Author

Aaryesh Shukla

DS23F2003825

23f2003825@ds.study.iitm.ac.in

A dedicated data science enthusiast also actively pursuing a B. Tech degree.

Description

Quiz Master - V1 is a Flask-HTML-based multi-user web application designed for exam preparation across various courses. The application serves as an interactive quiz management system where users can attempt quizzes, track their scores, and improve their knowledge.

The platform supports role-based authentication, allowing students to take quizzes and administrators to manage quizzes, questions, and user data. It is built with a scalable and modular architecture, ensuring maintainability and ease of future enhancements.

The system is designed to provide a structured and user-friendly approach to learning and assessment while offering administrators full control over quiz content and user activity.

Technologies used

- **Flask** – Backend framework for handling requests.
- **Flask-SQLAlchemy** – It simplifies database interactions by using Python classes instead of SQL queries.
- **Bootstrap** – Provides responsive UI design.
- **Chart.js** – Visualizes summary at both ends User and Admin
- **SQLite** – Stores user and quiz and quiz attempt data.
- **Flask Blueprint** – Organizes routes into modular components for better code structure.

DB Schema Design

1)Users Table:

Stores user credentials and role-based access data.

- **Columns:** id, username, password, fullname, type (admin/user), dob, qualification

2)Subjects Table:

Stores details of subjects available in the system.

- **Columns:** subject_id, name

3)Chapters Table:

Links to the subjects table and contains chapter details.

- **Columns:** id, subject_id, chapter_name

4)Quizzes Table:

Stores quiz-related data such as subject, chapter, and creator.

- **Columns:** id, subject_id, chapter_id, admin_id(created by), num_questions, duration

5)Question Table:

Contains the actual quiz questions, their options, and correct answers. It also connects it to the Quiz table.

- **Columns:**]id, quiz_id, question_text, option_a, option_b, option_c, option_d, correct_answer

6)User Attempts Table:

Stores user quiz attempts, linking them to users and quizzes, and holds their scores.

- **Columns:** id, user_id, quiz_id, score, timestamp

User Attempt Table: Links to the user and the quiz which attempted and holds the score.

API Design

The project includes RESTful APIs to manage subjects, chapters, quizzes, questions, user attempts, and results, allowing seamless data exchange between the frontend and backend.

Implemented API Endpoints:

- **Subjects API:** CRUD operations for adding, updating, deleting, and retrieving subjects.
- **Chapters API:** Manages chapters linked to subjects.
- **Quizzes API:** Allows admins to create, update, delete, and fetch quizzes.
- **Questions API:** Handles quiz questions, ensuring dynamic quiz generation.
- **User Attempts API:** Stores user quiz attempts, calculates scores, and retrieves past results.

API Implementation detail:

Each API follows the standard Flask route structure, processing requests via **GET** and **POST** methods. Flask-SQLAlchemy is used for database interactions, where:

- **db.session.add(object):** Used to add new records.
- **db.session.commit()** : Saves changes to the database.
- **query.get(id)** : Ensures retrieval of database records.

Architecture

The project follows the **MVC (Model-View-Controller)** architecture:

- **Models (Quiz master/models):** Defined using SQLAlchemy for handling database operations.
- **Views (Quiz master/templates):** HTML templates styled using Bootstrap.
- **Controllers (Quiz master/controllers):** Flask routes handle request processing and data manipulation.
- **Main Application (app.py):** Initializes the Flask app, registers blueprints for modularity, and configures database settings.

Features:

Core features:

- **User Authentication**- Supports user registration, login, logout, and role-based access (Admin/User).
- **Admin Dashboard**- Admin can edit and create subjects, chapters, and quizzes.
- **User Dashboard**- Users can attempt quizzes, and their responses are stored in the database. Users can view their past quiz attempts and scores.
- **Quiz management**- Each quiz consists of multiple-choice questions, and scores are calculated automatically
- **User management**-Admin can make an already existing user

Data Visualisation with Chart.js

- It shows quiz attempts per subject (pie chart) for users.
- It shows average top score per subject (bar graph) and number of quizzes per subject pie chart

Video:

https://drive.google.com/file/d/1YPqyo6KTPsjaCsouGOXEWvAC9rX11Ix1/view?usp=sharing