Durbar Mandal
21f3000030
21f3000030@ds.study.iitm.ac.in

I am a tech enthusiastic person who is a total nerd when it comes to anything related to data science, programming, development and everything related to computers. I am a curious learner currently working on a project focused on developing an interactive quiz app.

Description

The project I am working on focuses on developing an interactive quiz app designed to facilitate collaboration and evaluation between students and teachers or institutions. This app enables institutions to create and assign chapter-based quizzes, allowing students to attempt them regularly. Students can track their performance, while institutions can monitor their progress and assess their understanding of various subjects and topics, ensuring a more structured and effective learning experience

Technologies used

Flask: A lightweight web framework for Python used to create web applications

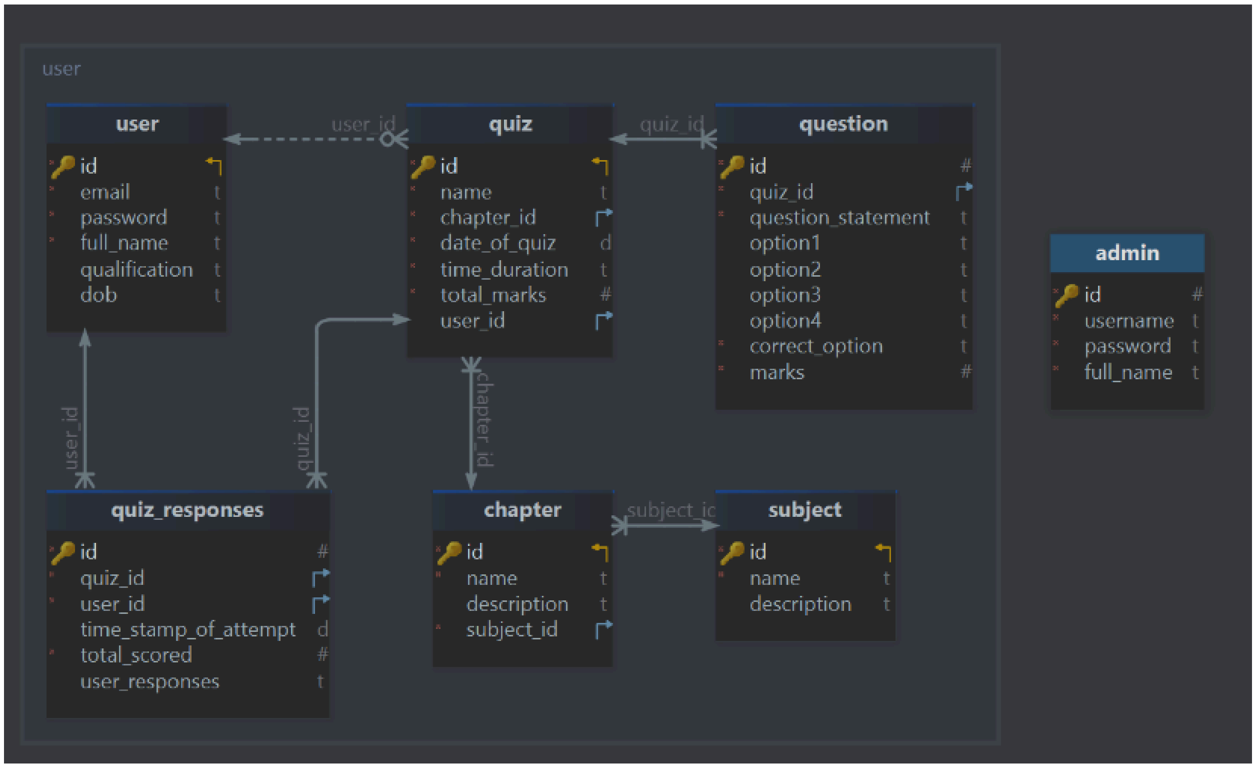
Modules used:

- Flask: Core framework for handling web requests and responses.
- flash: Used for displaying one-time messages (alerts) to users.
- render_template: For rendering HTML templates.
- request: To handle incoming HTTP requests (GET, POST, etc.).
- redirect: For redirecting users to another route.
- session: To manage user sessions.
- url_for: To generate URLs for routes.

HTML – Provides the structural foundation for the app’s user interface.
CSS – Enhances the visual design and layout for a responsive and appealing UI.
JavaScript – The primary purpose is to implement a dynamic timer on the quiz attempt page and enhance interactivity for a smoother user experience.
Bootstrap – Streamlines UI development with pre-styled components and a responsive grid system.
Jinja2 – A templating engine used in Flask to render dynamic content in HTML.

Flask – Also used for building the backend and handling requests.
Flask SQLAlchemy – An ORM that simplifies database interactions in Flask.
DB Browser – A GUI tool for managing and visualizing SQLite databases.
SQLite – A lightweight, file-based database used to store quiz data

DB Schema Design



This database is structured to ensure clear separation of concerns, maintain referential integrity, and support future scalability. The **user** and **admin** tables handle authentication by storing credentials separately, ensuring security through unique identifiers and hashed passwords. Subjects and chapters are organized in their own tables to categorize quizzes, while the quiz table captures essential metadata such as time duration and total marks. The question table stores MCQs and ties them to their respective quizzes, providing options and correct answers. Finally, the quiz_responses table records each user's attempt, linking them to a specific quiz with timestamps, scores, and submitted answers. By normalizing data across these entities, the design remains efficient, secure, and straightforward to maintain

API Design

User Authentication

- Registration: Creates a new user with email, password, and profile details; data is submitted via form and stored in the User model.
- Login: Authenticates users using email/password and starts a session upon success.

Admin Management

- Admin Login: Authenticates admin users and stores admin details in the session.
- Subject Management: Handles CRUD operations for subjects using form data.

Content Management

- Chapter Operations: Manages creation, updates, and deletion of chapters under specific subjects.
- Quiz Operations: Oversees quiz management within chapters, including setting time duration and total marks.
- Question Operations: Adds, edits, or deletes quiz questions with options and correct answers.

Quiz Interaction

- Submit Quiz: Processes user answers, calculates scores, and stores responses in Quiz Response.
- Quiz Result: Displays detailed results, including user responses and correct answers.

Data Retrieval

- User Statistics: Provides quiz attempt history, average scores, and performance trends for individual users.
- Admin Summary: Aggregates metrics such as total attempts, average scores, and subject-wise performance.

Architecture and Features

Architecture

- The project is organized in a modular structure to separate concerns.
- Controllers (or route handlers) are placed in the root folder as app.py this file is responsible for initialization of the app and to manage business logic and database interactions.
- HTML templates that render dynamic content using the Jinja2 templating engine are stored in a "templates" folder.
- Static assets such as CSS, JavaScript, and images reside in a "static" directory.
- Database models and configurations using Flask SQLAlchemy are maintained in the root folder as “config.py” and models.py

Features

- Robust user authentication is implemented, with separate endpoints for registration and login, utilizing hashed credentials and session management for security.
- Core quiz functionalities include endpoints to create, edit, and delete quizzes and questions, with a dynamic timer implemented via JavaScript for an interactive quiz experience.
- Administrative functions provide dashboards for monitoring quiz performance and generating user statistics, offering additional oversight and control.
- The use of Bootstrap ensures a responsive and visually appealing UI, while Jinja2 supports dynamic content rendering in HTML.
- Efficient data handling is achieved through Flask SQLAlchemy, which simplifies interactions with the SQLite database.

Video

2025-03-20 19-25-46.mkv