The MBTA tracker will allow its users to see and monitor the incoming trains and buses from the stops closest to them. Each stop will show a list of the incoming trains/buses alongside their line color, direction and arriving time in minutes. The tracker will also update every 15 seconds in order to provide frequent updates to the user.

A typical user, let's say Alice for example, will be able to open the MBTA tracker and see the incoming trains/buses closest to her. She will also be able to get updates of the stations without the need of reloading the page. If Alice were to move to a different location, the app would update accordingly and show her the closest stations from that location she moved to.

The app gets most of its data from the MBTA API. The page will get the user's location using HTML5 and will feed the coordinates to the API request. The API will then return a list of the closest stops from that location. The app then selects the three closest stops and requests their routes. For each one of these routes, the app gets their line, direction and estimated time of arrival.

The majority of the user actions will take place in the app's homepage. The app will automatically get the location and display the routes in the homepage. Here, the user has the ability to collapse or expand the stops to hide or display its routes. The user can also click on the stop's name and they will be taken to a new page that only displays the routes and information for that stop.

In order to get prepared for this project I did three experiments that allowed me to get more comfortable with the frameworks and API that I would use. My first experiment consisted on testing the API with my key and interpreting that data. I used Poison in Elixir to request the data from the API and decode it. After the data was decoded, I would go and extract the information that I needed for the app.

My second experiment was to use websockets to push timed updates. I created a channel and added some sample data to it. I then created an invisible button in the page and set it to click itself every 10 seconds. I would have preferred an alternative that would've allowed me to get real-time updates, but due to the API restrictions, a user is only allowed to make the same request every 10 seconds. After creating the invisible button, I added an event listener that would in this case modify the sample data and push it to the user.

My last experiment consisted of using handlebars to format the output from a sample JSON data. I created a simple template that would show the line, direction and estimated time of arrival. Then I placed sample data in JSON format on the app, and had handlebars compile the view using that sample data. While the output for this experiment is very simple, I plan on having a more complex design for the final app.

So far, I think that I have implemented about 50% of the app. The things that still remain to be done are creating a design for the app, tying everything together and implementing a few extra features.