Lisa Chan and Allyson Young

MBT APP

Our web app idea is to create a next bus tracker, using the MBTA API. This idea is modeled from the example project idea from Piazza, and we plan on having three major features to our app. Our first feature is the “next bus” tracker, which will use the user’s current location and show the user the real time location and distance of all buses that go through the station they are at. If they’re not at a station, the app will look for the closest bus stop near you and list all buses and their distances from that stop. Our second feature is “best route to take”, which will require a user to input a destination and determine what the best route is to get from their current location to the destination. Our third feature is having an “alerts” page, which will use channels and websockets to get real time updates of alerts from the MBTA API. This page will list all alerts about all routes/buses. A user must be registered and logged in to be able to use the app. Our app will give users an option to “favorite” a route or a bus stop so they can have easy access to those locations saved in their profiles.

This app has a few user stories. As a non-registered user, Nathaniel can only view the register/log in page and the about page of the app, which is a short description of how to navigate and use the app. As a registered user, Nathaniel can enable location services and use the next bus tracker feature to find the next bus coming to the stop he’s at or the stop nearest to him. As a registered user, Nathaniel can use the best route to take feature to find buses that go from his location to a destination he can input. As a registered user, Nathaniel can view the alerts page to see if any of the buses he takes have any delays or breakdowns.

We are storing our data using PostgreSQL in a User and Preferences resource. Each user with have a name, email, and password which is entered upon registration, as well as an optional profile picture which can be viewed from their profile. We are also keeping track of each user’s route preferences and favorite bus stops in the preferences resource.

The major pages of our application are basically the three main features of our app. The landing page/homepage is the registration/login page. A user cannot use the app without logging in or registering for an account. The main page will also contain a short blurb of what the app is about and what features it has. The main page when a user is logged is a homepage with multiple tabs at the top that the user can visit. Each tab will contain each of app’s features. The Next Bus and Directions features of our app will have a map embedded via the Google Maps API, and the user will have to enable

their location services for the geolocation to properly work. The Alerts tab will have a feed of alerts, with each latest alert being added to the top of the feed.

Our three experiments are a JavaScript front end framework experiment with React.js and the Google Maps API, accessing alerts from the MBTA API, and determining if retrieving data necessary for the real time bus tracking component is feasible. Regarding our front end JS framework experiment, we first didn't use any frameworks and just tested out the different ways to use the Google Maps API using HTML/CSS and a separate JavaScript file. After we experimented a little bit with geolocation and the directions features of the Google Maps API, which can be seen in an earlier commit in the "maps" directory, we decided to try to incorporate React as our front end framework. We started from scratch by generating the app by using "create-react-app". We found that using script tags for external JavaScript libraries and files was not an easy task in React because both React and Google Maps take time to load into the DOM. We also learned that we would need Google Maps to render only after the React app has been rendered since the map can't be embedded in a React component that doesn't exist yet. So, as a result, we're investigating further into how to do just that.

The following two experiments were both built on elixir applications. While working on these experiments, it was apparent that the documentation for the MBTA API is very clear which works to our advantage. In the experiment for querying the alerts data, we faced no apparent challenges. From the data, we were able to map the results based on specific attribute we need for the alerts page. The nextbus experiment had a similar conclusion. We were looking for evidence that it would be at least possible to query data by stops, routes, and buses. After working with the API and building three methods to query information pertaining to the nextbus page and specifically each of the stops, routes, and buses, we found that this is certainly feasible.

Regarding our project status, we have made a lot of progress on brainstorming details and exactly how our app will function with the stack we have in mind. Doing the experiments definitely made us more aware of how the different features of our app will come together, and we have run into problems that were important for us to see before diving into our actual project. The experiments did not show to us that we had to change anything. We are able to hit the MBTA API with our credentials and API key, and we're also able to query through their database. Regarding our front end, although we don't have Google Maps successfully embedded in our React framework, we're confident that we will be able to use the Google Maps API in our components after a bit more investigation on how to load asynchronously.