SFU HUB

DESCRIPTION

This project aims to develop a web based application called SFU HUB, which serves as a centralized platform for SFU students to access essential resources,event information, and campus-related services. The platform aims to enhance student experience by offering a one-stop solution for navigating campus life.

HOW IT WORKS

SFU HUB aggregates information from various capus resources and presents it in a user-friendly interface. On this application, students can easily access important SFU related commodities like event calendars, dining options, transportation schedules, and other essential services.

RETROSPECTIVE

What went wrong in Iteration 1

Time management

- We encountered challenges in improving our time management since Iteration 1 due to conflicting schedules and midterm season. Although we made an effort to complete our tasks individually and keep each other updated at regular intervals, a considerable amount of work remained unfinished until the final deadline. This led to some compromises in deliverables and a few tasks left pending. Moving forward, we recognize the importance of more coordinated planning to address these issues.

Actions Taken in Iteartion 2

Responsibility Distribution
- We clarified task ownership for each component and ensured each member knew who to approach for dependencies. This minimized time wasted on waiting and allowed us to complete dependent tasks sooner.
Defined Check-in Points
- We scheduled regular check-ins throughout the iteration, ensuring everyone was on the same page regarding deadlines. This helped us identify and resolve any lagging tasks early.
Clear Communication Protocols
- We established clear communication protocols for notifying each other about any delays or challenges. This allowed us to address issues early rather than at the last minute.

Action Items for Continuous Improvement

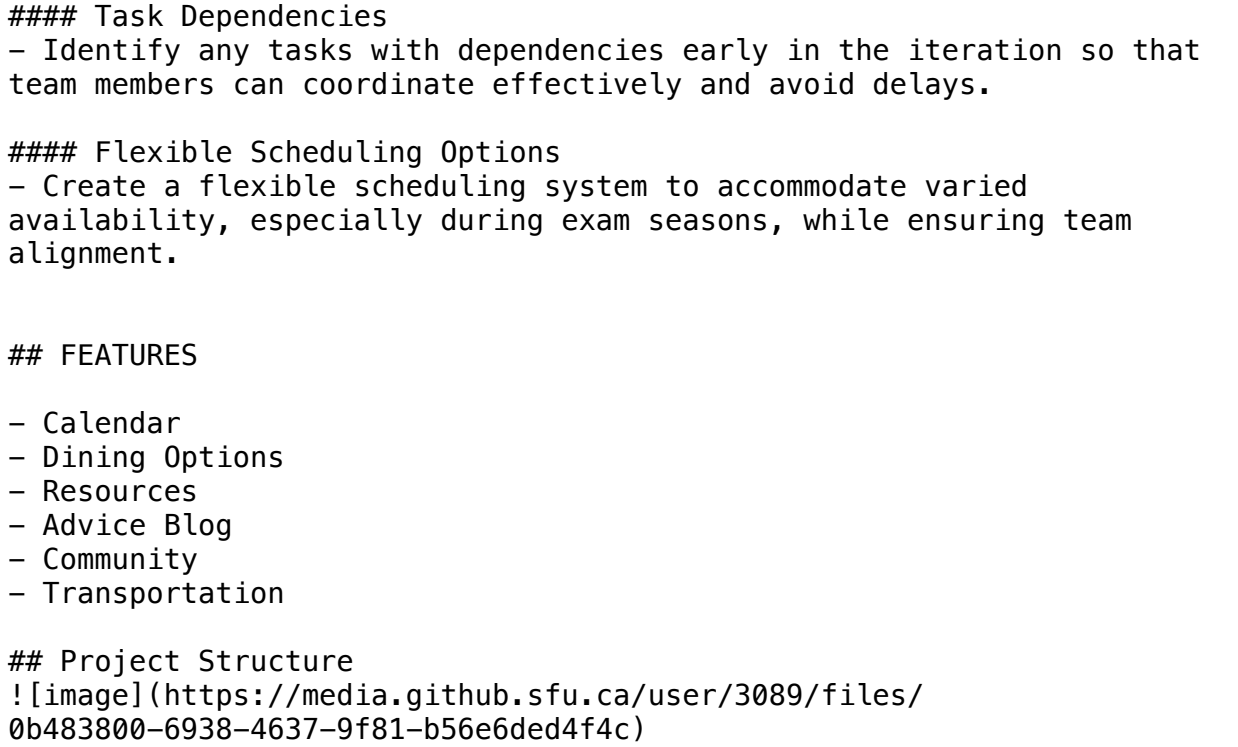
Mid-Iteration Checkpoint
- Schedule a formal mid-iteration checkpoint to evaluate our progress against goals and adjust timelines if necessary.

Task Dependencies
- Identify any tasks with dependencies early in the iteration so that team members can coordinate effectively and avoid delays.

Flexible Scheduling Options
- Create a flexible scheduling system to accommodate varied availability, especially during exam seasons, while ensuring team alignment.

FEATURES

- Calendar
- Dining Options
- Resources
- Advice Blog
- Community
- Transportation

Project Structure


REQUIREMENTS

Calendar:

- Display events: a basic monthly calendar view with events from SFU and student clubs
- Calendar can switch into monthly, weekly and daily views
- Responsive design: ensure the calendar works across devices (phone, desktop, tablet)
- Filtering and sorting events (Iteration 2)

Dining page

- List dining options: display a list of all on-campus dining locations (cafes, restaurants, food trucks)
- Basic info: each dining option should include essential details like name, hours of operation, menu link and location on campus
- Reviews: show simple ratings or reviews (can be static for iteration 1, real user reviews can be implemented later)
- Sort by location (near WMC, near university)

Resources page:

- Categories of resources, with a selector option and all on one page both
- Links to resources: each resource has a clickable link directing users to the appropriate website or contact info
- Search/filter: a simple search bar or filter that allows users to find resources by categories (health, academics, mental health, safety, etc)

Transportation page (Iteration 2):

- Show next bus times by bus stop
- Select bus stop to get bus times for with a selector
- Parking section with information on parking locations, rules (fees), etc
- Transit options between campuses

USAGE

Frontend

An instance of the frontend is available at https://sfuhub.ca.

To run the frontend locally:

1. Navigate to `apps/frontend`
2. Run `npm i`
3. Run `npm run dev` to build and run
4. Follow the steps as shown in the video below.

[![SFU_HUB_USAGE](https://github.sfu.ca/kaa80/SFU-HUB-CMPT276/blob/c095ae78ec471d07c97a98b9c16518640e7b9137/media/SFU_HUB_thumbnail.png)](https://github.sfu.ca/kaa80/SFU-HUB-CMPT276/blob/c095ae78ec471d07c97a98b9c16518640e7b9137/media/SFU%20Hub.mp4)

Backend

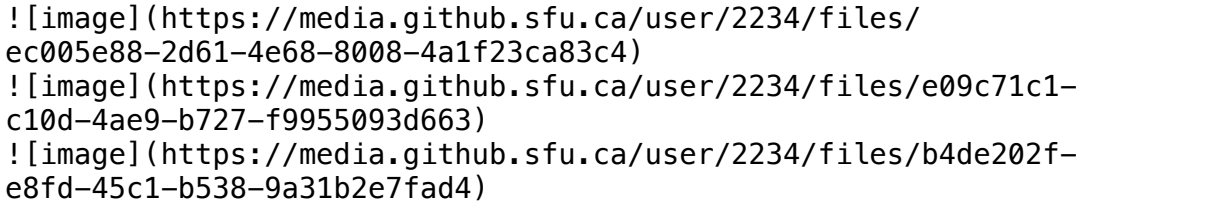
The frontend is using an instance of the backend, available at https://api.sfuhub.ca.

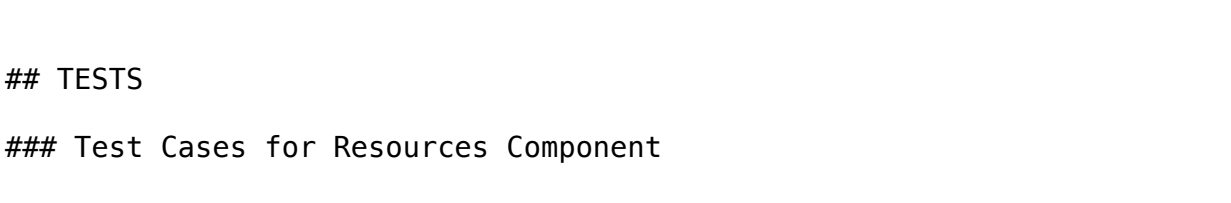
To run the backend locally,

1. Navigate to `apps/backend`
2. Run `npm i`
3. Run `npm run start` to build and run
4. Available paths can be found at https://api.sfuhub.ca/api-docs (if it isn't down, sorry!) or [in our OpenAPI file](https://github.sfu.ca/kaa80/SFU-HUB-CMPT276/blob/main/apps/backend/api/openapi.json).

The backend can also be built and ran using Docker with the provided Dockerfile.

FEATURE TRACKING





TESTS

Test Cases for Resources Component

1. Search Functionality:
   Steps: Search for "Health."
   Expected Result: Displays relevant resources.
   Actual Result: Displays relevant resources.
   Pass
2. No Results Handling:
   Steps: Search for a nonexistent Resource. I searched for "Nonexistent"
   Expected Result: Shows an error message.
   Actual Result: Shows an error message.
   Pass
3. Category Filtering:
   Steps: Filter by "Health."
   Expected Result: Displays only health resources.
   Actual Result: Displays only health resources. Shows SFU Mental Health Support and SFU Health and Counselling
   Pass
4. Clear Search:
   Steps: Clear the search input.
   Expected Result: All resources are displayed.
   Actual Result: All resources are displayed.
   Pass
5. Open Resource Links:
   Steps: Click on a resource link.
   Expected Result: Navigates to the correct URL.
   Actual Result: Navigates to the correct URL.
   Pass

The manual testing for the resources page was successful, with all test cases passing.

Test Cases for Dining Component

1. All Locations Displayes:
   Steps: Filter by 'All Locations'

   Expected Result: Displays all the dining options available on campus (only the ones put in the database currently).
   Actual Result: Displays all the dining options available on campus.
   PASS
2. Location Filtering:
   Steps: Filter by 'Cornerstone'
   Expected Result: Displays all the Restaurants at Cornerstone
   Actual Result: Displays only the dining options available at Cornerstone (only the ones put in the database currently).
   PASS
3. No Dining Options:
   Steps: Filter by 'Univercity' (As there are no dining options added there currently)
   Exoected Results: No results
   Actual Result: No dining option is displayed. Should consider adding an 'Oops..' message for the next iteration.
   PASS

Test Cases for Home and Calendar Components

Implemented unit tests for the **Home Component** using **React Testing Library** and **Jest**. The tests cover rendering and verifying essential sections of the `Home` component, such as the **hero section**, **image grid**, and the **calendar section**.

1. **Hero Section Rendering**

   - **Test:** Ensure that the hero section renders the correct title and text.
   - **What it checks:**
     - Verifies that the `SFU HUB` title is rendered.
     - Verifies that the text `Your central hub for all things SFU` is rendered.
2. **Image Grid Rendering**

   - **Test:** Ensure that the image grid renders the correct items for `Dining`, `Resources`, `Transportation`, and `Blogs`.
   - **What it checks:**
     - Verifies that the text links for `DINING`, `RESOURCES`, `TRANSPORTATION`, and `BLOGS` are rendered.
3. **Calendar Section Rendering**

   - **Test:** Ensure that the Calendar component is rendered correctly.
   - **What it checks:**
     - Mocked the `Calendar` component and verified that it is rendered with the placeholder text `Mocked Calendar`.
Testing for iteration 2
Parking
1. Default Category Selection- **Pass**

- Steps: Load the component with the default category selection "All Categories."
- Expected Outcome: All parking spots should be visible in the list.
- Result: Pass - All parking spots are displayed correctly when "All Categories" is selected.e selected category's parking spots, hiding non-matching items.

2. Map Display- **Pass**

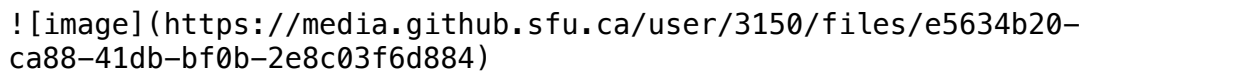
- Steps: Load the page and view the embedded map at the bottom.
- Expected Outcome: The Google Maps iframe should load and display the SFU parking map.
- Result: Pass - The SFU parking map displays within the iframe, loading and scaling correctly.

Campus to Campus testing
1. Partial Selection (Missing Fields)- **Pass**

- Steps: Select only the from field or to field.
- Expected Outcome: An error message should appear, saying, "Please select both the starting and destination campuses."
- Result: Error message appears correctly and no routes are listed.

2. Transit Icon Display- **Pass**

- Steps: Display routes that contain both "Bus" and "Expo Line" in the transit details.
- Expected Outcome: Both FaTrain and FaBus icons should display for each route where applicable.
- Result: Icons match the transit modes listed in route.transit. in this format
Test Results



Calendar API Tests

1. Creation Acceptance - **Pass**

- Steps: POST a valid event, with all required values in the request object
- Expected behavior: Valid calendar events can be created
- Result: Valid calendar event created

2. Creation Rejection - **Pass**

- Steps: POST an invalid event, with `start`, a required property,

missing
- Expected behavior: The request is rejected and a 400 Bad Request error is returned
- Result: The request is rejected

3. Creation Authentication - **Pass**

- Steps: POST a valid event (folowing our OpenAPI schema), with invalid credentials
- Expected behavior: The request is rejected and a 401 Unauthorized is returned
- Result: The request is rejected with error 401

4. Get Events - **Pass**

- Steps: GET the /calendar/events endpoint
- Expected behavior: An array of event objects following the OpenAPI schema are returned
- Result: The list of events is returned

Resources API Tests

1. Get Resources - **Pass**

- Steps: GET /resources endpoint
- Expected behavior: An array of resource objects following the OpenAPI schema is returned
- Result: The list of resources is returned

Dining API Tests

1. Get Restaurants - **Pass**

- Steps: GET /dining/restaurants endpoint
- Expected behavior: An array of restaurants objects following the OpenAPI schema is returned
- Result: The list of restaurants is returned

Transit API Tests (Iteration 2)

1. Get Transit - **Pass**

- Steps: GET /transit?stopNumbers=59314
- Expected behavior: An array of bus stops objects (with stop times in an array within) following the OpenAPI schema is returned
- Result: The expected result is returned

API Unit Tests (Iteration 2)

![image](https://media.github.sfu.ca/user/3093/files/

ff632bba-7e3c-4cd0-8a29-b878fe142fcb)

We have implemented code-based unit tests to ensure the reliability and functionality of our program.
- Allowing for continuous testing if code is modified
- Ensures all critical functionalities are thoroughly validated
- The tests allow for clear documentation of expected behaviors