QuizMaster – MAD 2 Project Report

Author

Name: Deon Levon Dmello
Roll Number: 21f3002473
Email: 21f3002473@ds.study.iitm.ac.in

Computer Science Engineer now working at a defence startup with passion for web development, AI and Deep Learning. Experienced in Python, Flask, full-stack web development and developing AI systems with focus on creating efficient and user-friendly applications.

Description

QuizMaster is an online exam preparation platform that facilitates secure quiz management for multiple subjects. It offers separate interfaces for students and administrators, real-time analytics, and a scalable backend with background task support and role-based access control.

Technologies Used

Frontend

- Vue.js 3 (Composition API)
- Vue Router 4
- Vuex 4
- Bootstrap 5
- Axios
- Chart.js

Purpose:
Used to build a responsive and modular single-page application (SPA) with real-time interactivity, routing, state management, and visualization of quiz analytics.

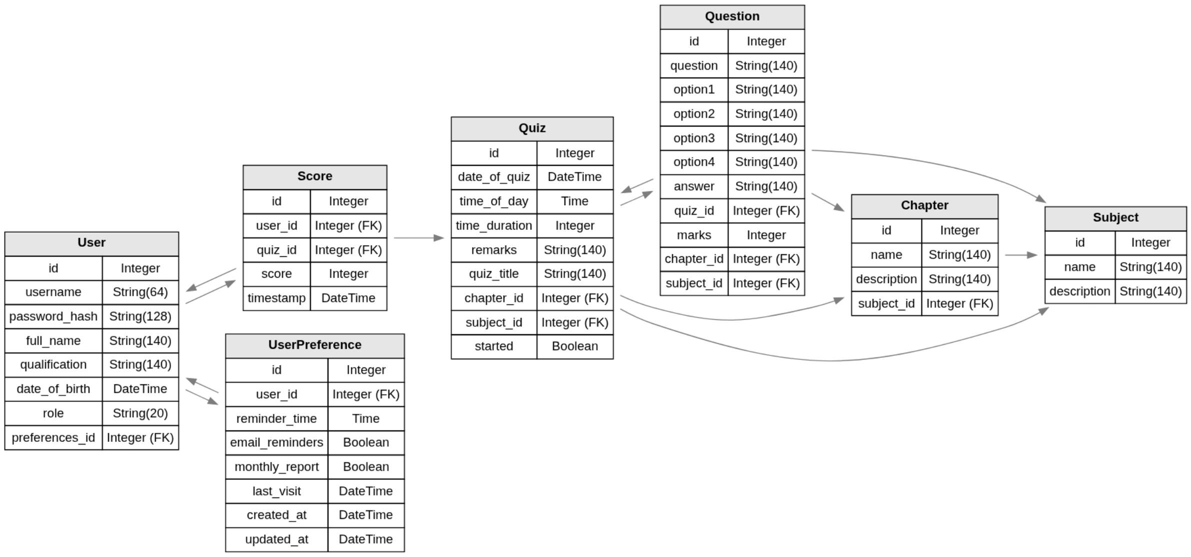
Backend

- Flask
- Flask-RESTful
- Flask-JWT-Extended
- SQLAlchemy ORM
- SQLite
- Celery
- Redis
- Flask-Mail (with MailHog)
- Werkzeug

Purpose:

- Flask and its extensions provide a secure and lightweight REST API framework.
- SQLAlchemy enables ORM-based database interactions.
- JWT for secure authentication.
- Celery + Redis for running scheduled and asynchronous tasks (e.g., score calculation, email sending).
- MailHog to test email functionality in dev without spamming real users.

DB Schema Design



User	
id	Integer
username	String(64)
password_hash	String(128)
full_name	String(140)
qualification	String(140)
date_of_birth	DateTime
role	String(20)
preferences_id	Integer (FK)

Score	
id	Integer
user_id	Integer (FK)
quiz_id	Integer (FK)
score	Integer
timestamp	DateTime

UserPreference	
id	Integer
user_id	Integer (FK)
reminder_time	Time
email_reminders	Boolean
monthly_report	Boolean
last_visit	DateTime
created_at	DateTime
updated_at	DateTime

Quiz	
id	Integer
date_of_quiz	DateTime
time_of_day	Time
time_duration	Integer
remarks	String(140)
quiz_title	String(140)
chapter_id	Integer (FK)
subject_id	Integer (FK)
started	Boolean

Question	
id	Integer
question	String(140)
option1	String(140)
option2	String(140)
option3	String(140)
option4	String(140)
answer	String(140)
quiz_id	Integer (FK)
marks	Integer
chapter_id	Integer (FK)
subject_id	Integer (FK)

Chapter	
id	Integer
name	String(140)
description	String(140)
subject_id	Integer (FK)

Subject	
id	Integer
name	String(140)
description	String(140)

API Design

Authentication

- POST /api/login: Login via credentials
- POST /api/register: User registration
- POST /api/logout: Invalidate session
- GET /api/check_token_valid: Token validation

Admin APIs

- GET/POST /api/subjects: Manage subjects
- GET/POST /api/quizzes: CRUD for quizzes
- GET/POST /api/questions: Manage quiz questions
- GET /api/admin/users: View user list

User APIs

- GET /api/dashboard: Personalized student dashboard
- GET /api/fetchQuestions?quiz_id=: Fetch quiz questions
- POST /api/takeResponse: Submit quiz answers
- GET /api/scoreboard: View user scores

OpenAPI YAML schema submitted separately.

Architecture and Features

Project Organization

- **Backend** (/backend)
    - app/models.py: SQLAlchemy ORM models
    - app/routes/: API route definitions and role-based logic
    - app/controllers/: Business logic (admin/user separated)
    - app/auth/: JWT authentication and middleware
    - app/tasks.py: Celery background tasks
    - app/utils/: Helper functions (e.g., email sending, scoring)
- **Frontend** (/frontend)
    - views/: Components for user/admin interfaces
    - store/: Vuex modules for state management
    - router.js: Vue Router configuration
    - api/: Axios wrappers for API communication

Features Implemented

Default Features

- **Role-based Access**: Admin vs. student dashboards
- **Authentication**: JWT login/logout with secure password hashing
- **Quiz Management**: Create, schedule, and activate/deactivate quizzes
- **Question Management**: Add/edit/delete quiz questions
- **Participation**: Students can attempt active quizzes and view scores
- **Score Calculation**: Handled via Celery and stored persistently
- **Dashboard Analytics**: Scoreboard, usage stats via Chart.js
- **Flash Messaging**: Feedback after login, quiz attempts, etc.

Additional Features

- Real-time countdown for quizzes
- Session timeout and token refresh handling
- API-level error logging and validation
- Admin revenue/user participation analytics (optional feature)
- Background score computation and email notification (via Celery)

Video Link:

MAD2VideoPresentation.mp4