Quiz master- V2

Kwizzy

✦

✦

✦

✦

Project Report by ***Kapil***
23f1001759

Modern Application Development – I

Jan, 2025 Term

Student details

Name - Kapil
Roll no. - 23f1001759
Email - 23f1001759@ds.study.iitm.ac.in
About me - Coding is where I find my creative flow. I love the chance to work on complex problems and find elegant solutions. Well, curiosity lands me learning a new programming language or framework. From designing intuitive websites to exploring the depths of machine learning, I am passionate about pushing the boundaries of technology. But when I am not glued to the screen, you will probably find me on the football pitch or on the dance floor.

Project Description

Kwizzy is an online platform designed for students and administrators, facilitating effective exam preparation across multiple courses. It provides a structured and interactive environment for students to prepare for exams effectively. Administrators can manage courses, quizzes, and user accounts, while students can take assessments, and track their progress. The application features interactive quizzes, performance analytics. With a user-friendly interface and mobile responsiveness, it provides a flexible learning experience tailored to individual needs.

How I approached the problem statement

- I began by breaking down the problem into core components: user management, quiz functionality, admin controls, and background jobs, creating the architecture of the app on a paper. After the architecture I moved on to the database schema design. Prioritized building a robust database schema, carefully designing relationships between users, subjects, chapters, quizzes, and results to ensure scalability and maintainability.
- On 31st Dec, I initialized the project on WSL and GitHub. Implemented authentication and authorization using JWT tokens, establishing a secure foundation for role-based access control (RBAC) between admin and student functionalities. Spent 4 hours on a silly mistake in the implementation of JWT, finally GitHub comments thread helped.
- Then developed the backend API layer using Flask-RESTful, focusing on creating clean, modular endpoints that follow RESTful principles and include proper error handling and validation. After I had coded a decent amount of backend and testing it using Insomnia, I moved to the frontend part.
- I created a responsive frontend using Vue.js, emphasizing user experience with intuitive navigation, real-time feedback, and a consistent design language across both student and admin interfaces. First designed the interface in Figma and then started coding it out. My Vue.js skills weren't strong initially, so followed a course too. Then added data visualization using Chart.js to provide meaningful insights to both admin and users, helping track progress and performance metrics effectively. Enhanced the application with caching mechanisms using Redis to optimize performance and reduce database load, particularly for frequently accessed data like quiz results and user statistics.
- Then finally in the end, I addressed the elephant in the room, "Backend Jobs". Integrated Celery with Redis for handling asynchronous tasks, implementing scheduled jobs for daily reminders and monthly reports, ensuring efficient background processing. Spent a good amount of time here and on the export file as CSV functionality. Finally made some minor UI changes, code refactoring and I was done with the project in the span of 14 days and approximately 120 hours.

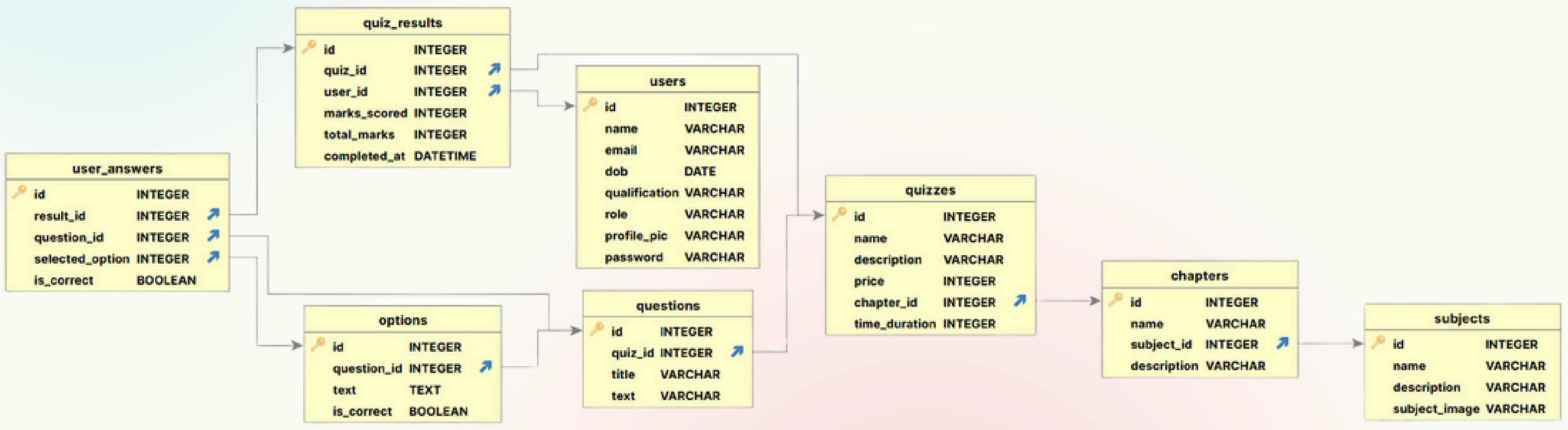
Ultimately, my proactive approach paid off, and I completed the project well ahead of the first deadline. This experience underscored the importance of perseverance and adaptability in overcoming challenges and achieving project goals.

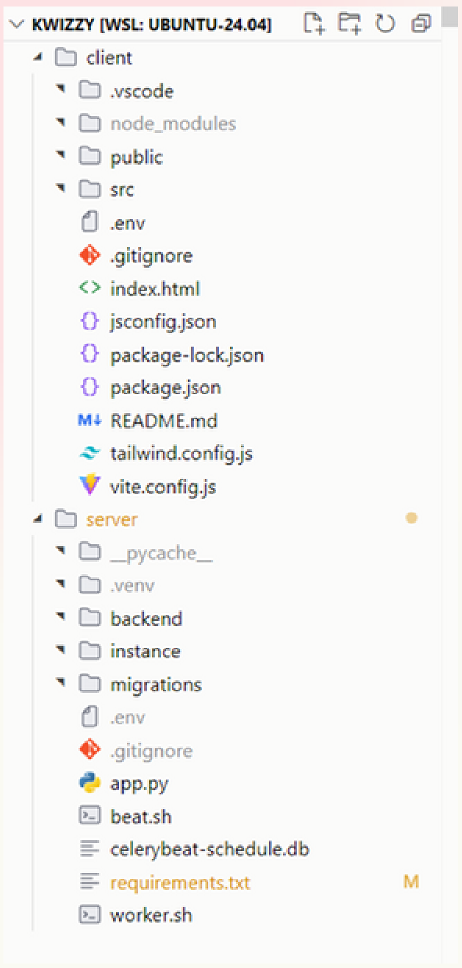
Technologies Used

- **Flask:** A lightweight backend framework for building web applications with Python.
- **SQL Alchemy:** ORM (Object-Relational Mapping) tool for database interactions.
- **SQLite:** Database management system for storing application data.
- **Vue.js:** A progressive JavaScript framework for building user interfaces and enhancing interactivity, complemented by Tailwind CSS for styling.
- **Flask JWT Extended:** An extension that provides tools for managing user sessions and authentication using JSON Web Tokens (JWT).
- **Flask Restful:** A Python library that simplifies the creation of RESTful APIs for web applications.
- **Celery:** An asynchronous task queue that enables the execution of background jobs and scheduled tasks.
- **Redis:** An in-memory data structure store used as a caching database to optimize application performance.
- **Werkzeug:** Utility for securely managing passwords and authentication.
- **ChartJS:** Used for creating interactive charts and visualizations on the admin + user dashboard.
- **Brevo:** An email service provider used for sending notifications and communications to users

DB Schema Design

The database schema is designed to effectively manage and organize the application's data by encompassing several key tables: users, subjects, chapters, quizzes, questions, options, user_answers, quiz_results, and transactions. Each table is structured to capture specific attributes relevant to its entity, ensuring comprehensive data representation. Relationships between these tables are established through foreign keys, allowing for seamless navigation and data integrity. For instance, the quizzes table links to subjects and chapters, while the user_answers table connects to both questions and users to track individual responses. This interconnected design enables efficient querying and reporting, facilitating the tracking of user activities and performance metrics within the system. Overall, the schema is optimized for scalability and maintainability, accommodating future enhancements and additional features.





```
KWIZZY [WSL: UBUNTU-24.04]
client
    .vscode
    node_modules
    public
    src
    .env
    .gitignore
    index.html
    jsconfig.json
    package-lock.json
    package.json
    README.md
    tailwind.config.js
    vite.config.js
server
    __pycache__
    .venv
    backend
    instance
    migrations
    .env
    .gitignore
    app.py
    beat.sh
    celerybeat-schedule.db
    requirements.txt                M
    worker.sh
```

API Endpoints

API Name	**Endpoint(s)**
Student	`/api/students, /api/student/<int:student_id>`
StudentActivity	`/api/student/<int:student_id>/activity`
StudentSubjectPerformance	`/api/student/<int:student_id>/subjects`
Login	`/api/login`
Register	`/api/register`
ForgotPasswordAPI	`/api/auth/forgot-password`
ResetPasswordAPI	`/api/auth/reset-password`
UserApi	`/api/user, /api/user/<int:user_id>`
SubjectApi	`/api/subject, /api/subject/<int:subject_id>`
ChapterApi	`/api/chapter, /api/chapter/<int:chapter_id>`
QuizApi	`/api/quizzes, /api/quizzes/chapter/<int:chapter_id>, /api/quizzes/<int:quiz_id>`
QuestionApi	`/api/quizzes/<int:quiz_id>/questions, /api/quizzes/<int:quiz_id>/questions/<int:question_id>`
OptionApi	`/api/questions/<int:question_id>/options, /api/options/<int:option_id>`
OptionsBulkApi	`/api/questions/<int:question_id>/options/bulk`
FileApi	`/api/uploads/subjects/<path:filename>`
QuizResultApi	`/api/quiz-results, /api/quiz-results/<int:result_id>`

✦ ✦ ✦ ✦

UserAnswerApi	`/api/user-answers, /api/user-answers/<int:answer_id>`
ChartDataApi	`/api/admin/charts, /api/charts/<string:chart_type>`
StudentChartsApi	`/api/student/charts`
TaskAPI	`/api/tasks, /api/tasks/<int:task_id>`
UserQuizExportAPI	`/api/export/user-csv`
AdminQuizExportAPI	`/api/export/admin-csv`
PaymentApi	`/api/payments,` `/api/payments/status/<int:user_id>/<int:quiz_id>`
TransactionHistoryAPI	`/api/payments/history,` `/api/payments/history/<int:user_id>`
TransactionExportAPI	`/api/export/transactions/<int:student_id>`