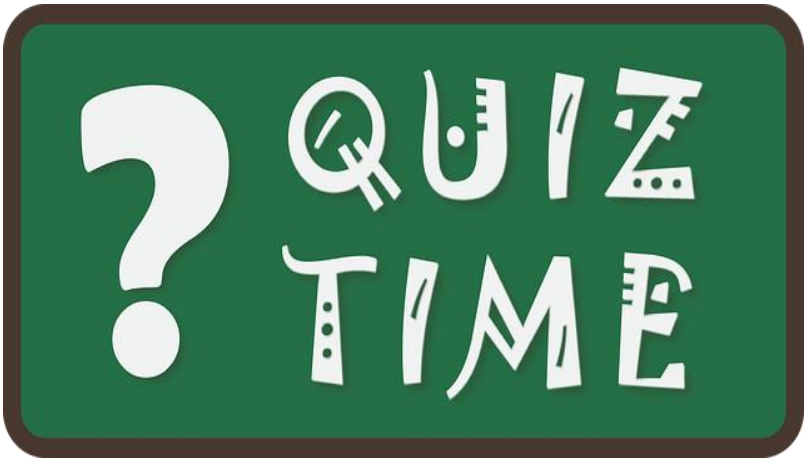


# Final Project Report

BY RAGHAV DIXIT

21f3000264 |Modern Application Development - I | Jan, 2025 batch

## AUTHOR

Name – RAGHAV DIXIT

Roll No - 21F3000264

Email - 21f3000264@ds.study.iitm.ac.in

About me – I've always been passionate about technology and truly enjoy coding. Turning ideas into reality through code gives me a sense of excitement and fulfilment. Exploring new challenges and solving problems with code is what keeps me motivated.

## DESCRIPTION

In this project, we need to build a WebApp that acts as an exam preparation platform. It will support multiple users, including an administrator and regular users. Users can register, attempt quizzes for different courses, and track their scores, while the administrator can manage users, courses, and questions with full CRUD functionality.

## TECHNOLOGIES USED

Here are the technologies I used in this project.

- Python – Core programming language used for building the project.
- Flask – Main framework for developing the WebApp.
- SQLite – Used for creating and managing the database.
- Flask-SQLAlchemy – ORM tool for connecting and interacting with the database.
- HTML/CSS – Used for designing the frontend and handling client-side interactions.
- Jinja – Templating engine for rendering dynamic content.
- Bootstrap – Used for styling and responsive design.
- Matplotlib's pyplot – Used for visualizing data.

## ARCHITECTURE

- **main.py**: Contains both the Flask views (routes) and database schema definitions. All logic for request handling and database interaction is centralized in this file for simplicity.
- **Templates Folder**: The templates/ folder is used to serve HTML files, such as admin_dashboard.html and admin_search.html.
- **Static Folder**: The static/ folder contains static assets like CSS and images:
- CSS files are stored in static/css/ (e.g., style.css).
- Images are stored in static/imgs/ (e.g., imgs1.png, imgs2.png).
- **Instance Folder**: The instance/ folder contains the SQLite database file (quizmaster.db) used to store quiz data and user information.
- **proenv**: The virtual environment (proenv) includes all required Python libraries used to build the application,ensuring consistent dependency management across different environments.

## FEATURES

**1.** Initial Setup:

- ✓ A pre-existing Admin (Quiz Master) account is created at the start.
- ✓ Admin credentials are predefined and cannot be registered manually.

**2**. Main Page (Login and Registration):

The Flask app opens at the main page where users can:

**a.** User Login:
- Username (email) and password are required fields in the login form.
- User should exist in the database or they can register themselves.
- If incorrect credentials are provided, a prompt is displayed to enter correct details.

**b.** Admin Login:
- Username and password are required fields in the form.
- Admin cannot register; admin credentials are predefined at the start.
- Incorrect credentials trigger a prompt.

**c.** Register User:
- Full name, username (email), and password are required in the form.
- Username should not already exist in the database.

**3.** Users:

- ✓ Can log in and attempt quizzes.
- ✓ Can select available quizzes.

**4.** Admin Page:

- ✓ Displays an overview of available quizzes and user activity.
- ✓ Provides options to update quizzes and questions.

**5.** Quiz and Question Management:

**a.** Quizzes:

- Can be added, renamed, or deleted.
- Form values are required when creating or updating a quiz.
- Cannot have negative time duration.
- Delete endpoint only supports GET requests.

**b.** Questions:

- Create: All fields (statement, options) are required.
- Update: All fields are optional; if no values are provided, default values are retained.

## VIDEO LINK

Video demonstration of my project is available here.

https://drive.google.com/file/d/19mY2m4nIfwejR_wdbqp9Ju9AviarnMK2/view?usp=drive_link

## DB SCHEMA DESIGN

I have created a total of 6 tables to make the Quiz Master project more structured and organized. The schema has been carefully normalized to eliminate data redundancy and maintain consistency. Each table is designed to handle specific entities such as users, subjects, chapters, quizzes, questions, and scores, with well-defined relationships and constraints to ensure data integrity. This structured approach makes it easier to manage quiz data, track user performance, and scale the application efficiently.
ER diagram for the schema has been attached by me on the last page, kindly refer to that for more details.

| Table Name | Columns | Details | Constraints |
|---|---|---|---|
| **Users** | id | Unique identifier for the user | Integer, Primary Key |
| | name | Full name of the user | String(80), Not Null |
| | email | Email ID of the user, used for login and account recovery | String(120), Unique, Not Null |
| | password | Password of the user's account | String(120), Not Null |
| | quali | Qualification of the user | String(120), Not Null |
| | dob | Date of birth of the user | String(120), Not Null |
| | is_admin | Specifies if the user is an admin | Boolean, Not Null, Default=False |

| Table Name | Columns | Details | Constraints |
|---|---|---|---|
| **Subjects** | id | Unique identifier for the subject | Integer, Primary Key |
| | name | Name of the subject | String(80), Not Null |
| | description | Description of the subject | String(120), Not Null |

| Table Name | Columns | Details | Constraints |
|---|---|---|---|
| **Chapters** | id | Unique identifier for the chapter | Integer, Primary Key |
| | name | Name of the chapter | String(80), Not Null |
| | description | Description of the chapter | String(120), Not Null |
| | subject_id | Foreign key linking to the subject table | Integer, Not Null |

| Table Name | Columns | Details | Constraints |
|---|---|---|---|
| **Quizzes** | id | Unique identifier for the quiz | Integer, Primary Key |
| | name | Name of the quiz | String(120), Not Null |
| | date_of_quiz | Date when the quiz will be held | Date, Not Null |
| | time_duration | Duration of the quiz in minutes | Integer, Not Null |
| | remarks | Additional remarks for the quiz | String(120), Not Null |
| | chapter_id | Foreign key linking to the chapter table | Integer, Not Null |

| Table Name | Columns | Details | Constraints |
|---|---|---|---|
| **Questions** | id | Unique identifier for the question | Integer, Primary Key |
| | question | Question statement | String(120), Not Null |
| | option1 | First option for the question | String(120), Not Null |
| | option2 | Second option for the question | String(120), Not Null |
| | option3 | Third option for the question | String(120), Not Null |
| | option4 | Fourth option for the question | String(120), Not Null |
| | correct_option | Correct option (1, 2, 3, or 4) | Integer, Not Null |
| | quiz_id | Foreign key linking to the quiz table | Integer, Not Null |

| Table Name | Columns | Details | Constraints |
|---|---|---|---|
| **Scores** | id | Unique identifier for the score | Integer, Primary Key |
| | score | Score obtained by the user in the quiz | Integer, Not Null |
| | user_id | Foreign key linking to the user table | Integer, Not Null |
| | quiz_id | Foreign key linking to the quiz table | Integer, Not Null |
| | time_stamp | Time when the quiz was attempted | DateTime, Not Null |
| | total_score | Total possible score for the quiz | Integer, Not Null |

## ER DIAGRAM

This is the ER diagram for my Database schema that I generated using a web tool.