Author

Mohamed Asif
24F1001713
24f1001713@ds.study.iitm.ac.in
I am a dual degree student pursuing B.Tech Smart Manufacturing at IIITDM Kancheepuram [current semester : 4th] and BS Data Science at IIT Madras [current level : diploma].

Description

To Build a quiz application using Flask for backend , HTML, CSS, Bootstrap for frontend , Jinja2 for templating and SQLAlchemy for database management with SQLite. The application should support multiple users, allow them to take quizzes, store their responses, and provide feedback on correct/incorrect answers. Also the admin should have entire access to manage subjects, chapters, and quizzes.

Technologies used

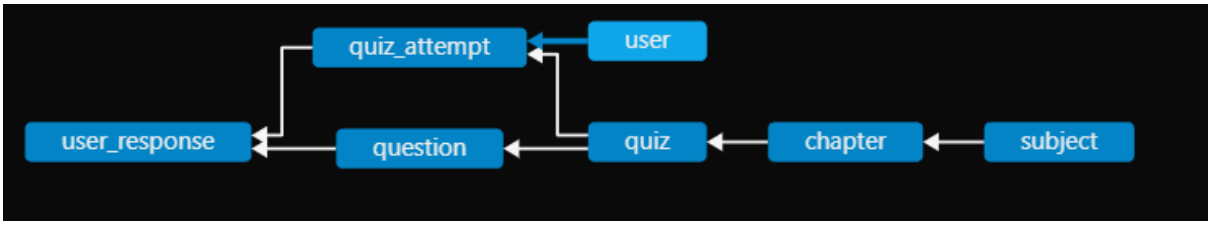
Backend:

- **Flask** – Web framework
- **Flask-SQLAlchemy** – ORM for database management
- **SQLite** – Database
- **Jinja2** – Templating
- **Pandas** – To convert data into pandas dataframe.
- **Matplotlib** – To create bar chart

Frontend:

- **HTML** – Structure
- **CSS** – Styling
- **Javascript** - used only for quiz timer and confirmation pop up for delete buttons.

DB Schema Design



1. User Table

- **id** (PK, Integer, Auto-increment), **username** (String, Unique, Not Null), **email** (String, Unique, Not Null), **password** (String, Not Null)

- **One-to-Many:** `QuizAttempt` (A user can attempt multiple quizzes, but each attempt belongs to one user)

2. Subject Table

- **id** (PK, Integer, Auto-increment), **sub_name** (String, Unique, Not Null), **sub_description** (String, Not Null)
- **One-to-Many:** `Chapter` (A subject has multiple chapters, but each chapter belongs to one subject)

3. Chapter Table

- **id** (PK, Integer, Auto-increment), **chapter_name** (String, Not Null), **subject_id** (FK → Subject.id, Not Null)
- **One-to-Many:** `Quiz` (A chapter has multiple quizzes, but each quiz belongs to one chapter)

4. Quiz Table

- **id** (PK, Integer, Auto-increment), **quiz_name** (String, Not Null), **chapter_id** (FK → Chapter.id, Not Null), **deadline** (String, Not Null), **duration** (String, Not Null)
- **One-to-Many:** `Question` (A quiz has multiple questions, but each question belongs to one quiz), `QuizAttempt` (A quiz can have multiple attempts, but each attempt belongs to one quiz)

5. Question Table

- **id** (PK, Integer, Auto-increment), **qno** (Integer, Not Null), **quiz_id** (FK → Quiz.id, Not Null), **question_statement** (Text, Not Null), **option1** (String, Not Null), **option2** (String, Not Null), **option3** (String, Not Null), **option4** (String, Not Null), **correct_option** (Integer, Not Null)
- **One-to-Many:** `UserResponse` (A question has multiple user responses, but each response belongs to one question)

6. QuizAttempt Table

- **id** (PK, Integer, Auto-increment), **score** (Integer), **user_id** (FK → User.id, Not Null), **quiz_id** (FK → Quiz.id, Not Null)

- **One-to-Many:** `UserResponse` (A quiz attempt has multiple user responses, but each response belongs to one attempt)

7. UserResponse Table

- **id** (PK, Integer, Auto-increment), **quiz_attempt_id** (FK → QuizAttempt.id, Not Null), **question_id** (FK → Question.id, Not Null), **chosen_option** (Integer, Nullable)

Architecture and Features

The project follows a modular Flask application structure:

`quiz_master_24f1001713/`

- **Controllers (`controllers/`)**: This folder contains `controller.py`, which handles request routing and application logic. Also contains `database.py` which Initializes the SQLAlchemy database instance for use throughout the project.
- **Models (`models/`)**: This folder contains `model1.py`, where SQLAlchemy models are defined to interact with the database.
- **Database (`instance/`)**: The SQLite database (`quiz.sqlite3`) is stored here, ensuring data persistence.
- **Templates (`templates/`)**: Contains all HTML files for rendering pages, such as `login.html`, `new_quiz.html`, and `scores.html`. These templates are dynamically populated with data from Flask.
- **Static Files (`static/`)**: Stores CSS stylesheets (`style1.css`, `style2.css`, etc.) and images like `top_scores_chart.png` for styling and displaying graphical content.
- **Main Application (`app.py`)**: The central file that initializes the Flask app, connects controllers and models, and runs the application.

Implemented Features

The project includes various features for quiz management:

- **User Authentication**: Handled via `login.html` and `register.html`, allowing users to sign up and log in.
- **Quiz Creation & Management**: Pages like `new_quiz.html`, `new_question.html`, `new_chapter.html` and `new_subject.html` enable quiz creation and editing. `view_quest.html` enables the admin to view the question created and to edit.

- **Quiz Attempt & Evaluation**: `view_attempt.html` and `start.html` allow users to take quizzes, view their responses, and receive correct/incorrect feedback.

- **Results**: `scores.html` and `summary.html` display user performance and quiz results.

- **Admin Panel**: `admin.html` and `quiz_management.html, user_details.html` gives admin dashboard for managing quizzes and users.
- **User Panel:** `user.html, score.html` gives user dashboard for viewing upcoming quizzes, attempted quizzes and scores.

- **Editing and Deleting:** Templates like `edit_quiz.html`, `edit_subject.html`, and `edit_quest.html, edit_chapter` allow modifications to quizzes and questions.

- **Styling Elements**: The presence of multiple CSS files for styling.

Video

https://drive.google.com/file/d/1L4e-sjFxpur83Fe-tg9W-hTfuz4-6G4v/view?usp=sharing