Author

Name: - Suraj Kumar
Roll no: - 24dp1000038
Email: - 24dp1000038@ds.study.iitm.ac.in

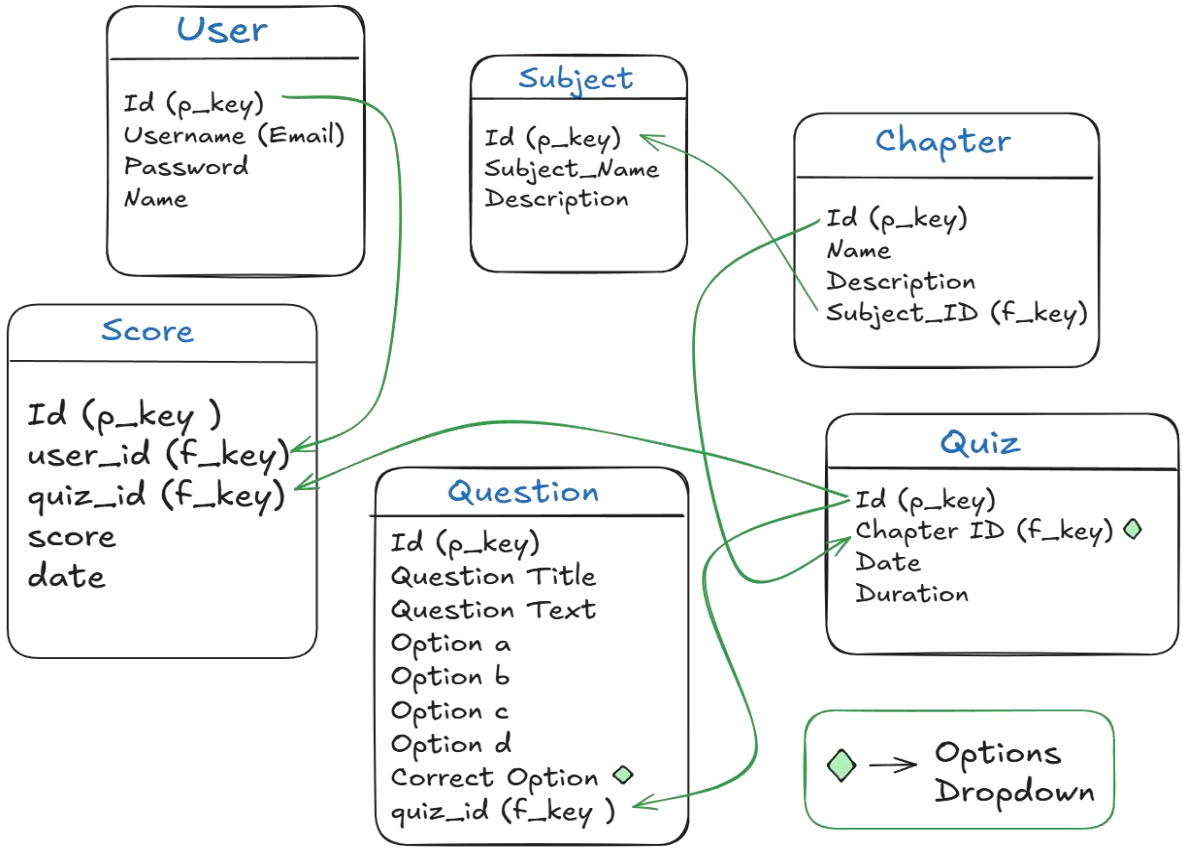
Description

The **Quiz Master** project is a Flask-based multi-user quiz application with admin and user roles. It allows admins to create and manage quizzes, while users can attempt quizzes and track their performance. The application includes user authentication, a search feature, and an intuitive UI using Jinja2 templating with an SQLite database.

Technologies Used & Purpose

1. **Flask** – web framework for building the quiz application.
2. **Flask-SQLAlchemy** – SQLite database.
3. **SQLite** – Lightweight relational database for storing quizzes, users, and results.
4. **Flask-Login** – Manages user authentication.
5. **Jinja2** – Templating engine for dynamic HTML rendering.
6. **Bootstrap** – Enhances UI/UX with responsive and modern design.
7. **JavaScript (Vanilla/ES6)** – Handles timer in quiz and password show and hide.
8. **HTML & CSS** – Structures and styles the frontend of the application.

ER Diagram of Database



Implementation Details

- The API is built using **Flask** and follows a modular structure with blueprints for different modules (users, quizzes, questions, and scoring).

- **Database Handling**: SQLite is used for data storage.
- **Error Handling**: Custom error messages and HTTP status codes (`404 Not Found`).
- **Input Validation**: Using frontend and backend validation of input data.

Architecture and Features

The **Quiz Master project is organized using a Flask-based MVC (Model-View-Controller) architecture**, ensuring separation of concerns and modularity. The key components are structured as follows:

- **`app.py`** – The main entry point for the Flask application, initializing routes, configurations, and database connections.
- **`backend/`** – Contains core backend logic:
  - **`controllers.py`** – Defines the application's request handling logic, including user authentication, quiz management, and API endpoints.
  - **`models.py`** – Handles database interactions, defining the structure of quiz-related tables in SQLite.
- **`instance/`** – Stores the SQLite database (`quiz_db.sqlite3`), keeping data persistent.
- **`static/`** – Contains static assets to enhance UI/UX:
  - **`css/style.css`** – Defines styles for the frontend.
  - **`image/`** – Stores visual assets like the logo, error indicators, and UI icons.
- **`templates/`** – Jinja2-based HTML templates, dynamically rendering quiz-related content.
- **`.gitignore`** – Specifies files to be excluded from version control.
- **`requirements.txt`** – Lists required dependencies for setting up the project environment.
- **`README.md`** – Provides documentation for installation, usage.

Implemented Features

The **Quiz Master application** includes a variety of features to support quiz creation, user management, and interactive gameplay. Below is a breakdown of the default and additional features along with their implementations:

✅ **User Authentication (Login & Logout)**
- Users provide credentials, which are validated against the database.
- Passwords are securely hashed before storage.

✅ **Quiz Creation & Management (Admin Role)**
- Admins can create, edit, and delete quizzes.
- Data is stored in **SQLite** using models defined in `models.py`.
- Admin routes are managed within `controllers.py`.

✅ **Taking Quizzes (User Role)**
- Users can attempt quizzes, and their responses are recorded.
- Each quiz dynamically loads from the database via Jinja2 templates.

✅ **Dynamic Search & Filtering**
- Admin can search for quizzes by title or category.

Video Demo:

https://drive.google.com/file/d/1cjmIIiWDeUsmxH4zh5vDeFcipqbcnErI/view