Project A.C.E- Analyze. Connect. Execute.

Submitted By

Student Name	**Student ID**
MD. Abdullah Khan	232-15-425
Sazzad Islam	232-15-835
Saif Ahmed	232-15-636

MINI LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course

CSE312: Database Management System Lab in the Computer Science and Engineering Department

DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

18 August 2025

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of **Israt Jahan**, **Lecturer (Senior Scale)**, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:

____

Israt Jahan

Lecturer (Senior Scale)

Department of Computer Science and Engineering,

Submitted by

______________________________ Student Name Student ID: Dept. of CSE, DIU	
______________________________ Student Name Student ID: Dept. of CSE, DIU	______________________________ Student Name Student ID: Dept. of CSE, DIU

COURSE & PROGRAM OUTCOME

The following course have course outcomes as following:

Table 1: Course Outcome Statements

CO's	**Statements**
CO1	**Define** and **Relate** classes, objects, members of the class, and relationships among them needed for solving specific problems
CO2	**Formulate** knowledge of object-oriented programming and Java in problem solving
CO3	**Analyze** Unified Modeling Language (UML) models to **Present** a specific problem
CO4	**Develop** solutions for real-world complex problems **applying** OOP concepts while evaluating their effectiveness based on industry standards.

Table 2: Mapping of CO, PO, Blooms, KP and CEP

CO	**PO**	**Blooms**	**KP**	**CEP**
CO1	PO1	C1, C2	KP3	EP1, EP3
CO2	PO2	C2	KP3	EP1, EP3
CO3	PO3	C4, A1	KP3	EP1, EP2
CO4	PO3	C3, C6, A3, P3	KP4	EP1, EP3

The mapping justification of this table is provided in section **4.3.1**, **4.3.2** and **4.3.3**.

Table of Contents

Chapter 1

Introduction

This chapter introduces the research background, problem statement, motivations, objectives, related works, gaps in existing studies, and expected outcomes of the project.

1.1 Introduction

Online quizzes are a simple and fast way to test knowledge. They are widely used in schools, colleges, job exams, and learning websites. However, many quiz systems are not user-friendly. Some do not save results properly, while others lack good data management.

Project A.C.E (Analyze. Connect. Execute) is designed to solve these problems. It is a web-based quiz system where users can take quizzes and instantly see their scores. All quiz data is stored in a structured relational database. The project uses Django for the backend and HTML/CSS for the frontend.

The main goal is to build a quiz system that is easy to use, works smoothly, and stores data properly.

1.2 Motivation

As a group of database course students, we were interested in building a practical, real-world web application that uses a proper database system. Online quizzes are very popular and useful, but many existing systems are not easy to use or do not manage data well. This motivated us to create our own quiz platform where we could apply our skills in Python (Django), database design, and web development. Working on this project as a team also helped us improve our collaboration, planning, and problem-solving skills.

1.3 Objectives

Our group had the following objectives:

- To develop a full-stack online quiz system.
- To allow users to take quizzes and receive instant feedback.
- To store all data (users, questions, answers, scores) in a relational database.
- To use Django as the backend framework to connect all parts.
- To provide admin features for managing quizzes and tracking performance.

- To improve our technical and teamwork skills through this project.

1.4 Feasibility Study

We studied popular quiz platforms like Kahoot, Quizizz, and Google Forms. These tools are effective, but most are either commercial or not customizable for academic learning purposes.

Some open-source projects also exist, but they often lack proper backend logic or are too complex for students. Our project is designed to be simple, educational, and useful for learning full-stack development — especially focusing on database management and system integration.

1.5 Gap Analysis

There is a clear gap in simple, customizable quiz systems designed for learning purposes. Most existing systems do not explain how data flows between the user, backend, and database.

Our project fills this gap by building a full-stack quiz platform from scratch, where every part — frontend, backend, and database — is connected and easy to understand. It is especially useful for students learning web development and database systems.

1.6 Project Outcome

At the end of this project, we achieved the following:

- A working quiz web application with full-stack features.
- A clean, relational database to store all quiz data.
- A user interface for taking quizzes and viewing results.
- Admin features for managing quizzes and user scores.
- Improved skills in Django, SQL, web development, and teamwork.

Chapter 2

Proposed Methodology/Architecture

This chapter explains the planning and design of the project. It covers requirement analysis, system architecture, UI design, and the overall project plan.

2.1 Requirement Analysis & Design Specification

2.1.1 Overview

Before development, we analyzed the requirements of the quiz system. The system needed to support user registration, quiz-taking, result viewing, and admin data management.

We identified **functional** and **non-functional** requirements:

Functional Requirements

- **User Authentication:** Register/login with different roles (Admin, Teacher, Student).
- **Quiz Taking (Student):** Attempt quizzes and get instant results.
- **Question Management (Admin/Teacher):** Add, edit, delete, and categorize questions.
- **User Score Management (Admin/Teacher):** View and filter scores, generate reports.
- **Quiz Management (Admin/Teacher):** Create quizzes, set time limits, passing scores, randomize questions.
- **Dashboard & Analytics (Admin/Teacher):** View class-wise performance, export reports.
- **User & Class Management (Admin/Teacher):** Manage students, assign teachers, track progress.
- **Feedback & Review (Teacher/Student):** Students review quiz attempts; teachers give feedback.

Non-Functional Requirements

- The system must be user-friendly and responsive.
- Data should be stored efficiently and securely.
- The system should run smoothly on multiple devices.

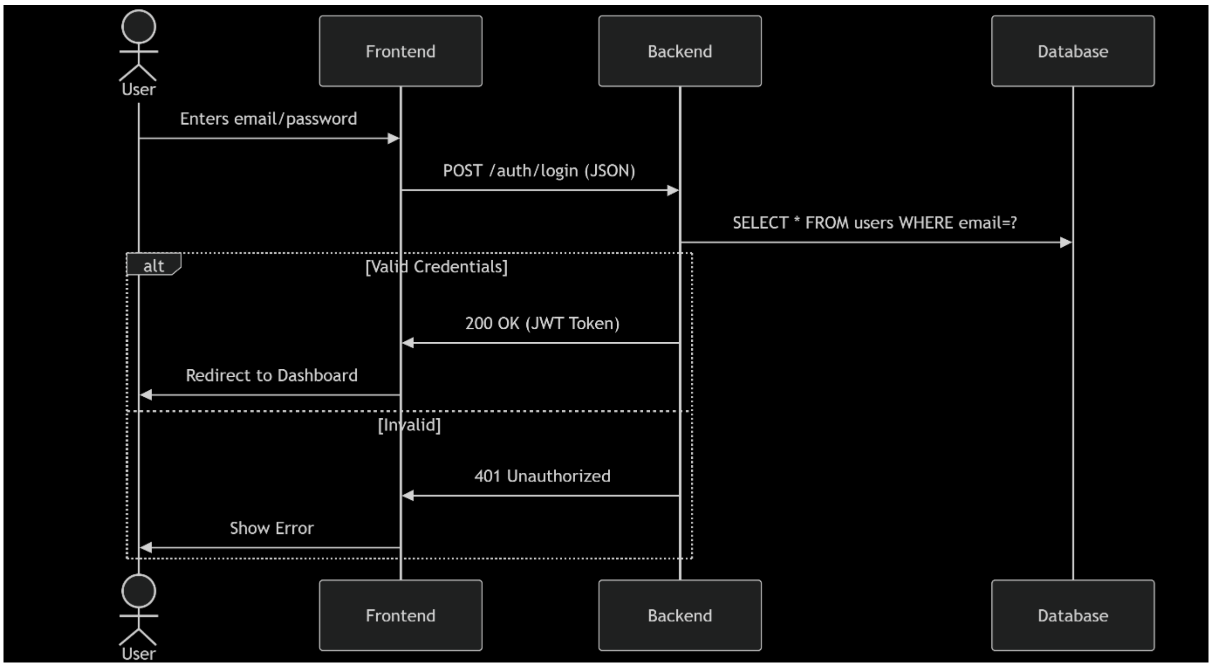
2.1.2 Proposed Methodology

We followed a **modular architecture** using Django's **Model-View-Template (MVT)** pattern:

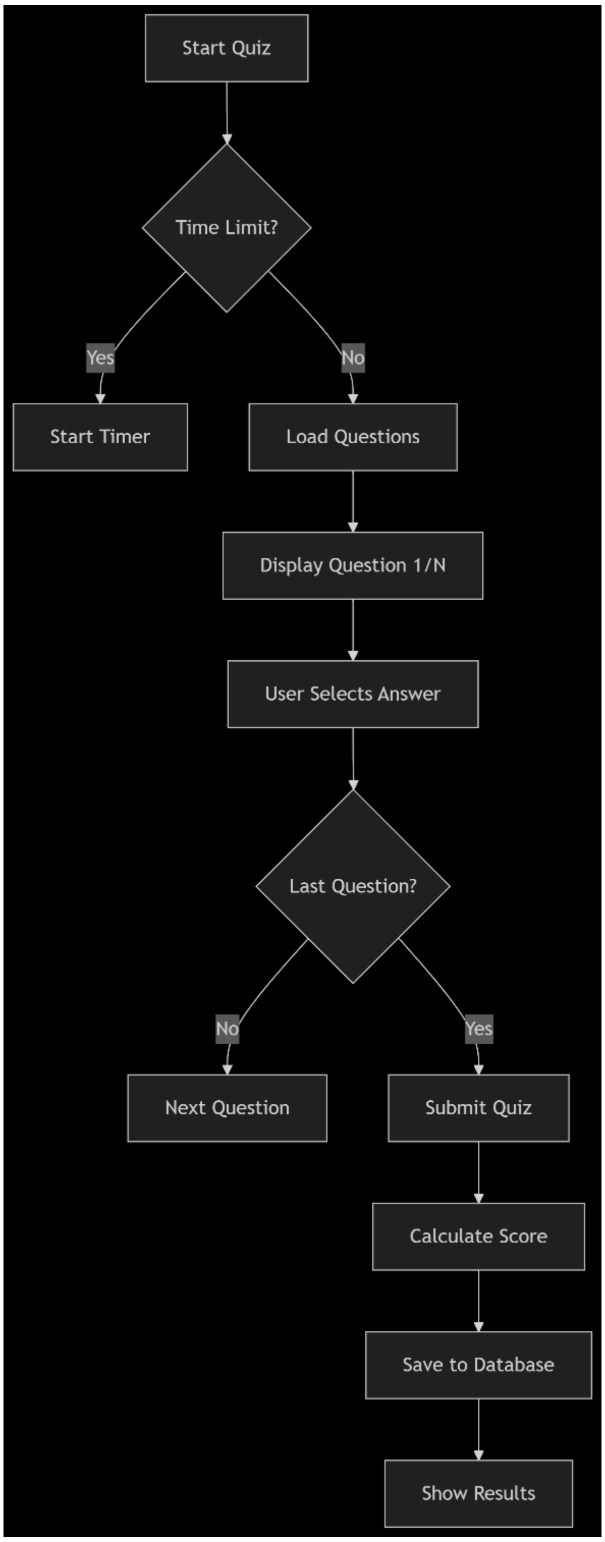
- **Frontend (View + Template):** HTML, CSS, Django templates.
- **Backend (Controller):** Django views and business logic.
- **Database (Model):** Stores users, questions, answers, and results.

System Design

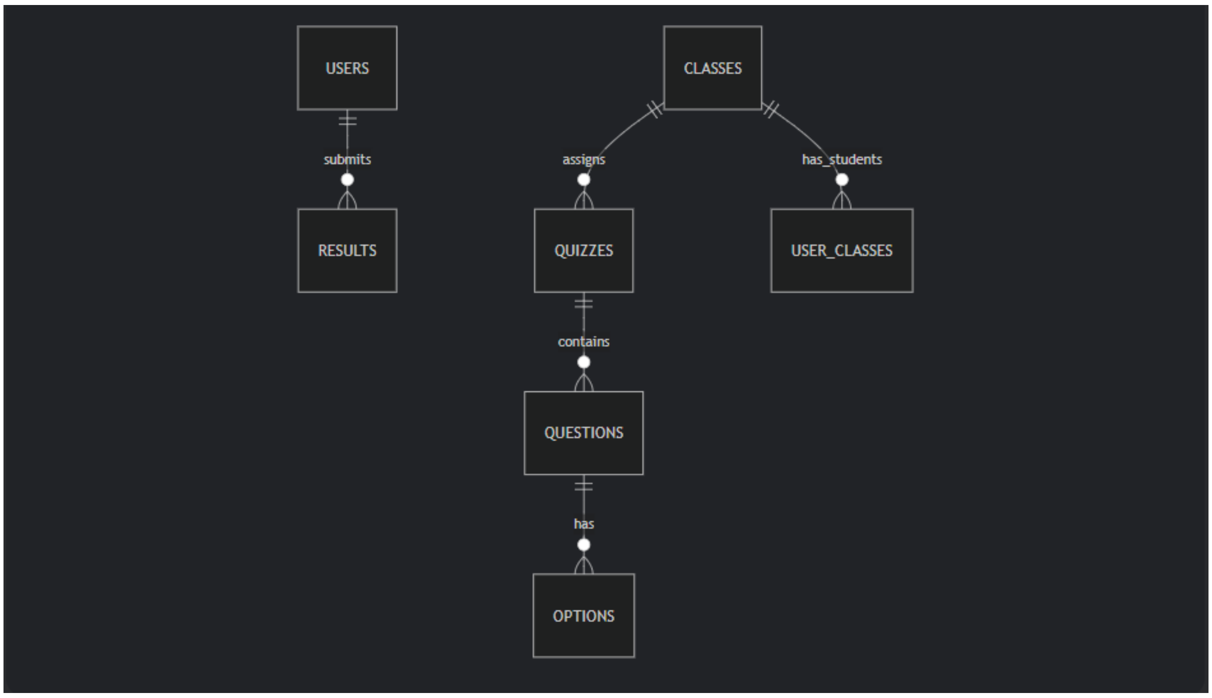
1. User Authentication Flow



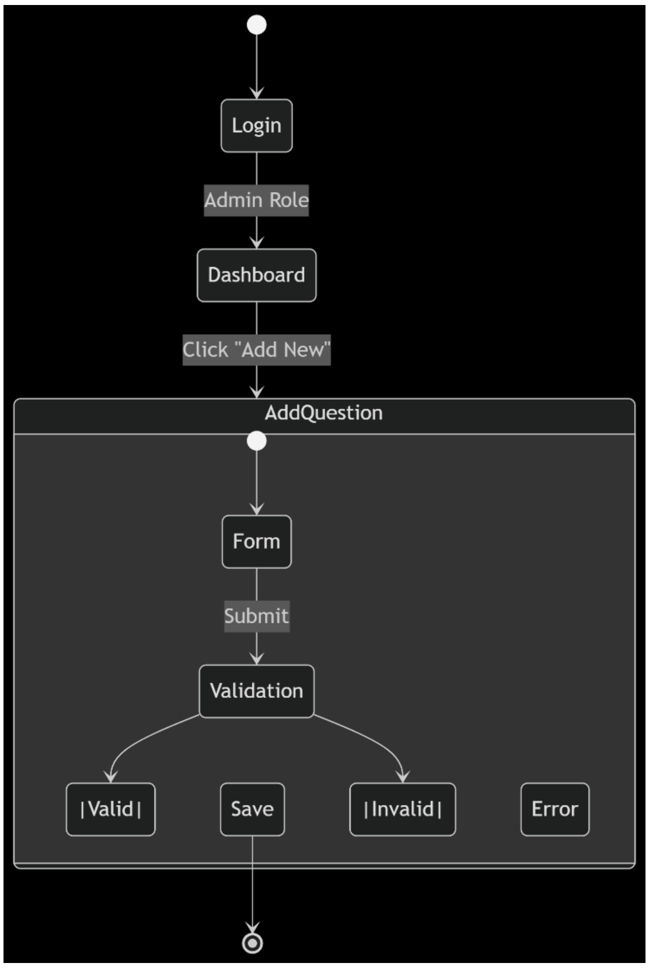
2. Quiz Taking Process



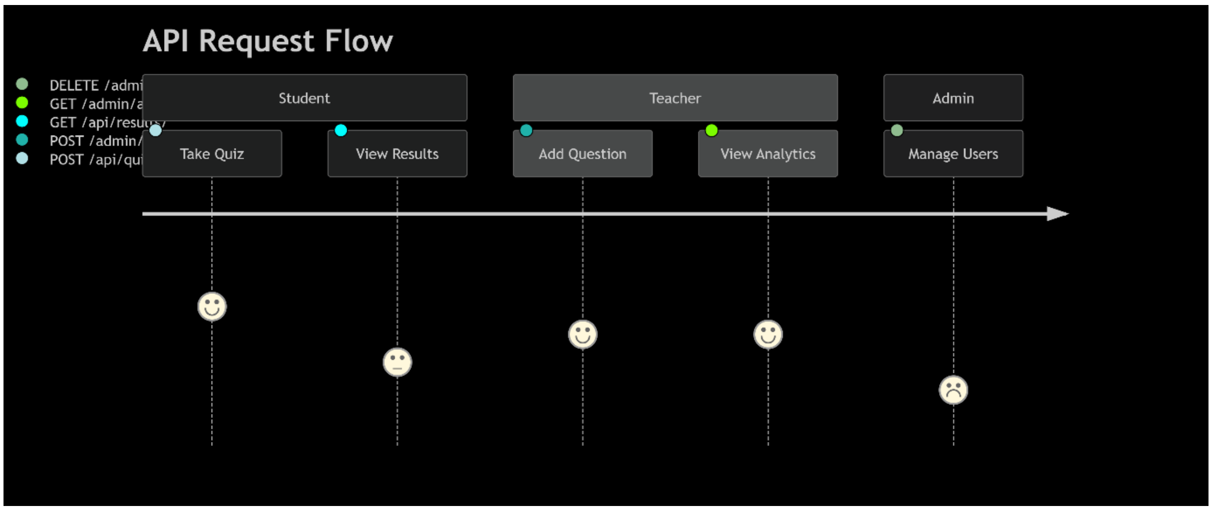
3. Database Schema with all tables, relationships, and field specifications



4. Admin Question Management



5. API Endpoint Map



6. System Security Layers

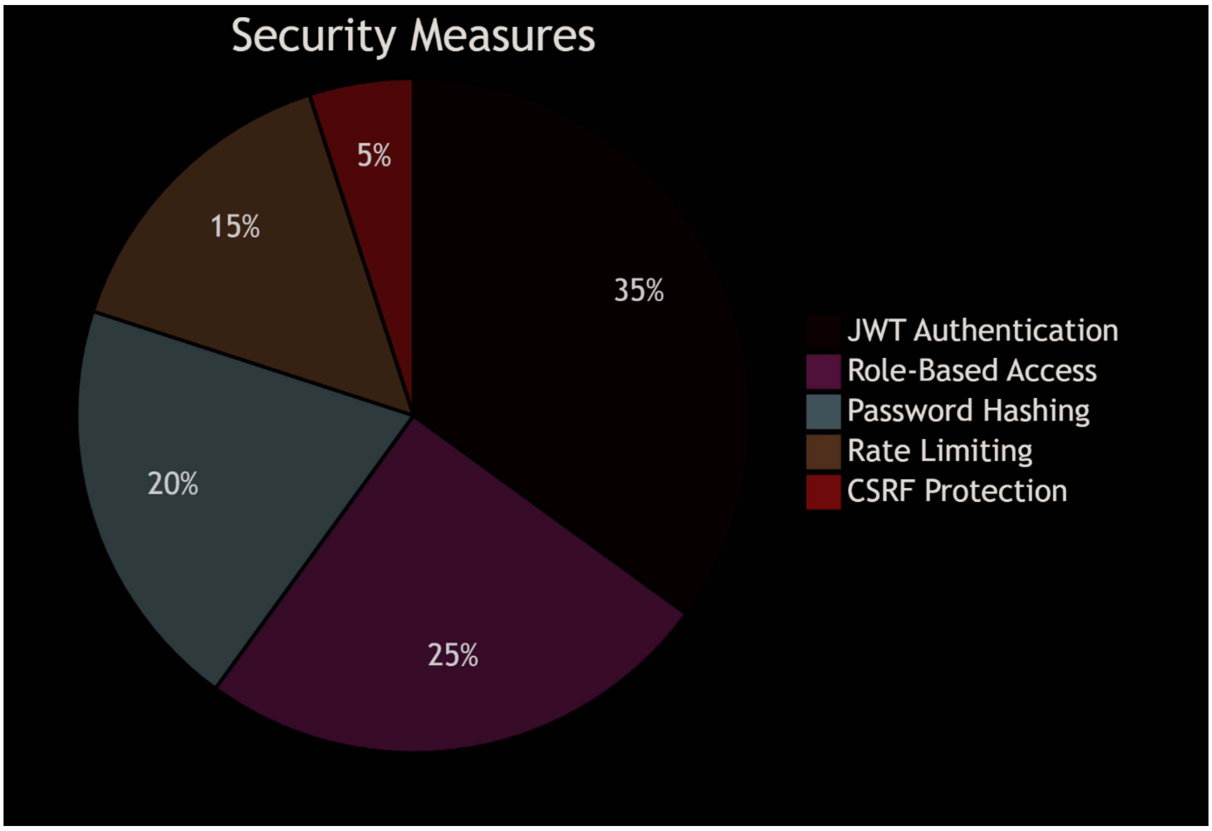


Figure 2.1: System Design Diagram

2.1.3 UI Design

The UI was kept clean and simple for easy navigation. Key pages include:

- Home Page – Login/registration.
- Quiz Page – Displays questions and options.
- Result Page – Shows scores and feedback.
- Admin Panel – Manage quizzes and users.
- Teacher Panel – Manage quizzes and classes.

We used HTML, CSS, and Bootstrap for responsive design.

2.2 Overall Project Plan

Phase	Description	Status
Planning	Requirement analysis & tools setup	Completed
Design	System design & UI wireframes	Completed
Development	Backend logic & database setup	Completed
Testing	Testing features & fixing bugs	Completed
Finalization	Report writing & submission	

Chapter 3

Implementation and Results

3.1 Implementation

We used:

- Backend: Django (Python).
- Frontend: HTML, CSS.
- Database: SQLite/MySQL with Django ORM.

Key features implemented:

- Secure user authentication (login, logout, registration).
- Quiz module with multiple-choice questions.
- Instant scoring system after quiz submission.
- Admin panel for managing questions and user results.
- Database models for Users, Questions, Options, and Results.

3.2 Performance Analysis

Testing showed:

- Fast page loading.
- Accurate result calculation.
- Smooth user interaction.
- Secure login and data handling.
- Efficient handling of multiple users.

3.3 Results & Discussion

The system achieved all objectives:

- Instant quiz results.
- Correct storage of quiz data.
- Easy admin management.
- Clean and simple UI.
- Smooth cross-device performance.

Chapter 4

Engineering Standards and Mapping

This chapter discusses how our project aligns with engineering standards, its impact on society and the environment, project management aspects, and how it maps to complex engineering problems and activities, as outlined in our DBMS Lab course.

4.1 Impact on Society, Environment and Sustainability

4.1.1 Impact on Life

Our quiz system promotes self-assessment and continuous learning. It benefits students and learners by making knowledge testing accessible and engaging. The instant feedback feature supports academic improvement.

4.1.2 Impact on Society & Environment

Being a fully digital system, it reduces the need for printed materials, which helps the environment. It also promotes digital education, which is especially helpful for remote and under-resourced communities.

4.1.3 Ethical Aspects

The system is designed to protect user data. We followed good practices such as secure login, limited access to admin features, and safe storage of quiz data. No personal data is collected unnecessarily.

4.1.4 Sustainability Plan

Our system is lightweight and scalable. It can be maintained and extended easily. Since it's built with open-source tools (Django, SQLite/MySQL), it’s sustainable for future academic or personal use without extra cost.

4.2 Project Management and Team Work

Our group of three students followed a team-based approach using task sharing and regular meetings. We divided responsibilities as follows:

- Member 1: Backend development (Django, models, views)
- Member 2: Frontend design (HTML/CSS)
- Member 3: Database design and testing

Cost Analysis

Item	Cost (BDT)	Remarks
Laptop/PC (personal)	0	Already owned
Internet (shared use)	300	Group use for 2 months
Software (Django, etc.)	0	All are open-source
Total Cost	**300**	Minimal cost for collaboration

Alternate Budget (If Hosted Online)

Item	Cost (BDT)	Remarks
Domain & Hosting	2000	For live deployment
SSL & Backend Server	1000	Optional, for secure hosting
Total	**3000**	If deployed on a live server

4.3 Complex Engineering Problem

Our project addressed a complex engineering problem by integrating multiple systems (frontend, backend, database) into a full-stack solution, aligning with the program's outcomes and demonstrating real-world application of database management concepts.

4.3.1 Mapping of Program Outcomes (POs)

POs	**Justification**
PO1	Demonstrated DBMS concepts, normalization, and SQL implementation.
PO2	Applied logical thinking to structure the backend and connect it to the database.
PO3	Designed and implemented a working web application solving a real-world educational problem.

4.3.2 Complex Problem Solving

We mapped the complexity of our problem using the following categories:

Table 4.2: Mapping with Complex Problem Solving

CEP Code	**Attribute**	**Rationale**
EP1	Depth of Knowledge	Required knowledge of Django, SQL, and web development.
EP2	Range of Conflicting Requirements	Balancing user needs, data storage, and admin controls.
EP3	Depth of Analysis	Designed relational models to handle quiz logic and scoring.
EP4	Familiarity of Issues	Faced issues like data validation and security handling.
EP5	Applicable Codes	Followed Django conventions, HTML structure, and DB design rules.
EP6	Stakeholder Involvement	Considered both users and admin experience in the design.
EP7	Interdependence	Linked UI, logic, and database to work as one complete system.

4.3.3 Engineering Activities

Table 4.3: Mapping with Engineering Activities

EA Code	Attribute	Rationale
EA1	Range of Resources	Used Django, SQLite/MySQL, HTML/CSS, GitHub for version control.
EA2	Level of Interaction	Collaborated as a team and tested all parts together.
EA3	Innovation	Created a custom quiz logic system with dynamic result handling.
EA4	Consequences for Society & Environment	Promoted paperless learning and accessible education.
EA5	Familiarity	Followed web development and DBMS best practices throughout.

Chapter 5

Conclusion

This chapter provides a summary of our work on **Project A.C.E**, discusses its limitations, and outlines future improvements. It reflects on the learning experience and practical outcomes of our group project.

5.1 Summary

We developed a full-stack quiz system named Project A.C.E. It allows users to register, take quizzes, and view instant results, while admins manage quizzes and scores. The project used Django, HTML/CSS, and SQL databases.

5.2 Limitations

- Basic UI design.
- No quiz timer or question randomization.
- Not fully optimized for mobile.
- Local deployment only (not online).
- Missing advanced features like password reset or email verification.

5.3 Future Work

- Improve UI/UX using Tailwind/Bootstrap.
- Add timer, randomization, and difficulty levels.
- Make mobile responsive.
- Deploy online with hosting services.
- Add email verification, password reset, and CAPTCHA.
- Provide analytics for teachers/admins.

References

1. Elmasri, R., & Navathe, S. B. (2015). *Fundamentals of Database Systems* (7th ed.). Pearson.
2. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database System Concepts* (7th ed.). McGraw-Hill.
3. Date, C. J. (2004). *An Introduction to Database Systems* (8th ed.). Addison-Wesley.
4. Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems* (3rd ed.). McGraw-Hill.
5. Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media.
6. Holovaty, A., & Kaplan-Moss, J. (2009). *The Definitive Guide to Django: Web Development Done Right* (2nd ed.). Apress.