Author :

Name : Harismitha D
Roll number : 23f2005726
student email : 23f2005726@ds.study.iitm.ac.in
About me : Dual Degrees - 2nd Year B.E. CSE (AI) at Sathyabama Institute and B.S. Degree in Data Science and applications at IIT-Madras

Description :

Quiz Master is a web-based quiz management system for creating, managing, and tracking quizzes by subjects and chapters. It supports authentication, quiz management, and performance analytics.

Technologies Used :

- Flask (v2.0.1) - handling routing, request processing, and template rendering.
- SQLAlchemy (v1.4.23) & Flask-SQLAlchemy (v2.5.1) - Provides an abstraction layer for database interactions and manages relationships between models.
- Matplotlib (v3.4.3) - generating performance analytics and visualizations
- PyJWT (v2.3.0) - ensuring safe user sessions and API access.
- HTML, CSS, Jinja2 - Front end
- Flask Blueprints : Application modularization.

Database Schema Design :

1. User(id, email[unique], password, is_admin) - Manages authentication and role-based access

2. Subject(id, name, description) - Top-level organization of quiz content

3. Chapter(id, name, subject_id[FK]) - Groups quizzes by topics within subjects

4. Quiz(id, chapter_id[FK], date, duration) - Stores individual quiz details with scheduling

5. Question(id, quiz_id[FK], statement, options[1-4], correct_option) - Multiple choice questions

6. Score(id, user_id[FK], quiz_id[FK], total_scored, timestamp) - Tracks user performance

Relationships:

- User 1:N Score (One user can have many scores)

- Quiz 1:N Score (One quiz can have many scores)

- Quiz 1:N Question (One quiz has many questions)

- Chapter 1:N Quiz (One chapter has many quizzes)

- Subject 1:N Chapter (One subject has many chapters)

Design Rationale:

- Hierarchical structure (Subject→Chapter→Quiz→Question) for organized content management

- Foreign key constraints with CASCADE deletion for data integrity

- Minimal but essential fields to keep the schema simple and efficient

API Design :

1. Authentication: /api/auth/login for user authentication and token generation

2. Quiz Management:

   - /api/quizzes for quiz listing and details

   - /api/subjects for subject management

3. User Features: /api/users/me/scores for personal score tracking

4.Admin Controls:

   - /api/admin/users for user management

   - /api/admin/scores for system-wide score monitoring

All endpoints are secured with JWT tokens, with additional admin-only route protection. The API follows RESTful principles, returns JSON responses, and is documented using OpenAPI specification in api.yaml.

Architecture & Features :

Project Structure

The **Quiz Master** project is structured for clarity and scalability. **Controllers** (`controllers/`) manage routing: `auth.py` handles authentication, `api.py` manages API requests, `admin.py` controls admin tasks, and `user.py` manages user interactions. **Models** (`models/`) define database structures like users, quizzes, and scores. **Templates** (`templates/`) store HTML files for UI rendering, while **static/** contains CSS and analytics images. The application starts with `app.py`, configured via `config.py`.

Core Features

- Admin/User login uses Flask routes (/login, /register) with SQLite (sqlite3) for storage; admin pre-inserted via init_db() in database.py.

- Admin dashboard (admin_dashboard.html) offers CRUD via Flask routes (e.g., /admin/subject/new) using SQLite queries (e.g., INSERT INTO subjects).

-User dashboard (user_dashboard.html) lists quizzes from SQLite, with attempts recorded in the scores table. Quizzes use MCQs stored in questions tables, set by Admin with date/duration via forms.

Video :
https://drive.google.com/file/d/12mTaiWYAdKXvkk5RrqSPgWj38xv7zqzL/view?usp=sharing