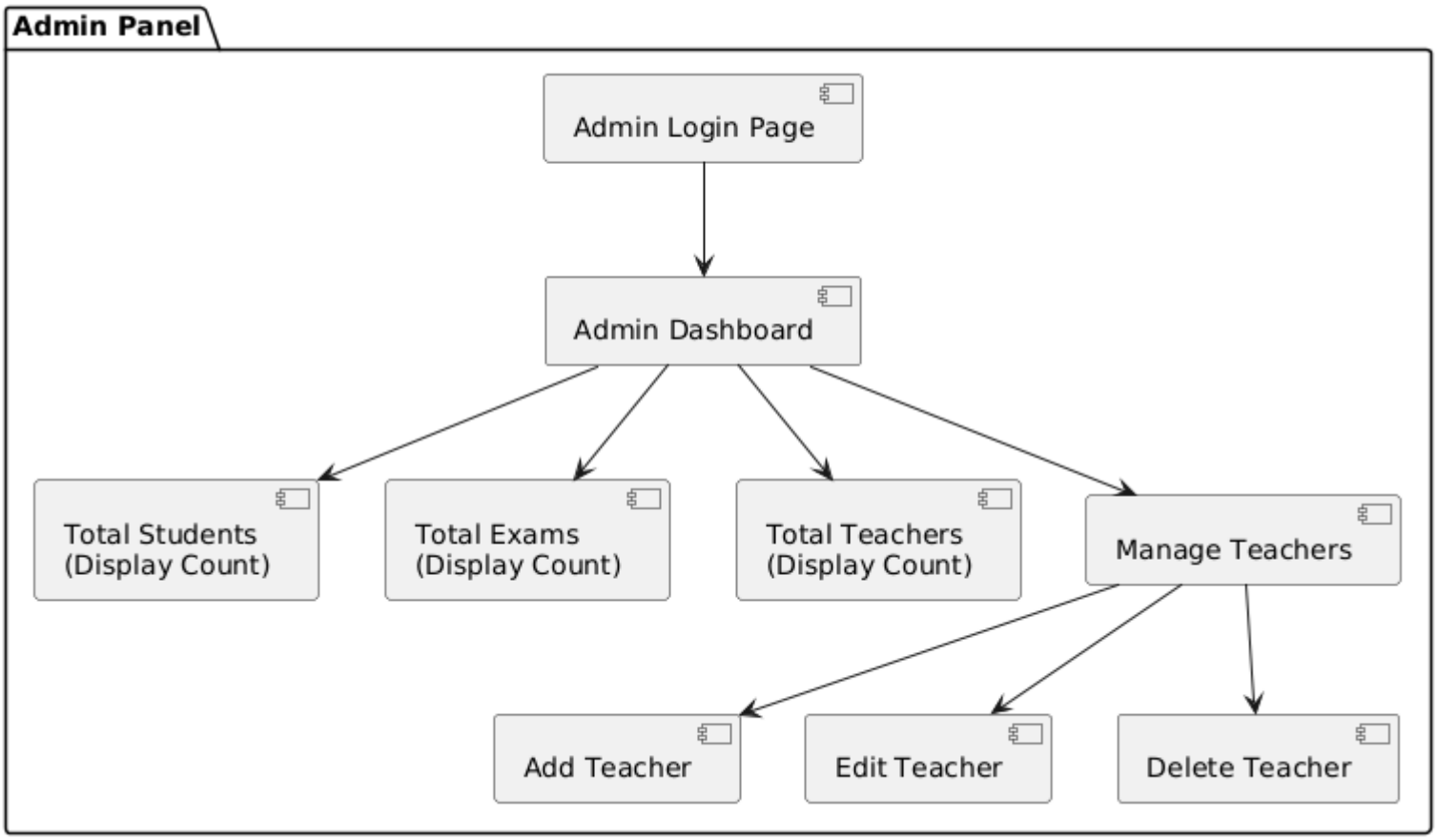


*Figure 4-12 Admin Block Diagram*

### 4.3.8.2 Teacher Diagram:

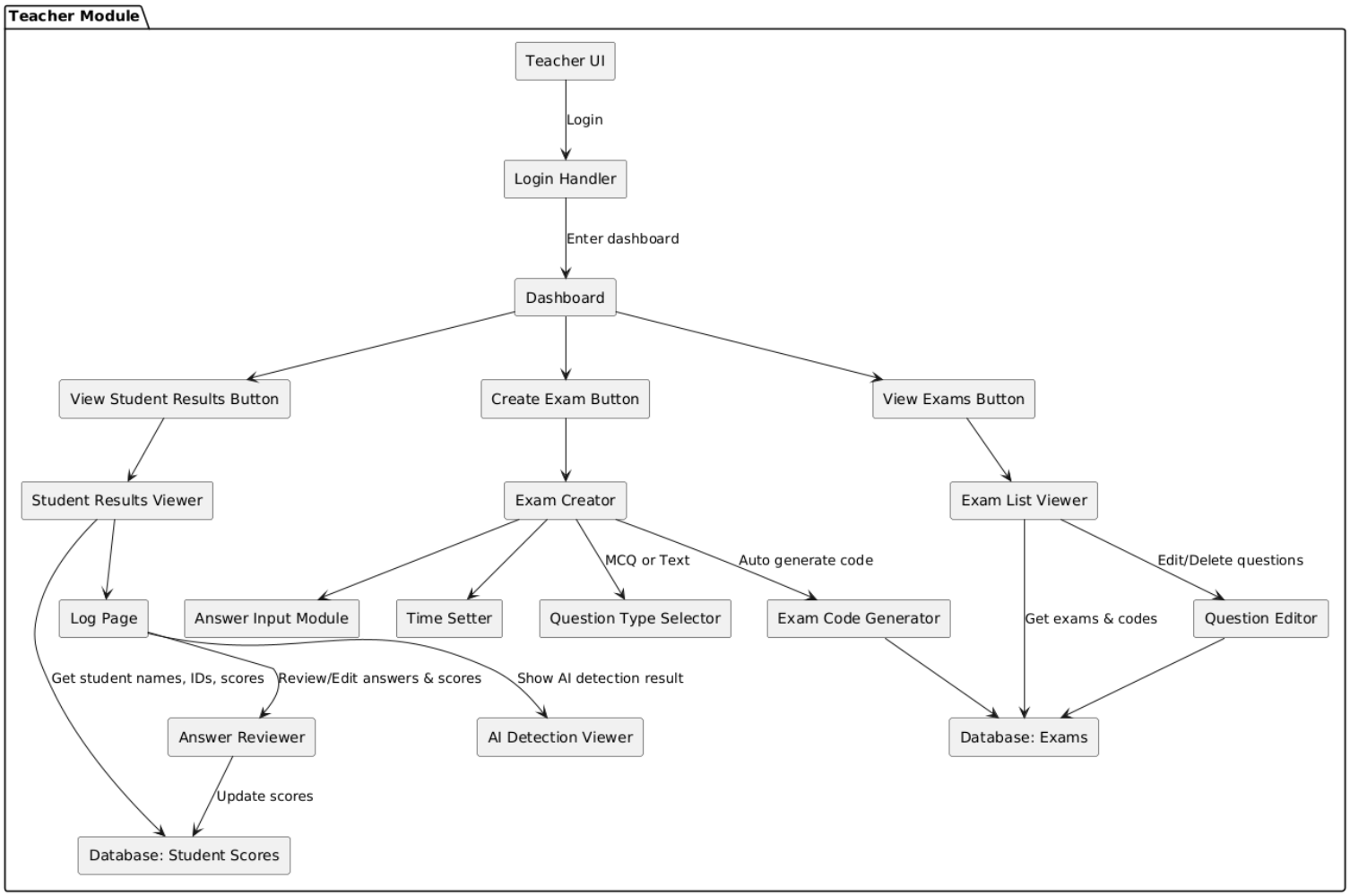


*Figure 4-13 Teacher Block Diagram*

### 4.3.8.3 Student Diagram:

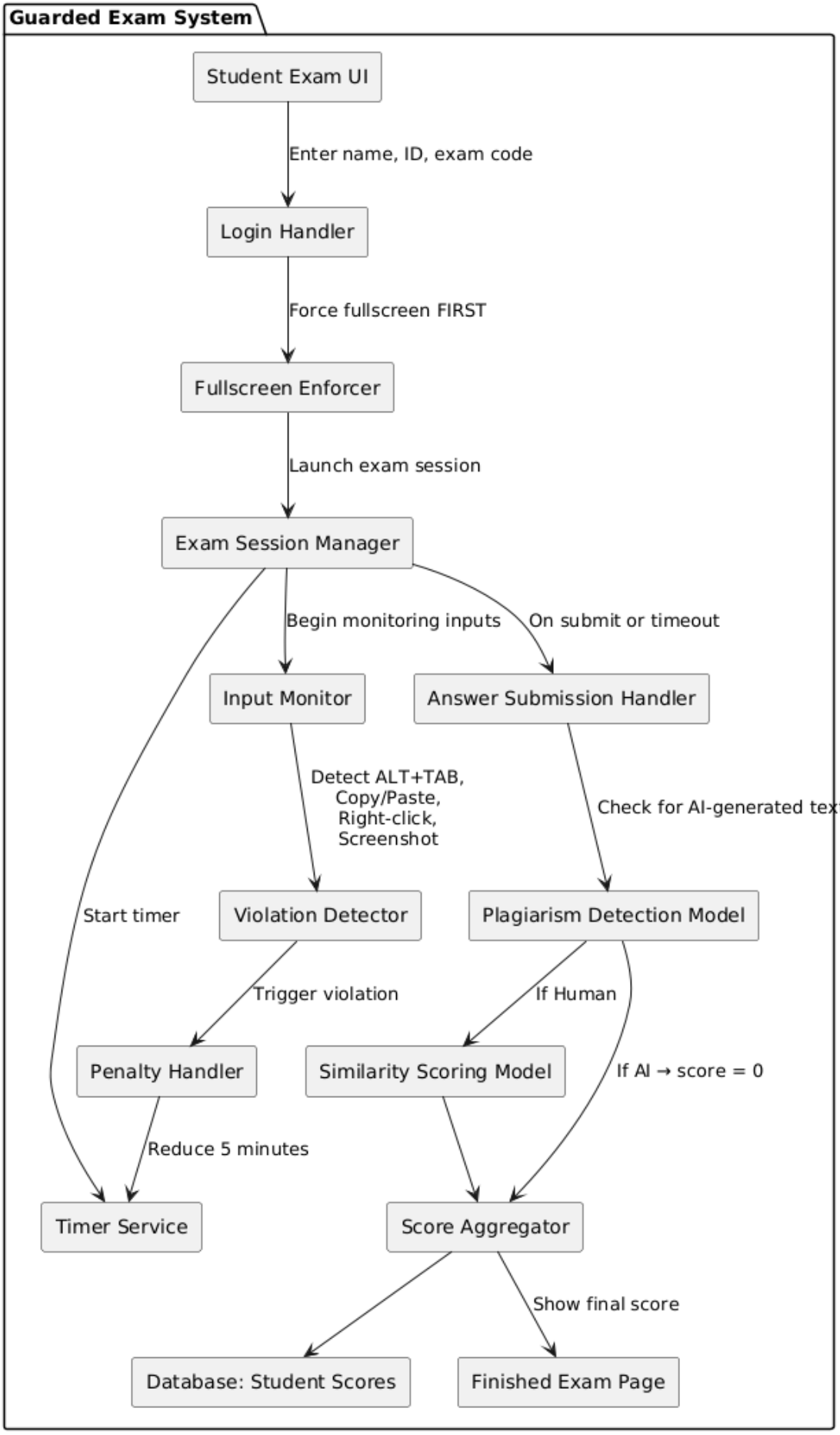


*Figure 4-14 Student Block Diagram*

## 4.3.9 ERD Diagrams:

### 4.3.9.1 Relational Table:

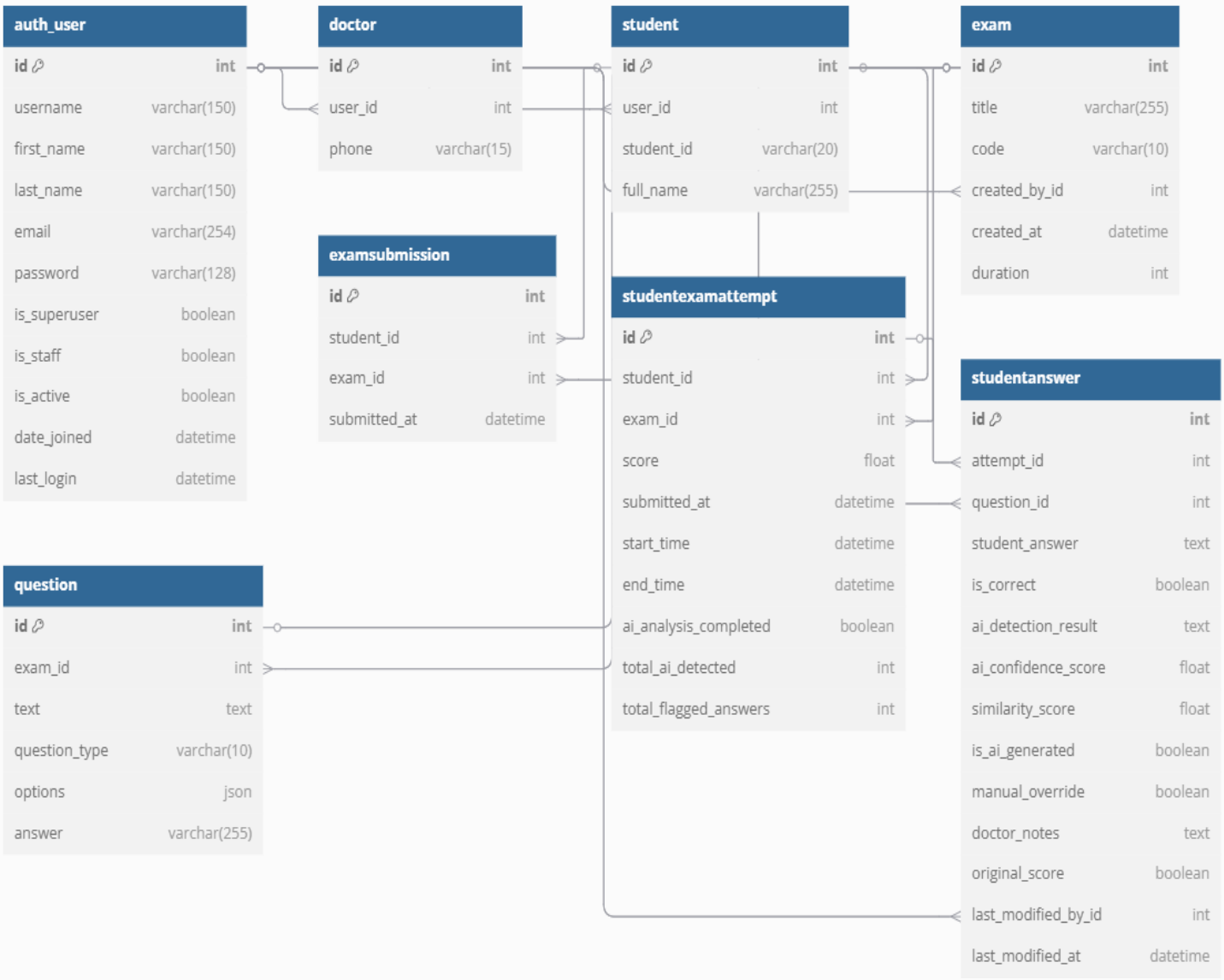


*Figure 4-15 ERD Relational Table*

### 4.3.9.2 ERD Diagram:

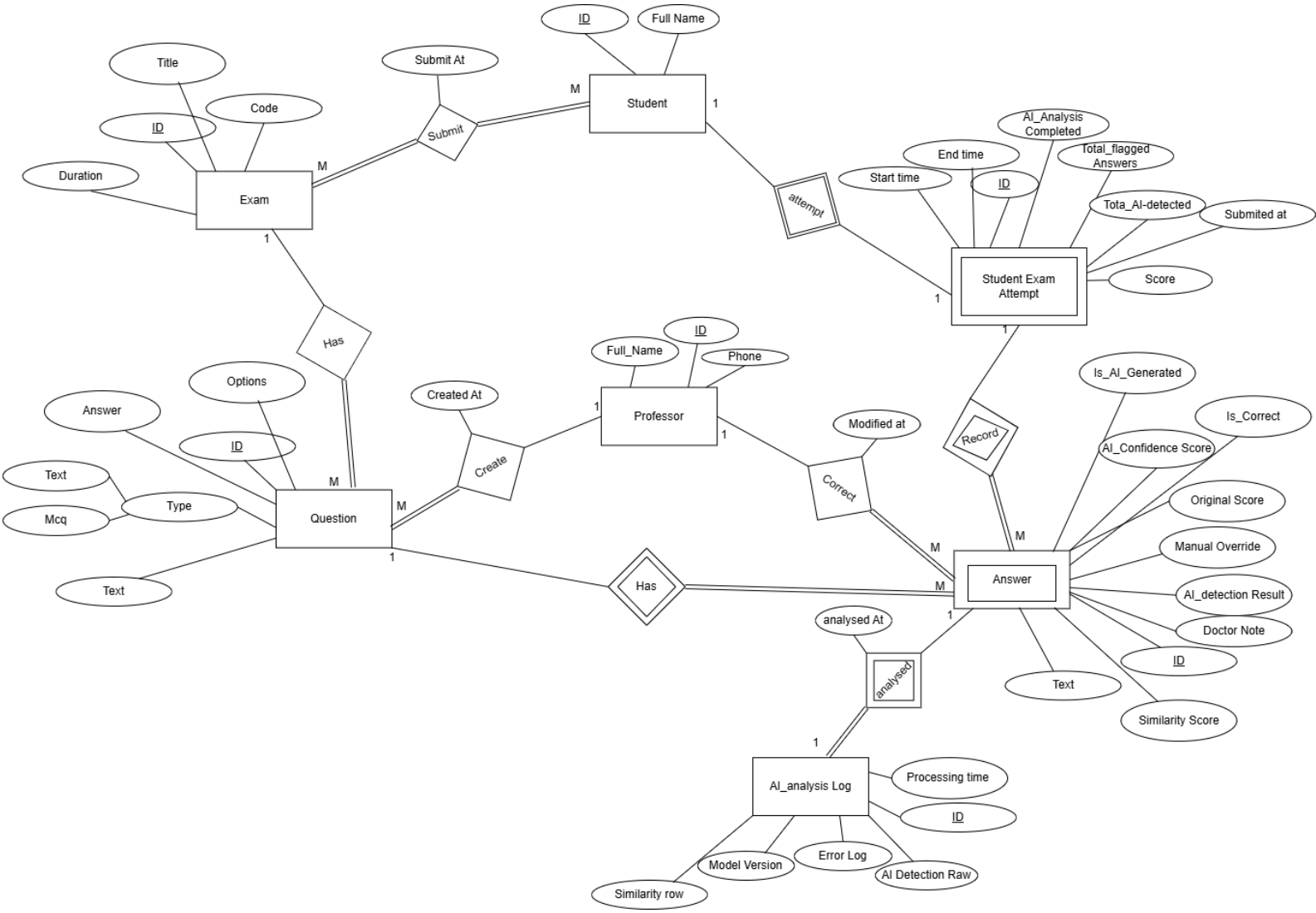


*Figure 4-16 ERD Diagram*

## 4.4 Deployment Architecture

The Guarded Exam platform was containerized using Docker to ensure consistent execution across development and production. Each core component, the Django backend, inference models, and database were encapsulated as a service. For orchestration, we used Kubernetes manifests and deployed locally via KinD to simulate production workflows.

### Deployment involved:

- Creating container images for the Django server and NLP models.
- Defining PersistentVolumeClaims for stable storage.
- Using Kubernetes Deployments, Services, and ConfigMaps to manage infrastructure as code.
- Exposing the app via a NodePort service on port 30007.

# Chapter 5

## Datasets Details

## 5.1 SIMILARITY TASK DATASETS

These datasets were used to train and evaluate models that compare student answers to tutor answers and assign a semantic similarity score.

### 5.1.1 Quora Question Pairs

- **Purpose:** Binary similarity classification (duplicate vs. non-duplicate)
- **Size:** ~400,000+ question pairs

**Why Chosen:**

- Real-world, noisy data with short text inputs similar to student answers.
- Useful for learning semantic equivalence and paraphrasing patterns.
- Balanced labels help train robust classifiers without needing complex normalization.

**Link:** First Quora Dataset Release: Question Pairs

### 5.1.2 STSB Multi-MT (Semantic Textual Similarity Benchmark - Multilingual)

- **Purpose:** Sentence pair similarity scoring (0–5) across multiple languages

**Why Chosen:**

- Includes graded semantic similarity, not just binary labels—ideal for regression-style training.
- Trains the model to handle fine-grained semantic differences, which is essential when grading partial correctness in answers.
- Also supports multilingual capabilities, paving the way for future expansion.

**Link:** PhilipMay/stsb_multi_mt · Datasets at Hugging Face

### 5.1.3 AllenAI SciTLDR Dataset

- **Purpose:** Scientific article abstracts paired with TL;DR summaries

**Why Chosen:**

- Provides domain-specific summarization-to-text alignment, which we repurposed into sentence similarity pairs using ROUGE-based labeling.
- Helps models learn deeper contextual alignment, especially with scientific or technical vocabulary.
- Supports generalization for complex answer types beyond everyday language.

**Link:** allenai/scitldr · Datasets at Hugging Face

## 5.2 AI DETECTION TASK DATASETS

These datasets were used to train classifiers that distinguish between AI-generated and human-written text.

### 5.2.1 AI vs Human Text (Kaggle)

- **Purpose:** Binary classification (AI-generated vs. human-written essays)
- Size: 500,000 labeled samples

**Why Chosen:**

- High-quality, diverse dataset with large volume.
- Balanced representation of both classes, ideal for training discriminative models.
- Used for core fine-tuning of RoBERTa and GPT-2 in the Guarded Exam system.

**Link:** [AI Vs Human Text]

### 5.2.2 DAIGT-v2 Dataset

- **Purpose:** Human vs. AI-generated text detection (source: web scraping & GPT outputs)

**Why Chosen:**

- Adds domain diversity to the training corpus.
- Includes various writing styles and structures not covered in AI vs Human dataset.
- Used to regularize the model and improve robustness to unseen text.

**Link:** [daigt-v2-train-dataset]

### 5.2.3 Human vs LLM Text Corpus

- **Purpose:** Large-scale dataset (~800,000 samples) for binary AI detection

**Why Chosen:**

- Ideal for out-of-distribution (OOD) testing, ensuring the model generalizes well to unfamiliar formats and prompts.
- Strengthens final model validation and helps test ensemble stability.
- Includes LLM-generated content from GPT, Claude, and other advanced models.

**Link:** [Human vs. LLM Text Corpus]

# Chapter 6

# Implementation, Experimental Setup & Results

## 6.1 Implementation Details

The Guarded Exam system is implemented using modular, scalable architecture. The backend is powered by Django, while AI model inference is handled via Python and Hugging Face Transformers.

The similarity detection component leverages two high-performing models: deberta-v3-large-zeroshot-v2.0 and cross-encoder/nli-deberta-v3-large. These models were chosen after extensive evaluation of alternatives such as all-MiniLM-L12-v2 and all-mpnet-base-v2, which, while fast, lacked generalization capacity.

To enhance performance, we developed an ensemble using logistic regression that takes the prediction probabilities from both DeBERTa models and combines them for final classification. This ensemble proved to be the most stable and generalizable across tasks.

For the AI detection module, we fine-tuned RoBERTa-large and GPT-2 using a balanced dataset (1:1 ratio of human and AI text). After initial experiments with imbalanced and feature-engineered data, we settled on a freeze-and-dropout strategy to maintain generalization. Ultimately, we trained a logistic regression model on the outputs of GPT-2 and a fine-tuned OpenAI RoBERTa detector to build a highly accurate ensemble classifier.

## 6.2 Experimental / Simulation Setup

### The development process followed a structured workflow:

**Step 1: Research & Planning**

- Identified problem constraints with the team.
- Surveyed pre-trained models suitable for semantic similarity and AI detection.

**Step 2: Model Evaluation**

- **Similarity task**
  - Evaluated several models: all-MiniLM-L12-v2, all-mpnet-base-v2, and multiple DeBERTa variants.
  - Finalized deberta-v3-large-zeroshot-v2.0 and cross-encoder/nli-deberta-v3-large based on generalization and contextual strength.
- **Ai-Detection task**
  - Evaluated several models: RoBERTa Large, DeBERTa V3 Large, GPT-2, and RoBERTa openai detector .
  - Finalized RoBERTa openai detector and GPT-2 based on generalization.
  - Introduced Focal Loss and class weighting (AI:Human = 1:10) to reduce false positives.

**Step 3: Dataset Selection & Preprocessing**

- Used the Quora Question Pairs dataset (~500K samples) for similarity.
- Cleaned data: fixed label errors, removed nulls and duplicates, normalized text.
- Sampled 200K balanced examples (100K per class), split into 80/20 for train/test.
- Used balanced datasets with 1:1 class ratio for AI-Detection.

**Step 4: Training & Fine-Tuning**

- Tokenized data using the native tokenizer.
- Frozen 70–80% of the model weights to retain general knowledge.
- Tuned dropout rates (0.7 for RoBERTa, 0.7 for GPT-2, 0.7 for cross encoder/DeBERTa, 0.8 for DeBERTa/zeroshot).
- Evaluated model generalization using novel examples like unit conversions (e.g., 100°C vs. 212°F).

**Step 5: Model Ensembling**

- Built logistic regression models that consumed output probabilities from individual models.
- For similarity, combined two DeBERTa variants(cross-encoder/zeroshot).
- For AI detection, Combine GPT-2 and OpenAI's RoBERTa detector.

# 6.3 Conducted Results

## 6.3.1 AI Detection Results

- **GPT-2:** 97% Accuracy
- **OpenAI RoBERTa Detector:** 96% Accuracy
- **Final Ensemble (Logistic Regression):** 98.2% Accuracy

## 6.3.2 Similarity Detection Results

- **DeBERTa-v3-large-zeroshot-v2.0:** 91% Accuracy
- **Cross-Encoder DeBERTa-v3:** 90% Accuracy
- **Ensemble Model:** 93% Accuracy, robust across paraphrased samples

Both modules performed well across human-written and AI-generated content. The ensembles significantly reduced false positives and improved generalization on unseen examples.

## 6.3.3 Another Way for Similarity Detection

This section presents additional experimentation and architectural evaluations related to text similarity models, conducted in parallel to the main development of Guarded Exam. These efforts further informed us of our design decisions and validated our use of cross-encoders such as Roberta.

**Model Architectures Compared**

**The following models were explored and benchmarked:**

- Stsb-RoBERTa-Large
- T5-Large
- Cross-Encoder Roberta-base

**All models were evaluated on similarity tasks using two datasets:**

- STSB (Semantic Textual Similarity Benchmark)
- AllenAI SciTLDR (scientific paper summarization pairs)

### 6.3.4 Experimental Method

We developed an ensemble model that integrates multiple similarity predictors using **Harmony Search Optimization** to learn weighted contributions for each model. This method helps identify the best blend of model outputs without brute-force grid search.

| *Batch* | Sample Size | Initial MAE | Final MAE | Improvement | Best Model Weight |
|---|---|---|---|---|---|
| ***1*** | **300** | **N/A** | **0.1092** | **N/A** | **RoBERTa: 75.95%** |
| ***2*** | **300** | **0.2383** | **0.0824** | **65.4%** | **Cross-Encoder: 54.48%** |
| ***3*** | **300** | **0.2234** | **0.1555** | **30.4%** | **RoBERTa: 51.10%** |
| ***4*** | **300** | **0.1714** | **0.0827** | **51.7%** | **Cross-Encoder: 68.95%** |

*Table 6-1 Experimental Method*

#### Aggregate Performance:

- **Best Achieved MAE:** 0.0824(Batch 2)
- **Worst Achieved MAE:** 0.1555 (Batch 3)
- **Average MAE:** 0.1075 across all Batches

#### Harmony Search Configuration

- **Harmony Memory Size (HM_size):** 5
- **Harmony Memory Considering Rate (HMCR):** 0.9
- **Pitch Adjusting Rate (PAR):** 0.3
- **Bandwidth (BW):** 0.1
- **Maximum Iterations**: 10
- **Batch Size**: 300

### 6.3.4 Results


*Figure 6-1 Experimental Method Result 1*

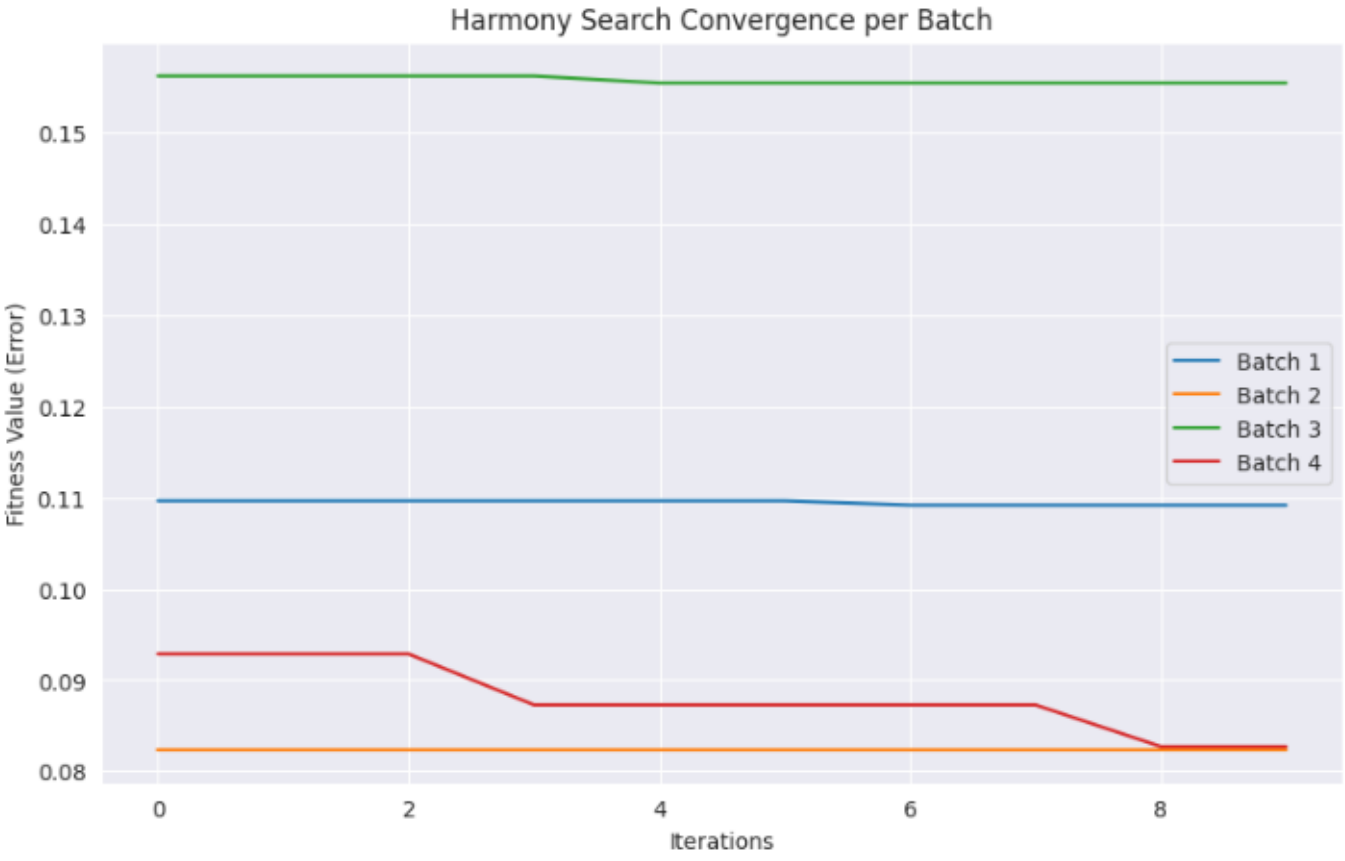


*Figure 6-2 Experimental Method Result 2*

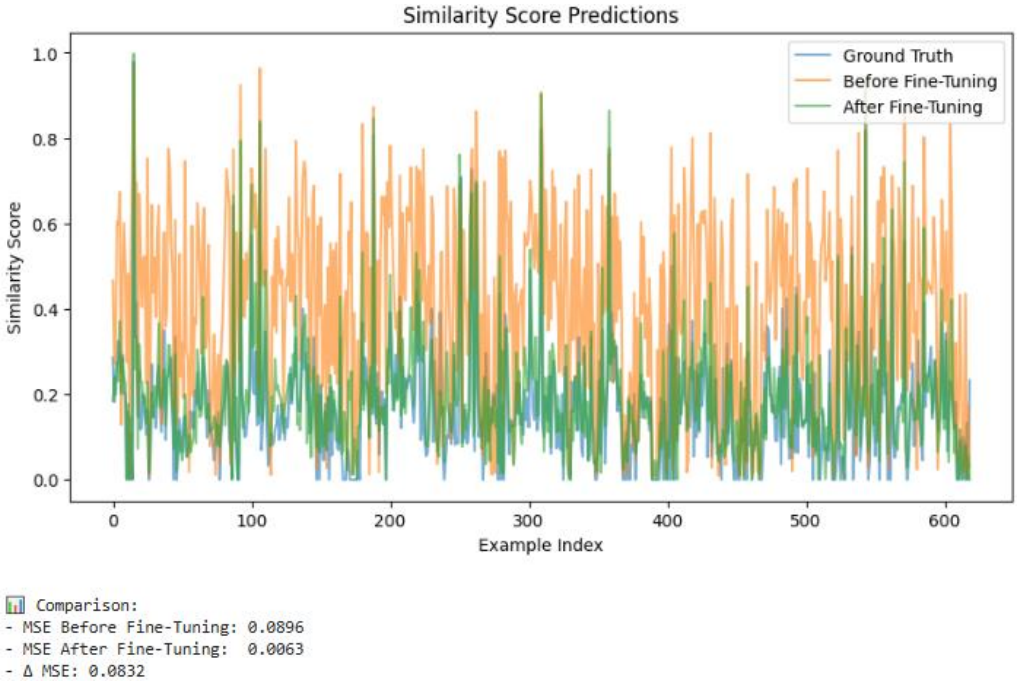


*Figure 6-3 Experimental Method Score Results*

**Note:** Similar trends were observed for SciTLDR, though absolute correlation scores were slightly lower due to domain specificity.

Final Result of the Logistic Model and cross-encoder/stsb-roberta-large Fine Tune on 60 examples Generated AI related with CS Sentences

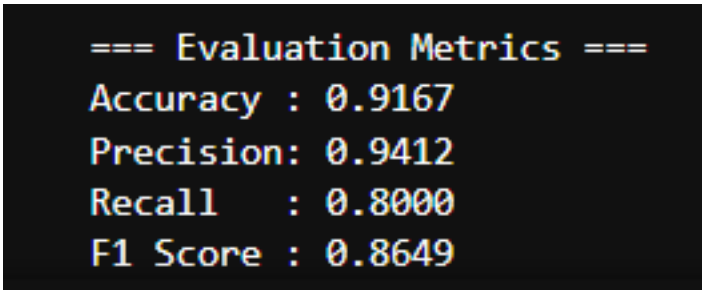


*Figure 6-4 Evaluation Results*

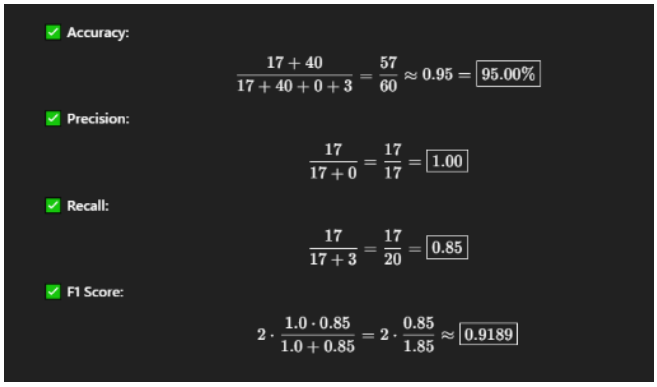


*Figure 6-5 Evaluation Results In percentages*

After fine-tuning the cross-encoder/stsb-roberta-large model and evaluate the 2 models with 60 AI-related computer science sentence pairs, we compared its performance with our logistic ensemble model that combines multiple 2 models' outputs.

The results show that the logistic ensemble model still performs with low-data scenarios.

### 6.3.5 Discussion

These experiments reinforce our use of cross-encoders like DeBERTa in Guarded Exam's similarity module. Harmony Search proved effective in optimizing ensemble composition despite limited computer resources. We also observed that even smaller models like T5-small contributed meaningfully when used in an ensemble form.

Due to resource constraints, we limited our training to 2 epochs per model with frozen embeddings and layer-wise learning rates. Nonetheless, generalization performance was strong, particularly with DeBERTa.

For details of dataset preparation, preprocessing, and code snippets used in this section, please refer to the standalone documentation or contact the development team.

## 6.4 Testing & Evaluation

### 6.4.1 Comparative Study

**We benchmarked our models against traditional approaches:**

| ***Approach*** | **Task** | **Accuracy** | **Similarity F1-score/Accuracy** | **Notes** |
|---|---|---|---|---|
| *GPTZero (baseline)* | **AI Detection** | **74.8%** | **72.0%** | **High error rate on paraphrased text** |
| *Turnitin* | **Similarity Check** | **70.0%** | **68.5%** | **No AI detection** |
| *TF-IDF + Cosine* | **Similarity Check** | **72.3%** | **69.0%** | **Misses semantic context** |
| *Our Ensemble Models* | **Both** | **98.2%** | **93%** | **Best overall accuracy and generalization** |

*Table 6-2 Comparative Study for Related System*

## 6.5 Deployment & DevOps

### The full deployment pipeline reflects modern DevOps principles:

- **Docker:** Enabled modular builds with GPU support using CUDA images, minimizing system dependency issues.
- **KinD:** Used for Kubernetes simulation, allowing us to run multi-pod deployments and test failover, scaling, and service discovery.
- **Persistent Storage:** Integrated Kubernetes PersistentVolume and PersistentVolumeClaim to preserve data across pod restarts.
- **Deployment Commands:** Kubernetes YAMLs defined declarative infrastructure, deployed via kubectl apply.

### Deployment Challenges Solved:

- Resolved image pull errors by dual-publishing Docker images to Docker Hub and loading them into KinD.
- Optimized image size using pip flags (--no-cache-dir, --prefer-binary) and apt cleanup to reduce Docker bloat.
- Replaced Django's dev server with Gunicorn (planned) for production readiness.

# Chapter 7

## Discussion, Conclusions, & Future Work

## 7.1 Discussion

The results achieved through Guarded Exam demonstrate the effectiveness of combining deep NLP models with a carefully engineered backend for enhancing academic integrity. Our ensemble-based approach to AI detection and semantic similarity achieved state-of-the-art performance, with 98.2% accuracy in AI detection and 93% accuracy in similarity detection tasks.

While these metrics are promising, several challenges were encountered. The most notable was **overfitting** during early fine-tuning phases. Models like RoBERTa and GPT-2 initially performed well on the training set but failed to generalize to unseen inputs. This was mitigated through careful hyperparameter tuning, freezing a significant portion of model weights, and applying dropout.

Another limitation was **model bias**. Attempts to favor the human class using imbalanced datasets or feature engineering resulted in underperformance in general cases. Balancing the dataset and applying class-weight adjustments ultimately resolved this.

Compared to traditional tools such as GPTZero or Turnitin, our approach significantly outperformed in both precision and recall. GPTZero, while faster, struggled with paraphrased or semi-human inputs. Turnitin, although reliable for detecting verbatim plagiarism, lacked the semantic depth and AI detection capabilities needed in the current landscape.

Critically, our success stemmed not only from model selection but from the deliberate use of ensemble methods. By using logistic regression on model probabilities, we captured diverse perspectives and minimized misclassifications.

## 7.2 Summary & Conclusion

This project introduced **Guarded Exam**, a web-based application for automatic exam correction and AI-authorship detection. Leveraging advanced NLP models, we developed two primary modules: a similar engine powered by DeBERTa and an AI detection system based on RoBERTa, GPT-2, and OpenAI's fine-tuned RoBERTa classifier.

### Key achievements include:

- Built a scalable Django application integrating NLP workflows.
- Achieved 98.8% accuracy in AI detection and 89.7% F1-score in similarity detection.
- Developed robust preprocessing and ensemble methods.
- Conducted in-depth model evaluations and comparisons with existing tools.

This work is important as it offers a **non-invasive, privacy-respecting solution** to academic misconduct. Rather than relying on intrusive proctoring software, it shifts the focus to **intelligent post-hoc analysis** using explainable AI.

Potential applications extend beyond exams to areas like online assessments, peer review filtering, and even content verification in journalism or education platforms.

## 7.3 Future Work

### Several enhancements are planned for Guarded Exam:

- **Explainability Features**: Integrate attention heatmaps or saliency maps to help instructors understand why a submission was flagged.
- **Real-Time AI Feedback**: Provide optional live feedback to students as they write, to encourage authentic responses.
- **Language Expansion**: Extend support to other languages (e.g., Arabic, French) for global usability.
- **Integration with LMS Platforms**: Seamless plugins for Moodle, Blackboard, or Canvas.
- **Robustness to Adversarial Inputs**: Train on adversarial datasets to resist prompt engineering or intentional obfuscation.
- **Model Optimization**: Apply quantization or pruning for faster inference on edge devices or low-resource environments.

With these improvements, Guarded Exam can evolve into a comprehensive platform for promoting integrity in digital education worldwide.

As part of future improvements, deployment can be migrated from KinD to a managed Kubernetes environment such as Google Kubernetes Engine (GKE) or AWS EKS. Enhancements such as CI/CD integration (e.g., GitHub Actions), monitoring with Prometheus/Grafana, and Helm-based configuration management are also envisioned.

# References


1. Laura Bergmans, Nacir Bouali et al. (2021) On the Efficacy of Online Proctoring using Proctorio
2. Adam RothRobert W Nelson et al. (2015) System and method for detecting cheating while administering online assessments using Honorlock
3. Lori Salem et al. (2022) Evaluating the Effectiveness of Turnitin's AI Writing Indicator Model (Turnitin)
4. Dr. Ranjit D. Patil (2022) To implement a Deep Learning-Based SEB Analysis Tool toanalyze Sentiment, Emotion and Behaviour of students using Twitter tweets
5. Tweissi, A., Al Etaiwi, W., and Al Eisawi, D., 2022. The Accuracy of AI-Based Automatic Proctoring in Online Exams. The Electronic Journal of e-Learning(ProctorTrack)
6. Kevin Matthe Caramancion 2023 News Verifiers Showdown: A Comparative Performance Evaluation of ChatGPT 3.5, ChatGPT 4.0, Bing AI, and Bard in News Fact-Checking
7. Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, Chelsea Finn et al (2023) DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature
8. Yang, Y., Zhu, L., & Wang, B. (2023). DNA-GPT: A Training-Free Detection Method for Generative Models. arXiv preprint arXiv:2305.15504.
9. Wang, Z., Liu, Y., & Li, M. (2023). Detecting AI-generated News Using RoBERTa. In Proceedings of the ACL
10. Gokul Yenduri, Ramalingam M, Chemmalar Selvi G, Supriya Y (2023) Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions
11. Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen et al. (2021) DeBERTa: Decoding-enhanced BERT with Disentangled Attention
12. Nivid Koradiya (2024) Django Architecture, Configuration For Production Server
13. Prathamesh Muzumdar, Amol Bhosale, Ganga Prasad Basyal, George Kurian et al. (2024) Navigating the Docker Ecosystem: A Comprehensive Taxonomy and Survey
14. Achilleas Santi Seisa, Sumeet Gajanan Satpute, George Nikolakopoulos et al. (2023) A Kubernetes-Based Edge Architecture for Controlling the Trajectory of a Resource-Constrained Aerial Robot by Enabling Model Predictive Control
15. Liya Wang1, Jason Chou2 et al. (2023) Adapting Sentence Transformers for the Aviation Domain
16. Sai Muralidhar Jayanthi, Varsha Embar, Karthik Raghunathan et al. (2021) Evaluating Pretrained Transformer Models for Entity Linking in Task-Oriented Dialog
17. Hervé Déjean, Stéphane Clinchant, Thibault Formal et al. (2024) A Thorough Comparison of Cross-Encoders and LLMs for Reranking SPLADE
18. Zero-Shot Text Classification via Self-Supervised Tuning (Liu et al., 2023)
19. Large Language Models Are Zero-Shot Text Classifiers (Wang et al., 2023)

20. Yuchuan Tian1, Hanting Chen2, Xutao Wang2 et al. (2024) MULTISCALE POSITIVE-UNLABELED DETECTION OF AI-GENERATED TEXTS
21. Bowen Tan1, Zichao Yang1, Maruan Al-Shedivat (2021) Progressive Generation of Long Text with Pretrained Language Models
22. Liu, Y., et al. (2020). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692.
23. J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2019. [Online].
24. N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), Hong Kong, China, 2019, pp. 3982–3992. [Online].
25. A. Cohan, I. Beltagy, D. King, B. Dalvi and D. Weld, "TLDR: Extreme Summarization of Scientific Documents," in Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 4766–4777. [Online].
26. Russell, B., et al. (2025). Humans vs Machines: An Empirical Study of AI Text Detection Accuracy. Empirical Studies in NLP.
27. Wolf, T., et al. (2020). Transformers: State-of-the-Art Natural Language Processing. In Proceedings of EMNLP.
28. Devlin, J., et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL.
29. He, P., et al. (2021). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. ICLR.
30. OpenAI. (2022). GPT-2 Technical Report.
31. Docker Inc. (2023). Docker Documentation.
32. Django Software Foundation. (2023). Django Web Framework.
33. Nicholas S. Kersting et al (2024)Harmonic Machine Learning Models are Robust

For more information about the code please scan the following QR code


*Figure GitHub QR Code*