Iteration 1 Report: CTFastTrak Live Map and Routes

Software Engineering CTFastTrak API Project

Bryan Davis, Robert Rotaru, Matthew Shafran, Brian Tardiff

Functionality

During the first iteration of the CTFastrak Application development, our team was able to realize many of the goals that we identified during its first iteration planning. The application has realized all of the user stories that were designated for its first iteration of development. The functionality of the system currently includes many basic functions that will be built upon in future iterations. The first and most important is having a visible map of the CTFastrak routes displayed for the user. This map contains bus and bus stop information, which contains both static and dynamic data. The bus stop information is static information that is generated from the GTFS data that represents the different stops and routes. It is represented on the visible map by blue dots, which can be clicked or hovered-over for more information pertaining to the specified stop. The bus information is real-time, dynamic data that is generated from the JSON data that the application receives from the CTFastrak API. This data refreshes every 30 seconds and shows where the specified bus currently resides on the visible map. The individual buses are represented on the map by small bus icons which are free to move around the map as the bus travels in real time. The traveler is able to view this data and note the different buses on the map with their various stops and routes. This gives the traveler two abilities: first, to get a bird's eye view of the map and bus routes as a whole, and second, the ability to find specific information about individual buses and stops.

Implemented User Stories

1. As a traveler, I want to get up-to-date CTFastrak information from the system so that I can be better able to plan a route. I want to be able to see buses and bus stops on the map interface and see details for each.
Pre-condition: BRT System is online and reachable. Traveler is connected to the system.
Post-condition: Traveler is able to view current information.

5. As a traveler, I want to see the most current bus terminal information, so I can plan my route better.
Pre-condition: System is online and reachable. User is connected to the system.
Post-condition: User is able to view most current bus terminal information

6. As a traveler, I want to be able to look at nearby bus terminal locations and locate the closest one as well as the next upcoming bus arrival time for that location so that I can figure out which bus terminal to go to.
Pre-condition: BRT System is online and reachable. Traveler is connected to the system.
Post-condition: The traveler is able to receive information about bus stop locations and projected arrival times.

11. As the JSON interface, I want to get live bus data from the CTFastrak API and display it to the user interface so that the traveler is able to view the buses live on the map as they move or have their condition/information updated.

Pre-condition: JSON API is available. Interface is able to access JSON data. There is bus information available.
Post-condition: System queries and receives live bus status information. This information is displayed in the user interface.

12a. As the JSON interface, I want to accept live bus route information from the CTFastrak API and display it to the user interface so that the traveler is able to view stops and live bus route conditions on the map.
Pre-condition: JSON API is available. Interface is able to access JSON data. There is bus route information available.
Post-condition: System queries and receives live bus route status information. This information is displayed in the user interface.

12b. As the GTFS interface, I want to accept live bus terminal information from the CTFastrak API in CSV format and convert it into JSON so that it may be displayed to the user interface and the traveler is able to view stops and live bus terminal conditions on the map.
Pre-condition: GTFS data is available. Interface is able to access GTFS data. There is bus terminal information available.
Post-condition: System receives and converts bus terminal information. This information is displayed in the user interface.

User Story Changes

For user Story 5, instead of getting bus terminal location, we decided to implement the bus route information because the JSON data provided bus route alerts and delays instead of bus terminal alerts. We also decided to break up user story 4 into two stories. One user story will be finding a path, from point A to point B, within a single bus route and the second user story will find a path, from point A to point B, for multiple bus routes. For user stories #11 and #12, instead of GTFS providing live data from CTFastrak, we are using the JSON API to query live data. Similarly, in user stories #8 and #9 we are getting the static data, such as bus terminals, from the GTFS data provided by CTFastrak. Finally, we decided to split up user story 12, 12a is retrieving live bus route information via the JSON API and 12b is converting GTFS bus stop data into JSON for use by our app, both of which we have completed as planned as part of our first iteration.

Lessons Learned

Originally, we were under the impression that the GTFS data would be a feed giving us the most updated information and the JSON data would be static. After working with the data, it turned out that the GTFS data was just a zipped folder of text files containing the bus stop information. The JSON data requires querying the CTFastrak API to get the updated bus and route information.

We also learned that user story 4 (picking a route) could be broken down into two separate user stories. One story would involve calculating the route between two bus stops that are on the same bus route. The second story would involve calculating the route between two bus stops that are on two different bus routes (which would require getting on more than one bus).

We ran into an issue trying to combine the CTFastrak data with the Google Maps API. The CTFastrak API would not work with cross-origin requests. In order to work around this, we had to use a separate API to access the data as a proxy (whateverorigin.org). This allowed us to query the CTFastrak API without having to build our own backend to act as a proxy. We immediately contacted the developers at CTTransit, but did not get a response until the end of the iteration. The issue has been resolved now on their end, but it would have helped if we had contact with someone working closely with the API.

In order to work more efficiently during this iteration, it would have been better if we had done more research into the CTFastrak API, specifically into the JSON and GTFS data. When we had originally started planning the project, we thought the GTFS data would act as a feed, pushing updates to our system. But after working with the data, we realized this assumption was incorrect. This caused us to change some of our iteration plans and edit and break down other user stories, including ones we had worked on.

Remaining User Stories

2. As a traveler, I want to access the application's 'get nearest location' feature and have the system determine the bus location nearest to me so that I can plan my route accordingly.
Pre-condition: BRT System is accessible; Traveler is logged into the system; Locations are available
Post-condition: Traveler is able to retrieve nearest stop location

3. As a traveler, I want to access the application's 'set destination' feature and specify a destination I would like to travel to so that the system can accommodate my travel plans.
Pre-condition: BRT System is accessible; Traveler is logged into the system; Locations are pre-loaded into the system
Post-condition: Traveler is able to set a destination

4a. As a traveler, I want to access the application's 'pick route' feature and select from a list of calculated routes from point A to point B along the same bus line that the

application has generated so that I can pick my preferred route to facilitate my travel needs.
Pre-condition: BRT System is accessible; Traveler is logged into the system; Routes have been calculated based on traveler's input
Post-condition: Traveler is able to pick a route

4b. As a traveler, I want to access the application's 'pick route' feature and select from a list of calculated routes from point A to point B across multiple bus lines that the application has generated so that I can pick my preferred route to facilitate my travel needs.
Pre-condition: BRT System is accessible; Traveler is logged into the system; Routes have been calculated based on traveler's input
Post-condition: Traveler is able to pick a route

7. As a traveler, I want to be able to receive any updated information about the bus arrival times, ensuring that the data shown is as accurate as possible. The notifications will update any important changes (i.e. a delayed bus). This will help be alerted as soon as changes occur and be able to plan a better route.
Pre-condition: System is online and reachable. User is connected to the system. The user is either allowing automatic notifications or requests an update of JSON data.
Post-condition: The user will receive all up to date information from the JSON data, including any changes to route information.

8. As the GTFS data Interface, I want to get the most up to date location of the bus terminals, so when the traveler requests a bus terminal location, I will be able to give accurate and relevant data.
Pre-condition: CTFastTrack provides bus terminal information in GTFS format
Post-condition: Bus terminal data gets sent to user

9. As the GTFS data interface, I want to get the most up to date location of the buses so that I can respond to the traveler's request for bus locations.
Pre-condition: CTFastTrack provides bus information in GTFS format
Post-condition: Bus data gets sent to traveler

10. As the JSON interface, I want to get live event data relayed from the CTFastrak API. I want to have this data as soon as it exists so that the user interface can be updated to alert a traveler via notifications of any route conditions or events
Pre-condition: JSON API is available. Interface is able to access JSON data.
Post-condition: BRT System listens and receives live event and delay information. This information is displayed in the user interface.

User Story Estimates

1	2	3	5	8
#3*	**#2***	**#8***	**#4a***	
		#9*	#4b	
			#7	
			#10	

Next Iteration User Stories (to be implemented)

For the next iteration, we want to begin implementing route calculation features of our app. In order to do this, as a pre-condition, we must first implement user story #2, which implements the functionality of choosing a starting destination for the route, and user story #3, which implements the functionality of choosing an ending destination for the route. For this iteration, we will only consider a simpler case where the two start and end destinations are along the same bus line. We will implement user story #4a, which is to calculate and display a route from the start to the end if those two points happen to be on the same bus line. Additionally, we plan to implement user stories #8 and #9 which make use of the GTFS data to get static information about buses, bus terminals, routes, route schedules, etc. The functionality of the system after this next iteration will include being able to choose a starting destination, an ending destination, and to calculate and display a route between the two, as well as making more static data available (such as bus schedules) for future calculations of whether a user will be able to make it to the next bus in time.