**“EduQuiz-Student Teacher Portal with MCQ Generator**”



A PROJECT WORK SUBMITTED TO THE

**DEPARTMENT OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND INFORMATION TECHNOLOGY (BSC.CSIT)**

**NATIONAL COLLEGE OF COMPUTER STUDIES**

**FOR THE AWARD OF**

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND INFORMATION TECHNOLOGY (BSC.CSIT)**

BY

**ATULLYA MAHARJAN**

**28863/078**

**5-2-551-67-2021**

UNDER THE SUPERVISION OF

**SUMIT GHISING**

**SEPTEMBER, 2025**

# SUPERVISOR RECOMMENDATION

It is my pleasure to recommend this project work report titled “**EduQuiz**” is prepare under my supervision by **Atullya Maharjan** [28863/078] in partial fulfilment of the requirement of the degree of Bachelor of Science in Computer Science and Information Technology (BSc.CSIT). There report is satisfactory to process for the future evaluation.

………………….

…………............

**Sumit Ghising**

**Supervisor**

Faculty Member

Department of Computer Science and Information Technology

National College of Computer studies

**3rd SEPTEMBER, 2025**

# LETTER OF APPROVAL

This is to certify that the report is evaluated and recommended to the Department of Computer Science and Information Technology for acceptance of the report entitled “**EDUQUIZ: Student Teacher Portal with MCQ Generator”** submitted by “ATULLYA MAHARJAN” in partial fulfilment for the degree of Bachelor of Science in Computer Science and Information Technology (B.Sc.CSIT), Institute of Science and Technology (IOST), Tribhuvan University.

|  |  |
| --- | --- |
| **SIGNATURE of Supervisor**  Sumit Ghising  Faculty Member  Department of Computer Science and Information Technology  National College of Computer Studies  Paknajol, Kathmandu  **SIGNATURE of Internal Examiner**  **Internal Examiner** | **SIGNATURE of Coordinator**  Rajan Poudel  Coordinator of CSIT  Department of Computer Science and Information Technology  National College of Computer Studies  Paknajol, Kathmandu  **SIGNATURE of External Examiner**  **External Examiner** |

# ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to National College of Computer Studies (NCCS) for providing this platform. In completing this project, I have been fortunate to have help. Support and encouragement for many people. I would like to acknowledge them for their cooperation.

Firstly, I extend my heartfelt thanks to my mentor **Mr. Sumit Ghising**, for guiding us through each and every step of the process with knowledge and support. Thank you for your advice, guidance and assistance.

This project has been a tremendous learning experience to showcase my ideas, knowledge, vision, etc. I would like to thank and appreciate my teacher and parents for their encouragement and support in developing the project.

Lastly, with full dedication, I have created a ‘EduQuiz’ that will undoubtedly benefit it’s user.

Atullya Maharjan

28863/078

5-2-551-67-2021

September, 2025

# ABSTRACT

EduQuiz is a web-based platform designed to streamline and enhance the management of classes, assignments, and assessments for administrators, teachers, and students. The platform allows administrators to efficiently add, update, and remove teacher and student profiles, as well as create and organize classes. Teachers can access student information, create and evaluate assignments, and manage their assigned classes. A key feature of EduQuiz is its intelligent multiple-choice question (MCQ) generation system, which automatically creates quizzes from uploaded text or PDF materials using TF-IDF and cosine similarity algorithms and rule-based logic. This automation reduces the time and effort required for quiz preparation while ensuring that assessments are closely aligned with the learning content. Students can easily view their assigned tasks, take the generated quizzes, and submit their work through an intuitive interface. By integrating user management with automated quiz generation, EduQuiz aims to improve the efficiency and engagement of the teaching and learning process.

***Keywords***: *NLP, PDF*, *MCQ Generation, Algorithm, Quiz, TF-IDF, Cosine Similarity*

# TABLE OF CONTENT

[SUPERVISOR RECOMMENDATION ii](#_Toc207646622)

[CERTIFICATE OF APPROVAL iii](#_Toc207646623)

[ACKNOWLEDGEMENTS iv](#_Toc207646624)

[ABSTRACT…………... v](#_Toc207646625)

[TABLE OF CONTENT vi](#_Toc207646626)

[LIST OF FIGURES viii](#_Toc207646627)

[LIST OF TABLES ix](#_Toc207646628)

[LIST OF ABBREVIATIONS x](#_Toc207646629)

[CHAPTER 1: INTRODUCTION 1](#_Toc207646630)

[1.1 Introduction 1](#_Toc207646631)

[1.2 Problem Statement 2](#_Toc207646632)

[1.3 Objectives 2](#_Toc207646633)

[1.4 Scope and Limitation 2](#_Toc207646634)

[1.4.1 Scope 2](#_Toc207646635)

[1.4.2 Limitation 2](#_Toc207646636)

[1.5 Development Methodology 3](#_Toc207646637)

[1.6 Report Organization 3](#_Toc207646638)

[CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW 5](#_Toc207646639)

[2.1 Background Study 5](#_Toc207646640)

[2.2 Literature Review 5](#_Toc207646641)

[CHAPTER 3: SYSTEM ANALYSIS 8](#_Toc207646642)

[3.1 System Analysis 8](#_Toc207646643)

[3.1.1 Requirement Analysis 8](#_Toc207646644)

[3.1.2 Feasibility Analysis 9](#_Toc207646645)

[3.1.3 Data Modeling (ER Diagram) 11](#_Toc207646646)

[3.1.4 Process Modeling (DFD) 12](#_Toc207646647)

[CHAPTER 4: SYSTEM DESIGN 14](#_Toc207646648)

[4.1 Design 14](#_Toc207646649)

[4.1.1 Database Schema Design 14](#_Toc207646650)

[4.1.2 Form Design 15](#_Toc207646651)

[4.1.3 Interface Design 15](#_Toc207646652)

[4.1.4 Dialogue Design 17](#_Toc207646653)

[4.2 Algorithm Details 17](#_Toc207646654)

[CHAPTER 5: IMPLEMENTATION AND TESTING 21](#_Toc207646655)

[5.1 Implementation 21](#_Toc207646656)

[5.1.1 Tools Used 21](#_Toc207646657)

[5.1.2 Implementation Details of Modules 21](#_Toc207646658)

[5.2 Testing 26](#_Toc207646659)

[5.2.1 Test Cases for Unit Testing 27](#_Toc207646660)

[5.2.2 Test Cases for System Testing 30](#_Toc207646661)

[5.3 Result Analysis 32](#_Toc207646662)

[5.3.1 Quantitative Analysis 32](#_Toc207646663)

[5.3.2 Qualitative Analysis 34](#_Toc207646664)

[CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATIONS 35](#_Toc207646665)

[6.1 Conclusion 35](#_Toc207646666)

[6.2 Future Recommendations 35](#_Toc207646667)

[REFERENCES …………………………………………………………………………36](#_Toc207646668)

[APPENDICES……………………………………… a](#_Toc207646669)

# LIST OF FIGURES

[**Figure** **1.1:** Incremental Delivery 3](#_Toc207646763)

[**Figure 3.1:**Use Case Diagram 8](#_Toc207646764)

[**Figure 3.2:** Gantt Chart 11](#_Toc207646765)

[**Figure 3.3:** ER Diagram 12](#_Toc207646766)

[**Figure 3.4:** Level 0 DFD for EduQuiz 13](#_Toc207646767)

[**Figure** **3.5:** Level 1 DFD for EduQuiz 13](#_Toc207646768)

[**Figure 4.1:** Database Schema Design 14](#_Toc207646769)

[**Figure 4.2:** User Login form 15](#_Toc207646770)

[**Figure** **4.3:** Dashboard Interface Design 16](#_Toc207646771)

[**Figure** **4.4:** MCQ generation Interface 16](#_Toc207646772)

[**Figure** **4.5:** Dialogue Design 17](#_Toc207646773)

[**Figure** **5.1:**Output of Data Input & Reading Module 21](#_Toc207646774)

[**Figure** **5.2:** Output of Preprocessing Module 22](#_Toc207646775)

[**Figure** **5.3:**Output of TF-IDF & Similarity Module 23](#_Toc207646776)

[**Figure** **5.4:**Output of Generating Distractors 24](#_Toc207646777)

[**Figure** **5.5:** Output of Question Generation Module 25](#_Toc207646778)

[**Figure** **5.6:** Upload PDF 26](#_Toc207646779)

[**Figure** **5.7:** Generated MCQ 26](#_Toc207646780)

[**Figure** **5.8:** ROUGE Score 33](#_Toc207646781)

# LIST OF TABLES

[**Table** **5.1:** Tool Used 21](#_Toc207430281)

[**Table** **5.2:**Test Case for User Registration (Teacher and Student) 25](#_Toc207430282)

[**Table** **5.3:** Test Case for Login 26](#_Toc207430283)

[**Table** **5.4:** Test Case for MCQ's Generation by Teacher 26](#_Toc207430284)

[**Table** **5.5:** Test Case for Create Assignment by Teacher 27](#_Toc207430285)

[**Table** **5.6:**Test for Quiz 28](#_Toc207430286)

[**Table** **5.7:** System Testing 29](#_Toc207430287)

# LIST OF ABBREVIATIONS

**AI** Artificial Intelligence

**API** Application Program Interface

**BERT** Bidirectional Encoder Representations from Transformers

**BLEU** Bilingual Evaluation Understudy

**CSS** Cascading Style Sheet

**DFD** Data Flow Diagram

**IDE** Integrated Development Environment

**MCQ** Multiple Choice Question

**MERN** MongoDB, Express.js, React, and Node.js

**NLP** Natural Language Processing

**NLTK** Natural Language Toolkit

**PDF** Portable Document Format

**ROUGE** Recall-Oriented Understudy for Gisting Evaluation

**TF-IDF** Term Frequency- Inverse Document Frequency

**UML** Unified Modeling Language

**VSCODE** Visual Studio Code

# INTRODUCTION

## Introduction

EduQuiz is a web-based platform developed to simplify and automate the educational workflow by integrating the roles of administrators, teachers, and students into a unified system. The platform provides administrators with comprehensive control over the academic environment, enabling them to efficiently add, remove, and update teacher and student profiles, as well as create and manage classes. This centralized management ensures smooth coordination and organization within educational institutions.

Teachers are equipped with a suite of tools that allow them to access detailed student information, create and assign various types of assignments, monitor student submissions, and evaluate performance. A key feature of EduQuiz is its intelligent MCQ generation system, which automatically creates multiple-choice questions from textual content or PDF documents provided by teachers. This is achieved using advanced Natural Language Processing (NLP) techniques, specifically TF-IDF (Term Frequency-Inverse Document Frequency), cosine similarity algorithms, and T5 transformer which analyze the content to identify important concepts and generate relevant, high-quality questions. This automation significantly reduces the time and effort teachers spend on preparing assessments, allowing them to focus more on teaching and student engagement.

Students interact with EduQuiz through a user-friendly interface where they can view their assigned classes and assignments, take automatically generated MCQ quizzes, submit their work, and receive instant feedback with scores calculated in real-time. The system also maintains a detailed history of quizzes and assignments, enabling both students and teachers to track progress and identify areas for improvement.

By combining role-based access, automated quiz generation, and streamlined assignment management, EduQuiz aims to enhance the overall teaching and learning experience. It supports continuous assessment, fosters better student engagement, and provides valuable insights into learning outcomes. education, where efficient, scalable, and adaptive assessment tools are essential.

## Problem Statement

Educational management tasks like user administration, class organization, assignment creation, and assessment generation are often handled manually and separately, causing inefficiencies. Administrators find it challenging to manage teacher and student data effectively, while teachers spend excessive time creating assignments and MCQs that may not align well with course content. Students lack a unified platform to access assignments, take quizzes, and hindering their ability to track learning progress.

## Objectives

* To develop a web-based platform with role-based access control for efficient management of users, classes, and assignments, while integrating TF-IDF and cosine similarity and T5 model to automatically generate multiple-choice questions (MCQs) from PDF or text content

## Scope and Limitation

### Scope

EduQuiz is a web-based educational platform that covers schools, colleges, universities, training centers, and online learning platforms where managing classes, assignments, and exams is important. The system can be used by teachers to quickly create quizzes and assignments, by students to access study materials and attempt tests, and by administrators to manage users and classes in an organized way. It is also useful for training institutes and corporate organizations that need to prepare assessments from study materials, manuals, or company documents. By reducing manual work and providing automated question generation, EduQuiz makes learning and evaluation easier, faster, and more efficient for everyone involved.

### Limitation

The system currently supports only PDF and plain text inputs for generating MCQs; other file formats like Word documents or images are not supported. The MCQ generation relies on TF-IDF and cosine similarity, which may limit the complexity and variety of questions compared to manual creation. Additionally, EduQuiz operates only in English, restricting its usability for non-English speakers.

## Development Methodology

The **development methodology** used for EduQuiz is **Incremental Delivery,** which means the project was built in small, manageable parts, with each part adding more features to the system.

This approach worked well because EduQuiz has multiple complex features like dynamic MCQ creation, an exam module, and user interactions. Breaking the project into smaller iterations made it easier to adjust, fix issues, and improve features step by step.

In the **first iteration,** the focus was on the core MCQ generation system, including key sentence identification, keyword extraction, and distractor generation. Later iterations added features for the **admin** to manage users and classes, for **teachers** to generate quizzes and create assignments, view classes and student progress, and for **students** to take MCQs, submit assignment and check results.

Overall, using the **Incremental Delivery** method made development flexible and adaptive, resulting in a stable and user-friendly platform for students, teachers, and admins.

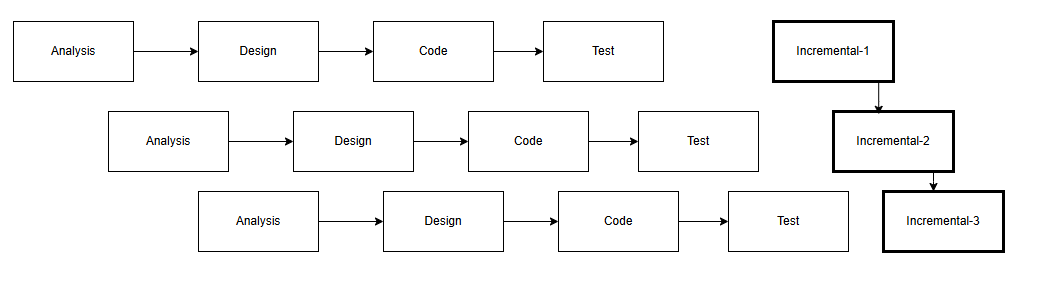


Figure .: Incremental Delivery

## Report Organization

The report on ‘EduQuiz’ is structured into 6 chapters, which include

**Chapter 1: Introduction**

It gives a brief overview of the project, including its goals, what it covers, its limitations, and the main problems it aims to solve.

**Chapter 2: Background Study and Literature Review**

This chapter covers background research, explaining basic ideas, important concepts, and terms related to the project. It also includes a review of similar projects and studies done by other researchers.

**Chapter 3: System Analysis**

This chapter includes requirement analysis and feasibility study of the system. It also consists of the UML diagrams of the project.

**Chapter 4: System Design**

This chapter describes how the system was built based on the analysis. It explains the design choices made, including the architecture of the system, database schema, and user interface layouts. It also discusses the tools, frameworks, programming languages, and database technologies used in the development.

**Chapter 5: Implementation and Testing**

This chapter presents how the system was implemented and tested. It describes how various modules were developed and integrated, and it details the testing process to ensure the system works as expected. It includes test cases, results, and analysis of system performance.

**Chapter 6: Conclusion and Future Recommendation**

It summarizes the key findings, insights, and outcomes derived from the research or project and provides a road map for continuous improvement and innovation based on conclusion drawn.

# BACKGROUND STUDY AND LITERATURE REVIEW

## Background Study

In recent years, the education sector has increasingly turned to digital platforms to enhance teaching and learning, yet educators still face challenges in creating and managing multiple-choice questions (MCQs), a crucial tool for evaluating student understanding. Traditional manual preparation is time-consuming, difficult to scale with large course materials, and often limited by generic question banks that may not align with specific class content, reducing assessment effectiveness. With advances in Artificial Intelligence (AI) and Natural Language Processing (NLP), these challenges can be addressed by leveraging techniques such as keyword extraction, summarization, and question generation, which transform raw educational content into structured, scalable, and contextually relevant assessments.

EduQuiz is a web-based platform that leverages these technologies to help teachers automatically generate and manage MCQs from their teaching materials, especially PDF files. The system extracts and processes text, identifies key topics using NLP techniques like keyword extraction, ranks important sentences, and creates questions with correct answers and believable distractors. Teachers can upload materials, review questions, and assign quizzes, while students can take quizzes, receive feedback, and track their progress. By automating MCQ creation, EduQuiz saves teachers significant time and ensures that assessments are closely aligned with course content, making it particularly valuable for modern, remote learning environments.

## Literature Review

A comprehensive pipeline has been proposed that integrates large language models (LLMs) with prompt engineering to automate MCQ generation and evaluation from university regulations. The multi-stage system includes document preprocessing, question generation, and an automated review process that aligns closely with expert human evaluations. This approach demonstrates the effectiveness of LLMs in reducing educators’ workload while maintaining high question quality. [1]

A system for automatic MCQ generation in the computer science domain was developed to provide continuous feedback for students while reducing the effort required for manual question creation. The approach follows three main steps: extracting informative sentences, identifying key concepts, and generating distractors. Techniques such as TF-IDF, Jaccard similarity, quality phrase mining, K-means clustering, and BERT (Bidirectional Encoder Representations from Transformers) were applied to process textual data from 9th and 11th-grade computer science textbooks. The generated MCQs were validated by domain experts, achieving 80% overall accuracy, with key generation and distractor generation accuracy at 83% and 77%, respectively. The solution was implemented as a desktop application that accepts textual input, generates MCQs, and displays them through an interface, aiming to assist teachers, students, and other stakeholders in automating the MCQ creation process. [2]

Ontology-based multiple-choice question (MCQ) generation has been investigated by leveraging existing ontologies to automatically create assessment items. Educational assessment theories were applied to evaluate MCQs generated from ontological concepts, while concept similarity measures were explored to reliably control question difficulty. The study demonstrated the feasibility of ontology-driven MCQ generation and highlighted its potential for producing valid and scalable assessment materials in educational contexts. [3]

An automatic question generation (AQG) system was developed focusing on Indonesian texts, addressing the scarcity of AQG research for this language. The system utilized the multilingual Text-to-Text Transfer Transformer (mT5), a modern transformer model, to generate questions from context. The model was fine-tuned on the Indonesian portion of the TyDiQA dataset, aiming to extract answers from context and form relevant questions. The system’s performance was evaluated using BLEU and ROUGE metrics, achieving BLEU-1, BLEU-2, BLEU-3, BLEU-4, and ROUGE-L scores of 36.54, 28.24, 22.61, 18.44, and 39.57, respectively. Manual validation confirmed that the generated questions were grammatically sound and semantically appropriate, demonstrating the potential of transformer-based models for AQG in low-resource languages. [4]

The design and implementation of project-based student portals were explored to support experiential learning during the COVID-19 pandemic. The system incorporated features such as resource sharing, assignment management, and feedback mechanisms, enabling seamless interaction between students, teachers, and administrators. Emphasis was placed on multi-role accessibility, usability, and real-time collaboration, which facilitated effective remote learning.[5]

The development of a school data portal utilizing open data was presented to improve access to educational information. A comprehensive user research process involving stakeholders such as students, teachers, and administrators was conducted to understand the needs and challenges in data accessibility. The portal was designed to provide transparent, user-friendly access to school performance metrics, demographic data, and resource allocation. The iterative development process emphasized usability, data accuracy, and responsiveness to user feedback, aiming to empower educational decision-making through open data. [6]

A web-based Student Information Management System (SIMS) was developed to simplify the maintenance and management of student data for educational institutions. The system provides a secure online interface to track student information throughout the academic lifecycle, including personal details, academic records, attendance, curriculum, exam results, and project submissions. It also manages faculty data, batch execution, and academic notifications, facilitating streamlined communication between administration, staff, and students. Additionally, SIMS supports generating detailed reports and queries related to students, courses, semesters, and certifications, thereby enhancing administrative efficiency and data accessibility.[7]

A Natural Language Processing (NLP) based model was proposed for automatic generation of diverse question-answer pairs to reduce the time required for manual question creation. The system generates complex questions using rule-based algorithms, while multiple-choice questions and fill-in-the-blanks are produced using a combination of the GloVe vector model and rule-based techniques. The model was trained and evaluated on the SQuAD dataset, demonstrating the ability to produce syntactically correct and varied question types.[8]

# SYSTEM ANALYSIS

## System Analysis

### Requirement Analysis

Requirement analysis is an important stage in the development of EduQuiz. It involves gathering and analysing the requirements in order to identify the goals, features, and functions that system should have.

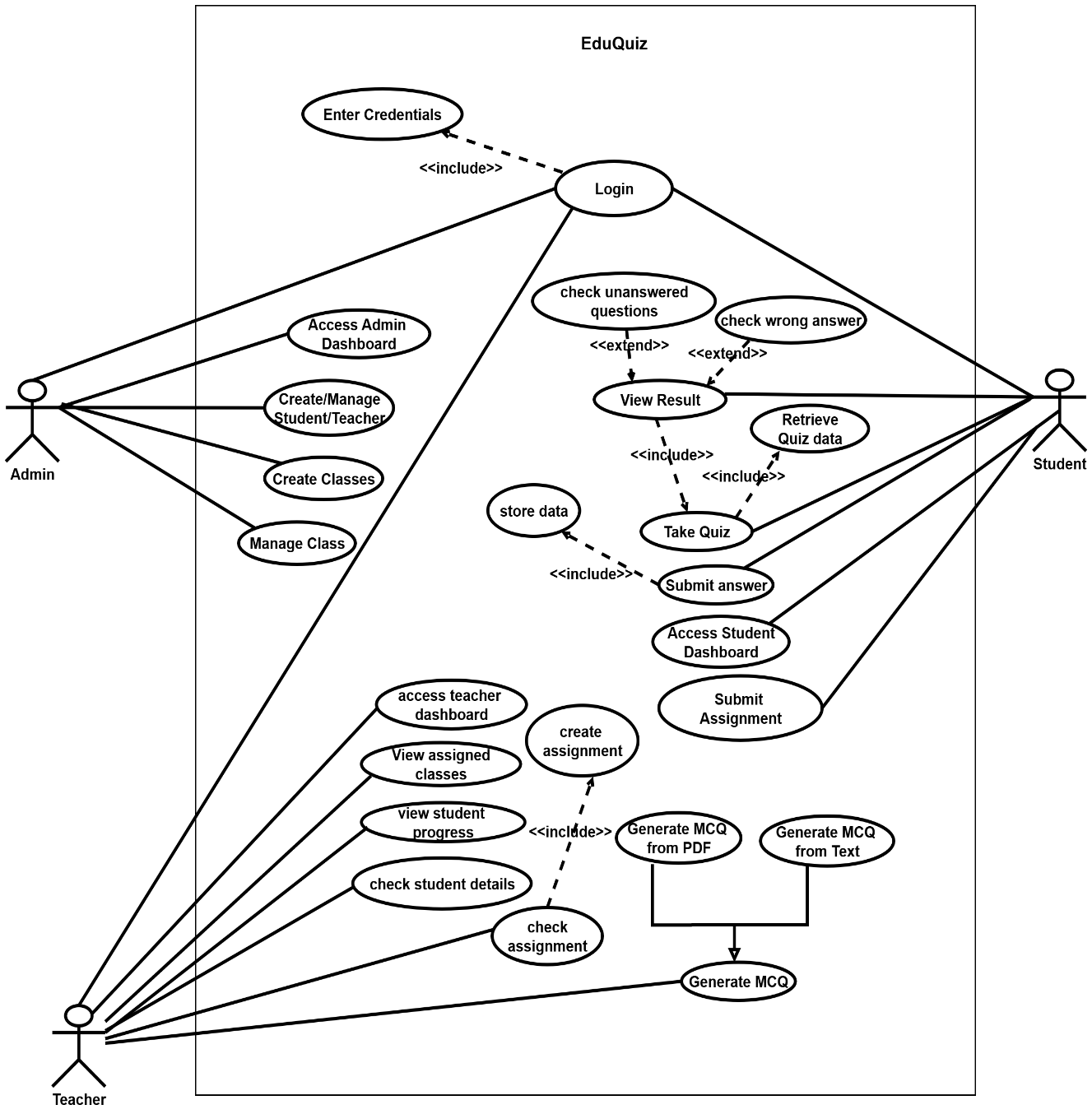
Functional Requirements 

Figure .:Use Case Diagram

The Use Case Diagram for EduQuiz highlights the key functionalities available to its three main actors: Student, Teacher, and Admin.

* Student: Can log in, take quizzes, submit answers and assignments, and view results. Additionally, students have the ability to review incorrect or unanswered responses for better understanding.
* Teacher: Can log in, create assignments, view submitted assignments, manage classes, view assigned classes, and generate MCQs in either PDF or text format. Teachers can also monitor student performance to track progress.
* Admin: Can register new teachers or students, create and manage classes, and oversee user management to ensure smooth system administration.

Non-Functional Requirements

* Performance: EduQuiz should work quickly and smoothly, even when many users are using it at the same time.
* Scalability: The system should be able to handle more users and activities as the number of students and teachers grows, without slowing down or needing big changes.
* Availability: EduQuiz should be available to users almost all the time, with only short breaks for updates or maintenance.
* Compliance: The system must follow all important laws and rules to keep users’ information safe and protect the platform.
* User Experience: EduQuiz should have a clean, simple design that makes it enjoyable and easy for users to navigate and keep using.
* Mobile Compatibility: The platform should work well on phones and tablets, adapting its layout so users have a good experience on any device.

### Feasibility Analysis

A feasibility study looks at whether the project can work in real life by checking three main areas: how it will operate, the technology needed, and the costs involved.

Technical Feasibility

The EduQuiz system is technically feasible because it is built using the MERN stack (MongoDB, Express.js, React.js, Node.js), which allows smooth integration between frontend and backend and supports scalability. Key features like user authentication, assignment management, and quiz participation are handled efficiently. For automatic MCQ generation, a Python backend service uses NLP techniques such as TF-IDF and cosine similarity to extract important content and create meaningful questions. The Node.js backend communicates with the Python service through REST APIs, ensuring seamless integration. The system is designed to handle multiple users at the same time while keeping data secure and consistent, making it practical and achievable from a technical perspective.

Economic Feasibility

The EduQuiz platform is financially viable because it uses the MERN stack (MongoDB, Express.js, React.js, Node.js), and python libraries which is open-source and free, minimizing software costs. However, development still requires investment in designing and customizing features like automatic MCQ generation, user management, and assignment handling. Additional expenses include web hosting, domain registration, and ongoing maintenance. Overall, the expected benefits of improving educational efficiency and automating assessments outweigh these costs, making the project economically feasible.

Operational Feasibility

The EduQuiz platform is easy to use for administrators, teachers, and students, ensuring smooth daily operations. It automates key tasks like MCQ generation, assigning assignments and quiz management, reducing manual work and errors. The system is designed to support users with different roles effectively, making it practical and useful for improving the learning and teaching experience. Overall, the platform meets the operational needs and can be successfully adopted by its users.

Schedule Feasibility

The timeline for the EduQuiz project is well-planned and aligns with academic deadlines. The planning, requirement analysis, and database design were completed during the proposal phase. Core backend functionalities, including user authentication, assignment creation, and quiz management, have been implemented. The Python-based MCQ generation service and REST API integration are currently in progress, followed by frontend integration and user interface refinements. Given the current progress and steady development pace, the remaining features are expected to be completed and tested before the final submission. The schedule is realistic, ensuring that the project will be delivered

on time.

Figure .: Gantt Chart

### Data Modeling (ER Diagram)

The ER diagram of EduQuiz illustrates the key entities such as Admin, Teacher, Student, Class, Assignment, and MCQ Quiz. The Admin manages Teachers and Students and creates Classes. Teachers create Assignments and MCQ Quizzes linked to specific Classes. Students belong to Classes and participate in quizzes and assignments. This diagram clearly shows how users, classes, and assessments interact within the system, providing a clear overview of the platform’s data structure and relationships.

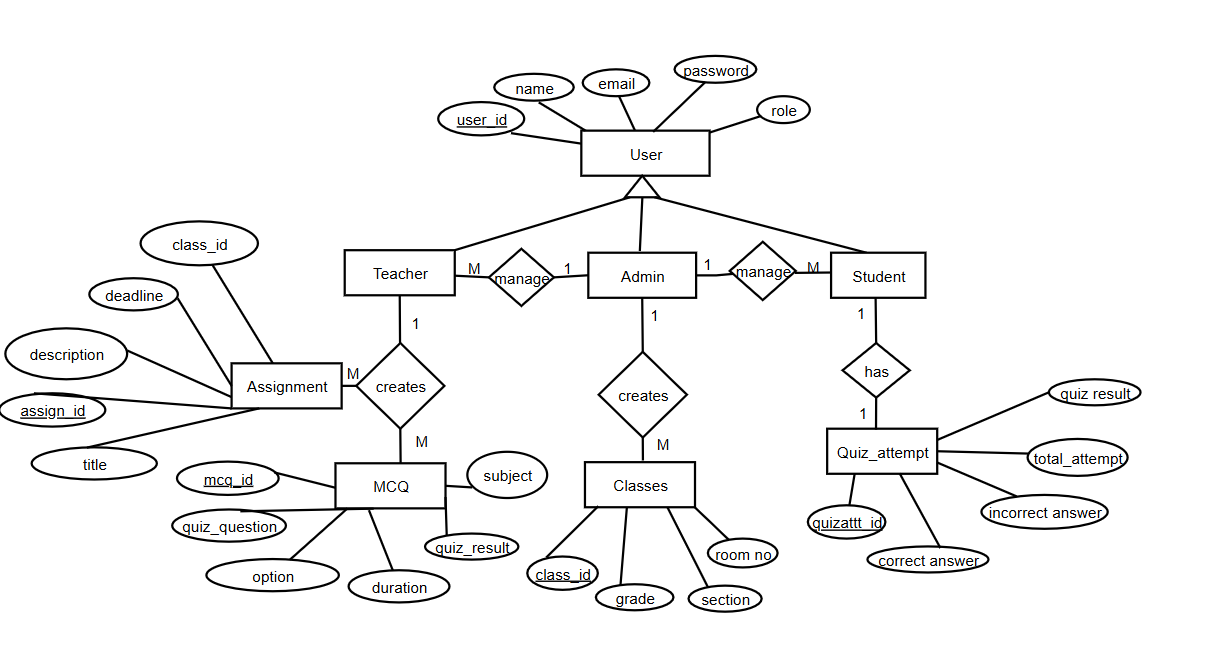


Figure .: ER Diagram

### Process Modeling(DFD)

The Level 0 Data Flow Diagram (DFD) for the EduQuiz system provides a high-level overview of how the platform interacts with its main users: Admin, Teacher, and Student. In this diagram, the central process "EduQuiz" receives management commands and text data from the Admin, who also gets confirmation and management capabilities in return. Teachers interact with the system by creating assignments and quizzes, and in turn receive student performance data. Students are able to receive quiz questions and assignments from EduQuiz and submit their completed assignments and quiz answers back to the system. This context diagram effectively illustrates the primary data flows between the EduQuiz platform and its external entities, capturing the essential interactions without detailing the internal processes or data storage.

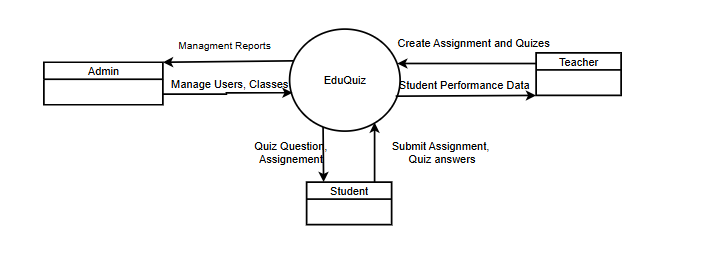


Figure .: Level 0 DFD for EduQuiz

This Level 1 Data Flow Diagram (DFD) for the EduQuiz system breaks down the main functions and shows how data flows between users, processes, and databases. It begins with user login and authentication for Admin, Teacher, and Student. The Admin manages users and classes, with data stored in the User and Class databases. Teachers create and assign quizzes and assignments, which are stored and managed in respective databases. Students receive assignments and quizzes, submit their answers, and the system automatically generates MCQs from uploaded materials.

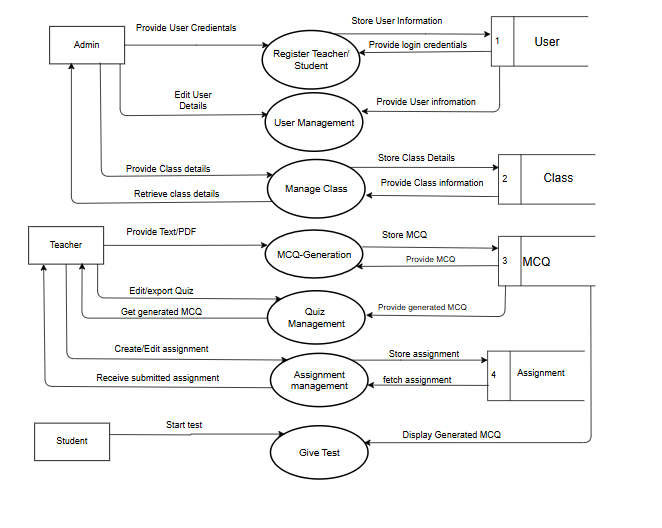
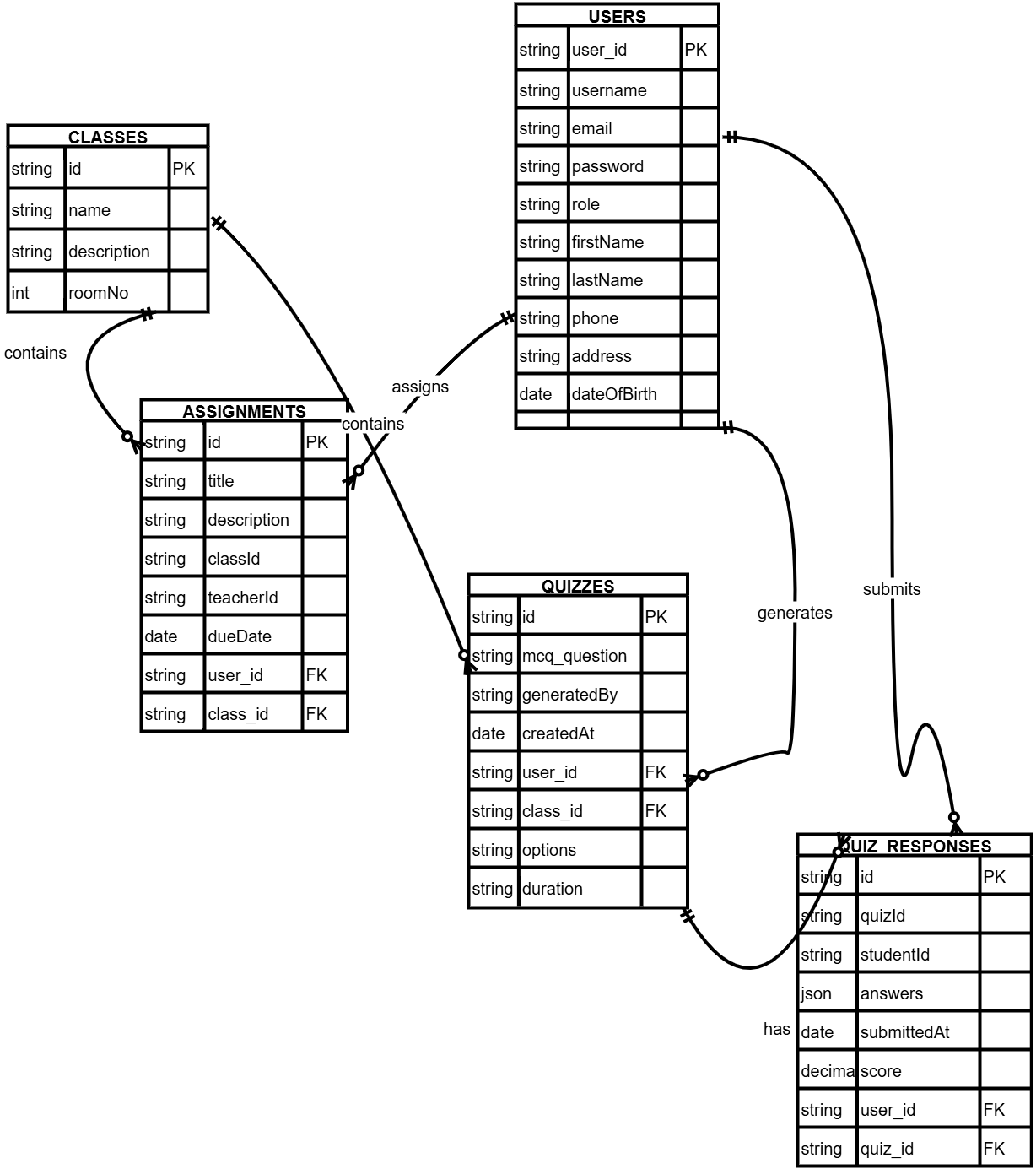


Figure .: Level 1 DFD for EduQuiz

# SYSTEM DESIGN

## Design

### Database Schema Design

The EduQuiz database contains tables to manage users, classes, assignments, quizzes, and submissions. The users table stores details of admins, teachers, and students with id as the primary key. The classes table holds class information, while class\_assignments links users to classes using foreign keys. The assignments table stores assignments created by teachers and links to both classes and users. Student submissions are stored in the submissions table, linked to assignments and users. 

**Figure 4.1: Database Schema Design**

### Form Design

Forms are an essential part of any system through which users are able to give input and submit data. The design for the forms in this system is designed to be friendly, responsive, and efficient. The different types of forms included in the system are user registration, login,

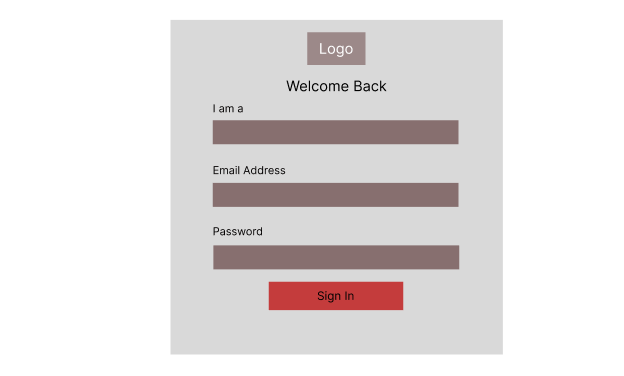
comment submission, input of questions etc. 

Figure 4.2: User Login form

The user login form is provided above. This form is designed to be simple and straightforward, with clear labels and fields that prompt the user to input their data.

### Interface Design

The UI refers to the visible part of the system with which user interacts. A good interface guarantees that user can effectively carry out their tasks without confusion or unnecessary steps. The design should focus on simplicity, clarity and ease of navigation.



Figure .: Dashboard Interface Design

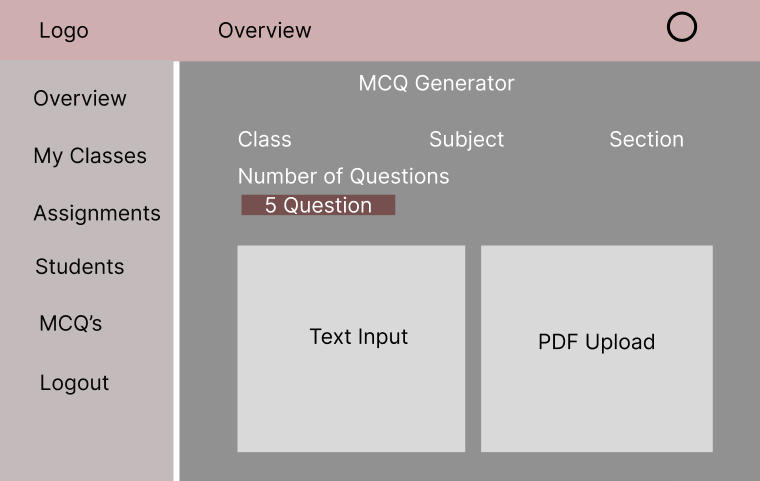


Figure .: MCQ generation Interface

### Dialogue Design

Dialogue are interactive components that enable communication between the system and the user. These should be in the form of alerts, messages, or prompts that provide feedback, request input, or confirm actions.

Example: The below shows the creating of assignment dialogue box.

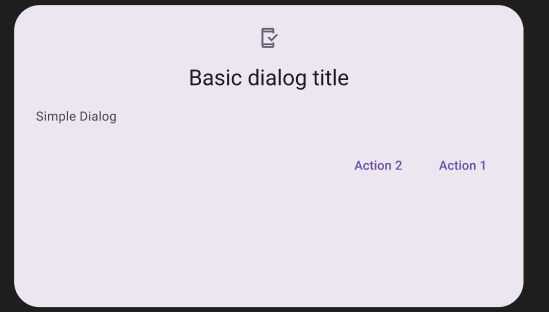


Figure .: Dialogue Design

## Algorithm Details

1. Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a numerical statistic that reflects how important a word is to a document in a collection or corpus. I

The Term Frequency (TF) measures how frequently a term appears in a document:

The Inverse Document Frequency (IDF) measures how important a term is:

Finally, TF-IDF is calculated as the product of TF and IDF:

TF-IDF(t, d, D) = TF(t, d) × IDF(t, D)

Higher TF-IDF scores indicate higher importance of a word in a specific document relative to the entire corpus. This metric can be used to assign scores to words within sentences, which then contribute to the overall importance score of a sentence.

**Purpose**: TF-IDF helps identify keywords in sentences that are likely to represent key concepts or facts, which are then used as answers or blanks in MCQs. TF-IDF also helps generate distractors by comparing the importance scores of other words in the context with the correct answer, allowing selection of words that are related but not identical.

**Example**

 Input sentence: “Nepal’s capital is Kathmandu and it lies in a valley.”

 High TF-IDF score: “Kathmandu” (chosen as the answer.)

 Low TF-IDF score: “is”, “in” (ignored)

1. Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. It measures the cosine of the angle between them. If the vectors are normalized to unit length, the dot product is the cosine similarity. For text summarization, sentences are often represented as vectors (e.g., using TF-IDF weights or word embeddings), and cosine similarity is used to find how similar two sentences are.

The formula for cosine similarity between two vectors A and B is:

Where Ai​ and Bi​ are components of vector A and B respectively. A value of 1 means the vectors are identical, 0 means they are orthogonal (no similarity), and -1 means they are diametrically opposed.

Purpose:

Cosine similarity helps:

* Identify terms or sentences that are similar.
* Generate distractors for MCQ options by choosing terms with higher similarity scores to correct answer.
* Avoid irrelevant words in MCQ points.

**Example**

* Correct answer: "Guido van Rossum"
* Candidate distractor: "Dennis Ritchie" → similarity = 0.7 → selected as a distractor
* Candidate distractor: "Python" → similarity = 0.1 → ignored

1. Rule Based Algorithm

A rule-based algorithm follows predefined if-then conditions to make decisions or build outputs. It does not depend on deep learning but rather makes use of the linguistic or logic rules to manipulate data.

Purpose:

Rule-based approach is used alongside AI to ensure meaningful and contextually accurate questions. Sentences from the text are first tokenized and POS-tagged using NLTK. Key terms such as dates, names, locations, and important concepts are identified using POS tags and simple rules or regex patterns. Questions are then generated by replacing these key terms with blanks or suitable WH-words.

Example

**Input Text:**

"Nepal's capital is Kathmandu. It lies in a valley surrounded by the Himalayas and is known for its historic temples and rich culture."

**Step 1: Identify Key Sentence**

Scan the text for sentences containing proper nouns or important facts.

Selected sentence: "Nepal's capital is Kathmandu."

**Step 2: Extract Key Term / Keyword**

Using POS tagging and rules, extract **“Kathmandu”** as the key term (proper noun, location).

**Step 3: Generate Question**

Replace the key term with a blank or WH-word.

Generated question: "What is the capital of Nepal?"

**Step 4: Generate Distractors using Cosine Similarity**

Candidate words from context: ["Pokhara", "Himalayas", "temples", "valley"]

Compare similarity of candidate words with the correct answer **“Kathmandu”** using TF-IDF + Cosine Similarity.

**Step 5: Select Distractors**

"Pokhara" → similarity = 0.6 → selected

"Himalayas" → similarity = 0.3 → selected

"valley" → similarity = 0.2 → selected

**Step 6: Final MCQ Output**

Question**:** "What is the capital of Nepal?"

Options**:** ["Kathmandu", "Pokhara", "Himalayas", "valley"]

Answer**:** "Kathmandu"

1. T5 Transformer Model (Deep Learning Model)

The T5 (Text-To-Text-Transfer Transformer) model is a powerful NLP model developed by Google. It treats every NLP problem as a text-to-text task, meaning that both input and output are always text, regardless of whether the task is translation, summarization, question answering or classification.

Example: Question Answering

* Input:

Context: “Albert Einstein was a theoretical physicist who developed the theory of relativity.”

Answer: “Albert Einstein”

* Output: “Who developed the theory of relativity?”

1. Named Entity Recognition (NER)

NER is a technique to automatically identify proper nouns and key entities in text, such as people, locations, organizations, and dates. In your project, it is implemented using NLTK’s ne\_chunk() and pos\_tag().

**Purpose:** NER helps the system automatically select meaningful candidate answers for MCQs, ensuring the generated questions are relevant, factual, and contextually accurate.

# IMPLEMENTATION AND TESTING

## Implementation

### Tools Used

Table .: Tool Used

|  |  |
| --- | --- |
| **Category** | **Tools/Technologies** |
| Frontend Tools | React.js, Tailwind CSS |
| Backend Tools | Node.js, Express.js, Python |
| Database | MongoDB |
| Documentation | |  | | --- | |  |   MS Word |
| Diagrammatic Tool | Draw.io, Figma |
| API Testing | Postman |
| Development IDE | |  | | --- | |  |  |  | | --- | | Visual Studio Code (VS Code) | |

### Implementation Details of Modules

The project implements following modules:

Data Input & Reading Module

import fitz

def read\_pdf\_text(file\_path):

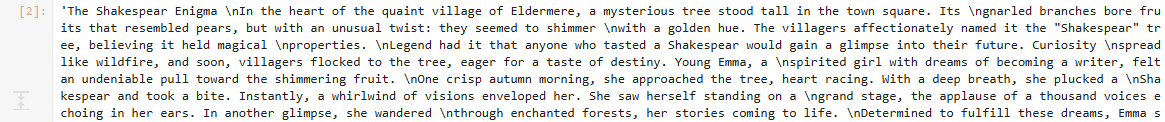
doc = fitz.open(file\_path)

text = ""

for page in doc:

text += page.get\_text()

return text

 Figure .:Output of Data Input & Reading Module

This module handles the extraction of text from uploaded PDFs or documents, providing the raw content that will be used for question generation.

Preprocessing Module

import re

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

from nltk import word\_tokenize

stop\_words = set(stopwords.words("english"))

lemmatizer = WordNetLemmatizer()

def preprocess(text):

text = text.lower()

text = re.sub(r"[^a-z\s]", " ", text)

tokens = word\_tokenize(text)

return [lemmatizer.lemmatize(t) for t in tokens if t not in stop\_words and len(t) > 2]

The preprocessing module cleans the text by converting it to lowercase, removing special characters, stopwords, and applying lemmatization to produce meaningful tokens for further processing.

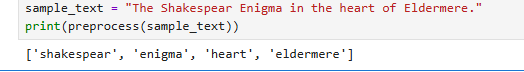


Figure .: Output of Preprocessing Module

TF-IDF & Similarity Module

import numpy as np

def compute\_tf(tokens):

tf = {}

for word in tokens:

tf[word] = tf.get(word, 0) + 1

total = len(tokens)

return {word: count / total for word, count in tf.items()}

def compute\_tf\_idf(tf, idf):

return {word: tf[word] \* idf.get(word, 0.0) for word in tf}

def cosine\_similarity(vec1, vec2):

words = set(vec1.keys()).union(set(vec2.keys()))

v1 = np.array([vec1.get(w, 0.0) for w in words])

v2 = np.array([vec2.get(w, 0.0) for w in words])

return np.dot(v1, v2) / (np.linalg.norm(v1) \* np.linalg.norm(v2) + 1e-8)

It calculates term frequency-inverse document frequency (TF-IDF) vectors and computes cosine similarity to measure the relevance of words, which helps in generating realistic distractors.

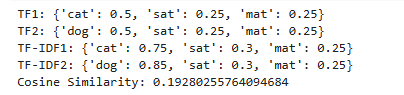


Figure .:Output of TF-IDF & Similarity Module

Generating Distractors

import random

def generate\_distractors(answer, context, idf, required=3):

answer\_tokens = preprocess(answer)

context\_tokens = preprocess(context)

answer\_vec = compute\_tf\_idf(compute\_tf(answer\_tokens), idf)

distractors = []

used\_words = set(answer\_tokens)

for word in set(context\_tokens):

if word in used\_words or word in answer.lower():

continue

word\_vec = compute\_tf\_idf(compute\_tf([word]), idf)

sim = cosine\_similarity(answer\_vec, word\_vec)

if 0.1 < sim < 0.9:

distractors.append((word, sim))

distractors = sorted(distractors, key=lambda x: -x[1])

distractor\_words = [w for w, \_ in distractors]

if len(distractor\_words) < required:

extras = list(set(context\_tokens) - set(distractor\_words) - used\_words)

random.shuffle(extras)

distractor\_words += extras[:required - len(distractor\_words)]

return distractor\_words[:required]

This module creates wrong but plausible answer options for each question by analyzing similarity scores and ensuring diversity among distractors.  
**

Figure .:Output of Generating Distractors

Questing generation Module (Transformers)

from transformers import T5Tokenizer, T5ForConditionalGeneration

tokenizer = T5Tokenizer.from\_pretrained("valhalla/t5-base-qg-hl", local\_files\_only=True)

model = T5ForConditionalGeneration.from\_pretrained("valhalla/t5-base-qg-hl", local\_files\_only=True)

def get\_relevant\_sentence(text, answer):

for sent in sent\_tokenize(text):

if answer.lower() in sent.lower():

return sent

return text

def generate\_question\_with\_fallback(context, answer):

sentence = get\_relevant\_sentence(context, answer)

highlighted = sentence.replace(answer, f"<hl>{answer}</hl>", 1)

input\_text = f"generate question: {highlighted} answer: {answer}"

input\_ids = tokenizer.encode(input\_text, return\_tensors="pt", max\_length=512, truncation=True)

output\_ids = model.generate(input\_ids, max\_length=64, num\_beams=4, early\_stopping=True)

question = tokenizer.decode(output\_ids[0], skip\_special\_tokens=True)

if not question or answer.lower() in question.lower():

question = f"What is {answer}?"

return question

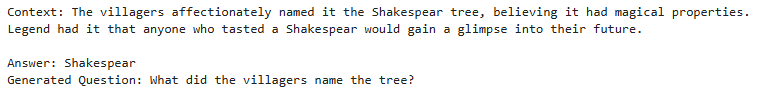


Figure .: Output of Question Generation Module

PDF-based Quiz Generation

router.post("/generate-pdf", upload.single("pdfFile"), async (req, res) => {

if (!req.file) {

return res

.status(400)

.json({ success: false, message: "PDF file required" });

}

try {

const FormData = (await import("form-data")).default;

const fsExtra = await import("fs");

const form = new FormData();

form.append("pdf\_file", fsExtra.createReadStream(req.file.path));

form.append("number\_of\_questions", req.body.numberOfQuestions || 5);

const response = await axios.post(`${FASTAPI\_URL}/generate-pdf`, form, {

headers: {

...form.getHeaders(), },

maxBodyLength: Number.POSITIVE\_INFINITY,

maxContentLength: Number.POSITIVE\_INFINITY,

});

fs.unlink(req.file.path, (err) => {

if (err) console.error("[FILE DELETE ERROR]", err.message);

});

res.json(response.data);

} catch (error) {

console.error("[ERROR IN /generate-pdf]", error.message);

res.status(500).json({ success: false, message: error.message });

}

});

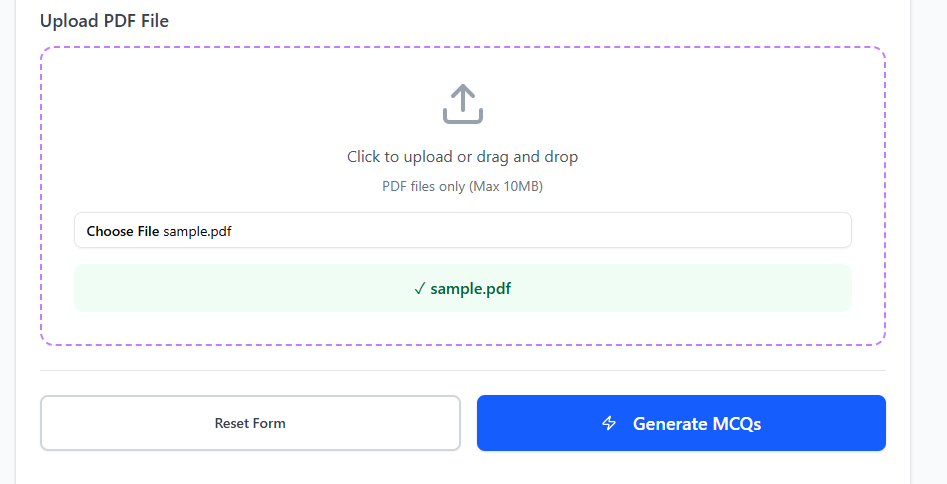


Figure .: Upload PDF

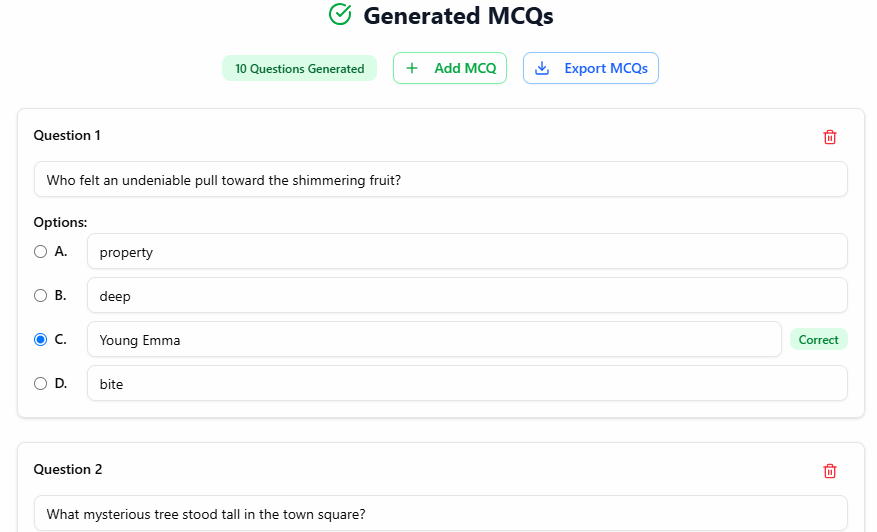


Figure .: Generated MCQ

## Testing

Testing is the process of evaluating and verifying whether the developed software or application works properly or not i.e, whether there is match between the actual results and expected results or not.

### Test Cases for Unit Testing

In unit testing, various modules of the website were tested.

Table . :Test Case for User Registration (Teacher and Student)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Data | Expected Result | Actual Result | Test Result |
| L1 | Admin registers a new teacher | username: JohnDoe email: john@example.com password: 123456 role: teacher | Registration successful | Registration successful | Pass |
| L2 | Admin registers a new student | username: JaneSmith email: jane@example.com password: 123456 role: student | Registration successful | Registration successful | Pass |
| L3 | Register user with missing fields | username: JohnDoe password: 123456 role: teacher | All fields are required | All fields are required | Pass |
| L4 | Register user with short password | username: JohnDoe email: john@example.com password: 123 role: teacher | Password must be at least 6 characters long | Password must be at least 6 characters long | Pass |
| L5 | Register user with existing email | username: NewName email: john@example.com password: 123456 role: teacher | User already exists with that email | User already exists with that email | Pass |
| L6 | Register teacher with valid data | username: Mark email: mark@example.com password: password1 role: teacher | Registration successful | Registration successful | Pass |
| L7 | Register student with valid data | username: Emma email: emma@example.com password: password1 role: student | Registration successful (201), user returned | Registration successful (201), user returned | Pass |

Table .: Test Case for Login

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case  Description | Test Data | Expected  Result | Actual  Result | Test Result |
| R1 | Teacher enters valid login information | email: teacher@example.com  password:1234567890 | Redirects to Teacher Dashboard | Redirects to Teacher Dashboard | Pass |
| R2 | Student enters valid login information | email: student@example.com  password: 1234567890 | Redirects to Student Dashboard | Redirects to Student Dashboard | Pass |

Table .: Test Case for MCQ's Generation by Teacher

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Data | Expected Result | Actual Result | Test Result |
| M1 | Teacher generates MCQ from valid text input | Text: *"Artificial Intelligence is a branch of computer science..."* | MCQs generated and displayed successfully | MCQs generated and displayed successfully | Pass |
| M2 | Teacher generates MCQ from valid PDF upload | PDF: *ai\_basics.pdf (valid, text-based)* | MCQs generated and displayed successfully | MCQs generated and displayed successfully | Pass |
| M3 | Teacher tries to generate MCQ from empty text input | Text: *(empty)* | Error message: "Input text is required" | Error message: "Input text is required" | Pass |
| M4 | Teacher uploads invalid PDF (image-only PDF) | PDF: *scanned\_image.pdf (no extractable text)* | Error message: "No text found in PDF" | Error message: "No text found in PDF" | Pass |

Table .: Test Case for Create Assignment by Teacher

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Data | Expected Result | Actual Result | Test Result |
| A1 | Teacher creates assignment with valid data | title: *"Math Assignment”*  description: *"Algebra basics"* dueDate: *2025-07-01* | Assignment created successfully | Assignment created successfully | Pass |
| A2 | Non-teacher user tries to create assignment | User role: *student* Any valid assignment data | error: "Only teachers can create assignments" | error: "Only teachers can create assignments" | Pass |
| A3 | Teacher tries to create assignment with missing data | title: *""* description: *"Algebra basics"* dueDate: *2025-07-01* | error: "Title is required" | error: "Title is required" | Pass |

Table .:Test for Quiz

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Description | Test Data | Expected Result | Actual Result | Test Result |
| Q1 | Student navigates to “Quiz” section | Navigate to the Available Quiz | Published quiz should be displayed correctly | All the published quiz are displayed | Pass |
| Q2 | Student selects a subject for quiz | Subject “English” | Quiz questions for selected subject displayed | Quiz questions displayed | Pass |
| Q3 | Quiz timer starts | Click “Start Quiz” | Timer begins counting down from predefined time | Timer starts correctly | Pass |
| Q4 | Answer a Question | Select an option | Answer should be recorded | Answer is recorded | Pass |
| Q5 | Skip a Question | Leave a question unanswered | The question should be marked an Unanswered | The question is marked Unanswered | Pass |
| Q6 | Submit Answers | Click “Submit” after answering questions | Quiz should be submitted successfully | Quiz is submitted successfully | Pass |

### Test Cases for System Testing

System testing means the process of verifying that a software system meets the specified requirements and words as intended.

Table .: System Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Test Data** | **Expected Result** | **Actual Result** | **Test Result** |
| S1 | User login (Teacher & Student) | Teacher: [johnteacher@example.com](mailto:johnteacher@example.com), password: 123456  Student: [janestudent@example.com](mailto:janestudent@example.com), password: 123456 | Redirect to respective dashboard | Redirected correctly | Pass |
| S2 | Generate MCQs (Teacher) | Input valid text / upload PDF | MCQs generated and displayed | MCQs generated correctly | Pass |
| S3 | View available quizzes (Student) | Select class, section, subject | Published quizzes display correctly | Quizzes displayed | Pass |
| S4 | Attempt a quiz & record answers (Student) | Answer all MCQs | Answers recorded; score calculated correctly | Answers recorded; score correct | Pass |
| S5 | Admin creates / registers teachers and students | Add new teacher/student with valid info | User successfully created and visible in system | Users created correctly | Pass |
| S6 | View student progress (Student) | Navigate to Progress tab | Total attempts, correct answers, average score displayed | Displayed correctly | Pass |
| S7 | Teacher view progress of all students | Navigate to Reports tab | List of students, attempts, and scores displayed | Displayed correctly | Pass |

## Result Analysis

This section aims to analyze the performance of our model using various evaluation metrics. The model is evaluated using automatic metrics such as F1-Score, ROUGE-1, ROUGE-2, ROUGE-L, and BLEU. These metrics are designed to measure the quality of the generated text, particularly in tasks like question generation, answer identification, and similarity assessment.

### Quantitative Analysis

Quantitative Analysis refers to the process of interpreting and understanding phenomena through numerical data and statistical methods. In the context of evaluating NLP models, quantitative analysis involves assessing model performance using defined metrics to obtain measurable and comparable results.

1. ROUGE Score

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is an evaluation metric widely used for text summarization and generation tasks. It compares the generated text with a human-written reference text using n-gram overlap statistics, capturing both lexical and structural similarity.

* ROUGE-1: 0.6667
* ROUGE-2: 0.2429
* ROUGE-L: 0.5833

**Interpretation:**

* ROUGE-1 (0.6667): Measures overlap of individual words (unigrams) between the generated text and the reference. A score of 0.667 indicates that the model successfully retains many of the key words from the reference questions, reflecting good performance in capturing essential content.
* ROUGE-2 (0.2429): Measures overlap of two consecutive words (bigrams). The score is lower than ROUGE-1, which is expected, as it is more difficult for the model to generate the exact word sequences from the reference. Despite being lower, this score shows that the model preserves some phrase-level coherence.
* ROUGE-L (0.5833): Measures the longest common subsequence (LCS) between generated and reference texts, capturing structural similarity. A score of 0.583 suggests that while the model may rephrase sentences, it maintains a reasonable similarity in overall structure and meaning.

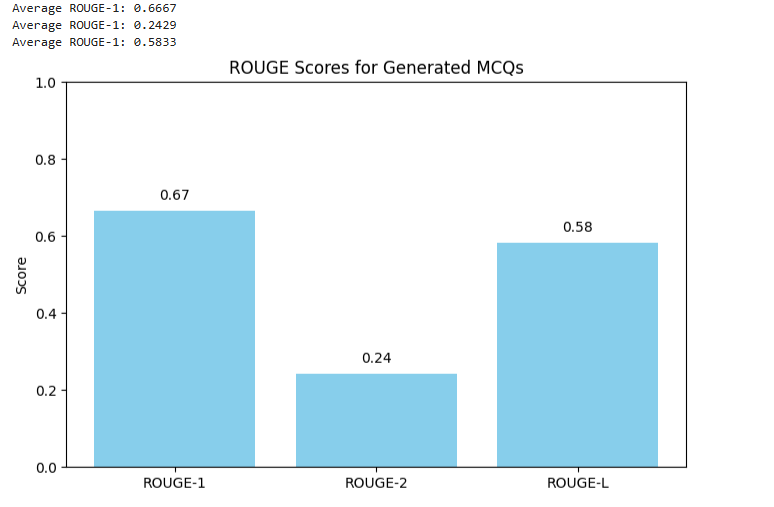


Figure .:ROUGE Score

1. BLEU Score

The BLUE score (Bilingual Evaluation Understudy) is a precision-based metric that evaluates the quality of machine generated text by comparing n-grams overlaps with reference text. The models average BLUE score is 0.62.

**Interpretation:**

The score indicates that the model is generating text that is very similar to the reference in terms of n-gram overlap. This indicates that model is capturing a lot of the information and wording in human summaries.

1. F1 Score

The F1 score is the harmonic mean of precision and recall, measuring how well the generated text captures the reference text in terms of word overlap. An average F1 score of 0.73 indicates that the model is generating questions that are very similar to human-written reference questions, balancing both the accuracy of the generated words (precision) and the coverage of reference words (recall).

**Interpretation:**

A high F1 score reflects that the model captures most of the important information from the reference. It shows that the word choice and structure of generated questions closely resemble human-written ones.

### Qualitative Analysis

This section evaluates the practical performance and usability of the system beyond numeric metrics. It focuses on how effectively the system fulfills its intended functionality and user experience.

* Correct Generation of MCQs / Classes

The system successfully generates questions for the provided context and answers. Classes are correctly identified and grouped, ensuring organized questions.

* Data Storage & Management

Generated questions, options, and answers are stored reliably in the database or system storage. Users can retrieve previously generated questions without loss or corruption.

* System Features & Functionality

Supports creation of multiple-choice questions with at least 3 distractors. Ensures that distractors are plausible, reducing the chance of obvious wrong answers. Allows users to view, edit, or delete generated questions.

* User Interaction & Usability

The system interface is intuitive for teachers/admins: generating, storing, and managing questions is straightforward.

* Error Handling & Robustness

The system handles edge cases (e.g., empty input, short context) properly, either by fallback methods or default question templates.

# CONCLUSION AND FUTURE RECOMMENDATIONS

## Conclusion

The EduQuiz project successfully creates a web-based platform using the MERN stack that facilitates seamless interaction among admins, teachers, and students. The admin efficiently manages users and classes, while teachers can create assignments, monitor student progress, and generate multiple-choice questions automatically from text or PDF using TF-IDF, T5 transformers, and cosine similarity techniques. Students benefit from easy access to assignments and quizzes, enhancing their learning experience. This system streamlines educational assessment and management by delivering a flexible, user-centric platform designed to support the needs of contemporary e-learning environments.

## Future Recommendations

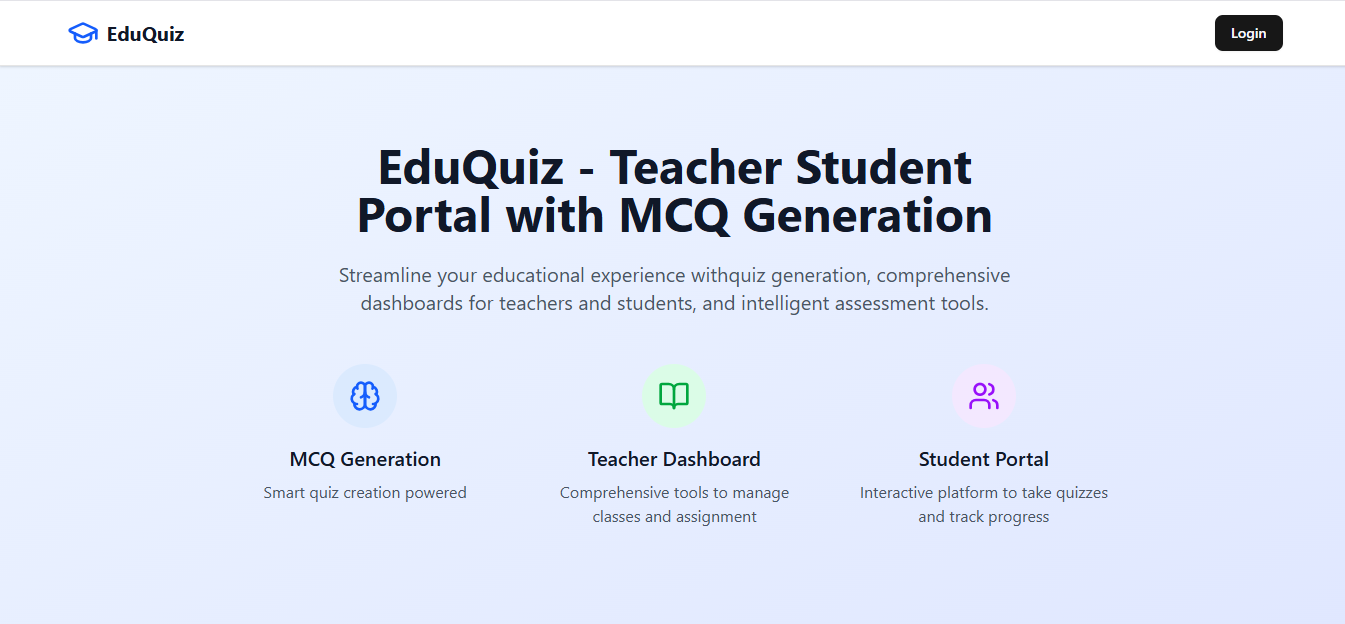
* Add AI features for personalized learning and automated grading.
* Improve security with multi-factor authentication.
* Integrate pre-trained language models (eg- BERT,GPT) to improve the quality and variety of automatically generated MCQ’s.
* Offer multilingual support.
* Support more question types beyond MCQs.

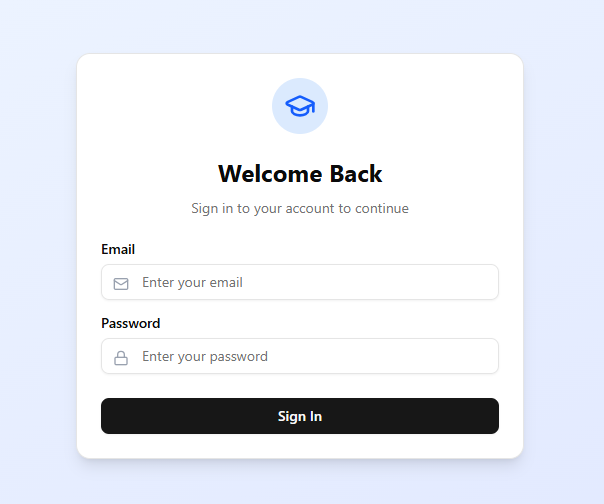
# REFERENCES

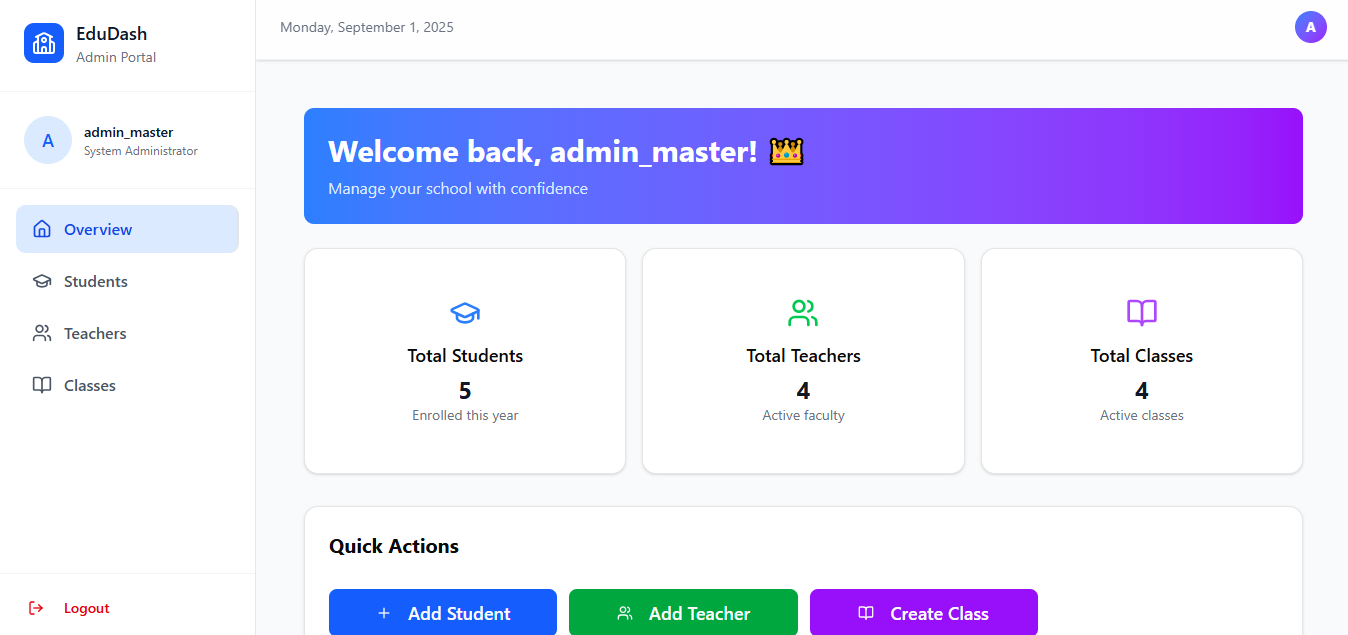
|  |  |
| --- | --- |
| [1] | S. S. P. T. M. M. F. W. B. C. S. d. S. A. F. &. O.-S. T. Mucciaccia, “ Automatic Multiple-Choice Question Generation and Evaluation Systems Based on LLM,” *Proceedings of the 31st International Conference on Computational Linguistics,* pp. 2246-2260, 2025. |
| [2] | F. M. a. M. Asif, “Automatic computer science domain multiple-choice questions generation based on informative sentences.,” *PeerJ Computer Science 8,* 2022. |
| [3] | T. P. B. &. S. U. Alsubait, “Ontology-based multiple choice question generation,” *KI-Künstliche Intelligenz 30,* pp. 183-188, 2016. |
| [4] | F. M. A. H. A. S. A. F. A. O. A. a. Y. Y. G. Maheen, “Automatic computer science domain multiple-choice questions generation based on informative sentences,” *PeerJ Computer Science 8,* 2022. |
| [5] | N. C. N. K. A. D. R More, “Student Portal of College,” 2022. |
| [6] | A. P. K. K. Steven Azeka, “Building a school data portal using open data,” *Heliyon,* vol. 8, no. 9 e10517, September 2022. |
| [7] | S. R. G. R. B. &. T. S. G. Bharamagoudar, “Web based student information management system.,” *International Journal of Advanced Research in Computer and Communication Engineering,* pp. 2342-2348, 2013. |
| [8] | P. R. J. T. N. K. Y. A. &. H. V. (. M. Rao, “Generating QA from Rule-based Algorithms,” *International Conference on Electronics and Renewable Systems (ICEARS),* pp. 1697-1703, 2022. |

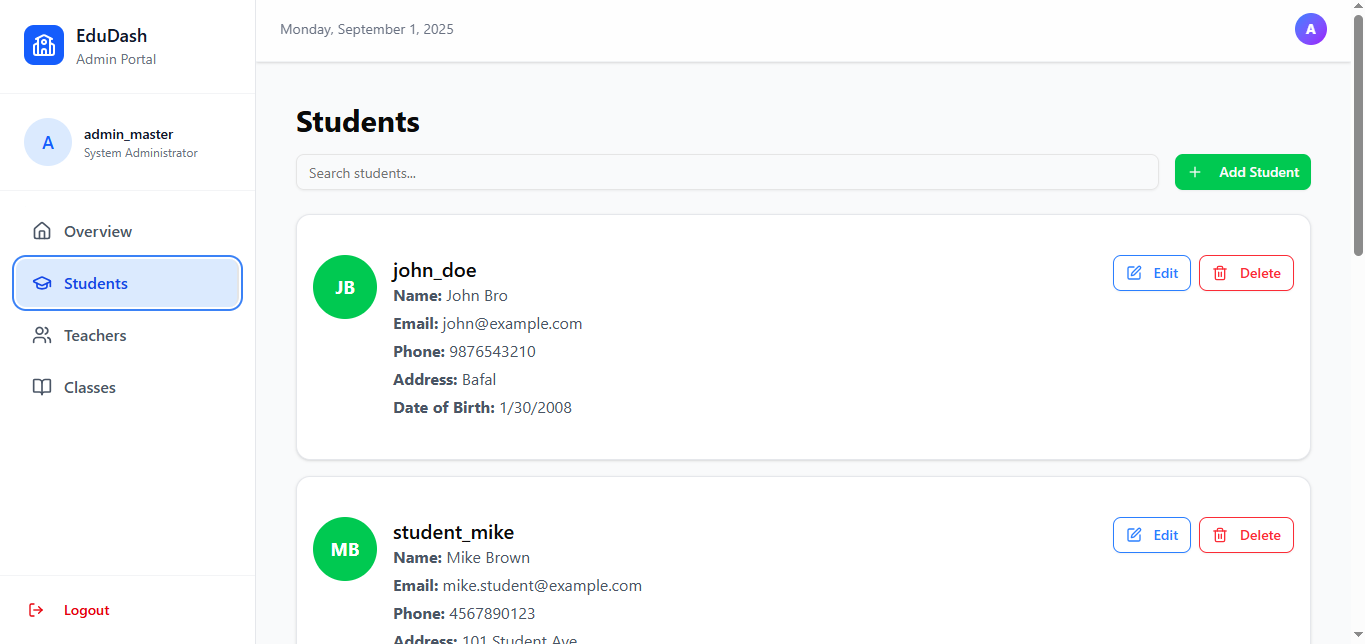
# APPENDICES

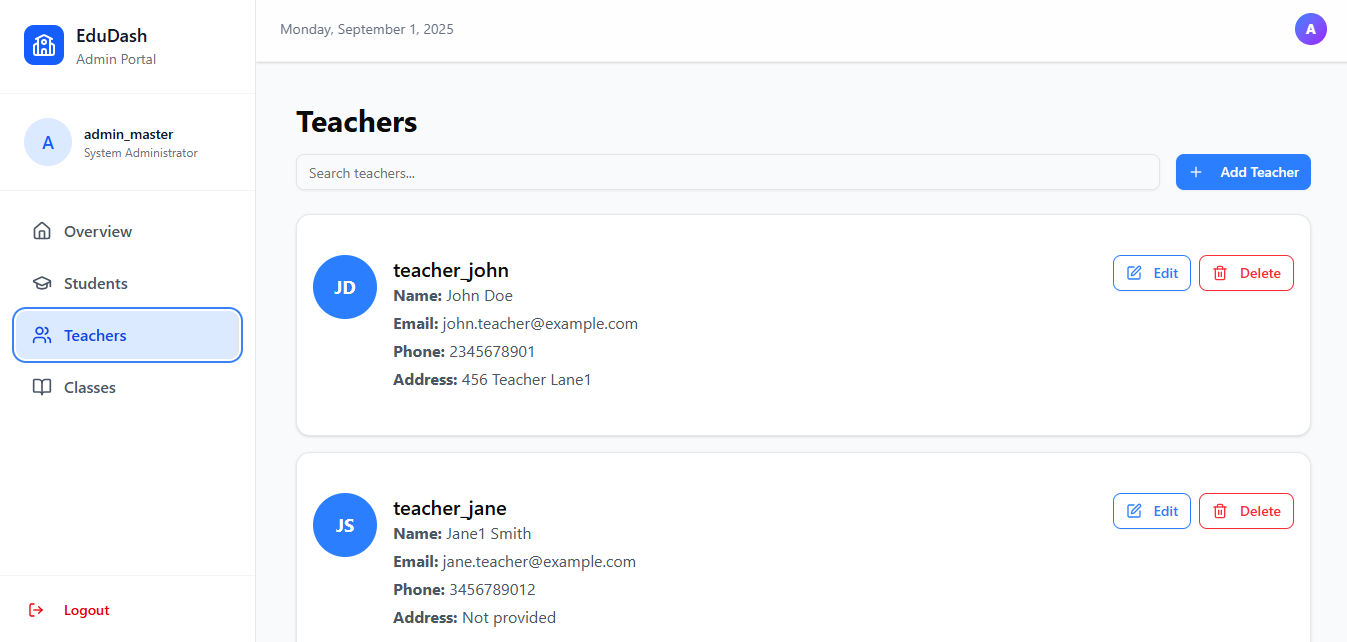
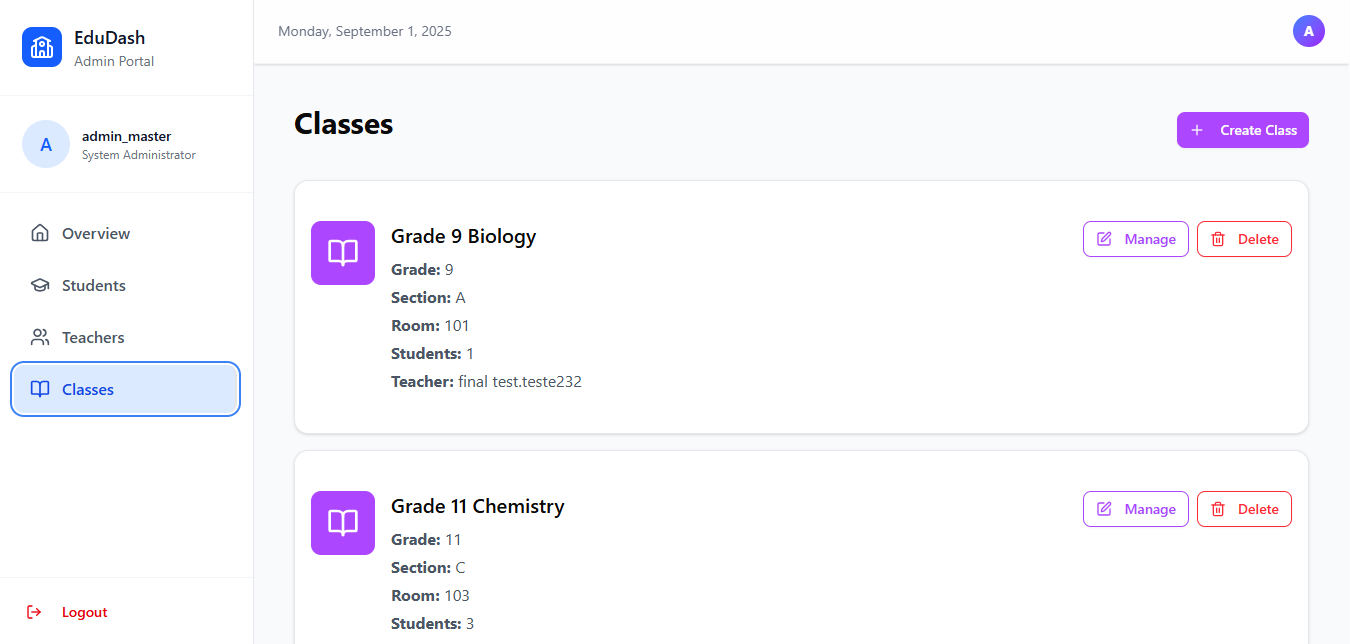
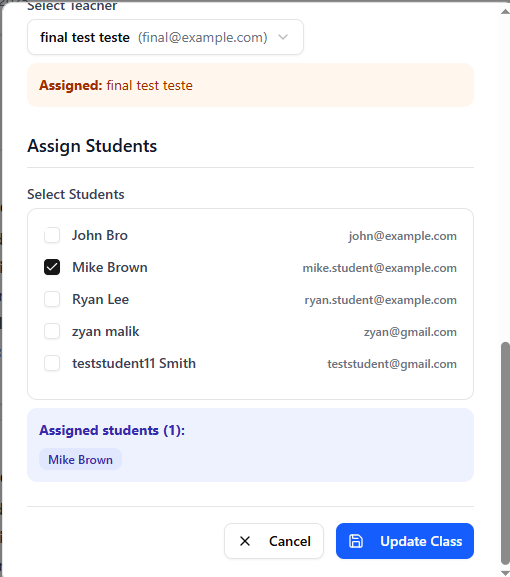
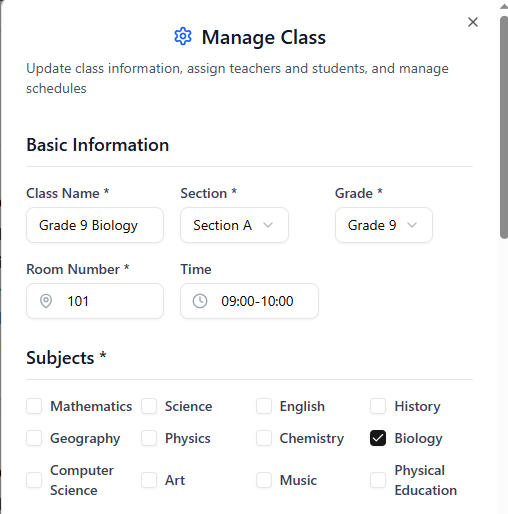
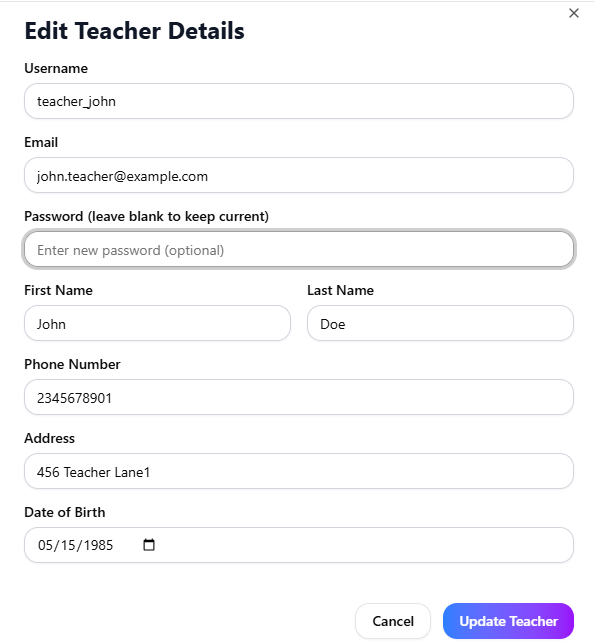
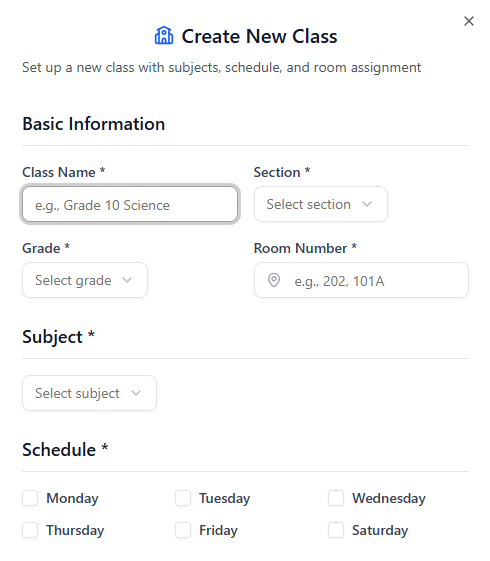
**Home Page:**

****

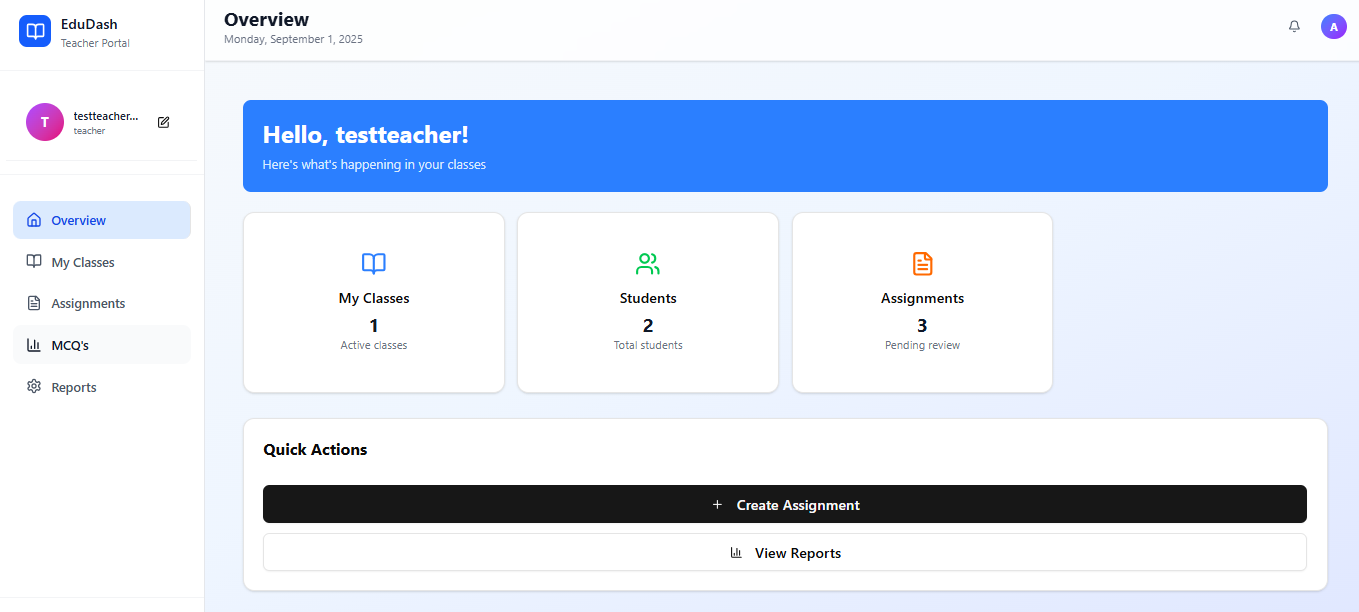
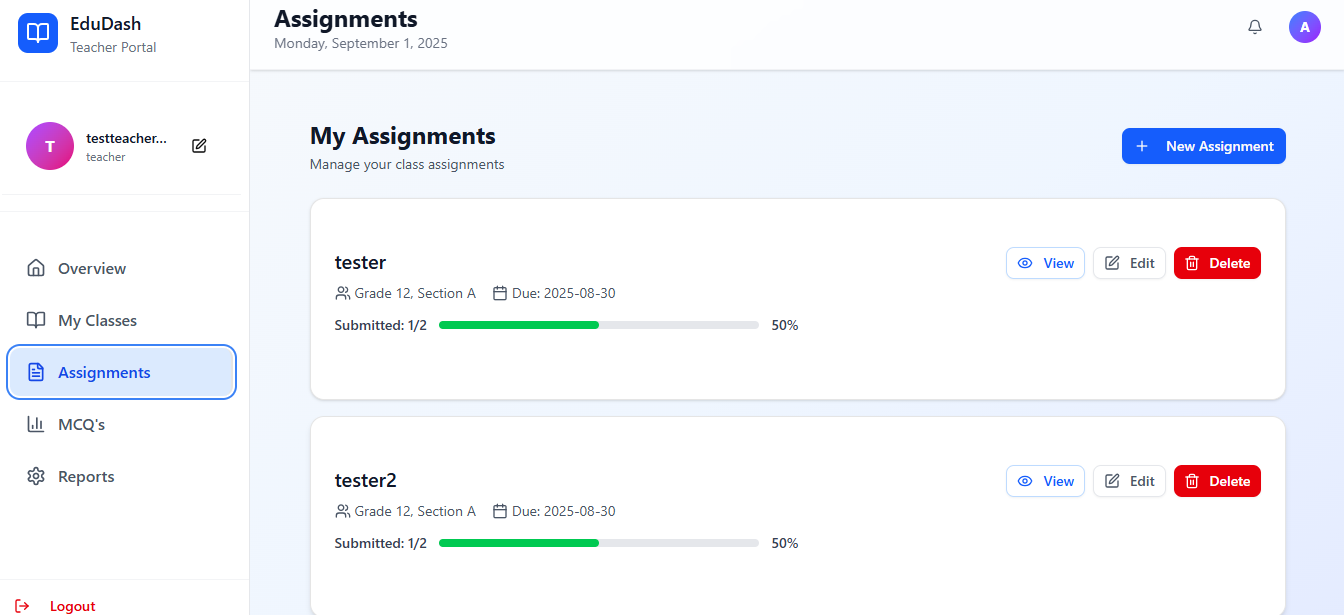
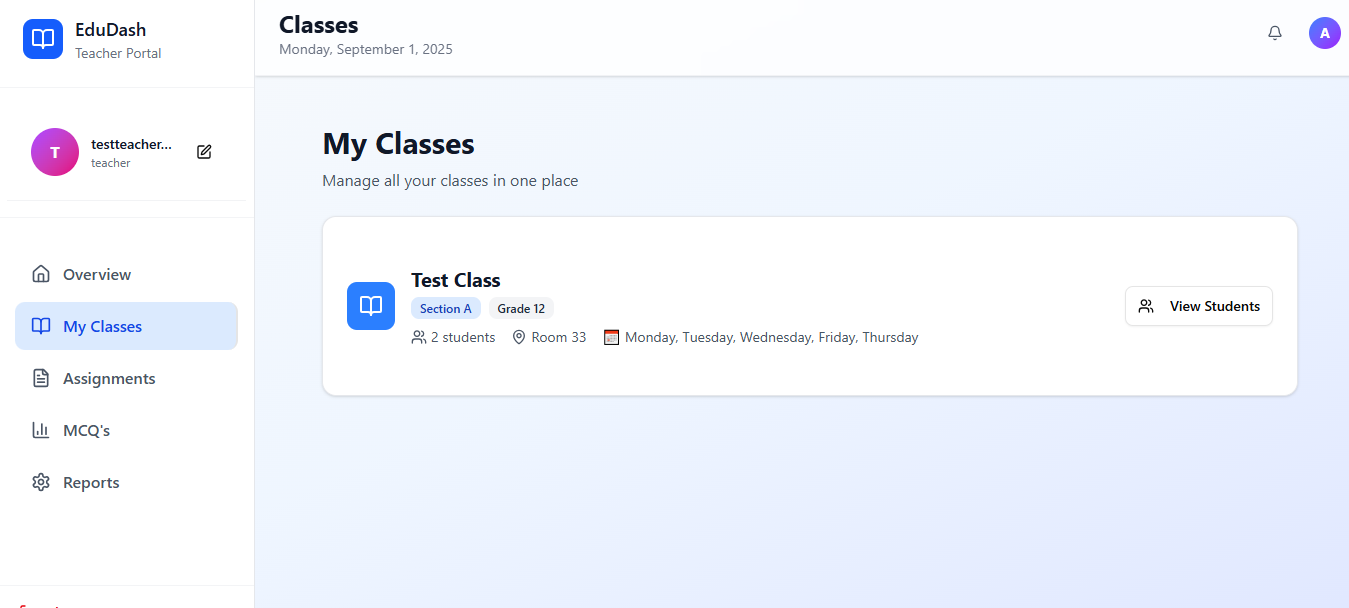
**Login Page: **

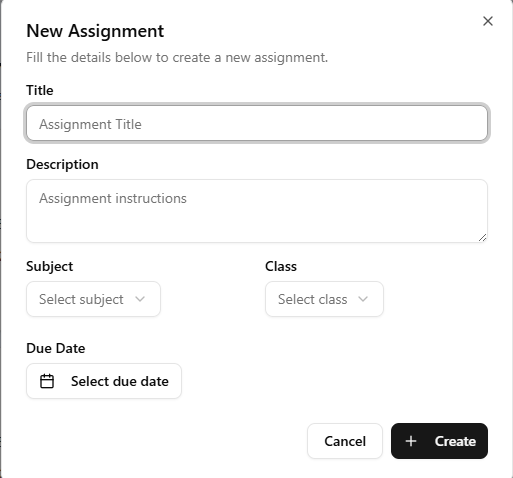
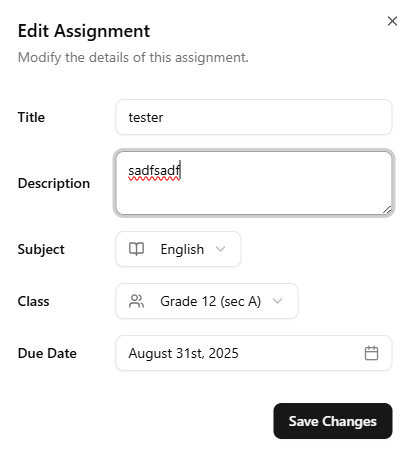
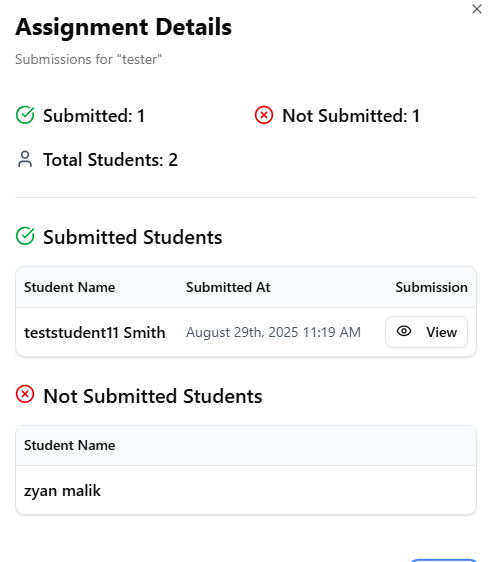
**Admin Dashboard:** 

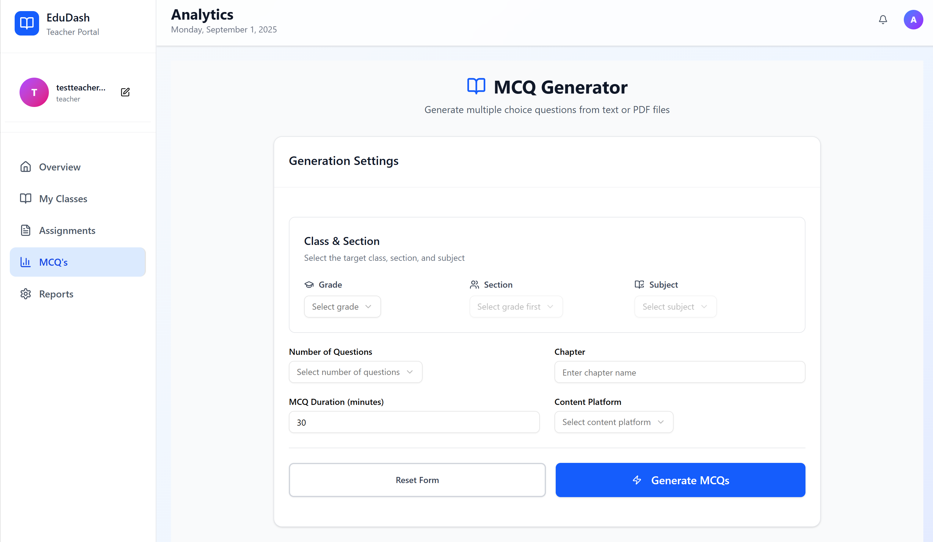
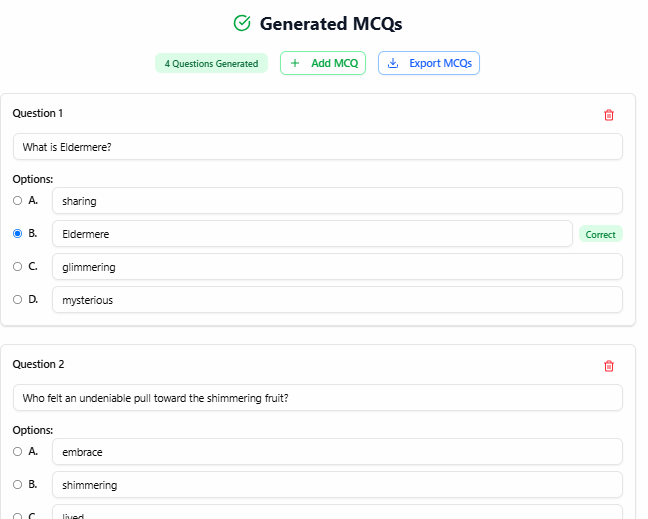
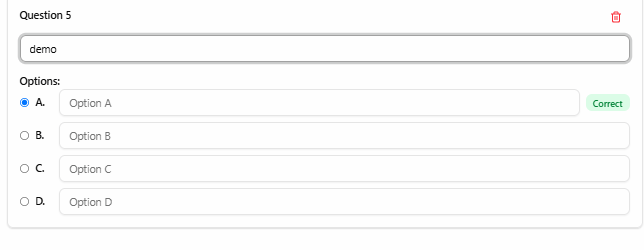
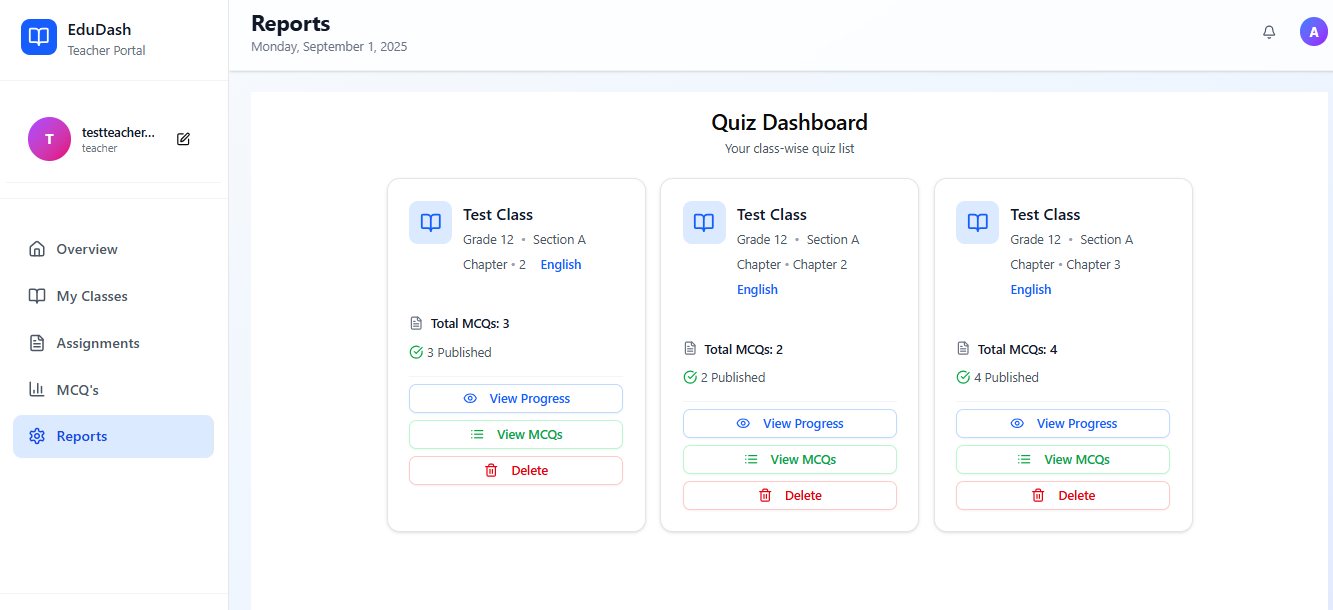
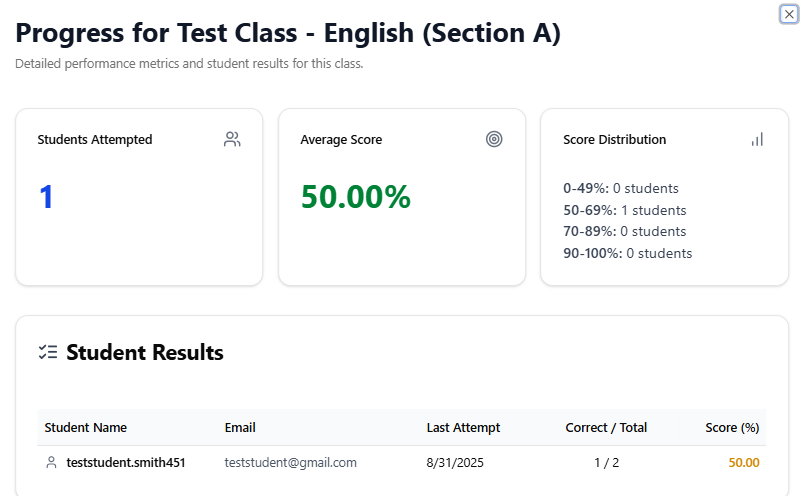
****

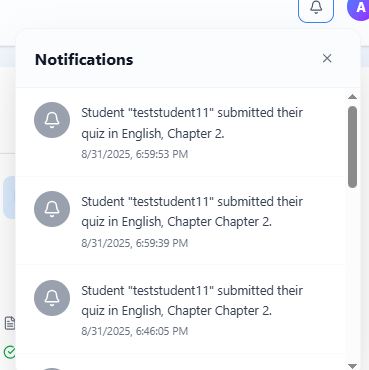
**** ****  

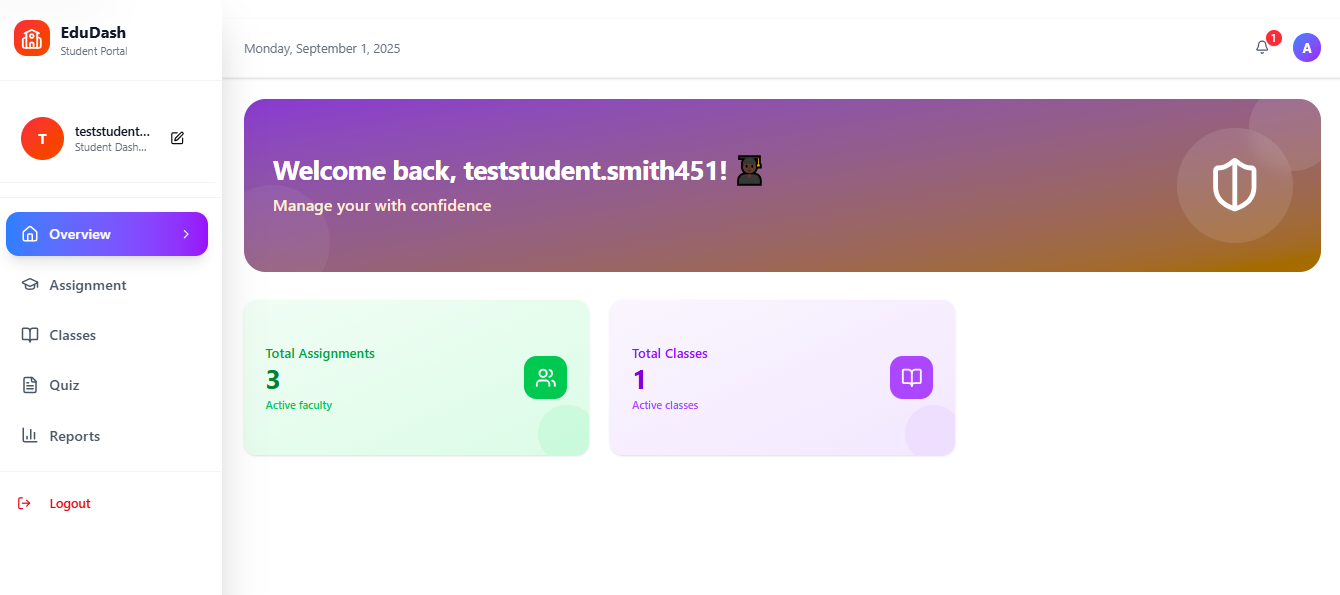
**Teacher Dashboard:**

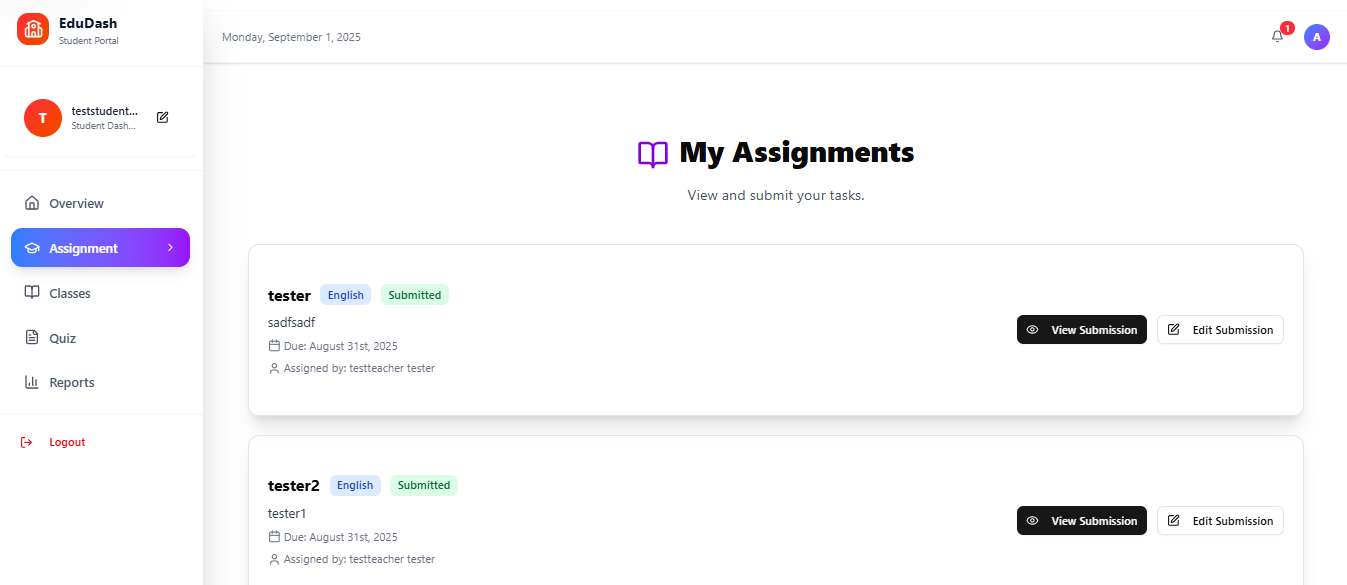
**** ****

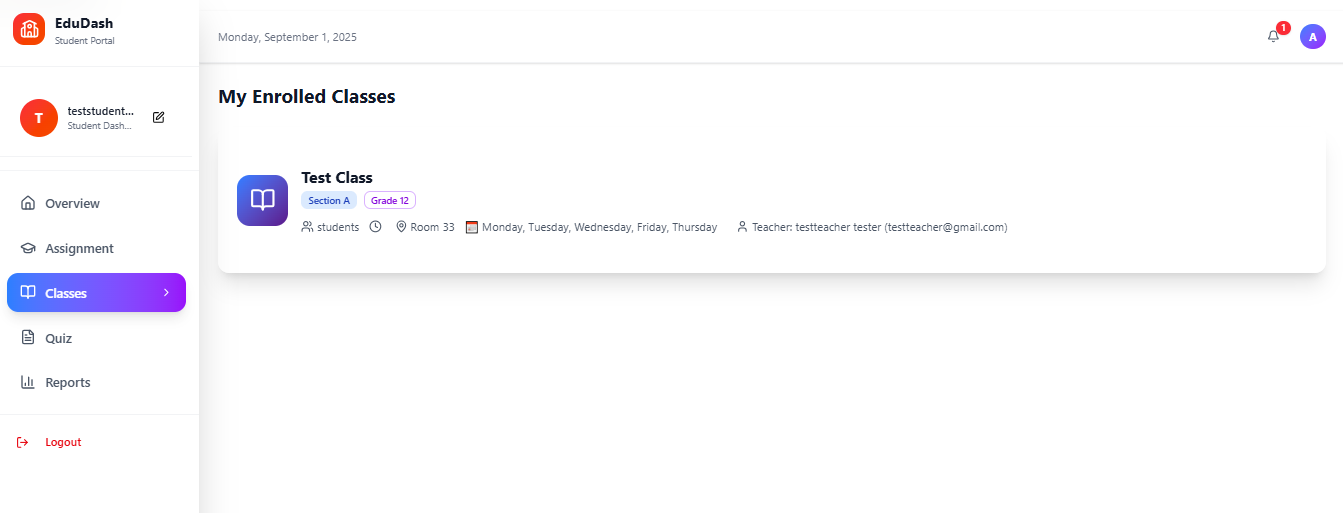
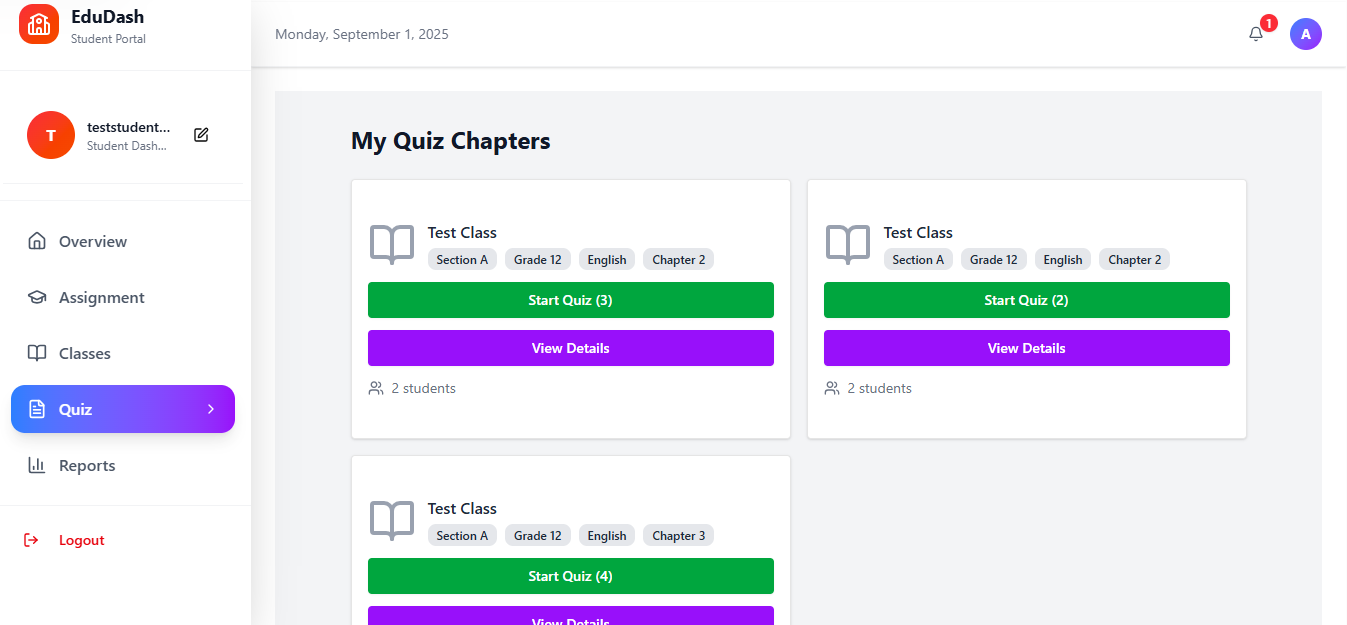
 ****



**Student Dashboard:** ****

****

**** **** 