QUIZMASTER V2

PROJECT REPORT

Modern Application Development - II

Ankita Dey

23f2001106

AUTHOR

Name: Ankita Dey

Roll No.: 23f2001106

Email: 23f2001106@ds.study.iitm.ac.in

I am an aspiring Data Scientist, currently pursuing a BS in Data Science and Applications at IIT Madras. I am eager to explore and contribute to the fields of machine learning, analytics, and artificial intelligence. Beyond coding and data science, I have a deep appreciation for art and painting, which allow me to express creativity and explore new perspectives.

DESCRIPTION

This project is part of the **Modern Application Development - II** course under the IITM BS Degree (Diploma in Programming).

Quizmaster is a role-based, multi-user quiz platform that supports quiz creation, timed attempts, score tracking, and performance analytics. It includes asynchronous jobs for email reminders, monthly reports, and data export. Built with Flask and VueJS, it uses JWT for secure authentication and Redis + Celery for task scheduling and caching.

TECHNOLOGIES USED

These technologies were used to build a secure, efficient, and responsive quiz application.

Backend:

- **Flask** – Handles routing and application logic.
- **Flask-SQLAlchemy** – ORM for managing database interactions efficiently.
- **Flask-Migrate** – Manages database schema migrations.
- **Flask-CORS** – Enables secure communication between frontend and backend.
- **Flask-RESTful** – Defines and manages RESTful APIs.
- **Werkzeug Security** – Ensures secure password hashing and

authentication.

- **JWT (JSON Web Tokens)** – Implements token-based authentication and role-based access control.
- **Celery** – Executes asynchronous and scheduled background tasks.
- **Redis** – Supports task queues and caching with expiry.
- **Requests** – For external API interactions.
- **SMTP** – Sends automated emails (daily reminders, monthly reports).

Frontend:

- **Bootstrap –** Provides styling and responsive design.
- **VueJS –** Builds a dynamic, component-based frontend.

Database & Data Handling:

- **SQLite** – A lightweight, file-based database for managing users, quizzes, scores, and other data.

Templating & Email:

- **Jinja2** – Renders dynamic HTML email templates (e.g., monthly reports).
- **HTML (Email templates only)** – For formatted mail content (PDF/report generation).

Analytics & Visualization:

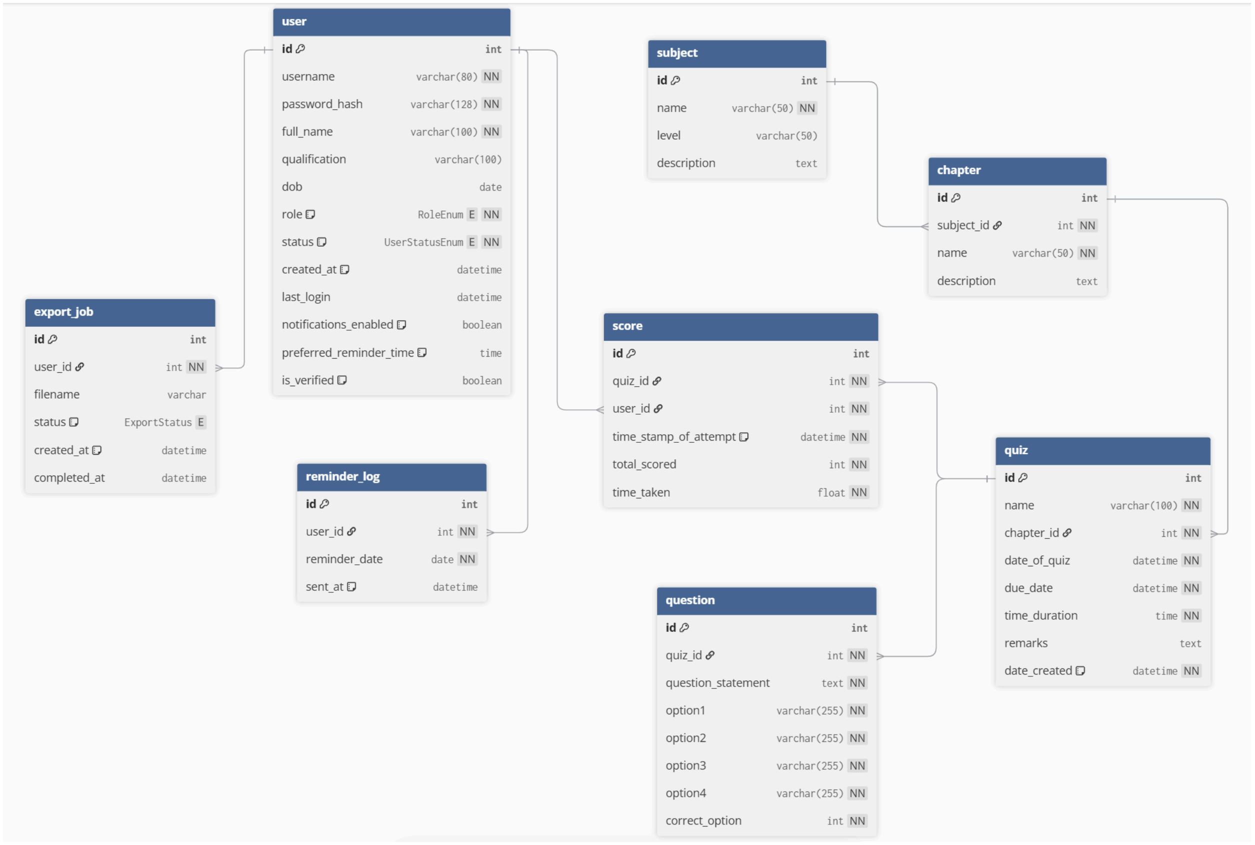
- **Chart.js** – Visualizes quiz performance and analytics in the frontend.

DATABASE SCHEMA DESIGN

The application uses **Flask-SQLAlchemy** as the ORM with **SQLite** as the database.

It features a **relational schema** with **eight** core models to ensure structured, consistent, and efficient data storage. Foreign keys connect subjects, chapters, quizzes, and questions, enabling seamless data retrieval and minimizing redundancy.

The following ER model represents the database schema design.



- **User:** Stores user details including roles, status, and preferences.
- **ReminderLog**: Tracks reminders sent to users by date.
- **Subject:** Represents different subjects.
- **Chapter:** Represents chapters under specific subjects.
- **Quiz:** Defines quizzes within a chapter and stores quiz details.
- **Question:** Contains multiple-choice questions for a quiz, where only one option is correct.
- **Score:** Records users' quiz results and time taken.
- **ExportJob**: Tracks data export tasks and their status.

FEATURES

User Features:

- **Secure Registration & Login** – JWT-based authentication with email verification.
- **Timed MCQ Quizzes** – Attempt scheduled quizzes with a countdown timer.
- **Score Tracking and CSV Export** – View past attempts and download data in CSV format.

- **Profile Management** – Edit personal details, change password, or delete account.
- **Reminder Preferences** – Toggle daily reminders and set preferred reminder time.
- **Analytics & Insights** – Visual charts showing individual performance.

Admin features:

- **Admin Dashboard** – Manage all content and users from a centralized interface.
- **Subject, Chapter, Quiz, and Question Management** – Create, edit, or delete content easily.
- **User Management** – Activate, suspend, or view user status and profiles.
- **Quiz Scheduling** – Set quiz start dates, due dates, and durations.
- **Analytics and Performance** – Monitor user scores and engagement through analytics.
- **Search Functionality** – Quickly find relevant quizzes/subjects/users.

System Features:

- **Role-Based Access Control (RBAC)** – Separate permissions for users and admin roles.
- **Email Verification** – Required upon registration to activate user accounts.
- **Scheduled Tasks** –
  - **Daily Email Reminders** for pending quizzes.
  - **Monthly PDF Reports** sent to users.
- **Performance & Caching** – Optimized API with caching and cache expiry for faster response.
- **Responsive UI** – Fully responsive and user-friendly design using Bootstrap.

API DESIGN

The application follows a **RESTful API architecture** using **Flask-RESTful** on the backend. It organizes endpoints into logical groups for **authentication**, **user operations**, and **admin functionality**, each prefixed accordingly (e.g., `/auth/`, `/user/`, `/admin/`).

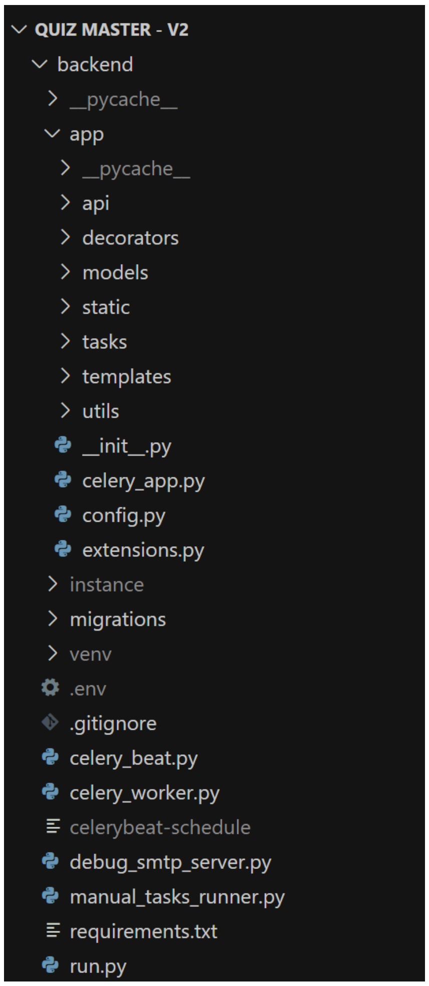
All endpoints return **JSON responses** and use standard HTTP methods (`GET`, `POST`, `PUT`, `DELETE`). The frontend communicates with the backend using **Axios** for asynchronous HTTP requests.

All protected routes require a **JWT token** in the request header:

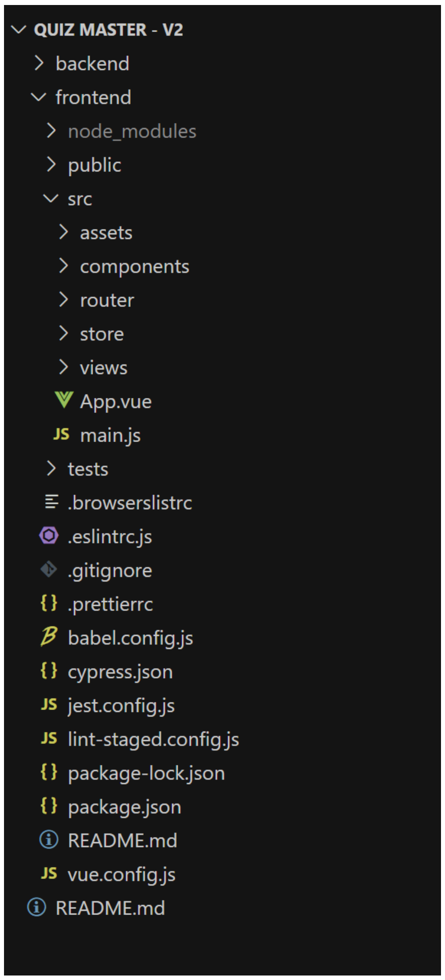
```
Authorization: Bearer <access_token>
```

Axios interceptors are used to attach JWT tokens to every request. Loading states and toast messages provide user feedback on API actions.

ARCHITECTURE



```
QUIZ MASTER - V2
  backend
    __pycache__
    app
      __pycache__
      api
      decorators
      models
      static
      tasks
      templates
      utils
      __init__.py
      celery_app.py
      config.py
      extensions.py
    instance
    migrations
    venv
    .env
    .gitignore
    celery_beat.py
    celery_worker.py
    celerybeat-schedule
    debug_smtp_server.py
    manual_tasks_runner.py
    requirements.txt
    run.py
```



```
QUIZ MASTER - V2
  backend
  frontend
    node_modules
    public
    src
      assets
      components
      router
      store
      views
      App.vue
      main.js
    tests
    .browserslistrc
    .eslintrc.js
    .gitignore
    .prettierrc
    babel.config.js
    cypress.json
    jest.config.js
    lint-staged.config.js
    package-lock.json
    package.json
    README.md
    vue.config.js
  README.md
```

VIDEO

https://drive.google.com/file/d/1GT-CYQQc-ieWXfikW5mJkEI2dXuCFYe2/view?usp=drive_link