Project Report

Author

Name: Jatin Nahata, **Roll no**: 23f2001705, **Email**: 23f2001705@ds.study.iitm.ac.in,
I am currently a student at diploma level in *IIT Madras BS Program*.

Installation Guide

1. To run the flask Api, create a virtual environment by using '**python -m venv venv**' in *backend folder* ***(cd backend)***
   Now, enable the created virtual environment you have, by running "**venv\Scripts\activate**" command on terminal.
   After creating a virtual environment, run command '**pip install -r requirements.txt**' and then all required packages will be installed and then run "api.py" file or run using the python command '**python api.py**' on the terminal.
   Or
   Simply, run command '**pip install -r requirements.txt**' in *backend folder* **(*cd backend*)** and then all required packages will be installed. and then run "api.py" file or run using the python command '**python api.py**' on the terminal.

2. Now, to create '**node_module**' folder run '**npm install**' in *frontend folder* ***(cd frontend)***. It will install all required package.

3. To run, the frontend, use '**npm run serve**' on terminal in *frontend folder* ***(cd frontend)***.

4. To install redis server use '**sudo apt install redis-server**' in wsl terminal and then start redis by using '**redis-server**' in wsl terminal.

   [To install wsl in windows: How to Install Ubuntu in WSL2 in Just 3 Steps]

5. Now to run celery worker using redis, run command '**python -m celery -A app.celery worker --loglevel=info --pool=solo**' on terminal in *backend folder* ***(cd backend)***.

6. Then, to run celery beat using redis, run command '**python -m celery -A app.celery beat --loglevel=info**' on terminal in *backend folder* ***(cd backend).***

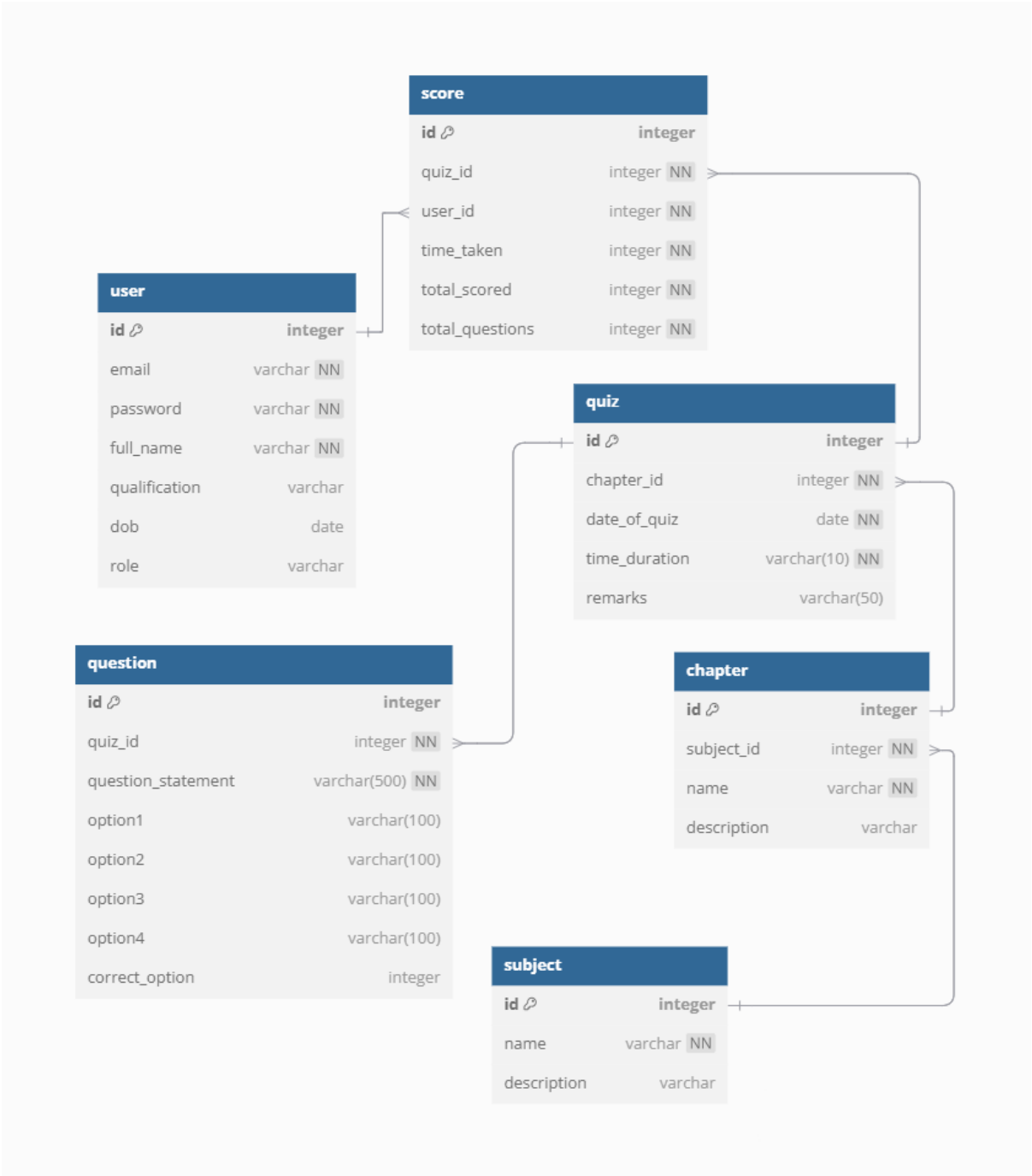
Description

StuiQ is a multi-user web application designed to help students prepare for various courses through structured quizzes. The platform supports two types of users: Administrator and Users (students). Administrator can manage course content, chapter, quizzes, and questions, while students engage in self-paced learning and assessments.

Technologies used

I. **Flask**: Handling HTTP requests, defining routes.
II. **Flask-SQLAlchemy**: ORM to interact with the SQLite database.
III. **Flask-Cors**: Enabling CORS (Cross-Origin Resource Sharing) between frontend and backend.
IV. **Flask-JWT-Extended**: Managing authentication and protecting routes using JSON Web Tokens.
V. **Flask-Caching + Redis**: Caching expensive computations or data frequently used
VI. **Flask-Limiter + Redis**: Rate-limiting API requests
VII. **Celery + Redis**: Handling asynchronous tasks and scheduled jobs (e.g., sending quiz results, processing long tasks).
VIII. **python-dotenv**: Managing environment variables securely.
IX. **Werkzeug**: Underlying Flask library for password hashing and security.
X. **reportlab**: Generating PDF reports (e.g., monthly reports)
XI. **SQLite**: Lightweight, file-based DB suitable for small to medium-scale apps and during development.
XII. **datetime**: Useful for tracking quiz timers, task schedules, timestamps.
XIII. **npm + node_modules**: Managing frontend dependencies (like Vue.js) and also Required to build and serve the frontend UI.
XIV. **io**: Ideal for generating CSV exports of user reports via APIs without saving temporary files to disk.

DB Schema Design (ERD Diagram)



Architecture and Features

Architecture

Following structure is recommended for backend work.

A. In *model folder,* "***models.py***" containing the **database schema** related definitions is divided into separate modules.
B. The *router folder*, contains the **routers of the application server** (divided into **admin**, **auth.** and **user**).
C. The *tasks folder*, contains the **task** (like **sending daily reminders**, **sending monthly reports**, **pdf design for monthly reports (*optional*)**, **downloading user reports**).
D. The *utils folder* is containing the **mail services of the application server** (like **mailer.py**, **report_mailer.py**).
E. The "***__init__.py***" setup a **base of application server** and **celery work**.
F. The "***config.py***" contains configuration variables that are used to control the behaviour of your application.
G. The "**api.py**" module serves as the entry point to initialize and run the flask application server.
H. ***Venv*** contains the *required python libraries* used to build the application server.

Following structure is recommended for frontend work.

A. The *public folder,* contains application base folder '**index.html**' and **favicon.**
B. The *src folder,* contains **assests**, **components**, **routers**, **services**, **views** which help to design and manage application in server side.
C. The "***package.json"***, contains dependencies which helps to develop, design application.

Features

- **Login System**: **Admin** and **Users** can login themselves by using email id and password.
- **Register System**: **Users** can register themselves by using their email id.
- **Authentication**: **Admin** and **Users** are authenticated using **JWT token** and **Role-Based Access Control (RBAC)**, with appropriate redirection based on role.
- **Admin Dashboard**: **Admin** can create, edit, and delete course, chapter and can also see quizzes inside a chapter.
- **Quiz Management**: **Admin** can create, edit, and delete a quiz, question.
- **Dark Mode:** Enhance visual comfort.
- **User Dashboard**: **Users** can attempt available quizzes.

- **Quiz Timer**: Quizzes have a timer for time-limited attempt.
- **Score Tracking**: Scores are automatically recorded after quiz completion.
- **Profile**: **Users** can see their personal details, badge they earned, and can download their report.
- **Search Functionality**: **Admin** can search users, courses, chapters, quizzes and questions whereas **Users** can search quizzes by entering the name of the chapter or course that quiz belongs to.
- **Summary Charts**: Admin and users can view summary charts for performance and activity.
- **Reminders**: Users will receive daily reminders at 18:30, if a new quiz is available and hasn't attempted by user.
- **Monthly Reports**: Users will receive a monthly report on 1st day of each month, which gives details about quizzes attempted by user, their rank, etc.

About APIs Endpoints:

/api/auth: Handles user authentication-related operations such as registration, login, and token creation and verification.

/api/admin: Handles creating, editing, deleting a course, chapter, quizzes and questions, also manage users, and can see summary charts of admin activity.

/api/user: Handles attempting quizzes, viewing scores and summary charts and user related functionality.

Video

For watching presentation: click here