MODERN APPLICATION DEVELOPMENT - 2

Author

Parth Kacha
23f1002650
23f1002650@ds.study.iitm.ac.in
I am a student in the BS Data Science program at IIT Madras, currently building modern application development project version 2 and pursuing Diploma courses of Data Science. I am also pursuing a BE in Artificial Intelligence and Data Science at Government Engineering College Rajkot, currently in 4th year.

Description

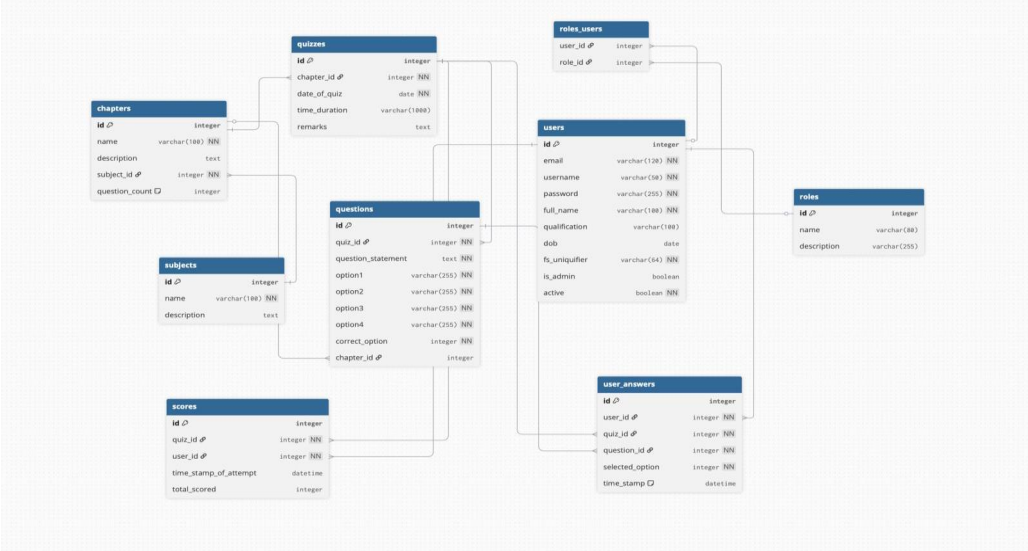
This project implements a full-stack Quiz Management System, enabling admins to create and manage quizzes, and users to attempt them with scoring and analytics.

AI/LLM: ~20-25% (mainly for debugging, API structure, and architectural guidance). Manual coding, design decisions, and feature implementation were prioritized throughout the project.

Technologies used

- **Flask** – Core web framework.
- **Flask-RESTful** – For building REST APIs.
- **Flask-SQLAlchemy** – ORM to interact with SQLite database.
- **Flask-Security-Too** – Authentication, roles, password hashing.
- **SQLite** – Lightweight database for development.
- **Celery** – Task queue for asynchronous/background tasks.
- **Redis** – Message broker for Celery.
- **Flask-Mail / MailHog** – Sending emails for reminders and reports.
- **Flask-Cors** – To allow CORS for API usage from Vue frontend.
- **Vue.js** – Core frontend framework.
- **Vue Router** – SPA navigation between Login, Dashboard, Exam, etc.
- **Bootstrap 5** – Styling (as per project rules).
- **localStorage** – Store authentication token securely.
- **Redis (Caching)** – Used to cache API responses (like home/dashboard data) to improve performance and reduce DB hits.

DB Schema Design



API Design

API Name	Endpoint(s)
HomePage	/
RegisterAPI	/api/register
LoginAPI	/api/login
LogoutAPI	/api/logout
UserDashboard	/api/home
AdminUserAPI	/api/admin/users

SubjectAPI	/api/subjects /api/subjects/<int:sub_id>
ChapterAPI	/api/subjects/<int:subject_id>/chapters /api/subjects/<int:subject_id>/chapters/<int:chapter_id>
QuizAPI	/api/chapters/<int:chapter_id>/quizzes /api/chapters/<int:chapter_id>/quizzes/<int:quiz_id> /api/quizzes /api/quizzes/<int:quiz_id>
QuestionAPI	/api/chapters/<int:chapter_id>/quizzes/<int:quiz_id>/questions /api/chapters/<int:chapter_id>/quizzes/<int:quiz_id>/questions/<int:question_id>
SubmitQuizAPI	/api/submit_quiz
ScoreAPI	/api/chapters/<int:chapter_id>/quizzes/<int:quiz_id>/score /api/quizzes/<int:quiz_id>/scores
QuizAttemptDetailsAPI	/api/chapters/<int:chapter_id>/quizzes/<int:quiz_id>/attempt-details
UserPerformanceAPI	/api/user/quiz-attempts-by-subject
TopScoresAPI	/api/admin/subject-top-scores
UserAttemptsAPI	/api/admin/subject-user-attempts
ExportCSVAPI	/api/export_quiz_csv /api/csv_result/<task_id>

Architecture and Features

The project is divided into two main parts: the backend (Flask) and the frontend (Vue.js). All main logic and APIs are in routes.py and resources.py inside the application folder. The HTML entry point

(index.html) is in the templates folder. The Vue frontend sends and receives data using secure APIs. Redis is used for background tasks and caching.

Main features include user login/register, admin quiz creation, users can attempt quizzes with timers, scores are calculated automatically, and admins can download reports. Extra features like daily email reminders and monthly summary reports are handled in the background using Celery and Redis. Some quiz data is cached using Redis to make it load faster.

Video

https://drive.google.com/drive/folders/1dbYDbAyS0f-24MPpZJy_tn8BDm1mrSaR?usp=sharing