

Author

Vishal Singh Baraiya

23f2005593

23f2005593@ds.study.iitm.ac.in

I am a student passionate about AI/ML and web development. I enjoy building efficient systems and exploring new technologies to solve problems.

Project: Quiz Master - V1

It is a multi-user web application designed as an exam preparation platform for multiple courses. The app supports two roles: an administrator (Quiz Master) and users. The administrator creates subjects, chapters, quizzes, and questions, while users can register, attempt quizzes, and track their scores.

Frameworks Used

Flask – Backend framework for handling HTTP requests and responses

Jinja2 – Template engine for rendering dynamic web pages

Flask-SQLAlchemy – ORM for database management

Flask-WTF – For handling forms securely

Flask-Login – For user authentication and session management

Bootstrap – Frontend framework for responsive UI design

SQLite – Database to store quiz data

Chart.js – External library for graphical representation of user performance

Purpose of These Technologies:

Flask provides a lightweight and flexible backend solution, while Jinja2 enables dynamic content rendering. Flask-SQLAlchemy ensures efficient database interactions, and Flask-WTF improves form security. Bootstrap enhances the UI, and SQLite serves as a lightweight yet powerful database. Chart.js is used for displaying quiz analytics.

DB Schema Design

ER Diagram:



Tables:

- User** (Stores user details)
 - id** (Integer, Primary Key, Auto Increment)
 - username** (String, Unique, Not Null)
 - email** (String, Unique, Not Null)
 - password_hash** (String, Not Null)
 - full_name** (String, Not Null)
 - qualification** (String, Nullable)
 - dob** (Date, Nullable)
 - is_admin** (Boolean, default=False)
 - scores** relationship('Score', backref='user', lazy=True)
- Subject**
 - id** (Integer, Primary Key, Auto Increment)
 - name** (String, Unique, Not Null)
 - description** (Text, Nullable)
 - chapters** relationship('Chapter', backref='subject', lazy=True, cascade="all, delete-orphan")
- Chapter**
 - id** (Integer, primary_key=True)
 - name** (String(100), nullable=False)

`description` (Text)
`subject_id` (Integer, ForeignKey('subject.id'), nullable=False)
`quizzes` relationship('Quiz', backref='chapter', lazy=True, cascade="all, delete-orphan")

4. Quiz

`id` (Integer, Primary Key, Auto Increment)
`chapter_id` (Integer, Foreign Key referencing Chapter.id, Not Null)
`date_of_quiz` (Date, Not Null)
`start_time` (DateTime, Not Null)
`end_time` (DateTime, Not Null)
`time_duration` (Time, Not Null)
`questions` relationship('Question', backref='quiz', lazy=True, cascade="all, delete-orphan")
`scores` relationship('Score', backref='quiz', lazy=True, cascade="all, delete-orphan")
`remarks` (Text, Nullable)

5. Question

`id` (Integer, Primary Key, Auto Increment)
`quiz_id` (Integer, Foreign Key referencing Quiz.id, Not Null)
`question_statement` (Text, Not Null)
`option_a` (String, Not Null)
`option_b` (String, Not Null)
`option_c` (String, Not Null)
`option_d` (String, Not Null)
`correct_option` (String, Not Null)

6. Score

`id` (Integer, Primary Key, Auto Increment)
`quiz_id` (Integer, Foreign Key referencing Quiz.id, Not Null)
`user_id` (Integer, Foreign Key referencing User.id, Not Null)
`timestamp_of_attempt` (Datetime, Default=current timestamp)
`total_score` (Integer, Not Null)
`total_questions` (Integer, nullable=False)

Reasons Behind the Design:

Normalization ensures data integrity and avoids redundancy,
Foreign keys establish relationships between users, quizzes, and responses and
`created_at` fields allow tracking of data creation.

API Design

The application exposes APIs for:

Subjects: <http://localhost:port/api/subjects>.

Chapters: <http://localhost:port/api/chapters/subject-id>.

Quizzes: <http://localhost:port/api/quizzes/chapter-id>.

Architecture and Features

The project follows the **MVC (Model-View-Controller)** pattern:

Models: Defined in `models.py`, using Flask-SQLAlchemy.

Views: Handled using Flask routes in `routes.py`.

Controllers: Flask handles controller logic within route functions.

Templates: Stored in `templates/` directory, using Jinja2.

Static Files: CSS, JavaScript, and images stored in `static/`.

Implemented Features:

Admin Login: Pre-existing admin account for managing subjects, chapters, quizzes, and users.

User Registration & Authentication: Users can register and log in.

Subject & Chapter Management: Admin can create, edit, and delete subjects and chapters.

Quiz Creation & Management: Admin can create quizzes with multiple-choice questions.

Quiz Attempt & Scoring: Users can attempt quizzes and their scores are recorded.

User Dashboard: Users can track their quiz attempts and performance.

Admin Dashboard: Admin can manage quizzes, view user performance, and see analytics.

Search Functionality: Admin can search for users, subjects, and quizzes.

Graphical Analysis: Chart.js integration for quiz performance visualization.

Additional Features:

Timer for Quizzes: Optional countdown timer during quiz attempts.

Responsive UI: Bootstrap-based design for a seamless experience.

Enhanced Security: Flask-Login for authentication and session management.

Video

[Video Click HERE](#) :

https://drive.google.com/file/d/11BGuqk3qDwYjZu_VVIEo4OPY3_V8sX9f/view?usp=sharing