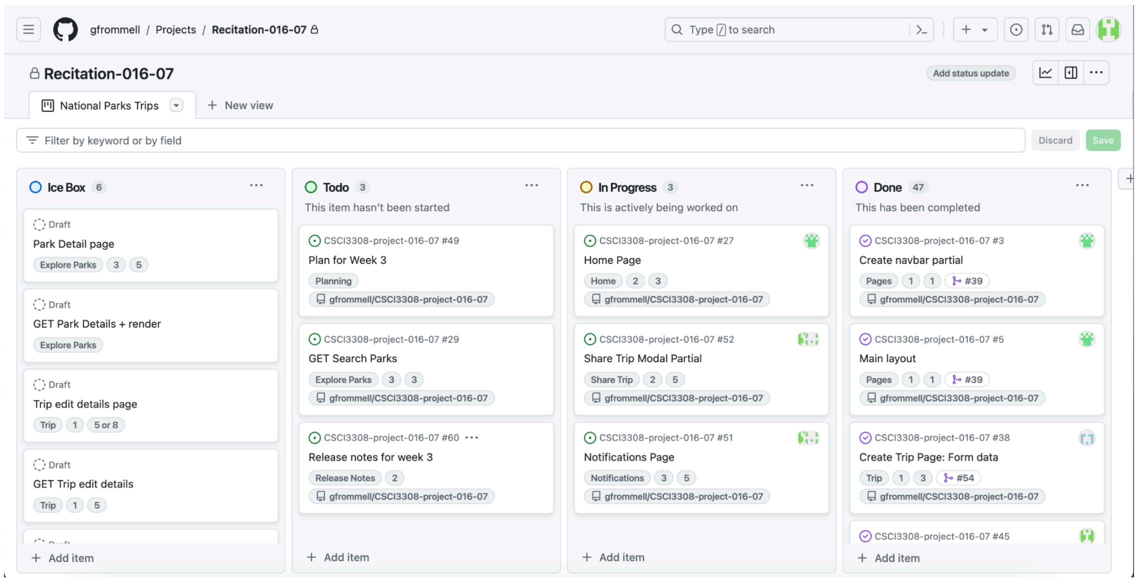
National Parks Road Trips (TrekTerra™)

Name	Github
Tehya Laughlin	@tehya-laughlin
Griffin King	@Griffin-king
Devin Green	@DevinGreen2002
Jack Schwalbach	@jasc9673 / @jackschwalbach1
Derick Sayavong	@Drock54651
George Frommell	@gfrommell

Our project was an application designed for planning trips to national parks. The goal was to deliver a seamless and intuitive user interface that simplifies the process of researching and scheduling trips to national parks, ensuring a smooth experience from start to finish. We utilized the National Parks Service API to retrieve data on national parks. Furthermore, we implemented a sharing feature that allowed users to share their trip plans with friends, with a goal of fostering a sense of community, akin to applications such as Strava. Although still in the beginning stages, the final product would be a social media like platform in which users can post their completed trips with images, and users may review trails and other activities in national parks so that others may optimize their trips. Users can easily browse through a curated list of parks, view images from the parks, and access details to help them plan their trips effectively. With a user friendly interface, and an intuitive site navigation, our goal was to empower users to explore the beauty and serenity of our national parks.

Project Tracker

https://github.com/users/gfrommell/projects/1/views/1



gfrommell / Projects / Recitation-016-07

Recitation-016-07

National Parks Trips

Ice Box 6

- Draft — Park Detail page — Explore Parks, 3, 5
- Draft — GET Park Details + render — Explore Parks
- Draft — Trip edit details page — Trip, 1, 5 or 8
- Draft — GET Trip edit details — Trip, 1, 5

Todo 3 — This item hasn't been started

- CSCI3308-project-016-07 #49 — Plan for Week 3 — Planning — gfrommell/CSCI3308-project-016-07
- CSCI3308-project-016-07 #29 — GET Search Parks — Explore Parks, 3, 3 — gfrommell/CSCI3308-project-016-07
- CSCI3308-project-016-07 #60 — Release notes for week 3 — Release Notes, 2 — gfrommell/CSCI3308-project-016-07

In Progress 3 — This is actively being worked on

- CSCI3308-project-016-07 #27 — Home Page — Home, 2, 3 — gfrommell/CSCI3308-project-016-07
- CSCI3308-project-016-07 #52 — Share Trip Modal Partial — Share Trip, 2, 5 — gfrommell/CSCI3308-project-016-07
- CSCI3308-project-016-07 #51 — Notifications Page — Notifications, 3, 5 — gfrommell/CSCI3308-project-016-07

Done 47 — This has been completed

- CSCI3308-project-016-07 #3 — Create navbar partial — Pages, 1, 1, #39 — gfrommell/CSCI3308-project-016-07
- CSCI3308-project-016-07 #5 — Main layout — Pages, 1, 1, #39 — gfrommell/CSCI3308-project-016-07
- CSCI3308-project-016-07 #38 — Create Trip Page: Form data — Trip, 1, 3, #54 — gfrommell/CSCI3308-project-016-07
- CSCI3308-project-016-07 #45

We used the Github project tracker to write up acceptance criteria for user stories and management tasks. Group members claimed tasks that they could manage, and we discussed who would take which tasks on Sunday of our sprint. We would check in on Wednesday of our sprint, and either move tasks out a week or our team would take on more tasks if we had time. Tasks were either placed in "In-Progress" or "Done" by the end of our sprint on Friday.

Video Demo

Link to video in Google Drive

https://drive.google.com/file/d/1fHvqvWjfGZbnYPPfelgiP95nxQ7Jon4Z/view?usp=drive_link

Link to video in repository

https://github.com/gfrommell/CSCI3308-project-016-07/blob/main/MilestoneSubmissions/VideoCSCI3308.mp4

VCS

https://github.com/gfrommell/CSCI3308-project-016-07.git

Contributions

Tehya Laughlin's Contributions

Product owner, scrum master, and full \stack developer. Tehya utilized SQL, Express JS and Node JS to get and post data to the back end. Tehya also used handlebars to create interfaces with html. Features worked on include:

- SQL database
- "Share Trip" UI and backend
- Trip details page UI and backend
- Editing trip title, date, and duration modal and backend
- Wholeistic UI changes to each page: colors, padding, rounded edges, shadows
- Partial and whole search functionality for exploring parks
- Park details page image carousel with automatic rotation

Griffin King's Contributions

- Boiler plating for the pages
- created the basic get APIs for explore parks, create trips, etc.
- Explore Parks in depth Backend for retrieving and displaying images
- Explore parks front end design for element placement and display
- Future Scope slide for the presentation

Devin Green's Contributions

Front end Developer. Devin utilized: HTML, Bootstrap, and Handlebars.

Small amount of backend support: PostgreSQL, Node.js

Partials

- head.hbs
- footer.hbs
- navbar.hbs
- messages.hbs

Pages

- Home.hbs: Populated Trips table using PostgreSQL database, this handlebar was implemented on index.js. The Trips table originally appeared on our home page. As a group we decided to move this table to our “All Trips” page.
- Implemented “Welcome” home page UI group decision.
- Edited trip title date for original home page trips table

Other

- User Test Case Evaluation.

Jack Schwalbach’s Contributions

Full stack developer.

Front end tools used: HTML, CSS, Handlebars, and Bootstrap.

Back end tools used: Javascript, PostgreSQL, and NodeJS

Features worked on:

- Notifications page
    - Created notifications SQL table.
    - Created notification boilerplate using Handlebars.
    - Used Bootstrap to create a UI element for the notifications page.
    - Used NodeJs and SQL queries to allow users to share and receive notifications, as well as accepting or declining notifications.
    - Updated the query on the all trips page to include shared users.
    - Created three different endpoints in index.js. (/notifications, /notifications/accepted, and /notifications/declined).

- Park details page
    - Used frontend and backend tools to dynamically display activities and things to do for any park selected on the explore parks page.
    - Joined multiple SQL tables to connect data.
    - Used Handlebars to display data given by SQL queries.
    - Used Bootstrap to create UI elements for the page.
    - Created endpoint in index.js (/park_details/:park_code).

Derick Sayavong’s Contributions

Full-stack developer. Utilized front-end languages and frameworks such as HTML, CSS, JS, Bootstrap, and Handlebars. Also used Express, NodeJS, and PostgreSQL for back-end related features.

Features worked on:

- Registration page UI and back-end
- Login page UI and back-end
- Create Trips page UI and back-end
- UI and Functionality to add park and items to each day when editing a trip
- Querying and displaying data from database for endpoints relating to the above features

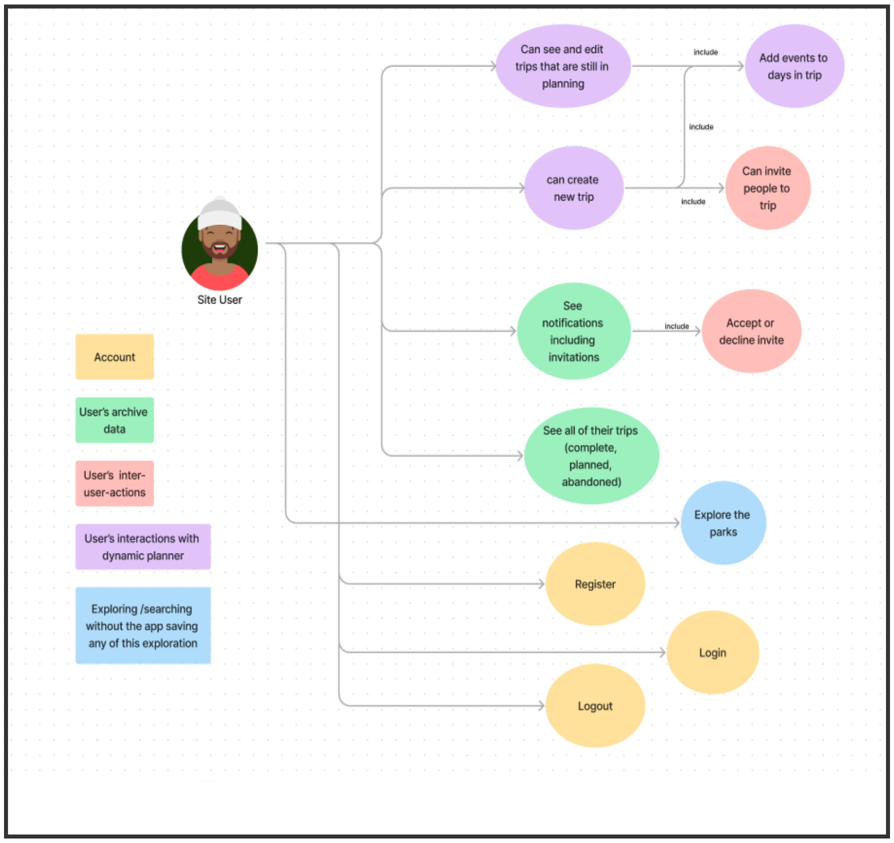
George Frommell’s Contributions

Full-stack developer. Used HTML, CSS, Bootstrap, Handlebars for front-end features. Mostly worked with PostgreSQL, JS and NodeJS for Backend features.

Features worked on:

- Github repo
- Alltrips.hbs w/ UI
- GET alltrips endpoint
- Logout.hbs and UI
- Edit trip details endpoint
- Querying and dynamically rendering data from database for above features

Use Case Diagram

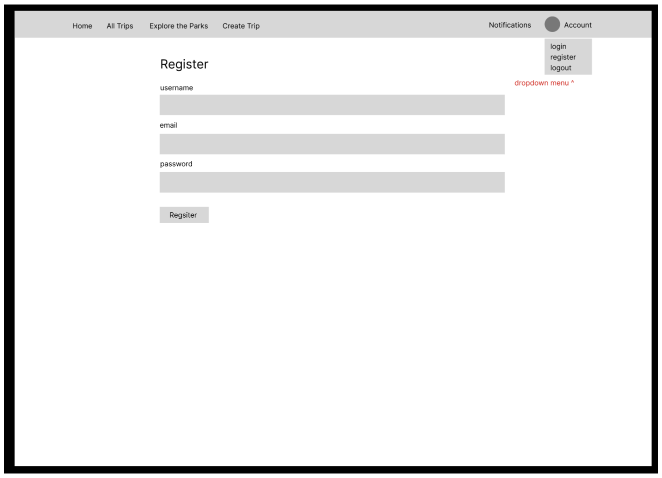

Can see and edit trips that are still in planning
include
Add events to days in trip
include
can create new trip
include
Can invite people to trip
Site User
See notifications including invitations
include
Accept or decline invite
Account
User's archive data
User's inter-user-actions
User's interactions with dynamic planner
Exploring /searching without the app saving any of this exploration
See all of their trips (complete, planned, abandoned)
Explore the parks
Register
Login
Logout

This is the use case diagram we made at the beginning of the project. In general, we were able to complete all features with at least Create capabilities, if not 'RUD capabilities.

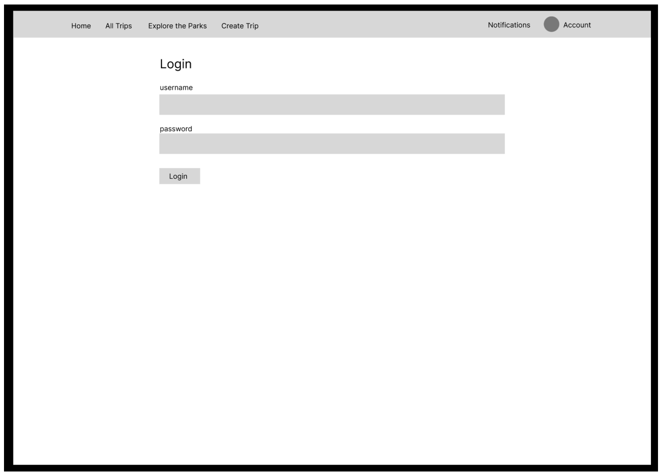
Wireframes

Link to zip file of all wireframes: https://drive.google.com/file/d/1aXzI1ZUhO4C4nr2bW7kiuyBb9C0jLdne/view?usp=drive_link

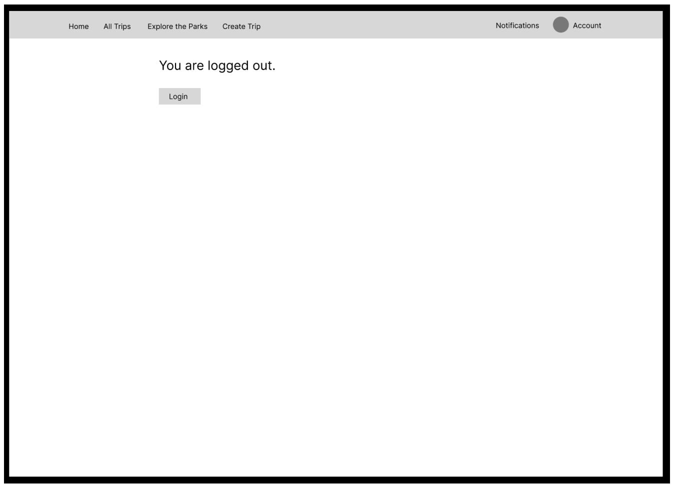
Register



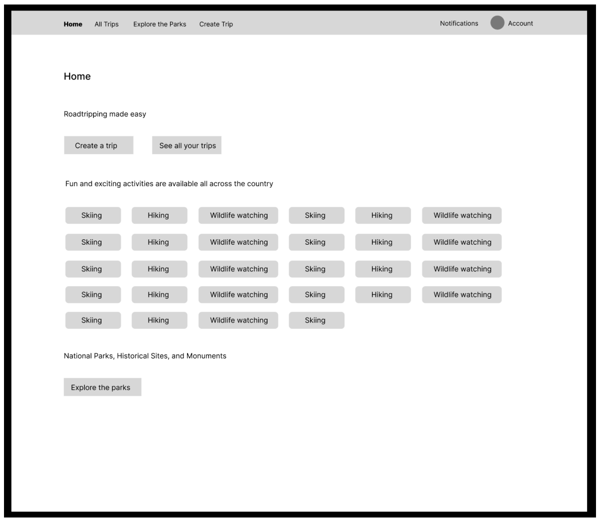
Login



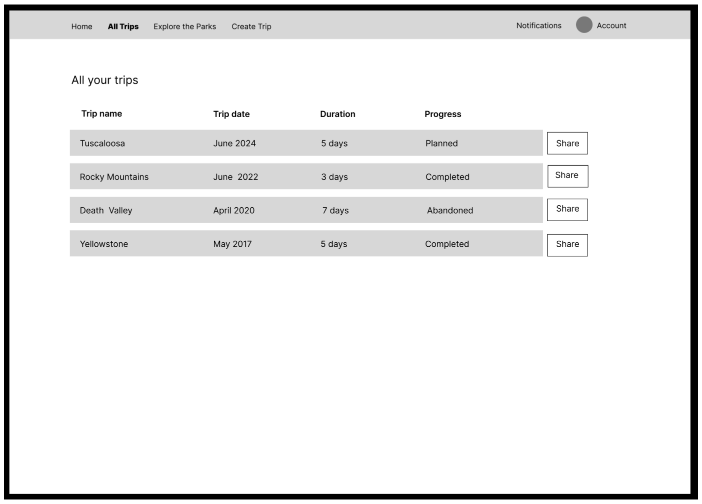
Logged out



Home



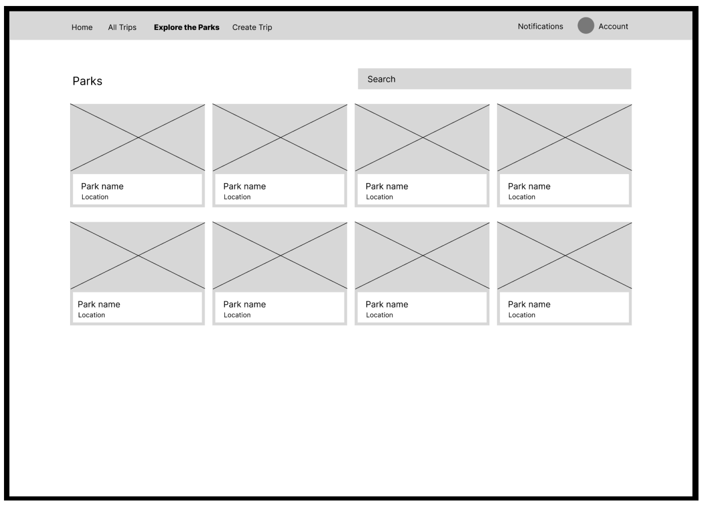
All Trips



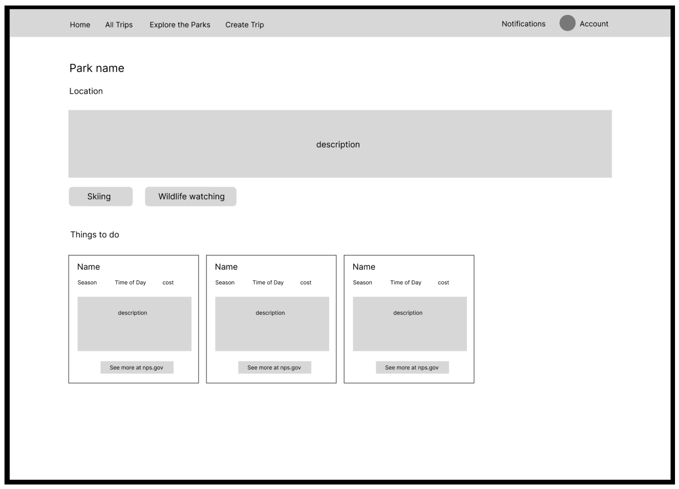
All your trips

Trip name	Trip date	Duration	Progress	
Tuscaloosa	June 2024	5 days	Planned	Share
Rocky Mountains	June 2022	3 days	Completed	Share
Death Valley	April 2020	7 days	Abandoned	Share
Yellowstone	May 2017	5 days	Completed	Share

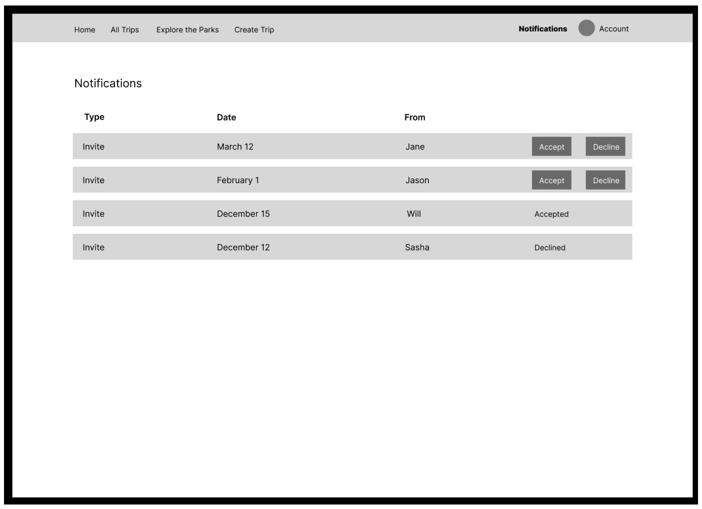
Explore the Parks



Park Details page



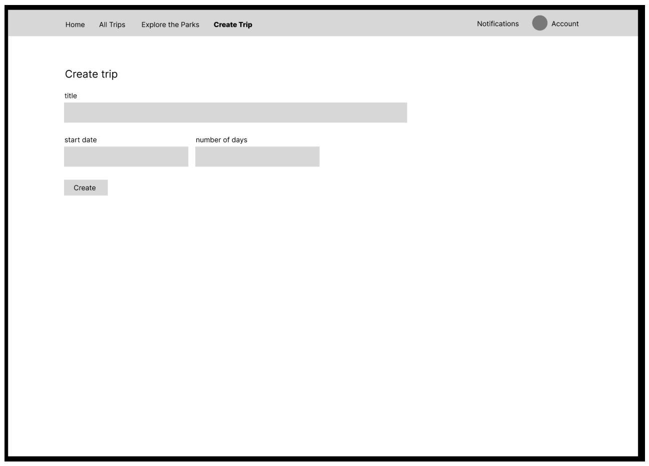
Notifications



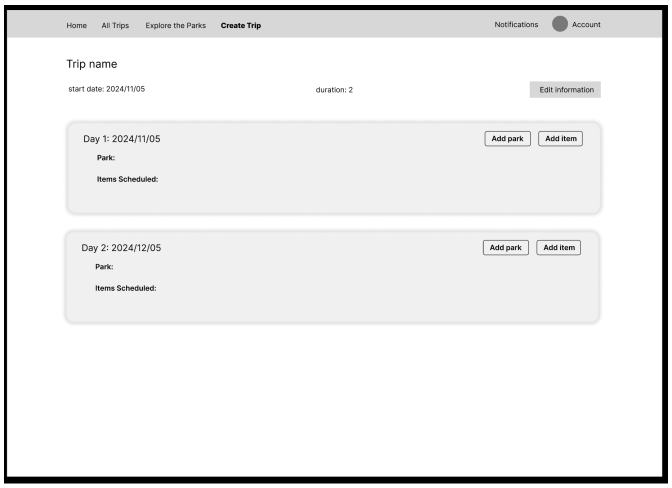
Notifications

Type	Date	From		
Invite	March 12	Jane	Accept	Decline
Invite	February 1	Jason	Accept	Decline
Invite	December 15	Will	Accepted	
Invite	December 12	Sasha	Declined	

Create Trip



Trip Details page



Test results

User Registration

- User: Student outside of CSCI (Devin's Roomate)
- Goal: Create a personal account or log in to access personalized features.

The user had no issue with the current registration implementation during the test, so we did not need to make any changes.

User Login

- User: Student outside of CSCI (Devin's Roomate)
- Goal: Login to previously created account Main Flow

Again, our initial implementation created before testing did not have any issues with logging in.

User Logout

- User: Student outside of CSCI (Devin's Roomate)
- Goal: Logout of previously created account

Our initial implantation was fully functional and working. After the Logout test case we did not need to alter any of our code for the logout functionality.

User Create Trip

- User: Student outside of CSCI (Devin's Roomate)
- Goal: Browse list of National Parks and create trip

The user had no issues performing the goal on our initial implementation, so we again did not need to alter any of our code pertaining to "create trips".

Deployment

Our application was deployed locally using Docker containers. Docker provided a convenient way to package the application and its dependencies into isolated environments.

Deployment Process:

Containerization: We containerized our application using Docker. This involved creating a Dockerfile to define the application's environment and dependencies, as well as a docker-compose.yml file to configure the services required for our app, such as the web server and database.

Building the Docker Image: Once the Dockerfile and docker-compose.yml files were set up, we built the Docker image using the docker-compose build command. This command fetched the necessary dependencies and created a Docker image containing our application.

Running the Containers: After building the Docker image, we used the *docker-compose up* command to start the Docker containers. This command launched the containers specified in the *docker-compose.yml file*

Accessing the Application: With the Docker containers running, the application was accessible locally via localhost. We accessed the application through a web browser using the port 3000 specified in the docker-compose.yml file.

Access Instructions:

To access and run the application on your local machine, follow these steps:

1. Ensure Docker is installed on your system. If not, you can download and install Docker from the official Docker website (https://www.docker.com/get-started).
2. Clone the repository containing the project files to your local machine.
3. Navigate to the project directory in your terminal or command prompt. In this case, CSCI3308-project-016-07/ProjectSourceCode.
4. Start the docker containers by running the following in terminal:

```
docker-compose up
```

5. Once the containers are up and running, open a web browser and navigate to http://loalhost:3000