Project Report for Final Project - Modern Application Development II (Quiz Pilot)

Author

Samyabrata Roy
22f2001443
22f2001443@ds.study.iitm.ac.in

I am pursuing a BS in Data Science from IIT Madras along with BSc in Statistics from Sister Nivedita University. Currently, I am working as a student intern at IDEAS - TIH, ISI Kolkata. I have a strong interest in Non-Parametric Statistical Inference and Machine Learning and truly enjoyed the journey of developing the WebApp for my projects.

Description

In this project, I was tasked with developing a multi-user quiz application using Flask, Vue, SQLite and Bootstrap. The application features an admin who manages users, subjects, chapters, quizzes and its questions while users can attempt quizzes within a situated time set by the admin and view or download their scores along with daily notification and monthly report sending features . It supports CRUD operations, user authentication, quiz management, and data visualization.

Technologies used

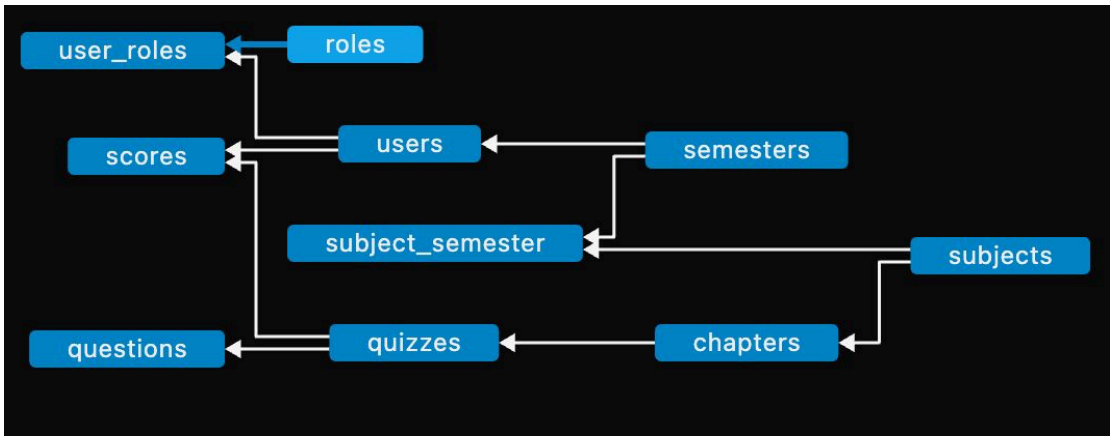
For Backend:
Flask : Creating and handling the APIs
SQLite : a lightweight, self-contained, serverless relational database management system (RDBMS)
Redis : Remote Dictionary Server, an open-source, in-memory key-value store that is here used as a:

- Database
- Cache
- Message broker
- Task queue backend

Celery : an asynchronous task queue/job queue based on distributed message passing
Jinja2 : Used for rendering HTML templates.

For Frontend:
Vue : For creating UI
Pinia : For State management
ChartJS: For rendering charts

DB Schema Design



API Design

Name	Endpoint	Description	Methods
Login	/api/auth/login	Log in an existing user	GET, POST
Register	/api/auth/register	Register a new user	GET, POST
Logout	/api/auth/logout	Log out the current user	POST
AdminMatrics	/api/admin/metrics	View platform-wide admin metrics	GET
AdminUserManagement	/api/admin/users	Manage user	GET, PUT, DELETE
AdminSubjectManagement	/api/admin/subjects	Manage academic subject	GET, POST, PUT, DELETE
AdminChapterManagement	/api/admin/chapters/<subject_id>	Manage chapters under a given subject	GET, POST, PUT, DELETE
AdminQuizManagement	/api/admin/quizzes	Manage quizzes	GET, POST, PUT, DELETE
AdminQuestionManagement	/api/admin/questions/<quiz_id>	Manage questions for a given quiz	GET, POST, PUT, DELETE
GetStudentDistribution	/api/admin/analytics/student-distribution	View distribution of students	GET
GetSubjectAttempts	/api/admin/analytics/subject-attempts	View number of attempts per subject	GET
GetQuizPerformance	/api/admin/analytics/quiz-performance	Get quiz performance summary	GET
GetLeaderboard	/api/admin/analytics/leaderboard	Get quiz performance summary	GET
GetStudentStatusCount	/api/admin/analytics/student-status	View counts of students by status	GET
DownloadProgress	/api/admin/analytics/download	Download overall progress reports	GET
UserProfile	/api/user/profile/<user_id>	Get user’s profile info	GET
UserDashboard	'/api/user/dashboard/<user_id>	Get the list of all available quizzes and subjects enrolled	GET
QuizStart	/api/user/quiz/start/<quiz_id>	Start a quiz attempt	GET, POST
getQuestion	/api/user/quiz/questions/<score_entry_id>	Get questions for a given quiz session	GET, POST
QuizResults	/api/user/quiz/results/<string:quiz_session_id>	Get quiz result for a specific session	GET
UserDownloadProgress	/api/user/download/progress/<user_id>	Download user’s quiz progress data	GET

Architecture and Features

```
Backend/
├── requirements.txt                              # List of python package on needed for the app to run
(dependencies)
├── .env                                          # Environment variables
├── .env.example                                  # Template for environment variables
├── .gitignore                                    # Files/folders to be ignored by Git
├── README.md                                     # Project documentation (this file)
├── app.py                  # Entry point of the Flask app (Main app); .env, Sem seed, app, cache clr, db,
CORS
├── data/
│   ├── quiz-db-v2.sqlite                         # SQLite Database
│   ├── dump.rdb                                  # Redis Dumb db, relocated here by redis-config.conf
│   ├── celerybeat-schedule.db                    # Celery beat schedule state;
│   └── backup/  # This is the backup of the data stored in DB; incase I have to delete the .sqlite file
│       ├── backup-creation.sql
│       └── backup-data.sql
├── utils/                                        # All the configuration files are being stored here
│   ├── redis-config.conf                         # Redis server configuration
│   ├── config.py                                 # General app configuration; decides db file's location
│   └── CeleryConfig/
│       ├── __init__.py                           # Celery-specific configuration
│       ├── celery-beat.py                        # A programmatic way to launch the Celery Beat scheduler.
│       ├── celery-worker.py  # Celery worker initialization; this uses CreateApp()function
│       └── mailer.py                             # Mail sending utility via Celery;
├── controller/
│   ├── __init__.py           # imports and exports send_daily_quiz_reminders, send_monthly_report_to_users
│   ├── extensions.py             # Enables security, redis_client, cache, limiter and celery
│   ├── models/                   # All db and its models are defined here; Using SQLAlchemy, the ORM
│   │   ├── __init__.py
│   │   ├── chapter_model.py
│   │   ├── enums.py
│   │   ├── question_model.py
│   │   ├── quiz_model.py
│   │   ├── score_model.py
│   │   ├── semester_model.py
│   │   ├── subject_model.py
│   │   └── user_model.py
│   ├── routes/
│   │   ├── __init__.py                           # Registers all the APIs
│   │   ├── user.py                               # Defines the API classes used by the users
│   │   ├── auth.py                               # Defines the API classes used for login, logout, register
│   │   └── admin/                                # Defines the API classes used by the admin
|   |       ├── __init__.py
|   |       └── analytics.py  # Defines the AI classes required for ChartJS in the front to render the
charts
│   ├── tasks/
│   │   ├── __init__.py                           # Importing and exporting the tasks
|   |   ├── README.md                             # Listed things could be added here in the tasks
│   │   ├── send_emails/
│   │   |   ├── __init__.py                       # Importing and exporting the tasks
│   │   |   ├── send_reports.py  # Created a celery task for sending monthly reports to all the active
users
│   │   |   ├── send_reminders.py # created a celery task to send daily noti of pending quizzes to users
│   │   |   └── html-templates/                   # Contains all the HTML templates to be used in the mail
│   │   |       ├── reminder.html
│   │   |       └── report.html
```

```
frontend/
├── .gitignore                                  # Git ignored files and folders
├── vite.config.js                              # Vite configuration file ; here the port, poxy and alias are
defined
├── README.md                                   # Frontend readme (this file)
├── package.json                                # Project metadata and dependencies
├── package-lock.json                           # Dependency tree lock (auto-generated)
├── index.html                                  # Main HTML entry point
├── public/                                     # Static public assets
│   └── vite.svg                                # Vite logo; used in <title> tag of index.html
├── src/                                        # Main source code
│   ├── App.vue                                 # Root Vue component
│   ├── main.js                                 # JS entry point
│   ├── style.css                               # Global styles
│   ├── api/
│   │   └── axios.js                  # Axios instance setup; axiosPublic and axiosPrivate is defined here
│   ├── assets/                                 # Static resources (images/icons)
│   │   ├── logo.png
│   │   └── vue.svg
│   ├── constants/
│   │   └── appInfo.js                          # Application metadata and constants; APPNAME is defined here
│   ├── globalComponents/                       # Reusable components
│   │   ├── Admin/                              # Nothing is here
│   │   ├── User/                               # Nothing is in here
│   │   ├── Footers.vue                         # Footer component
│   │   ├── Header.vue                          # Header component
│   │   └── UseAuth.js                          # Contains logout function
│   ├── routes/
│   │   ├── admin.js                            # Routes only accessible to Admin
│   │   ├── user.js                             # Routes only accessible to User
│   │   └── index.js                            # Routes useable for everyone; No authentication needed
│   ├── store/
│   │   └── authStore.js                        # Pinia store; Auth-related state management
│   ├── views/                                  # Page views
│   │   ├── HomeView/
│   │   │   └── HomeView.vue                    # This is the page we'll see when running app at the beginning
│   │   ├── AuthView/
│   │   │   ├── AuthFormView.vue                # This is the PAge being used fo both Login and Register
│   │   │   └── Components/
│   │   │       └── DynamicForm.vue             # Renders the form dynamically
│   │   ├── admin/
│   │   │   ├── DashboardView.vue               # This is the page used for Admin Dashboard
│   │   │   ├── UserView.vue                    # This is the page used to view User Card under the Dashboard
│   │   │   ├── SubjectView.vue                 # This is the page used to view Subject Card under the
Dashboard
│   │   │   ├── ChapterView.vue                 # This is the page used to view Chapter paga under each
Subjects
│   │   │   ├── QuizView.vue                    # This is the page used to view Quiz Card under the Dashboard
│   │   │   ├── QuestionView.vue                # This is the page used to view Questions under each quizzes
│   │   │   ├── AnalyticsView.vue             # This is the page used to view Analytics Card under the
Dashboard
│   │   │   └── Components/
│   │   │       ├── AddQuestionView.vue     # This is to handle the modal required for Adding/Editing
Questions
│   │   │       ├── AddSubjectView.vue      # This is to handle the modal required for Adding/Editing
Subjects
│   │   │       ├── ChapterModalView.vue    # This is to handle the modal required for Adding/Editing
Chapters
│   │   │       └── QuizModalView.vue       # This is to handle the modal required for Adding/Editing Quizzes
│   │   └── user/
│   │       ├── ProfileView.vue               # This is the page used to view User's profile
│   │       ├── DashboardView/ # This page is used as User Dashboard;Shows available quizzes│   │   │   │
└── DashboardView.vue
│   │   │   └── QuizView/
```

```
|   |   |       ├── QuizLandingView.vue     # Shows quizzes info before starting
|   |   |       ├── QuizQuestionView.vue    # Shows the questions; Timer is working in this page
|   |   |       ├── QuizResultView.vue      # After submission this page shows the Result
|   |   |       └── UseStartQuiz.js         # Prevent duplicate submission and unauthorized access
```

VIDEO

Link: https://drive.google.com/file/d/1h2BPRwqGloJK-ouZrI7VdxC3k1vXNB42/view?usp=drive_link