QUIZ
MASTER V2

Project Report by Rishi Rao,
23F1001335,
Modern Application Development – II,
Jan, 2025 Term

Student details:

Name - Rishi Rao
Roll no. - 23F1001335
Email - 23F1001335@ds.study.iitm.ac.in

About me - I find my creative flow in coding, where I tackle complex problems, solve puzzles, and strive for elegant solutions. My curiosity often leads me to explore new programming languages or frameworks. When I'm not immersed in code, you'll likely find me playing sports.

Project Description:

The Quiz Master - V2 is a multi-user web application designed to facilitate exam preparation across multiple courses. It allows administrators to create and manage subjects, chapters, quizzes, and questions, while users can register, log in, and attempt quizzes. The platform records quiz scores, provides performance insights, and generates reports.

How I approached the problem statement:

Having some basic experience with Flask, I felt relatively comfortable with the backend development aspect of the project. However, I had little to no experience with Vue.js, Redis, and Celery, which initially made me feel a bit uncertain. To overcome this, I started by thoroughly understanding the project requirements and breaking the solution down into smaller, manageable modules.

In the first week, I focused on setting up the basic project structure with Flask and SQLite. I implemented the admin login functionality and created the database models for users, subjects, chapters, quizzes, and scores. During this time, I also familiarized myself with Redis and Celery by going through online tutorials and documentation.

In the following weeks, I gradually introduced Vue.js for the frontend. Although it was initially challenging, I practiced by building simple components and slowly integrated them with the Flask APIs. I spent extra time learning how to connect Vue with Flask for API calls.

For batch jobs, I experimented with Celery and Redis to implement scheduled and asynchronous tasks. I tested the reminder emails and CSV export functionality, ensuring they worked as expected.

Throughout the project, I adopted an iterative approach—developing and testing each module individually before integrating them. This helped me identify and resolve issues early on. Despite the initial challenges, consistent practice and troubleshooting allowed me to build a fully functional application with all the required features. This project significantly enhanced my skills in Flask, Redis, Celery, and Vue.js, and it reinforced the importance of continuous learning and problem-solving.

Frameworks and libraries used:

Flask and Extensions

- **Flask**: Web framework for building the application.
- **flask_security**: For authentication and security management (auth_required, verify_password, hash_password).
- **flask_sqlalchemy**: ORM for database interactions.
- **flask_excel:** Library for handling Excel file operations.

Database and ORM

- **SQLAlchemy**: ORM for managing database operations.

Celery and Asynchronous Tasks

- **Celery-worker**: For background task management.
- **Celery-beat**: For periodic task management.

Data Visualization and PDF Generation

- **matplotlib**: For generating charts and plots.
- **reportlab**: For PDF generation.
- **Html2pdf**: For PDF generation.

Email Handling

- **smtplib**: For sending emails through SMTP.
- **email**: For creating email messages with multiple parts.

File Handling

- **io**: For in-memory file operations.
- **os**: For file system interactions.
- **time**: For handling time-related operations.

API Endpoints:

Subject Endpoints

- GET /api/sub/<sub_id> → Get a specific subject by ID
- DELETE /api/sub/<sub_id> → Delete a specific subject by ID
- PUT /api/sub/<sub_id> → Update a subject by ID
- GET /api/sub → Get all subjects
- POST /api/sub → Create a new subject

Chapter Endpoints

- GET /api/chap/<id> → Get all chapters for a specific subject by sub_id
- DELETE /api/chap/<id> → Delete a specific chapter by ID
- PUT /api/chap/<id> → Update a chapter by ID
- GET /api/chap → Get all chapters
- POST /api/chap → Create a new chapter

Quiz Endpoints

- GET /api/quiz/<id> → Get all quizzes for a specific chapter by chap_id
- POST /api/quiz/<id> → Create a new quiz for a chapter by chap_id
- DELETE /api/quiz/<id> → Delete a specific quiz by ID
- PUT /api/quiz/<id> → Update a quiz by ID

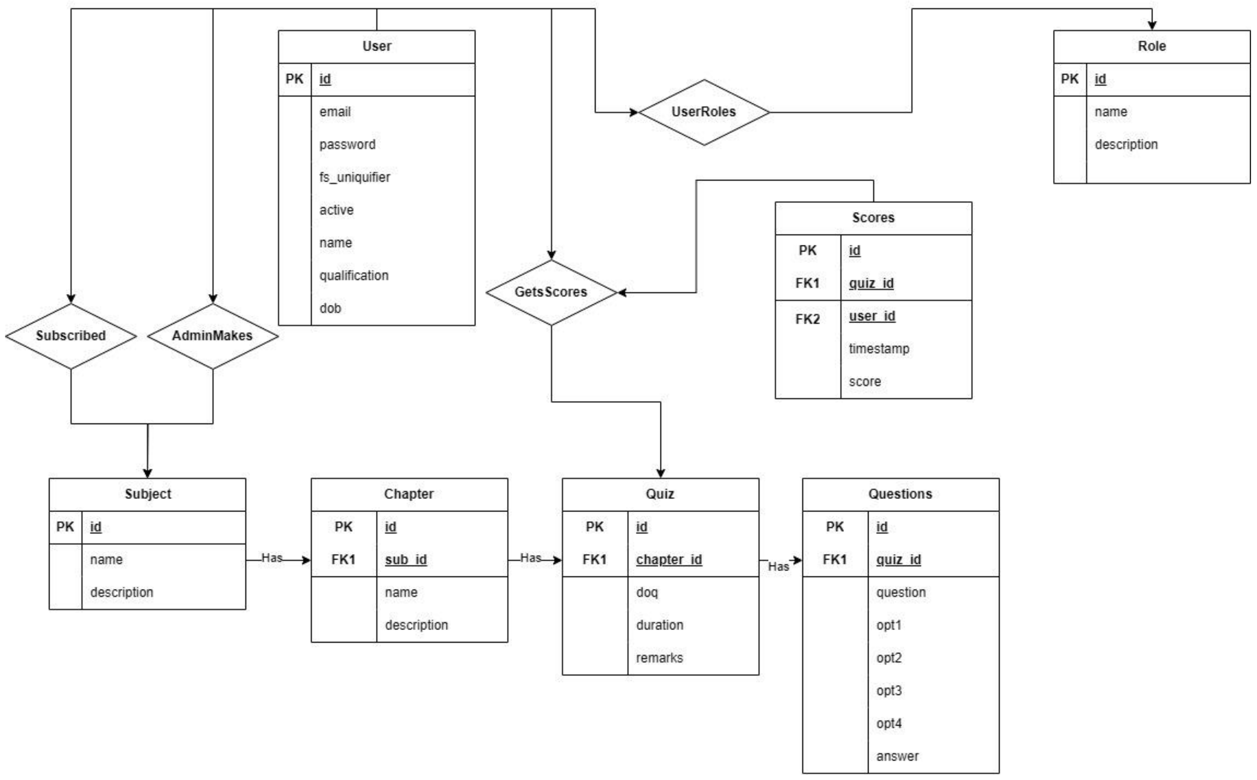
Questions Endpoints

- GET /api/q/<id> → Get all questions for a specific quiz by quiz_id
- POST /api/q/<id> → Create a new question for a quiz by quiz_id
- DELETE /api/q/<id> → Delete a specific question by ID
- PUT /api/q/<id> → Update a question by ID

Subscription Endpoints

- GET /api/subscribe/<user_id>/<sub_id> → Check if a user is subscribed to a subject
- PUT /api/subscribe/<user_id>/<sub_id> → Subscribe a user to a subject

ER Diagram:



Drive link of the presentation video:

https://drive.google.com/file/d/1EGt4VJHS5Arr4PrkjEwlx7QSypajnDU8/view?usp=shari
ng