Quiz Master V1 - Project Report

Student details

Name - Abhay Sahu

Roll no. - 23f1001119

Email - 23f1001119@ds.study.iitm.ac.in

About me - Coding is where I find my creative flow. I love the chance to work on complex problems and find elegant solutions. Well, curiosity lands me learning a new programming language or framework. From designing intuitive websites to exploring the depths of machine learning, I am passionate about pushing the boundaries of technology. But when I am not glued to the screen, you will probably find me on the football pitch or on the dance floor.

Project Description

The Quiz Master V1 project is a multi-user web application designed to serve as an exam preparation platform for multiple courses. The application features two distinct user roles: an administrator (Quiz Master) with full system control and regular users who can attempt quizzes. The administrator can create subjects, chapters, and quizzes, while users can register, log in, select subjects/chapters of interest, and attempt quizzes to test their knowledge.

How I approached the problem statement

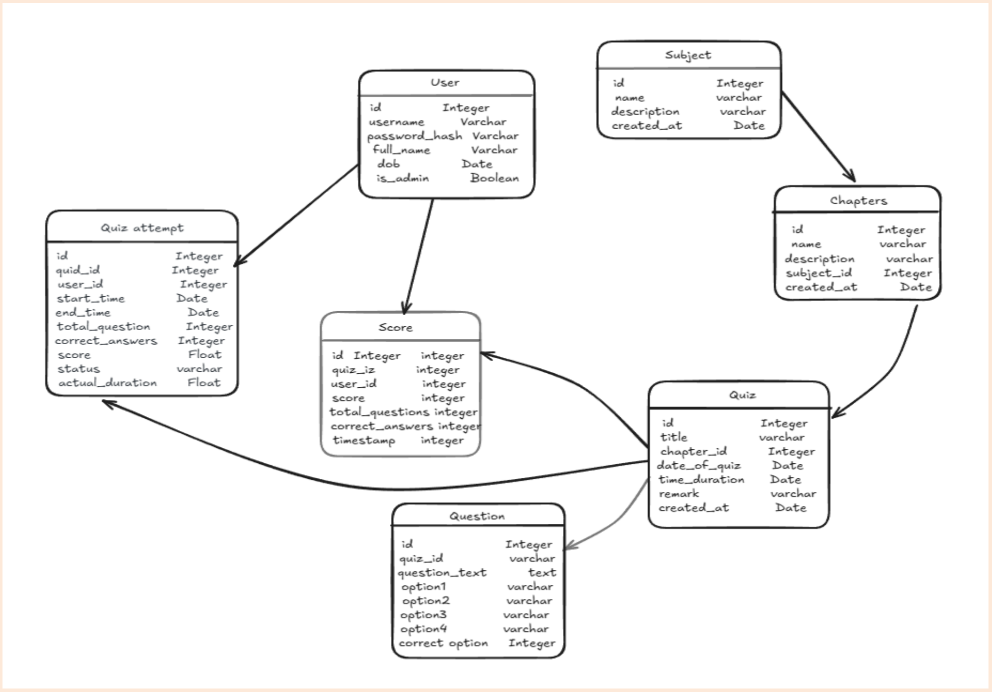
I began by breaking down the problem into core components: user management, quiz functionality, admin controls, and background jobs, creating the architecture of the app on a paper. After the architecture I moved on to the database schema design. Prioritized building a robust database schema, carefully designing relationships between users, subjects, chapters, quizzes, and results to ensure scalability and maintainability. I initialized the project on GitHub. Implemented authentication and authorization, establishing a secure foundation for role-based access control (RBAC) between admin and User. Then developed the backend part, I moved to the frontend part. I created a responsive frontend using HTML and CSS, emphasizing user experience with intuitive navigation, real-time feedback, and a consistent design language across both User and admin interfaces. First designed the interface in Exceli draw and then started coding it out. My flask and coding skills weren't strong initially, so followed a course too. Then added data visualization using Chart.js to provide meaningful insights to both admin and users, helping track progress and performance metrics effectively. Finally made some minor UI changes, code refactoring and I was done with the project in the span of 30 days and approximately 130 hours. Ultimately, my proactive approach paid off, and I completed the project. This experience underscored the importance of perseverance and adaptability in overcoming challenges and achieving project goals.

My approach to implementing this project was structured as follows:

1. **Requirement Analysis:** First, I analyzed the project requirements to identify necessary models, relationships, and functionalities.
2. **Database Design:** Created an entity-relationship diagram to visualize the relationships between different data models (Subject, Chapter, Quiz, Question, User, etc.).
3. **Application Structure:** Designed the application using the Flask Blueprint pattern to maintain a clean separation of concerns between different functional areas.
4. **Secure Authentication:** Implemented user authentication using Flask-Login with proper role-based access control to distinguish between regular users and administrators.
5. **User Interface:** Developed responsive views using Bootstrap and Jinja2 templates that adjust appropriately to different screen sizes.
6. **Data Visualization:** Integrated Chart.js to provide visual analytics for both admin and user dashboards.
7. **Testing:** Conducted thorough testing to ensure all core functionalities work as expected.

DB Schema Design

The database schema is designed to effectively manage and organize the application's data by encompassing several key tables: users, subjects, chapters, quizzes, questions, scores and quiz_attempt. Each table is structured to capture specific attributes relevant to its entity, ensuring comprehensive data representation. Relationships between these tables are established through foreign keys, allowing for seamless navigation and data integrity. For instance, the quizzes table links to subjects and chapters, while the scores table connects to both quiz_attempt and users to track individual responses. This interconnected design enables efficient querying and reporting, facilitating the tracking of user activities and performance metrics within the system. Overall, the schema is optimized for scalability and maintainability, accommodating future enhancements and additional features.



Frameworks and Libraries Used

Core Technologies

• Flask: A lightweight backend framework for building web applications with Python.

• SQLite: Database management system for storing application data
• Jinja2: Template engine for rendering views
• SQLAlchemy: ORM (Object-Relational Mapping) tool for database interactions.

Frontend

• Bootstrap 5: Frontend framework for responsive design
• HTML5/CSS: Frontend markup and styling
• JavaScript: The Little use for Client-side validation and interactions
• Chart.js: Data visualization for dashboards

Extensions

• Flask-Login: User authentication and session management
• Flask-WTF: Form handling and validation
• Flask-SQLAlchemy: SQLAlchemy integration for Flask
• Werkzeug: Password hashing and security utilities

API Resource Endpoints

Subject APIs

• GET /api/subjects - Get all subjects

Chapter APIs

• GET /api/chapters - Get all chapters

Quiz APIs

• GET /api/quizzes - Get all quizzes

Key Features Implemented

Core Features

• Admin and user authentication with role-based access control
• Subject, chapter, and quiz management (CRUD operations)
• Question creation and management for quizzes
• Timed quiz interface for users
• Score recording and history viewing
• Dashboard with analytics for both admin and users

Challenges and Solutions

Challenge 1 : Database Relationships
Solution: Used SQLAlchemy's relationship constructs with cascade delete options to maintain referential integrity.

Challenge 2 : Quiz Timer Implementation
Solution: Used a combination of server-side timestamp tracking and client-side JavaScript to create a robust timer system.

Challenge 3 : Data Visualization
Solution: Leveraged Chart.js with custom data preprocessing to generate insightful charts showing performance trends and subject distribution.

Conclusion

The Quiz Master V1 application successfully fulfills all the core requirements specified in the project statement. Future enhancements could include adding more question types, a sophisticated scoring system, and social features to enable competition among users.

Demonstration Link - https://www.loom.com/share/219fca00d11b4e40b91b4b54518d282a?sid=2e390977-aa94-4a32-8a02-544f73d71656