Modern Application Development– I
Project on Quiz Master

Author

Name: Smriti S
Roll Number: 23f2000599
Student Email: 23f2000599@ds.study.iitm.ac.in
About Me: I am a dual degree student pursuing Computer Science in Chennai and Data Science at IIT Madras (Diploma Level, Term 3). I have experience in web development, databases, and API design, and have worked on gamified learning, cybersecurity, and emergency response projects through hackathons and research conferences.

Description

This project implements an online quiz system with two roles: Admin and User. Users can attempt quizzes multiple times, view their scores, and track performance based on previous attempts. Admins can create, edit, and delete quizzes while also monitoring user performance.

Technologies used

- **Flask** – Handles routes, requests, and templates.
- **Flask-SQLAlchemy** – Manages the database.
- **Flask-Login** – Handles user authentication.
- **Jinja2** – Renders dynamic HTML.
- **SQLite/PostgreSQL** – Stores user data, quizzes, and scores.
- **Flask-Session** – Manages user sessions.
- **Bootstrap, CSS, HTML, JavaScript** – Designs the frontend.
- **Datetime & Timedelta** – Tracks quiz timestamps.
- **Functools (wraps)** – Helps with authentication decorators.

DB Schema Design

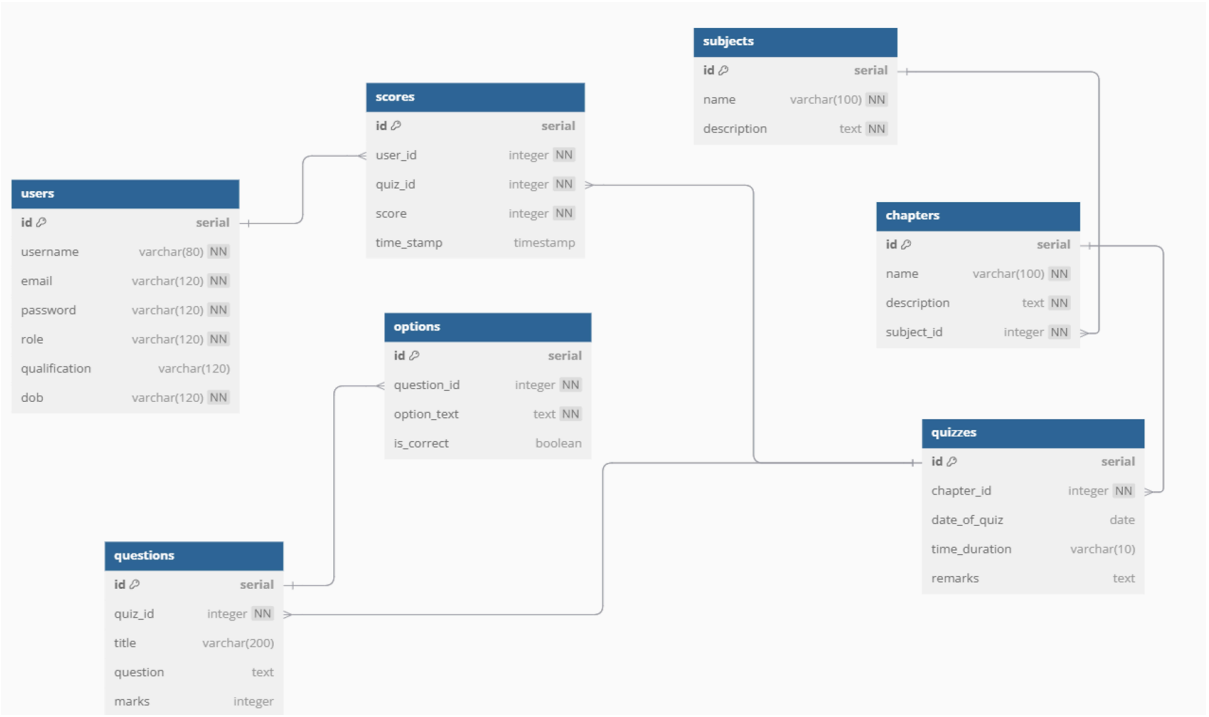

Figure 1.1

Figure 1.1 shows the **Entity-Relationship (ER) Diagram** of the project, which explains how different parts of the database are connected. It includes entities like Users, Subjects, Chapters, Quizzes, Questions, and Scores. Each entity has a unique ID (primary key), and they are linked using foreign keys. The one-to-many relationships, such as one subject having multiple chapters or one quiz containing multiple questions, enable efficient scalability for real-world usage.

This setup also helps maintain data accuracy by automatically deleting (cascading deletions) related data when something is removed. For example, if a subject is deleted, all its chapters, quizzes, and questions are also removed. The system also lets users take quizzes multiple times, allowing both the admin and users to track progress over time. This makes the quiz system flexible and user-friendly.

API Design

The API includes key endpoints for authentication, quiz management, content handling, user and admin controls, and search. Authentication APIs like **/signup, /login, and /forgot-password** handle user registration, login, and password resets. Quiz management uses **/quiz/add, /quiz/start/<quiz_id>/<question_number>, and /quiz/submit/<quiz_id>** to let admins create quizzes and users attempt and submit them. Content management allows admins to create, edit, and delete subjects, chapters, and quiz questions through **/subject/add, /chapter/add/<subject_id>, and /quiz/<quiz_id>/question/add**. User management includes **/user/dashboard, /scores, and /user/summary** for accessing quizzes, tracking scores, and analyzing performance. Admin controls feature **/admin/dashboard, /admin/unified-search, and /admin/delete_user/<user_id>** for managing subjects, users, and searching content. The search API, **/search,** helps users search for subjects and quizzes.

Architecture and Features

The main application logic and API functions are in **apis.py**, while database operations are handled in **database.py.** The **main.py** file is for the starting of the app.

All webpage templates, like add_chapter.html and admin_navbar.html, are stored in the templates folder. The project uses a virtual environment (env directory) to manage dependencies.The instance folder contains the database used by the application. It stores all subject/chapter/quiz-related data, user information, and other records.

The Quiz Master project includes features for both admins and users. Admins have full control with **CRUD** operations, allowing them to create, edit, delete, and manage subjects, quizzes, chapters, and questions. They can search for users, quizzes, subjects, and questions, and also delete user accounts if needed. Admins get summary charts to track system activity and performance.

Users can attempt quizzes within a set time limit, with a timer ensuring auto-submission when time runs out. If they fail a quiz (score below 50%), they get a "Needs Improvement" message and can retry the quiz. After each quiz, users can see their current and past scores, helping them track progress.

A summary graph shows their overall performance. Users can also search for subjects and quizzes and securely log out when finished.

Video

https://drive.google.com/file/d/1QD_-I9IzKQKzURh4FABwMWPAuAuYZmCk/view?usp=sharing