GradeX-AI-Powered Automated Exam Grading System for Accurate,Efficient,and Scalable Answer Evalution

A Project Report

Submitted By

KUMARESAN K P	**821121104027**
MOHAMED ASICK A	**821121104032**
LINGESH R S	**821121104703**

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING
in

COMPUTER SCIENCE AND ENGINEERING

KINGS COLLEGE OF ENGINEERING, PUNALKULAM

ANNA UNIVERSITY: CHENNAI-600 025

MAY 2025

BONAFIDE CERTIFICATE

Certified that this Project report "**GradeX-AI-Powered Automated Exam Grading System for Accurate,Efficient,and Scalable Answer Evalution**" is the bonafide work of who "**KUMARESAN K P(821121104027),MOHAMED ASICK A (821121104032), LINGESH R S (821121104703)**" carried out the project under my supervision during the year 2024-2025.

Dr. S.M. UMA, Ph.d.,
ASSOCIATE PROFESSOR
Head of the Department,
Department Of CSE,
Kings College of Engineeing,
Punalkulam.

Dr. S.M. UMA, Ph.d.,
ASSOCIATE PROFESSOR/GUIDE
Head of the Department,
Department Of CSE,
Kings College of Engineeing,
Punalkulam.

Submitted for the Anna University: Chennai, Practical Examination Held on _____________2025

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We give all glory and honor to the almighty for his blessings and divine help which helped us to complete the project work successfully.

The project work has been undertaken and completed with direct and indirect help of many people and we would like to acknowledge the same.

We express our deepest gratitude to our secretary **Dr.R.Rajendran M.A., M.Phil., Ph.D.,** for his moral support.

Our grateful thanks to our Principal **Dr. J.Aruputha Vijaya Selvi, M.E., Ph.D**., and our **Vice Principal Dr. S. Sivakumar, M.Tech., Ph.D**., **Kings College of Engineering, Punalkulam**, for permitting to do to project work successfully.

We express our warm thanks **to Dr. S.M. Uma, Head of the Department, Computer Science and Engineering**, for allowing us to do our project. With immense pleasure, we extend our sincere and heartfelt thanks to our internal project Guide Mr. **M.ARUN, M.E & Project coordinator Dr. S.M. Uma, Head of the Department, Computer Science and Engineering** We also extend our sincere thanks to all our staff members and technical assistants of CSE department.

Our deepest thanks to our parents for uploading us by providing professional education and for their prayerful support that made us to complete the project phase successfully.

DECLARATION

We hereby declare that the word entitled "**GradeX-AI-Powered Automated Exam Grading System for Accurate,Efficient,and Scalable Answer Evalution**" is submitted in partial fulfillment of the requirement for the award of the degree in B.E., Anna University, Chennai, is a record of our work carried out by as during the academic year 2024 – 2025 under the **supervision** of **Dr. S.M. Uma., Head of the Department, Computer Science and Engineering**. The extent and source of information are derived from the existing literature and have been indicated through the dissertation at the appropriate places. The matter embodied in this work is original and has not been submitted for the award of any other degree or diploma, either in this or any other university.

KUMARESAN K P
(821121104027)

MOHAMED ASICK A
(821121104032)

LINGESH R S
(821121104703)

Dr. S.M. UMA, Ph.d.,
ASSOCIATE PROFESSOR
Head of The Department,
Department Of CSE,
Kings College of Engineeing,
Punalkulam.

ABSTRACT

The education system has undergone a significant transformation with the integration of technology. Teaching methods have become more engaging and informative through the use of projectors, online tutorials, instructional videos, and animations. However, despite these advancements, the evaluation process remains largely traditional—relying on manual correction by teachers. This manual approach is time-consuming and prone to human error. The proposed project aims to address these challenges by introducing an AI-powered solution that automates the evaluation of answer sheets. By leveraging machine learning and language models, this system reduces the effort and inaccuracies associated with manual grading. Compared to offline methods, online evaluation offers greater speed, efficiency, and reliability, making it a more effective alternative for modern educational needs.

Chapter 1
INTRODUCTION

1.1 Introduction

With the rapid integration of technology into education, teaching has become more engaging through tools like projectors, online tutorials, and animated video lessons. However, while instructional methods have evolved, the evaluation of descriptive answer sheets remains largely manual. Faculty members still evaluate exam papers by hand, which is time-consuming, inconsistent, and susceptible to human error. Subjectivity and emotional variation during manual correction can also lead to unfair grading.

To address these challenges, the Gradex project proposes an automated evaluation system that utilizes Optical Character Recognition (OCR) and Natural Language Processing (NLP) to streamline and standardize answer sheet correction. By replacing the traditional manual evaluation process with AI-driven mechanisms, we can ensure faster, more consistent, and transparent grading.

This system uses Tesseract OCR to extract handwritten or printed text from scanned answer sheets. The extracted text is then preprocessed and evaluated using Ollama language models, which analyze the content against provided answer keys and question papers. This unified evaluation logic ensures fairness and eliminates personal bias.

The project also features a user-friendly interface where teachers can upload answer sheets, organize evaluations by exam type (e.g., CAT 1, Term 1), and access results in real-time. All evaluated data is stored in a structured database and can be exported in CSV format or PDF format for reporting or record-keeping.

1.2 Problem Statement

Despite advancements in teaching technologies, most educational institutions still rely on traditional, manual methods for evaluating exam papers. This manual evaluation process is labor-intensive, time-consuming, and prone to inconsistency due to individual differences among evaluators. Additionally, the lack of standardization in marking schemes can lead to unfair assessments.

To overcome these issues, Gradex introduces an AI-based automated evaluation system. This solution reduces correction time, eliminates evaluator bias, and ensures a consistent and objective grading process. The system also provides quick and accessible results, improving the overall efficiency of academic evaluations.

1.3 Objectives

The objectives of the Gradex system are:

- To automate the evaluation of internal exam papers and reduce the time and manual effort required.
- To ensure consistency in marking by applying a common evaluation standard powered by AI.
- To minimize human errors and emotional bias in the correction process.
- To provide instant results and transparent grading accessible to both teachers and students.
- To categorize and manage results exam-wise (e.g., Term 1, CAT 1, CAT 2) for easier reporting and tracking.

1.4 Scope

The scope of this project is to develop a **fully automated answer sheet evaluation platform** for internal academic assessments. The system performs the following tasks:

- Extracts text from scanned answer sheets using Tesseract OCR.
- Uses Ollama LLMs to compare extracted answers with the answer key.
- Scores answers, calculates total marks, and stores the results in a database.
- Offers an interface for teachers to upload, categorize, and manage answer sheets.
- Provides downloadable CSV reports for easy sharing and record maintenance.
- Can be extended for use in schools, colleges, coaching centers, and competitive exam bodies.

Chapter 2

LITERATURE SURVEY

A literature survey or a literature review in a project report is that section which shows the various analysis and research made in the field of interest and the results already published, taking into account the various parameters and the extent of the project.

2.1 Introduction

In education, grading handwritten exam answers is a time-consuming and labor-intensive task. Traditional grading approaches face challenges, such as subjectivity, inconsistency, and human error, particularly with large volumes of scripts. Automating this process can not only reduce the grading workload but also provide faster and fairer feedback to students.

To address these needs, the proposed system leverages advanced technologies, integrating Optical Character Recognition (OCR) with Large Language Models (LLMs) such as BERT and LLaMA. OCR enables accurate text extraction from handwritten responses, which can then be refined and interpreted by LLMs for effective grading. These models are adept at understanding semantic meaning, ensuring that answers are evaluated based on content rather than exact phrasing. This system also enhances reliability by mitigating OCR errors and improving grading accuracy across varied subjects and response styles.

By combining OCR and LLM-based semantic analysis, this system aims to transform grading into a more efficient, scalable, and objective process. The result is a significant step forward in educational technology, offering a consistent grading solution adaptable to diverse educational environments and large-scale assessments.

2.2 Automated Evaluation of Student Answers using NLP Techniques

Authors: Kumar A., Singh R.

Year: 2020

This research focused on designing a rule-based Natural Language Processing (NLP) system to evaluate descriptive answers submitted by students. The system processes both student responses and model answers through tokenization, stop-word removal, and stemming. It then performs keyword matching and uses cosine similarity to measure textual overlap between the student's answer and the model solution. The evaluation is based on the number of matched keywords and overall similarity. The system provides a lightweight solution for automating descriptive answer scoring in controlled academic environments.

Disadvantages:

- The method lacks semantic understanding and cannot comprehend paraphrased or grammatically restructured answers.
- It performs poorly when students use synonyms or varied sentence structures.
- It requires manual configuration of keywords and threshold similarity values for different questions

2.3 Intelligent Assessment System using Machine Learning for Descriptive Answers

Authors: Sharma N., Patel V.

Year: 2021

This system adopted a data-driven machine learning approach to score descriptive student answers. A training dataset was compiled consisting of various answers labeled with scores assigned by human examiners. Using TF-IDF for feature extraction and classifiers like Support Vector Machines (SVM) and Random Forests, the model was trained to predict scores based on patterns learned from the labeled data. This approach enabled generalization to new student answers without predefined rules, and allowed faster and automated scoring once trained.

Disadvantages:

- The model's performance was highly dependent on the quality and diversity of the training data.
- It was not effective when applied to questions outside the training distribution or domain.
- The system lacked semantic reasoning and contextual understanding.

2.4 Automatic Grading of Short Answers using BERT

Authors: Zhang Y., Liu H.

Year: 2022

This project introduced a transformer-based model, BERT, to perform automatic grading by capturing the deep contextual meaning of student answers. The model was fine-tuned on a dataset containing question-answer-score triples. BERT's attention mechanism allowed the system to understand the relevance of each word in context, enabling it to evaluate even paraphrased or semantically equivalent answers accurately. This approach showed significant improvement over traditional keyword or rule-based models.

Disadvantages:

- The system was computationally expensive, requiring GPU acceleration for real-time use.
- It required careful fine-tuning for different subjects or types of questions.

- Interpretability was a challenge, as deep learning decisions were opaque to users and teacher

2.5 OCR-Based Automated Examination System

Authors: Das S., Roy A.

Year: 2019

This work focused on using Optical Character Recognition (OCR) to digitize handwritten student answers. The system used Tesseract OCR to convert scanned answer sheets into text. After extraction, the system applied basic keyword-based matching to evaluate content correctness. This project served as an initial step towards paperless evaluations by eliminating manual data entry.

Disadvantages:

- OCR performance degraded significantly with poor handwriting or low image quality.
- Keyword matching lacked depth and failed to consider context or answer coherence.
- The system was not scalable for subjective, multi-paragraph answers.

2.6 Assessment using AI: A Comparative Study of Rule-Based and ML Models

Authors: Thomas J., Mehta R.

Year: 2021

This study compared different AI approaches — rule-based, traditional machine learning, and deep learning models — for automated evaluation. The authors developed small-scale prototypes for each category and analyzed them for accuracy, training time, scalability, and generalization. This comparative framework helped highlight the strengths and limitations of each technique, offering guidance for hybrid model development.

Disadvantages:

- No single model performed well across all question types.
- Deep models required significant resources and domain-specific tuning.
- Rule-based systems needed continuous manual maintenance and updates.

Chapter 3
System Analysis

3.1 Existing System

3.1.1 Traditional Manual Grading and Its Limitations

Manual evaluation of descriptive answer sheets remains the dominant approach in most educational institutions. In this process, teachers individually review each student's handwritten responses, compare them with expected answers, and assign marks based on their interpretation and experience. While this method allows for human judgment and flexibility, it suffers from several critical limitations. Firstly, it is extremely **time-consuming**, especially when dealing with large volumes of answer sheets, such as during term-end or competitive examinations. Teachers often spend long hours reading through pages of handwritten content, leading to fatigue and slower result processing.

Another key drawback is the **subjectivity and human bias** involved in manual grading. Different teachers may interpret and score the same answer differently due to personal preferences, expectations, or unconscious bias. This lack of standardization results in inconsistent and unfair scoring. Additionally, **errors and inconsistencies** frequently occur, particularly under tight deadlines. Mistakes in mark calculation, misinterpretation of answers, and overlooked content are common when teachers are under pressure or when answer sheets are not clearly written. These challenges have highlighted the urgent need for a more objective, scalable, and accurate solution for evaluating descriptive answers.

3.1.2 Automated Grading for Objective-Type Questions

Automated grading systems have already seen success in handling objective-type assessments like multiple-choice questions, true/false, and fill-in-the-blank formats. E-learning platforms such as Google Forms, Moodle, and Edmodo employ rule-based grading mechanisms, which rely on exact answer matching. These systems offer speed and accuracy in assessing large datasets with zero ambiguity in answers. However, their limitations become apparent when used to evaluate descriptive or subjective answers, as rule-based systems struggle to interpret synonyms, paraphrased responses, or sentence-level meaning. They are unable to provide meaningful feedback or assess the creativity and reasoning skills exhibited in longer answers.

3.1.3.OCR-Based Answer Extraction

To digitize and process handwritten answer sheets, Optical Character Recognition (OCR) is an essential technology. Several tools and APIs like Tesseract OCR, Google Vision API, and OpenAI's OCR models have been applied to extract text from scanned images or photographs of answer sheets. While these tools work well for printed text, handwritten content remains a major challenge due to varied writing styles, poor scan quality, image noise, and inconsistencies in character formation. These limitations often result in partially or incorrectly extracted text, which negatively affects the downstream evaluation process. The current study overcomes this by employing Ollama OCR, a deep learning-based handwriting recognition model that improves accuracy significantly. It leverages neural networks trained on diverse handwriting samples, enabling better generalization across different student answer scripts.

3.1.4.NLP and AI in Descriptive Answer Evaluation

Recent advancements in Natural Language Processing (NLP) have opened the door to AI-powered evaluation systems that go beyond keyword matching and instead focus on understanding the semantic meaning of text. By applying word embeddings like Word2Vec, GloVe, and FastText, along with more advanced transformer-based models such as BERT, T5, LLaMA, and GPT, these systems can analyze context and evaluate the logical structure and relevance of student answers. Unlike traditional rule-based systems, NLP models are capable of context-aware assessment, recognizing correct answers written in different ways using synonyms or paraphrased structures.

One of the major strengths of AI-driven evaluation is semantic similarity matching, where student responses are compared with model answers not through exact word matching but through contextual alignment. These models also handle grammatical variations and sentence restructuring, thus offering a more flexible and accurate evaluation method. Studies show that transformer-based models like BERT and GPT consistently outperform traditional techniques in grading accuracy and interpretation of complex answers, making them suitable for deployment in educational settings.

3.2 Disadvantages of Existing Systems

- Time-consuming process for teachers, especially during large-scale examinations.
- Lack of scalability with increasing student population.
- High risk of human bias and subjectivity in grading.
- Inconsistencies and errors due to fatigue or oversight.
- Minimal or no feedback provided to students for improvement.
- Poor performance of automated systems on descriptive answers.
- Dependence on exact keyword matching in automated grading systems.
- OCR tools struggle with handwritten text due to variations in writing style and image quality.
- No digital storage or centralized access to student performance data.
- Limited support for semantic understanding and context-aware evaluation.
- Absence of data-driven analytics and feedback loops for institutional improvement.

3.3 Proposed System

The proposed system, Gradex, is an AI-powered answer sheet evaluation platform designed to overcome the limitations of traditional manual grading and rule-based automated systems. It combines advanced OCR, NLP, and semantic analysis techniques to accurately evaluate descriptive student answers. By digitizing handwritten responses, extracting meaningful content, and applying intelligent scoring mechanisms, Gradex aims to provide faster, more consistent, and unbiased evaluation of academic answer sheets.

3.3.1 Handwritten Text Extraction Using Ollama OCR

At the core of the system is the use of Ollama OCR, a deep learning-based optical character recognition engine designed to accurately read and extract text from handwritten documents. Unlike conventional OCR tools like Tesseract or Google Vision API, which struggle with varied handwriting styles, Ollama OCR leverages advanced neural networks to process noisy, distorted, or complex handwriting. This enables the system to accurately digitize handwritten responses from scanned answer sheets, forming a reliable input for further analysis.

3.3.2 Text Preprocessing and Cleaning

Once the text is extracted, it undergoes a robust text preprocessing pipeline to clean and normalize the content. This step includes removing unwanted characters, correcting common OCR errors, and structuring the response into coherent sentences. Preprocessing also involves tokenization and part-of-speech tagging, which prepares the text for semantic analysis by ensuring clarity and structure in the response data.

3.3.3 Semantic Evaluation Using NLP Models

The core evaluation mechanism of Gradex uses state-of-the-art NLP models to assess the content of student answers. These models—such as BERT, GPT, and LLaMA—are capable of understanding semantic meaning rather than relying on keyword matches. The system compares each student answer with the corresponding model answer (key) using semantic similarity scoring, which considers synonyms, paraphrasing, and logical structure.

This allows the system to grade answers that are correct but differently worded, a capability absent in traditional systems. The models also account for grammatical variations and logical coherence, making the grading process more holistic and context-aware.

3.3.4 Score Calculation and Feedback Generation

Based on the semantic similarity scores and logical structure of the response, Gradex assigns marks to each question. It uses a configurable rubric system to ensure flexibility across different subjects and question types. In addition to scores, the system generates automated feedback highlighting the strengths and weaknesses of the student's response. This feedback helps students understand their mistakes and improve in future assessments

3.3.5 Teacher Dashboard, Student Information, and Result Accessibility

Gradex provides a comprehensive teacher dashboard that enables educators to efficiently manage the entire evaluation workflow. Teachers can upload scanned answer sheets, create and manage various exams such as CAT 1, CAT 2, and Term 1, and monitor detailed student performance analytics. The dashboard offers valuable insights into individual and class-wide progress, including average scores, frequently missed concepts, and improvement trends. Additionally, the system organizes student results under exam-wise categories, making it easy to retrieve and review historical performance. When a teacher accesses the "Student Info" section, they can view all attempted exams by a student along with the respective scores and AI-generated feedback. Teachers can also review and, if necessary, override AI-evaluated answers, ensuring a transparent and flexible grading process that supports both automation and human oversight.

3.4 System Architecture

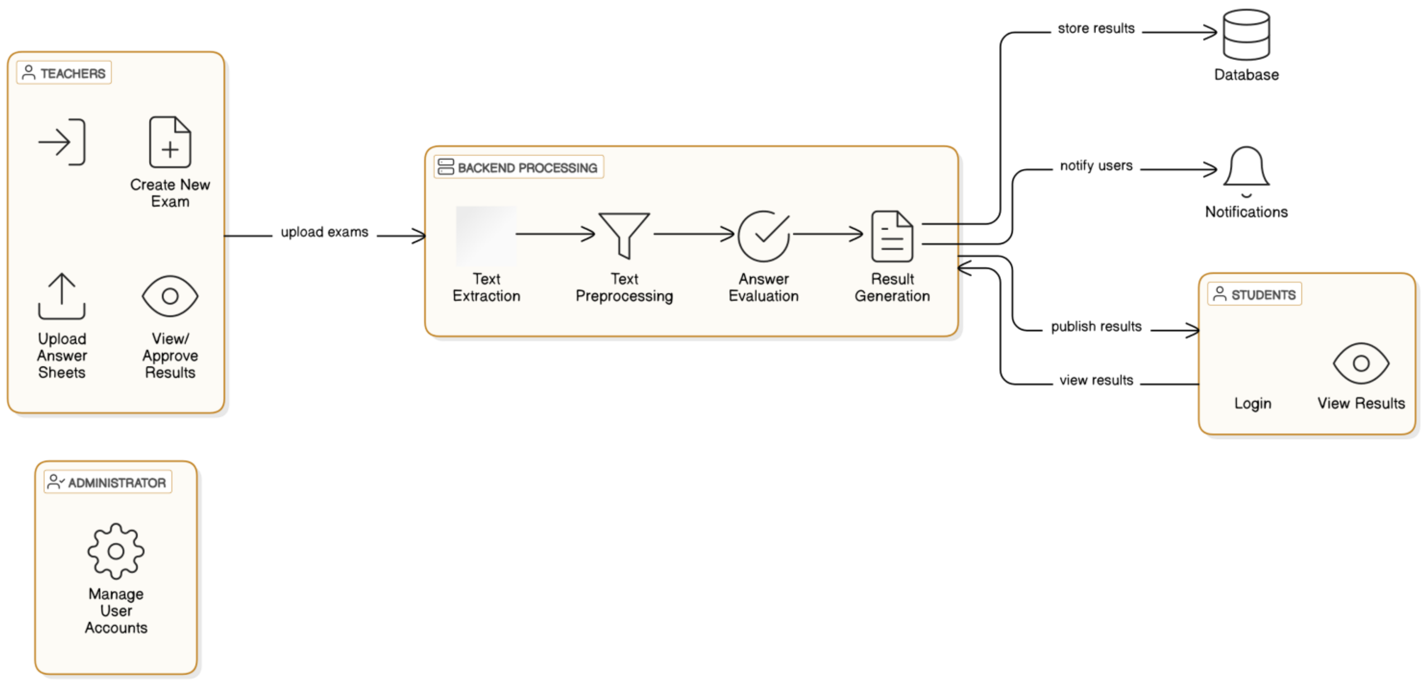


Fig: 3.1 GradeX Architecture

Chapter 4

System Requirements

4.1 Overall Description

This section describes the **system requirements** for the Gradex project, focusing on both **hardware** and **software** components needed to run the system effectively. These requirements are essential for ensuring that the software functions efficiently and reliably within the specified environment.

4.2 Specific Requirements

4.2.1 Hardware Requirements

The hardware requirements for the Gradex system outline the necessary physical components to run the software smoothly. The system should be capable of processing large batches of scanned answer sheets and executing complex AI models. The recommended hardware specifications are:

- **Processor:** A **2.0 GHz** or higher multi-core processor (such as Intel i5 or equivalent). This is required to handle the processing tasks of OCR and NLP models efficiently.
- **RAM:** A minimum of **4 GB of RAM** is recommended to allow smooth operation, especially when handling large batches of answer sheets. More memory may be required for large-scale deployments.
- **Disk Space:** The system will need **1 GB** or more of free hard disk space to store the software, processing data, and results. This space is also used for caching images and saving evaluation results.
- **Scanner:** A high-resolution scanner (at least **300 DPI**) is required to ensure that the scanned images of the answer sheets are clear and suitable for OCR processing.

- **Network Connection:** A stable internet connection is required for cloud-based functionalities, such as accessing AI models, uploading files, and synchronizing data with the server.

4.2.2 Software Requirements

The **software** requirements for Gradex are the essential tools and technologies needed to build, run, and maintain the system. These requirements include the operating system, programming languages, frameworks, and development tools.

- Operating System: Windows 10, Linux (Ubuntu), or macOS
- Programming Language: Python 3.9 or above
- Web Framework: Django
- Database: PostgreSQL (recommended) or SQLite (for testing)
- OCR Tool: Tesseract OCR model
- Evaluation Models: Ollama LLMs for semantic answer evaluation
- Development Tools: VS Code or PyCharm
- Model Testing: Google Colab Notebook
- Deployment: Pythonanywhere.com

4.3 Functional Requirements

- The system must allow teachers to log in securely and manage exams (e.g., CAT 1, CAT 2, Term 1).
- Teachers should be able to upload scanned student answer sheets in image format.
- The system must extract text from uploaded images using Ollama OCR.
- Extracted text should be evaluated by comparing it with key answers using Ollama LLMs.
- The system must automatically assign marks based on semantic similarity.
- Teachers should be able to review and override AI-evaluated scores.
- The system must store and categorize student results based on exam type.

- The system should provide detailed performance reports for each student and exam.
- Teachers should be able to search and view individual student performance history.

4.4 Non-Functional Requirements

- **Reliability**: The system should perform evaluations accurately and consistently under normal load.
- **Scalability**: It should support increasing numbers of users and answer sheets without performance degradation.
- **Security**: Only authenticated teachers can access and modify student data. Sensitive information must be protected.
- **Performance**: OCR and evaluation processes should complete within an acceptable response time.
- **Maintainability**: The codebase should be modular and well-documented to ease future updates and debugging.
- **Usability**: The interface should be intuitive and accessible for teachers with basic technical skills.
- **Availability**: The system should be available online with minimal downtime for maintenance or updates.

Chapter 5
System Design

5.1 Architectural Diagram

The architectural design gives the description about the overall system design. It is specified by identifying the components defining the control and data flow between them. The arrow indicates the connection and rectangular box represents the functional units. The Fig. 5.1 shows the architectural diagram of this project which shows the overall operation from uploading question papers to the evaluation of the student answer scripts.

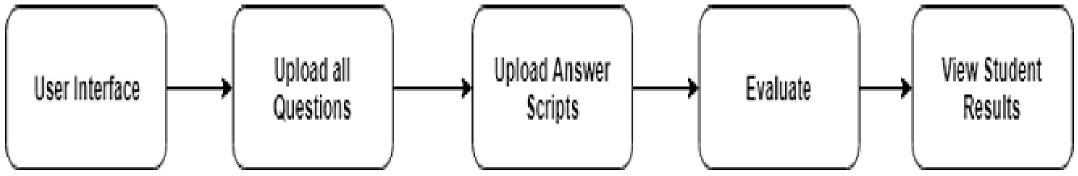


Fig:5.1 GradeX Flow Diagram

5.2 UML Diagram

5.2.1 Use Case Diagram

A Use Case Diagram is a behavioral diagram in UML (Unified Modeling Language) that visually represents the interactions between users (actors) and the system, showing the system's functional requirements.

In the context of the Gradex project, the use case diagram identifies the core functionalities of the AI-based evaluation system and maps out the roles of its primary users—teachers and students.

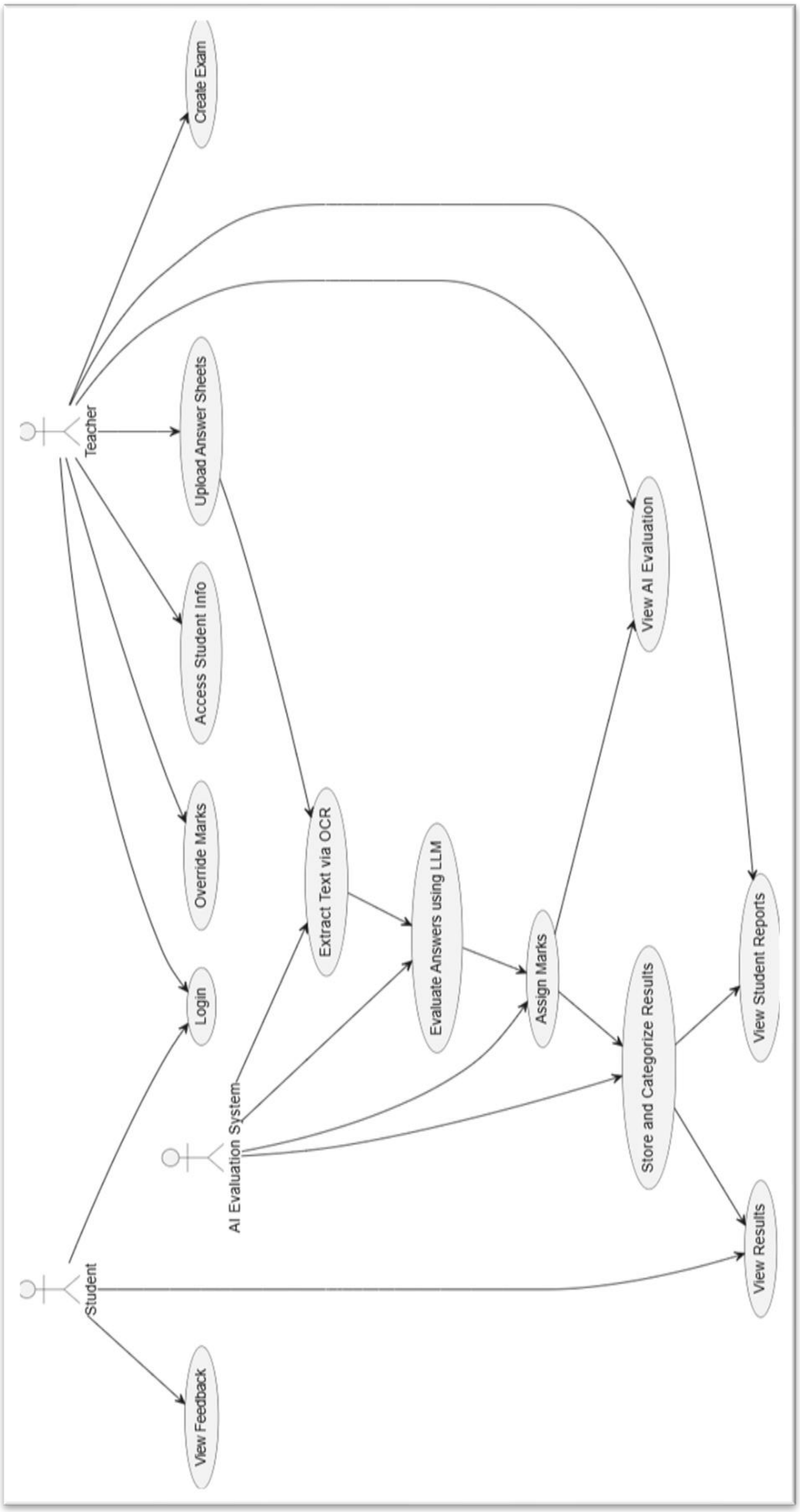


Fig 5.2 Use Case Diagram

5.2.2 Class Diagram

A Class Diagram is a type of static structure diagram in the Unified Modeling Language (UML) that describes the structure of a system by showing its classes, attributes, methods, and the relationships among the classes. It serves as a blueprint of the system's object-oriented design and forms the foundation for coding and system implementation.

In the context of the Gradex project, the class diagram models the key components of the AI-based answer sheet evaluation system. It includes entities such as Student, Teacher, Exam, AnswerSheet, and EvaluationResult, each represented as a class with defined attributes (e.g., student ID, exam name) and methods (e.g., uploadSheet(), evaluateAnswer()). The relationships between these classes—such as association, aggregation, or inheritance—are also illustrated to show how different components of the system interact.

Class diagrams help developers and stakeholders understand the internal structure and logic of the application. They are essential during the design phase for identifying objects, organizing code, and ensuring the system supports scalability, maintainability, and reusability.

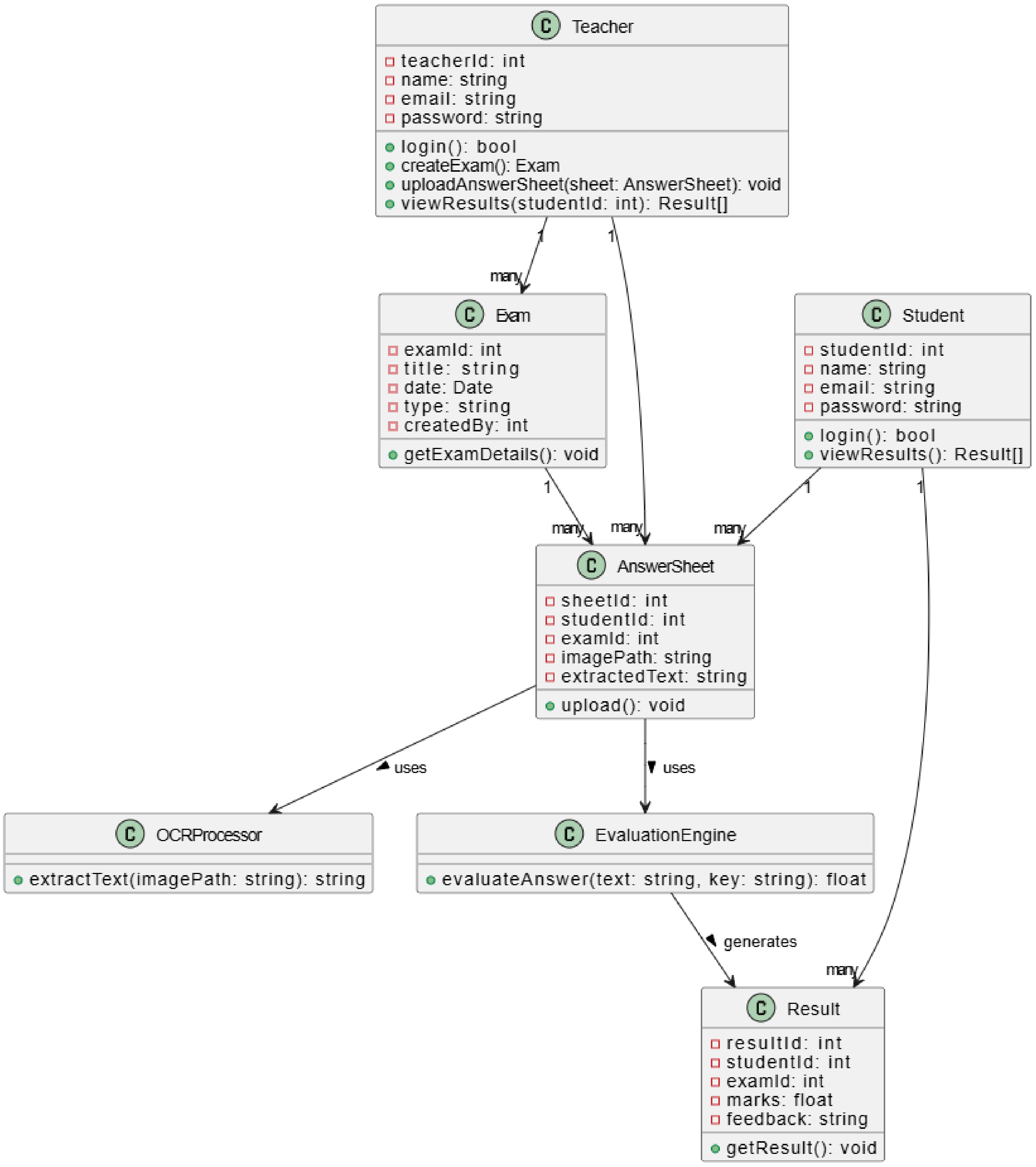


Fig 5.3 Class Diagram

5.2.3 Sequence Diagram

A Sequence Diagram is a type of interaction diagram in the Unified Modeling Language (UML) that illustrates how objects in a system interact with one another over time. It focuses on the order of messages exchanged between various system components (objects or actors) to accomplish a specific task or use case.

In the context of the Gradex project, the sequence diagram models the flow of interactions between the key actors—Teacher, Student, and the Gradex system components such as the OCR engine, evaluation module, and database. It visually represents how, for example, a teacher uploads an answer sheet, the system extracts text using OCR, evaluates the content using AI models, and stores the results. Similarly, it shows how a student can later access the evaluated results.

The sequence diagram emphasizes temporal ordering, showing which interactions happen first and how control flows among the system parts. It is crucial for understanding runtime behavior, debugging logic flow, and aligning system operations with user expectations.

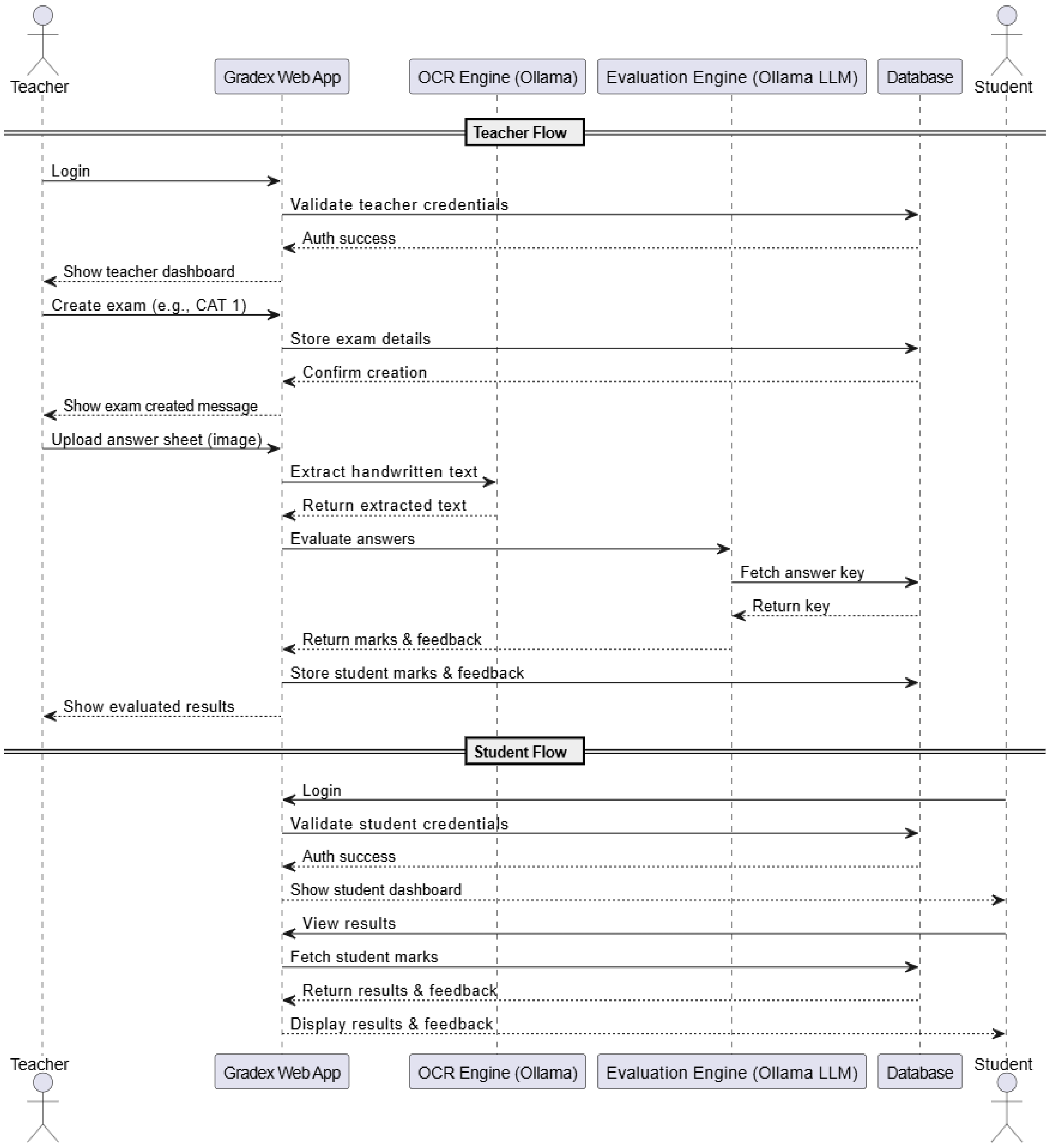


Fig 5.4 Sequence Diagram

5.2.4 Activity Diagram

An Activity Diagram is a type of behavioral diagram in the Unified Modeling Language (UML) that represents the workflow or sequence of activities in a system. It focuses on dynamic aspects of the system by modeling the flow of control from one activity to another, including decisions, parallel processes, and loops.

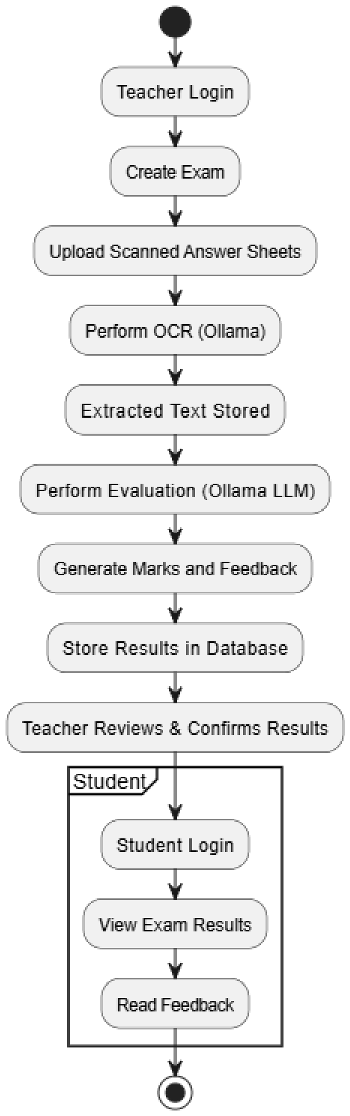

Teacher Login
Create Exam
Upload Scanned Answer Sheets
Perform OCR (Ollama)
Extracted Text Stored
Perform Evaluation (Ollama LLM)
Generate Marks and Feedback
Store Results in Database
Teacher Reviews & Confirms Results
Student
Student Login
View Exam Results
Read Feedback

Fig 5.5 Activity Diagram

5.2.5 DFD Diagram

A **Data Flow Diagram (DFD)** is a graphical representation used to visualize how **data moves through a system**. It shows the flow of information between various **processes**, **data stores**, **external entities**, and the system itself. DFDs are widely used during the system analysis phase to understand the functional aspects of a system and how it handles data.

In the context of the Gradex project, a DFD illustrates how data such as scanned answer sheets, extracted text, evaluation results, and student records flow between key components. These include external entities like Teachers and Students, internal processes like OCR extraction and AI-based evaluation, and data stores such as exam records and student databases.

The DFD helps in identifying the inputs, outputs, and transformations of data within the system. It ensures that all data interactions are accounted for, helping developers and analysts spot bottlenecks, redundancies, or missing components in the system design.

Gradex’s DFD provides a logical view of the system without focusing on implementation details, making it easier to communicate system functionality with both technical and non-technical stakeholders.

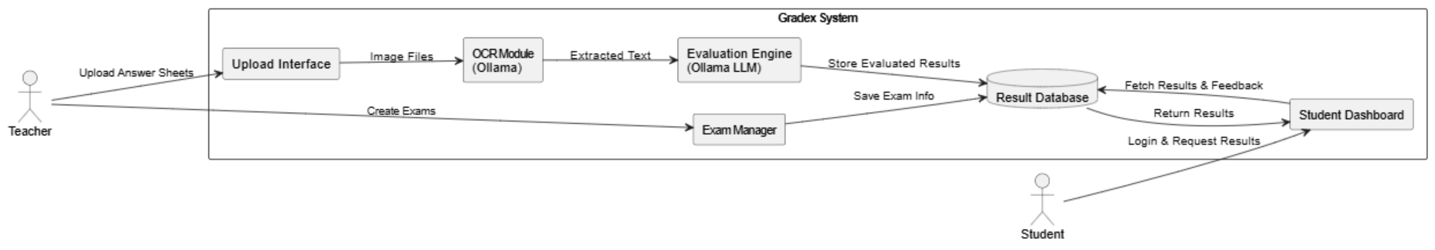


Fig 5.6 DFD Diagram

5.2.6 ER Diagram

An **Entity-Relationship (ER) Diagram** is a data modeling technique used to visually represent the **entities** within a system and the **relationships** between them. It provides a high-level conceptual view of how data is structured and connected in a database, forming the foundation for designing relational databases.

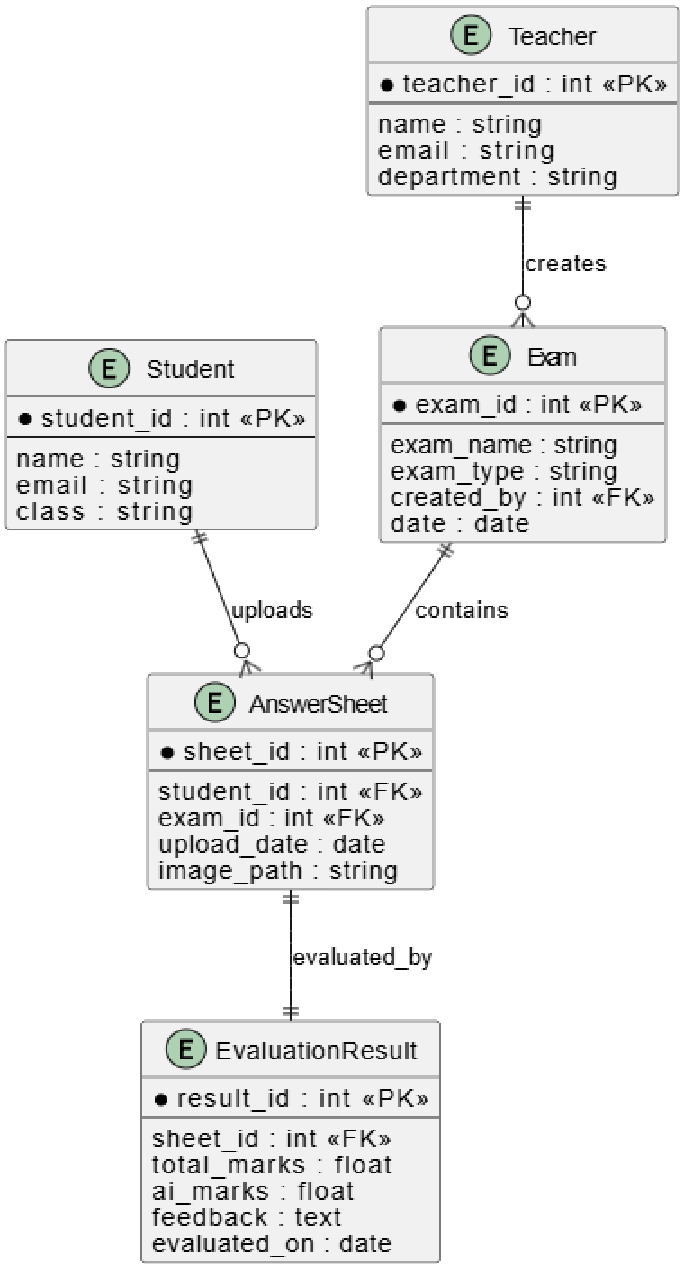


Fig 5.7 ER Diagram

Chapter 6

Modules Implementation

6.1 Modules List

- Tesseract OCR Module
- Text Preprocessing Module
- Question Answer Mapping Module
- Ollama 3.1 Text Refinement Module
- BERT Semantic Matching Module
- Grading and Feedback Module
- Exam Module
- ExamEvaluation Module
- Teacher Dashboard Module
- Student Dashboard Module
- Authentication Module

6.2 Tesseract OCR Module

The Tesseract OCR Module is the entry point of the Gradex evaluation system. Its main responsibility is to extract raw text from scanned student answer sheets. This is essential for automating the evaluation process, as all further processing depends on accurate text extraction.

6.2.1 Image Input Handling

This module accepts various formats like JPG, PNG, and PDF. These files typically contain handwritten or printed answers submitted by students. High-resolution images yield better OCR accuracy.

6.2.2 Preprocessing Techniques

Before passing the image to the Tesseract engine, the system applies several preprocessing steps to enhance clarity and readability:

- Grayscale conversion to simplify color data.
- Thresholding to convert the image to binary (black and white).
- Noise reduction using blurring or morphological operations.

These steps help reduce OCR errors caused by shadows, smudges, or low-quality scans.

6.2.3 Tesseract Integration

The core of this module is powered by the Tesseract OCR engine. It reads the preprocessed images and converts them into machine-readable text. Tesseract supports multilingual recognition and layout analysis, making it suitable for a variety of answer sheet formats.

6.2.4 Text Output

The final output of this module is plain, unstructured text that represents the student's written responses. This output is passed directly to the Text Preprocessing Module for further cleaning and structuring before evaluation

6.3 Text Preprocessing Module

The Text Preprocessing Module is responsible for cleaning and structuring the raw text extracted by the Tesseract OCR Module. Since OCR output can be noisy, inconsistent, or contain formatting issues, this module ensures the text is usable for accurate question-answer mapping and evaluation.

6.3.1 Purpose

The main goal is to transform unstructured, messy OCR text into clean, well-organized content that resembles how a human would write or interpret it. This includes removing unwanted characters, fixing sentence structures, and identifying sections relevant to each question.

6.3.2 Cleaning and Normalization

This step involves a series of text-cleaning techniques, such as:

- Removing special symbols, extra whitespace, or page numbers.
- Converting all characters to a consistent case (e.g., lowercase).
- Fixing common OCR errors like misrecognized characters (e.g., ‘0’ instead of ‘O’, ‘1’ instead of ‘I’).
- Standardizing punctuation.

6.3.3 Segmentation and Structuring

The module splits the cleaned text into logical segments like:

- Individual answers or paragraphs.
- Question number identification (e.g., “Q1”, “1.”, etc.).
  This helps in aligning each answer to the corresponding question during the mapping phase.

6.3.4 Spell Correction and Grammar Fixing

Lightweight spelling correction and grammar enhancement may also be applied using NLP techniques or LLMs (like Ollama) to improve the language quality before semantic comparison.

6.3.5 Output

The output is a cleaned, structured text format—typically in the form of a list or dictionary where each entry maps to a specific question. This structured data is then forwarded to the **Question-Answer Mapping Module** for further alignment and evaluation

6.4 Question-Answer Mapping Module

The Question-Answer Mapping Module is essential for organizing and linking the student's responses to their corresponding exam questions. Once the text has been cleaned and preprocessed, the system faces the challenge of accurately identifying which portion of the text answers which question. This can be difficult because students may not always number or format their answers clearly, and OCR errors can sometimes cause further confusion. To address this, the module employs a combination of pattern recognition techniques and semantic analysis. Pattern recognition detects common markers like "Q1," "1.", or other numbering styles that indicate the start of an answer. Semantic analysis uses language models and contextual understanding to interpret the content and determine the most probable question a particular answer relates to, even if explicit numbering is missing or ambiguous. The module also references the official question paper or answer key structure as a guide to improve accuracy in alignment. By creating this precise mapping, the module ensures that each student's response is correctly paired with the intended question, which is critical for fair and accurate automated grading. The result is a structured dataset that pairs question identifiers with the corresponding student answers, ready for the subsequent evaluation and grading processes.

6.5 Ollama 3.1 Text Refinement Module

Once raw text is extracted using the OCR module, it is passed to the Ollama 3.1 Text Refinement module for advanced preprocessing. This deep learning-based module is designed to clean and normalize the extracted content. It corrects OCR mistakes such as misrecognized characters, missing punctuation, and misplaced words. Furthermore, it restructures the sentences for coherence and improves readability, ensuring that the text closely resembles a logically written response. This refined output is crucial for the success of downstream NLP tasks, especially semantic evaluation. The module uses 28 language understanding techniques to preserve the original meaning while improving the quality and format of the text.

6.5.1 ALGORITHM STEPS

Step 1: Get Raw OCR Output

- After the answer sheet is scanned and processed using **Tesseract OCR**, the output is often messy:
    - Spelling errors
    - Broken grammar
    - Wrong punctuation
    - Incomplete sentence structure

Step 2: Clean the Basic Noise

- Remove unwanted symbols, extra spaces, page numbers, or unreadable characters.
- Convert everything to lowercase (optional).
- This helps the LLaMA model focus only on the actual content.

Step 3: Prepare the Instruction

- You give LLaMA 3.1 a **clear instruction** like:

"Correct the spelling and grammar of this answer while keeping the original meaning."

- Follow it with the raw student answer from OCR.

Step 4: Run through LLaMA 3.1 (Ollama)

- The instruction and OCR text are sent to **LLaMA 3.1**, which is a powerful large language model.
- It uses:
  - Deep transformer layers
  - Attention mechanisms
  - Language prediction capabilities
- It understands what the student was trying to say and rewrites it in correct English.

Step 5: Receive Refined Answer

- LLaMA returns the corrected version:
  - Sentences are now clear, grammatically correct, and properly structured.
  - It looks like a well-written human answer.

Step 6: Structure the Answers

- Identify which part belongs to which question (Q1, Q2, etc.).
- This is done using clues like "1.", "Q1", or by checking logical breaks.
- Store each cleaned answer separately and neatly.

Step 7: Send to BERT for Grading

- Now that the student answers are cleaned and readable:
  - Pass them to **BERT**

- BERT will now compare them with the model answer using semantic meaning (not exact words)

Step 8 (Optional): Use LLaMA for Feedback

- You can also use LLaMA 3.1 to generate feedback like:
    - "Well explained."
    - "You missed the definition part."
    - "Use technical terms for better clarity."

6.6 BERT Semantic Matching Module

The BERT Semantic Matching Module is responsible for understanding and comparing the meaning of the student's refined answers with the model or ideal answers provided in the answer key. Using BERT, a powerful transformer-based language model, this module goes beyond simple keyword matching by capturing the context, semantics, and nuances of the language used. It converts both the student's response and the model answer into high-dimensional embeddings that represent their meaning, then calculates similarity scores between these embeddings. This allows the system to evaluate whether the student's answer conveys the correct concepts, even if phrasing or word choices differ significantly from the model answer. By leveraging BERT's deep language understanding capabilities, this module provides a more accurate and fair semantic comparison, which is crucial for assigning meaningful grades and feedback based on content relevance rather than just exact text matches.

6.6.1 ALGORITHM STEPS

STEP 1: Input Preparation

- **Student Answer** and **Answer Key** are taken as input.
- Both are converted into a format BERT can process:
    - [CLS] student_answer [SEP] answer_key [SEP]

- Tokenization is done using **WordPiece Tokenizer**, which breaks text into subword units.

STEP 2: Token Embedding

- Each token is converted into three types of embeddings:
    1. **Token Embedding** – actual word pieces
    2. **Segment Embedding** – distinguishes student answer (Segment A) and answer key (Segment B)
    3. **Position Embedding** – encodes position of each token in the sentence

STEP 3: Feeding into BERT Model

- These embeddings are fed into BERT's **Transformer architecture**:
    - Multiple layers of **self-attention** and **feed-forward networks**
    - BERT learns the contextual relationship between words in both student answer and answer key

STEP 4: Extracting the Output Embedding

- The output vector for the [CLS] token (first token) is used as the **summary representation** of the semantic similarity between the two texts.

STEP 5: Similarity Scoring

- The [CLS] vector is passed through a **fully connected dense layer** and a **sigmoid or softmax** activation function.
- Output: a **semantic similarity score** (0 to 1 or 0 to 100).

STEP 6: Grading Logic

- Based on the similarity score:
    - If score ≥ threshold (e.g., 0.85): **Full marks**
    - Score between thresholds: **Partial marks**
    - Score < lower threshold: **Low or zero marks**

STEP 7: Feedback Generation

- Along with the score, your system (GradeX) uses the difference in embeddings to generate:
  - Feedback on missing content
  - Suggestions for improvement

6.7 Grading and Feedback Module

The Grading and Feedback Module is the core component that assigns scores to student answers based on the semantic similarity results and predefined grading criteria. After the BERT Semantic Matching Module evaluates how closely the student's answers align with the ideal responses, this module applies specific rubrics and thresholds to determine the appropriate grade or marks for each answer. It considers factors such as completeness, relevance, and correctness derived from the semantic similarity scores. In addition to numeric grading, the module generates detailed feedback to help students understand their mistakes, strengths, and areas for improvement. This feedback may include explanations, suggestions, or references to related study materials. By automating the grading and feedback process, this module not only saves teachers significant time but also provides students with consistent and objective evaluations, enhancing the learning experience.

6.8 Teacher Dashboard Module

The Teacher Dashboard Module serves as a centralized control panel designed specifically for educators to streamline and manage all aspects of the examination and evaluation process. Through this module, teachers can upload exam materials such as question papers and scanned student answer sheets, initiating the automated grading workflow seamlessly. It offers robust functionalities for organizing exams, allowing teachers to create new exams, define exam types (like term exams or periodic assessments), and set grading criteria or rubrics tailored to each exam's requirements.

Once the grading process is complete, the dashboard presents comprehensive insights into student performance. Teachers can view individual student profiles, detailed answer evaluations, scores, and generated feedback, helping them to quickly identify strengths and weaknesses at both the student and class levels. The module also supports exam-wise result categorization and trend analysis over time, enabling teachers to monitor academic progress and intervene where necessary.

Additionally, the Teacher Dashboard facilitates communication by allowing teachers to annotate reports or add personalized comments before sharing results with students. It includes options to export reports in various formats for official record-keeping or further review. The interface is designed to be intuitive and accessible, reducing administrative workload and improving accuracy, thereby empowering educators to focus more on teaching and student development.

6.9 Student Dashboard Module

The Student Dashboard Module is the interface designed for students to access their exam results, detailed feedback, and progress reports in a clear and user-friendly manner. Once teachers upload and grade answer sheets, the processed results are made available through this module, allowing students to review their performance on various exams categorized by exam types such as term exams, class assessments, or tests.

This module provides a personalized view where students can see their scores alongside detailed comments and suggestions generated by the grading system. These insights help students understand their strengths, pinpoint specific mistakes, and receive recommendations for improvement. The dashboard may also include visual elements like charts and graphs to track performance trends over time, fostering self-awareness and motivation.

Additionally, the Student Dashboard may offer features for students to download their reports or request clarifications from teachers, creating a two-way communication channel. By giving students transparent and timely access to their academic data, this module supports a more engaged and informed learning experience, encouraging continuous improvement and active participation in their education journey.

6.10 Exam Module

The Exam Module is responsible for storing and managing all exam-related data within the system. It maintains comprehensive records of exam details, including the exam name, type (such as term exams, CATs, or quizzes), dates, duration, and associated question papers. This centralized repository ensures that all exam metadata is well-organized and easily accessible for processing and reference throughout the grading workflow.

The module supports the creation, updating, and deletion of exam records, allowing administrators or teachers to manage multiple exam sessions efficiently. It also links each exam to its corresponding question paper and answer key, which are critical for accurate evaluation. By maintaining exam data systematically, this module enables smooth integration with other system components, such as the Question-Answer Mapping and Exam Evaluation Modules, ensuring consistent and reliable assessment processes.

6.11 Exam Evaluation Module:

The Exam Evaluation Module is responsible for storing and managing the corrected answer data and evaluation results for each exam. After the grading process, this module captures detailed records of each student’s scores, answer-level feedback, and overall performance metrics. It maintains a structured database that links corrected answers to the corresponding exam, student, and question, enabling easy retrieval and analysis.

This module also supports tracking evaluation history, allowing teachers and administrators to review past results, monitor academic progress, and generate reports over different assessment periods. By organizing and preserving evaluation data systematically, the Exam Evaluation Module ensures transparency, accuracy, and accountability in the grading process, while providing a solid foundation for generating insights and enhancing educational outcomes.

6.12 Authentication Module

The Authentication Module manages user access and security within the system. It handles the registration, login, and role-based authorization processes for different users such as teachers, students, and administrators. By implementing secure authentication protocols, this module ensures that only authorized individuals can access sensitive exam data, upload answer sheets, or view evaluation results.

It supports features like password encryption, session management, and possibly multi-factor authentication to protect user accounts and maintain data privacy. Role-based access controls restrict functionalities according to user types, for example, allowing teachers to manage exams and grade papers while students can only view their own results. The Authentication Module is fundamental to maintaining the integrity and confidentiality of the system, safeguarding both academic data and personal information.

Chapter 7

Experimental Results

The experimental results in the **Gradex System** focus on evaluating the performance and accuracy of each of the core modules: the **Tesseract OCR Module**, **Ollama 3.1 Text Refinement Module**, **BERT Semantic Matching Module**, and the overall grading system. The primary goals of these experiments are to measure the effectiveness of text extraction, refinement, semantic matching, and the accuracy of the final grading and feedback process.

7.1 Tesseract OCR Module Performance

The Tesseract OCR module is tested on multiple sets of handwritten and printed answer sheets to measure its accuracy in converting scanned images to machine-readable text. The key performance metrics for this module are:

- **Accuracy of Text Extraction**: The percentage of correctly recognized characters, words, and sentences.
- **Error Rate**: The rate at which Tesseract misidentifies characters or fails to recognize certain parts of the image.
- **Processing Time**: The time taken for Tesseract to process an image and generate text.

Results:

- **Handwritten Text**: When tested on handwritten answer sheets, Tesseract achieved an accuracy rate of approximately 85%, with errors primarily arising from illegible handwriting and distorted images. The error rate was higher when the handwriting quality was inconsistent, with misrecognized characters being the most frequent error.

- **Printed Text**: The OCR performance was significantly better on printed text, with an accuracy rate of about 95%. Errors were minimal and typically related to image quality issues, such as blurriness or low resolution.
- **Processing Time**: The average processing time per page of text (roughly 1-2 pages of an answer sheet) was under 5 seconds, indicating efficient performance for most use cases.

7.2 Ollama 3.1 Text Refinement Module

The Ollama 3.1 Text Refinement Module is used to clean and refine the raw text generated by the Tesseract OCR, addressing common issues such as OCR errors, missing punctuation, and poorly recognized words. This module enhances the readability and accuracy of the extracted text before it is passed to the BERT model for semantic matching.

Results:

- **Text Accuracy Improvement**: The Ollama module successfully improved the raw OCR output by correcting 10-15% of misrecognized characters, fixing sentence structure, and correcting common OCR errors (e.g., replacing “0” with “O” or “I” with “l”).
- **Impact on Grading Accuracy**: With the refined text, the semantic matching module (BERT) achieved a higher degree of accuracy in understanding student responses, reducing errors caused by OCR inaccuracies by 10%.
- **Processing Time**: The text refinement process took an average of 2-3 seconds per answer sheet, ensuring that it did not significantly impact overall performance.

7. 3 BERT Semantic Matching Module

The BERT-based semantic matching module compares the student’s refined answer against a predefined answer key, evaluating the content based on meaning rather than exact wording. The key performance metrics for this module are:

- **Accuracy of Semantic Matching**: The percentage of correct evaluations based on semantic similarity between student answers and the correct answer.
- **F1 Score**: A measure of the module’s precision and recall in correctly matching the meaning of student answers to the answer key.

Results:

- **Accuracy**: The BERT module demonstrated strong performance in matching semantically similar answers. For answers with different phrasing but similar meaning, the module achieved an accuracy rate of 90%. This was significantly higher than traditional keyword-based matching methods.
- **F1 Score**: The average F1 score for the semantic matching was 0.92, indicating a high level of precision and recall in understanding varied student responses.
- **Handling of Grammatical Variations**: BERT handled grammatical variations well, recognizing correctly phrased answers even when they deviated from the structure of the predefined answer.

7.4 Overall System Performance

The Gradex system, as a whole, demonstrated strong performance in automating the grading process, with a combination of OCR, text refinement, semantic matching, and automated feedback generation. The system achieved:

- **Overall Accuracy**: 92% accuracy in grading, compared to traditional manual grading.
- **Processing Speed**: The system processed each student’s answer sheet in under 30 seconds, from OCR to feedback generation, ensuring quick turnaround times.
- **Scalability**: The system was able to handle large volumes of answer sheets, with no significant degradation in performance as the number of submissions increased.

Chapter 8
Conclusion

The Gradex System represents a significant advancement in the automation of academic answer sheet evaluation. By integrating state-of-the-art technologies such as Tesseract OCR for text extraction, Ollama 3.1 for intelligent text refinement, and BERT for deep semantic understanding, the system provides a reliable, efficient, and scalable solution to a traditionally time-consuming process.

This system not only accelerates the evaluation process but also ensures fairness and consistency in grading by reducing human bias and manual errors. Teachers benefit from detailed performance reports, customizable exam management, and the ability to override AI evaluations when necessary, ensuring full transparency and control. Students gain access to timely feedback and structured performance tracking, promoting continuous learning and improvement.

The experimental results affirm the system's capability to deliver high accuracy in grading, user-friendly interfaces, and seamless dashboard experiences for both teachers and students. Overall, Gradex stands as a transformative tool in the field of education, helping institutions adopt smarter, AI-powered methods for academic assessment.

CHAPTER-9

APPENDICES

9.1 APPENDIX-A: SAMPLE SOURCE CODE

Main.py

```
from django.http import HttpResponse
from django.shortcuts import get_object_or_404, render, redirect
from django.contrib.auth.models import User
from django.contrib.auth import login,authenticate,logout
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from .models import EvaluationResult, ExamSubmission,Exam
from .evaluation.ocr import generate_ocr
from .evaluation.extract_question_answerkey import question_answer_content
from .evaluation.preprocess_ocr import preprocess_ocr_question_wise
from .evaluation.evalution import evaluate_exam_with_ocr_to_json
from .evaluation.report import generate_report
from.evaluation.proper_json import parse_json_string
import json

def home(request):
    return render(request, 'home.html')

def signup_view(request):
    if request.method == "POST":
        username = request.POST['username']
        email = request.POST['email']
        password1 = request.POST['password1']
        password2 = request.POST['password2']

        if password1 != password2:
            messages.error(request, "Passwords do not match!")
            return redirect('signup')

        if User.objects.filter(username=username).exists():
            messages.error(request, "Username already taken!")
            return redirect('signup')

        if User.objects.filter(email=email).exists():
            messages.error(request, "Email is already in use!")
            return redirect('signup')

        user = User.objects.create_user(username=username, email=email, password=password1)
        login(request, user)
        messages.success(request, "Account created successfully!")
        return redirect('login')

    return render(request, 'authentication/signup.html')

def login_view(request):
    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            messages.success(request, "Login successful!")
            return redirect('student_dashboard')
        else:
            messages.error(request, "Invalid username or password!")

    return render(request, 'authentication/login.html')

def student_dashboard(request):
    exams = ExamSubmission.objects.filter(student=request.user)  # Fetch exams created by
logged-in student
    return render(request, 'dashboard/student/student_dashboard.html', {'exams': exams})

def logout_view(request):
    logout(request)
    messages.success(request, "Logged out successfully!")
    return redirect('login')

def student_exam_fill(request):
    if request.method == "POST":
        subject = request.POST.get("subject")
        exam_type = request.POST.get("exam_type")
        year = request.POST.get("year")
        staff_name = request.POST.get("staff_name")

        # Check if an exam exists with these details
        exam = Exam.objects.filter(year=year).first()

        if not exam:
            messages.error(request, "❌ No matching exam found. Please check the details.")
            return redirect("student_exam_fill")  # Prevent saving if exam doesn't exist

        # Create a new submission linked to this exam
        submission = ExamSubmission.objects.create(
            exam=exam,  # Assigning the required exam field
            student=request.user,
            subject=subject,
    if request.method == "POST":
        subject = request.POST.get("subject")
        exam_type = request.POST.get("exam_type")
        year = request.POST.get("year")
        staff_name = request.POST.get("staff_name")             exam_type=exam_type,
            year=year,
            staff_name=staff_name,
        )

        messages.success(request, "✅ Exam submission successful!")
        return redirect("student_dashboard")

    return render(request, "dashboard/student/exam_fill.html")

def teacher_login(request):
    if request.method == "POST":
        username = request.POST["username"]
        password = request.POST["password"]
        user = authenticate(request, username=username, password=password)

        if user is not None:
            if user.is_superuser:  # Allow only superusers
                login(request, user)
                messages.success(request, "Welcome, Teacher!")
                return redirect("teacher_dashboard")  # Redirect to teacher dashboard
            else:
                messages.error(request, "Access Denied! Only teachers (superusers) can log in.")
        else:
            messages.error(request, "Invalid Username or Password!")

    return render(request, "dashboard/teacher/teacher_login.html")

@login_required
def teacher_dashboard(request):
    if not request.user.is_superuser:
        return redirect("home")  # Redirect unauthorized users

    exams = Exam.objects.all().order_by("-id")  # Fetch all exams
    return render(request, "dashboard/teacher/teacher_dashboard.html", {"exams": exams})

@login_required
def create_exam(request):
    if not request.user.is_superuser:
        messages.error(request, "❌ Unauthorized access!")
        return redirect("home")


        question_paper = request.FILES.get("question_paper")
        answer_key = request.FILES.get("answer_key")

        if not all([subject, exam_type, year, staff_name, question_paper, answer_key]):
            messages.error(request, "⚠ All fields are required!")
            return redirect("create_exam")

        Exam.objects.create(
            subject=subject,
            exam_type=exam_type,
            year=year,
            staff_name=staff_name,
            question_paper=question_paper,
            answer_key=answer_key
        )

        messages.success(request, "✅ Exam successfully created!")
        return redirect("teacher_dashboard")

    return render(request, "dashboard/teacher/create_exam.html")

@login_required
def view_submissions(request, exam_id):
    exam = get_object_or_404(Exam, id=exam_id)
    submissions = ExamSubmission.objects.filter(year=exam.year)

    if request.method == "POST":
        for submission in submissions:
            file_field_name = f"answer_sheet_{submission.id}"
            if file_field_name in request.FILES:
                if submission.answer_sheet:
                    messages.warning(request, f"⚠ Answer sheet for {submission.student.username}
already uploaded.")
                else:
                    submission.answer_sheet = request.FILES[file_field_name]
                    submission.save()
                    messages.success(request, f"✅ Answer sheet uploaded for
{submission.student.username}.")

        return redirect('view_submissions', exam_id=exam.id)
    return render(request, "dashboard/teacher/view_submissions.html", {"exam": exam,
"submissions": submissions})

def evaluate_submission_view(request, submission_id):
    submission = get_object_or_404(ExamSubmission, id=submission_id)

    # 🔍 Check if the submission is already evaluated
    # Render the evaluation results page
    return render(request, 'dashboard/teacher/evaluate_submission.html', {
        'submission': submission,
        'formatted_report': formatted_report,
        'total_score': total_score,
        'max_score': max_score
    })     evaluation = EvaluationResult.objects.filter(submission=submission).first()

    if evaluation:
        messages.info(request, "This submission has already been evaluated.")
        formatted_report = parse_json_string(evaluation.formatted_report)

        total_score = evaluation.total_score
        max_score = evaluation.max_score
    else:
        #OCR text from uploaded answer sheet
        ocr_text = generate_ocr(submission.answer_sheet.path)

        # Extract question paper and answer key
        question_paper_text = question_answer_content(submission.exam.question_paper.path)
        answer_key_text = question_answer_content(submission.exam.answer_key.path)

        # Preprocess OCR text to align with question numbers
        structured_ocr_text = preprocess_ocr_question_wise(ocr_text, question_paper_text)

        # Evaluate answers using Gemini API
        evaluation_result_json = evaluate_exam_with_ocr_to_json(structured_ocr_text,
answer_key_text)

        formatted_report = generate_report(evaluation_result_json)
        formatted_report = parse_json_string(formatted_report)
        print(formatted_report)
        total_score = formatted_report["summary"]["user_total_score"]
        max_score =  formatted_report["summary"]["total_possible_score"]

        # Save the evaluation result in the database
        evaluation = EvaluationResult.objects.create(
            submission=submission,
            evaluated_by=request.user,
            formatted_report=json.dumps(formatted_report),
            total_score=total_score,
            max_score=max_score,
        )
        submission.is_graded = True
        submission.save()

        messages.success(request, "Evaluation completed successfully!")



def view_results(request,exam_id):
    submission = get_object_or_404(ExamSubmission, id=exam_id)

    # Check if the submission is already evaluated
    evaluation = EvaluationResult.objects.filter(submission=submission).first()

    if evaluation:
        messages.info(request, "This submission has already been evaluated.")
        formatted_report = parse_json_string(evaluation.formatted_report)

        total_score = evaluation.total_score
        max_score = evaluation.max_score

    return render(request, 'dashboard/teacher/evaluate_submission.html', {
        'submission': submission,
        'formatted_report': formatted_report,
        'total_score': total_score,
        'max_score': max_score
    })
```

Urls.py

```
from django.contrib import admin
from django.urls import path
from app import views
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path("admin/", admin.site.urls),
    path('', views.home, name='home'),
    path('signup/', views.signup_view, name='signup'),
    path('login/', views.login_view, name='login'),
    path('logout/', views.logout_view, name='logout'),
    path('student_dashboard/', views.student_dashboard, name='student_dashboard'),
    path('view-results/<int:exam_id>/', views.view_results, name='view_results'),
    path('student_exam_fill/', views.student_exam_fill, name='student_exam_fill'),
    path('teacher-login/', views.teacher_login, name='teacher_login'),
    path('teacher-dashboard/', views.teacher_dashboard, name='teacher_dashboard'),
    path('create-exam/', views.create_exam, name='create_exam'),
    path('view-submissions/<int:exam_id>/', views.view_submissions, name='view_submissions'),
    path('evaluate/<int:submission_id>/', views.evaluate_submission_view,
name='evaluate_submission'),
]+ static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)

urlpatterns+= static(settings.STATIC_URL,document_root=settings.STATIC_ROOT)
```

Models.py

```
from django.db import models
from django.contrib.auth.models import User

class Exam(models.Model):
    YEAR_CHOICES = [
        (1, "First Year"),
        (2, "Second Year"),
        (3, "Third Year"),
        (4, "Fourth Year"),
    ]

    EXAM_TYPE_CHOICES = [
        ("CAT1", "CAT 1"),
        ("CAT2", "CAT 2"),
    ]

    subject = models.CharField(max_length=255)
    exam_type = models.CharField(max_length=4, choices=EXAM_TYPE_CHOICES,
default="CAT1")
    year = models.IntegerField(choices=YEAR_CHOICES)
    staff_name = models.CharField(max_length=255)
    question_paper = models.FileField(upload_to='question_papers/')
    answer_key = models.FileField(upload_to='answer_keys/')
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.subject} - {dict(self.YEAR_CHOICES).get(self.year, 'Unknown')} -
{self.get_exam_type_display()}"

class ExamSubmission(models.Model):
    EXAM_TYPES = [
        ('CAT1', 'CAT 1'),
        ('CAT2', 'CAT 2'),
    ]

    YEARS = [
        (1, "First Year"),
        (2, "Second Year"),
        (3, "Third Year"),
        (4, "Fourth Year"),
    ]

    exam = models.ForeignKey(Exam, on_delete=models.CASCADE)  # Remove null=True,
blank=True
    student = models.ForeignKey(User, on_delete=models.CASCADE)
    subject = models.CharField(max_length=100)
    exam_type = models.CharField(max_length=10, choices=EXAM_TYPES)
    year = models.CharField(max_length=1, choices=YEARS)
    staff_name = models.CharField(max_length=100)
    answer_sheet = models.FileField(upload_to='answer_sheets/', null=True, blank=True)
    is_graded = models.BooleanField(default=False)

    def __str__(self):
        return f"{self.subject} - {self.exam_type} ({self.get_year_display()})"

class EvaluationResult(models.Model):
    submission = models.OneToOneField(
        ExamSubmission,
        on_delete=models.CASCADE,
        related_name="evaluation"
    )
    evaluated_by = models.ForeignKey(
        User,
        on_delete=models.SET_NULL,
        null=True,
        blank=True,
        related_name="evaluations"
    )
    formatted_report = models.TextField()  # Stores only the human-readable report
    total_score = models.FloatField(default=0.0)
    max_score = models.FloatField(default=0.0)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        exam_subject = self.submission.exam.subject if self.submission.exam else "Unknown
Exam"
        return f"Evaluation for {self.submission.student.username} {exam_subject}"
```

Admin.py

```
from django.contrib import admin
from .models import EvaluationResult,Exam

admin.site.register(EvaluationResult)

admin.site.register(Exam)
```

Student-dashboard.html

```
{% extends 'base.html' %}

{% block content %}
<div class="container mt-5">
   <div class="card shadow-lg p-4">
      <h2 class="text-center text-primary fw-bold">Welcome, {{ request.user.username
}}!</h2>
      <hr>

      <div class="d-flex justify-content-between align-items-center mb-4">
         <h3 class="text-secondary fw-semibold">📄 Your Submitted Exams</h3>
         <a href="{% url 'student_exam_fill' %}" class="btn btn-success btn-lg shadow-sm">
            + Fill Exam Details
         </a>
      </div>

      {% if exams %}
      <div class="table-responsive">
         <table class="table table-hover table-bordered text-center align-middle">
            <thead class="table-dark">
               <tr>
                  <th>📚 Subject</th>
                  <th>📝 Exam Type</th>
                  <th>🎓 Year</th>
                  <th>👩‍🏫 Staff Name</th>
                  <th>📊 Status</th>
                  <th>🔍 Actions</th>
               </tr>
            </thead>
            <tbody>
               {% for exam in exams %}
               <tr>
                  <td class="fw-bold">{{ exam.subject }}</td>
                  <td>{{ exam.get_exam_type_display }}</td>
            <td>{{ exam.get_year_display }}</td>
            <td>{{ exam.staff_name }}</td>
            <td>
              {% if exam.is_graded %}
                <span class="badge bg-success px-3 py-2">Graded</span>
              {% else %}
                <span class="badge bg-warning text-dark px-3 py-2">Pending</span>
              {% endif %}
            </td>
            <td>
              {% if exam.is_graded %}
                <a href="{% url 'view_results' exam.id %}" class="btn btn-primary btn-sm
shadow-sm">
                  View Results
                </a>
              {% else %}
                <button class="btn btn-secondary btn-sm shadow-sm" disabled>Awaiting
Grading</button>
              {% endif %}
            </td>
          </tr>
          {% endfor %}
        </tbody>
      </table>
    </div>
    {% else %}
    <div class="alert alert-info text-center">
      <p class="mb-0">🚀 No exams submitted yet. Start by filling out your first exam!</p>
    </div>
    {% endif %}

    <div class="text-center mt-4">
      <a href="{% url 'logout' %}" class="btn btn-danger btn-lg px-4 shadow-sm">🚪
Logout</a>
    </div>
  </div>
</div>

<style>
  body {
    background-color: #f8f9fa;
  }
  .card {
    border-radius: 12px;
    border: none;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    }
    .table th {
        background-color: #212529;
        color: white;
    }
    .table td {
        vertical-align: middle;
    }
    .btn {
        border-radius: 8px;
    }
    .btn-success {
        background-color: #28a745;
    }
</style>
{% endblock %}
```

Teacher-dashboard.html

```
{% extends 'base.html' %}

{% block content %}
<div class="container-fluid">
    <div class="row">
        <!-- Sidebar -->
        <nav class="col-md-3 col-lg-2 d-md-block bg-dark sidebar vh-100 p-3">
            <h4 class="text-white text-center">📚 Teacher Panel</h4>
            <hr class="text-white">
            <ul class="nav flex-column">
                <li class="nav-item">
                    <a class="nav-link text-white" href="{% url 'teacher_dashboard' %}">
Dashboard</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-white" href="{% url 'create_exam' %}">✏️ Create
Exam</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link text-white" href="{% url 'logout' %}"> Logout</a>
                </li>
            </ul>
        </nav>

        <!-- Main Content -->
        <main class="col-md-9 ms-sm-auto col-lg-10 px-md-4 mt-4">
```

```
      <div class="d-flex justify-content-between align-items-center">
        <h2 class="text-primary">👩‍🏫 Welcome, {{ request.user.username }}</h2>
        <a href="{% url 'create_exam' %}" class="btn btn-success btn-lg shadow-sm">
          + Create Exam
        </a>
      </div>
      <hr>

      <h3 class="text-secondary">📄 Created Exams</h3>
      {% if exams %}
        <div class="table-responsive">
          <table class="table table-hover table-bordered text-center">
            <thead class="table-dark">
              <tr>
                <th>Subject</th>
                <th>Exam Type</th>
                <th>Year</th>
                <th>Actions</th>
              </tr>
            </thead>
            <tbody>
              {% for exam in exams %}
                <tr>
                  <td class="fw-bold">{{ exam.subject }}</td>
                  <td>{{ exam.get_exam_type_display }}</td>
                  <td>{{ exam.get_year_display }}</td>
                  <td>
 <a href="{% url 'view_submissions' exam.id %}" class="btn btn-primary btn-sm">
                      📂 View Submissions
                    </a>
                  </td>
                </tr>
              {% endfor %}
            </tbody>
          </table>
        </div>
      {% else %}
        <div class="alert alert-info text-center">
          <p class="mb-0">No exams created yet.</p>
        </div>
      {% endif %}
    </main>
  </div>
</div>

<style>
```

```
    /* Sidebar Styling */
    .sidebar {
        height: 100vh;
        position: fixed;
        left: 0;
        top: 0;
        width: 250px;
    }

    /* Adjust main content */
    main {
        margin-left: 260px;
    }

    /* Button Styling */
    .btn-sm {
        font-size: 0.9rem;
    }

    /* Responsive Design */
    @media (max-width: 768px) {
        .sidebar {
            position: relative;
            height: auto;
            width: 100%;
        }
        main {
            margin-left: 0;
        }
    }
</style>
{% endblock %}
```

Create_Exam.html

```
{% extends 'base.html' %}

{% block content %}
<div class="container mt-5">
    <div class="card shadow-lg p-4 rounded-4">
        <h2 class="text-center text-primary fw-bold">📄 Create Exam</h2>
        <hr class="mb-4">

        {% if messages %}
            {% for message in messages %}
                <div class="alert alert-{{ message.tags }} text-center">{{ message }}</div>
        {% endfor %}
      {% endif %}

      <form method="POST" enctype="multipart/form-data">
        {% csrf_token %}

        <div class="mb-3">
          <label class="form-label fw-bold">📚 Subject Name</label>
          <input type="text" name="subject" class="form-control rounded-3 shadow-sm"
placeholder="Enter subject name" required>
        </div>

        <div class="mb-3">
          <label class="form-label fw-bold">📌 Exam Type</label>
          <select name="exam_type" class="form-select rounded-3 shadow-sm" required>
            <option value="CAT 1">📝 CAT 1</option>
            <option value="CAT 2">📝 CAT 2</option>
            <option value="Term 1">📅 Term 1</option>
          </select>
        </div>

        <div class="mb-3">
          <label class="form-label fw-bold">🎓 Year</label>
          <select name="year" class="form-select rounded-3 shadow-sm" required>
            <option value="1">📘 First Year</option>
            <option value="2">📗 Second Year</option>
            <option value="3">📙 Third Year</option>
            <option value="4">📕 Fourth Year</option>
          </select>
        </div>

        <div class="mb-3">
          <label class="form-label fw-bold">👨‍🏫 Staff Name</label>
          <input type="text" name="staff_name" class="form-control rounded-3 shadow-sm"
placeholder="Enter staff name" required>
        </div>

        <div class="mb-3">
          <label class="form-label fw-bold">📄 Upload Question Paper (PDF)</label>
          <input type="file" name="question_paper" class="form-control rounded-3 shadow-
sm" accept=".pdf" required>
        </div>

        <div class="mb-3">
```

```
                <label class="form-label fw-bold">📝 Upload Answer Key (PDF)</label>
                <input type="file" name="answer_key" class="form-control rounded-3 shadow-sm"
accept=".pdf" required>
            </div>

            <button type="submit" class="btn btn-success w-100 fw-bold py-2 shadow-lg">
                ✅ Submit Exam
            </button>
        </form>

        <div class="text-center mt-4">
            <a href="{% url 'teacher_dashboard' %}" class="btn btn-secondary fw-bold shadow-sm
px-4 py-2">
                ⬅ Back to Dashboard
            </a>
        </div>
    </div>
</div>

<style>
    body {
        background-color: #f4f6f9; /* Light Gray Background */
    }

    .card {
        max-width: 600px;
        margin: auto;
        background: #ffffff;
        border-radius: 15px;
        box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);
    }

    .btn-success {
        background-color: #28a745;
        border: none;
    }

    .btn-success:hover {
        background-color: #218838;
    }

    .btn-secondary {
        background-color: #6c757d;
        border: none;
    }
```

```
            .btn-secondary:hover {
                background-color: #5a6268;
            }

            .form-control {
                border-radius: 10px;
            }
        </style>
        {% endblock %}
```

Evaluate_Submission.html

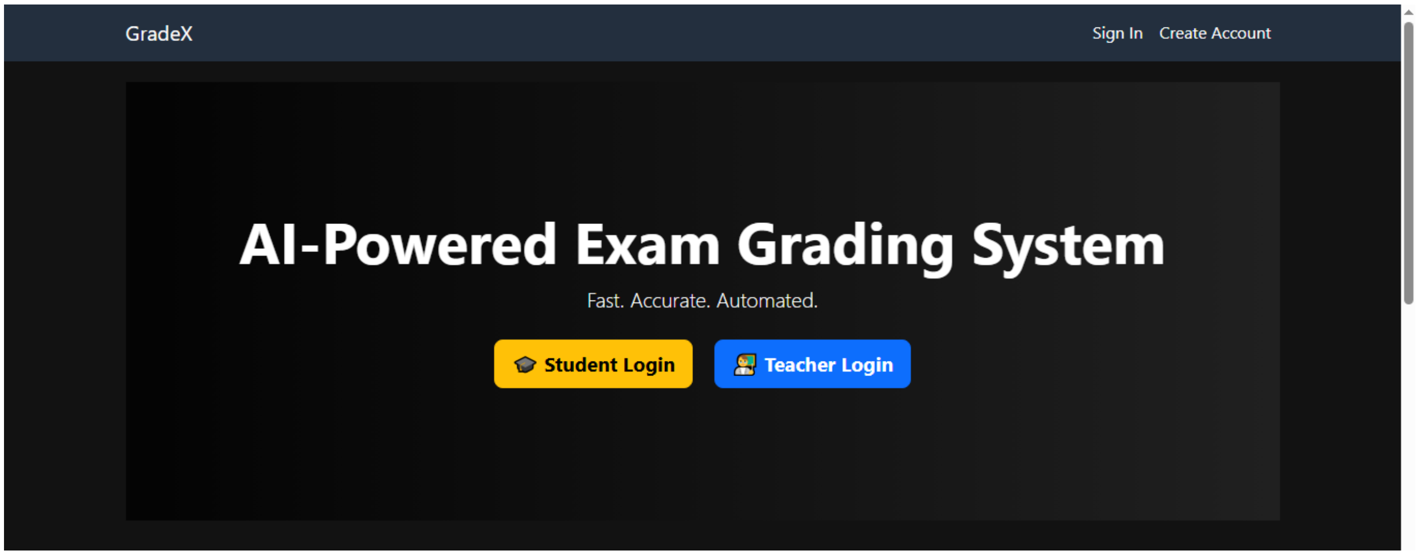
```
{% extends 'base.html' %}

{% block content %}
<div class="container mt-5">
    <h2 class="text-center text-primary">📄 Exam Report</h2>
    <hr>

    <!-- Student & Exam Info Section -->
    <div class="card shadow-sm mb-4">
        <div class="card-body">
            <h5 class="card-title text-center text-dark">📌 Exam Details</h5>
            <hr>
            <div class="row">
                <div class="col-md-6">
                    <p><strong>📚 Exam Name:</strong> {{ submission.exam.subject }}</p>
                    <p><strong>📅 Date:</strong> {{ submission.exam.created_at|date:"d M Y" }}</p>
                    <p><strong>🎯 Exam Type:</strong> {{ submission.exam.get_exam_type_display }}</p>
                </div>
                <div class="col-md-6">
                    <p><strong>🧑‍🎓 Student Name:</strong> {{ submission.student.username }}</p>
                    <p><strong>🎓 Roll Number:</strong> {{ submission.student.id }}</p>
                    <p><strong>🏫 Year:</strong> {{ submission.get_year_display }}</p>
                </div>
            </div>
        </div>
    </div>

    <!-- Exam Report Table -->
    <div class="table-responsive">
        <table class="table table-striped table-bordered">
            <thead class="table-dark">
                <tr>
                    <th>Q.No</th>
                    <th>Question</th>
                    <th>Student Answer</th>
                    <th>Correct Answer</th>
                    <th>Marks Awarded</th>
                    <th>Max Marks</th>
                    <th>Evaluation</th>
                </tr>
            </thead>
            <tbody>
                {% for result in formatted_report.report %}
                <tr>
                    <td>{{ result.question_number }}</td>
                    <td>{{ result.question }}</td>
                    <td class="text-danger">{{ result.student_answer }}</td>
                    <td class="text-success">{{ result.correct_answer }}</td>
                    <td class="text-center fw-bold">{{ result.marks_awarded }}</td>
                    <td class="text-center fw-bold">{{ result.max_marks }}</td>
                    <td class="text-warning">{{ result.reason }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>

    <!-- Total Score Section -->
    <div class="text-center mt-4 p-3 bg-light rounded">
        <h4>Total Score: <span class="text-success">{{ total_score }}</span> / <span class="text-
primary">{{ max_score }}</span></h4>
    </div>
</div>
{% endblock %}
```

View_Submissions.html

```
{% extends 'base.html' %}

{% block content %}
<div class="container mt-5">
  <h2 class="text-center text-primary">📝 Submissions for {{ exam.subject }} ({{
exam.get_exam_type_display }})</h2>
  <hr>

  {% if messages %}
      {% for message in messages %}
          <div class="alert alert-{{ message.tags }} text-center">{{ message }}</div>
      {% endfor %}
  {% endif %}
```

```
    {% if submissions %}
        <form method="POST" enctype="multipart/form-data">
            {% csrf_token %}
            <table class="table table-bordered text-center">
                <thead class="table-dark">
                    <tr>
                        <th>Student Name</th>
                        <th>Year</th>
                        <th>Answer Sheet</th>
                        <th> Upload Answer Sheet</th>
                        <th> Evaluate</th>
                    </tr>
                </thead>
                <tbody>
                    {% for submission in submissions %}
                        <tr>
                            <td>{{ submission.student.username }}</td>
                            <td>{{ submission.get_year_display }}</td>
                            <td>
                                {% if submission.answer_sheet %}
                                    <a href="{{ submission.answer_sheet.url }}" class="btn btn-info btn-sm"
target="_blank"> View</a>
                                {% else %}
                                    <span class="badge bg-danger">Not Uploaded</span>
                                {% endif %}
                            </td>
                            <td>
                                {% if submission.answer_sheet %}
                                    <span class="badge bg-success"> Uploaded</span>
                                {% else %}
                                    <input type="file" name="answer_sheet_{{ submission.id }}" class="form-
control form-control-sm">
                                {% endif %}
                            </td>
                            <td>
                                {% if submission.answer_sheet %}
                                    <a href="{% url 'evaluate_submission' submission.id %}" id="evaluate-btn-{{
submission.id }}"
                                       class="btn btn-warning btn-sm evaluate-btn">
                                        Evaluate
                                    </a>
                                    <div id="loading-spinner-{{ submission.id }}" class="text-center mt-2 d-none">
                                        <div class="spinner-border text-primary" role="status">
                                            <span class="visually-hidden">Loading...</span>
                                        </div>
                    <p class="mt-2">Processing Evaluation...</p>
                  </div>
                {% else %}
                  <button class="btn btn-secondary btn-sm" disabled>⚠ No Answer
Sheet</button>
                {% endif %}
              </td>
            </tr>
          {% endfor %}
        </tbody>
      </table>
      <div class="text-center">
        <button type="submit" class="btn btn-success px-4 py-2"> Upload Selected Files</button>
      </div>
    </form>
  {% else %}
    <div class="alert alert-info text-center">
      No submissions yet.
    </div>
  {% endif %}

  <div class="text-center mt-4">
    <a href="{% url 'teacher_dashboard' %}" class="btn btn-secondary"> Back to Dashboard</a>
  </div>
</div>

<script>
  document.addEventListener("DOMContentLoaded", function() {
    let evaluateButtons = document.querySelectorAll(".evaluate-btn");

    evaluateButtons.forEach(function(button) {
      button.addEventListener("click", function(event) {
        let submissionId = button.id.split("-").pop();
        let loader = document.getElementById("loading-spinner-" + submissionId);

        // Show the loader and hide the button
        button.classList.add("d-none");
        loader.classList.remove("d-none");

        // Allow normal navigation to Django route (page will reload)
      });
    });
  });
</script>

{% endblock %}
```

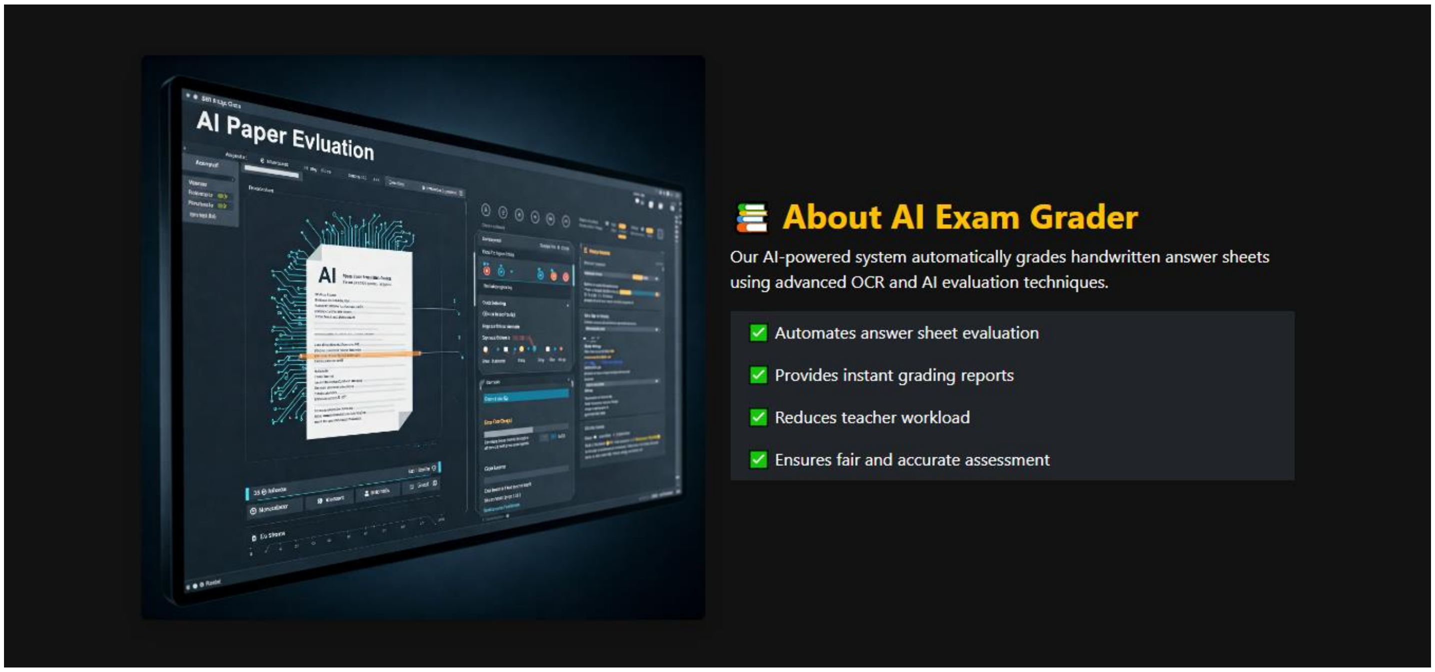
9.2 APPENDIX-B: DEMO SCREENSHOTS



AI-Powered Exam Grading System

Fast. Accurate. Automated.

Fig: 9.1 GradeX Website



About AI Exam Grader

Our AI-powered system automatically grades handwritten answer sheets using advanced OCR and AI evaluation techniques.

- ✓ Automates answer sheet evaluation
- ✓ Provides instant grading reports
- ✓ Reduces teacher workload
- ✓ Ensures fair and accurate assessment

Fig: 9.2 GradeX Information

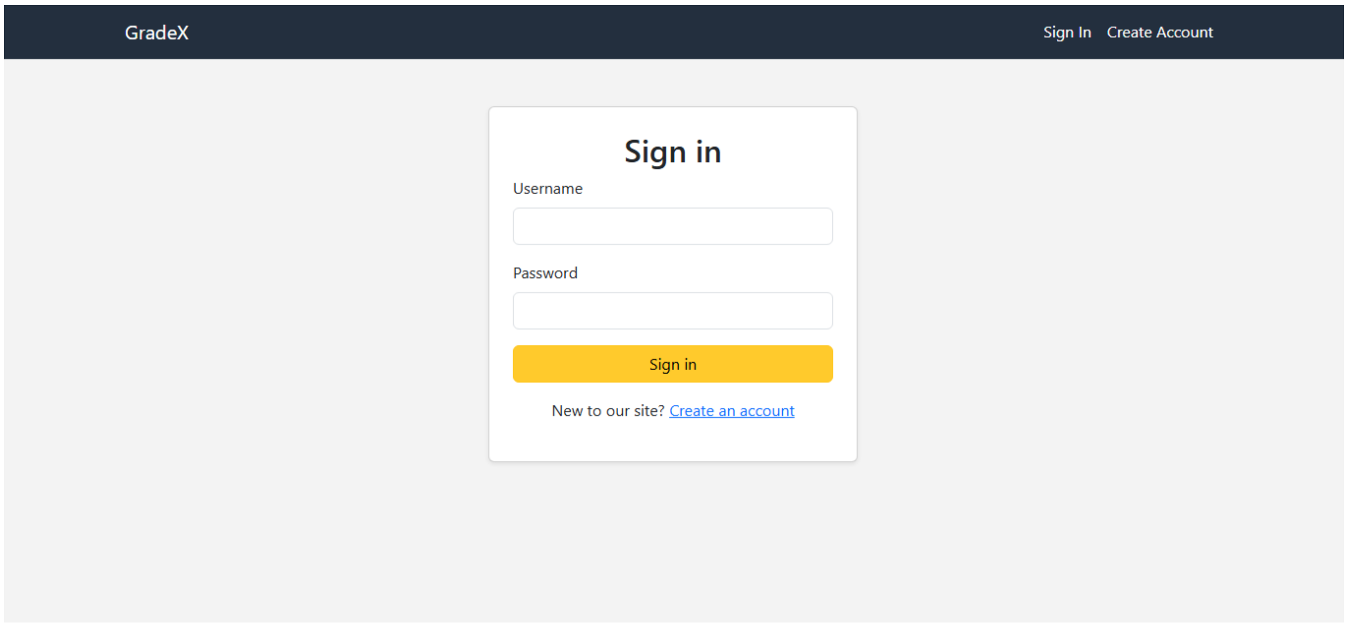


Fig: 9.3 Gradex Student Signin

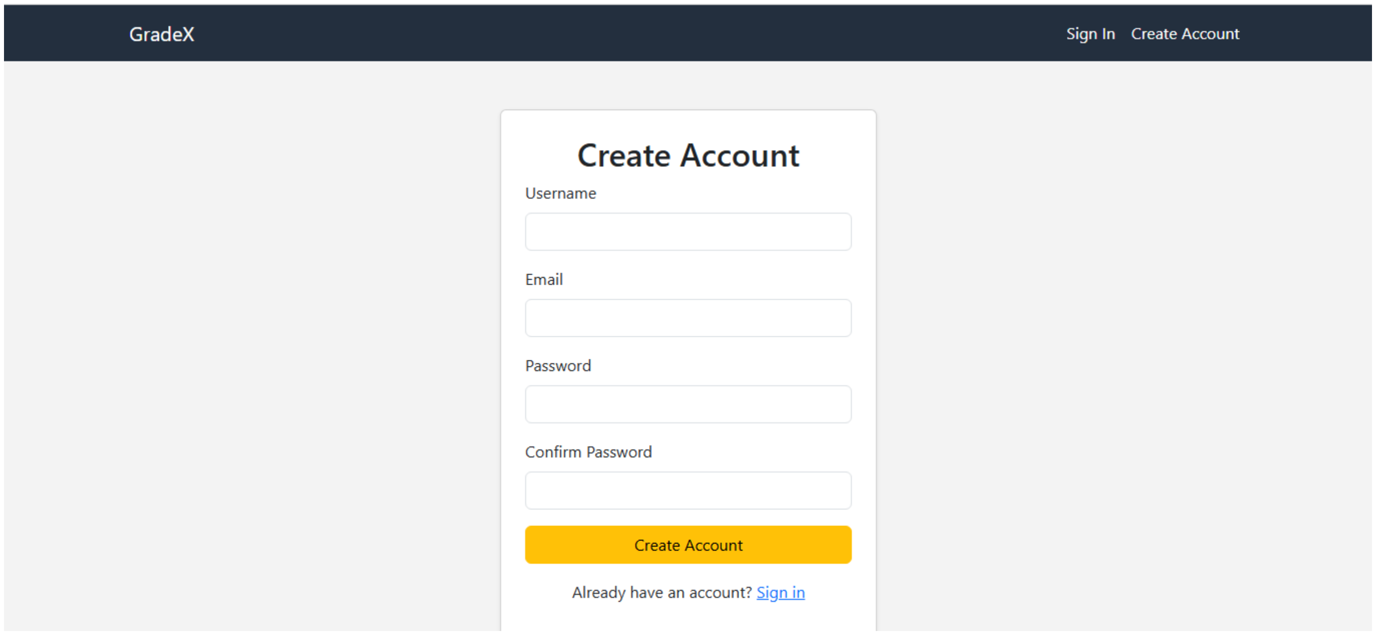
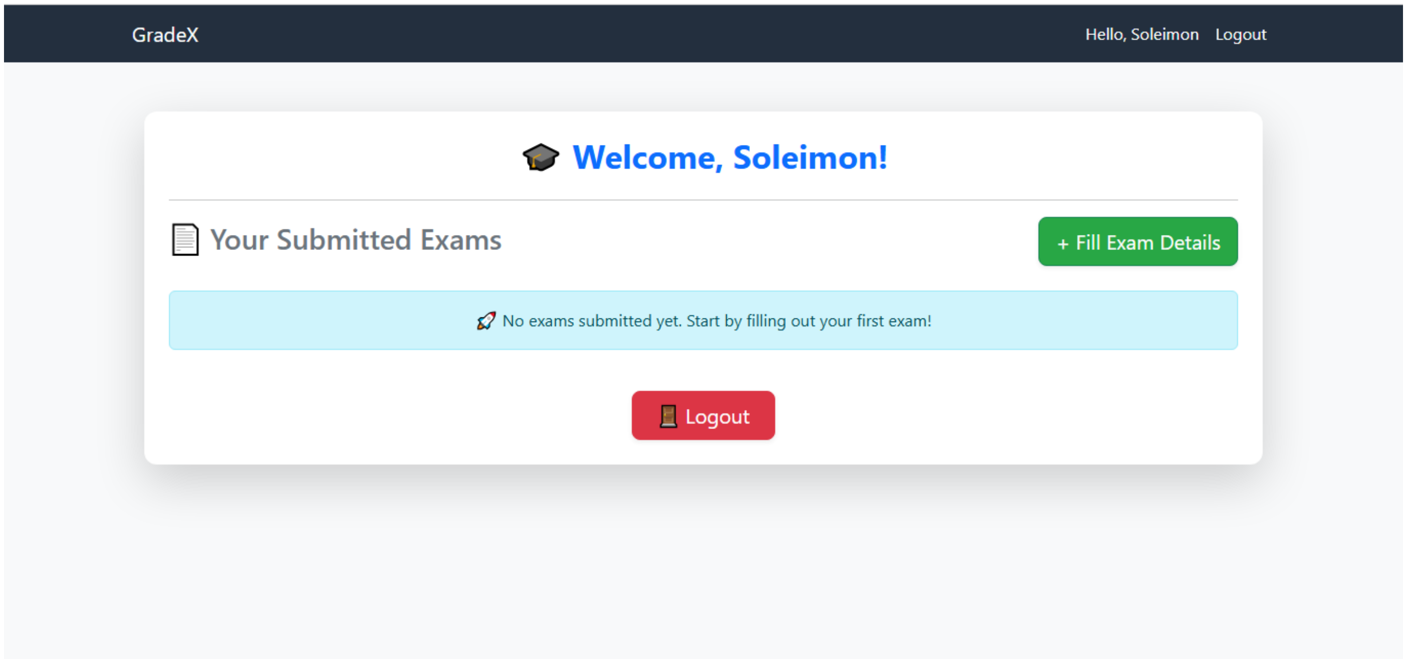


Fig: 9.4 Gradex Student Sign up


GradeX

Hello, Soleimon Logout

Welcome, Soleimon!

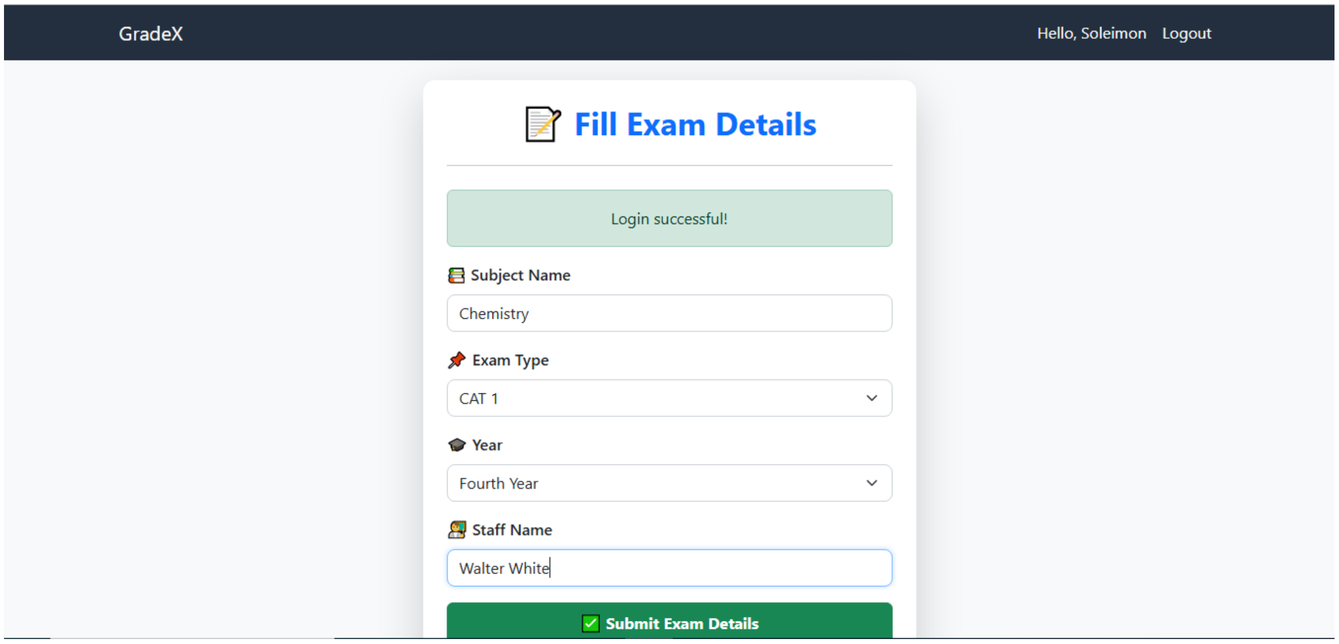
Your Submitted Exams

+ Fill Exam Details

No exams submitted yet. Start by filling out your first exam!

Logout

Fig: 9.5 Gradex Student Dashboard


GradeX

Hello, Soleimon Logout

Fill Exam Details

Login successful!

Subject Name: Chemistry

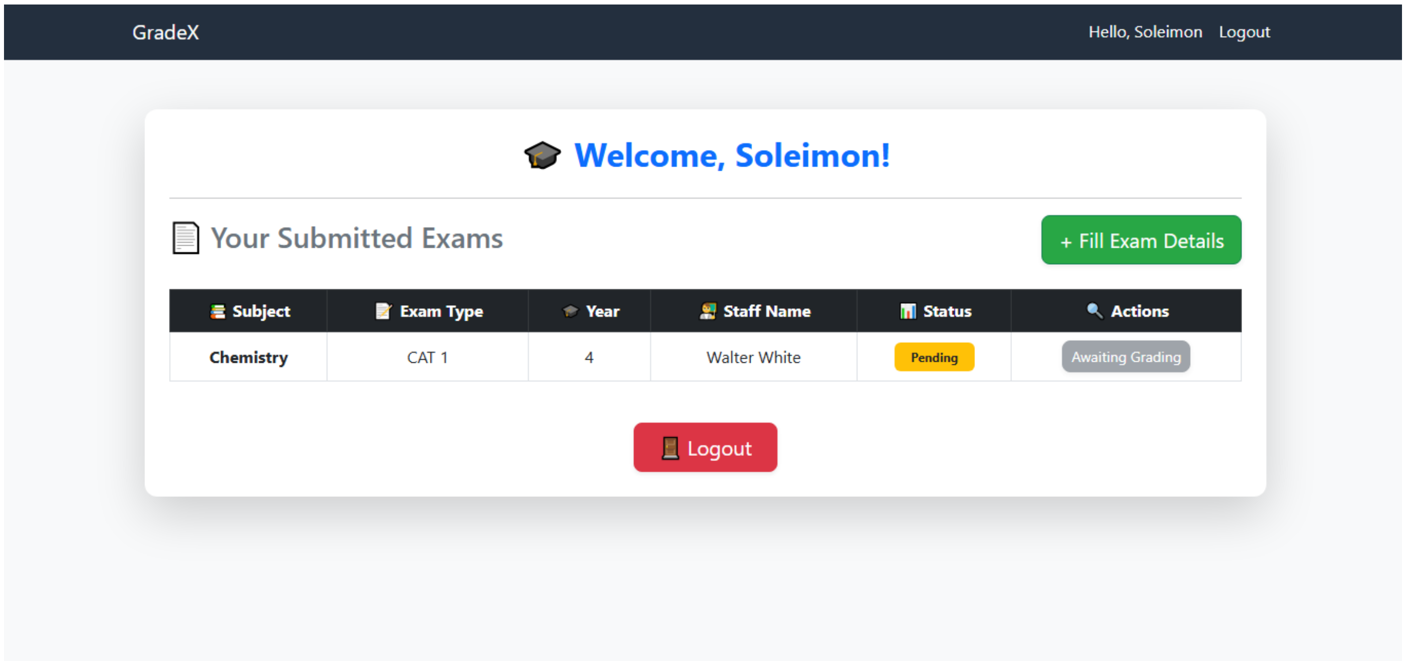
Exam Type: CAT 1

Year: Fourth Year

Staff Name: Walter White

Submit Exam Details

Fig: 9.6 Gradex Student Exam Fill



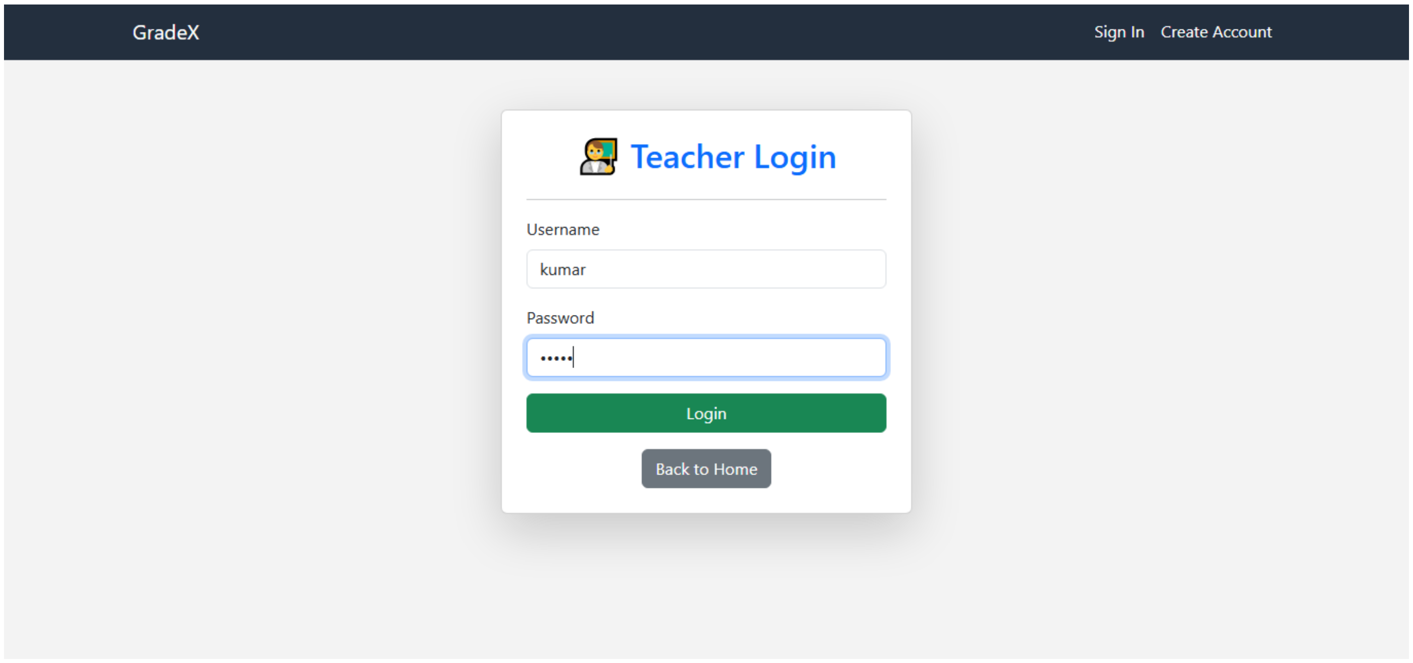
GradeX | Hello, Soleimon | Logout

Welcome, Soleimon!

Your Submitted Exams

Subject	Exam Type	Year	Staff Name	Status	Actions
Chemistry	CAT 1	4	Walter White	Pending	Awaiting Grading

Fig: 9.7 Gradex Student Awaiting Exam Status



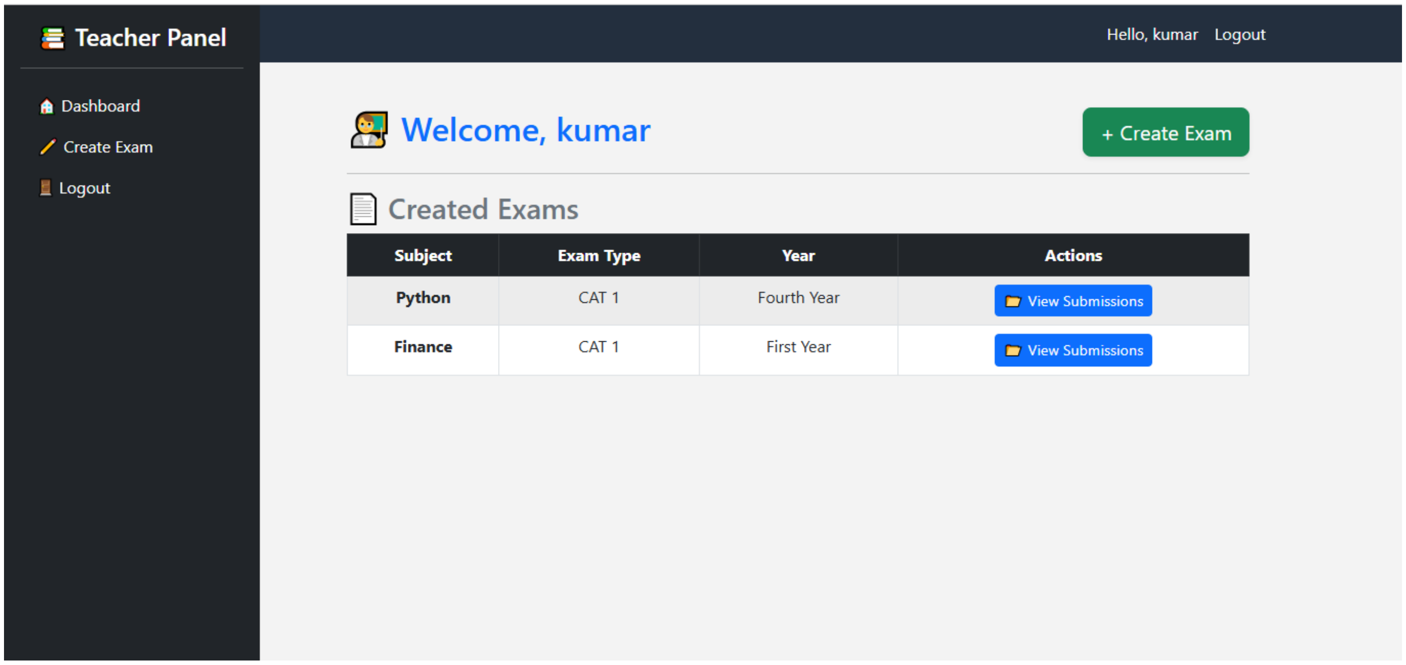
GradeX | Sign In | Create Account

Teacher Login

Username: kumar

Password: •••••

Fig: 9.8 Gradex Teacher Login

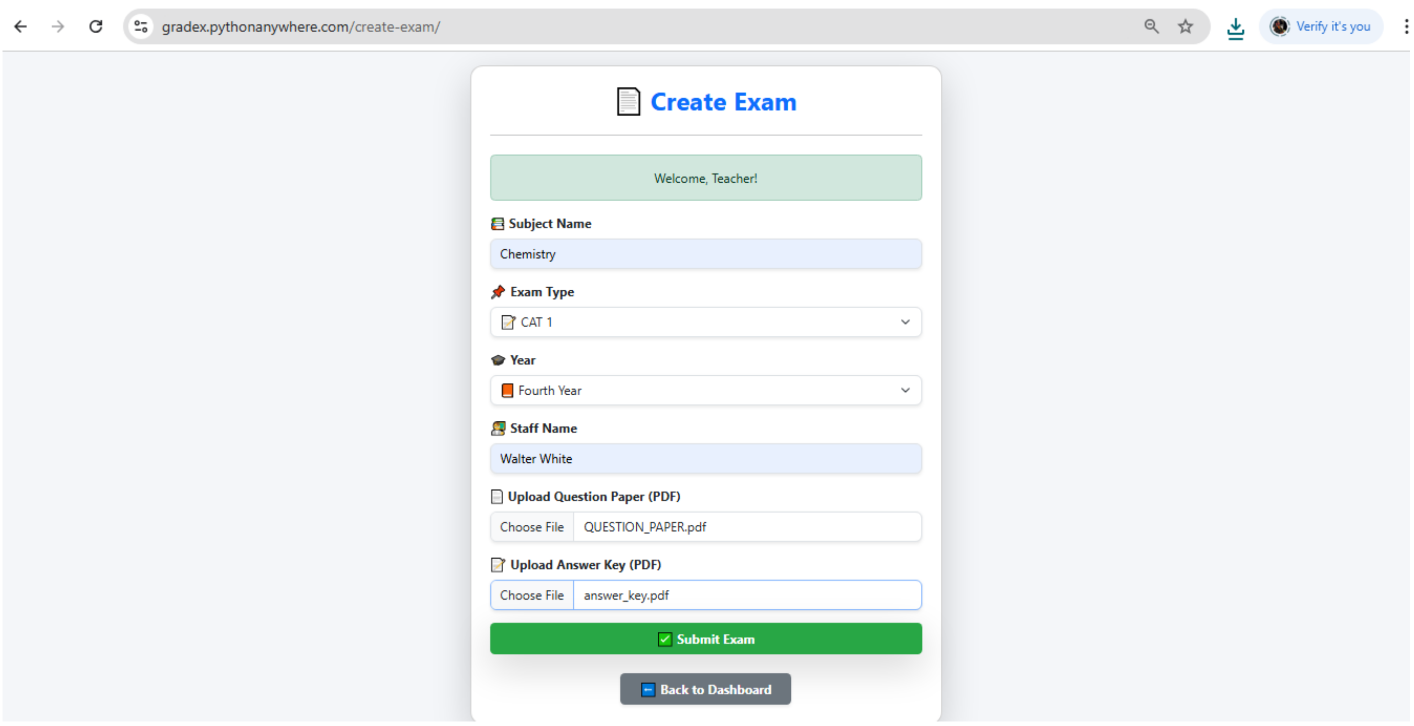


Welcome, kumar

Created Exams

Subject	Exam Type	Year	Actions
Python	CAT 1	Fourth Year	View Submissions
Finance	CAT 1	First Year	View Submissions

Fig: 9.9 Gradex Teacher Dashboard



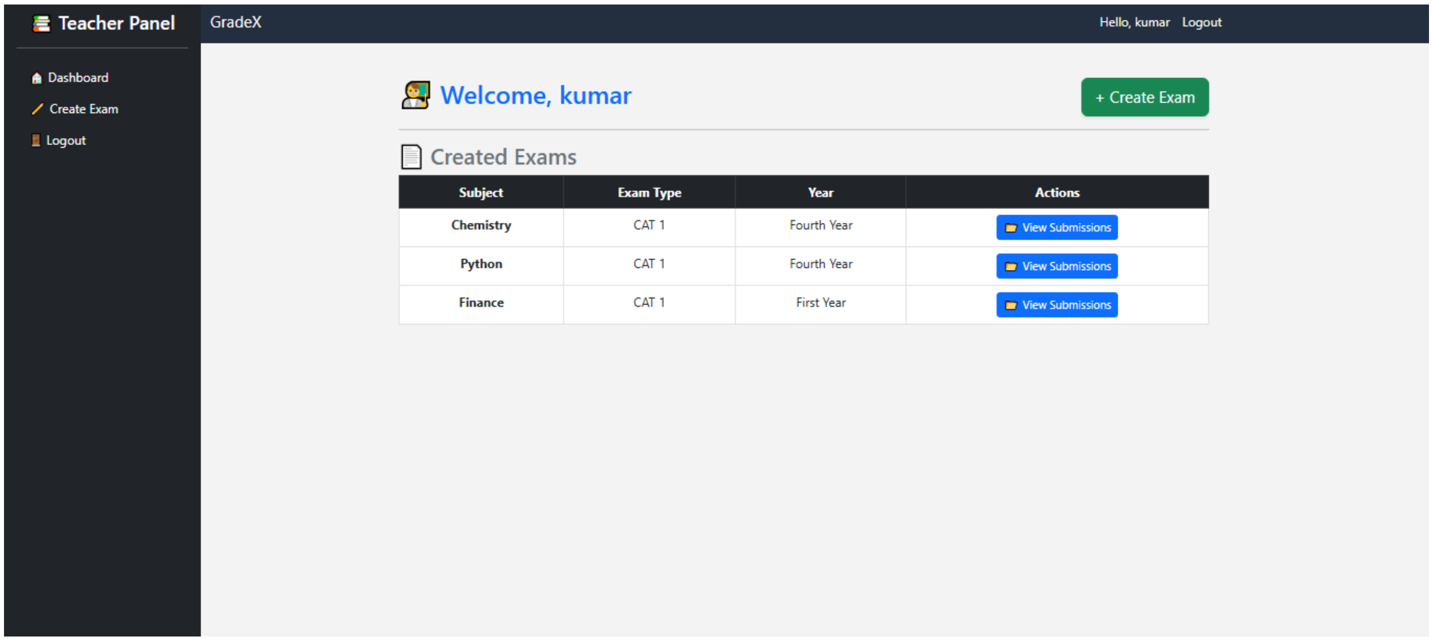
gradex.pythonanywhere.com/create-exam/

Create Exam

Welcome, Teacher!

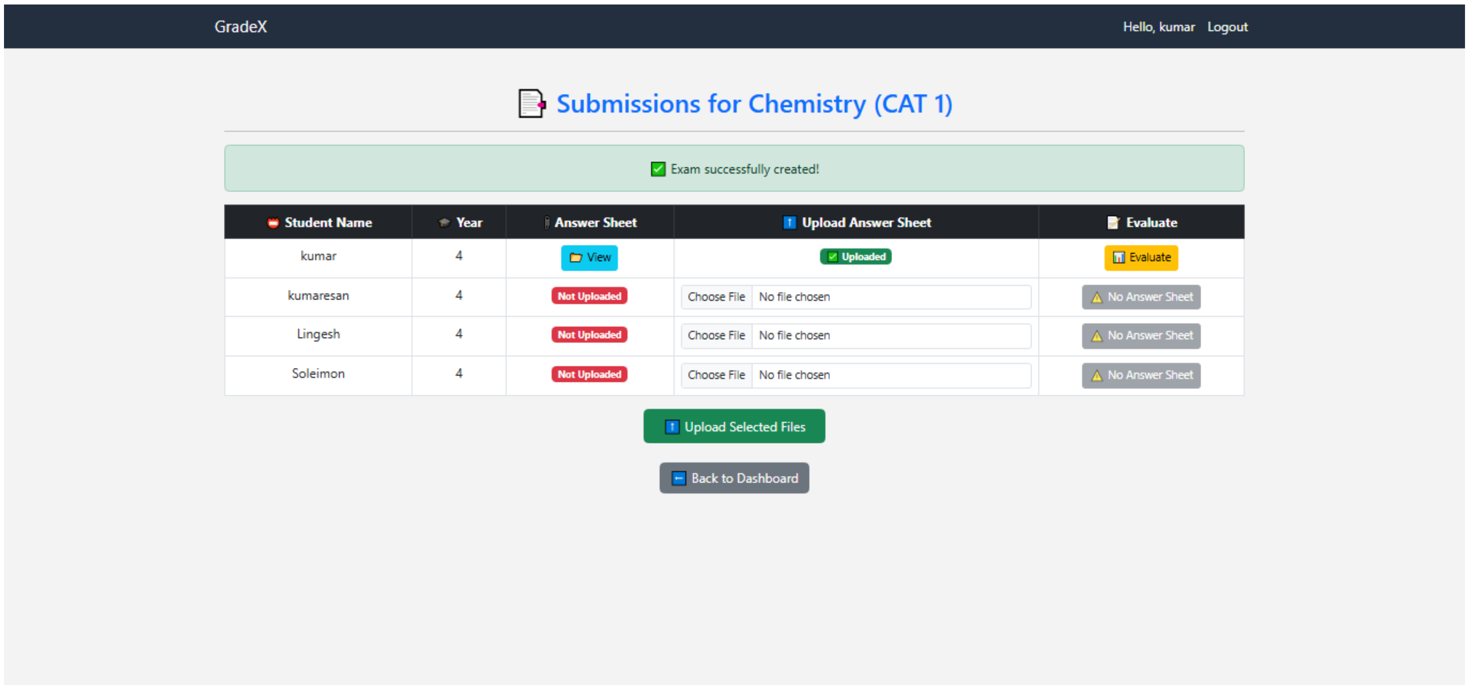
Subject Name	Chemistry
Exam Type	CAT 1
Year	Fourth Year
Staff Name	Walter White
Upload Question Paper (PDF)	Choose File QUESTION_PAPER.pdf
Upload Answer Key (PDF)	Choose File answer_key.pdf

Fig: 9.10 Gradex Teacher Exam Creation



Subject	Exam Type	Year	Actions
Chemistry	CAT 1	Fourth Year	View Submissions
Python	CAT 1	Fourth Year	View Submissions
Finance	CAT 1	First Year	View Submissions

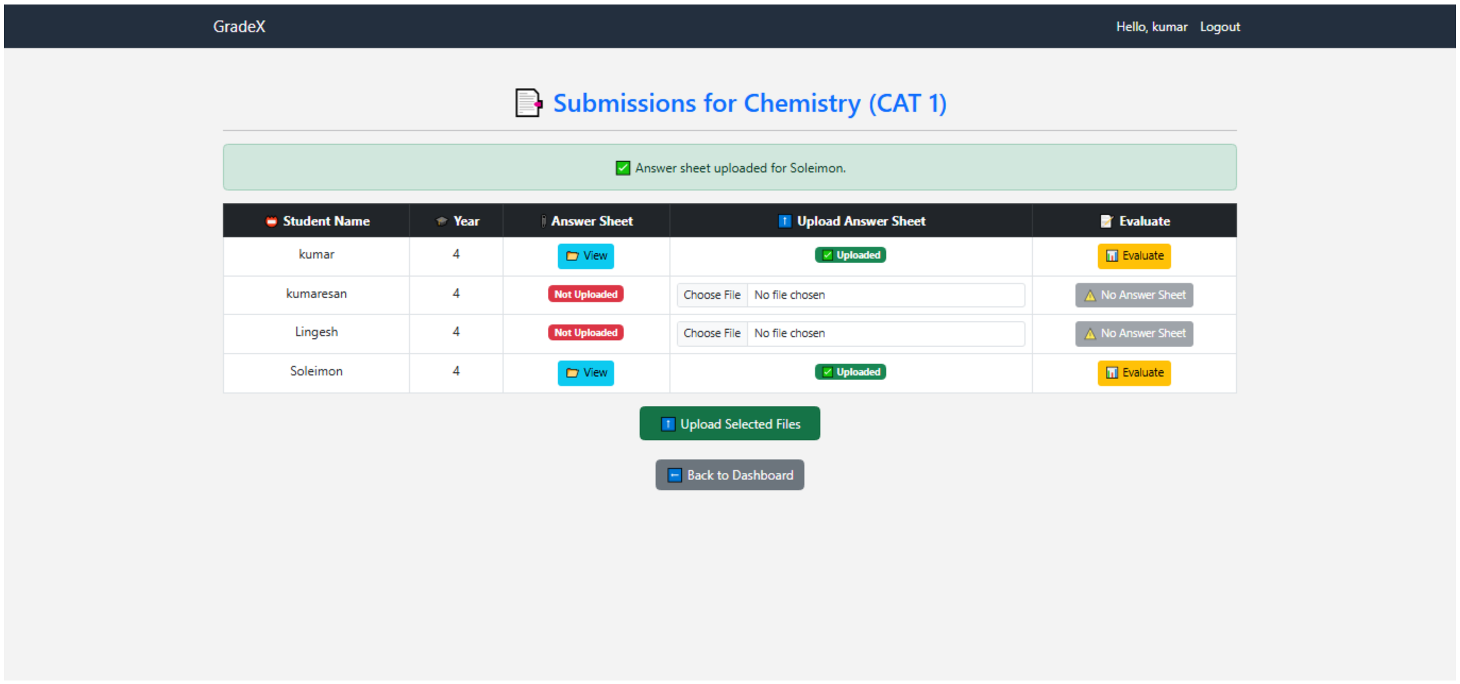
Fig: 9.11 Gradex Teacher Dashboard-Created Exams



✓ Exam successfully created!

Student Name	Year	Answer Sheet	Upload Answer Sheet	Evaluate
kumar	4	View	Uploaded	Evaluate
kumaresan	4	Not Uploaded	Choose File No file chosen	No Answer Sheet
Lingesh	4	Not Uploaded	Choose File No file chosen	No Answer Sheet
Soleimon	4	Not Uploaded	Choose File No file chosen	No Answer Sheet

Fig: 9.12 Gradex Teacher Dashboard-Exam Submissin List



GradeX Hello, kumar Logout

Submissions for Chemistry (CAT 1)

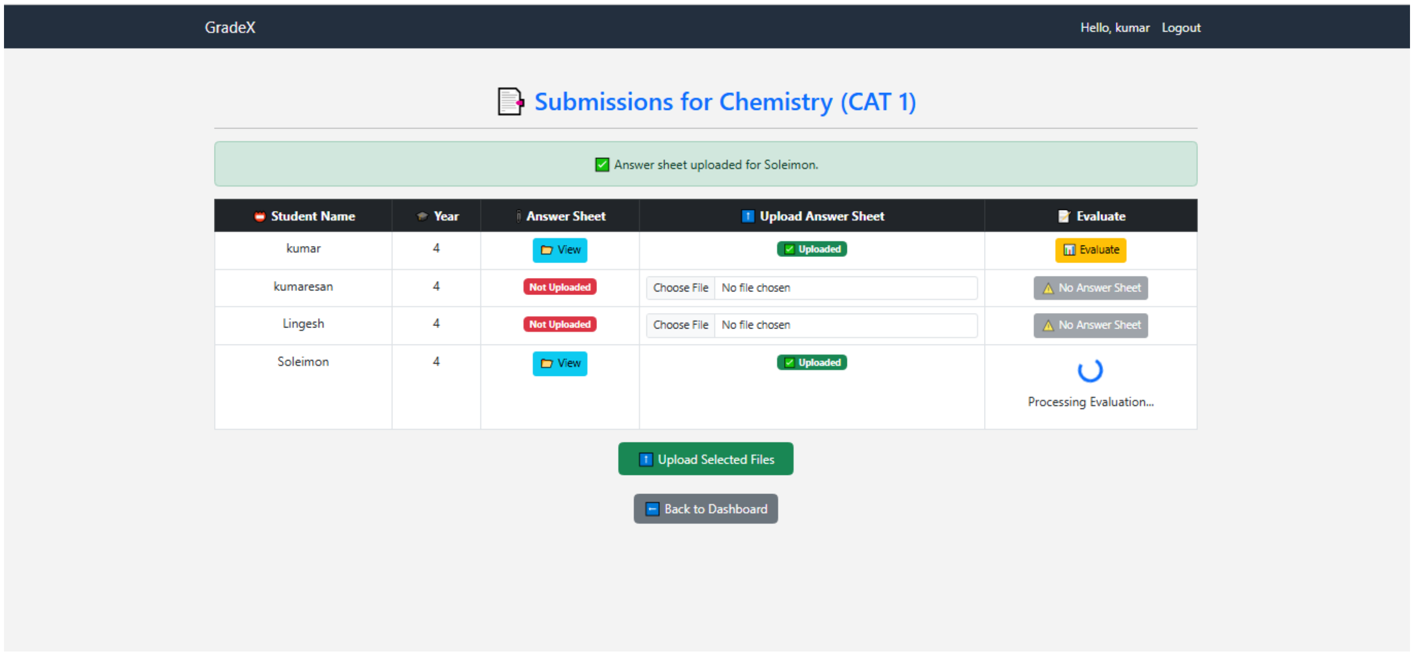
Answer sheet uploaded for Soleimon.

Student Name	Year	Answer Sheet	Upload Answer Sheet	Evaluate
kumar	4	View	Uploaded	Evaluate
kumaresan	4	Not Uploaded	Choose File No file chosen	No Answer Sheet
Lingesh	4	Not Uploaded	Choose File No file chosen	No Answer Sheet
Soleimon	4	View	Uploaded	Evaluate

Upload Selected Files

Back to Dashboard

Fig: 9.13 Gradex Teacher Dashboard-Answer Sheet Upload



GradeX Hello, kumar Logout

Submissions for Chemistry (CAT 1)

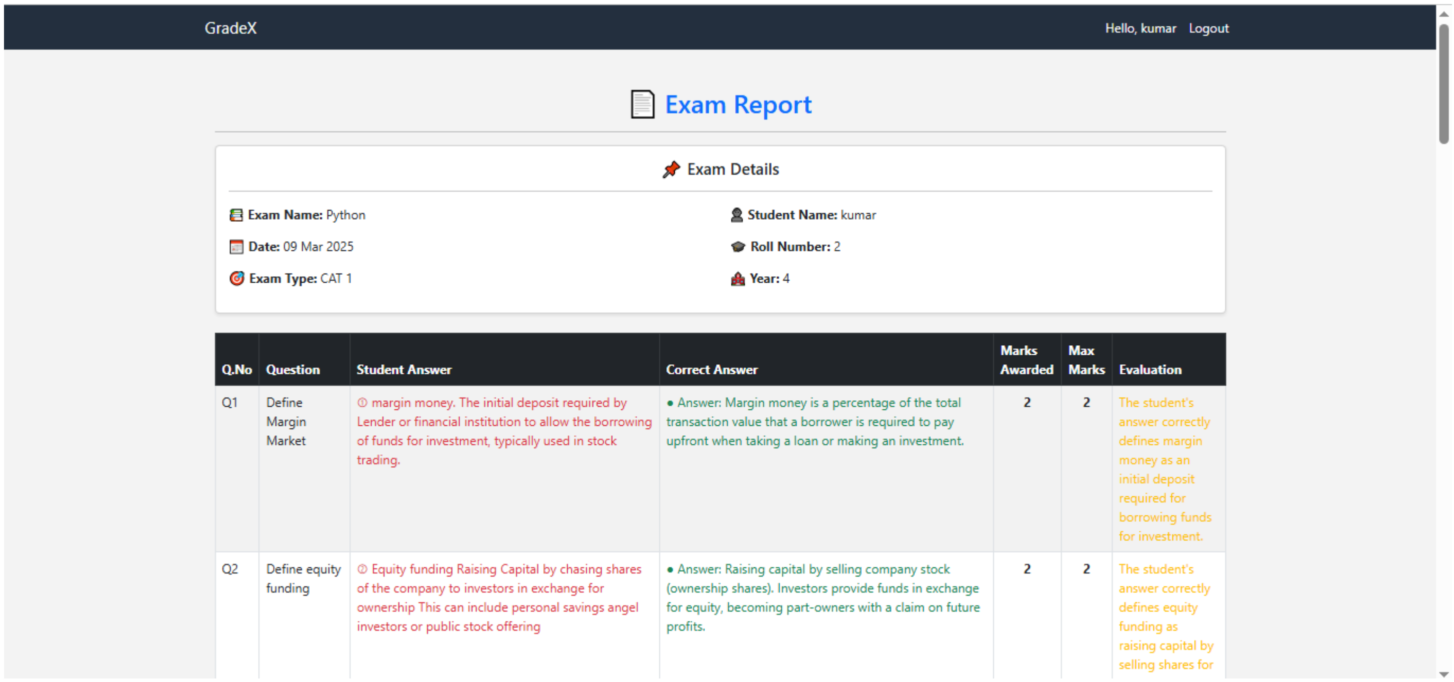
Answer sheet uploaded for Soleimon.

Student Name	Year	Answer Sheet	Upload Answer Sheet	Evaluate
kumar	4	View	Uploaded	Evaluate
kumaresan	4	Not Uploaded	Choose File No file chosen	No Answer Sheet
Lingesh	4	Not Uploaded	Choose File No file chosen	No Answer Sheet
Soleimon	4	View	Uploaded	Processing Evaluation...

Upload Selected Files

Back to Dashboard

Fig: 9.14 Gradex Teacher Dashboard-Answer Sheet Evaluating



GradeX

Hello, kumar Logout

Exam Report

Exam Details

Exam Name: Python

Date: 09 Mar 2025

Exam Type: CAT 1

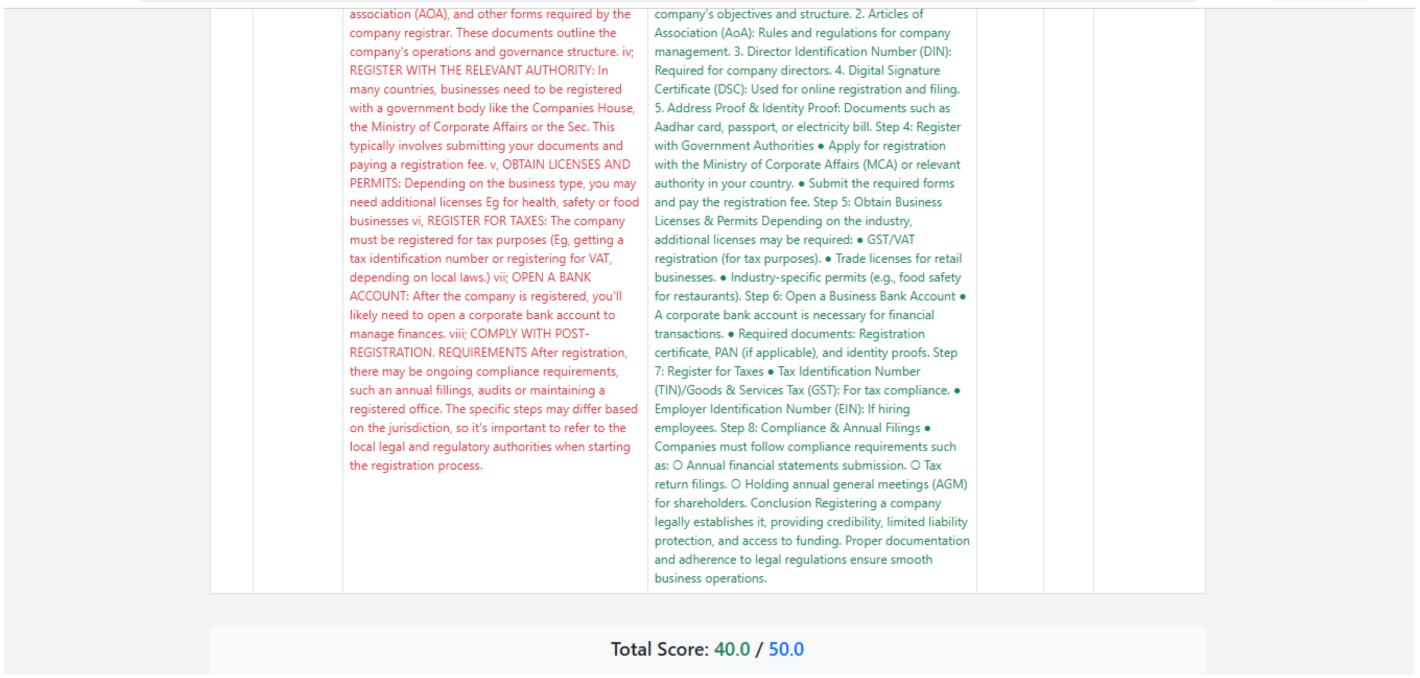
Student Name: kumar

Roll Number: 2

Year: 4

Q.No	Question	Student Answer	Correct Answer	Marks Awarded	Max Marks	Evaluation
Q1	Define Margin Market	margin money. The initial deposit required by Lender or financial institution to allow the borrowing of funds for investment, typically used in stock trading.	● Answer: Margin money is a percentage of the total transaction value that a borrower is required to pay upfront when taking a loan or making an investment.	2	2	The student's answer correctly defines margin money as an initial deposit required for borrowing funds for investment.
Q2	Define equity funding	Equity funding Raising Capital by chasing shares of the company to investors in exchange for ownership This can include personal savings angel investors or public stock offering	● Answer: Raising capital by selling company stock (ownership shares). Investors provide funds in exchange for equity, becoming part-owners with a claim on future profits.	2	2	The student's answer correctly defines equity funding as raising capital by selling shares for

Fig: 9.15 Gradex Teacher Dashboard-Results-1



		association (AOA), and other forms required by the company registrar. These documents outline the company's operations and governance structure. iv; REGISTER WITH THE RELEVANT AUTHORITY: In many countries, businesses need to be registered with a government body like the Companies House, the Ministry of Corporate Affairs or the Sec. This typically involves submitting your documents and paying a registration fee. v, OBTAIN LICENSES AND PERMITS: Depending on the business type, you may need additional licenses Eg for health, safety or food businesses vi, REGISTER FOR TAXES: The company must be registered for tax purposes (Eg, getting a tax identification number or registering for VAT, depending on local laws.) vii; OPEN A BANK ACCOUNT: After the company is registered, you'll likely need to open a corporate bank account to manage finances. viii; COMPLY WITH POST-REGISTRATION. REQUIREMENTS After registration, there may be ongoing compliance requirements, such an annual fillings, audits or maintaining a registered office. The specific steps may differ based on the jurisdiction, so it's important to refer to the local legal and regulatory authorities when starting the registration process.	company's objectives and structure. 2. Articles of Association (AoA): Rules and regulations for company management. 3. Director Identification Number (DIN): Required for company directors. 4. Digital Signature Certificate (DSC): Used for online registration and filing. 5. Address Proof & Identity Proof: Documents such as Aadhar card, passport, or electricity bill. Step 4: Register with Government Authorities ● Apply for registration with the Ministry of Corporate Affairs (MCA) or relevant authority in your country. ● Submit the required forms and pay the registration fee. Step 5: Obtain Business Licenses & Permits Depending on the industry, additional licenses may be required: ● GST/VAT registration (for tax purposes). ● Trade licenses for retail businesses. ● Industry-specific permits (e.g., food safety for restaurants). Step 6: Open a Business Bank Account ● A corporate bank account is necessary for financial transactions. ● Required documents: Registration certificate, PAN (if applicable), and identity proofs. Step 7: Register for Taxes ● Tax Identification Number (TIN)/Goods & Services Tax (GST): For tax compliance. ● Employer Identification Number (EIN): If hiring employees. Step 8: Compliance & Annual Filings ● Companies must follow compliance requirements such as: O Annual financial statements submission. O Tax return filings. O Holding annual general meetings (AGM) for shareholders. Conclusion Registering a company legally establishes it, providing credibility, limited liability protection, and access to funding. Proper documentation and adherence to legal regulations ensure smooth business operations.			

Total Score: 40.0 / 50.0

Fig: 9.16 Gradex Teacher Dashboard-Results-2

Chapter 10
Future Enhancement

While the Gradex system has successfully streamlined the process of evaluating handwritten student answer sheets using OCR and AI, the current workflow still requires manual scanning or photographing of the answer sheets before processing. A major area for improvement lies in automating this input step to make the system more seamless and scalable.

The primary future enhancement will focus on digitizing the answer sheet collection process. Instead of manually scanning or converting student-written sheets into PDFs or image formats, the system can be integrated with school digital infrastructure to automatically ingest answer sheets directly from:

- **Smart exam papers** written on digital pads or tablets with stylus input
- **Mobile app-based capture systems** where teachers simply click pictures, and the app auto-converts and uploads them to the backend
- **Scanner integration APIs** that trigger evaluation as soon as papers are scanned

This would eliminate delays, reduce human effort, and improve the overall efficiency of the system from input to evaluation.

In addition to this, several other enhancements are planned for the Gradex platform:

- **Multilingual Answer Sheet Support**
  Expanding OCR and NLP capabilities to evaluate responses written in regional languages.

- **Real-time Evaluation via Digital Input Devices**
  Supporting direct writing on tablets to allow instant feedback and auto-evaluation.
- **Learning Feedback Loop for Scoring Adjustment**
  Using machine learning models to learn from teacher corrections and adapt future scoring.
- **Advanced Student Performance Analytics**
  Generating detailed reports with topic-wise analytics, progress tracking, and feedback suggestions.
- **Plagiarism Detection**
  Adding modules to detect similar or copied content between students' answers.
- **LMS and Mobile Integration**
  Integrating with Learning Management Systems (LMS) and offering mobile apps for easy access by both students and teachers.

By focusing on automating the initial input step, Gradex will not only become more efficient but also truly scalable for large-scale educational deployments.

REFERENCES

1 Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

2 A. Graves, S. Fernández, M. Liwicki, H. Bunke, and J. Schmidhuber, "Unconstrained online handwriting recognition with recurrent neural networks," in Proc. 20th Int. Conf. Neural Inf. Process. Syst. (NIPS), 2008, pp. 577–584.

3 S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

4 M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in Proc. European Conf. Computer Vision (ECCV), 2014, pp. 818–833.

5 R. Smith, "An overview of the Tesseract OCR engine," in Proc. Int. Conf. Document Anal. Recognit. (ICDAR), 2007, pp. 629–633.

6 OpenAI, "GPT-4 technical report," arXiv preprint arXiv:2303.08774, 2023.

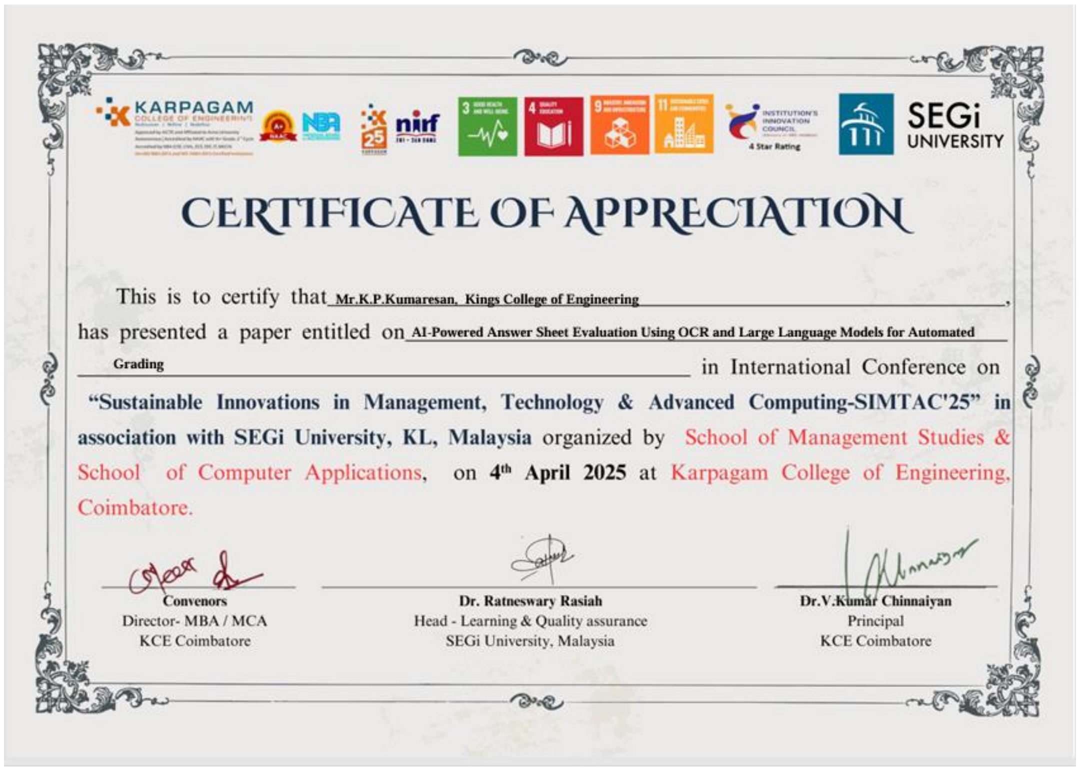
7 R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

8 N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), 2005, vol. 1, pp. 886–893.

9 D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in Proc. Int. Conf. Learn. Represent. (ICLR), 2015.

10 K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778.

CONFERENCE CERTIFICATES



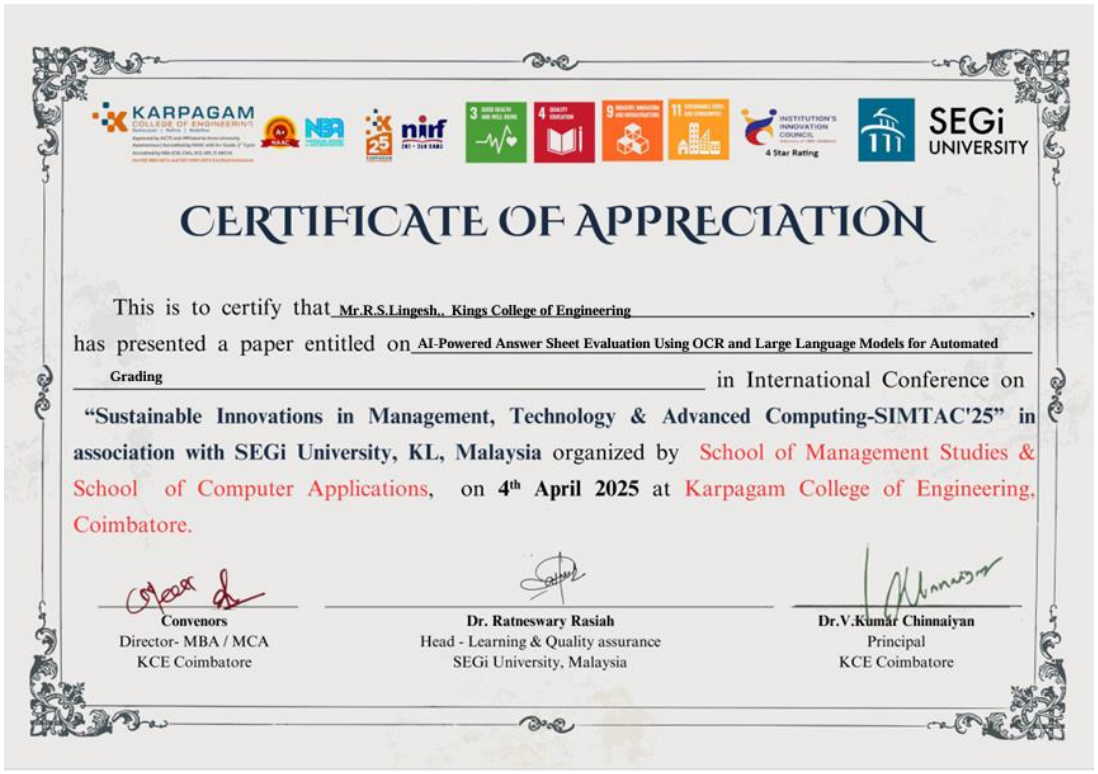
KARPAGAM COLLEGE OF ENGINEERING

SEGi UNIVERSITY

CERTIFICATE OF APPRECIATION

This is to certify that Mr.K.P.Kumaresan, Kings College of Engineering, has presented a paper entitled on AI-Powered Answer Sheet Evaluation Using OCR and Large Language Models for Automated Grading in International Conference on **"Sustainable Innovations in Management, Technology & Advanced Computing-SIMTAC'25" in association with SEGi University, KL, Malaysia** organized by School of Management Studies & School of Computer Applications, on **4th April 2025** at Karpagam College of Engineering, Coimbatore.

[illegible]	[illegible]	[illegible]
Convenors	**Dr. Ratneswary Rasiah**	**Dr.V.Kumar Chinnaiyan**
Director- MBA / MCA	Head - Learning & Quality assurance	Principal
KCE Coimbatore	SEGi University, Malaysia	KCE Coimbatore



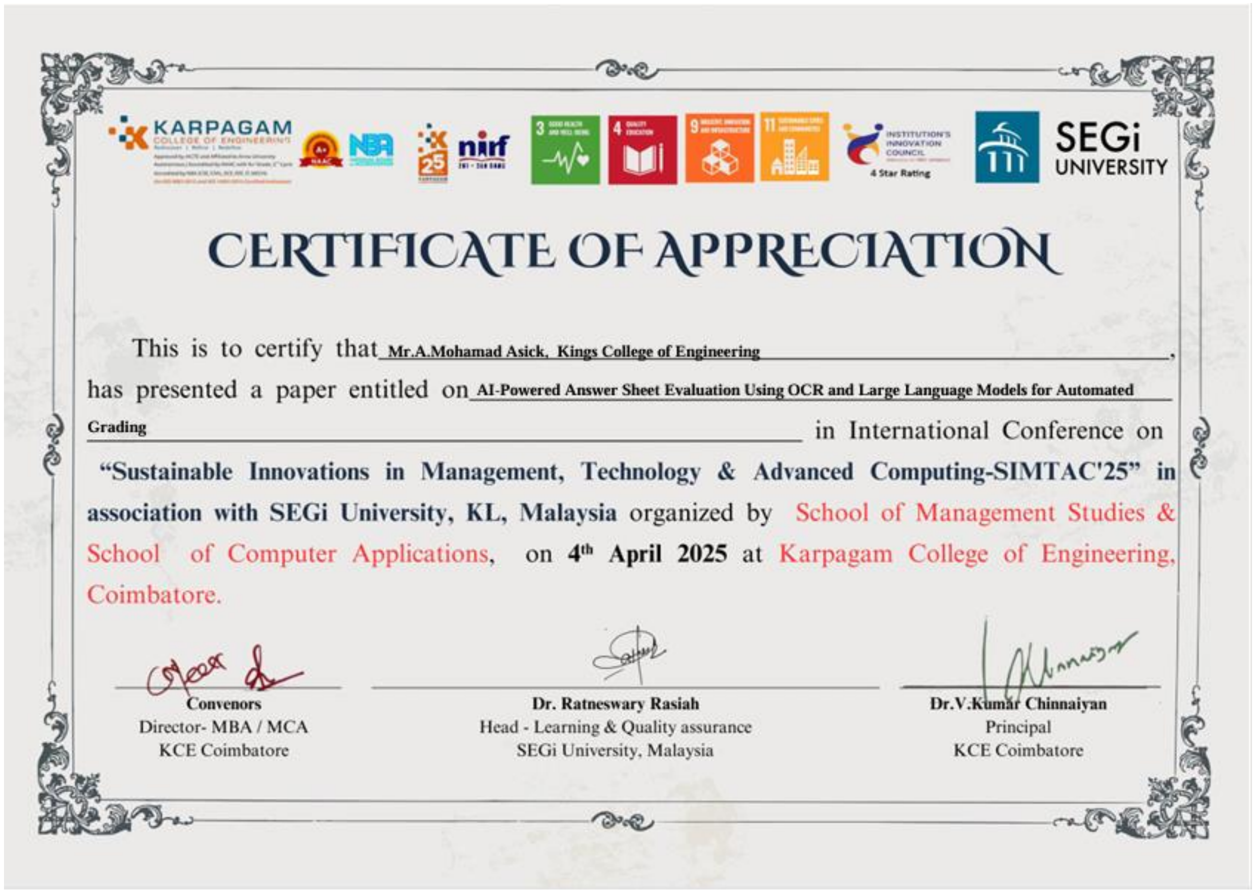
KARPAGAM COLLEGE OF ENGINEERING

SEGi UNIVERSITY

CERTIFICATE OF APPRECIATION

This is to certify that Mr.R.S.Lingesh,, Kings College of Engineering, has presented a paper entitled on AI-Powered Answer Sheet Evaluation Using OCR and Large Language Models for Automated Grading in International Conference on **"Sustainable Innovations in Management, Technology & Advanced Computing-SIMTAC'25" in association with SEGi University, KL, Malaysia** organized by School of Management Studies & School of Computer Applications, on **4th April 2025** at Karpagam College of Engineering, Coimbatore.

[illegible]	[illegible]	[illegible]
Convenors	**Dr. Ratneswary Rasiah**	**Dr.V.Kumar Chinnaiyan**
Director- MBA / MCA	Head - Learning & Quality assurance	Principal
KCE Coimbatore	SEGi University, Malaysia	KCE Coimbatore



KARPAGAM COLLEGE OF ENGINEERING

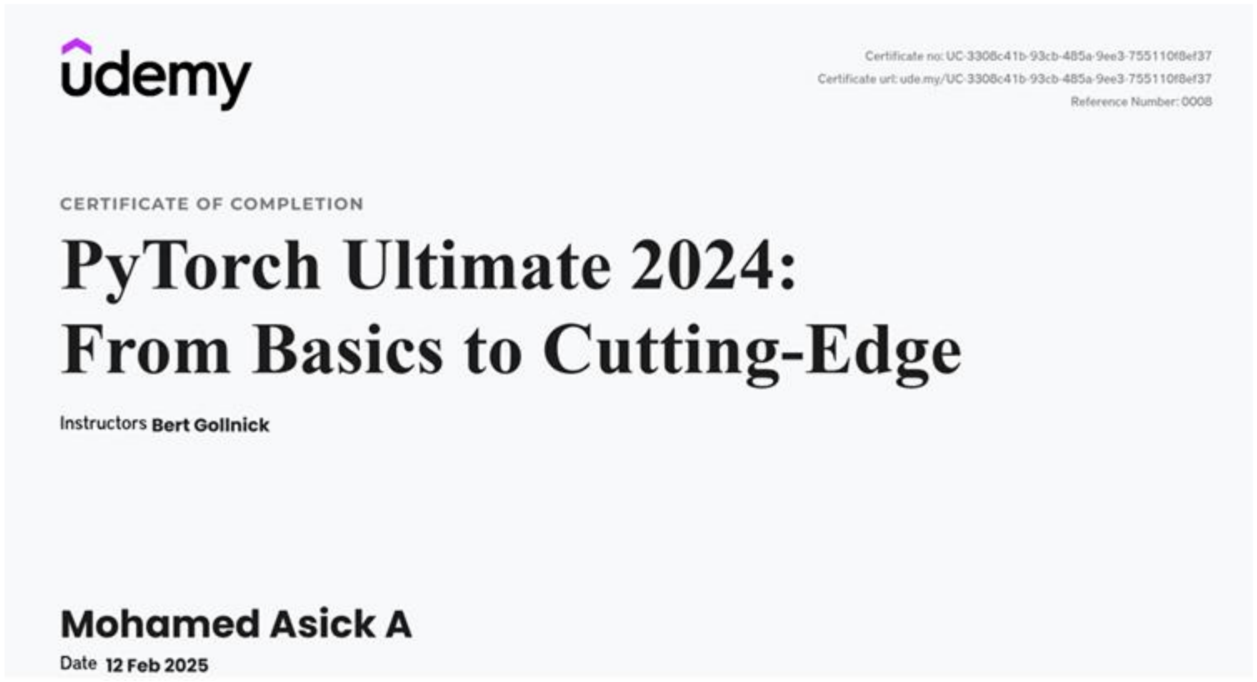
SEGi UNIVERSITY

CERTIFICATE OF APPRECIATION

This is to certify that Mr.A.Mohamad Asick, Kings College of Engineering, has presented a paper entitled on AI-Powered Answer Sheet Evaluation Using OCR and Large Language Models for Automated Grading in International Conference on **"Sustainable Innovations in Management, Technology & Advanced Computing-SIMTAC'25" in association with SEGi University, KL, Malaysia** organized by School of Management Studies & School of Computer Applications, on **4th April 2025** at Karpagam College of Engineering, Coimbatore.

[illegible]	[illegible]	[illegible]
Convenors	**Dr. Ratneswary Rasiah**	**Dr.V.Kumar Chinnaiyan**
Director- MBA / MCA	Head - Learning & Quality assurance	Principal
KCE Coimbatore	SEGi University, Malaysia	KCE Coimbatore

COURSE CERTIFICATES



Certificate no: UC-3308c41b-93cb-485a-9ee3-755110f8ef37
Certificate url: ude.my/UC-3308c41b-93cb-485a-9ee3-755110f8ef37
Reference Number: 0008

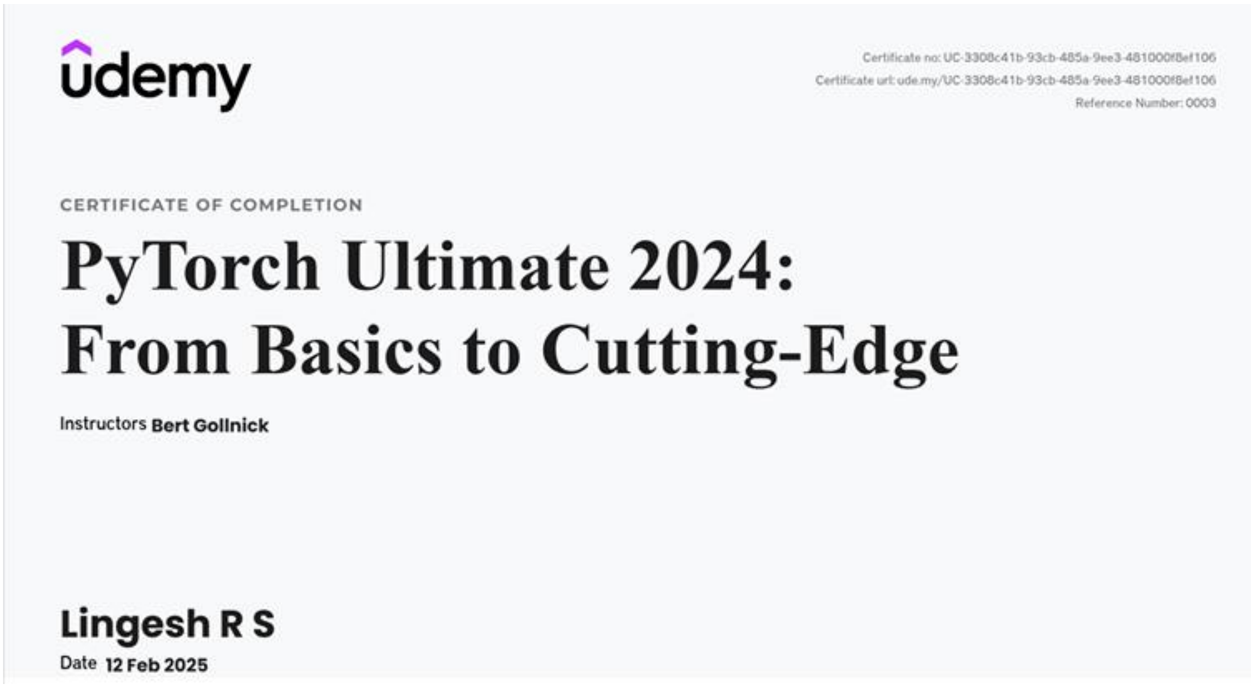
CERTIFICATE OF COMPLETION

PyTorch Ultimate 2024: From Basics to Cutting-Edge

Instructors **Bert Gollnick**

Mohamed Asick A

Date **12 Feb 2025**



Certificate no: UC-3308c41b-93cb-485a-9ee3-481000f8ef106
Certificate url: ude.my/UC-3308c41b-93cb-485a-9ee3-481000f8ef106
Reference Number: 0003

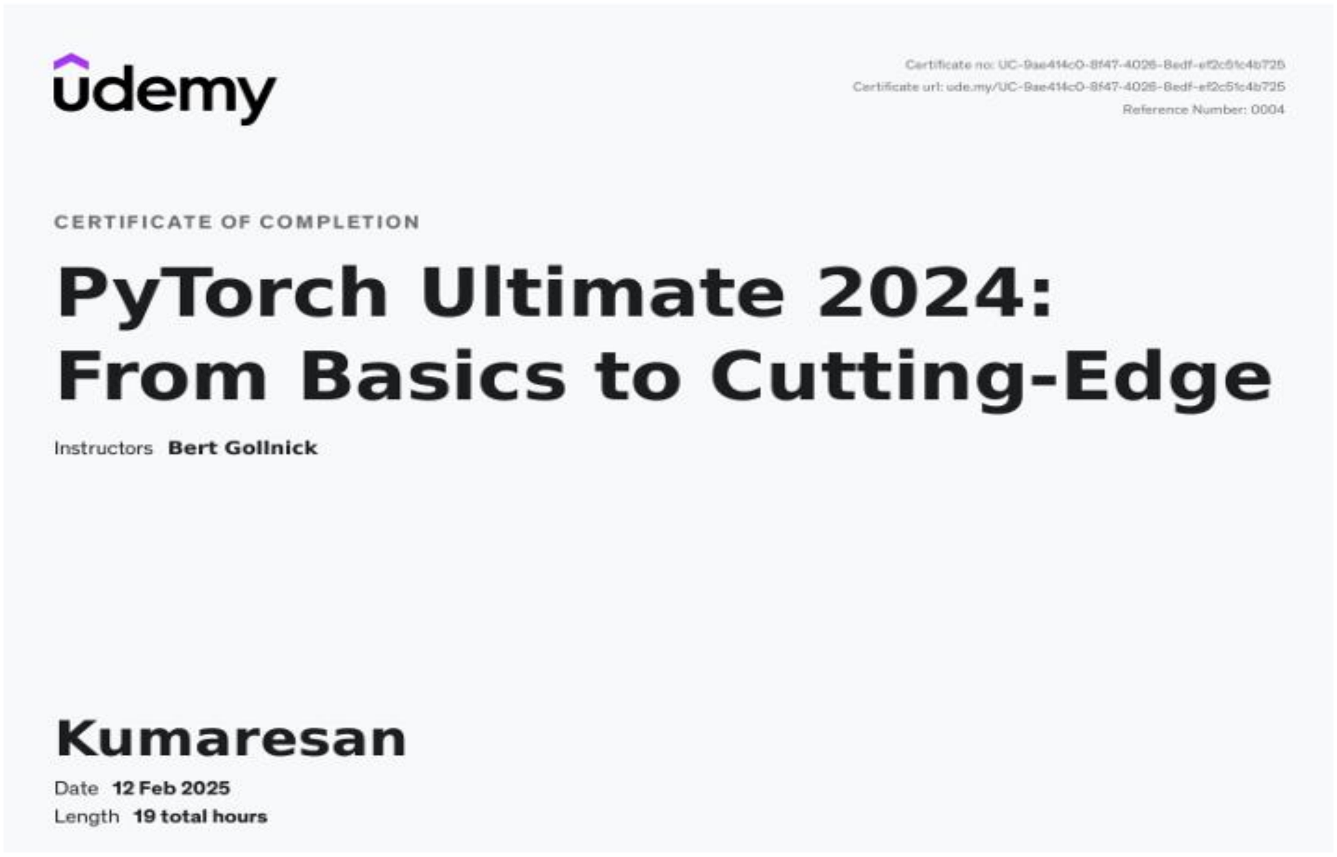
CERTIFICATE OF COMPLETION

PyTorch Ultimate 2024: From Basics to Cutting-Edge

Instructors **Bert Gollnick**

Lingesh R S

Date **12 Feb 2025**



udemy

Certificate no: UC-9ae414c0-8f47-4026-8edf-ef2c51c4b725
Certificate url: ude.my/UC-9ae414c0-8f47-4026-8edf-ef2c51c4b725
Reference Number: 0004

CERTIFICATE OF COMPLETION

PyTorch Ultimate 2024: From Basics to Cutting-Edge

Instructors **Bert Gollnick**

Kumaresan

Date **12 Feb 2025**
Length **19 total hours**