Project Report: Quiz Master Management System

Developer

Avinash Kumar
Roll Number: 21F1006385
Email: 21f1006385@ds.study.iitm.ac.in

I am a student of IIT Madras. When I first started learning web development, I was very excited and dreamed of creating my own website. Now, that dream has turned into reality as I can successfully build web applications using the concepts I've learned. I'm eager to continue exploring new ideas and techniques to make my websites more attractive, dynamic, and user-friendly.

Description

This project is a Quiz Management System where the admin can perform CRUD operations on relevant fields such as subjects, chapters, and questions. The admin can also view user details, use search functionality, and visualize quiz data through graphs. Users can take quizzes, view their results, and track their performance using graphical insights

Technologies Used

- **Python (Flask)** - For building the web application backend.
- **Jinja2** - For rendering dynamic content in HTML templates.
- **SQLite** - For lightweight database management.
- **SQL Alchemy** - ORM for efficient database interaction.
- **Bootstrap** - For styling and responsive design.
- **CSS** – Use for some other component design.

DB Schema Design

The database is designed to efficiently manage quiz-related data with proper relationships and constraints. Key entities include:

- **User Info: Manages user details with fields like email, password, and role. Linked with the Score table to track user performance.**
- **Subject: Contains subject details with a unique code and credit field. Linked with the Chapter table.**
- **Chapter: Stores chapter information and links to the respective subject.**
- **Quiz: Contains quiz details like quiz title, duration, and date_of_quiz. Linked with both the Question and Score tables.**
- **Question: Defines each quiz question with options and the correct answer.**
- **Score: Tracks user performance in quizzes with fields like total scored, percentage scored, and pass_fail_status.**

API Design

The project includes a RESTful API using Flask's flask restful extension. Currently, the following endpoint is implemented:

- **GET /api/get_subject**
    - **Purpose:** Retrieves all subjects from the database.
    - **Response Format:** JSON object containing subject details such as id, name, code, credit, and description.
    - **Implementation Details:** The endpoint queries the Subject table and formats the data into JSON for improved readability.

Architecture and Features

The project follows the **MVC (Model-View-Controller)** architecture to ensure organized and maintainable code:

- **Models**: Represent database tables using SQL Alchemy ORM. Relationships, constraints, and data structures are defined here.
- **Application Entry Point (app.py):** Acts as the starting point for the application, initializing Flask, setting configurations, and registering routes.
- **Controllers**: Handle route logic, API endpoint, and database interactions. The Subject Api class manages data retrieval for subjects.
- **Templates**: Jinja2 templates are used to render dynamic content, improving flexibility and reducing redundancy.
- **Static Files**: CSS and JavaScript files are included to enhance the UI and add interactivity.

Key Features

- **Admin Dashboard: A user-friendly interface for managing quiz content, ensuring efficient data management**
- **Comprehensive CRUD Operations:** Allows admins to create, read, update, and delete subjects, chapters, and quiz questions.
- **Enhanced UI with Bootstrap:** Ensures responsive design and improved usability across devices.

Video

Click here to view the project demo