

Author

Sudhanshu Shekhar

23f2005067

23f2005067@ds.study.iitm.ac.in

Hello, my name is Sudhanshu. I am currently in the Diploma Level of IITM BS Degree. I am doing this degree as a full-time student. I am interested in web development and have developed this application as part of my coursework.

Description

In this project, I have developed a multi-user web application named **Quiz Master** using Flask. This app is designed for exam preparation across multiple courses. It allows an admin to create and manage subjects, chapters, quizzes, and questions, while users can register, attempt quizzes, and track their performance. The admin has root access and manages all users and content. Users can select a subject, attempt quizzes, and view their scores.

Technologies used

1. Flask Framework

- Core web framework for routing and request handling.
- Chosen for its simplicity and flexibility.

2. Flask Extensions

- Flask-SQLAlchemy: Database ORM for efficient data management.
- Flask-Login: Secure user authentication and session handling.
- Flask-WTF: Form handling and CSRF protection.
- Flask-Mail: Email notifications. (Tried but not working)

3. Frontend

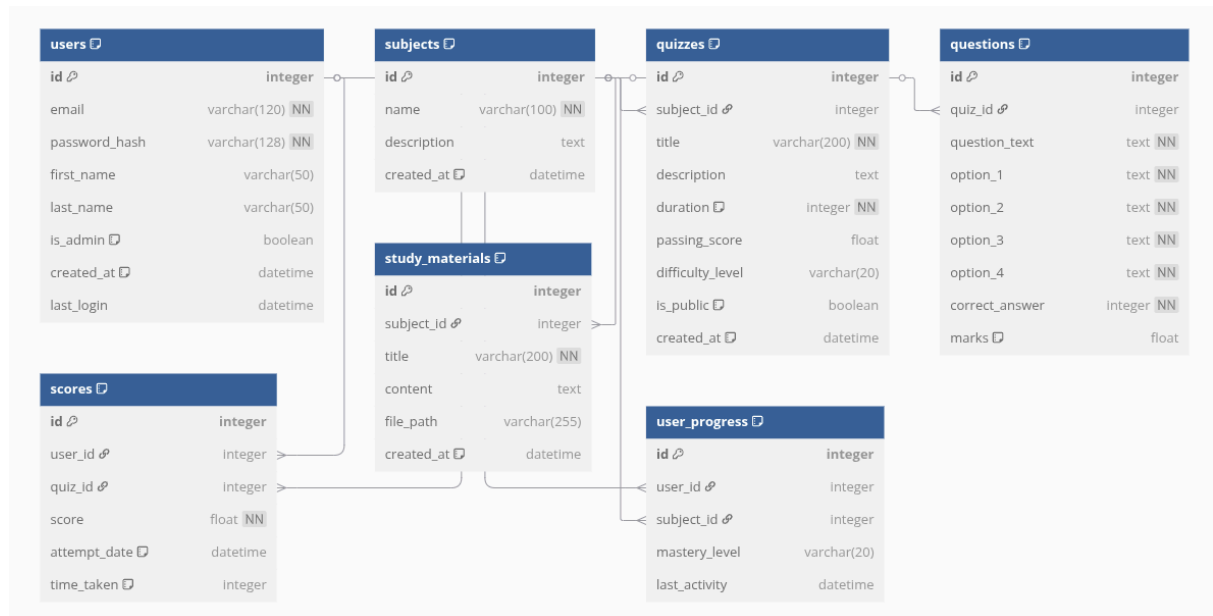
- Bootstrap 5: Responsive design framework for building a mobile-first interface.
- Chart.js: For creating interactive performance analytics and visualizations.
- jQuery: For DOM manipulation and handling AJAX requests.

4. Database

- SQLite: Development database used for storing application data locally.
- PostgreSQL: Production database, chosen for scalability and reliability in a production environment.

DB Schema Design

<https://dbdiagram.io/d/67e5f52a4f7afba18482538d>




API Design

I have created APIs for user authentication, quiz management, and tracking user progress. These APIs were implemented using Flask routes, providing endpoints for login, registration, quiz creation, and retrieving user performance data in a structured JSON format.

Architecture and Features

The architecture of the project follows the **Model-View-Controller (MVC)** design pattern. The **Model** layer is responsible for data management, using **Flask-SQLAlchemy** to interact with the SQLite (for development) or PostgreSQL (for production) database. The **View** layer is the front-end, built using **Jinja2** templates, with **Bootstrap 5** for responsive design and **Chart.js** for data visualization. The **Controller** layer consists of **Flask routes** that handle user requests, interact with the model, and return the appropriate views. Extensions like **Flask-Login** (authentication), **Flask-WTF** (form handling), and **Flask-Mail** (email notifications) enhance the application's functionality. This MVC structure ensures separation of concerns, maintainability, and scalability of the application.

Video

 presentation.mp4