Project Report

Author

Name - Veer Vikram Singh
Roll No. - 23f3000375
Student Email - 23f3000375@ds.study.iitm.ac.in
I am a problem solver by passion, it's my first time building a Web-Application. Learned so many things in the journey of building this project and excited for the next milestones.

Description

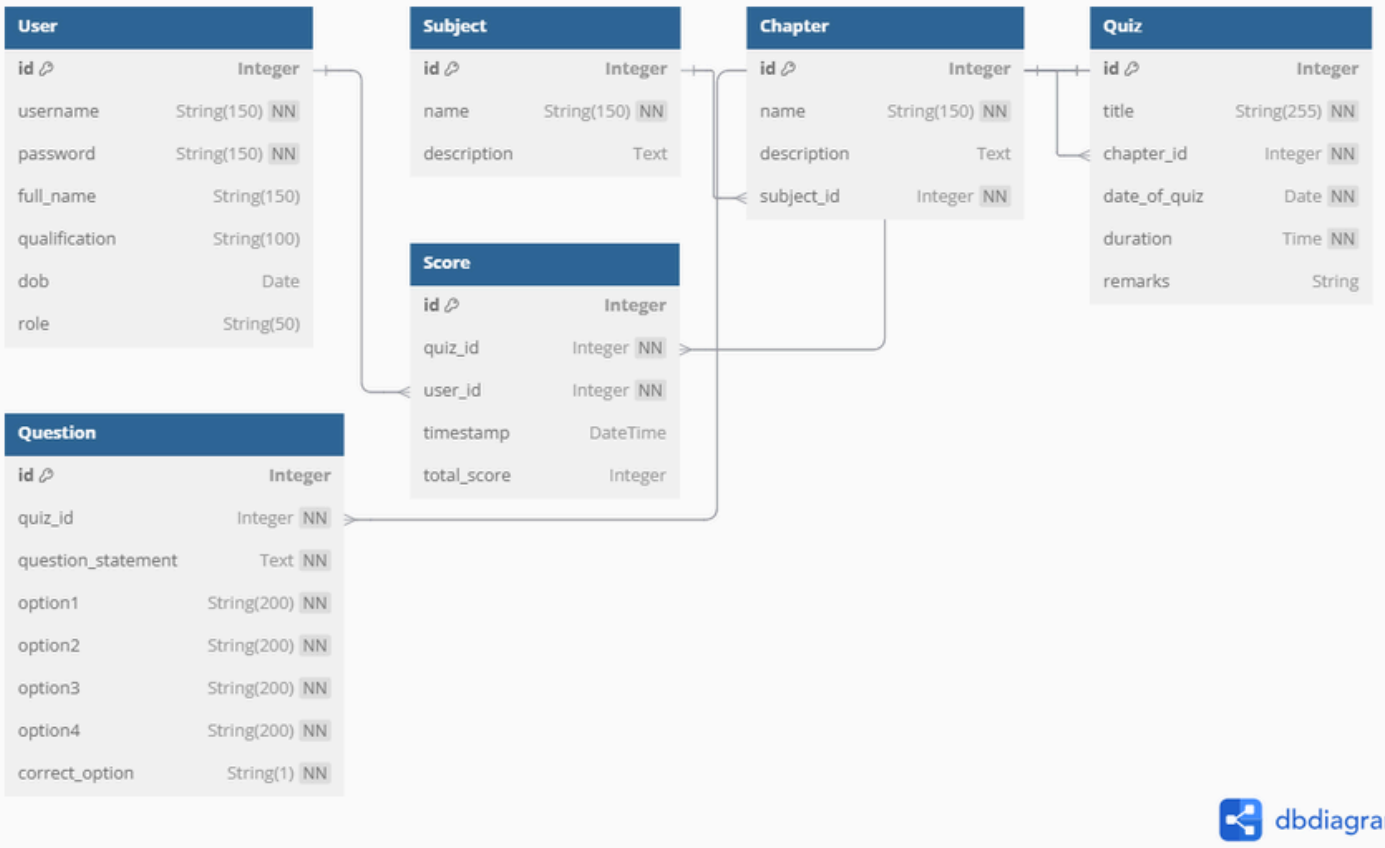
This project is a multi-user Flask-based application designed as an exam prep platform, there will be an admin and registered users. The admin manages subjects, chapters, quizzes, and MCQ questions and can track scores of all users, while users register, log in, attempt quizzes, and view scores.

Technologies Used

- **Flask:** A lightweight Python web framework used as the application back-end to handle routing, request processing, and server-side logic for quiz management and user interactions.
- **Jinja2:** A templating engine integrated with Flask to dynamically render HTML pages, enabling reusable and data-driven front-end templates.
- **HTML:** The core markup language for structuring the application's front-end, providing the foundation for user interfaces like login forms and dashboards.
- **CSS:** Used for styling the application, ensuring a visually appealing and responsive design for both admin and user views.
- **Bootstrap:** A CSS framework employed to enhance front-end development with pre-built components, making the UI consistent, responsive, and user-friendly.
- **SQLite:** A lightweight relational database chosen for storing user data, subjects, chapters, quizzes, questions, and scores, meeting the project's requirement for local demo capability.
- **Flask-SQLAlchemy:** A Flask extension potentially used to simplify database operations, providing an ORM(Object-Relational Mapping) layer for programmatic table creation and management.

Purpose behind using : These technologies collectively enable a robust, scalable, and interactive quiz management system. Flask and SQLite ensure efficient back-end and data handling, while Jinja2, HTML, CSS, and Bootstrap create an intuitive front-end. Flask-SQLAlchemy streamlines database interactions, aligning with the project's mandate for programmatic database creation because of ORM.

DB Schema Design



Models

user : Stores user details (username, password, full name, qualification, dob, etc.)

subject : Stores name and a description about subject.

chapter : Stores name of chapter and a description about it.

quiz : Have a title of quiz, chapter_id from which chapter it is, date_of_quiz, duration and remarks(optional).

score : Stores quiz id, user id, time when user submitted the quiz and total score.

question : Stores quiz id, statement of the question, four options and one correct option.

API Design

I've created a RESTful API for a quiz management system using Flask, with endpoints for user authentication (/login, /register, /logout), admin operations (/admin/subject/new, /admin/quiz/new, /add_question/<quiz_id>), and user functionalities (/start_quiz/<quiz_id>, /scores). It uses Flask-SQLAlchemy for database operations (models: User, Subject, Chapter, Quiz, Question, Score) and Flask-Login for session management. The API supports CRUD operations via HTTP methods (GET, POST).
The yaml file consisting the responses is there in the root folder by name quiz_master_api.yaml

Architecture and Features

Project Architecture

The Quiz Master project is a Flask-based web app with a clean structure. The core logic is in app.py, managing routing, requests, and business rules.

It uses Flask-SQLAlchemy for database handling (SQLite: quiz_master.sqlite3 in the instance folder), with models like User, Subject, and Quiz etc. defined in app.py. Templates are organized in templates (subfolders: admin for admin_dashboard.html, new_quiz.html, etc; user for user_dashboard.html, start_quiz.html, etc). Static files (CSS, JS) follow Flask's static convention. Flask-Login handles sessions, and Flask-Migrate supports database migrations for scalability.

Features Implemented

The project offers a solid quiz system. Default features include user authentication (login, signup, logout) with Flask-Login and Werkzeug hashing, role-based access (admin/user), and a quiz-taking flow (/start_quiz/<quiz_id>) with timers and score storage (Score model). Admins manage subjects, quizzes, and questions via CRUD (e.g., /admin/subject/new, /add_question/<quiz_id>). Extra features include search for admins (/admin/search) and users (/search) using SQLAlchemy, a summary dashboard (/summary) with score charts, and real-time quiz progress tracking via sessions. Built with Flask routing, SQLAlchemy ORM, and Jinja2 templating for a smooth experience.

Video

Drive link : https://drive.google.com/file/d/1Cg9SeogolVXDGStObusMccgm7X0Eu2P9/view?usp=sharing