Whiz.it

(Modern Application Development – II)

Student details

Name – Atharv Khare
Roll no. – 23f2004201
Email – 23f2004201@ds.study.iitm.ac.in
Mindful learner striving for personal and academic growth with integrity.

Project Overview

Whiz.it is an online quiz application where administrators wield precise control, defining subjects, chapters, and quizzes with detail. Users engage with timed, multiple-choice assessments, receiving immediate, actionable feedback. The platform's core strength lies in its efficient architecture, seamlessly managing asynchronous tasks like scheduled notifications and data exports, ensuring a reliable and streamlined learning environment.

My Approach

1. Database Schema Design:

Designed the database schema, which serves as the foundation for storing and managing the application's data, and its relationships between tables for clarity.

2. User Flow

To visualize the user journey and interaction with the application, I created detailed layouts, navigation, and UI elements for each screen.

3. Implementation:

I defined Flask APIs to handle different user actions and requests.
These apis were organized based on user roles. Then I implemented token based authentication and authorization, followed by CRUD operations for various entities. Finally features, such as quiz view, search functionality, profile, and notifications, were integrated.

Features Implemented

1. Multi-User System:

Admin: Full control.
Users: Register, login, take quizzes.

2. Structured Content Management:

Subject > Chapter > Quiz hierarchy.
Easy content CRUD (Create, Read, Update, Delete).

3. Interactive Quizzes:

MCQ format, instant feedback.Timers, scheduled access.

4. Score Management and Analytics

Detailed score tracking.Quiz summaries, performance insights.
Leaderboard, Explore Feature.

Technologies used

Backend:

Flask: Python micro web framework
SQLAlchemy: Object-Relational Mapper (ORM)
SQLite: Database management system
Celery: A distributed task queue that enables you to run tasks asynchronously in the background.
flask_mailman: A Flask extension that simplifies sending emails
flask_caching: A Flask extension that adds caching capabilities

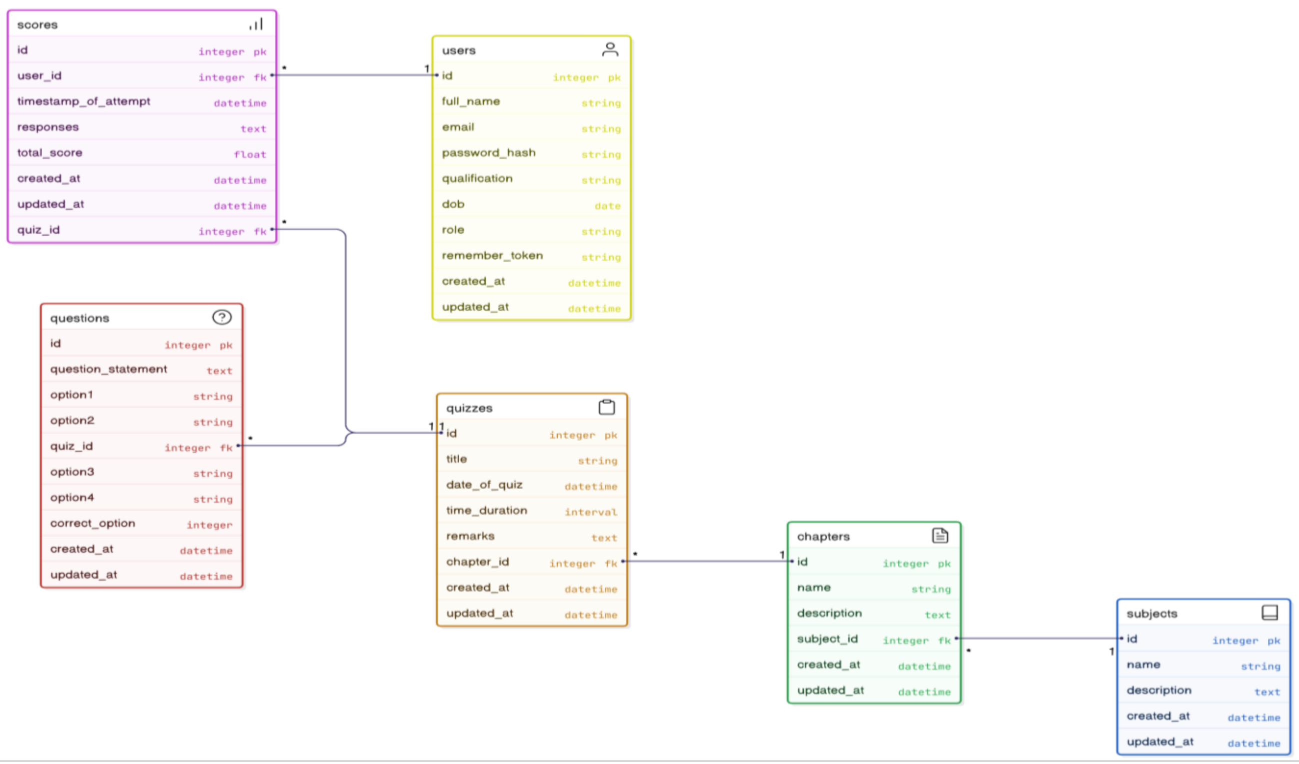
Frontend:

HTML: Structure of web pages
Vue.JS: progressive JavaScript framework used for building user interfaces. 1 It's known for its simplicity and flexibility
Vuex: A state management library for Vue.js applications. It provides a centralized store for all the components in an application
Vue Router: The official router for Vue.js
CSS: Styling of web pages
Bootstrap: CSS framework (used for accordion element)
Chart.js: JavaScript library for creating charts and graphs

Additional Libraries and Tools:

Flask-Session: Session management for storing user data across requests
Datetime: Date and time handling
Werkzeug: Secure password hashing and authentication
Pytz: Library for working with time zones (timezone)
Sqlalchemy.func: Provides functions for performing database
Sqlalchemy.or_: Function for building OR clauses in SQL queries.
flask_jwt_extended: A Flask extension that provides JWT authentication
io: The built-in Python module for working with various types of input and output
dateutil: A powerful Python library that extends the standard datetime module with more advanced date and time functionality

E-R Diagram ↗



API ↗

The Whiz.it API provides various functionalities for user authentication, quiz management, user administration, and notifications. Below is a brief description of the available API endpoints.

Authentication & User Management
•POST /signup – Registers a new user.
•POST /login – Authenticates a user and provides a JWT token.
•POST /logout – Logs out the user by revoking the JWT token (requires authentication).
•GET /api/user/{username} – Retrieves profile details of a user by username.
•GET /api/user/{id} – Fetches user data by ID (Admin only).
•GET /api/block_user/{id} – Blocks a user (Admin only).
•GET /api/unblock_user/{id} – Unblocks a user (Admin only).

Quiz & Score Management
•POST /api/quizzes/{quiz_id}/submit – Submits a quiz with user-provided answers.
•GET /api/quizzes/{score_id}/user/{user}/history – Retrieves quiz history for a specific user and score.
•GET /api/users/scores – Fetches the current user's quiz scores.
•GET /api/allusers/scores – Retrieves quiz scores for all users.
•GET /api/leaderboard – Fetches the leaderboard based on user scores.

Subjects Management (Admin Only)
•GET /api/subjects – Retrieves a list of all subjects.
•POST /api/subjects – Creates a new subject.
•PUT /api/subjects/{id} – Updates details of an existing subject.
•DELETE /api/subjects/{id} – Deletes a subject.

Notifications & Exports
•POST /api/send-notification – Queues an email notification.
•POST /api/export-quiz-history – Initiates a CSV export of quiz history.
•GET /api/export_csv – Retrieves the generated CSV file.
•POST /api/export_csv – Requests a new CSV export for user data.

Admin Dashboard
•GET /api/admin_dash – Retrieves statistics for the admin dashboard.

This API is secured using JWT-based authentication for protected endpoints. Admin-specific actions such as user blocking, quiz management, and subject creation require authentication.

Project structure↗

Demonstration Video

https://drive.google.com/file/d/1A0A15A65O236JLmgzdsDCOdO4dLkElOh/view?usp=sharing