Quiz Master - V1

Name: KUSHAL YARLAGADDA
Roll Number: 24f1001740
Course Name: Modern Application Development I
Submission Date: 10-03-2025

1. Overview

Objective of the Study:

The Quiz Master - V1 is an application that is multi-user based and created with the aim of giving an interactive environment for exam preparation. Users can try quizzes for different subjects and chapters, while administrators can deal with the content, i.e., chapters, subjects, quizzes, and questions. The application is created such that the process of creating quizzes, administering quizzes, and monitoring the performance of users as well as admins is made easy.

Problem Statement:

In the busy, dynamic world of the present day, both students and professionals require a suitable platform to gauge their knowledge and prepare for examinations. The conventional means of quiz preparation and administration are time-consuming and have no interactivity. The Quiz Master - V1 solves this issue by offering a user-friendly interface for the preparation, administration, and monitoring of quiz performance.

Scope of the Project

The application possesses the following characteristics:

- Admin Dashboard: Enables admins to add, edit, and remove subjects, chapters, quizzes, and questions.
- User Dashboard: Facilitates user registration, login, and quiz completion.
- Quiz Timer: Ensures tests are completed in a time that has been allocated.
- Performance Monitoring: Monitors user scores and presents them in a graphical format utilizing Chart.js.
- Search Functionality: Allows administrators to search for users, topics, and quizzes.

2. Project Design

Technologies and Frameworks Employed

The application development employs several frameworks and technologies in the following order:

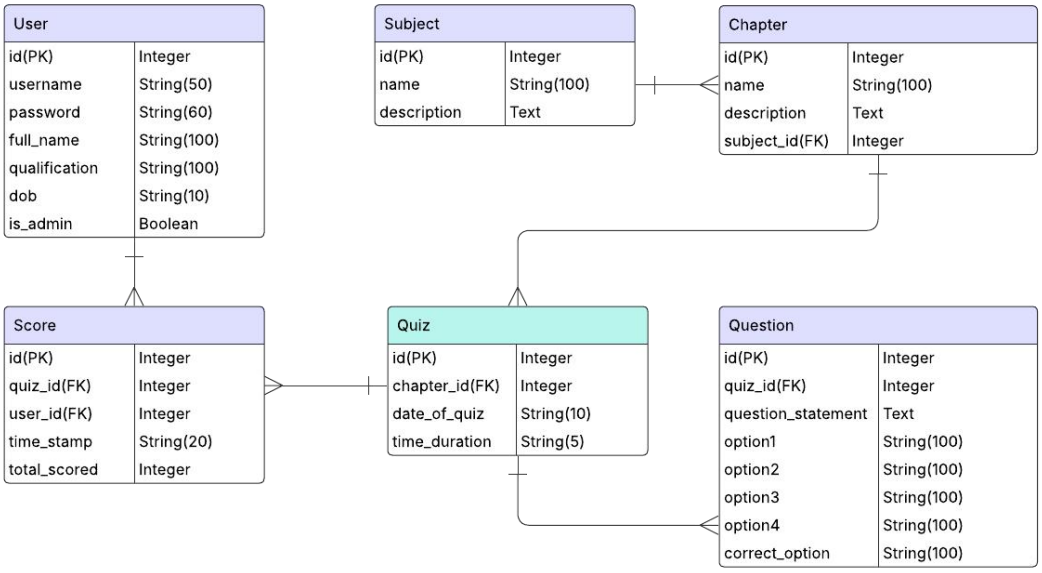
- Flask is a light web framework for Python applied in backend development.
- SQLite: A lightweight database system used to store application data.
- Jinja2: A templating engine to directly render HTML templates at runtime.
- HTML, CSS, Bootstrap: Used for frontend layout and styling.
- Chart.js: A JavaScript library for graphing user performance data.

Database Design

The database contains the following tables:

- User Table: Stores user information like username, password, full name, qualification, and date of birth.
- Subject Table: Holds information about subjects, i.e., their names and descriptions.
- Chapter Table: Stores chapter details such as name, description, and the subject it falls under.
- Quiz Table: It stores quiz information like date, duration, and the corresponding chapter.
- Question Table: Serves to store quiz questions, such as the question statement, options, and correct answer.
- Score Table: Saves user quiz scores with quiz ID, user ID, date, and total score.

Entity-Relationship Diagram:



API Endpoints:

The application uses the following API endpoints:

- GET /api/subjects: Gets all the subjects.
- GET /api/chapters: All chapters are retrieved.
- GET /api/quizzes: Returns all quizzes.
- GET /api/questions: Retrieves all the questions.
- GET /api/scores: Returns all scores.

3. Implementation Key Features

Admin Dashboard:

- Admins can add, modify, and remove subjects, chapters, quizzes, and questions.
- Admins can search for users, subjects, and quizzes.
- Admins can view user performance graphs using Chart.js.

User Dashboard:

- Users can create accounts and use the application.
- Users are shown available quizzes that they can attempt.
- Scores and user performance can be accessed.

Assessment Timekeeper:

- There is a time limit for each quiz, and there is a countdown timer to ensure users finish the quiz within the time.
- Once the timer expires, then the quiz is submitted.

Performance Monitoring:

- User scores are saved within the database and are shown in a bar chart through Chart.j.
- Admins can view the performance of all users.

Search Facility:

- Admins are able to search for users, subjects, and quizzes by keyword.

Difficulties Faced

Quiz Timer Implementation:

- Having a countdown timer that worked flawlessly in all browsers was a challenge.
- Solution: Employed JavaScript to design a countdown timer and submit the quiz automatically when time is over.

Database Relationships:

- Planning of relationships between questions, quizzes, chapters, and subjects was required.

- Solution: Used SQLAlchemy to create relationships between tables and ensure data integrity.

Performance Diagram:

- It was difficult to display user performance information in an aesthetically pleasing way.
- Solution: Used Chart.js to create a bar chart that dynamically adjusts based on user scores.

Validation and Error Handling

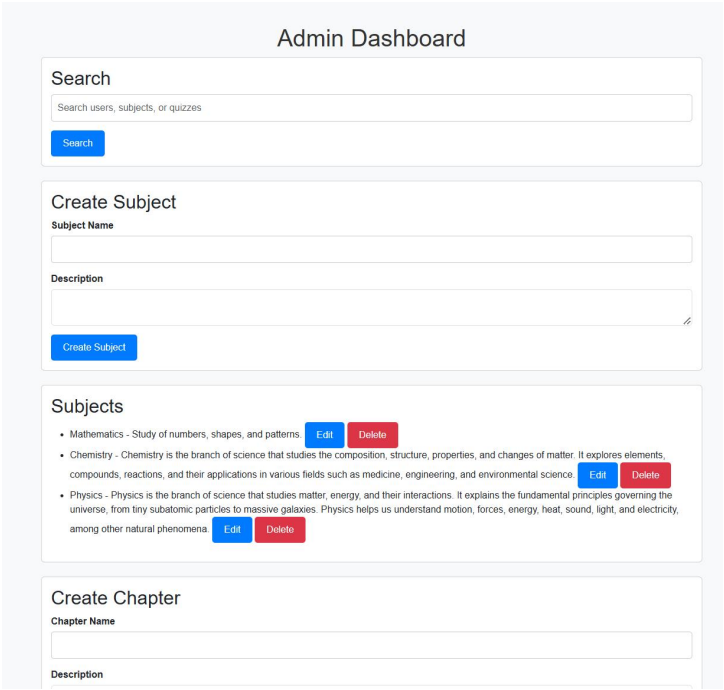
Frontend Validation:

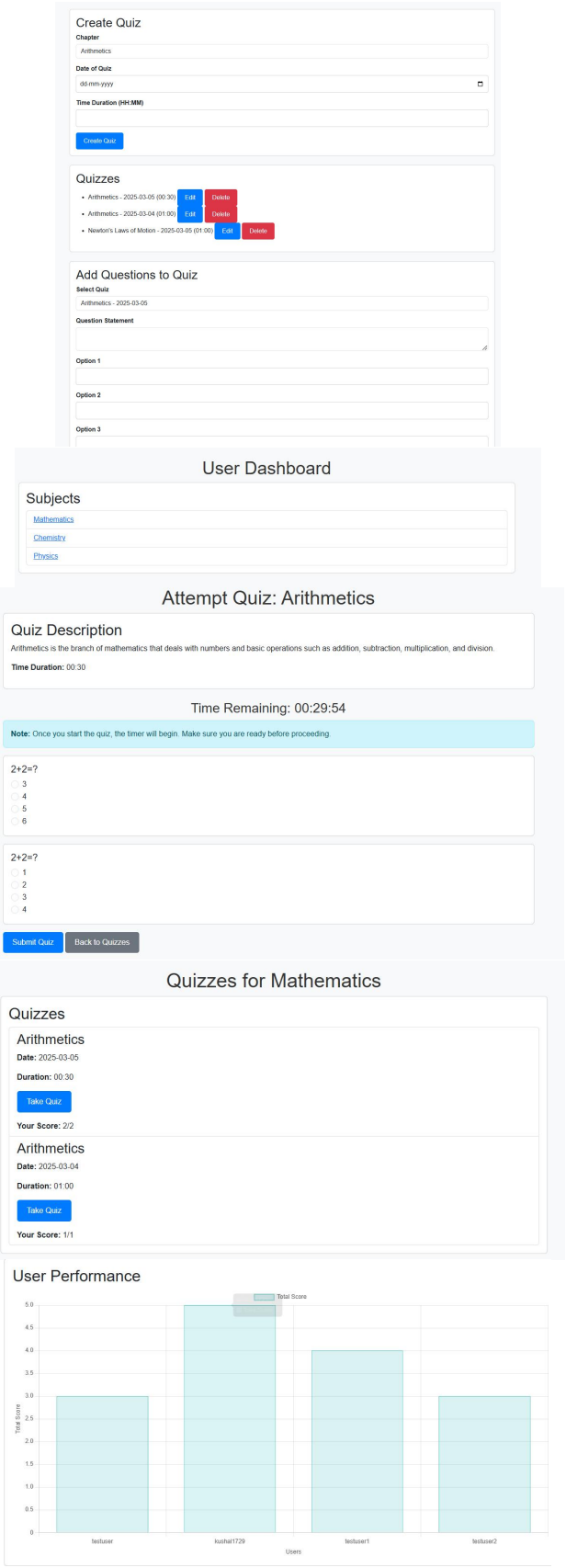
- HTML5 attributes such as required, type="date," and type="text," were used to enable users to enter accurate data.

Backend Validation:

- Flask routes incorporate validation mechanisms for quiz times in the proper format (HH:MM) and all fields having been properly completed.

4. Results and Screenshots:





5. **Link of the presentation video:**

   https://youtu.be/fy7Ieh8HvII