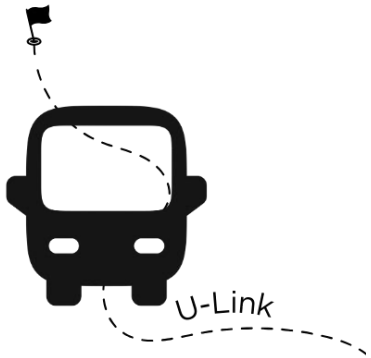


U-Link
Milestone 1

Team 3: Dongpei Zhang, Jason Vo, Ian Rapko, Sean McGonigle
Date: 3/30/2025

Project Overview & Team Members

U-Link is a live mbta tracker that gives incentive to take more public transit by gamifying traveling. Our Goal is to lower greenhouse gas emissions by making public transit accessible by providing bus routes and time of arrivals.

Roles:

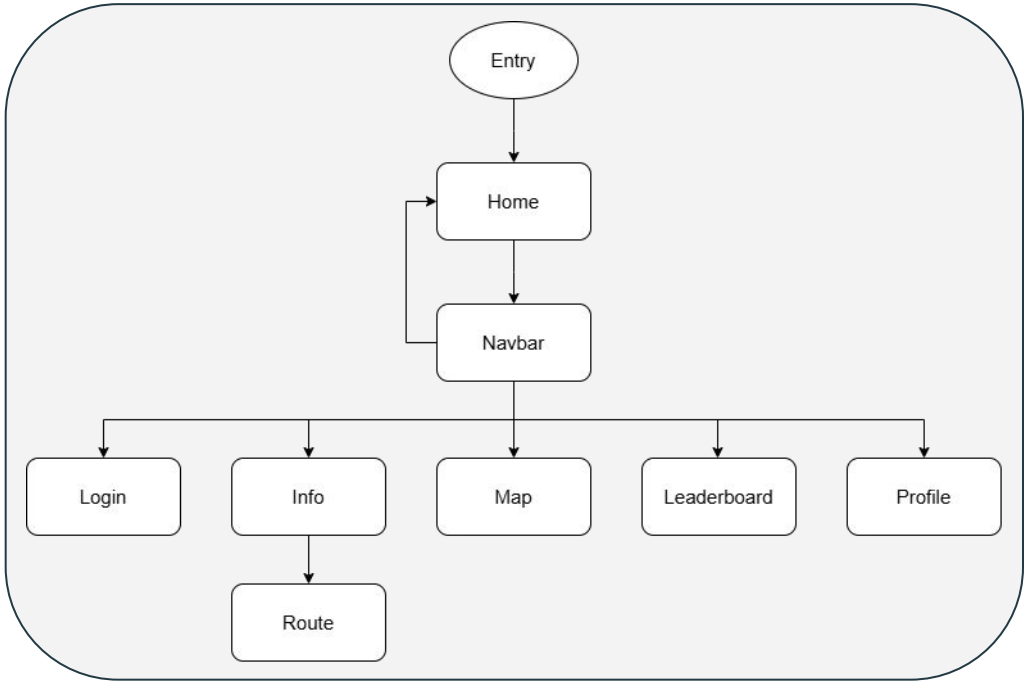
Dongpei Zhang - Developer

Jason Vo - Developer

Ian Rapko - Developer & UI Design

Sean McGonigle - Developer

Software Architecture Overview

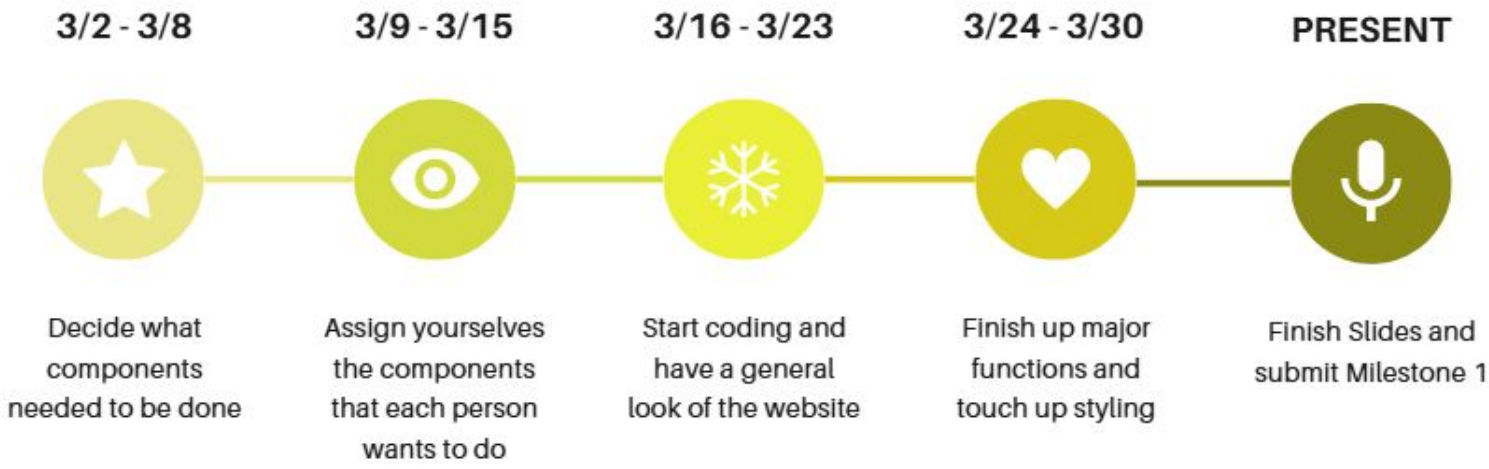


User Flow: Users begin on the app's home page, and then utilize the Navbar to visit other pages. React Router was chosen to facilitate navigation.
State is currently managed via local state (useState) within individual components.

```
src/
├── components/
│   ├── BusRoutesList.tsx
│   ├── HeaderContainer.tsx
│   ├── Leaderboard.tsx
│   ├── Login.tsx
│   ├── Map.tsx
│   ├── NavBar.tsx
│   ├── Profile.tsx
│   ├── RouteInfo.tsx
│   └── Title.tsx
│
├── pages/
│   ├── InfoPage.tsx
│   ├── LeaderboardPage.tsx
│   ├── LoginPage.tsx
│   ├── MapPage.tsx
│   ├── ProfilePage.tsx
│   └── RoutePage.tsx
│
└── main.tsx
```

Component Structure

Historical Development Timeline



Design and Styling Guidelines

Styling Document

Overview: We chose tailwind CSS as the CSS framework for this project. The main color scheme that we chose for the background is a neutral gray color and the text color is all white. We did this with a global CSS file and used `@apply bg-neutral-800` and `@apply text-white`.

For our components, we decided on a frost glass look using a blur effect. (`className="backdrop-blur-md bg-white/10 shadow-2xl rounded-lg overflow-hidden "`)

For the layout, the website follows a conventional website layout. The follow has a flex column grid system and the buttons in the application have consistent styling of `mt-2 px-4 py-2`

Accessibility considerations and responsive design strategies: Considering accessibility, we designed the UI to be easily navigated. All the pages are accessed via a navbar on the top of the page and all of the information is displayed upon click. Each page is also responsive with minimal lag and the user experience is simple. Since this app is a live MBTA bus tracker, the user has to gain information fast. With this in consideration, we made a separate info page that instantly displays the routes and location of buses. The map page also displays with the location of Boston as its preset.

Performance Considerations

- For our front-end implementation we tried to avoid unnecessary rerenders to avoid performance issues for the user
- leaderboard component we are going to limit the amount of users are displayed e.g. Showing top 50 and/or showing 50 +/- of your own rank.
- We also made sure to separate our components to remote modularity.

Individual Contributions

Assigned Work Summary - Sean McGonigle

Assigned & Completed Tasks/Issues:

- Active Lines List Issue
  - Create a functional UI list that displays all active lines as clickable items
- Info Page Issue
  - Create a functional UI info page for each line w/ stops, ratings and student comments

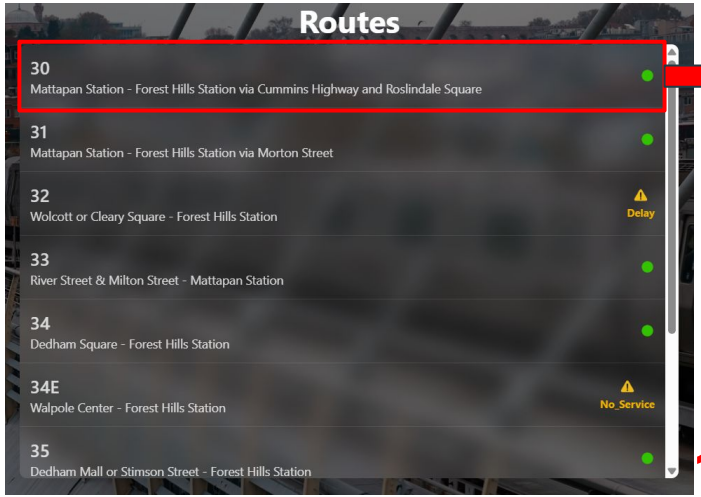
Assigned to All Members:

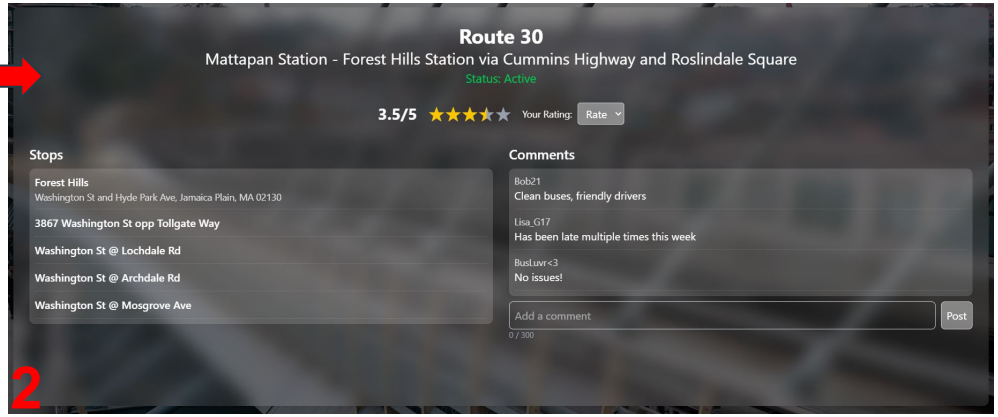
- Component Development Issue
  - Develop each core component separately
- Mock Data Issue
  - Create mock data with a JSON file format and display it into the UI components
- Styling and Responsiveness Issue
  - Apply CSS/Tailwind/Bootstrap or another styling approach. Ensure the UI is responsive across different screen sizes.

Pull Requests:

- Created Bus Routes List Component
  - Created component that displays all (currently mocked) bus routes with number/id, name, and a status indicator. Each route will be made clickable, leading to specific route page. Linked the component in main.tsx so it comes up when users click "Info" in the navbar
- Adjusted Routes List to Match Styling of Other Components
  - Routes list now matches styling of other components and has its own scroll bar
- Route Page
  - Created the page that opens when users click on a bus route from the list. It includes the route id, name, status, list of bus stops, rating out of 5 stars, and user comments. The rating and comments accept user input. All data is currently mocked.

Code & UI Explanation - Sean McGonigle





Pictured above are the two main components I developed: the list of bus routes (1) and the route info page (2). Both are styled according to our team's accepted style guidelines.

Component Hierarchy & Interaction: From the app's home page, the user clicks "Info" in the navbar, bringing them to the routes list (1). They then click on one of the routes in the list, bringing them to the corresponding route info page (2).

```tsx
{/* Stops List */}
<div>
  <h2 className="text-xl font-semibold mb-2">Stops</h2>
  <ol className="space-y-2 bg-white/15 rounded-lg p-2 overflow-y-auto max-h-60">
    {stops.map((stop, index) => (
      <li key={index} className="border-b pb-2 border-white/10">
        <p className="font-medium">{stop.name}</p>
        {stop.address && (
          <p className="text-sm text-white/80">{stop.address}</p>
        )}
      </li>
    ))}
  </ol>
</div>
```

This code snippet shows the tsx code I wrote for one small piece of the route info page (2), the bus stops list. It also demonstrates the use of Tailwind CSS to adhere to the accepted style guidelines.

Challenges and Insights - Sean McGonigle

Obstacles Faced:

- It was difficult to mock some of the data related to bus routes without knowing the exact format this data will be in once we have a backend. This could lead to the components needing some adjustments later on.
- Coordinating every decision, such as styling guidelines, component interaction, and sprint submissions, with the team can be challenging when everyone has their own schedules and responsibilities.

Key Takeaways:

- Communication is key
- Leave extra time before deadlines for making sure everyone is on the same page
- Start work early in case you run into issues

Assigned Work Summary - Jason Vo

Assigned Work + Completed Work:

Header Container Issue:

- Created a navbar component and header container component that holds the navbar.
- All navbar links properly route to different pages using React router.
- Added background to the page for styling

Map Page Issue:

- Created a functional UI map page that displays a map. Utilized Google Maps API.

Pull Requests:

- Added Navbar + Styling
- Converted all styling to Tailwind CSS
- Added react routers to navbar links
- Added background video + title styling
- Added Google Maps to map page
- Made all titles have consistent styling + added styling documentation

Assigned to All Members:

Component Development Issue

- Develop each core component separately

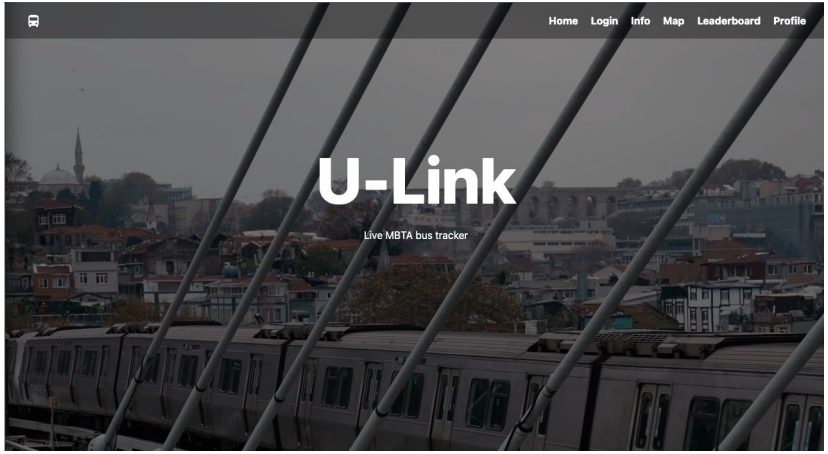
Mock Data Issue

- Create mock data with a JSON file format and display it into the UI components

Styling and Responsiveness Issue

- Apply CSS/Tailwind/Bootstrap or another styling approach. Ensure the UI is responsive across different screen sizes.

Code & UI Explanation - Jason Vo



- Background uses a built in video tag that plays a video from the public folder on auto play. Video is overlaid with a transparent background using z-10 to overlay.
- Title is styled with tailwind and uses transition-transform hover: to add hover effects to the title that enlarges it on hover.

```
<video
  autoPlay
  loop
  muted
  className="absolute inset-0 w-full h-full object-cover"
>
  <source src="/background.mp4" type="video/mp4" />
</video>
```

```
const Title = () => {
  return (
    <div>
      <h1 className="flex justify-center items-center mt-[200px] font-extrabold text-[100px] transition-transform duration-300 ease-in-out hover:scale-110">
       U-Link
      </h1>
      <p className="flex justify-center items-center "> Live MBTA bus tracker </p>
    </div>

  );
};

export default Title;
```

Code & UI Explanation - Jason Vo

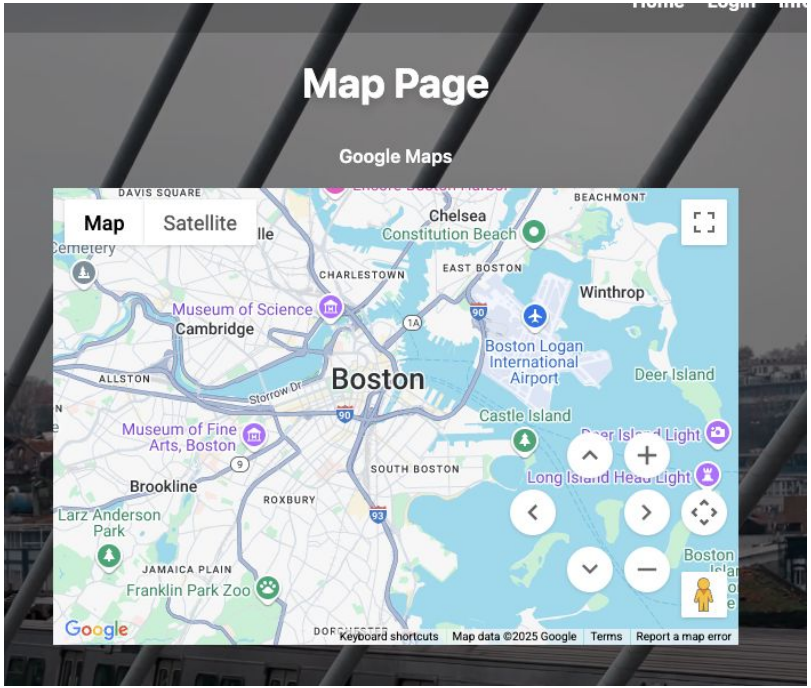

```
<Routes>
  <Route path="/" element={<HeaderContainer />} />
  <Route path="/map" element={<MapPage />} />
  <Route path="/login" element={<LoginPage />} />
  <Route path="/profile" element={<ProfilePage />} />
  <Route path="/info" element={<InfoPage />} />
  <Route path="/leaderboard" element={<LeaderboardPage />} />
  <Route path="/route/:id" element={<RoutePage />} />
</Routes>
```

Navbar is created with a nav tag and links are put inside an unordered list. A DirectionsBusIcon component is also imported using materialUI. Routes are added and successfully link the pages and the navbar + title components are put inside a HeaderContainer that gets called in main.tsx.

```
const NavBar = () => {
    return (
        <nav className="fixed top-0 left-0 flex w-full items-center justify-end bg-black/20 py-7">
            <div className="absolute left-8">
                <DirectionsBusIcon className="h-8 w-8 text-white" />
            </div>
            <ul className="absolute right-8 flex gap-5 list-none m-0">
                <li><Link to="/" className="font-bold text-white hover:underline">Home</Link></li>
                <li><Link to="/login" className="font-bold text-white hover:underline">Login</Link></li>
                <li><Link to="/info" className="font-bold text-white hover:underline">Info</Link></li>
                <li><Link to="/map" className="font-bold text-white hover:underline">Map</Link></li>
                <li><Link to="/leaderboard" className="font-bold text-white hover:underline">Leaderboard</Link></li>
                <li><Link to="/profile" className="font-bold text-white hover:underline">Profile</Link></li>
            </ul>
        </nav>
    );
}
```

```
const HeaderContainer = () => {
    return (
        <div>
            <Title />
            <NavBar />
        </div>
    )
}
```

Code & UI Explanation - Jason Vo



Map page and component is created by importing GoogleMap and LoadScript from google maps. This is installed by “npm i -S @react-google-maps/api”. An API key is generated using google cloud and is called from an .env file.

```
// boston as center
const center = {
  lat: 42.3601,
  lng: -71.0589,
};

const Map = () => {
  return (
    <div className="flex flex-col justify-center items-center">
      <h2 className="font-semibold mb-4">Google Maps</h2>
      <LoadScript googleMapsApiKey={import.meta.env.VITE_GOOGLE_MAPS_API_KEY}>
        <GoogleMap mapContainerStyle={containerStyle} center={center} zoom={10} />
      </LoadScript>
    </div>
  );
};

export default Map;
```

Challenges and Insights - Jason Vo

Challenges:

- *Learning Tailwind CSS*
  - Learning tailwind was an obstacle at first however after reading the documentation and coding, it gets easier and the logic is intuitive.
- *Learning how to integrate google maps into the application*
  - In order to integrate google maps you have to generate an API key using google cloud. Researching how to implement this coming in I did not know how to do it however react makes it very easy. By simply installing @react-google-maps/api via node, react has built in components that can be called and then all you have to do is input the API key that is generated for you by google cloud.

Insights:

- Learning how to implement the components was a challenge however given the timeframe I was able to properly read documentation and implement them. For future milestones, now I know the time needed in order to properly learn these technologies and given that can communicate better with my team on the development timeline.

Assigned Work Summary - Dongpei Zhang

Assigned Work Summary/complete

Profile Page

- Create a profile page that currently takes mock data to display the current user eg. total rides, name, etc
- Users can change their own bio using the edit bio button.

Login Page

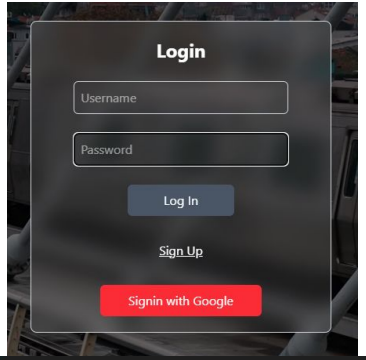
- Created a login page that currently doesn't work since I want to work with Google OAuth, so right now it just the appearance.
- Contains username and password input fields that currently shows in console after login button click, which shows that its getting the input correctly

Pull Requests

- https://github.com/JasonnVo/U-Link/pull/24
- https://github.com/JasonnVo/U-Link/pull/25
- https://github.com/JasonnVo/U-Link/pull/35
- https://github.com/JasonnVo/U-Link/pull/39
- https://github.com/JasonnVo/U-Link/pull/40

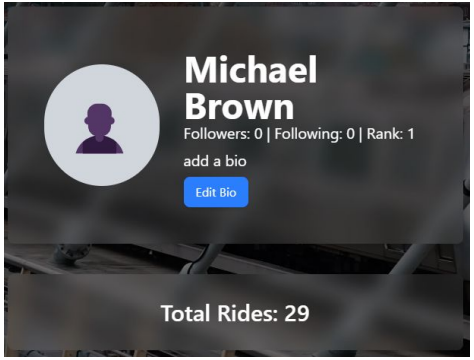
Code & UI Explanation - Dongpei Zhang

```
const [students] = useState<Student[]>([
    { name: "Michael Brown", rides: 29, points: 9000 }
]);
```



Login Page

- Current appearance of the Login page
- Currently a shell that takes in username and passwords that is displayed in console
- Waiting on the development of the backend to fully implement functionality



- Currently the profile page displays mock data of {name, total ride, points}
- Has a button when clicked provides a input field that allows the user to change their bio, which currently isn't saved into any database.

```
<input
    className="w-4/5 p-2 border border-gray-300 rou
    placeholder="Username"
    value={username}
    onChange={(e) => setUsername(e.target.value)}
/>
<input
    className="w-4/5 p-2 border border-gray-300 rou
    placeholder="Password"
    type="password"
    value={password}
    onChange={(e) => setPassword(e.target.value)}
/>
<button onClick={handleLogin} className="w-2/5 p-2
    Log In
</button>
<button onClick={handleSignup} className="w-2/5 p-2
    Sign Up
</button>
<button className="w-3/5 p-2 bg-red-500 text-white
    Signin with Google
</button>
```

```
const handleLogin = () =>{
    console.log("Logging in with:", { username, password })
};
```

These features will impact the overall the UI architecture because to gain access to profile page you need to login and the profile page interacts with the leaderboard so that you can tell what rank you are among other users. (the gamification of mbta)

```
<h1 className="text-5xl font-bold mt-2 text-white">{students[0].name}</h1>
<p className="text-xl text-white">Followers: 0 | Following: 0 | Rank: 1</p>
{isEditing ? (
    <div className="mt-2">
        <input
            type="text"
            value={tempBio}
            onChange={(e) => setTempBio(e.target.value)}
            className="p-2 border rounded w-full text-white"
        />
        <button onClick={handleSaveProfile} className="mt-2 px-4 py-2 bg-gre
            Save Bio
        </button>
    </div>
) : (
    <p className="text-xl text-white mt-2">{bio}</p>
)}

{!isEditing && (
    <button onClick={handleEditProfile} className="mt-2 px-4 py-2 bg-blue-50
        Edit Bio
    </button>
)}
```

Challenges and Insights - Dongpei Zhang

Challenges:

- Since I am pretty new to React it was a challenge getting used to using things like useState and other react functionalities. Along with that this was my first experience with tail wind so there was a lot of googling and documentation reading
- I think communication was a bit of a challenge these couple week, since everyone has their own schedules and it is around of time of midterms, so I understand.

Insights:

- I think next milestone we need to have better communication especially with us probably working with the backend
- Better time management

Assigned Work Summary - Ian Rapko

Assigned Work + Completed Work:

Header Page Issue:

- Creating a page that updates live user data for rides completed
- Automatically ranks and sorts riders all time based on number of rides
- Styled with tailwind

Pull Requests:

- Seperate Pages
- Leaderboard Page with mock data
- Adding Leaderboard tab/page

Assigned to All Members:

Component Development Issue

- Develop each core component separately

Mock Data Issue

- Use and create mock data with a JSON file format and display it into the UI components

Styling and Responsiveness Issue

- Used tailwind for creating a clean page, responsive across different screens.

Challenges and Insights - Ian Rapko

Challenges:

- Understanding how to use props and react elements within normal js functions took a bit to get down for the sorting, as I had only ever created relatively static sites before this
- I wasn't entirely sure on what the end goals are for my tab as we didn't talk about what we want each issue to look like when completed.

Insights:

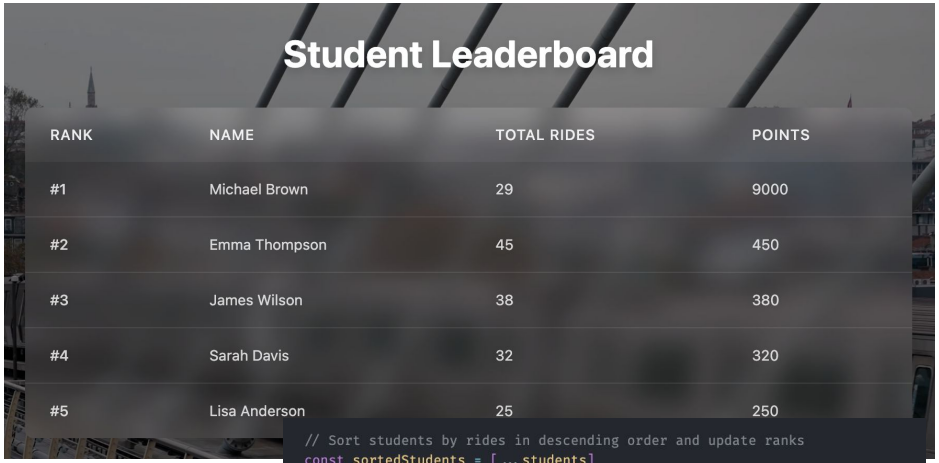
- Better communication and more strict issue creation/pull requests have to be made
    - In this milestone a lot of our work is based on additions we made throughout the working process, not direct through the issues we initially made
    - We have to just be more clear about what we plan to do in each PR.

Code UI & Explanation - Ian Rapko

Current appearance of the leaderboard page, frosted glass design, and auto sorted rows on the leaderboard (currently mock data)

```
<tbody className="divide-y divide-white/10">
    {/* Map through students and create a row for each */}
    {sortedStudents.map((student) => (
        <tr
            key={student.rank}
            className="hover:bg-white/5 transition-colors
            duration-150"
        >
            <td className="px-6 py-4 whitespace-nowrap">
                <div className="text-sm font-medium text-white/
                80">
                    {/* Display the student's rank */}#{student.
                    rank}
                </div>
            </td>
            <td className="px-6 py-4 whitespace-nowrap">
                {/* Display the student's name */}
                <div className="text-sm text-white/80">{student.
                name}</div>
            </td>
            <td className="px-6 py-4 whitespace-nowrap">
                {/* Display the student's total rides */}
                <div className="text-sm text-white/80">{student.
                rides}</div>
            </td>
            <td className="px-6 py-4 whitespace-nowrap">
                {/* Display the student's total points */}
                <div className="text-sm text-white/80">{student.
                points}</div>
            </td>
        </tr>
    ))
```

Table creation, sorted by the different properties of the student class, and row for rank, created by the algorithm to the right



Function to take in student data, and sort the array of students for amount of points

```
// Sort students by rides in descending order and update ranks
const sortedStudents = [...students]
    .sort((a, b) => {
        // Sort by points first
        if (b.points !== a.points) {
            return b.points - a.points;
        }
        // If points are equal, sort by rides
        return b.rides - a.rides;
    })
    .map((student, index) => ({
        ...student,
        rank: index + 1, // Add rank during rendering
    }));
```

Future Improvements & Next Steps

- Use real-time bus data from the MBTA API
- Add a feature allowing users to log their bus rides (for use in leaderboard)
- In the route info page, limit the number of user inputs (rating & comments) in a given time period to optimize performance and prevent spam
- Coordinate bus data usage between components to avoid unnecessary API calls
- Implement Google authentication for user login
- Add markers on the google maps that will update in real time based on MBTA API data