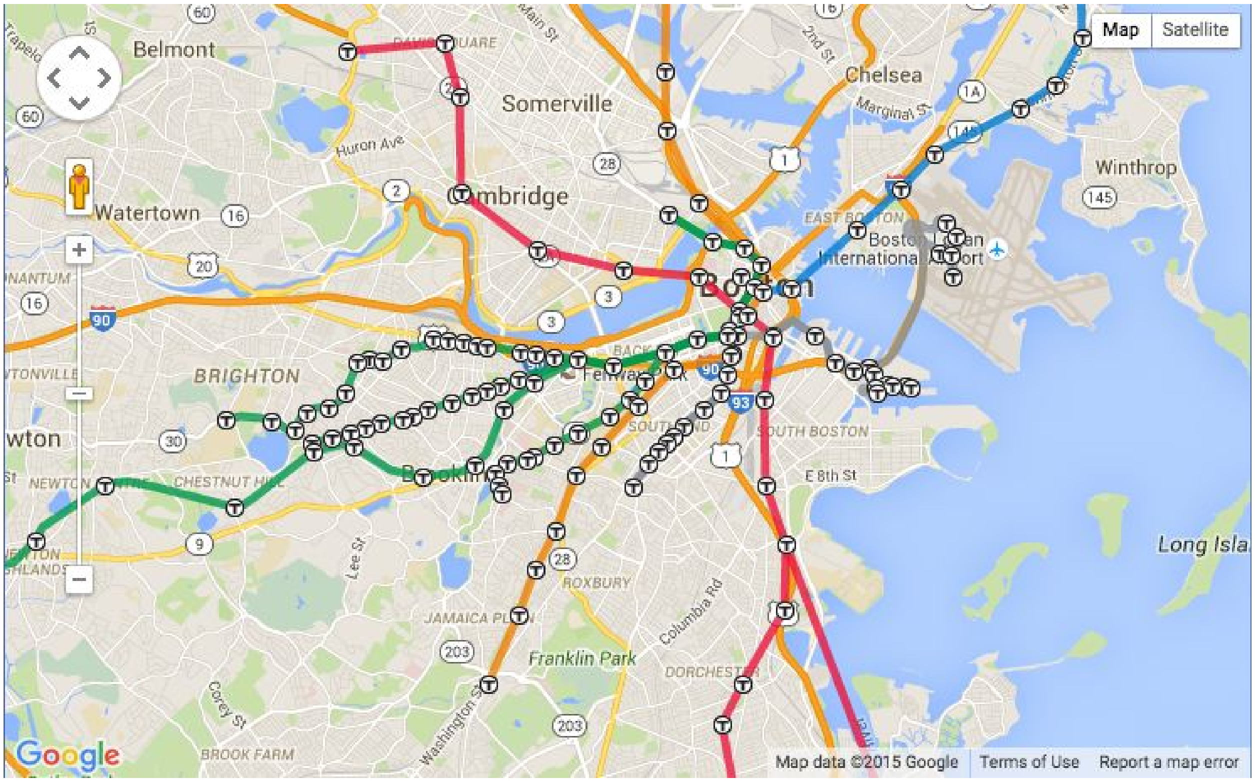
CM 523 Project Proposal: Custom MBTA

Purpose

Allow for users to customize the base MBTA map to their specifications. Users can add and delete stops to their liking. At start-up, users are presented with the original MBTA map plotted on top of a map of Boston. See figure below for an idea of how the starting map will appear.



To add a stop, users can click a position on the map. A popup will appear at the corresponding latitude-longitude location where the user clicked and ask the user for information about the new stop (name, line color, previous stop, next stop). Once the user finishes specifying this information and submits, the map is redrawn with the new stop at the position the user clicked and the correct connections to its previous and next stops. User-created stops can be made visually distinct from existing MBTA stops by using a different color or shape for the stop marker.

To delete a stop, users can click on any stop, user-created or existing, and a pop-up will prompt users if they wish to delete or edit the stop. Once the user confirms the deletion of the stop, the map will be redrawn with the stop removed.

Audience

Boston-based public transit enthusiasts.

Technology

We plan to use the Leaflet.js JavaScript library to represent the map on which users will plan their ideal MBTA map, in conjunction with HTML and CSS. Leaflet.js has a number of useful functions relevant to our project, including map layers, markers (for marking existing and user-created stops), popups (for user interaction when creating and removing stops), and support for drawing paths on the map (for drawing different-colored MBTA lines), to name a few. We propose a doubly-linked list data structure to store data about the stops and their relation to each other, where each stop is represented with a node and contains associate metadata (name, color of line, latitude and longitude, etc), with pointers to its previous and next stops. As we are working heavily with geographical data, we will be doing a great deal of reading, writing, and updating of geoJSON files.

We are new to developing for mobile before so we anticipate some struggles when designing which touch gestures map to what functions in our application. We are also unclear as to how Leaflet.js works with mobile, though its website does state that iOS and Android are supported.

~~We plan to use the Leaflet.js library in conjunction with HTML and CSS to build our application. The backend logic will be programmed in a to-be-determined programming language and some JavaScript. Stops are marked on the map by Leaflet-supported markers, which can be plotted all at once using a geoJSON object that stores data (name, location, previous stop, next stop) for each stop. Creation of new stops on desktop is done by clicking on a position on the map, which will make a call to Leaflet's mouseEventToLatLng(<MouseEvent> *ev*). The mouseEventToLatLng will return a latitude-longitude position where the user clicked. A pop up appears asking the user to supply some information about the stop, namely its name and previous stop (so that it can be placed on the correct line). The name, previous stop, and latitude-longitude information is then written as a new stop to the geoJSON, and then the map is redrawn with the new stop.~~