Mass TRAC

A Tool that displays Real-Time
Arrival Countdowns for the MBTA

Dan Brook
Austin Veino
Jivani Cassar
Project Proposal
91.462 GUI Programming 2
Professor Heines
February 7, 2013

Table of Contents

Project Goal 2
Program Features 2
The Desktop and Tablet Site 2
The Smartphone Interface 5
Common Features 6
Required Software Components 7
Existing Software as a Model 7
Interface Components to Write 8
Intended Audience 9
Issues to Address for Completed Product 9
Development Timeline 10
Release Functionality Requirements 12
Future Features (Time Permitting) 12
Influential Long Term Goals 13

Project Goal

We'll create a web application that provides public transit users in the Greater Boston area a way to see predicted arrival times before leaving on their trips, allowing them to avoid long waits on station platforms.

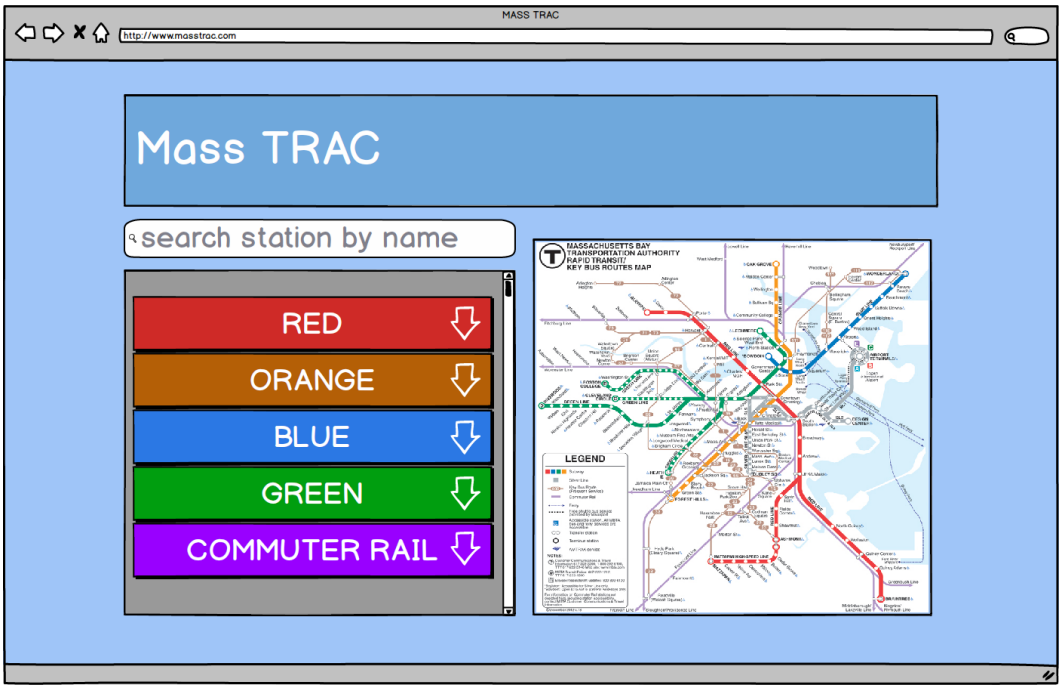
Program Features

Mass TRAC will use the experimental arrival prediction data made available by the Massachusetts Bay Transportation Authority (herein referred to as the MBTA) to show transit riders consistently formatted human-readable arrival predictions (for both Commuter Rail and Rapid Transit lines), service alerts, and other related information all from one place. Users will also be able to customize how the information is displayed as well as use the interface most appropriate for their platform, be it a smartphone, desktop, and if time permits, a standalone kiosk mode that businesses near a particular station could use to display arrival data for their transit-using customers.

The Desktop and Tablet Site

When users first navigate to the site, their device will be detected and they'll be directed to the appropriate version (mobile or full-featured) of the site. Afterward, the station selection screen will launch and allow them to select for which station to display arrival information. Station selection will be accomplished either by entering the station name and having an auto-complete suggestion list appear, or

by using the accordion-style selection below the text entry widget. The focus of this web application is for people familiar with the MBTA, so making the auto-complete widget prominently displayed will be ideal, perhaps also using cookies to hold favorite stations. Should the user wish to use the accordion style selection, the system map will reflect what line has been expanded in the accordion widget.



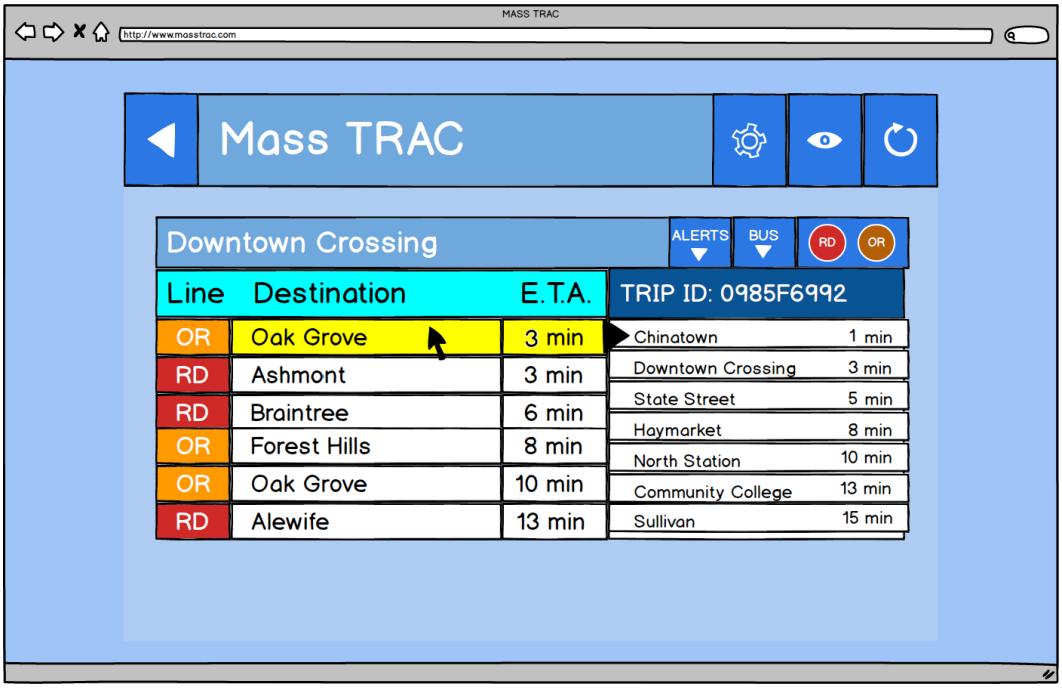
After a selection has been made, a new page will load which contains the name of the station, drop-downs for any alerts for the station (such as an elevator failure) or the entire line the station is on (like a 10 minute delay impacting the Red Line), and available bus connections at the station will be provided. If more than one line services a chosen station, icons representing toggle buttons will appear to turn

arrival information on or off for those lines. The arrival table generated from the data provided by the MBTA will appear below that, displaying the status of incoming trains like the line, destination, notes, and a countdown timer.

According to the MBTA Developer Guide, trains are generally either *scheduled* (not yet running) *predicted* (a train is running and a predicted time based on its velocity is available), and *delayed* (a train is stopped and delayed like at a signal). Our display will use textual and other visual clues to represent these different modes, like countdowns on gray boxes for scheduled times.

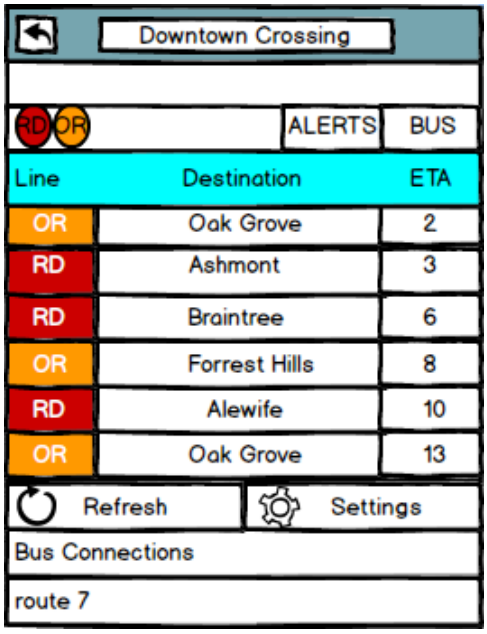
The station page will also provide a way for users to see all the predictions for a given trip instance of the route. By clicking one of the arrival entries on the left, a pop-out will appear that displays all the predictions available for that train. For example, if a person were to board a train at Wellington that arrives in 5 minutes, they could see that that train should reach Downtown Crossing 16 minutes later.

Another piece of the interface will be the heading bar, which will provide a place for a back button as not all devices' browsers have one, and for a tablet this would be an important feature to have, a place for our logo, a settings button to bring up the configuration dialog, a button to hide the title bar and turn on kiosk mode (if we get to implement it), and a refresh button.



The Smartphone Interface

The smartphone interface will be designed to closely mimic the desktop, but there will be several aspects that will set it apart. The main difference will be a smaller title bar, which just provides a back button, the station name, and a refresh button, with the line toggle switches below it. The overall theme of the mobile site will be consistent to make users feel familiar with both versions if they wish to use both. Many of the jQuery UI controls used on the desktop site will not be as desirable on small screens like those on modern smartphones, and data use on mobile devices in general must be limited due to wireless plans that no longer feature unlimited data access.



Common Features

In both the version of our application, users will be able to change how data is displayed to them. The customizable features we've deemed as effective starting points are as follows.

- Changeable expected arrival time format (e.g. "5 min" versus "4:55 pm")
- Sortable arrival entries by:
    - Expected arrival time in ascending order only
    - The line the train is running on, then expected arrival
    - The line the train is running on, then destination, and finally expected arrival
- Select one of the themes available (a darker or lighter version)
- Total number of arrival entries to show

We hope that these common capabilities will provide users with a similar experience no matter their platform of choice, while still respecting the strengths and limitations of each platform.

Required Software Components

The MBTA provides the data for Commuter Rail and Subway arrivals via JSON format, which we've already tested some processing of during feasibility testing. Therefore, most of the work involved with generating our own data structures for display will be done with JavaScript. In order to overcome the same origin policy of AJAX which is needed to retrieve the JSON data, we've created a small PHP script to bring the MBTA JSON data to weblab.cs.uml.edu. When a user requests arrival data, the PHP script will run and check if the stored JSON file is new enough and grab a new copy if it isn't. After this operation, AJAX calls will only be done to weblab.cs.uml.edu, following the same origin restriction.

Existing Software as a Model

Several transit tracking sites and applications already exist, and seeing them run first hand has provided insight about effective interface practices. This is primarily what led us to look at different interfaces for smartphones (simple and quick to use) and desktops or tablets. Unfortunately, many of the more common MBTA tracking utilities use the first version of the data feed which is now deprecated, and others do almost all the processing on the server side and then

serve up static pages based on their cached information. Mass TRAC will be different because it will build the data structures on the users' devices, allowing the interface code to produce information in more flexible ways.

Interface Components to Write

We've determined the following organization to our JavaScript code to interpret the JSON data from the MBTA and organize it focused on the user's selected station. We'll have functions to interpret the subway arrival data and commuter rail arrival data because their arrival data files are organized very differently. Ultimately, however, we want the users to see a standardized output regardless of the line they choose, so these functions will populate the same array. To assure that this data is current, these functions will interact with the server-side PHP code for updating JSON data from the MBTA on weblab.cs.uml.edu.

Another component we'll need to write is one to process Green Line data, because it isn't tracked like the other subway lines. This requires a separate function and data structure to hold estimated average wait times based on the time of day and the officially-published MBTA Green Line schedule.

The third component will specifically process MBTA alerts and service advisories, which are kept in an RSS feed that we'll have to write an interface with. Ideally, we'd only display alerts that impact the line(s) and station being displayed, but there's also the potential that displaying more alerts will be

beneficial to riders transferring elsewhere in the system. However, showing more data like this may cause information overload, and end up being less helpful.

In order to make a single unified interface for station arrivals, we have already created a JavaScript object that contains station names, the lines that stop at each station, any available bus connections, and the geographic coordinates of the station. In the event the system is expanded, adding to this object will make it easy to make site-wide changes with minimal effort.

Intended Audience

Mass TRAC is intended mainly for commuters or casual users who have a working knowledge of the MBTA. Despite the widespread digital signage being deployed recently in subway stations across the MBTA (see below), users of the system don't actually see that data until they've *already entered* the station. With Mass TRAC, users will see if they have time to run a quick errand or grab a coffee before heading out to the platform.

Issues to Address for Completed Product

We've already completed some feasibility tests with processing the JSON data we've stored on weblab and haven't encountered any obstacles. With a proper

PHP backend for controlling the data refreshes, the bulk of the future programming lies with making a usable front-end interface, with jQuery UI features and other organizational methods. Developing the more feature-rich desktop and tablet mode site will be rather approachable, but when it comes to working on the mobile pages, we will have to spend some extra time picking up effective layout techniques, data (bandwidth) saving options, and other principals of mobile user interfaces. Lastly, because the feed is experimental, there are occasionally some strange entries such as:

- Trips that never become predicted even as they are traveling their route,
- Trips that are scheduled but far beyond the 2 hour window the MBTA says they should appear within,
- Trips that go after midnight get an extra hour added to their schedules

As suggested by the MBTA, effectively communicating the experimental nature of their data feeds is crucial, because they are still working on the predictions too.

Development Timeline

Date	Goal	Responsibility
11-Feb	*Project Source Control* We will be constructing our project collaboratively using GitHub source control, and we will have our individual contributions migrated there by this stage.	All
18-Feb	*Initial Arrival Time Processing* This stage includes constructing data tables from JSON data using separate Commuter Rail and Subway functions, and ensuring the PHP code is crafted well. This stage also includes having a simple page that renders a table with this information.	Brook

20-Feb	*Launch Page Construction* For this part, we'll have a simple front page that can do basic station selection which will launch the station arrival page, and display the live arrival data for that station. We will also begin integrating and testing jQuery UI elements on the site.	Veino
22-Feb	*Preliminary Graphics and Logos* By this date, we'll have some solid foundations for displaying graphics and other context sensitive visual elements. This will also include experiments with font sizes and graphics, as well as trying to find a balance between these elements on the pages.	Cassar
28-Feb	*Alpha Status Project* By the alpha release, the application will have a mostly-complete launch page and be able to display arrival predictions at any station using some of the jQuery UI capabilities. Additionally, by this date, we will have some basic functionality regarding the gathering and displaying of MBTA alert RSS feeds.	All
18-Mar	*Mobile Site Experimentation* By this phase, we will have basic interaction and testing on mobile version of the site completed. *Settings Dialog and Configuration Options* Additionally, we will have the ability for visitors to customize the rendering of the arrival times, and save their setting and/or favorite stations using browser cookies.	Assigned as available
25-Mar	*Settle on Final Site Appearance, Interaction* Through this period, we will aggressively try rendering the site on as many platforms as possible to determine if any bugs are introduced due to system differences, as well as to test the graceful degradation of any jQuery UI features we use.	All
4-Apr	*Beta Project and Usability Test* The beta phase will consist of all the features we set out to create, with some potential rough edges that need attention.	All

18-Apr	*Review Beta Test Feedback* We will have the bulk of any fixes or suggestions that are deemed possible within the scope of the application based on the results of the usability tests at least partially implemented on the site. Through this period until 1-May, we will continue polishing the user interaction based on any changes or bugs introduced with changes after the beta tests. This will be considered the feature freeze date.	All
1-May	*Completed Project for Presentation*	All

Release Functionality Requirements

At release time, we expect a fully functional desktop and tablet version of the pages with jQuery UI features to make the interface very easy to navigate without creating unnecessarily-long scrolling pages. The data presentation should be customizable to users' liking through a configuration dialog with cookies to save the selections. Station selection should be accomplished using auto-completion or the accordion style navigation. The smartphone interface will also be completed but with less visual effects than the full site, and faster access to favorite stations.

Future Features (Time Permitting)

Using an HTML 5 canvas widget could be a powerful way to show operating trains for an entire line by mapping latitude and longitude to a scaled-down geographically accurate map. We feel this could consume considerable effort to program, especially if displaying multiple lines simultaneously.

Another feature that could be helpful for users is rendering the departure boards at North Station and South Station. These boards are used because many Commuter Rail lines converge at these terminal stations and the trains don't pass through like they do at all the other stations. The MBTA says this data was released but we've been unable to find these feeds thus far.

Influential Long Term Goals

At release time, our application is going to be primarily commuter-oriented, which leaves people unfamiliar with the system without any mechanism to get directions from one starting point to the next. Google Maps can provide directions via public transit in the Greater Boston area because of the available General Transit Feed Specification (GTFS) data, but we would need to spend additional time working to mesh their API and our arrival predictions in a graceful manner.

Finally, according to the MBTA, while the Green Line currently lacks tracking capabilities due to its lack of Automatic Train Control (ATC), the system is expected to be upgraded in 2015. We've already integrated Green Line stops into the station data and will provide average expected wait times based on the time of day and the officially published Green Line schedule in the interim.