Software Requirements Specification

Shuttle Tracking and Ride Request Service (STARRS)

Version 1.0 approved
Prepared by Will Gross, Zhuofan Zhang, Zilin Chen
CS 397 Group 3
20 Oct 2017

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Will Gross	10/20/17	Creation of Document	1.0
All members	12/12/17	Modifications addressing comments, and additions of use case diagrams.	2.0

List of Figures

1. Introduction

1.1 Purpose

STARRS is a system aimed at improving the situational awareness of Colby security, Colby students, and jitney drivers as it relates to Colby provided transportation. To do this, STARRS will provide security dispatch and students with a live map of the location of the shuttle and jitney, it will provide security administration with usage data, it will provide drivers with a live queue of ride requests, and it will provide students with a live queue of pending requests as well as a way to add a request.

1.2 Document Conventions

Priorities relating to high level requirements are inherited by all sub-requirements.

1.3 Product Scope

This system is intended to streamline the process of using and managing Colby run transportation in preparation for the college's mission to increase community relations and the opening of a new dorm on Main Street. This will be achieved by making a user initiated request process instead of the current system of relaying verbal requests through a dispatcher, auto-logging ride data instead of the current method of manual entry. It will also reduce calls to security by making the schedule for the jiney readily available.

2. Overall Description

2.1 Product Perspective

STARRS is a brand new system to replace the currently used manual methods for using and managing the operations of the Colby shuttle and jitney. The elements of this system include a gps tracker for the shuttle, the phone of the jitney driver to transmit location and allow the driver to interface with the system, and a web based interface for student users and security management.

2.2 Product Functions

STARRS must allow students to see the locations of the shuttle and jitney if in operation, the current operation schedule for the jitney, the request queue for the jitney, and it must allow them to submit and remove a request for a ride. It must allow the driver of the jitney to always know what requests are pending and be able to select from those requests a job to accept. It must also allow security to do all of the functions of a student user, as well as the ability to remove any request from the queue, and the ability to retrieve data from the ride information log.

2.3 User Classes and Characteristics

The three main user classes are the student user, the jitney driver, and the security dispatcher. The student user is the standard Colby student. They expect a fast and simple interface that allows them to predict when their ride will arrive.
The jitney driver requires a simple, non-invasive interface that doesn't distract them from driving, but allows them to manage the service requests.
Security dispatch needs the ability to quickly review and remove a request, as well as the ability to constantly monitor the location and status of the jitney and shuttle and retrieve ride history to analyze for service evaluations.

2.4 Operating Environment

STARRS will mainly operate from a browser as a web based system. In later iterations, it will also interface with native Android and iOS apps.

2.5 Design and Implementation Constraints

The design must be simple, straightforward, and easy to understand as well as easy to expand to a larger fleet. This is important due to the fact that the customer will be responsible for maintaining and updating the software.

2.6 Assumptions and Dependencies

We are dependent on third party gps hardware as well as 3rd party navigation APIs for the success of this project. We are also operating under the assumption that our software will be fully implemented and utilized by the jitney staff and student body; if it is used in conjunction with the current system, it will fail to be useful and provide the promised benefits.

3. System Features

3.1 Logging In

3.1.1 Objective, Actor, and Priority

The feature allows the user/dispatcher/driver, as the actor, to log in the system in order to perform any activities. This feature has **High** priority.

3.1.2 Use Case

3.1.2.1 Assumptions

The system is connected to the myColby system, so that users can use their myColby accounts for identification.

3.1.2.2 Preconditions

Network connections are not interrupted.

3.1.2.3 Basic Flow of Events

1. The user opens the webpage/app.
2. The system checks the login status of the user.
3. The user clicks “Login”.
4. The interface redirects to the myColby page and prompts the user to enter the username and password with the myColby system.
5. The interface acquires the user information and jumps to the information page.

3.1.2.4 Alternative Flow of Events

2a. The system finds out that the user is already logged in.

2a1. The system jumps to step 5.

3a. The Internet connection is not available.

3a1. The system pops up the error message “Internet connection currently unavailable. Please check your settings or contact your service provider, and then try again.”

5a. The user selects to log out.

5a1. The system processes logout through myColby.

5a2. Jump to step 1.

5b. The user is a dispatcher.

5b1. The system directs the user to the dispatcher page.

5c. The user is a driver.

5c1. The system directs the user to the driver’s main page.

3.1.2.5 Postconditions

If the user isn’t a driver or the dispatcher, then the interface displays the main page, where information including driver shift hours (hours of operation), Jitney status, Jitney tracking map, and also buttons linking to other pages such as the current Jitney queue page, the Jitney program webpage, the Security telephone number, etc.
If the user is the dispatcher, then the interface also provides access to the database where data for requested rides are stored, and a notification system to communicate with drivers.

If the user is a driver, then the interface only includes an embedded map for the driver to look up directions, a current queue, a link to the ongoing request list (where the dispatcher can remove a pending request), and a link to the notification system.

3.1.3 Functional Requirements

3.1.3.1 The system shall be able to connect to myColby for login information validation.
3.1.3.2 The system shall provide different main page interfaces to users with different administrative levels.

3.2 Tracking the Jitney Location

3.2.1 Objective, Actor, and Priority

The feature allows the user, as the actor, to see the real-time location of Jitney on the map. This feature has **High** priority.

3.2.2 Use Case

3.2.2.1 Assumptions

The real-time GPS information is sent and stored on the server.

3.2.2.2 Preconditions

Network connections are not interrupted.

3.2.2.3 Basic Flow of Events

1. The user goes to the webpage / opens the app where the real-time maps is displayed.
2. The map interface checks the current state of Jitney, and receives real-time GPS data.
3. The map marks the location, and repeatedly goes to step 2 after 0.5s, until the user switches to other interfaces.

3.2.2.4 Alternative Flow of Events

2a. Jitney is not currently operating.
    2a1. Interface pops message “Jitney is not currently running”.
2b. Jitney is currently operating, but GPS information is not received, or the location is out of range (Waterville, ME and Oakland, ME).
    2b1. Interface displays message “Currently unable to track Jitney”, and goes to step 2.

3.2.2.5 Postconditions

The interface displays the map, with Jitney’s location marked, and updates 2 times per second, until the user quits the display by quitting the app or switching to another interface.

3.2.3 Functional Requirements

3.2.3.1 The interface shall display the map as an embedded Google Maps API.
3.2.3.2 The system shall check the condition of Jitney and GPS data transmission internally.
3.2.3.3 The interface shall notify the user about error that occurs in connection, Jitney condition, or GPS information transmission.

3.3 Creating A Pickup Request

3.3.1 Objective, Actor, and Priority

The feature allows the user, as the actor, to place a pickup request in the system, with information on pickup location and number of passengers. The dispatcher can also be the actor, but the event flow won't have any difference. This feature has **High** priority.

3.3.2 Use Case

3.3.2.1 Assumptions

The user can submit the GPS location, or has Internet access.
The system can recognize the identity of the user, and match it with pickup requests sent by the user.

3.3.2.2 Preconditions

Network connections are not interrupted.

3.3.2.3 Basic Flow of Events

1. The user enters the interface for pickup requests.
2. The user selects a location from the Google Maps interface.
3. The interface asks the user to confirm if the selected location is the desired pickup location.
4. The user confirms that the selected location is the pickup location.
5. The system prompts the user to enter description of location.
6. The user either enters description of location and save description, or skip the step.
7. The system prompts the user to select end location.
8. The user selects a location from the Google Maps interface.
9. The interface asks for confirmation of destination.
10. The user confirms the destination.
11. The interface asks for number of passengers.
12. The user chooses the number of passengers from the popped-up list.
13. The interface notifies the user that the request is successfully placed, and the system uploads the request internally to the server.
14. The interface switches to the page that shows the current queue of the system.

3.3.2.4 Alternative Flow of Events

1a. Jitney is not currently operating.

1a1. Interface pops message “Jitney is not currently running”, and returns to home page.

1b. The user already has one pending request stored in the queue.

1b1. The interface pops up error message “You already have one request in queue”, and jumps to the queue page.

2a. The user presses “Cancel”. (This applies to any step from 2 to 12)

2a1. Interface discards all entered information, and returns to home page.

2b. The user presses “current location”.

2b1. The system gets the current location of the user, and goes to step 3.

2b2. If the system fails to get the current location, the interface pops message “Failed to obtain current location”, and goes to step 2.

2c. The selected location is out of service range (Waterville and Oakland, ME).

2c1. Interface pops up message “Pickup location out of service range”.

2c2. Goes to step 2.

4b. The user refuses to confirm.

4b1. Goes to step 2.

8b. The user presses one of the “popular locations” on the interface.

8b1. The system retrieves the location that matches the selected location.

8b2. Goes to step 9.

10b. The user refuses to confirm.

10b1. Goes to step 7.

3.3.2.5 Postconditions

The interface displays the page showing the current queue (or goes back to the main screen if the user cancels at any time), and the user’s entry would be saved and uploaded to the system.

3.3.3 Functional Requirements

3.3.3.1 The interface shall display the map as an embedded Google Maps API.

3.3.3.2 The system shall check the condition of Jitney and GPS data transmission internally.

3.3.3.3 The interface shall notify the user about error that occurs in connection, Jitney condition, or GPS information transmission.

3.3.3.4 User shall be able to make a pick up request, and the interface shall confirm the location with the user.

3.4 Cancelling A Pickup Request

3.4.1 Objective, Actor, and Priority

The feature allows the user/dispatcher, as the actor, to recall a pickup request in the system earlier placed by the same user. This feature has **Medium** priority.

3.4.2 Use Case

3.4.2.1 Assumptions

The system can recognize the identity of the user, and match it with pickup requests sent by the user.

3.4.2.2 Preconditions

Network connections are not interrupted.
The system is able to recognize the user's identity, and give corresponding ability to manage the queue.
The user is at the queue page.

3.4.2.3 Basic Flow of Events

1. When the screen is first loaded, the system takes in the identity of the user, and decides that the user should only be able to manage their own requests.
2. The screen displays the current queue, and shows a "Cancel" button only for rides requested by the user.
3. The user clicks the "Cancel" button next to one request.
4. The interface asks the user to confirm cancellation.
5. The user confirms cancellation.
6. The system removes all requests made by the user from the queue.

3.4.2.4 Alternative Flow of Events

1a. The system can't verify the user's identity.
    1a1. Interface pops error message "We can't confirm your identity right now. Please consider re-logging-in or calling the Security office".
    1a2. Interface goes to step 2.
1b. The system identifies that the user's identity is the dispatcher.
    1b1. The screen displays the current queue, and a "Cancel" button is shown for each requested ride in the queue.
2a. The user doesn't cancel anything before exiting the current page.
    2a1. The interface follows the user's instruction, leaving no change on the queue.
5a. The user cancels the cancellation.
    5a1. Interface goes to step 2.

3.4.2.5 Postconditions

The interface displays the updated queue, where the user's previous request is deleted if the user has chosen to.

3.4.3 Functional Requirements

3.4.3.1 The interface shall obtain and display the current request queue.
3.4.3.2 The system shall be able to remove a request in the queue.

3.5 Responding to A Pickup Request

3.5.1 Objective, Actor, and Priority

The feature allows the driver, as the actor, to take on a pickup request in the system. This feature has **High** priority.

3.5.2 Use Case

3.5.2.1 Assumptions

Network connections are not interrupted.
The system can recognize the identity of the user.
The driver has their own screen for picking up requests from queues.
The system constantly updates the current GPS location of the driver's device.

3.5.2.2 Preconditions

The driver is at the driver's page (with identity validated).
There are pending pickup requests in queue.
The current status of Jitney is "Idle".

3.5.2.3 Basic Flow of Events

1. The driver selects one request from the queue at his/her own discretion.
2. The system asks the driver to confirm the choice.
3. The driver confirms the choice.
4. The system removes the selected request from queue, and adds it to the list of ongoing requests.
5. The system records the current time (i.e. departure time), the start location of pickup, and the destination of pickup and uploads the information to database. The system also takes note on the expected number of passengers, and change the status of Jitney into "Heading for pickup".
6. When the driver reaches the pickup location, the driver selects the correct request from the list of ongoing requests with button "Pick up".
7. The system asks the driver to confirm that the passengers are already picked up.
8. The driver confirms that the pickup is done by confirming or re-entering the actual number of passengers.
9. The system records the current time (i.e. pickup time) and the number of passengers. It also updates the status into "Heading for destination".
10. When the driver reaches the destination, he/she selects the correct request from the ongoing request list with button "Drop off".
11. The system asks the driver to confirm the choice.
12. The driver confirms the choice.
13. The system records the current time and request number, and then removes the selected request from the list of ongoing requests, and changes the Jitney status to "Idle".

3.5.2.4 Alternative Flow of Events

1a. The sum of the number of expected passengers for the selected request and the number of passengers on the ongoing list of requests exceeds the maximum number of passengers the Jitney allows.

1a1. The interface pops up error message “Total number of passengers will exceed maximum capacity. Try to finish the current requests first, or select another request.”

1a2. Jump to step 1.

1b. The driver chooses to cancel selection.

1b1. The interface returns to the previous page (i.e. if Jitney status is)

3a. The driver refuses to confirm.

3a1. Jump to step 1.

5a. The system is unable to upload the information successfully.

5a1. The interface notifies the driver to manually record the relevant information, namely, the current time, the pickup time, the time when arriving at destination, the number of passengers, and the pickup location and destination.

6a. The driver hasn’t updated the status of a request in the ongoing list within 30 minutes.

6a1. The interface pops a notification when the driver opens the interface the next time, asking the driver to confirm if this request is already dealt with or not.

6a2. The driver confirms the condition of the request, and the system goes to step 8, 11, or 6, depending on the driver’s selection.

6a2a. The driver confirms that the user behind the request is picked up, and the system goes to step 8.

6a2b. The driver confirms that the user behind the request has reached destination, and the system goes to step 11.

6a2c. The driver confirms that the request is actually still pending, and the user is not picked up. The system goes to step 6 and updates the timer for this request.

6b. The passenger doesn’t show up at the pickup location.

6b1. The driver cancels the corresponding request from the ongoing list.

6b2. The interface asks the driver to confirm the cancellation.

6b3. The driver confirms cancellation.

6b4. The system records the relevant information about the cancelled request and uploads it to the server.

6b5. The system removes the cancelled request.

6b6. Go to step 13.

6c. The driver presses “Carpool” to deal with another pickup request simultaneously.

6c1. Go to step 1.

8a. The driver refuses to confirm.

8a1. Jump to step 6.

8b. The driver enters a different number of passengers.
8b1. The system updates the number of passengers for the request.
8b2. Jump to step 9.
9a. There are requests in the ongoing list that are not picked up yet.
9a1. The system keeps the status “Heading for pickup”, and executes the rest before going to step 10.
12a. The driver refuses to confirm.
12a1. Jump to step 10.
13a. The ongoing list is not empty when the selected request is removed.
13a1. If there are requests not picked up yet in the list, then the system updates the status to “Heading for pickup” and goes to step 6.
13a2. If all the remaining requests in the list are picked up, then the system updates the status to “Heading for destination” and goes to step 10.

3.5.2.5 Postconditions

The selected request is removed from the queue and the system, but all relevant information such as the actual time for pickup and arrival, the number of passengers, the pickup locations, and the destination, is uploaded to the data server.

3.5.3 Functional Requirements

3.5.3.1 The interface shall be able to obtain and maintain the list of ongoing requests.
3.5.3.2 The system shall be able to fetch a request by its request number.
3.5.3.3 The system shall be able to record relevant information about each request.
3.5.3.4 The system shall record the time since one request is last updated.

3.6 Notifying the Driver

3.6.1 Objective, Actor, and Priority

The feature allows the dispatcher, as the actor, to communicate with the driver on the shift with a message system (the role can also be reversed: The driver can communicate with the dispatcher). This feature has **Low** priority.

3.6.2 Use Case

3.6.2.1 Assumptions

The system can recognize the identity of the user, and match it with pickup requests sent by the user.
The system has storage space for messages.

3.6.2.2 Preconditions

Network connections are not interrupted.
The user is at the main page.

3.6.2.3 Basic Flow of Events

1. The user clicks the link to the message system.
2. The system recognizes the user as the dispatcher, and prompts the user to enter text for notification.
3. The user enters the content of the notification.
4. The user clicks “Send to the current driver: <Driver name>”.
5. The message gets stored in the system, labelled as “Unread”.

3.6.2.4 Alternative Flow of Events

2a. The system finds out that there are currently no drivers on shift.

2a1. The interface pops the error message “No driver is currently working. Please contact the targeted receiver(s) through email.”

2a2. The interface jumps to the main page.

4a. The input text is too long.

4a1. The screen displays the current length of input, and pops an error message “The maximum length of input is xxx characters”.

4b. The user clicks on “Cancel”.

4b1. The interface pops up “Discard the text?”

4b2. The user selects “Yes”. (Otherwise, go back to step 4b)

4b3. The system removes the input text, and jumps to the home page.

5a. The message isn’t successfully uploaded to server.

5a1. Interface pops error message “Failed to send message. Please check your Internet connection, or contact the driver with other methods”.

3.6.2.5 Postconditions

The dispatcher’s message is stored on server, with status “Unread”, and the recipient being the driver currently on shift.

3.6.3 Functional Requirements

3.6.3.1 The system shall have access to a message system.

3.7 Receiving and Replying Notification

3.7.1 Objective, Actor, and Priority

The feature allows the driver, as the actor, to communicate with the dispatcher on the shift with a message system (the role can also be reversed: The dispatcher can communicate with the driver, using use case 3.6). This feature has **Low** priority.

3.7.2 Use Case

3.7.2.1 Assumptions

The system can recognize the identity of the user, and match it with pickup requests sent by the user.

The system has storage space for messages.

3.7.2.2 Preconditions

Network connections are not interrupted.
The user is at the main page.

3.7.2.3 Basic Flow of Events

1. The user opens the interface.
2. The system recognizes the user as the driver, and searches for any new message with timestamp within 3 days, labelled as “Unread”, and recipient the same as the current driver.
3. The interface informs the user that they have new unread messages.
4. The user clicks on the button to the message system from main page.
5. The system searches for all messages sent to the user within the last 3 days, and displays all of them in the order from new to old.
6. The user clicks on the check button of a message.
7. The selected message is marked as “Read” in the server.
8. The user goes to the notification system to reply, following the use case in 3.6.

3.7.2.4 Alternative Flow of Events

2a. The system doesn’t find any new message.
2a1. Jump to step 4.

3.7.2.5 Postconditions

All recent messages sent to the user are displayed, and the user can choose to reply.

3.7.3 Functional Requirements

3.7.3.1 The system shall have access to a message system.

3.8 Tracking the Shuttle Location

3.8.1 Objective, Actor, and Priority

The feature allows the user, as the actor, to see the real-time location of the shuttle on the map. This feature has **Medium** priority.

3.8.2 Use Case

3.8.2.1 Assumptions

A GPS device is installed on the shuttle, and the real-time GPS information is sent and stored on the server.

3.8.2.2 Preconditions

Network connections are not interrupted.

3.8.2.3 Basic Flow of Events

1. The user goes to the webpage / opens the app where the real-time maps is displayed.
2. The map interface checks the current state of the shuttle, and receives real-time GPS data.
3. The map marks the location, and repeatedly goes to step 2 after 0.5s, until the user switches to other interfaces.

3.8.2.4 Alternative Flow of Events

2a. The system checks the current time and finds out that the shuttle is not currently operating.

2a1. Interface pops message “Shuttle is not currently running”.

2b. Shuttle is currently operating, but GPS information is not received, or the location is out of range (Waterville, ME and Oakland, ME).

2b1. Interface displays message “Currently unable to track shuttle”, and goes to step 2.

3.8.2.5 Postconditions

The interface displays the map, with shuttle’s location marked, and updates 2 times per second, until the user quits the display by quitting the app or switching to another interface.

3.8.3 Functional Requirements

3.8.3.1 The interface shall display the map as an embedded Google Maps API.
3.8.3.2 The system shall check the condition of shuttle and GPS data transmission internally.
3.8.3.3 The interface shall notify the user about error that occurs in connection, shuttle condition, or GPS information transmission.
3.8.3.4 When GPS signal is not available, the interface shall display estimation on shuttle location according to current time.

4. External Interface Requirements

4.1 User Interfaces

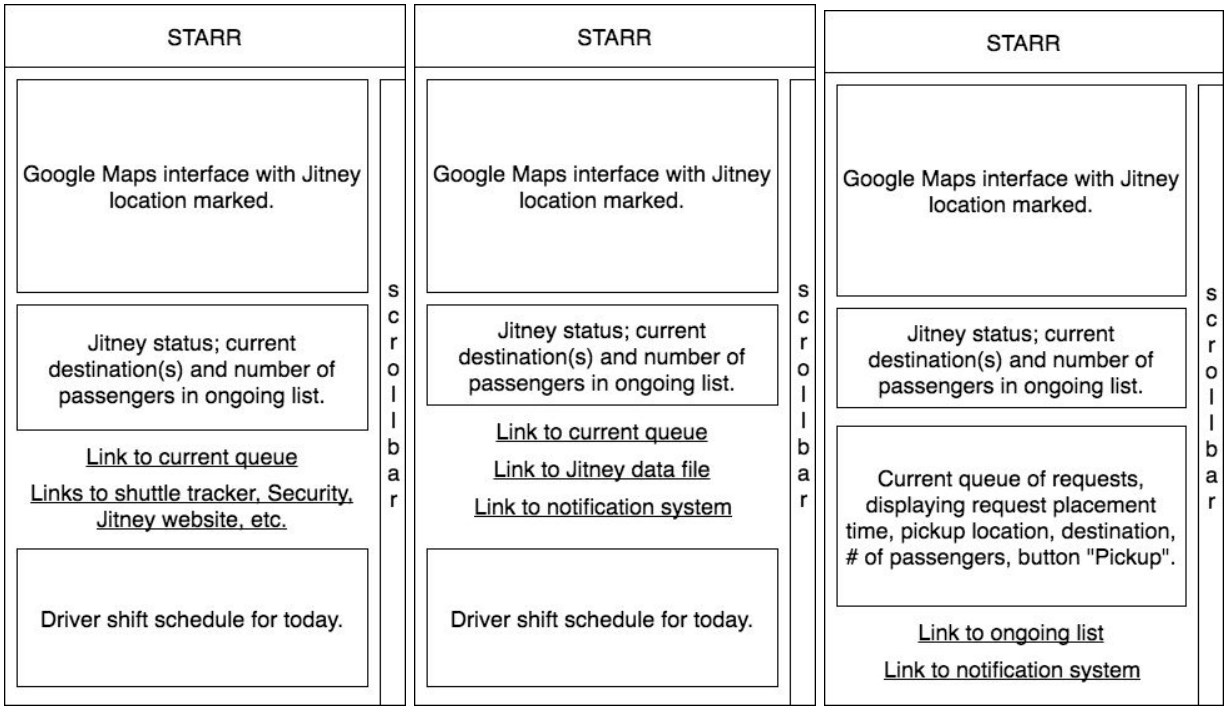

Figure 4.1.1 Main page for general users (left), dispatcher (middle), and drivers (right).

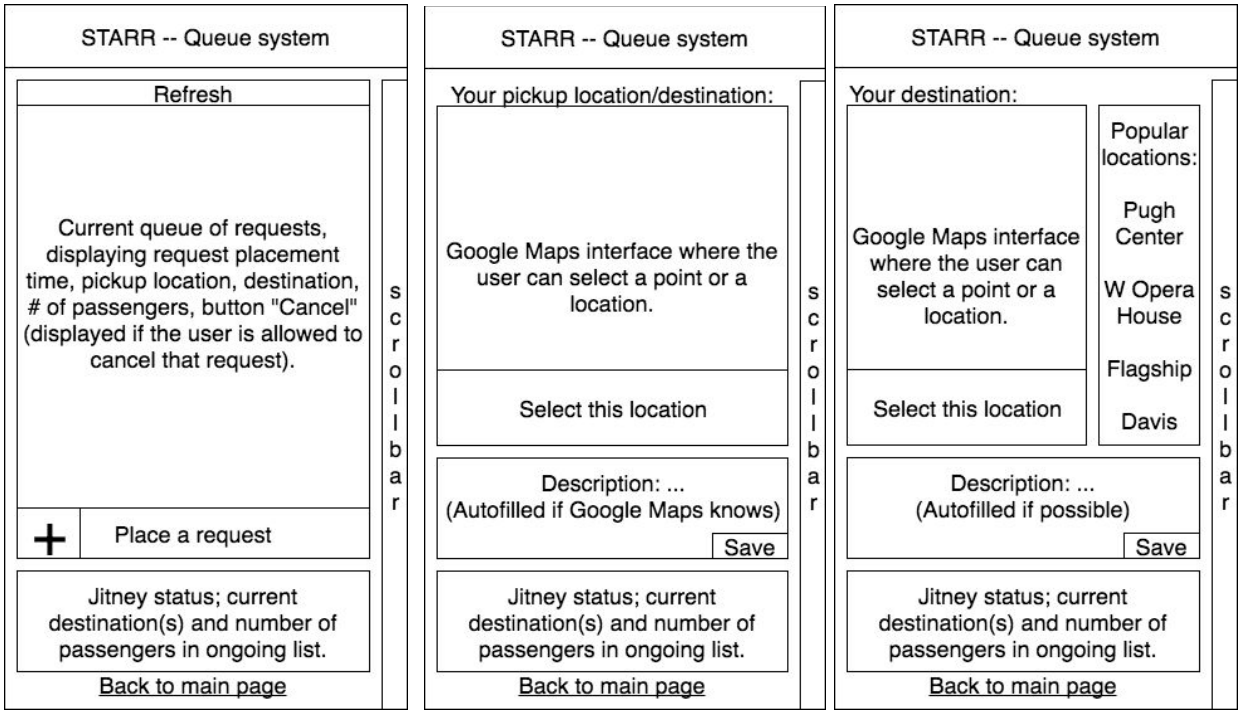

Figure 4.1.2 Current queue page. **Figure 4.1.3** Pickup request page (3.3.2.3.5, 3.3.2.3.8)

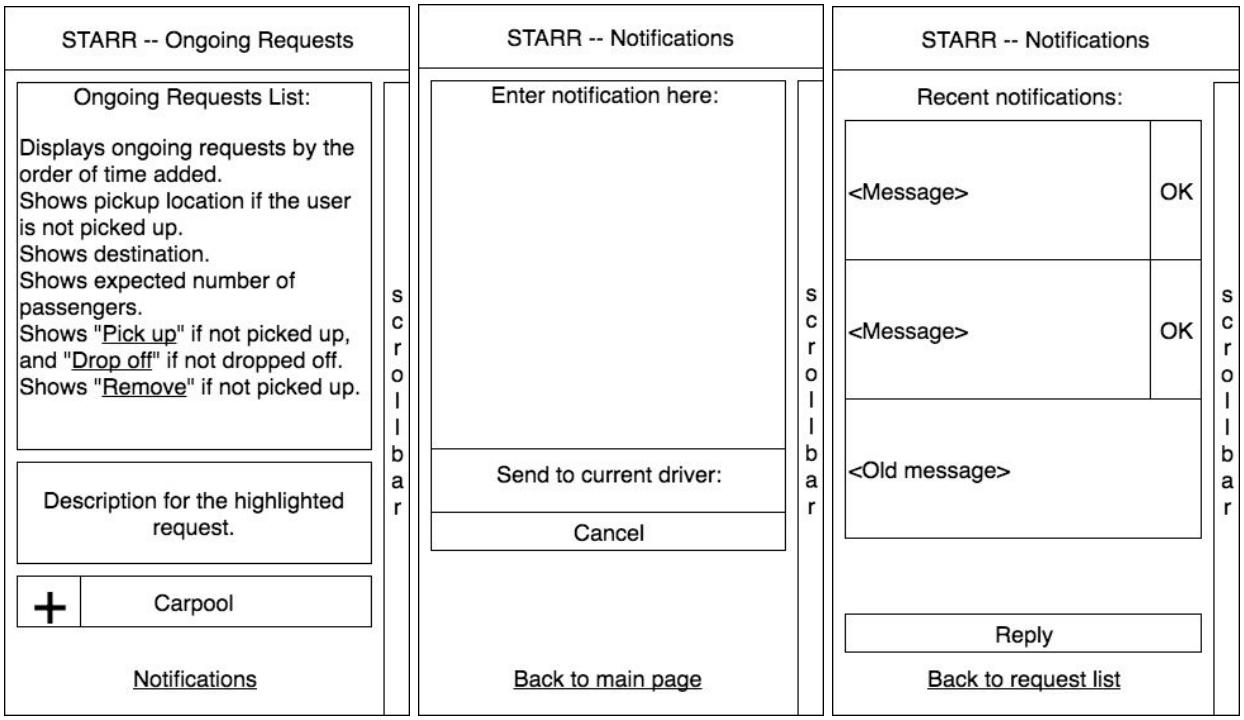

Figure 4.1.4 Ongoing request list page. **Figure 4.1.5** Notification page (3.6.2.3, 3.7.2.3)

The function of each part in the interface figures can be found in the use cases. Besides the depicted interface demonstrations, the system shall also be able to pop notifications from the web page / app, ask for confirmations by creating a smaller window on top of the existing display, and include drop-down lists.

4.2 Hardware Interfaces

The interface would require the ability to access and send GPS data from the phone app.

4.3 Software Interfaces

The app shall have two versions, one for Android and one for iOS.
The system shall be able to load on a webpage inside a browser, with the web server recording, processing, and transmitting data.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 The system shall be interactive among jitney drivers, users, and the security.

5.1.2 The shuttle tracking page shall support at least 5000 users at a time, which covers all students and faculties at Colby.
5.1.3 Any delay that is involved in the interaction process must be less than 3 seconds, so there are only acceptable delays in the action-response of the system.
5.1.4 When users try to enter the webpage, there might be delay because of unsuccessful connection to the internet or incorrect login information.
5.1.5 When security attempts to shut down the jitney requesting system, there should be no delay.

5.2 Safety Requirements

5.2.1 Because our system is a web page based system, the possible loss for users will be the hacking of their colby accounts. Since we will link our website to the myColby page, the security of the webpage as well as student accounts will be take care by the Community Department at Colby College.

5.2.2 Secondly, we will ensure that any requesting and responding information transmission will be transmitted to the server safely and unchanged, so there will not be ambiguous or incorrect communications among students and jitney drivers.

5.3 Security Requirements

5.3.1 The most important security concern for users is the security of their login accounts. Since we will manage to link our website to the myColby webpage, we are less concerned about such an issue. The myColby page is well designed, and such security concerns will be managed by the Community Department that will ensure the safety of information of student's Colby accounts.

5.3.2 Furthermore, it is possible that the GPS used for the shuttle or jitney tracking may be damaged from the outside or stolen. We must reach an agreement among the shuttle drivers and jitney drivers about the importance of protecting the GPS from potential damage. The shuttle drivers or at least the jitney drivers should report the condition of the GPS both before and after their shifts, so if the GPS is damaged on purpose or lost, we know who will be responsible for it.

5.3.3 Last, we will also provide additional access to the database to the security, and the access to the database should at no means be available to other users like students or jitney drivers.

5.4 Software Quality Attributes

5.4.1 The information shown on the website should be correct and updated frequently.

5.4.2 The website will not be available to users when there is an internet disruption. But if users are trying to send information when the disruption occurs, then the information can be present for verification after the internet is successfully connected.
5.4.3 In our design of the system, we will implement different functionalities in separated components. Different functionalities can include, for example, logging, validation, and requesting. And the functionalities may be reused.
5.4.4 For the user inputs on the website, we will validate them for length, format and type in order to prevent malicious inputs. And we will encode outputs that are displayed to users.
We will plan regular testing to our system during the development life cycles.

5.5 Business Rules

5.5.1 Students or customers should be able to see the shuttle's schedule and current location on the website all the time.
5.5.2 Students or customers should be able to see the scheduled time and driver for jitney all the time.
5.5.3 Students or customers should be able to see the availability of the jitney.
5.5.4 Students or customers should be able to request jitney when jitney is available.
5.5.5 Students or customers should be able to enter the location where they want to be picked up.
5.5.6 Students or customers should be able to cancel request.
5.5.7 Students or customers should be able to see the estimated time for the jitney arrival.
5.5.8 Jitney drivers should be able to see customers' request.
5.5.9 Jitney drivers should be able to accepting request.
5.5.10 Jitney drivers should be able to confirm pickup.
5.5.11 Jitney drivers should be able to enter the condition for the GPS.
5.5.12 Jitney drivers should be able to enter their time shift on the website.
5.5.13 The security should be able to access the database.
5.5.14 The security should be able to edit information on the website.
5.5.15 The security should be able to activate the jitney service.
5.5.16 The security should be able to shut down the jitney service.
5.5.17 The security should be able to sending request to jitney.
5.5.18 The security should be able to canceling request for jitney.

Appendix A: Analysis Models and Use Case Diagrams

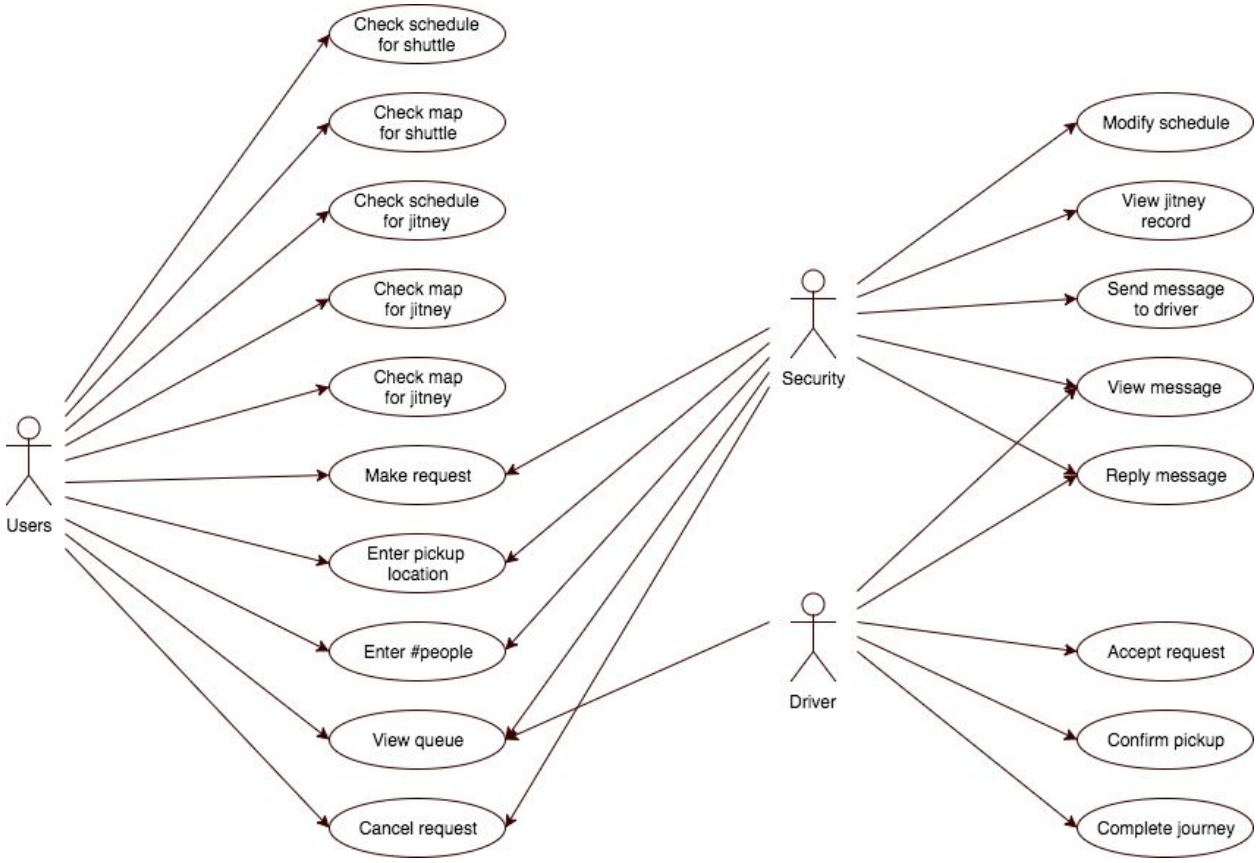

Fig A.1. High level use case diagram

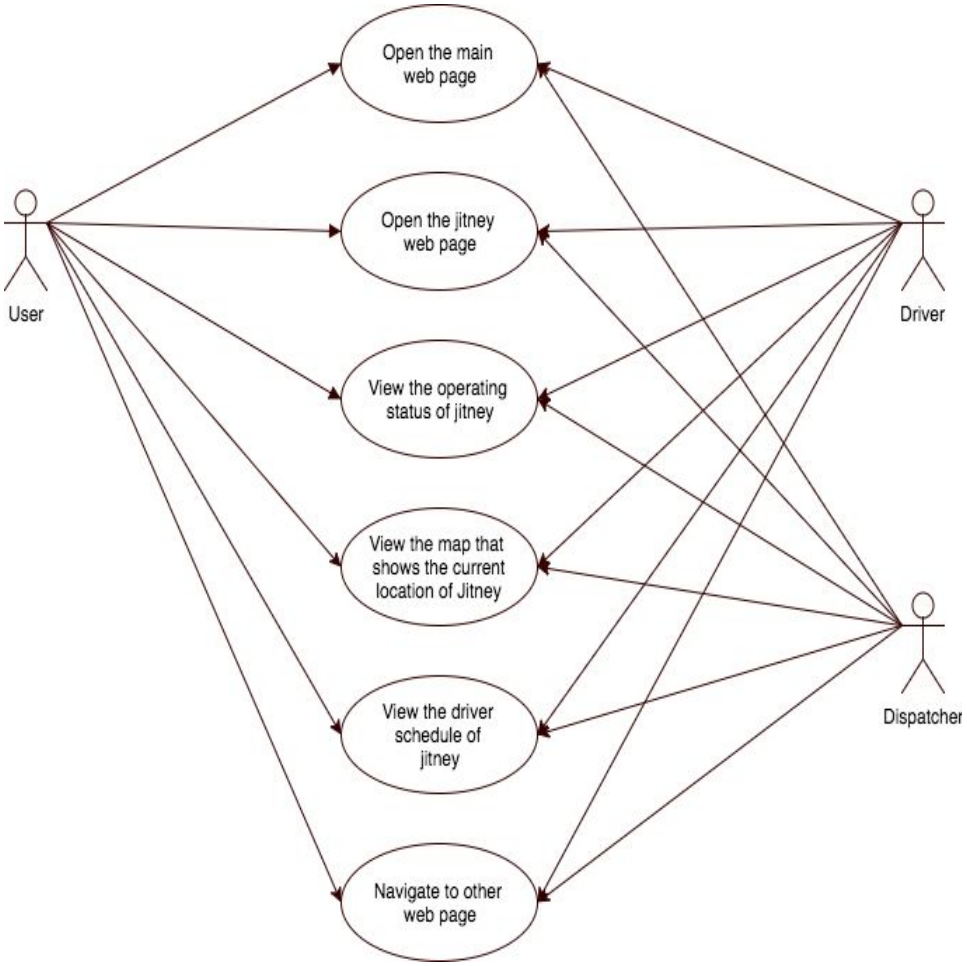

Fig A.2. Use case diagram for tracking jitney location.

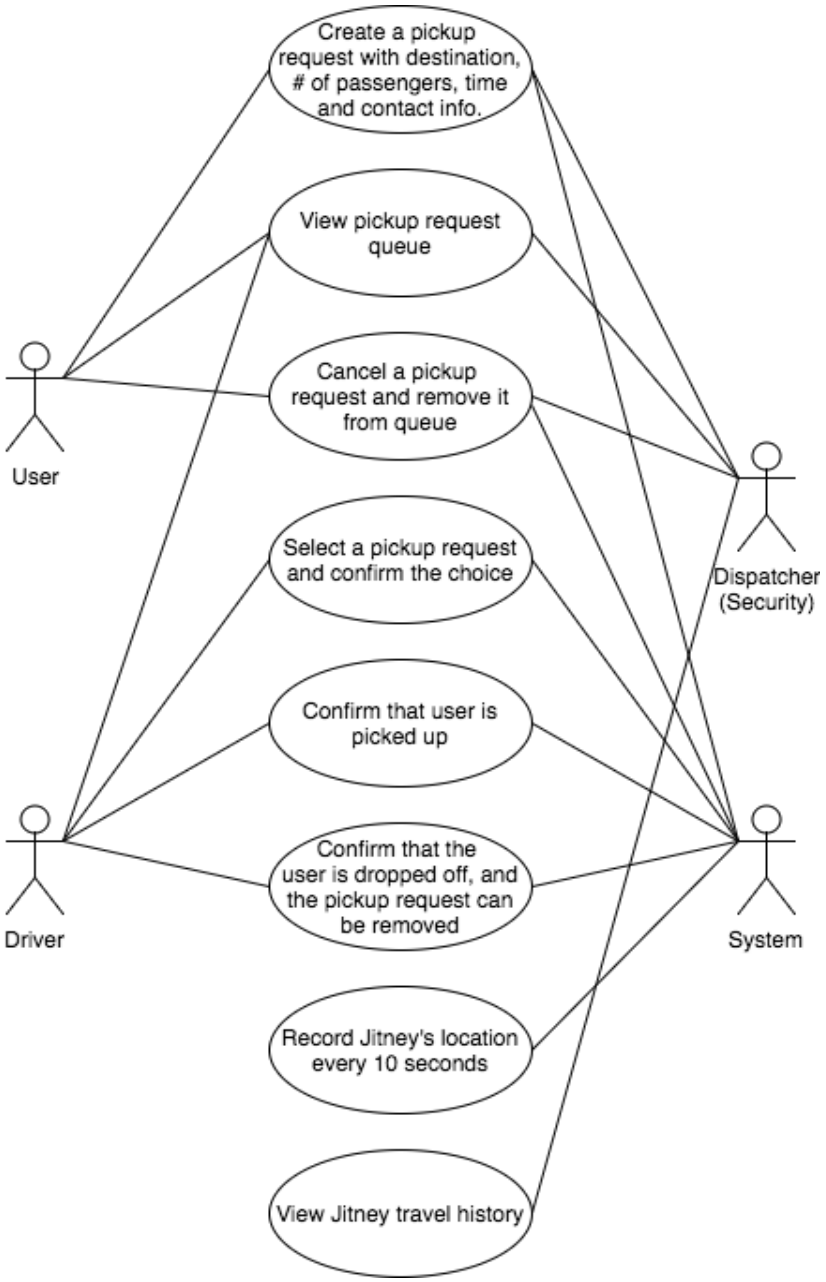

Fig. A.3 Use case diagram for creating/removing, selecting, and processing pickup requests.

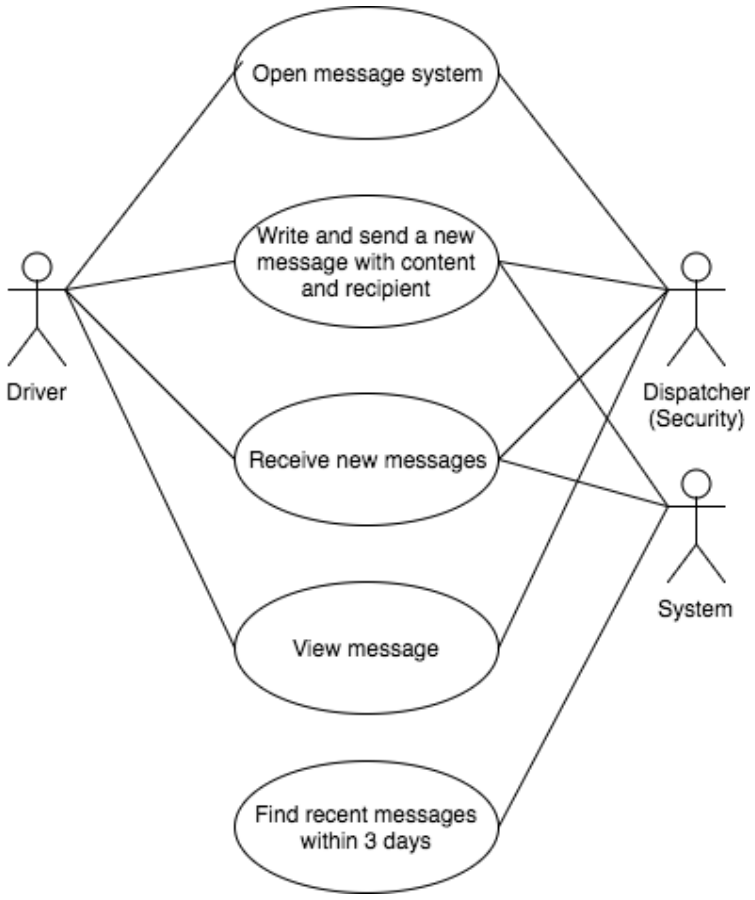

Fig. A.4 Use case diagram for message notification system.