Author

Shirin Shabbir Sabuwala
23f2003577
23f2003577@ds.study.iitm.ac.in
I am a passionate Data Scientist student with a keen interest in problem solving and logical reasoning. Along with the IITM online degree, I am currently enrolled into a B.Sc (Hons) degree in Data Science and Big Data Analytics. With a curious mind, I love exploring new concepts and am always open to learning more and more.

Description

The Quiz Master is an application designed for students of all ages to take quizzes on various subjects and test their knowledge. The user side of the app allows one to take multiple quizzes, reattempt quizzes, view their scores and visualize their learning. The admin side of the app allows creation, updation and deletion of quizzes, along with monitoring users' quiz attempts.

Technologies used

At the backend, this project is built using Flask, a web framework for Python, along with Flask-SQLAlchemy for database management, and SQLite as the default database. Flask-WTF is used for form handling and validation, while Flask-Login manages the user authentication. To display the content efficiently, Jinja2 templating is used for rendering the HTML pages. The frontend is styled using Bootstrap, ensuring the design is as responsive and user-friendly as possible. Putting all these technologies together, the project is developed to create a seamless quiz platform that is easy to navigate, secure, and aesthetically pleasing.

DB Schema Design

The database is structured using Flask-SQLAlchemy, with tables designed to store and manage all quiz-related data. The main tables include User, Subject, Chapter, Quiz, Question, and Score. The User table stores user details with fields like id (primary key), username (unique), email (unique), password_hash, and role (to distinguish between admin and regular users). The Subject and Chapter tables organize quizzes hierarchically, with foreign key constraints linking them to create a structured flow. The Quiz table includes fields like id, name, date_of_quiz, and deadline, ensuring time-bound quiz availability. The Question table links to the Quiz table via a foreign key and stores the question text. It also contains the options as multiple-choice answers, with a separate field for the correct answer. Finally, the Score table records user attempts, linking users, quizzes, and scores. This relational design aims to ensure data integrity, scalability, and efficient querying while preventing redundancy with the help of constraints like primary keys, foreign keys, unique constraints, and not-null constraints.

API Design

The API is implemented using Flask-RESTful to provide structured endpoints for managing quizzes, questions, and user interactions. APIs have been created for user authentication (login, registration), quiz retrieval, handling questions and submitting result. The implementation follows REST principles, using GET, POST, PUT, and DELETE methods wherever necessary. Each API endpoint interacts with the database via SQLAlchemy models, The API also includes proper request validation and error handling, making it robust and user-friendly.

Architecture and Features

This quiz master project, QuizWhiz, follows a structured Flask MVC (Model-View-Controller) architecture, ensuring a clean distinction between logic, data, and presentation. The controllers (routes/views) are located systematically in the app.py file, handling requests and responses. The database models are defined clearly in app.py, using SQLAlchemy to manage users, quizzes, questions, and results. The templates (HTML files) are stored in the templates/ folder, utilizing Jinja2 for dynamic rendering. Static assets like CSS, JavaScript, and images are placed in the static/ directory. Configuration settings, including database connection and security keys, have also been managed well in app.py. The project is further modularized into separate files for authentication, quiz handling, and user management to maintain scalability and clarity.
The project includes a wide set of features for both students and administrators.

- Default Features:
    - User Authentication (for users, using hashed passwords)
    - Quiz Management (for admins)
    - Attempting Quizzes (for users)
    - Scoring System (for users)
    - Easy and Efficient Dashboard (for users and admins)
- Additional Features:
    - Deadline-based Quizzes
    - Multiple Quiz Categories
    - Responsive UI with Bootstrap
    - Flash Messages & Alerts

Each feature is implemented using Flask's request-handling mechanisms, SQLAlchemy for data management, and Jinja2 for rendering interactive pages.

Video

https://drive.google.com/file/d/1cke6Wc1d0pkTeViWB6v_WGvjvTjeTPRb/view?usp=sharing