Author

Name: Aryan Mishra
Roll Number: 21f2000307
Student Email: 21f2000307@ds.study.iitm.ac.in

About: Dedicated and results-driven professional with leadership experience in student governance, event management, and technical projects. Passionate about problem-solving, effective communication, and fostering collaboration.

Description

The project involves creating a Quiz Master web application using Flask, where an admin can manage subjects, chapters, and quizzes, and users can register, log in, and attempt quizzes. The application requires implementing user authentication, quiz creation, quiz attempts, score tracking, and additional features like search functionality and API endpoints along with summaries using Chart.js.

Technologies Used

- Python (Backend development)
  - Flask (Web application framework)
  - Flask-SQLAlchemy (Database ORM and interaction)
  - Jinja2 (HTML templating)
  - Werkzeug (Web application utilities and security)
  - python-dotenv (Environment configuration management)
  - Related dependencies for package compatibility
- SQLite (Data Storage)
  - Stores user data, quiz information, scores, and application state
- Frontend Technologies
  - HTML5 (Structural markup)
  - CSS3 (Custom styling)
  - Bootstrap (Responsive design and pre-built components)
  - JavaScript (Client-side interactivity and form validations; Quiz Timer)
- Development and Deployment Tools
  - Git (Version control)
  - GitHub (Repository hosting)

API Design

Admin: Handles administrative operations like user management, subject creation, and dashboard functionalities.

Auth: Manages user authentication, login, registration, and session handling for both admin and regular users.

Chapter: Provides endpoints for creating, retrieving, updating, and deleting chapters associated with specific subjects.

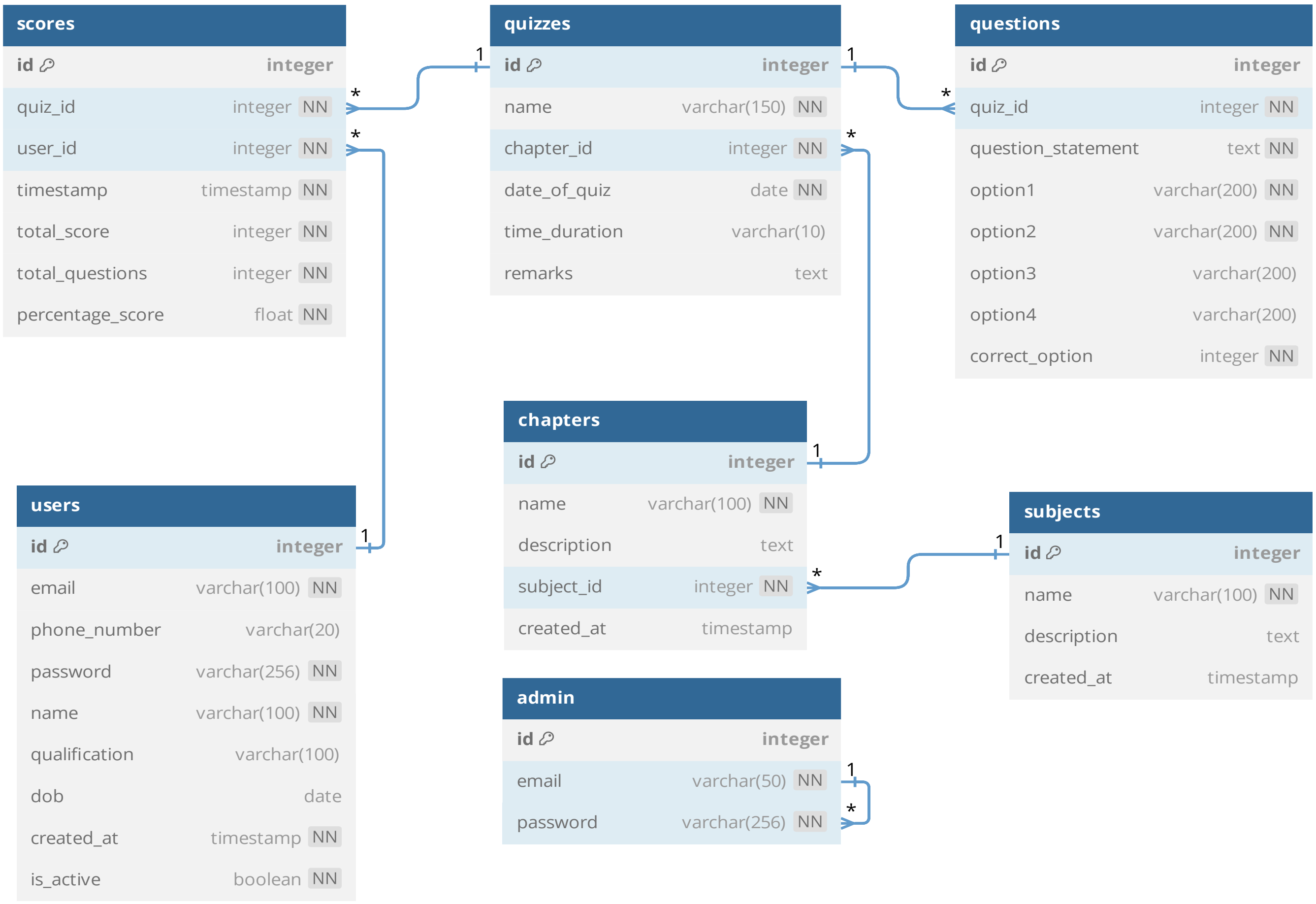
Question Supports CRUD operations for quiz questions, allowing admins to manage question banks for different quizzes.

Quiz: Enables quiz creation, retrieval, and management, including setting quiz parameters and tracking quiz attempts.

Subject: Allows creation, listing, and management of subjects that serve as top-level categories for quizzes.

User: Manages user profile operations, including user information retrieval, updates, and related user-specific functionalities.

DB Schema Design



scores	
id	integer
quiz_id	integer NN
user_id	integer NN
timestamp	timestamp NN
total_score	integer NN
total_questions	integer NN
percentage_score	float NN

quizzes	
id	integer
name	varchar(150) NN
chapter_id	integer NN
date_of_quiz	date NN
time_duration	varchar(10)
remarks	text

questions	
id	integer
quiz_id	integer NN
question_statement	text NN
option1	varchar(200) NN
option2	varchar(200) NN
option3	varchar(200)
option4	varchar(200)
correct_option	integer NN

chapters	
id	integer
name	varchar(100) NN
description	text
subject_id	integer NN
created_at	timestamp

subjects	
id	integer
name	varchar(100) NN
description	text
created_at	timestamp

users	
id	integer
email	varchar(100) NN
phone_number	varchar(20)
password	varchar(256) NN
name	varchar(100) NN
qualification	varchar(100)
dob	date
created_at	timestamp NN
is_active	boolean NN

admin	
id	integer
email	varchar(50) NN
password	varchar(256) NN

The database for the Quiz Master application was meticulously designed to support a scalable educational quiz platform. Drawing inspiration from modular educational management systems, the schema establishes clear relationships between core entities such as users, subjects, chapters, quizzes, and performance metrics. The database structure enables comprehensive quiz management, user tracking, and seamless navigation across educational content through carefully defined one-to-many and many-to-many relationships, ensuring flexibility and robust data management for an interactive learning environment.

Architecture and Features

The **Quiz Master** project follows the **MVC architecture**, ensuring modularity and maintainability. The **Model** (models.py) defines the SQLite database schema. The **View** consists of Jinja2-based templates in templates/, structured into subdirectories for different entities. The **Controller** logic is split into controllers/, with API routes (api/) using **Flask Blueprints** and authentication handling (auth/). Static assets are stored in static/, and app.py serves as the application entry point.

Core features include **user authentication**, **admin dashboard**, **quiz management**, **user quiz attempts**, and **score tracking**. The admin manages subjects, chapters, quizzes, and users, while users can attempt quizzes and view scores. Additional features include **search functionality**, **quiz timing**, and **summary charts**. The API, structured via Flask Blueprints, organizes endpoints (no JSON). The UI, built with **Bootstrap**, ensures responsiveness, and backend validation enforces security.

Video

Video Link: