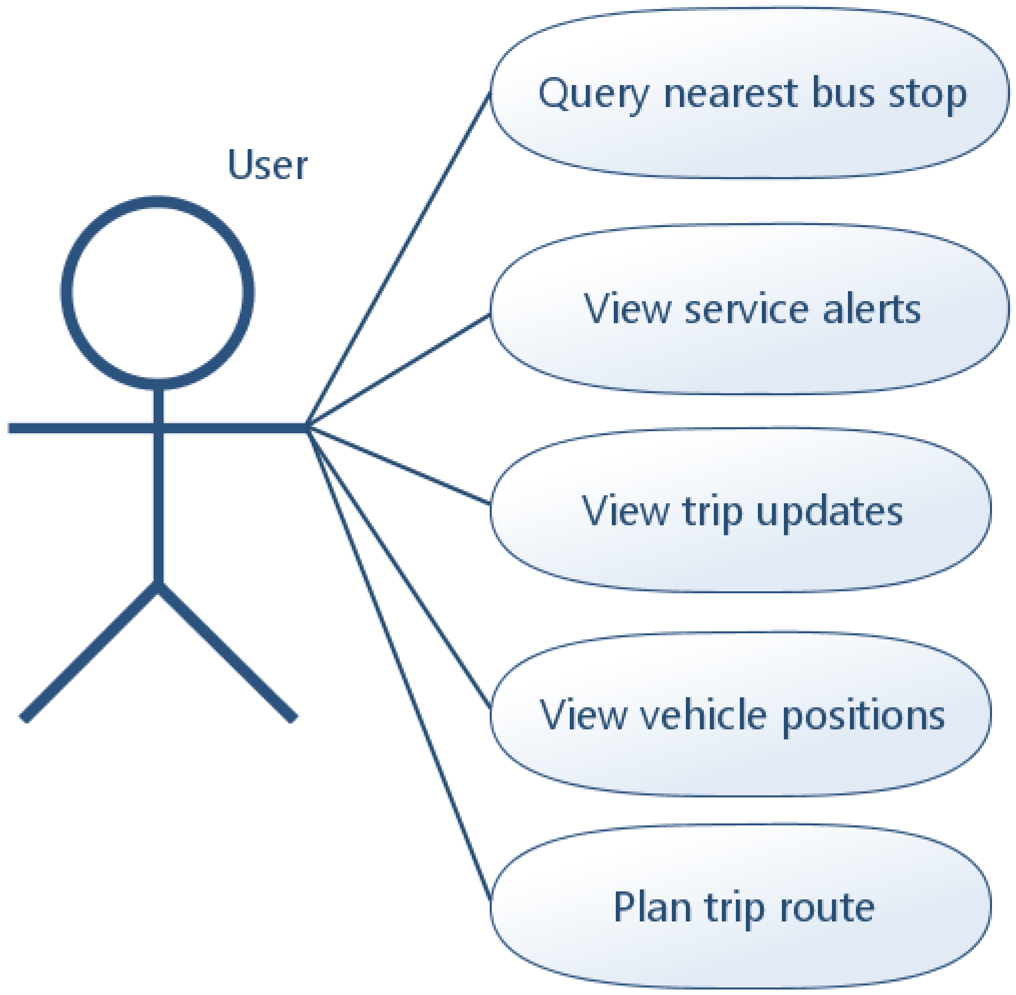
CTFastrak Application

System Requirements

Scrumbledore:
Daniel Bereza – Main Contact / Submission
Mike Sheldon – Documentation
Colin Frink – Deliverable Review
Phil Hoffman – Requirements Check

10/06/16

The CT FastTrak Application will be a Mobile Android Application written with a Java front-end GUI and a JSON backend, the two will complement each other. The backend will parse the users input parameters, information from the Fastrak website and Google maps to perform the necessary Dijkstra's shortest route algorithm. The GUI will have many drop downs with a submit button and map which will parse the users input parameters, being their location and where they want to go and the route calculations will be base solely on this information.



The above use cases for the user actor is the process of having the information presented view three types of streams being trip updates, vehicle positions, service alerts, or a combination of viewing all three of them. The information will be sent as a query to the route decision system and a recommendation will be made to the end user for the closest bus route which will help the user plan the trip route.

1. As a user, I can access the entire CTtransit system statewide, not just by region so that I can access application throughout the entire region.

    - Pre-condition:
      The user must be registered and logged in to the application.

    - Post-condition:
      The user will be able to access data feeds throughout the region, not just in one area.

    - This story will have to be broken down into at least 2 or 3 separate user stories. There could be one user story just having the map extended to cover the entire region for view and a second to actually take into consideration the new larger area in route calculation.

2. As a user, I want multiple recommendations about my fastest arrival, not just one so that I can see multiple routes.

    - Pre-condition:
      The user only has one recommendation as to the fastest route.

    - Post-condition:
      After the user logs in, will have a top three fastest recommendations just like Google maps.

    - This story may need to be decomposed, with one user story relating to the generation of the route, and another relating to how the options are presented and selected.

3. As a user, I want the map display to change brightness and colors, so I can see the application clearly regardless of time of day.

    - Pre-condition:
      The user is viewing the application.

    - Post-condition:

The user is seeing the map in day view, or night view as appropriate to time of day.

- This story shouldn't need to be broken down, as graphical elements will simply need their colors to change based on the device's clock.

4. As a user, I want my route information to continue to be visible even if I lose connection or close the application, so I only need to request my route once.

   - Pre-condition:
     The user loses connection to the data servers after having received a route recommendation.

   - Post-condition:
     The user continues to have access to the existing route.

   - This story may need to be broken down, as handling how the application will function when offline might require a separate user story from ensuring that route information is stored when the application is closed.

5. As a user, I want the application to notify me of incoming buses so that I know my bus is arriving.

   - Pre-condition:
     The user has received a route recommendation.

   - Post-condition:
     The user receives a notification that the bus is arriving at the bus stop.

   - This story might need to be broken down, with one story covering the application's behavior in tracking buses, and another relating to how and when the user is notified.

Nonfunctional Requirements:

1. Security – make sure that people won't be able to hack the fastrak application to see people's location.

2. Backup - If there is a data loss event, backup the system and the archived user information for anyone using the application.

3. Maintainability – To be able to isolate defects and their cause to have system working at peak performance.

4. Portability - Make sure that installed program files can transfer to another computer with the same architecture.

5. Performance - Make sure that there is a short response time for when the user puts in a request for the shortest bus route.

6. Accessibility – Interacting with the application should be intuitive for the average user.

Glossary:

Consistency: are there any requirements conflicts?

Completeness: are all functions required by the customer included?

Implicitness: Domain specialists understand the area so well that they do not think of making the domain requirements explicit.

Prototyping: Using an executable model of the system to check requirements.

Realism: can the requirements be implemented given available budget and technology

Sunsetting: is the practice of shutting down redundant or obsolete business applications while retaining access to the historical data.

Test-case generation: Developing tests for requirements to check testability.

Understandability: Requirements are expressed in the language of the application domain, which is not always understood by software engineers developing the system.

Validity: does the system provide the functions which best support the customer's needs?

Verifiability: can the requirements be checked?