sprint1-8

Part I. Architectural Design

Backend Architecture Overview

The backend structure is organized as follows:

```
.
├── apps
│   ├── community
│   ├── events
│   ├── infodisplay
│   ├── recommendations
│   └── reminders
├── core
├── db
├── main.py
└── utils
```

Detailed Explanation

1. `apps` **Directory**:
   - Each subdirectory under `apps` represents a separate application module within the project. This modular structure helps in maintaining the scalability and manageability of the project.
   - **Subdirectories**:
     - `community`: Handles all functionalities related to user community interactions like forums or user groups.
     - `events`: Contains code and logic for event management and bookings.
     - `infodisplay`: Manages the display of information to the user.
     - `recommendations`: Implements algorithms for suggesting events or content based on user preferences.
     - `reminders`: Manages sending notifications and reminders to users.
2. `core` **Directory**:
   - This includes essential middleware, configuration files, and utility modules that are core to the operation of the backend.
3. `db` **Directory**:
   - Manages database interactions, includes models, schema definitions, and database access utilities.
4. `utils` **Directory**:
   - Contains helper functions and utilities that support various backend functionalities but are not directly exposed as web services.
5. `main.py`:
   - The entry point for the FastAPI application. This script configures and starts the web server and includes route definitions.

Frontend Architecture Overview

The frontend structure is organized as follows:

```
.
├── App.vue
├── assets
│   └── logo.png
├── components
│   ├── CommunityPage.vue
```

```
│   ├── EventBooking.vue
│   ├── EventsPage.vue
│   ├── Footer.vue
│   ├── InfoDisplay.vue
│   ├── Navbar.vue
│   ├── Recommendations.vue
│   ├── Reminders.vue
│   └── UserProfile.vue
└── main.js
```

Detailed Explanation

1. `App.vue`:
   - Serves as the root component where all other components are imported and integrated.
2. `main.js`:
   - Entry script for the Vue application which sets up the Vue instance, mounts the app, and includes essential plugins like Vue Router.
3. `assets` **Directory**:
   - Houses static resources such as images and styling files used across the application.
4. `components` Directory

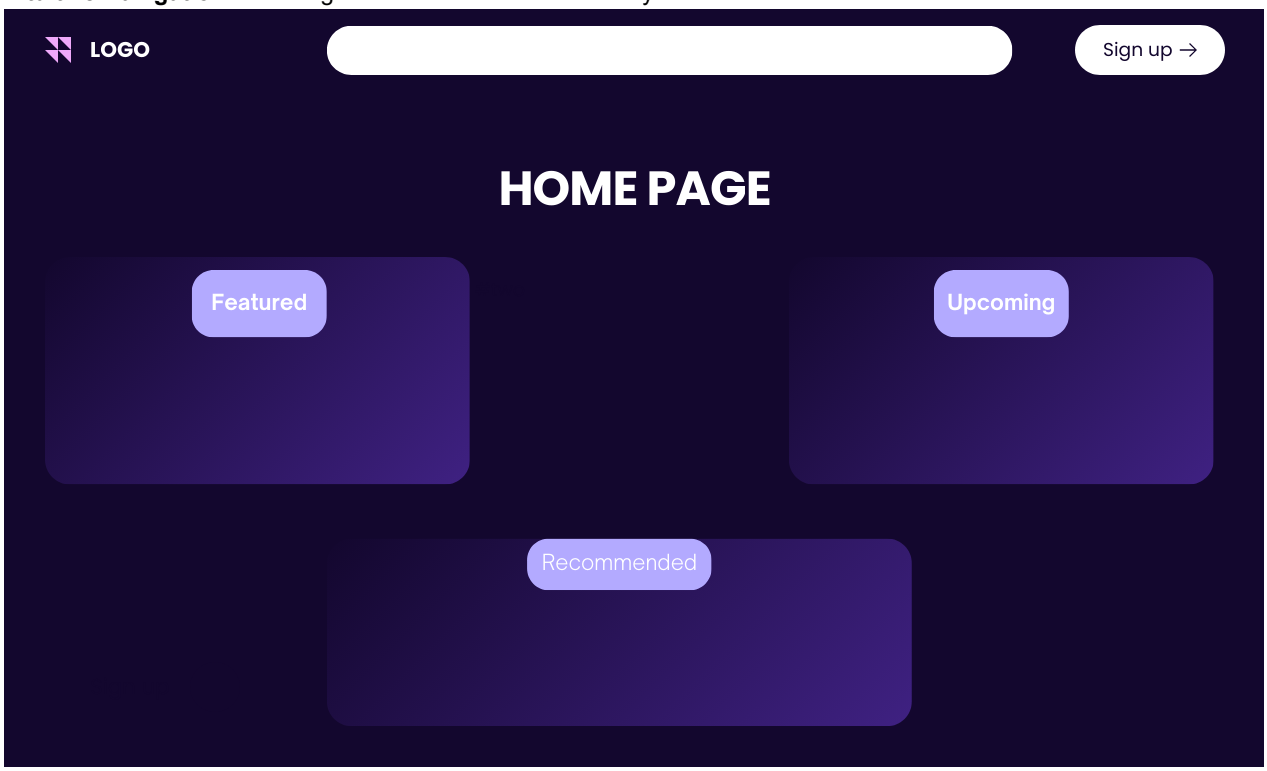
- This directory includes all the Vue component files used to build various parts of the application interface. Each component is designed to handle specific functionality within the app's ecosystem. Here are the details for each component:
- Navbar.vue: This component contains the navigation menu, which allows users to easily access different pages such as the Community, Events, and Recommendations pages. It is crucial for facilitating user movement throughout the app.
- CommunityPage.vue: This page is centered around social interaction and content sharing among users. It might include sub-components like a post list and post details, enabling users to engage with community content.
- EventsPage.vue: Displays a list of upcoming events that users can browse and choose to book. It serves as the main interface for users looking to participate in events.
- EventBooking.vue: Manages the logic for booking events. This includes form submissions for booking, choosing tickets, and other related functionalities.
- Recommendations.vue: Shows personalized event recommendations based on users' past activities and preferences. This component uses data analytics to suggest relevant events to enhance user experience.
- InfoDisplay.vue: Provides detailed information about events, such as category, date, location, and a description. This component helps users make informed decisions about attending events.
- Reminders.vue: Responsible for sending and displaying reminders about events. It helps ensure that users are aware of upcoming events they are interested in or have registered for.
- UserProfile.vue: Allows users to view and edit their personal information and settings. This personalization helps users manage their experience and preferences within the app.
- Footer.vue: Displays copyright and navigation links at the bottom of each page. It provides essential links back to the main sections of the app and legal information.
- These components together create a cohesive and functional user interface that supports the diverse needs of the app’s user base. Each component is modular and can be updated independently, making the app scalable and maintainable.

Part II. UI Design

Design Philosophy

- **Minimalist Color Scheme**: We use a simple palette primarily based on whites and grays to ensure readability and focus.
- **Generous Use of Space**: To avoid clutter and enhance user focus on content.

- **Intuitive Navigation**: Ensuring that the user interface is easy to understand and interact with.



Key Interfaces

1. **Home Page**:
   - Features primary navigation and key visuals that represent the core functionalities of the platform.
2. **Community Page**:
   - Designed to foster user interaction and content sharing.
3. **Events Booking Page**:
   - Allows users to view and book events. Simplifies the process through an intuitive layout.
4. **User Profile Page**:
   - Provides users with the ability to edit their personal information in a user-friendly format.

Screenshots for Demo

- Please replace this text with actual images or links to images that showcase the UI designs for each key interface mentioned above. These images are crucial for visualizing the implemented designs and will be displayed during the project presentations.

Part III. Retrospective and Planning

First Sprint Review

What Went Well:

- The initial UI framework has been set up successfully. This is a significant achievement as it lays the foundation for further frontend and backend integration.
- Despite the challenges, the team was able to reorganize the project plan and make considerable progress by adjusting the roadmap and strategies accordingly.

Challenges:

- The initial project plan and roadmap were not perfect. We underestimated the complexity and the time required to complete certain tasks.

- We did not fully account for the learning curve associated with our technology stack, which led to delays. There was also a lack of consideration in the choice of the technology stack.
- There was no comprehensive 'from scratch' plan for building the entire project architecture, which hindered systematic progress in the early stages.

Actions for Next Sprint:

- We will focus on completing the backend setup based on the groundwork laid by the frontend framework established in this sprint.
- Improve overall project test coverage, especially for the backend to ensure stability and functionality.
- Re-evaluate and possibly enhance the choice of technology stack to better suit our project needs and to reduce the learning curve.
- Develop a more robust project management strategy that includes a clear, detailed roadmap and contingency plans to handle unexpected challenges.

This revised approach will help streamline our efforts and ensure that we meet our project milestones more efficiently. The lessons learned from this sprint are invaluable and will guide our strategies moving forward.

Demo and Deliverables

- **Current Implementation**:

functionalities such as user login, event booking, and profile management will be shown.

- These features represent the core deliverables for this sprint.

Team Contributions

- **Member Contributions**:
  Yukun Yang: Four pages of Frontend & Report
  Honglie Li: Backend Skeleton & Report
  Zhelin Chu: Two pages of Frontend
  Hanting Li: Two pages of Frontend
  Jiaxuan Li: Two pages of Frontend
- Contributions are based on individual tasks and responsibilities, ensuring that all members are acknowledged for their specific roles.