Project Report: Quiz Master – V1

Name: Amirtha Varshini Elavarasu

Roll No.: 24dp1000036

Student email: 24dp1000036@ds.study.iitm.ac.in

About Myself: I am in the final term of my programming diploma, building a strong foundation in web development. With a Bachelor of Commerce (B.Com), I have transitioned into the tech industry to expand my technical expertise and career opportunities.

DESCRIPTION

This project involves developing a web application for students to practice quizzes. Students can view chapters by subject, take quizzes, and review their marks, while the admin manages subjects, chapters, quizzes, and questions.

Technologies used:

Backend Technologies

- Python - The primary language for backend development.
- Flask – A lightweight web framework used to build the application.

Flask Extensions & Libraries

- Flask-SQLALchemy – Provides an ORM for managing the database efficiently.
- Flask-Bcrypt – Used for secure password hashing.
- Chart.Js – Used for interactive Charts for both admin and user summaries.

Datebase Management

- SQLALchemy – ORM for handling database operations.
- SQlite – Used as the database.

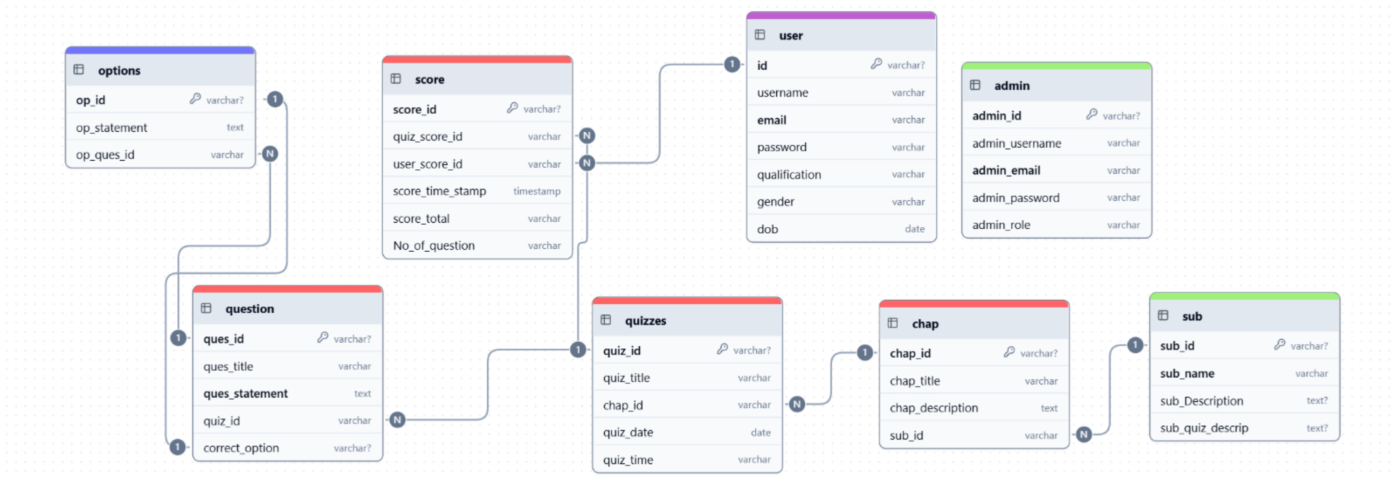
Security & Utilities

- Werkzeug Security – Used for password hashing and verification.
- Datetime (python standard library) – Used to manage timestamps for quizzes and other time-related operations.
- Random (python standard library) – Used to chart random coloring.
- Calendar (python standard library) – used for date-related functionalities.

Frontend Technologies

- HTML, CSS – Used for designing the user interface.
- JavaScript – Handles core functionality such as dynamic quiz interactions, timer, search functionality and enhance interactivity.

DB SCHEMA DESIGN



Reason:

Normalization:

- The schema follows **3rd Normal Form (3NF)** by ensuring no redundant data storage.

Scalability:

- **Supports expansion**, allowing more subjects, quizzes, and users without modifying the structure.

Security:

- Separate **admin** and **user** tables prevent unauthorized access.
- Storing **hashed passwords** improves security.

Efficient Querying:

- **Foreign key relationships** make it easy to retrieve related data efficiently.

API Design for Search Functionality

Purpose:

Provides a search feature for admins and students to query **Users, Subjects, Quizzes, Chapters, and Scores** based on keywords.

Implementation:

- **Route:** @app.route('/search', methods=['GET'])
- **Query Parameters:**
  - q → Search query (string)
  - source → Request origin (admin-navbar.html or user-navbar.html)
- **Authorization:**
  - Requires an **admin** or **user** session.

Search Behavior:

- **Admin:** Can search **all** categories.
- **User:** Can search **Subjects, Quizzes, Chapters**, and **their own Scores**.

Response Format:

- Returns JSON with search results categorized into "Users", "Subjects", "Quizzes", "Chapters", and "Scores".

The YAML file for API submitted separately.

ARCHITECTURE AND FEATURES

Project Organization:

This project follows a structured MVC (Model-View-Controller) architecture using Flask as the backend framework. The codebase is organized into distinct directories to ensure modularity and maintainability:

- static/ - Contains CSS and JavaScript files for styling and frontend behavior.
- templates/ - Stores Jinja2 HTML templates for rendering dynamic content.
- models.py – Defines database models using Flask-SQLALchemy, including tables for users, subjects, chapters, quizzes, questions, scores and admin.
- static/script.js – Includes JavaScript functionality for user interactions and Chart.js for visual analytics.
- app.py – The main entry point of the application, responsible for initializing Flask and its extensions. It handles application logic, user interactions, authentication, quiz management, result processing, and CRUD operations. When app.py is executed, it starts a **local development server** at **http://127.0.0.1:5000/**, allowing testing and debugging in a controlled environment.

Features Implemented

Core Features (Default)

- **User Authentication** – Users can register and log in, while the admin has a predefined login.
- **Quiz Management (Admin)** – The admin can create, edit, and delete **subjects, chapters, quizzes, and questions**.
- **Quiz Attempt (User)** – Users can view quizzes along with their corresponding subjects and chapters, then choose which quiz they want to attempt with a time limit.
- **Score Tracking** – Quiz results are stored and displayed for users in their dashboards.
- **Admin Dashboard** – Provides an overview of subjects, quizzes, users, and performance summaries.
- **User Dashboard** – Displays available quizzes, quiz history, scores, and summaries.
- **Search Functionality** – JavaScript-based search to quickly find quizzes, subjects, and users for both admin
- **Data Persistence** – All quiz attempts and user details are stored using SQLite.

Additional Features

- **Interactive Charts** – Implemented using **Chart.js** to visualize quiz performance for both admin and users.
- **View Quiz details** - Admin can view the details of quiz from quiz dashboard.
- **Average Score –** Admin can view average score of all users in summary page.

Video

Project Demo Link :- https://drive.google.com/file/d/129nNfK6ruZK28WMNGQoueyRPkj4r2VbQ/view?usp=sharing