Project Report: QuizMaster - An Online Quiz Platform

1. Student Details

- **Name**: Akshit Garg
- **Email**: 24f2000777@ds.study.iitm.ac.in
- **Course**: Modern Application Development 1
- **Institution**: IIT Madras
- **Roll Number**: 24f2000777
- **Date of Submission**: March 26, 2025

---

2. Project Details

2.1 Project Overview

QuizMaster is an online quiz platform designed to facilitate quiz creation, management, and participation. It allows admins to manage subjects, chapters, quizzes, and questions, while users (students) can attempt quizzes, view their scores, and track their performance through data visualizations. The platform includes features like user authentication, quiz and, score tracking, and summary charts for performance analysis.

2.2 Problem Statement

The objective was to develop a web application that:

- Enables admins to create and manage subjects, chapters, quizzes, and questions.
- Allows users to sign up, log in, attempt quizzes, and view their scores.
- Provides data visualization for performance tracking (e.g., quiz scores for users and summary statistics for admins).
- Implements search functionality for both admins and users to find subjects, quizzes, and users.
- Ensures a user-friendly interface with proper database management and secure authentication.

2.3 Approach to the Problem

The project was developed using the Flask framework in Python, following a modular approach:

1. **Database Design**:
   - Designed a relational database using SQLite and SQLAlchemy with tables for users, admins, subjects, chapters, quizzes, questions, and scores.
   - Established relationships between tables (e.g., a quiz belongs to a chapter, a chapter belongs to a subject).
2. **User Authentication**:
   - Implemented signup and login functionality for users using password hashing (werkzeug.security).
   - Added a hardcoded admin account (admin@example.com) for administrative access.
3. **Admin Features**:
   - Created routes for adding, editing, and deleting subjects, chapters, quizzes, and questions.
   - Added search functionality to find users, subjects, and quizzes.
   - Implemented summary statistics and charts for admins to view average quiz scores per subject.
4. **User Features**:
   - Developed a user dashboard to display available quizzes.
   - Enabled users to attempt quizzes, submit answers, and view their scores.
   - Added search functionality for subjects and quizzes.
   - Provided a summary page with a chart of the user's quiz performance.
5. **Data Visualization**:
   - Used Matplotlib to generate bar charts for quiz performance (user scores and admin summary statistics).
   - Saved charts as PNG images in the static folder and displayed them in the summary.html template.
6. **Testing and Debugging**:
   - Tested all routes and features (e.g., quiz submission, score calculation, chart generation).
   - Fixed issues like UndefinedError in templates by ensuring all variables are passed correctly.
   - Adjusted Matplotlib charts to fix font and overlapping issues.

---

3. Frameworks and Libraries Used

- **Flask**: A lightweight web framework for Python, used to build the web application and handle routing.
- **Flask-SQLAlchemy**: An ORM for Flask to interact with the SQLite database.
- **SQLite**: A lightweight database for storing user data, quizzes, questions, and scores.

- **Werkzeug**: Used for password hashing and verification (generate_password_hash, check_password_hash).
- **Matplotlib**: Used for generating bar charts to visualize quiz performance and summary statistics.
- **Jinja2**: Templating engine for rendering dynamic HTML pages.
- **Bootstrap**: Used in templates (via CDN) for responsive and styled UI components.

---

4. ER Diagram of the Database

The database consists of the following tables with their relationships:

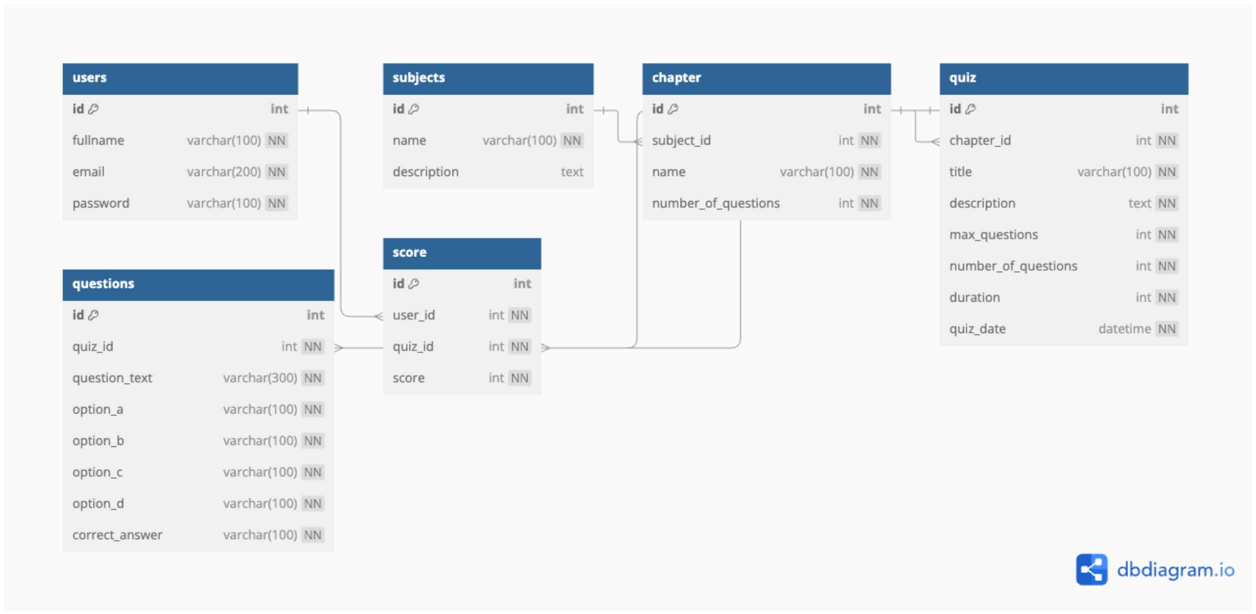
4.1 Tables and Columns

1. **Users**:
   - id (Primary Key, Integer)
   - fullname (String, Not Null)
   - email (String, Unique, Not Null)
   - password (String, Not Null)
2. **Subjects**:
   - id (Primary Key, Integer)
   - name (String, Unique, Not Null)
   - description (Text, Nullable)
3. **Chapter**:
   - id (Primary Key, Integer)
   - subject_id (Foreign Key to Subjects.id, Not Null)
   - name (String, Not Null)
   - number_of_questions (Integer, Not Null)
4. **Quiz**:
   - id (Primary Key, Integer)
   - chapter_id (Foreign Key to Chapter.id, Not Null)
   - title (String, Not Null)
   - description (Text, Not Null)
   - max_questions (Integer, Not Null, Default=5)
   - number_of_questions (Integer, Not Null)
   - duration (Integer, Not Null)
   - quiz_date (DateTime, Not Null)
5. **Questions**:
   - id (Primary Key, Integer)
   - quiz_id (Foreign Key to Quiz.id, Not Null, Cascade on Delete)
   - question_text (String, Not Null)
   - option_a, option_b, option_c, option_d (String, Not Null)
   - correct_answer (String, Not Null)

6. **Score**:
   - id (Primary Key, Integer)
   - user_id (Foreign Key to Users.id, Not Null)
   - quiz_id (Foreign Key to Quiz.id, Not Null)
   - score (Integer, Not Null)

4.2 Relationships

- **Subjects ↔ Chapter**: One-to-Many (a subject can have multiple chapters).
- **Chapter ↔ Quiz**: One-to-Many (a chapter can have multiple quizzes).
- **Quiz ↔ Questions**: One-to-Many (a quiz can have multiple questions, with cascade delete).
- **Quiz ↔ Score**: One-to-Many (a quiz can have multiple scores).
- **Users ↔ Score**: One-to-Many (a user can have multiple scores).



5. Project Demonstration Video

project demonstration video.mov