Quiz Master - Project Report

Author:
Ishant Kumar

23F2001685

23f2001685@ds.study.iitm.ac.in

I am a student pursuing a degree in Data Science & Applications. I am interested in full stack development and AI, and I'm exploring real-world project architecture through hands-on experience.

Description

This project implements a Quiz Management System that supports both admin and user roles. Admins can create quizzes and manage users, while users can take quizzes and view their performance.
AI/LLM Used: 10% - AI was used only for generating the project report and API document YAML file(2%) and for bootstrap/css styling(8%).

Technologies Used

- **Backend**: Flask, Flask-Restful, Flask-Migrate, Flask-Caching, Flask-Security-Too, SQLAlchemy
- **Frontend**: Vue.js 3, Vite, Vue Router, Vuex
- **Database**: SQLite
- **Other Tools**: Vite (build tool), Python-dotenv, Flask-CORS, Celery with beat scheduler, Redis

Flask was chosen for its lightweight API development, Celery for background tasks like quiz scheduling or reports, and Vue for building a reactive SPA.

DB Schema Design

The schema includes the following main tables:

- **users**: id (PK), email (unique), password, full_name, qualification, dob, active (boolean), fs_uniquifier (unique)
- **role**: id (PK), name (unique), description
- **roles_users**: user_id (FK to users), role_id (FK to role)

- **subjects**: id (PK), name (unique), description, created_at (datetime)
- **chapters**: id (PK), subject_id (FK), name, description, created_at (datetime)
- **quizzes**: id (PK), chapter_id (FK), date_of_quiz (datetime), time_duration (int), is_active (boolean), title, remarks
- **questions**: id (PK), quiz_id (FK), question_statement, option1-4, correct_option (int)
- **quiz_attempts**: id (PK), user_id (FK), quiz_id (FK), timestamp (datetime), total_score, max_score, percentage
- **alembic_version**: version_num (PK)

API Design

RESTful APIs are implemented using Flask-Restful. Endpoints are provided for:

- User registration, login
- Quiz creation, question addition (admin only)
- Quiz listing, submission (user only)
- Score retrieval and summary

The full OpenAPI YAML specification is provided separately.

Architecture and Features

Backend

- Entry Point: backend/app.py initializes and runs the Flask application.
- Organized under backend/application/:
  - Data Layer: data/models.py defines SQLAlchemy models.
  - API Endpoints: resources/ contains Flask-RESTful resource definitions.
  - Standard Routes: routes.py defines basic web routes.
- Asynchronous Tasks: tasks.py defines Celery tasks, configured via celery_config.py.

Frontend

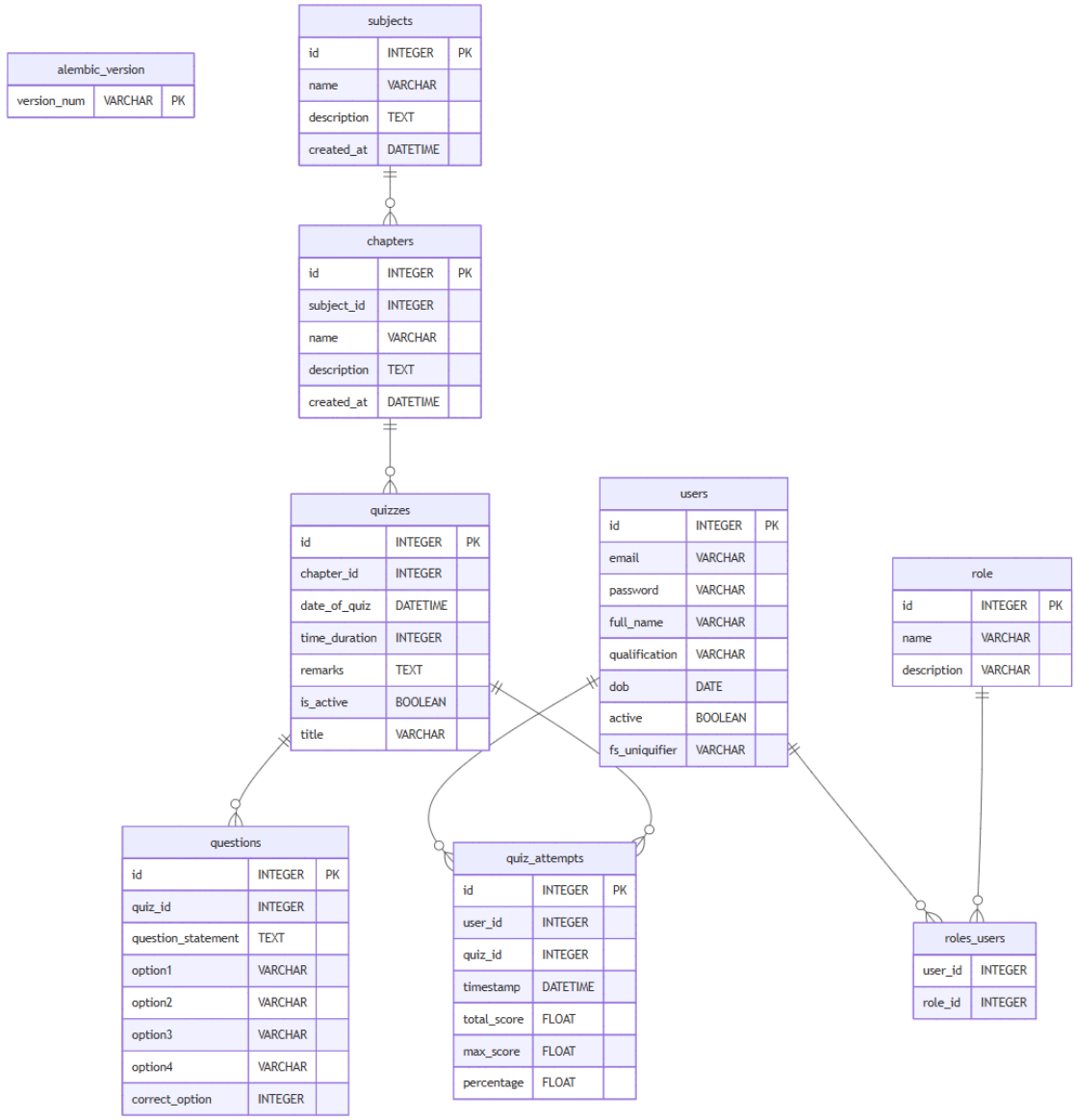
- Vue 3 app under frontend/
  - Views: src/views/ (user/admin/auth split)
  - Routing: src/router/index.js
  - State: src/store/index.js

Features Implemented

- User authentication (JWT-based)
- Admin dashboard: Add/manage quizzes and users

- User dashboard: Take quizzes, view scores
- Responsive design
- Toast notifications, modal components, and loading spinner
- Celery scheduling (for future extensions like quiz reminders)

ER Diagram



Video

[Video Link Placeholder - Add Google Drive Link Here]