

## Author

**Name:** Riya Chandrabel

**Roll Number:** 22F3003173

**Student Email:** 22f3003173@ds.study.iitm.ac.in

**Self Introduction:** I am a full-time student in this program, based in MP, India. This is my first experience building and coding a web-based application. I found the process both challenging and rewarding, especially as I navigated the complexities of backend logic, database integration, and front-end design from scratch. The journey taught me the importance of structured planning, debugging, and leveraging community resources to overcome obstacles.

## Description

According to my understanding, the aim of this project was to develop an educational web application, **QuizMaster**, that allows users to take quizzes on various subjects and tracks their progress, while providing administrators with tools to manage content (subjects, chapters, quizzes, and questions) and monitor user performance. The application includes features for user authentication, quiz-taking with a countdown timer, and graphical representations of progress, making it a comprehensive learning management tool. The project evolved from a basic quiz system to include advanced admin functionalities and visual analytics, reflecting my growth in understanding web development concepts.

## Technologies used

Technologies Used	Purpose behind using the technology
Python	Used for backend development to handle business logic, route management, and database interactions. Its simplicity and extensive libraries made it ideal for a beginner project.
Flask	This framework was particularly useful in linking my backend python code to my front end HTML.
flask_sqlalchemy	Facilitated the integration of SQLite with the Flask app, allowing easy creation of database tables, relationships, and queries.

Sqlite	Chosen as the database to store and retrieve data (users, subjects, quizzes, etc.), providing a lightweight solution suitable for this project.
Chart.js	Used to create interactive charts (e.g., user performance and score progression), providing visual insights into quiz results.
HTML	Formed the backbone of the front-end structure, displaying data from the backend to users and admins.
Jinja 2	Enabled the creation of reusable templates, reducing redundancy in HTML code and allowing dynamic data display based on backend responses.
Bootstrap	While HTML does help in displaying data, it may not be aesthetically pleasing all the time. Bootstrap helped me make the app look better and was a quick fix in making the app look better instead of writing lines of CSS code

## Development Process

The development of QuizMaster followed an iterative approach:

1. **Planning:** I began by outlining the database schema and defining user and admin roles based on the project requirements.
2. **Backend Development:** I implemented the user.py blueprint, starting with authentication and quiz logic, then added the countdown timer and graph functionality.
3. **Frontend Design:** Using Jinja2 templates, I created start\_quiz.html and admin templates, integrating Bootstrap for styling and Chart.js for visualizations.
4. **Testing:** I tested individual features (e.g., timer, graph) with mock data, identifying and fixing issues like the 500 error in user\_summary.
5. **Refinement:** Incorporated feedback from debugging sessions, adding session-based timer persistence and fixing template errors (e.g., missing canvas in user\_progress.html).

## Challenges Faced

- **Database Integration:** Initial difficulties with SQLAlchemy relationships (e.g., lazy loading issues in user\_summary) required learning about joinedload and debugging 500 errors.
- **Timer Implementation:** Creating a countdown timer that auto-submits and persists across reloads was complex, involving JavaScript and Flask sessions, which took significant trial and error.
- **Chart Rendering:** The absence of a canvas in user\_progress.html caused chart failures, highlighting the need for thorough template validation.
- **Admin Blueprint:** The lack of an admin.py file left many admin features unimplemented, requiring me to plan additional routes and logic.
- **Schema Errors:** OCR discrepancies in the database schema (e.g., user\_quiz\_progress) posed a challenge, necessitating manual corrections.

## DB Schema Design

The database schema for QuizMaster is designed to support the relationships between users, subjects, chapters, quizzes, questions, and progress tracking. The structure is as follows:

- **One-to-Many Relationships:**
  - Each Subject can have multiple Chapters (e.g., "MLP" has "Chapter 1").
  - Each Chapter can have multiple Quizzes (e.g., "Chapter 1" has "Quiz1").
  - Each Quiz can have multiple Questions (e.g., "Quiz1" has multiple-choice questions).
  - Each User can have multiple UserQuizProgress entries (e.g., one per completed quiz).
- **Tables:**

- user: Stores user details (id, username, password, full\_name, qualification, dob, role).
- subject: Stores subjects (id, name, description).
- chapter: Stores chapters with a foreign key to subject (id, name, subject\_id, description).
- quiz: Stores quizzes with a foreign key to chapter (id, title, chapter\_id, date, duration).
- question: Stores questions with a foreign key to quiz (id, quiz\_id, question\_text, option\_1, option\_2, option\_3, option\_4, correct\_option).
- user\_quiz\_progress: Tracks user quiz progress (id, user\_id, quiz\_id, score, completed\_on, user\_answers).
- **Design Notes:**
  - Avoided hardcoding by creating separate tables for subjects and chapters.
  - The user\_answers field in user\_quiz\_progress uses JSON to store user responses flexibly.
  - Potential OCR errors (e.g., user\_quiz\_progres s with a space, missing foreign keys) need correction to align with the model names.

## Architecture and Features

- **Backend Code:** Located in user.py (user blueprint) and planned admin.py (admin blueprint) in the current directory.
- **Templates:** Stored in the templates folder, including start\_quiz.html, user\_dashboard.html, and admin templates (view\_chapters.html, manage\_quiz\_questions.html, etc.).
- **Static Files:** CSS and JavaScript libraries (e.g., Bootstrap, Chart.js) are linked via CDNs.

## Features Implemented:

### User Features:


- **View Subjects and Quizzes:** Access available subjects and quizzes on the dashboard.
- **Take Quizzes:** Start a quiz with a countdown timer based on quiz.duration, answering multiple-choice questions.
- **View Progress:** Review scores and completion status on the dashboard, with a graphical summary via Chart.js.
- **Review Results:** View completed quizzes with user answers and correct answers.
- **Authentication:** Sign up and log in with username and password.

### Admin Features:

- **Subject Management:** Create, edit, and delete subjects.
- **Chapter Management:** Create, edit, and delete chapters, with options to view and create quizzes.
- **Quiz Management:** Manage questions, add new questions, edit or delete existing ones, and delete quizzes.

- **User Progress Monitoring:** View a table of user progress and a score progression chart.
- **Search Users:** Search users by name via a modal on the admin dashboard.
- **User Performance Graph:** Display average scores per user on the dashboard.

## Video

Video link-  Quiz Master - Google Chrome 2025-03-15 19-00-54.mp4