

# SweatTogether

A platform for connecting with future workout partners

Team 007

(Mary Foley, Sophia Hubscher, Sagarika Sonni, Khoa Ha)

# Project Overview

**SweatTogether** is a web application that helps users build and participate in a supportive community to motivate them and make working out more approachable.

## Key features:

- Profile with information on a user, their workout habits, and their goals
- Explore page to view other profiles
- Messaging system for users to communicate with each other
- Login/authentication system

**Repository link:** <https://github.com/sophia-hubscher/cs426-sweattogether> (to go beyond the authentication page, log in with one of these mock accounts: [link](#))

# Team Members

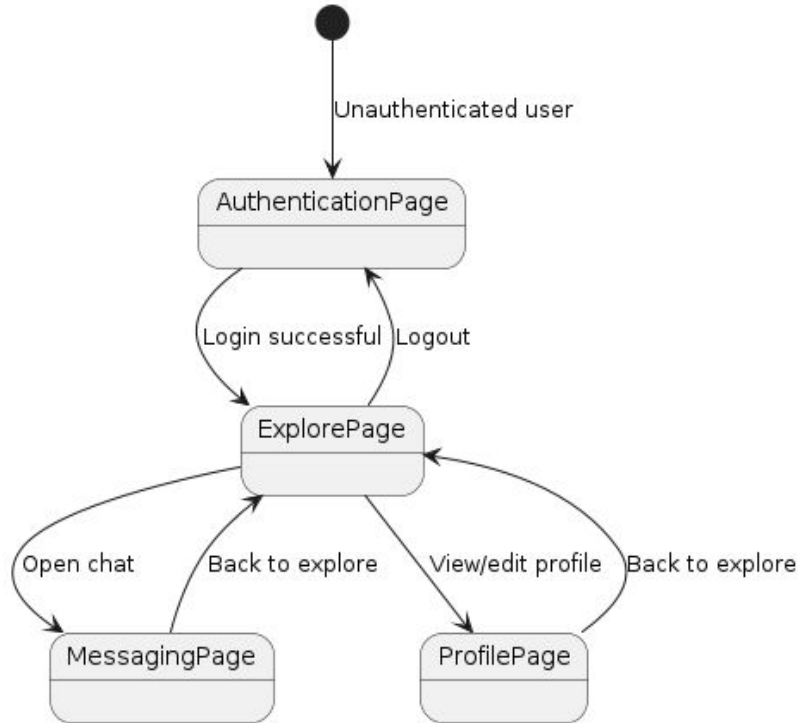
**Mary Foley:** Created data models, set up routes, and made the navigation bar.

**Sophia Hubscher:** Created the base project with Tailwind set up, and created the explore page with a working filter bar and sorting system.

**Sagarika Sonni:** Created the login page and profile page, where users can input information about themselves.

**Khoa Ho:** Created the messaging page for interacting with other users.

# UI Architecture Overview



**Authentication Page:** The entry point for users, where they log in. If unauthenticated, the user is redirected to this page.

**Explore Page:** The central hub after authentication, where users can browse potential workout partners, search, and filter available options.

**Messaging Page:** Users can chat with potential workout partners.

**Profile Page:** Users can view and edit their personal profiles, including workout preferences and details.

## Data Flow and State Management:

- State management is handled through the AuthProvider context, ensuring authentication states are maintained across pages.
- The navigation between pages is controlled by a combination of protected routes (ProtectedRoute) and public routes (e.g., Authentication Page).
- When a user logs in, they are redirected to the Explore Page, with all other pages accessible via navigation from the top navigation bar.

Note: All pages other than the Authentication Page can be accessed via the navigation bar, but the most common user flow is demonstrated on the left.

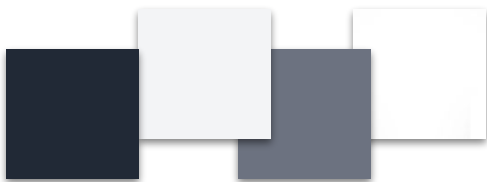
# Historical Development Timeline

Tasks/Issues	Start, End Date	February			March	Issue Numbers Completed
Initial Wireframing	2/16/25 - 2/22/25					NA
Navigation Bar Development	2/27/25 - 3/6/25					#4, #13
Explore/Home Page Development	3/1/25 - 3/13/25					#9
Login/Signup Page Development	3/13/25 - 3/26/25					#5
Profile Page Development	3/13/25 - 3/26/25					#7
Messaging Page Development	3/20/25 - 3/30/25					#6
Integration of Distance Algorithm	3/1/25 - 3/13/25					#16
User Testing and Final Refinements	3/24/25 - 3/30/25					#8

# Design and Styling Guidelines

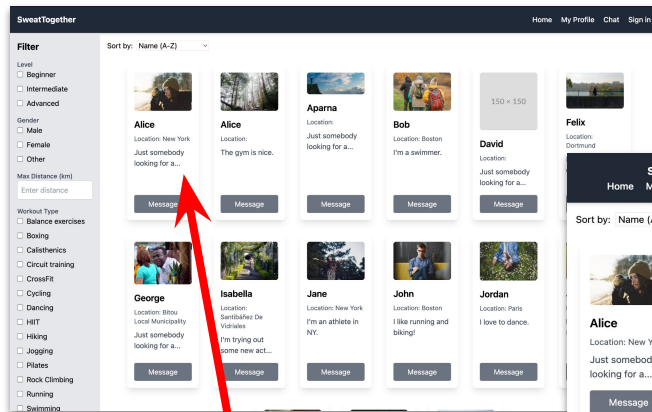
**Font:** UI Sans Serif (San Francisco on macOS, Segoe UI on Windows, and Roboto on Android)

**Colorscheme:** Greyscale

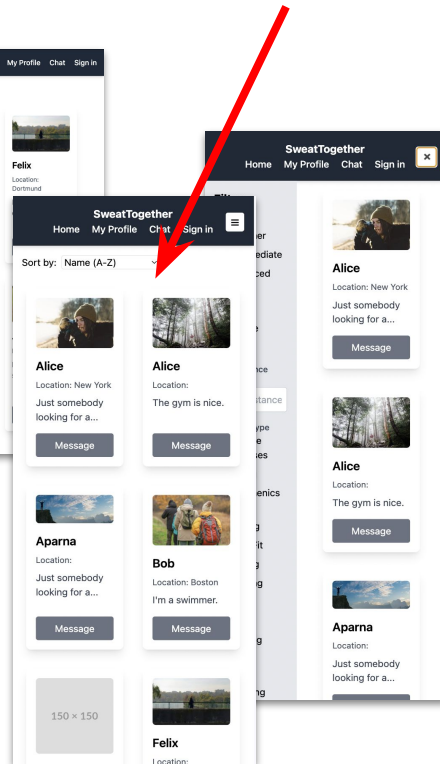


[Link to guidelines document.](#)

Page elements resize on smaller screens



High contrast ensures readability



# Performance Considerations

- **Using Optimized State Management**
  - Used React Context (AuthProvider) efficiently to prevent unnecessary re-renders across authentication states.
  - Minimized prop drilling by structuring state updates at the component level.
- **Efficient Navigation & Routing**
  - Used protected routes to prevent unnecessary authentication checks after login.
- **Image & Asset Optimization**
  - Used compressed SVGs and WebP images for UI assets to reduce.
- **Virtualized Lists for Explore Page**
  - Optimized user list rendering using react window to efficiently handle large datasets when searching for workout partners.

# Sophia: Assigned Work Summary

Issue: [Set up blank React project #2](#) PR: [Create blank React app #10](#)

Issue: [Add Tailwind formatting to navigation bar #13](#) PR: [Add Tailwind alignment to navbar #12](#)

Issue: [Make explore page with profile cards and filter bar #9](#) PR: [Explore page #15](#)

Issue: [Create distance utilities #16](#) PR: [Add 'max distance' filter #17](#)

Other PRs:

- [Create ui-guidelines.md #24](#)
- [Update browser tab look #21](#)

*Note: Commits can be viewed most conveniently by viewing the PRs for each issue.*



# Sophia: Code & UI Explanation

This code is for a piece of the filtering and sorting UI on the 'Explore' page. It uses the color scheme and spacing guidelines established in the design guide, and the 'toggleFilter' button appears only on small screen sizes to create a more responsive interface.

Fitting the filter bar on small screens was difficult, and thus, I used a React state `isFilterOpen` that was toggled depending on screen size or if a button was pressed.

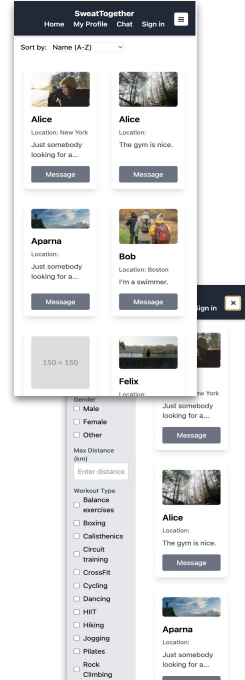
```
TS Countries.ts  TS CountryModel.ts  FilterBar.tsx  TS ProfileUtils.ts  NavBar.tsx  TS ProfileModel.ts

src > pages > Explore.tsx > Explore
function Explore() {
  // ...

  <NavBar />
  <button
    onClick={toggleFilter}
    className="md:hidden fixed bg-white top-7 right-4 px-2 py-1 rounded text-black"
  />
  {isFilterOpen ? 'x' : <span>#9776</span>}
  </button>

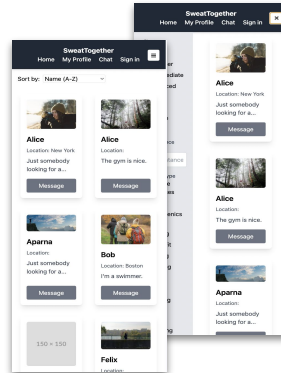
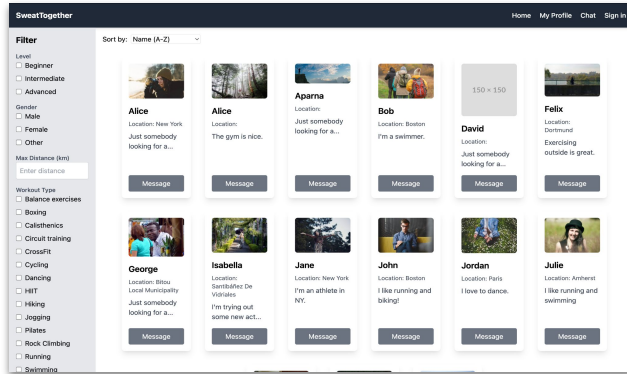
  <div className="flex">
    <FilterBar isFilterOpen={isFilterOpen} onFilterChange={handleFilterChange} />
    <div className="w-full">
      <div className={`p-4 flex ${isFilterOpen ? 'hidden' : ''} md:block justify-start`>
        <label htmlFor="sort" className="mr-2">Sort by:</label>
        <select
          id="sort"
          value={sortOption}
          onChange={handleSortChange}
          className="border rounded"
        >
          <option value="name-az">Name (A-Z)</option>
          <option value="level-low-high">Level (Low to High)</option>
          <option value="level-high-low">Level (High to Low)</option>
        </select>
      </div>

      <div
        className="flex flex-wrap gap-8 p-4 justify-center transition-transform duration-300 max-w-full overflow-x-auto"
        onClick={() => isFilterOpen && toggleFilter()}
      >
        {profiles.length > 0 ? (
          profiles.map((profile) => (
            <ProfileCard
              key={profile.id}
              id={profile.id}
              image={profile.image}
              name={profile.name}
              age={profile.age}
            />
          ))
        ) : (
          <div>No profiles found</div>
        )}
      </div>
    </div>
  </div>
}
```



# Sophia: Screenshots and Demonstration

My primary contribution this milestone was to create the Explore Page which contains a filter bar, sorting, and profile cards. The filter bar and sorting are functional. This page is accessed to find other users to message.



# Sophia: Challenges & Insights

## 1. Collaborating on a large codebase

**Obstacle:** Working with a large codebase required navigating different coding styles and understanding existing structures.

**Lesson Learned:** Effective communication and consistent coding practices, such as clear documentation and regular check-ins, are really helpful.

## 2. Creating responsive interfaces

**Obstacle:** Ensuring the UI remained functional and nice looking across different screen sizes required adjusting layouts, handling breakpoints, and testing on various aspect ratios.

**Lesson Learned:** Using tools like Flexbox/Grid and testing iteratively are essential to make layouts responsive.

# Sophia: Future Improvements & Next Steps

**Distance filter calculations:** The distance filter is currently being computed inefficiently, as each calculation is repeated unnecessarily. To optimize performance, we should store the results of each calculation so they can be reused instead of recomputed. This would reduce processing time and improve the overall efficiency of our application.

**Image uploads:** Uploaded images should be downsized upon upload to prevent excessive storage use. Resizing images to the correct dimensions for our interface will ensure faster load times and a more consistent user experience.

# Khoa: Assigned Work Summary

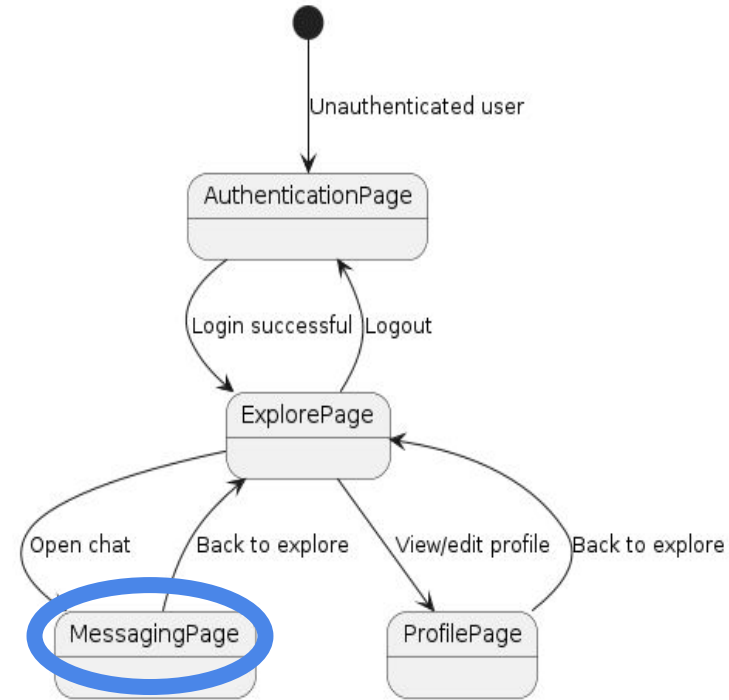
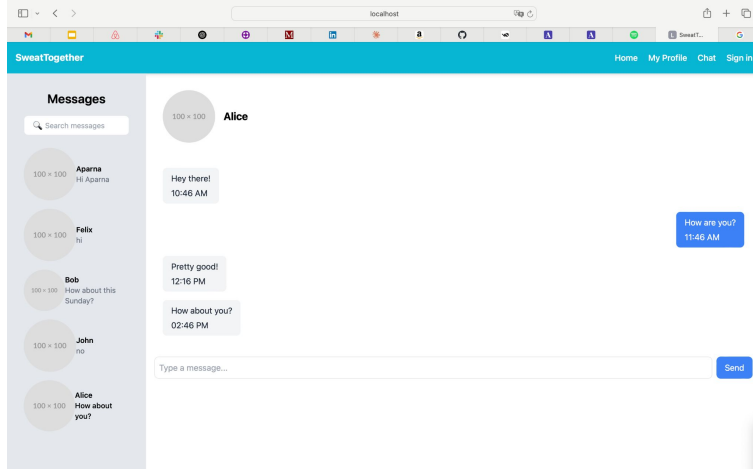
**Issue:** [Set up basic styling](#), [Add Tailwind formatting to navigation bar #13](#)

**Commits:** Implemented a simple UI for the main messaging page with a search bar for people that the user messaged, Fixed chat page UI and link each mock profile to its own separate chat, Added the chat sidebar to the messaging page, Added timestamp to messages, moved user's messages to the right...

**PR:** [Fixes the chat sidebar UI and changed the navbar color to blue](#)

# Khoa: Screenshots and Demonstration

My primary contribution this milestone was to create the messaging page with a left sidebar showing people the user has messaged and their most recent messages, a search bar to find them and the main chat area.



# Khoa: Code & UI Explanation

```
interface ChatSidebarProps {
  chats: { id: number; profile: { name: string; image: string } }[]
  selectedChat: number
  handleChatSelect: (id: number) => void
}

export const ChatSidebar: React.FC<ChatSidebarProps> = ({ chats, selectedChat, handleChatSelect }) => {
  const [searchQuery, setSearchQuery] = useState("")

  const filteredChats = chats.filter((chat) => chat.profile.name.toLowerCase().includes(searchQuery.toLowerCase()))

  return (
    <div className="w-1/4 min-h-screen bg-gray-200 p-4">
      <div className="p-4 border-b border-gray-200">
        <h1 className="text-2xl font-semibold text-gray-800 mb-4 text-center">Messages</h1>
        <div className="relative">
          <input
            type="text"
            placeholder="🔍 Search messages"
            className="w-full px-4 py-2 bg-gray-50 border border-gray-200 rounded-lg text-sm focus:outline-none focus:ring"
            value={searchQuery}
            onChange={(e) => setSearchQuery(e.target.value)}
          />
        </div>
      </div>

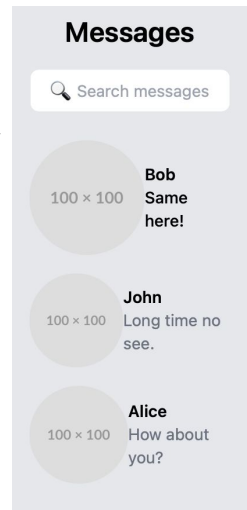
      <div className="flex-1 overflow-y-auto">
        {filteredChats.length > 0 ? (
          filteredChats.map((chat) => {
            const isActive = selectedChat === chat.id
            const hasMessages = mockChats[chat.id] && mockChats[chat.id].length > 0
            const lastMessage = hasMessages
              ? mockChats[chat.id][mockChats[chat.id].length - 1].text
              : "No messages yet."
          })
        ) : (
          <div>No messages yet.</div>
        )}
      </div>
    </div>
  )
}
```

This is the code for the ChatSidebar of the messaging page. It displays conversation previews, highlights the active chat, and provides search functionality to quickly find specific conversations.

Example of adhering to the application's style guidelines: I used consistent spacing and typography (text-2xl for headers, text-sm for content).

**Challenge:** Ensuring only relevant conversations appear in the sidebar

**Solution:** Implemented filtering logic that shows only profiles with existing messages and adds new conversations when messages are sent



# Khoa: Challenges & Insights

## 1. UI Layout Challenges:

- **Obstacle:** Making the chat area span the entire remaining page
- **Lesson Learned:** Flexbox layouts with proper height constraints (w-full, h-full) are essential for creating responsive, full-height interfaces.

## 2. State Management Complexities:

- **Obstacle:** Coordinating chat selection between sidebar and main content
- **Lesson Learned:** Lifting state to the nearest common ancestor component prevents synchronization issues. Props drilling is acceptable for moderate component depth but should be replaced with Context for deeper hierarchies.

**Key takeaways from working within my team:** I should have been more willing to ask for help when I'm stuck and not wait until the last week to do my task.



# Khoa: Future Improvements & Next Steps

**UI Improvements:** Fix the chat input bar positioning to always remain at the bottom of the page, resolve navbar visibility issues when navigating between pages, ensure all profile images stay the same size on the chat sidebar, implement message reactions or emojis to enhance user experience

**Technical Debt to Address:** Move from mock data to a proper backend integration, implement proper state management (Context API or Redux) for complex state, add integration tests for the complete messaging flow, implement consistent error handling throughout the messaging system

# Sagarika: Assigned Work Summary

Issue: [Profile Page UI](#)

Issue: [Make Login/Sign Up Page UI new feature](#)

PRs:

[Sagi profile](#)

# Sagarika: Screenshots and Demonstration

My contribution this milestone was to make the profile and authentication pages and also to set up the basic logic behind authentication. I also modified the routing to add protected routes. This added a layer of security to our application.

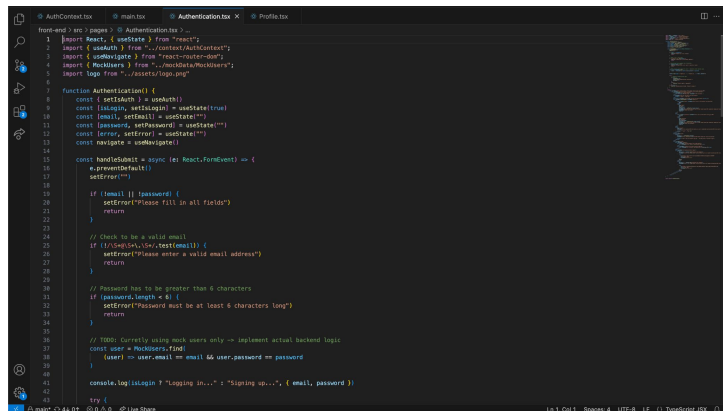
The screenshot shows the 'Edit Profile' page of the 'SweatTogether' application. The page has a dark header with the app name and navigation links: Home, My Profile, Chat, and Sign In. Below the header, there is a 'Choose File' button and a placeholder for a profile picture. The main form is titled 'Edit Profile' and includes fields for Name, Age (set to 0), City, and State. There are checkboxes for 'Workout Preferences' with options: Male, Female, and Other. Below these are dropdown menus for 'Country' (set to Andorra) and 'Level' (set to Beginner), and a 'Zip Code' field. A section for 'Workout Preferences' lists various activities with checkboxes: Balance exercises, Cardio, CrossFit, Dancing, Hiking, Pilates, Running, Walking, Yoga, Boxing, Circuit training, Cycling, HIIT, Jogging, Rock Climbing, Swimming, Weightlifting, and Other. A 'Bio' text area is at the bottom, followed by a 'Submit' button.

The screenshot shows the 'Sign Up' page of the 'SweatTogether' application. It features the app's logo at the top, followed by the title 'Sign Up'. There are input fields for 'Email' and 'Password', each with a placeholder 'Enter your email' and 'Enter your password' respectively. A blue 'Sign Up' button is positioned below the password field. A link 'Already have an account? Login' is located below the button. At the bottom, there is a section 'Or continue with' with buttons for 'Google' and 'GitHub'.

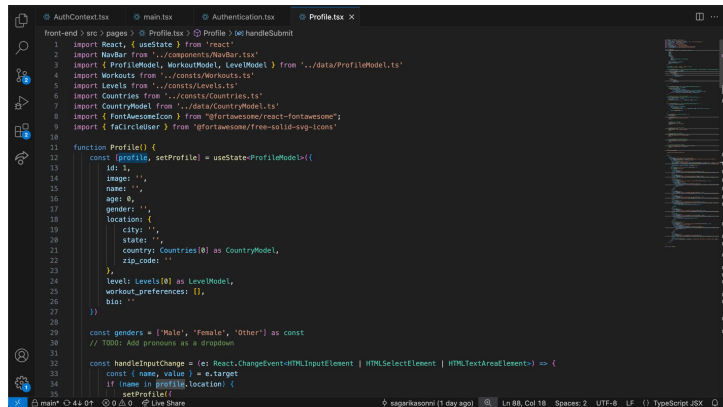
The screenshot shows the 'Login' page of the 'SweatTogether' application. It features the app's logo at the top, followed by the title 'Login'. There are input fields for 'Email' and 'Password', each with a placeholder 'Enter your email' and 'Enter your password' respectively. A blue 'Login' button is positioned below the password field. A link 'Don't have an account? Sign Up' is located below the button. At the bottom, there is a section 'Or continue with' with buttons for 'Google' and 'GitHub'.

# Sagarika: Code & UI Explanation

This code is for the Profile and Authentication pages. For the profile, the user can input multiple fields and a profile picture. The authentication is set up to allow users to create an account with the application by using an email. This enables the user to personalize their application per their needs.



```
front-end > src > pages > Authentication.tsx
1 import React, { useState } from 'react';
2 import { useAuth } from '../context/AuthContext';
3 import { useNavigate } from 'react-router-dom';
4 import { useRouter } from 'react-router-dom';
5 import { useAuth } from '../context/AuthContext';
6
7 function Authentication() {
8   const { setIsAuthenticated } = useAuth();
9   const [loginEmail, setLoginEmail] = useState('');
10   const [loginPassword, setLoginPassword] = useState('');
11   const [registerEmail, setRegisterEmail] = useState('');
12   const [registerPassword, setRegisterPassword] = useState('');
13   const navigate = useNavigate();
14
15   const handleSubmit = async (e: React.FormEvent) => {
16     e.preventDefault();
17     setErrors('');
18
19     if (loginEmail) {
20       setErrors('Please fill in all fields');
21       return;
22     }
23
24     // Check to see if email exists
25     if (!loginEmail || !loginPassword) {
26       setErrors('Please enter a valid email address');
27       return;
28     }
29
30     // Password has to be greater than 6 characters
31     if (loginPassword.length < 6) {
32       setErrors('Password must be at least 6 characters long');
33       return;
34     }
35
36     // TODO: Currently using mock users only - implement actual backend logic
37     const user = MockUsers.find(
38       (user) => user.email === loginEmail && user.password === loginPassword
39     );
40
41     console.log('Logging in...');
42     if (user) {
43       // Login successful
44       setIsAuthenticated(true);
45       navigate('/profile');
46     } else {
47       setErrors('Invalid email or password');
48     }
49   };
50
51   const handleRegister = async (e: React.FormEvent) => {
52     e.preventDefault();
53     setErrors('');
54
55     if (registerEmail) {
56       setErrors('Please fill in all fields');
57       return;
58     }
59
60     // Check to see if email exists
61     if (!registerEmail || !registerPassword) {
62       setErrors('Please enter a valid email address');
63       return;
64     }
65
66     // Password has to be greater than 6 characters
67     if (registerPassword.length < 6) {
68       setErrors('Password must be at least 6 characters long');
69       return;
70     }
71
72     // TODO: Currently using mock users only - implement actual backend logic
73     const user = MockUsers.find(
74       (user) => user.email === registerEmail && user.password === registerPassword
75     );
76
77     console.log('Registering...');
78     if (!user) {
79       // Register successful
80       setIsAuthenticated(true);
81       navigate('/profile');
82     } else {
83       setErrors('Email already exists');
84     }
85   };
86
87   return (
88     <div>
89       <h2>Authentication</h2>
90       <div>
91         <input type="text" value={loginEmail} onChange={setLoginEmail} />
92         <input type="password" value={loginPassword} onChange={setLoginPassword} />
93         <button type="button" onClick={handleSubmit}>Login</button>
94       </div>
95       <div>
96         <input type="text" value={registerEmail} onChange={setRegisterEmail} />
97         <input type="password" value={registerPassword} onChange={setRegisterPassword} />
98         <button type="button" onClick={handleRegister}>Register</button>
99       </div>
100     </div>
101   );
102 }
```



```
front-end > src > pages > Profile.tsx
1 import React, { useState } from 'react';
2 import { useProfile } from '../context/ProfileContext';
3 import { ProfileModel, WorkoutModel, LevelModel } from '../data/ProfileModel.ts';
4 import { useNavigate } from 'react-router-dom';
5 import { useAuth } from '../context/AuthContext';
6 import { useCountries } from '../context/CountriesContext';
7 import { useWorkouts } from '../context/WorkoutsContext';
8 import { useLevels } from '../context/LevelsContext';
9 import { useCountries } from '../context/CountriesContext';
10
11 function Profile() {
12   const { profile, setProfile } = useProfile();
13   const [id, setId] = useState('');
14   const [name, setName] = useState('');
15   const [age, setAge] = useState(0);
16   const [gender, setGender] = useState('');
17   const [location, setLocation] = useState({
18     city: '',
19     state: '',
20     country: Countries[0] as CountryModel,
21     zip_code: ''
22   });
23   const [level, setLevel] = useState(Levels[0] as LevelModel);
24   const [workout_preferences, setWorkoutPreferences] = useState('');
25   const [bio, setBio] = useState('');
26
27   // TODO: Add pronouns as a dropdown
28
29   const genders = ['Male', 'Female', 'Other'] as const;
30
31   const handleSubmit = (e: React.FormEvent) => {
32     e.preventDefault();
33     if (name in profile.location) {
34       setProfile({
35         id: id,
36         name: name,
37         age: age,
38         gender: gender,
39         location: location,
40         level: level,
41         workout_preferences: workout_preferences,
42         bio: bio
43       });
44     }
45   };
46
47   return (
48     <div>
49       <h2>Profile</h2>
50       <div>
51         <input type="text" value={id} onChange={setId} />
52         <input type="text" value={name} onChange={setName} />
53         <input type="text" value={age} onChange={setAge} />
54         <div>
55           <div>Gender</div>
56           <div>
57             {genders.map((gender) => (
58               <div>
59                 <input type="radio" value={gender} /> {gender}
60               </div>
61             ))}
62           </div>
63         </div>
64         <div>
65           <div>Location</div>
66           <div>
67             <input type="text" value={location.city} onChange={setLocation.city} />
68             <input type="text" value={location.state} onChange={setLocation.state} />
69             <div>
70               {Countries.map((country) => (
71                 <div>
72                   <input type="radio" value={country} /> {country.name}
73                 </div>
74               ))}
75             </div>
76             <input type="text" value={location.zip_code} onChange={setLocation.zip_code} />
77           </div>
78         </div>
79         <div>
80           <div>Level</div>
81           <div>
82             {Levels.map((level) => (
83               <div>
84                 <input type="radio" value={level} /> {level.name}
85               </div>
86             ))}
87           </div>
88         </div>
89         <div>
90           <div>Workout Preferences</div>
91           <div>
92             <input type="text" value={workout_preferences} onChange={setWorkoutPreferences} />
93           </div>
94         </div>
95         <div>
96           <div>Bio</div>
97           <div>
98             <input type="text" value={bio} onChange={setBio} />
99           </div>
100         </div>
101         <button type="button" onClick={handleSubmit}>Save</button>
102       </div>
103     </div>
104   );
105 }
```

# Sagarika: Challenges & Insights

## 1. Authentication Flow Complexity

**Obstacle:** Ensuring only authenticated users could access certain pages while maintaining a smooth user experience.

**Lesson Learned:** Implementing AuthProvider for centralized authentication and using protected routes improves security and state management.

## 2. State Management in Profile page

**Obstacle:** Preventing unnecessary re-renders and ensuring user data updates correctly without performance issues.

**Lesson Learned:** Using React's state management techniques such as useEffect helped optimize performance and maintain a responsive UI.

# Sagarika: Future Improvements & Next Steps

**Enhancing Authentication Security:** The next step would be to replace the existing placeholders for authentication with the actual backend. This would mean to implement the actual OAuth logins.

**Optimizing Profile Page Performance:** The next step for the profile page would be to implement lazy loading for profile-related data to improve load times and optimize state management to reduce unnecessary re-renders.

# Mary: Assigned Work Summary

Issue: [Set up mock data for profiles](#)

Issue: [Make navigation bar that routes between pages #4](#)

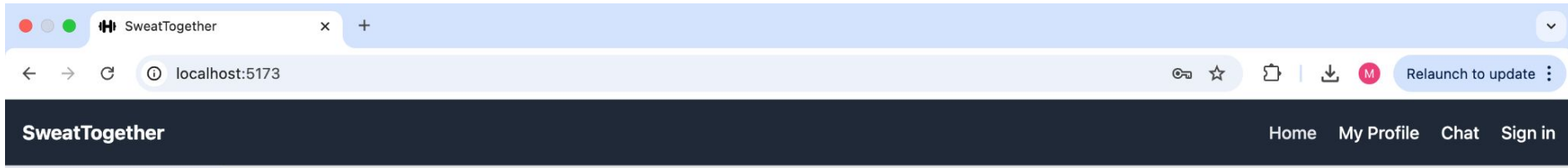
My commits were mainly related to the assigned issues.

PR's: [Set Up Pages and Navigation Bar #11](#), [Profile Mockup #14](#), [added mock data #22](#), [replaced the images for mock profile data #23](#)

Specific tasks completed within Milestone 1:

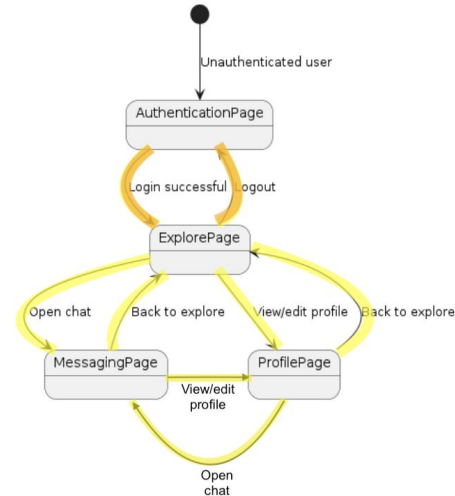
I created the navigation bar for our application and implemented routing between our main pages. I also created a type definition for the profile data and created mock data instances.

# Mary: Screenshots and Demonstration



The main component of the UI, I built was the navigation bar. The main functionality this provides our app is the ability to route between the different pages of the app. Our router has different routes connecting it to each of the pages: Home, My Profile, Chat, and Sign In.

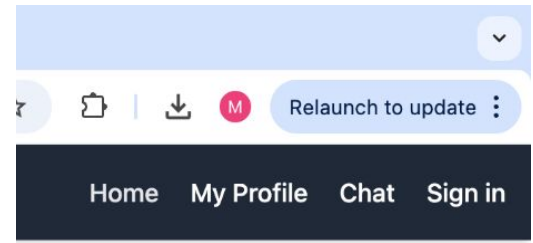
I initially built the navbar component to navigate along the yellow arrows, and later the functionality was extended to include the orange trajectory as well.





# Mary: Code & UI Explanation

This is the code for the nav bar, each of these pages, excluding sign in, also displays the nav bar as a component, so that the user can navigate back and forth from any of these pages. A challenge I faced was implementing a navigation bar as I didn't have familiarity with creating that kind of component. The solution I pursued was learning how to use react router and to implement the functionality I was looking for.



sagarikasonni, 6 days ago | 3 authors (Sophia Hubscher and others)

```
import { NavLink } from "react-router";
```

```
function NavBar() {
  const navLinks = [
    { to: "/", label: "Home", end: true },
    { to: "/profile", label: "My Profile", end: true },
    { to: "/messaging", label: "Chat" },
    { to: "/authentication", label: "Sign in" }
  ];

  return (
    <nav className="bg-gray-800 p-4 shadow-md flex flex-col md:flex-row justify-between items-center">
      <h1 className="text-white text-lg font-bold">SweatTogether</h1>
      <ul className="flex space-x-5 text-white font-medium">
        {navLinks.map(({ to, label, end }) => (
          <li key={to}>
            <NavLink
              to={to}
              end={end}
              className={({ isActive }) => isActive ? "text-gray-200" : "hover:text-gray-200"}
            >
              {label}
            </NavLink>
          </li>
        ))}
      </ul>
    </nav>
  );
}

export default NavBar;
```

Sophia Hubscher, 3 weeks ago • Add Tailwind alignment to navbar

# Mary: Challenges & Insights

## 1. Navigation in Project

**Obstacle:** Keeping components of our application clearly defined

**Lesson Learned:** While implementing routing for the pages in our app, I also created separate folders for the pages and components of our app to keep everything organized clearly. Thinking about modularity and organization beforehand was really useful for me.

## 2. Profile Data

**Obstacle:** Ensuring consistency for data referencing

**Lesson Learned:** To make sure that everyone uses the same calls for manipulating our data, setting up a type definition for our profile data was really useful.

# Mary: Future Improvements & Next Steps

**Updating Navigation Bar:** To make our navigation bar more similar to other existing apps and having it work the way users expect, I think we should add an icon of our app's main logo as a visual users can press to bring them to the home/explore page.

**Updating Profile Information:** In order for our app to be more inclusive, we've discussed adding pronouns as an additional field in our profile data type that will consist of multiple options. This is an improvement we can add in the near future.

**Distance Filter Calculations:** Perhaps to further optimize our distance filter algorithm, the computation need not be performed when measuring compatibility between users within separate continents.