Software Engineering

RU Bus Management System

Report 1

Group #11

Yaocheng Tong
Yufeng Lin
Viraj Patel
Minghao Qin
Yuhai Zhang
Lu Jin
Haowei Li
Aaron Hu

Instructor: Prof. Ivan Marsic
Teaching Assistant: Parsa Hosseini
Graders: Aymen Al-Saadi, George Koubbe

Github Page

Table of Contents

Individual Contributions Breakdown

Everyone currently has equal contribution

Section 1. Customer Statement of Requirements (CSR)

A. Problem Statement.

BACKGROUND:

Rutgers is becoming more diverse because there are many new students matriculating each year. At the same time, the school bus requirements are facing challenges due to high pedestrian traffic. In the past few years, Rutgers has been vulnerable to a cyber attack due to a poor implementation of the bus scheduling system. While it has been worked on, it can still be improved to increase reliability, functionality, and user friendliness. The core of the issue is that the current system that is in place is unable to account for external factors such as traffic due to an overflow of pedestrians during rush hour time slots between popular class periods at various hotspots like the student centers, Scott Hall, and Hill Center. Therefore, having a improved version of school bus manage system seems to be necessary and critical.

PROBLEM DIAGNOSIS:

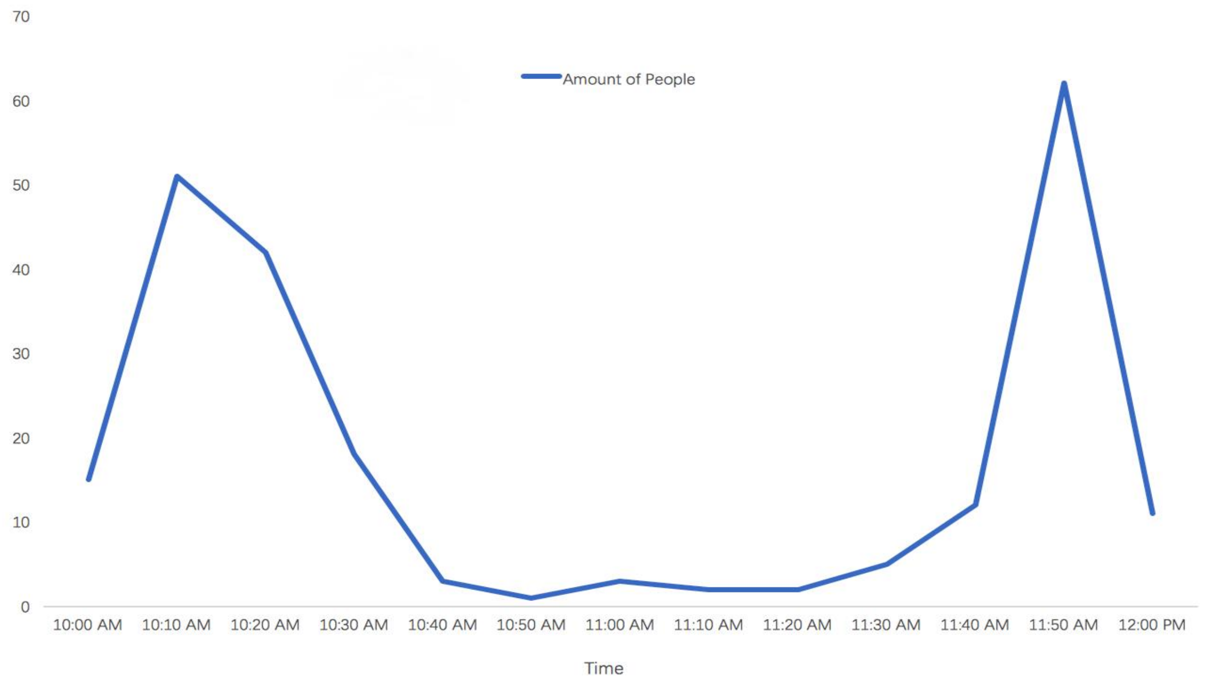
The current school transportation system takes quite a bit of work for the school to manage and the university needs to put forth more effort in order to improve the flow of bus operations. We could take surveys of Rutgers students to identify and analyze the current problems confronting us or garner some suggestions for improvements to the system.

Many of the students use the MyRutgers mobile app to check when the next bus is arriving, but due to some specific time slots such as noon, evening, after-class, and exam periods, many popular bus stops such as Student Activities Center at College Avenue Campus, Hill Center at Busch campus, and Livingston Plaza at Livingston campus, face unexpected pedestrian traffic. As a result, some students are unable to get on the buses on time even after waiting for a long time. This scenario shows a relatively commonplace issue that may need a better solution to resolve. Therefore, this app is our plan to provide students with a better platform to reflect their needs. How can the data reflect the level of needs? If there is a student arriving at the Hill center and taking “H” bus back to College Avenue Campus, he/she can tap the “H” option below “Hill center” and the app will automatically count it by adding one. A student may only click once for their preferred route within a 20 minute timespan. When the number of requirements reaches a peak, it can be determined as a urgent and the system will send notifications to the bus drivers and reduce the time gap between two successive buses. Ideally, this would at least reduce the delay between buses by 50%.. Also, students can leave comments on the app on how they feel about the buses and the app can collect data then send it to the bus management office for improvement.

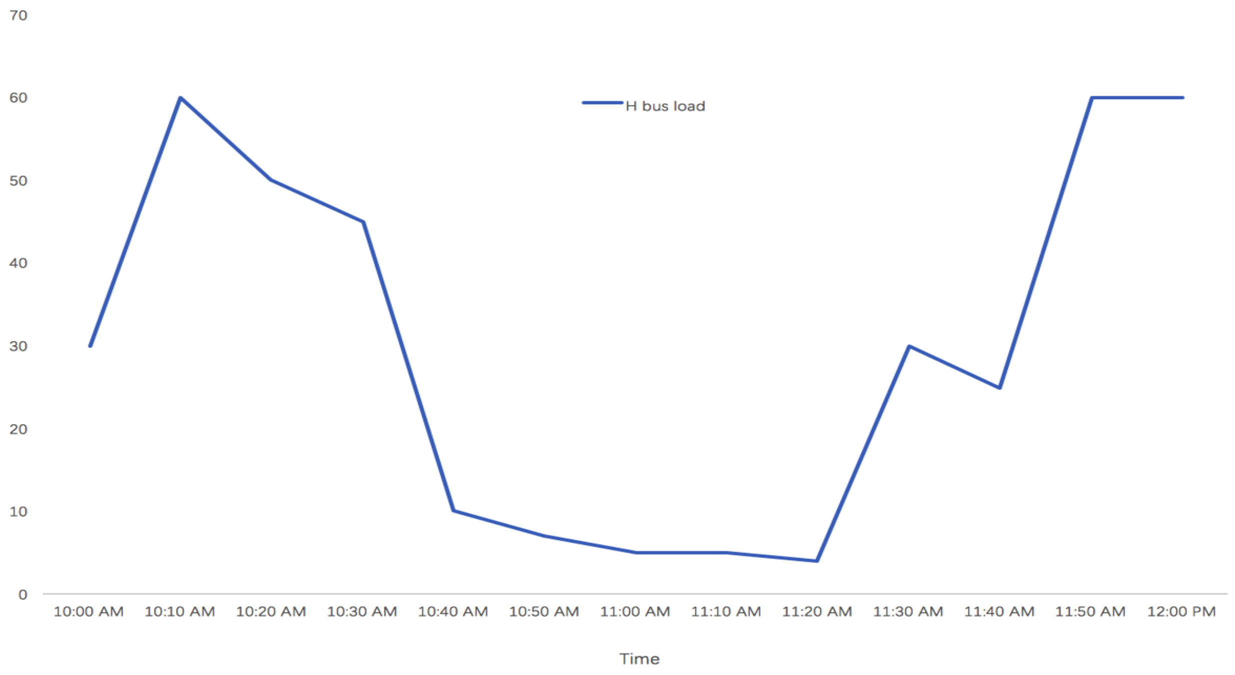
From our observations, there are two main issues with the current system and several common scenarios exist that justify the need for a new school bus management system. First, there is the problem of lack of peak-hours transportation capability. Rutgers has a massive student body and at a specific time, the bus stops will have hundreds of student waiting for campus buses. Since the buses will be overcrowded, a good portion of waiting students can not board upon arrival. Those students who are left behind at the station have only left the options of waiting for the next bus, which could take a significant amount of time depending on other factors. There are even situations in which students may miss multiple consecutive buses due to the increasing crowding at stops. This results in many students being tardy for class and a cyclical degradation of bussing services until the crowds are slowly dispersed over time.

During the peak hours, buses are usually loaded to capacity with as many students as they can hold, which can bring up bus overloading and safety concerns. Whenever such a situation occurs, not a single student can get on or off the bus easily, further increasing delays. Another issue to a lesser degree, the bus will be much heavier than usual and consequently have reduced control, resulting in increased chances of a traffic accident. To observe these relatively common scenario, we have recorded some pertinent data and plotted it in the following charts.

The number of people waiting at the Hill Center bus stop on the Busch Campus between 10 AM to 12 PM, a common rush hour time slot. (Data recorded on January 17th, 2018)



The number of passengers on the H-bus spotted at the Hill Center bus station between 10 AM to 12 PM (rough estimation). (Data recorded on January 17th, 2018)



Parsing through these two data sets, we can see that between 10:10 AM and 11:50 AM, the buses are unable to handle the large influx of bystanders. Students remained at the bus stop and continued to wait for the next bus causing possible lateness to their classes. The H-Buses were loaded to capacity with passengers between 10:10 AM and 11:50 AM. Some relevant information on the vehicle used for the H route from the manufacturer's website:

H-Bus type: ENC AXESS
ENC AXESS specifications:

TECHNICAL SPECIFICATIONS	35′ MODEL	40′ MODEL
GVWR	43,420 lbs.*	43,420 lbs.*
Body Length	35′ 11.5″	40′ 11.5″
Body Width	102″	102″
Wheelbase	215″	275″
Seating	Up to 35	Up to 43

The currently used model is designed for 35 seats, but during peak hours it usually loaded up to 50-60 passengers. As long as it loaded to such numbers, the bus will be much heavier and more difficult to handle. Due to this, the H-bus will have higher risks of traffic accidents on the way to College Avenue.

Another issue found within the current system is an inefficient use of resources during off-hours (i.e. not between class scheduling blocks). By controlling the intervals of the arrival times of two adjacents buses and reducing their variation, the peak load during rush hour can be significantly reduced while off-hour transport would see minimal impact. During normal class times, Rutgers bus stops have many less students waiting because most students are in class. However, the frequency of bus routes are exactly the same as during peak-hours, so generally only a few passengers are transported per bus (sometimes not even a single student at all). Looking back at the previously mentioned data sets, between 10AM and 12PM, Hill center has an H-bus arriving every 9 minutes, but between 10:40 AM and 11:20 AM only a few passengers are aboard.

Another scenario in which the inefficiency of the current system is evident is when a bus is broken down on the road (some emergency issue) or cancellation of bus service. In both cases, there will be a large gap in service that cannot be filled due to the inflexibility of the current implementation. Students who are waiting at the bus stop for a long time have method of being notified about current conditions other than an estimated arrival time, leading to an understandably frustrating and unintuitive setup. Furthermore, when bus service are up and running again, the scheduling will not be in sync with what it was before the disruption in service, causing further issues later on.

The primary function of the bus system improvements should allow students and the bus control center that monitors bus activity to account for external factors and maintain flexibility in such cases in order to reduce the waiting time and reduce service disruption to a minimum. It would also allocate resources more evenly during peak hours to increase efficiency without increasing the number of total routes per business day.

PROPOSED SOLUTION:

We are planning to tackle the issue of data collection head on by collecting rough pedestrian traffic numbers at various points in time at hotspots like Hill Center. By plotting the data, we can see the general trend of crowding throughout the day and consequently the time periods in which buses routes are required at shorter intervals. Section 3 will go over how this will be achieved in greater detail. The below figure shows a simplified visualisation of the I/O workflow of the proposed software.

Benchmark goals by which we will be judging the functionality of our software will include:

- Ease wait time for bystanders at bus stops
- Reduce wait time between bus arrivals during peak hours
- Notification system for events that may affect bus services
- Visualization of current traffic available to users
- A method of bus dispatch depending on transportation needs
- A dynamic scheduling system in which bus drivers will know when peak hours are occuring

VALUE:

Successful implementation of the before-mentioned system will result in more efficient use of the currently allocated resources with reduced overhead. Buses will be running more frequently during busy periods and less frequently during off hours. As a result, students can plan their daily schedules with more leeway and increase productivity as well as academic performance.

B. Glossary of Terms

- *Application(APP)* - main program that the user will be interacting with; the program will be designed to meet the needs of the many students at Rutgers
- *Schedule* - buses will have a separate schedule for regular class days, weekends, events, and holidays
- *Bus driver* - person driving the bus, taking directions from the manager/administrator
- *Manager/administrator* - person(s) who will manage and coordinate the bus drivers and the bus allocations for each route
- *Student* - undergraduate or graduate students who will take Rutgers buses
- *Database* - server that will contain user data, bus routes, peak-hours
- *Mobile application* - software for Android OS phones that allow users to access the bus management system
- *Mobile application user* - someone who is using the mobile app
- *Peak-hours* - times of the day when more traffic is expected on the road
- *Off-hours* - times of the day when less traffic is expected on the road
- *GPS(Tracking Service)* - (Global Positioning System) devices/services to track the location of a bus
- *API* - application programming interface
- *NextBus API* - API that can output tracking data of Rutgers buses
- *Firebase* - Google cloud database and calculation services for applications
- *Mobile Interface* - interface that can interact with mobile users
- *Web Interface* - an interface that can interact with web users

Section 2. System Requirements

A. Enumerated Functional Requirements

Identifier	Manager User Story	Size
ST-1	As a passenger, I can log into the application with my NetID and Password.	2 pts
ST-2	As a passenger, I can find all types of campus buses in menu selection. I can select at most two types of buses I want in the application.	3 pts
ST-3	As a passenger, I will be asked by application about whether I allow application to find my current location or not.	3 pts
ST-4	As a passenger, my current bus stop will be automatically notified if I agree application can access my current location data. I can select my current bus stop if the application made mistake on my current bus stop.	6 pts
ST-5	As a passenger, I can be notified next bus arriving time after I selected some types of buses I want to take.	4 pts
ST-6	As a passenger, I can give feedback on selecting if I am successfully get on the bus on time.	2 pts
ST-7	As a bus driver, I am able to receive alerts on what time I should begin my route according to a preset schedule	5 pts
ST-8	As a bus driver, I am able to find the fastest way when I receive alerts.	2 pts
ST-9	As a bus driver, I am able to receive alerts on which campus I should begin a route on.	2 pts

ST-10	As a bus driver, I am able to send information if I can not finish schedule and managers can reschedule the driver.	4 pts
ST-11	As a bus driver, I am able to receive messages from the manager.	3 pts
ST-12	As a manager, I can know the exact amount of buses needed to solve the overcrowding	6 pts
ST-13	As a manager, I can know which bus stop needs extra buses	4 pts
ST-14	As a manager, I can manage specific drivers operating specific buses	2 pts
ST-15	As a manager, I will receive the notification which reminds me the upcoming activity and adjust the extra buses for demanded stations	6 pts
ST-16	As a manager, I can know when to dispatch the buses that the bus drivers can reach the destination in the required time	4 pts
ST-17	As a manager, I will be able to give solutions to any bus that runs into mechanical issues, if they should occur.	1 pts
ST-18	As a manager, I can edit my previous announcements so that I can amend any errors that mistakenly assigned to bus drivers	3 pts
ST-19	As a bus driver, I am able to see notifications on traffic or special events	2 pts
ST-20	Aa a manager, I can use my credentials to log in to the web interface to access the bus management system.	2 pts

B. Enumerated Nonfunctional Requirements

Identifier	Manager User Story	Size
ST-21	As a passenger, I can change my password by clicking the link in login page to transfer to Rutgers NetID management website.	1 pts
ST-22	As a passenger, I can select my destination of prefered bus route.	2 pts
ST-23	As a passenger, I want to be able to have contact information of Rutgers police to keep the safety.	0.5 pts
ST-24	As a passenger, I can give feedback of whether bus is crowded or not.	0.5 pts
ST-25	As a passenger, I need announcements when buses routes are being changed.	1 pts
ST-26	As a passenger, I want to be able to see some announcements (road closed, traffic accident, holidays etc), since I could know any emergency issue and aware any maintenance around Rutgers Campus, so I can arrange my schedule.	4 pts
ST-27	As a passenger, I can find the current bus dispatch rate.	0.5 pts
ST-28	As a passenger, I can uninstall the app if I want.	0.5 pts
ST-29	As a bus driver, I am able to know some special activities according to a schedule.	2 pts
ST-30	As a bus driver, I am able to get extract paid if I need to work longer because of the special activities.	1 pts
ST-31	As a bus driver, I am able to schedule the time	2 pts

	that I can work for special activities.	
ST-32	As a manager, I know how to communicate with all the bus drivers	0.5 pts

C. On-Screen Appearance Requirements

Identifier	Manager User Story	Size
ST-33	As a passenger, I can access rutgers campus map data to check, so that I will have a clearer overview of rutgers building.	1 pts
ST-34	As a passenger, I can use application interface to check information I need.	4 pts.
ST-35	As a manager, I can see the real-time requests from different bus stops.	4 pts
ST-36	As a manager, I can monitor the traffic and thereby schedule buses accordingly (Data Visualization)	2 pts

D. FURS Table

Functionality	Use to record and analyze student demand in order to arrange buses. All users will use this system everyday. Well security by Rutgers log in system
Usability	Clear Map instruction inside the applications and very easy to use,click,send request,check information they need.
Reliability	Application used by student who need bus to take them to the class, manage need to check the current bus demand and driver need instructions of schedule. Every operation process is very simple for system users and cloud-based database management can ensure important data will be used and not leaked.

Performance	Very quick responses of real-time demand from student.Clear bus schedule will be created to driver. NextBus API will used to track buses which will be very useful.
Supportability	Only support Andriod device right now, web application only for manager, and driver only have a sheet of paper which has their schedule on it. Cloud-based server side can created our own API for future uses.

Section 3. Functional Requirements Specification

A. Stakeholders

- Students
- Bus drivers
- Managers

B. Actors and Goals

- Students
    - Initiating Actor
    - Students have the ability to access the mobile Application that is designed for authenticated users. They may have a better chance of getting on their bus earlier.
- Bus driver
    - Participating Actor
    - The goal of bus driver is to follow the dynamic schedule from the Application that has been designed by a manager and provides feedback in terms of how useful and efficient the current system is.
- User Application
    - Participating Actor
    - The goal of User Application is to receive the requests from the user and forward them to dispatch center. The App is designed for optimizing the dispatch rate and dynamically rescheduling the current bus route.
- Manager Application (Web interface)
    - Participating Actor
    - The goal of the Manager Application is to tell both the manager and bus driver when and where extra buses might be required and suggests several options. Database
    - Participating Actor
    - The goal of Database is to tell the manager the most likely predictions and decisions based on the large amount of crowdsourced information (bus routes, time estimates, number of students at each bus stops, requirements for each bus routes, etc.) stored in the database. It will also store the previous user demand data from user Application as well as feedback from users.
- Mapping Service
    - Participating Actor

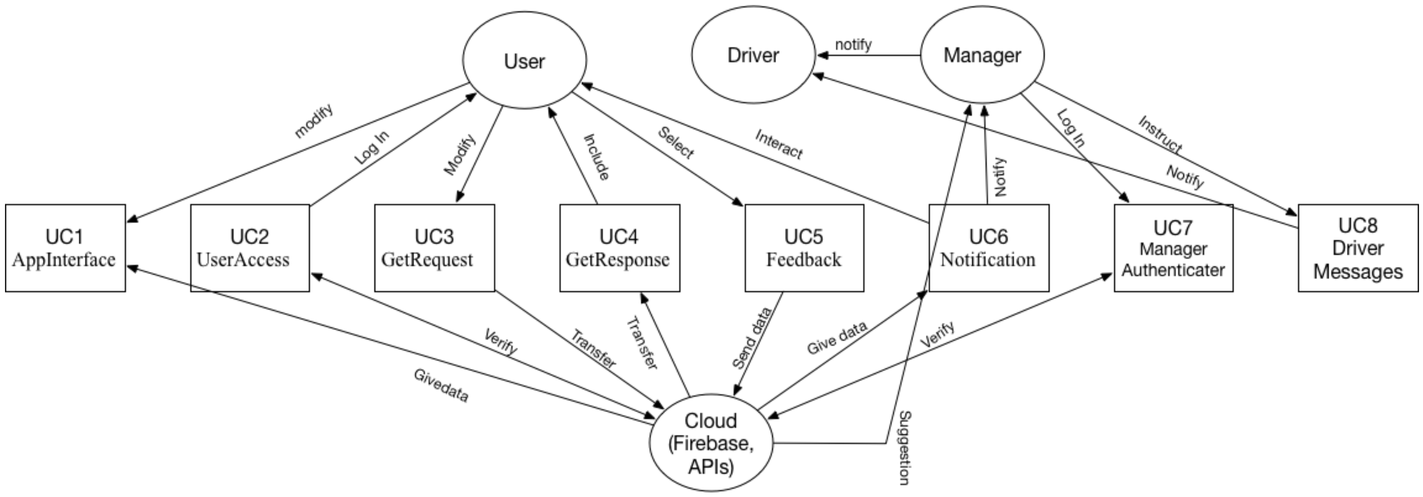
- The goal of mapping service is to show users' current location and based on the current location, determine whether they can make a request or not.

C. Use Cases

I. Casual Description

- UC1: AppInterface
    - Pages of the application that enable users to log in, make request, know the time of arrival and send feedback.
- UC2: UserAccess
    - A login page will be created to verify that the user is a Rutgers student. When the user successfully logs in, the application will determine whether the user can make the request or not based on current location.
- UC3: GetRequest
    - AppInterface will request some information from the user to specify their needs. User will tell the AppInterface the original bus stop, the preferred bus route, the destination bus stop.
- UC4: GetResponse
    - A response page will be created after the user puts all the details into the UC3 page. Users are able to know the estimated time of arrival (ETA).
- UC5: SendFeedback
    - User can based on bus ETA, send some feedback to database to let the system know whether they get on the bus or not.
- UC6: GetNotification
    - User will receive notifications from the AppInterface. Notification include the expected rush hour in rutgers and the unexpected demand surge.
- UC7: ManagerAuthenticator (sub-use case)
    - Manager will be able to log in to the web interface designed for accessing the bus management system. They can visualize the number of requests for each routes and stations on the screen. (Details are shown in the manager webpage interface design)
- UC8: SendMessageToDriver
    - Manager will be able to send message to a specific bus driver or broadcast a message to all the bus drivers.

II. Use Case Diagram



III. Traceability Matrix

Req	Priority Weights	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8
ST1	2	X	X						
ST2	3	X		X					
ST3	3		X						
ST4	6			X					
ST5	4				X	X	X		
ST6	2	X				X			
ST7	5								
ST8	2						X		
ST9	2								
ST10	4								
ST11	3								X

ST12	6								
ST13	4								
ST14	2								
ST15	6						X		
ST16	4				X				
ST17	1								
ST18	3								
ST19	2						X		
ST20	2							X	
ST21	1	X	X						
ST22	2	X		X					
ST23	0.5					X			
ST24	0.5	X				X			
ST25	1						X		
ST26	4								
ST27	0.5								
ST28	0.5								
ST29	2								
ST30	1								X
ST31	2								
ST32	0.5								
ST33	1								
ST34	4	X	X			X			
Total PW		14.5	10	11	8	11	15	2	4

IV. Fully-Dressed Description

UC-1: App interface

Use Case UC	App Interface
Related Requirement:	All STs interacting with Application
Initiating Actor:	App Users
Actor's goal:	Includes several interfaces (Log in page, Request page, feedback page). Interfaces are going to help users visualize which button to press and request.
Participating Actor:	Application
Preconditions:	● For Request page and feedback page, it may require users to log in successfully by entering correct username and password.
Postconditions:	● Display correct page when requested
Flow of Events for Main Success Scenario	1. App users start to use the application, display log-in page. 2. App can display correct pages automatically all the time (If authentication failed, remain at loggin page etc) 3. Icons can be pressed and are able to jump to specific pages when needed.

UC-2

A login page will be created to identify the user's rutgers student identity. When user successfully login, the application will determine whether user can make the request or not based on current user location.

Use Case UC	UserAccess
Related Requirement:	ST-1
Initiating Actor:	Students or visitors

Actor's goal:	To login in the app and request a bus
Participating Actor:	Bus driver, User Application, Database, Mapping service
Preconditions:	● The system can recognize the user's identity ● System ask users for NetID and password
Postconditions:	● system allow users who enter the correct information to another page ● system make users who enter the wrong information stay at this page
Flow of Events for Main Success Scenario	-> 1.Student or Visitor touch the icon on smartphone to open the "User Application". 2. Student or Visitor type in their NetID and password. <- 3. System check the information and jump to page "GetRequest"

UC-3

Use Case 3	GetRequest
Related Requirement	ST-2 ST-4 ST-22
Initiating Actor	User who wants to take bus
Actor's goal	App need to collect some information from users to specify their demands.
Participating Actor	AppInterface, Database
Preconditions	-Users know which buses they want to take and where they want to go. -App can correctly receive information and can specify information from different users.
Postconditions	-App receive users' information and specify their demands. -After users enter information, AppInterface will link users to next page, "GetResponse".
Flow of Events for Main Success Scenario	1. User begins to enter three important information into App: -Original bus stop. -The preferred bus route. -Destination bus stop. 2. App receive information and specify demand. 3. Users will jump to next page, "GetResponse".

Use Case 4

Use case 4	Get Response
Related Requirement	ST-5
Initiating Actor	User who logged in and request for the bus
Actor’s Goal	User will know the estimated time of arrival (ETA)
Participating Actor	Application,Database,Tracking Service
Preconditions	Application is available Database is not empty
Postconditions	1.Students know the bus route they are waiting. 2.Students know the waiting time of selected bus to their current bus stop.
Flow of Events for Main Success Scenario	←the system track the buses and display buses arrival time (ETA)

UC5

Use Case UC-5	Feedback
Related Requirement:	ST-5,ST-6,ST-23,ST-24,ST-34
Initiating Actor:	Users who login and request for bus
Actor’s goal:	To check users get on bus or not
Participating Actor:	Database
Preconditions:	A route of bus is requested by users
Postconditions:	A bus has passed the requested station
Flow of Events for Main Success Scenario	1. The App sends a notification when requested bus arrives. 2. Users choose “I’m on the bus” or “I’m not on the bus” 3. Users click “I’m on the bus” 4. System signals affirmation,

	signals to CheckDevice to complete users' request. signals to MessageDevice to send users a questionnaire about improving advices signals to ScheduleDevice to collect successful sample
Flow of Events for Extensions(Alternate Scenarios)	1. The App sends a notification when requested bus arrives. 2. Users choose "I'm on the bus" or "I'm not on the bus" 3. Users click "I'm not on the bus" 4. System signals affirmation, signals to CheckDevice to resend the request. signals to Timer to recalculate the bus arriving time signals to MessageDevice to send users a notification about next bus signals to ScheduleDevice to collect failure sample

UC-6

Use Case 6	GetNotification
Related Requirement	ST-5, ST-7, ST-15, ST-19, ST-25, ST-26
Initiating Actor	User who wants to receive notifications
Actor's goal	The app will send a notification through native mobile push services to alert users at a set interval of the current condition of the bus services and expected disruptions or events
Participating Actor	Application, Database
Preconditions	● Has set in the options what they want to receive notifications on and at what time intervals
Postconditions	● The timer for notifications is reset
Flow of Events for Main Success Scenario	1. After a set period of time has passed, the App Interface will request information from the database automatically in the background. 2. A summary of the information is sent through push services.

UC-7 (ManagerAuthenticator)

Use Case UC-7	ManagerAuthenticator (sub-use case)
Related Requirements:	ST-20
Initiating Actor:	Manager
Actor's Goal:	To be positively identified by the system (at the web interface)
Participating Actors:	ManagerApplication (web interface)
Preconditions:	● The set of valid keys stored in the system
Postconditions:	● The manager is authenticated and logged into the web interface for managing the buses/drivers
Flow of Events for Main Success Scenario:	← 1: System prompts the initiating actor for identification, e.g., alphanumeric key → 2: Manager (initiating actor) supplies a valid identification key. ← 3: System (a) verifies that the key is valid, and (b) signals to the actor the key validity
Flow of Events for Extensions (Alternate Scenarios):	← 2a: Manager enters an invalid identification key ← 1: System (a) detects error, (b) signals to the actor, (c) gives the actor an option of retrieving the identification key using text message → 2: Manager supplies a valid identification key 3: Same a Step 3 above

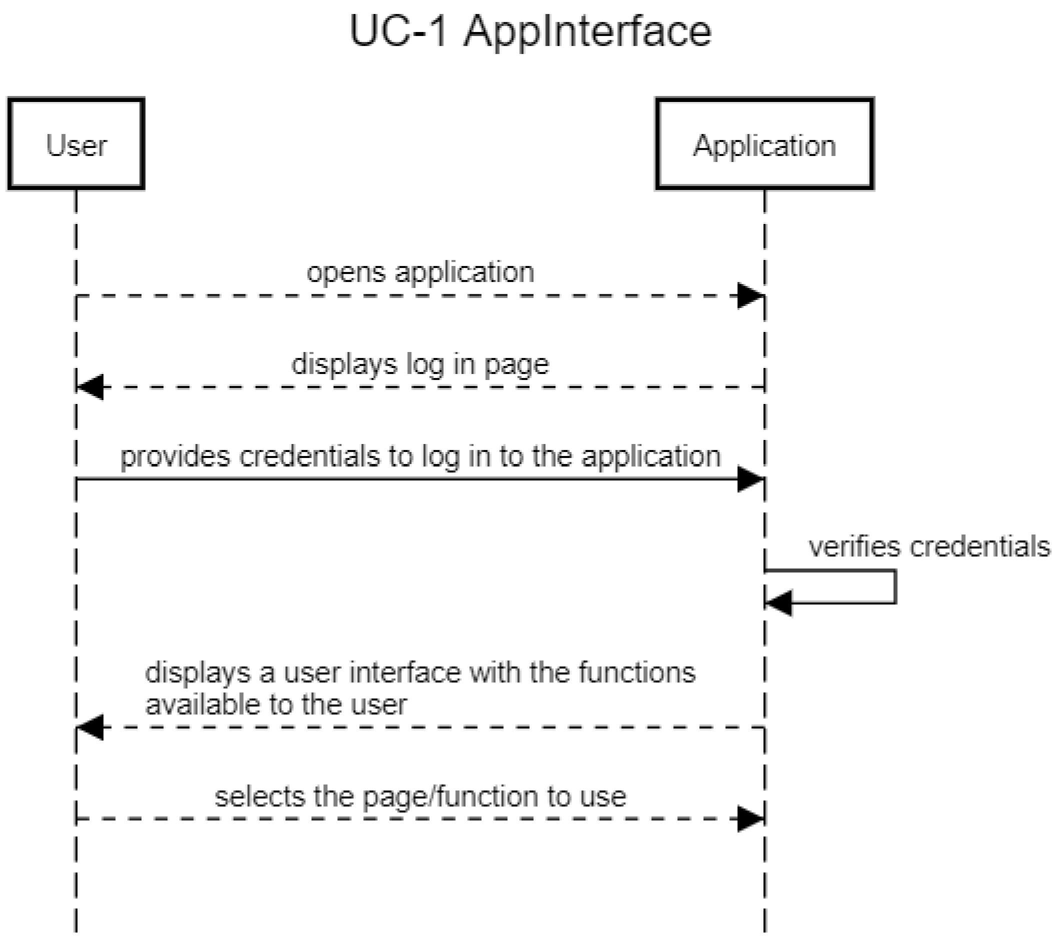
UC-8 (SendMessageToDriver)

Use case UC-8	SendMessageToDriver
Related Requirements:	ST-30, ST-11
Initiating Actor:	Manager
Actor's Goal:	To successfully send a message to a specific bus driver or broadcast a message to all bus drivers
Participating Actors:	Manager Application (web interface), Bus Driver

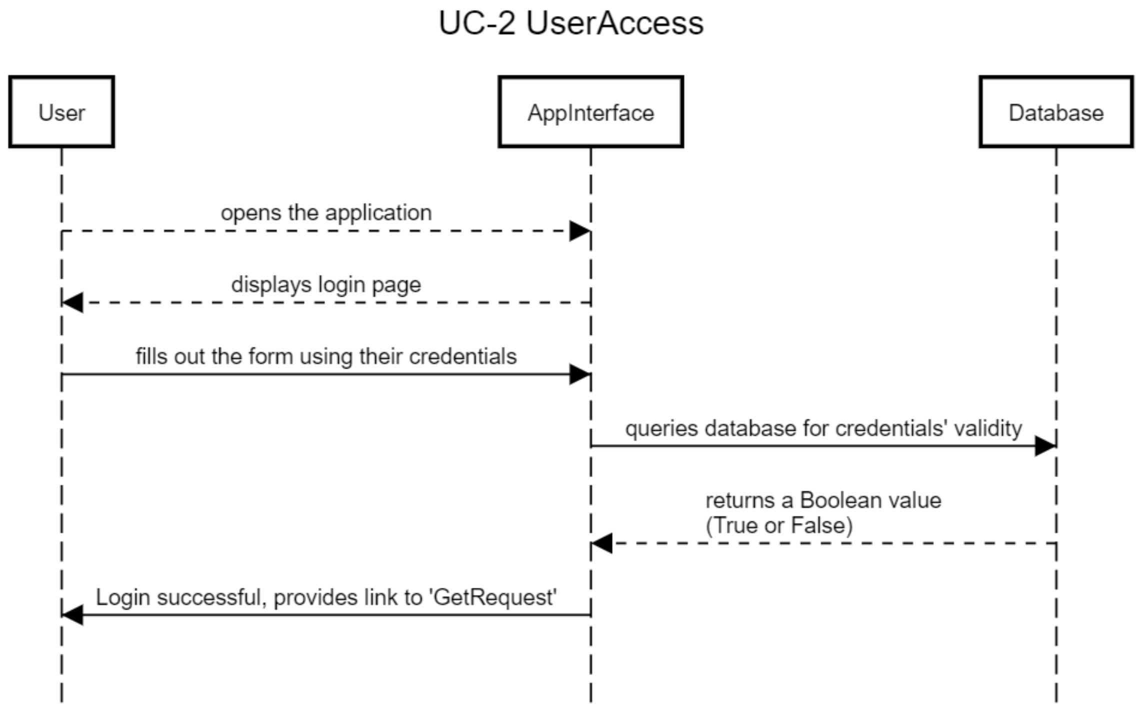
Preconditions:	● The system displays the menu of available functions to the manager ● The bus driver(s) is/are able to receive notifications from the manager, i.e., cell phone switched on and active mobile signal
Postconditions:	● The message that the manager intended to send is received by the bus driver(s).
Flow of events for main success scenario:	← 1: The manager logs into the web interface and opts to send a message to either a specific bus driver or broadcast a message to all the bus drivers 2: include: ManagerAuthenticator (UC-7) → 2: The manager selects ‘Broadcast’ or a specific bus driver, types a message and presses ‘Submit’ ← 3: System (a) signals to the manager that the message was sent. → 4: The bus driver receives the message that the manager sent and either responds or acts accordingly.

D. System Sequence Diagram

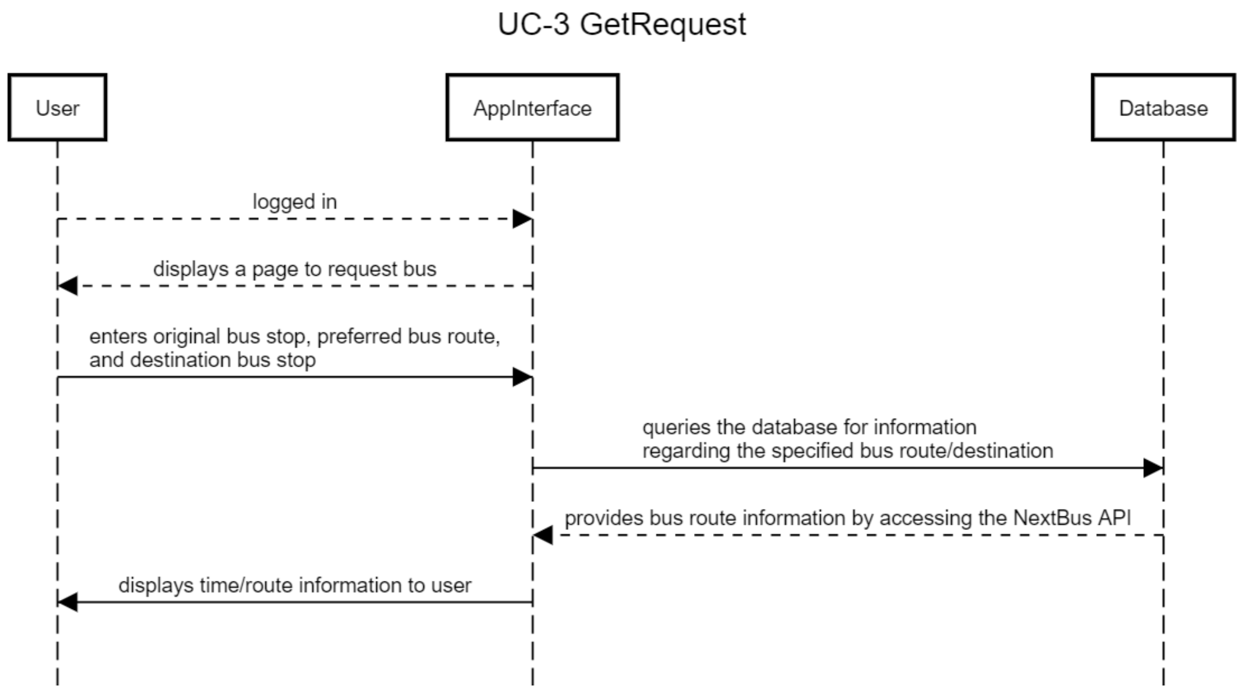
Use Case 1: AppInterface


UC-1 AppInterface
User
Application
opens application
displays log in page
provides credentials to log in to the application
verifies credentials
displays a user interface with the functions available to the user
selects the page/function to use

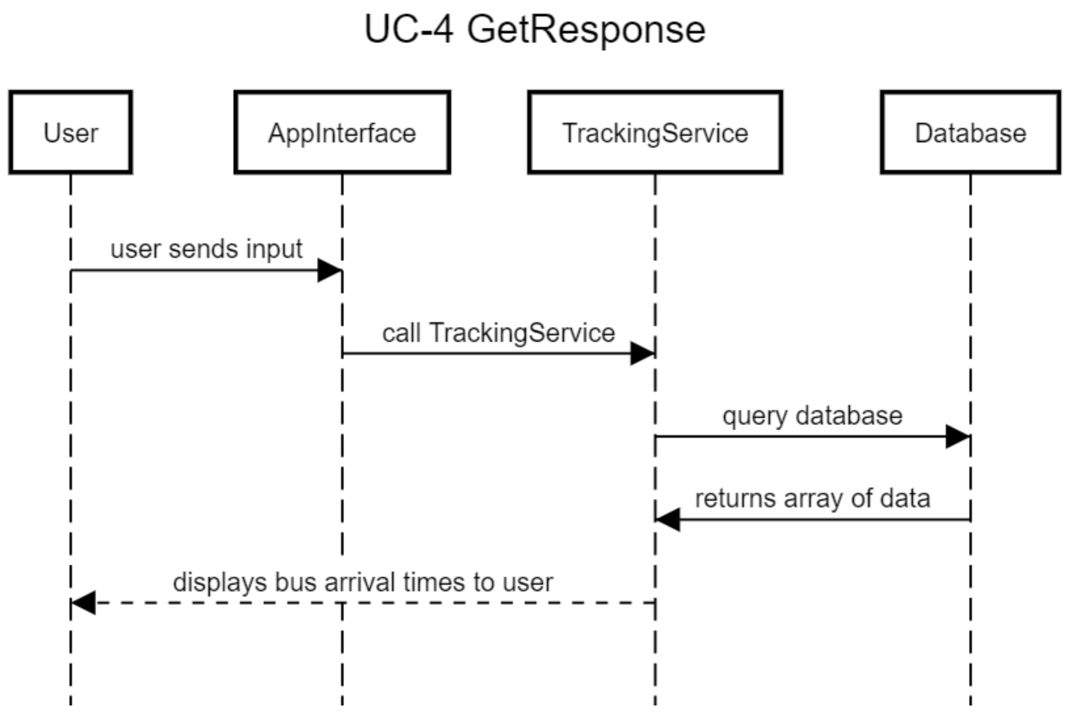
Use Case 2: UserAccess


UC-2 UserAccess
User
AppInterface
Database
opens the application
displays login page
fills out the form using their credentials
queries database for credentials' validity
returns a Boolean value
(True or False)
Login successful, provides link to 'GetRequest'

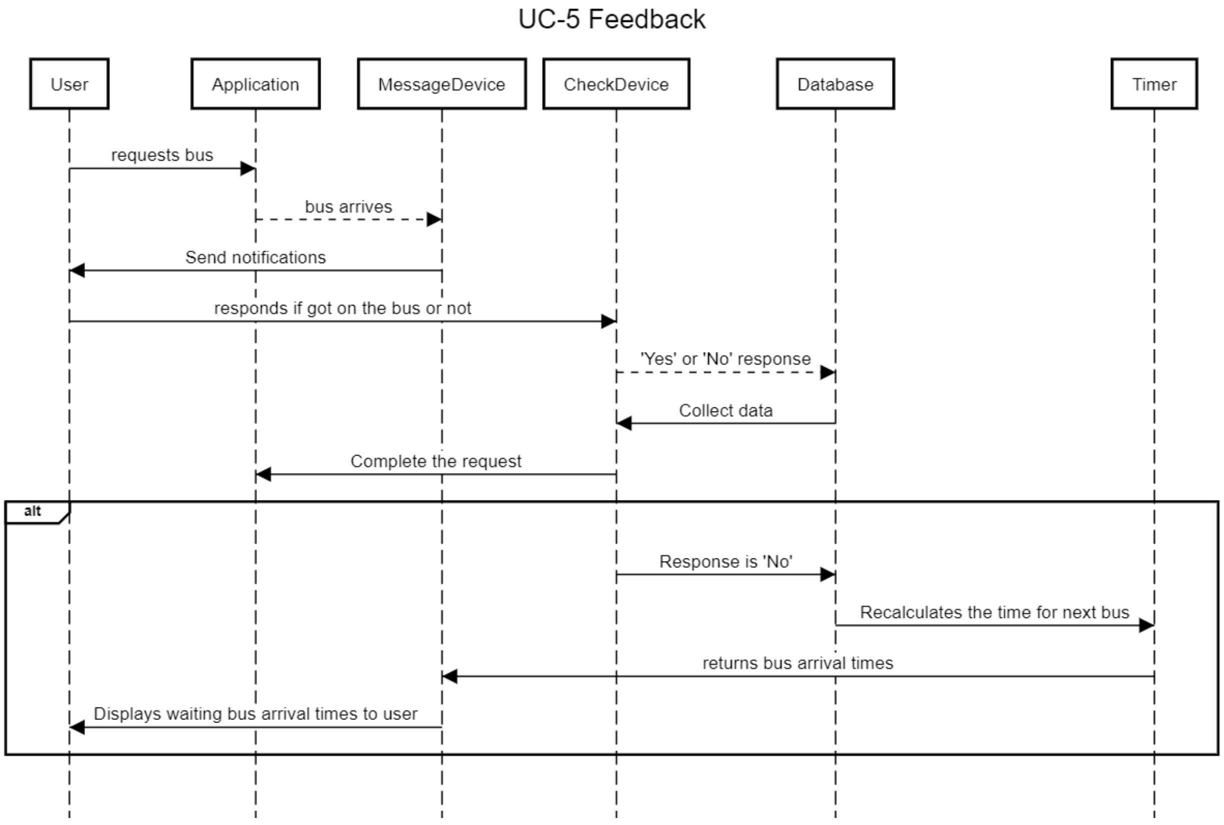
Use Case 3: GetRequest
Note: This use case sequence takes place once the user has logged in successfully into the application and is on the page to select the original bus stop, preferred bus route, and destination bus stop.


UC-3 GetRequest
User
AppInterface
Database
logged in
displays a page to request bus
enters original bus stop, preferred bus route,
and destination bus stop
queries the database for information
regarding the specified bus route/destination
provides bus route information by accessing the NextBus API
displays time/route information to user

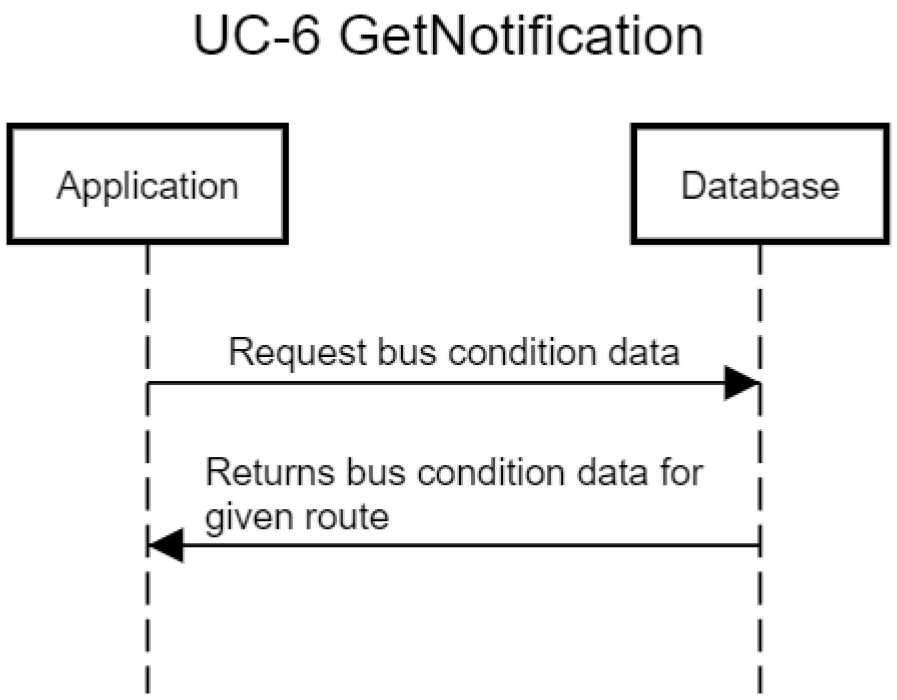
Use Case 4: GetResponse



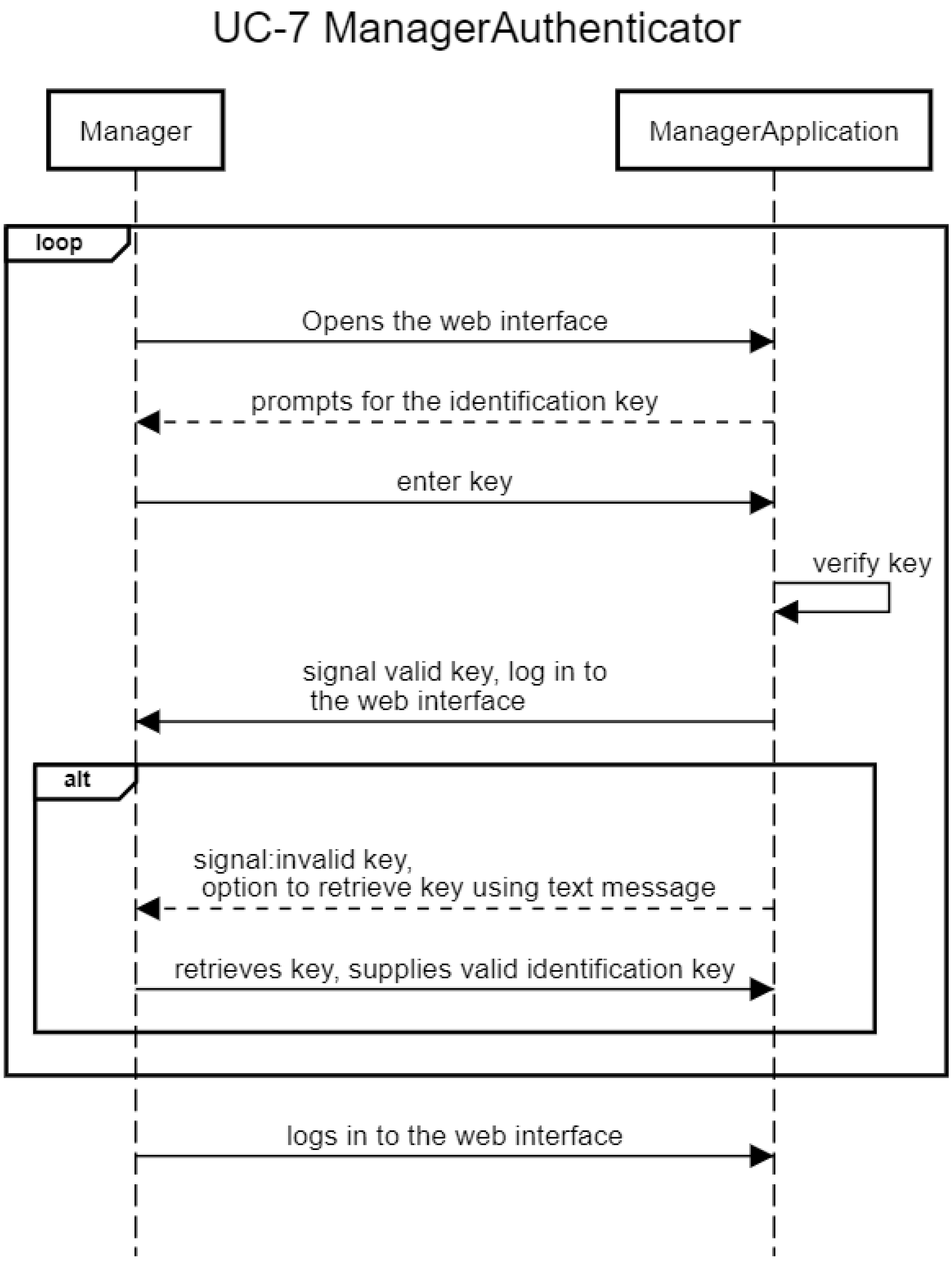
Use Case 5: Feedback



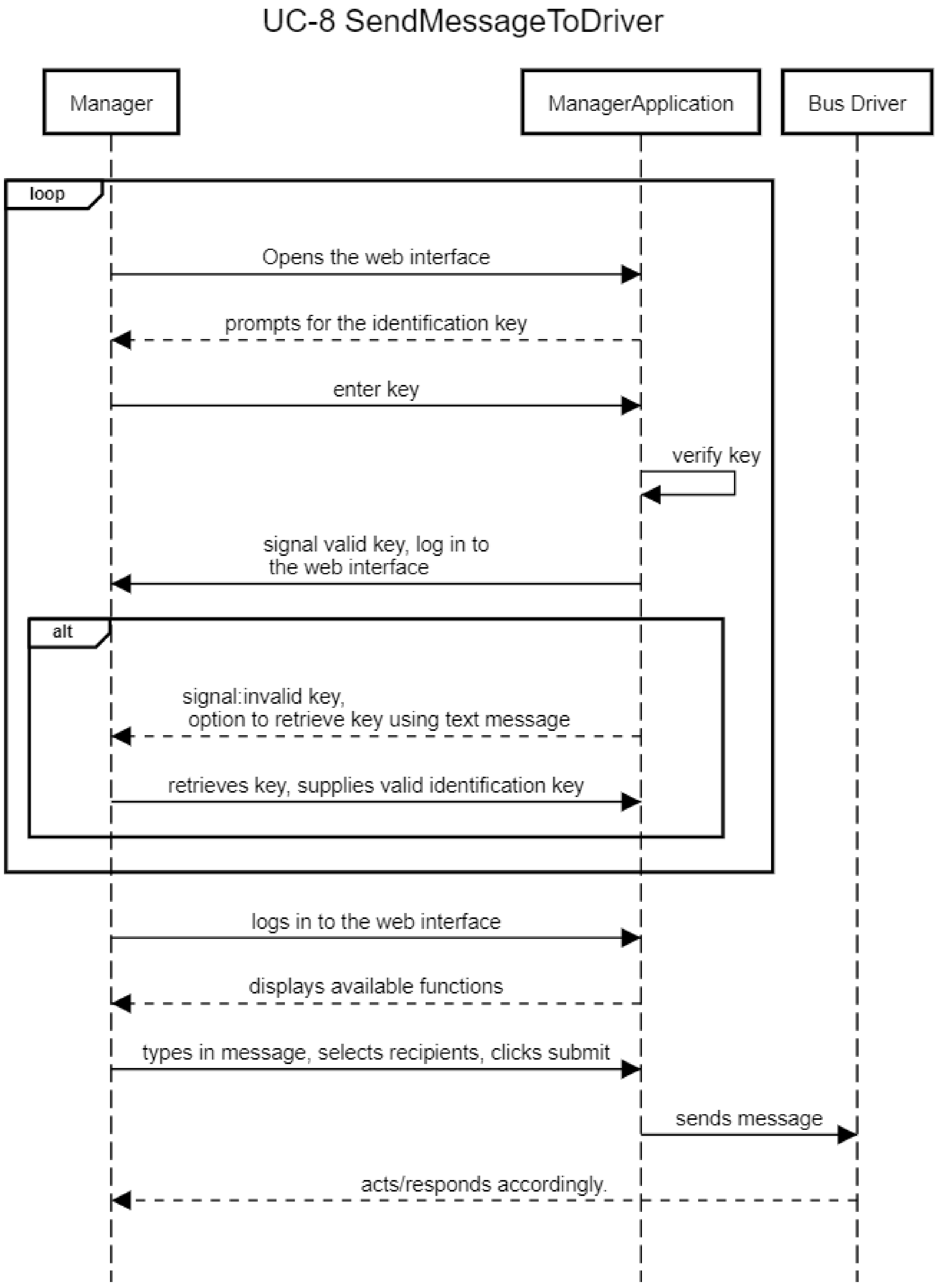
Use Case 6: GetNotification


UC-6 GetNotification
Application
Database
Request bus condition data
Returns bus condition data for given route

Use Case 7: ManagerAuthenticator


UC-7 ManagerAuthenticator
Manager
ManagerApplication
loop
Opens the web interface
prompts for the identification key
enter key
verify key
signal valid key, log in to the web interface
alt
signal:invalid key, option to retrieve key using text message
retrieves key, supplies valid identification key
logs in to the web interface

Use Case 8: SendMessageToDriver

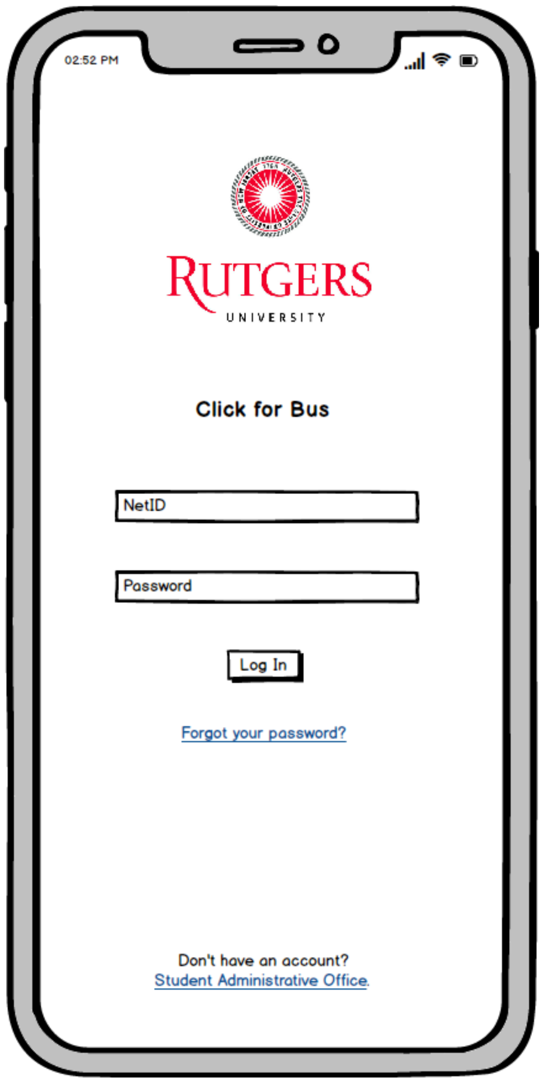

UC-8 SendMessageToDriver
Manager
ManagerApplication
Bus Driver
loop
Opens the web interface
prompts for the identification key
enter key
verify key
signal valid key, log in to the web interface
alt
signal:invalid key, option to retrieve key using text message
retrieves key, supplies valid identification key
logs in to the web interface
displays available functions
types in message, selects recipients, clicks submit
sends message
acts/responds accordingly.

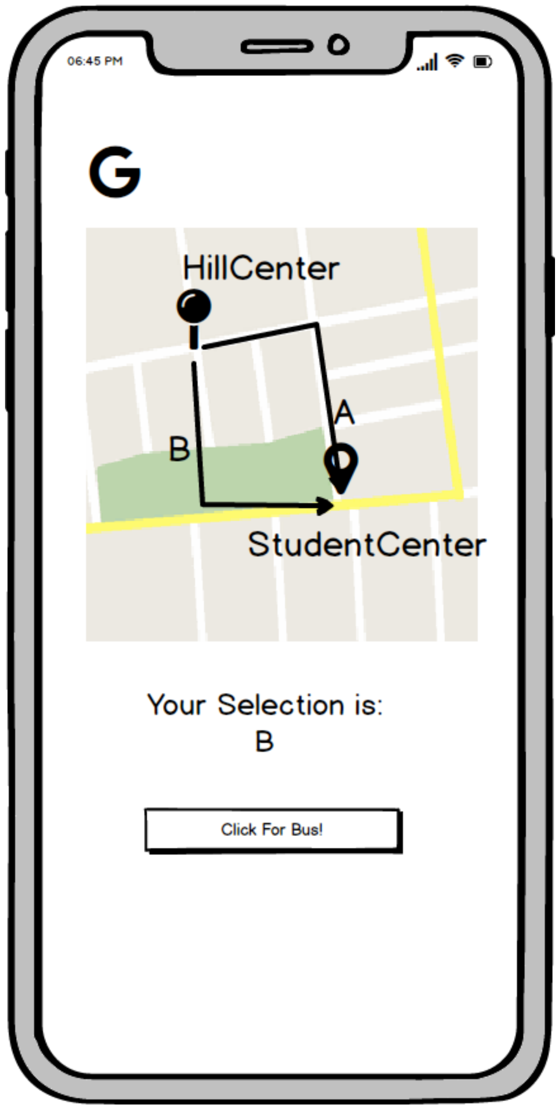
Section 4. User Interface Specification

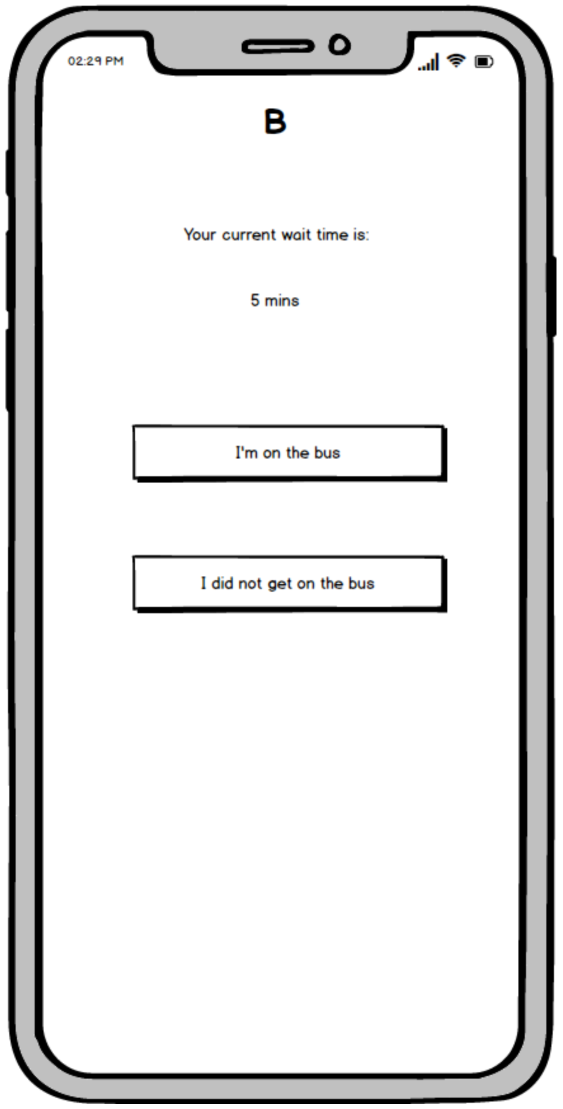
A. Preliminary Design

<u>User Interface:</u>

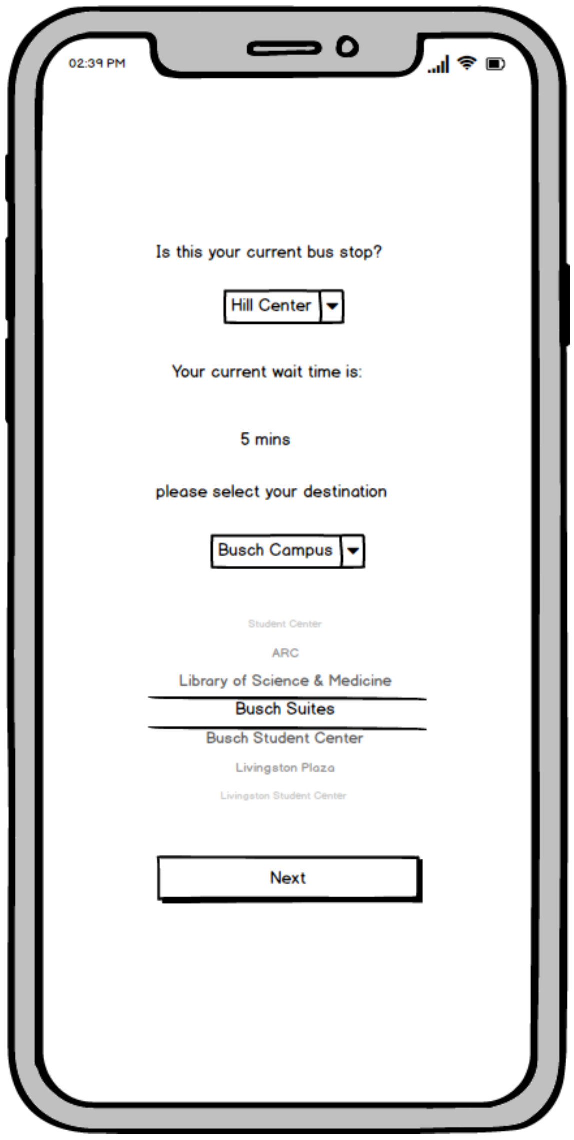
Student version:




06:45 PM
G
HillCenter
A
B
StudentCenter
Your Selection is:
B
Click For Bus!


02:29 PM
B
Your current wait time is:
5 mins
I'm on the bus
I did not get on the bus

Alternate:


02:39 PM

Is this your current bus stop?

Hill Center

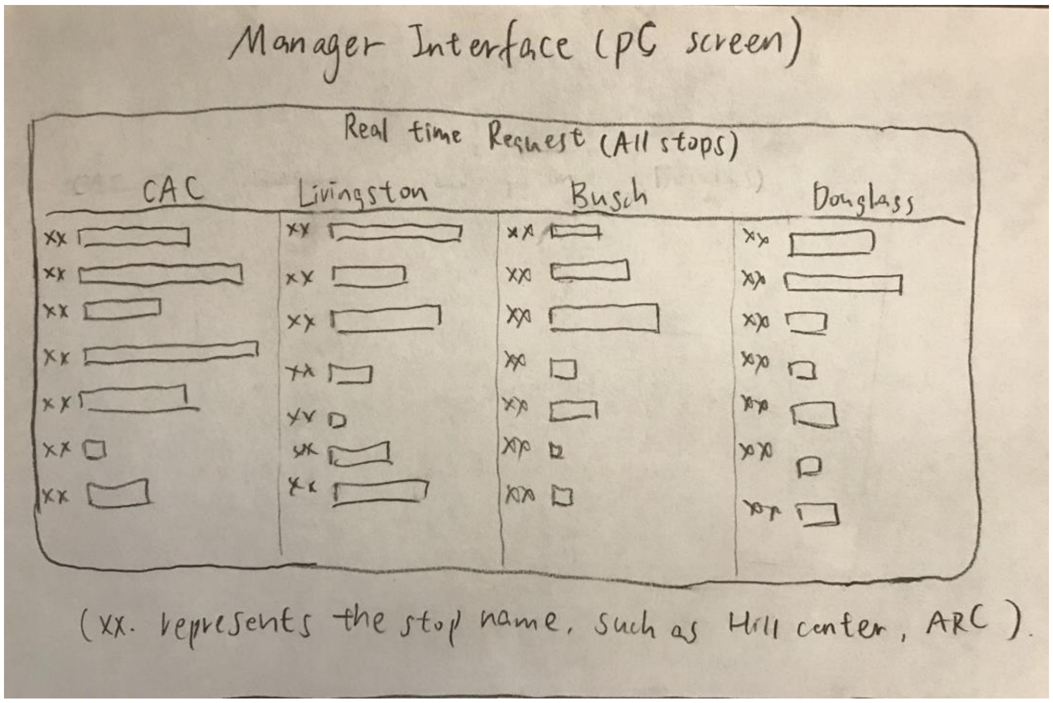
Your current wait time is:

5 mins

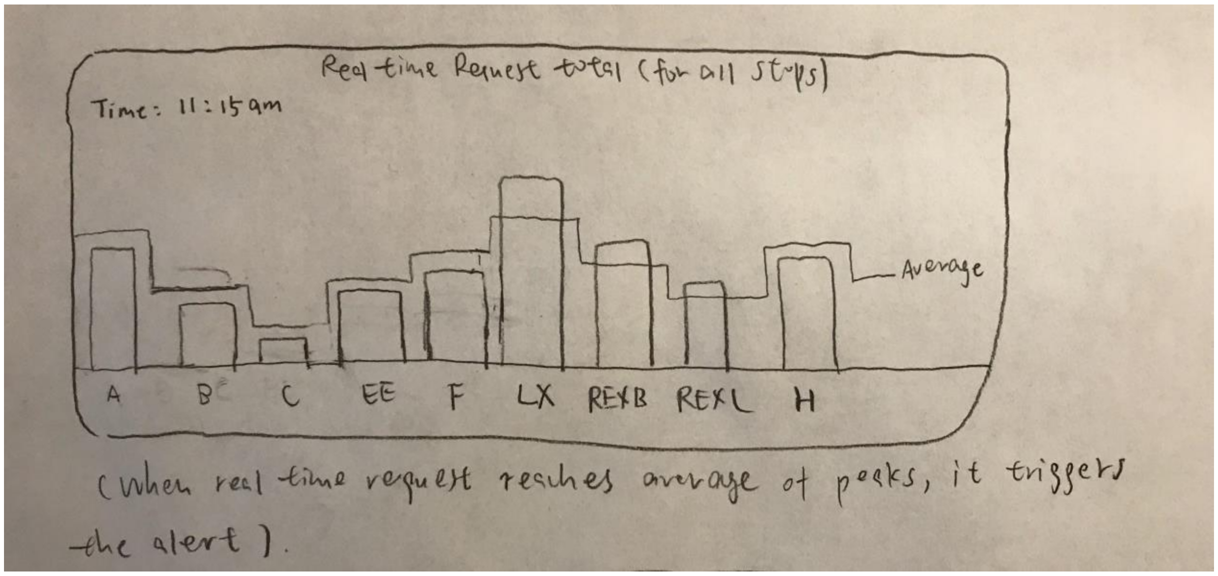
please select your destination

Busch Campus

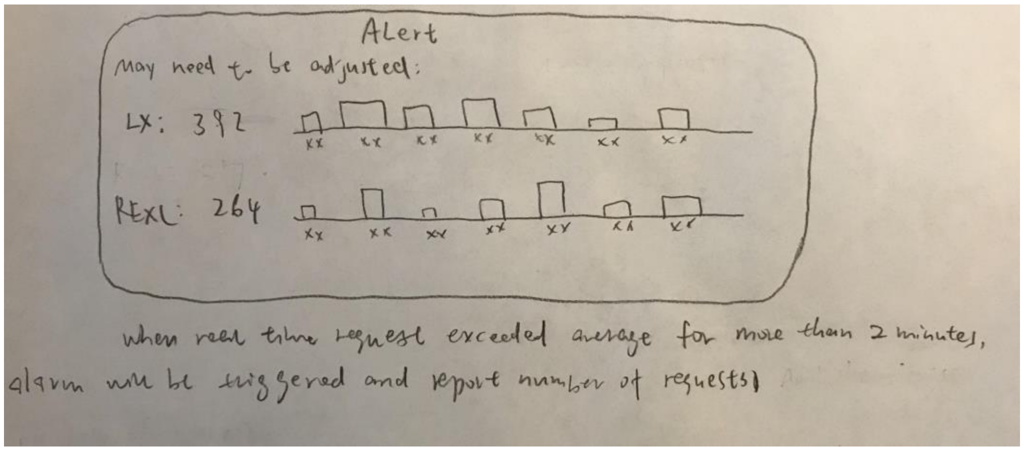
Manager version:


Manager Interface (pC screen)
Real time Request (All stops)
CAC
Livingston
Busch
Douglass
(xx. represents the stop name, such as Hill center, ARC).

This interface shows the real time requests of all campus and displays the number of requests of every stop of each campus.

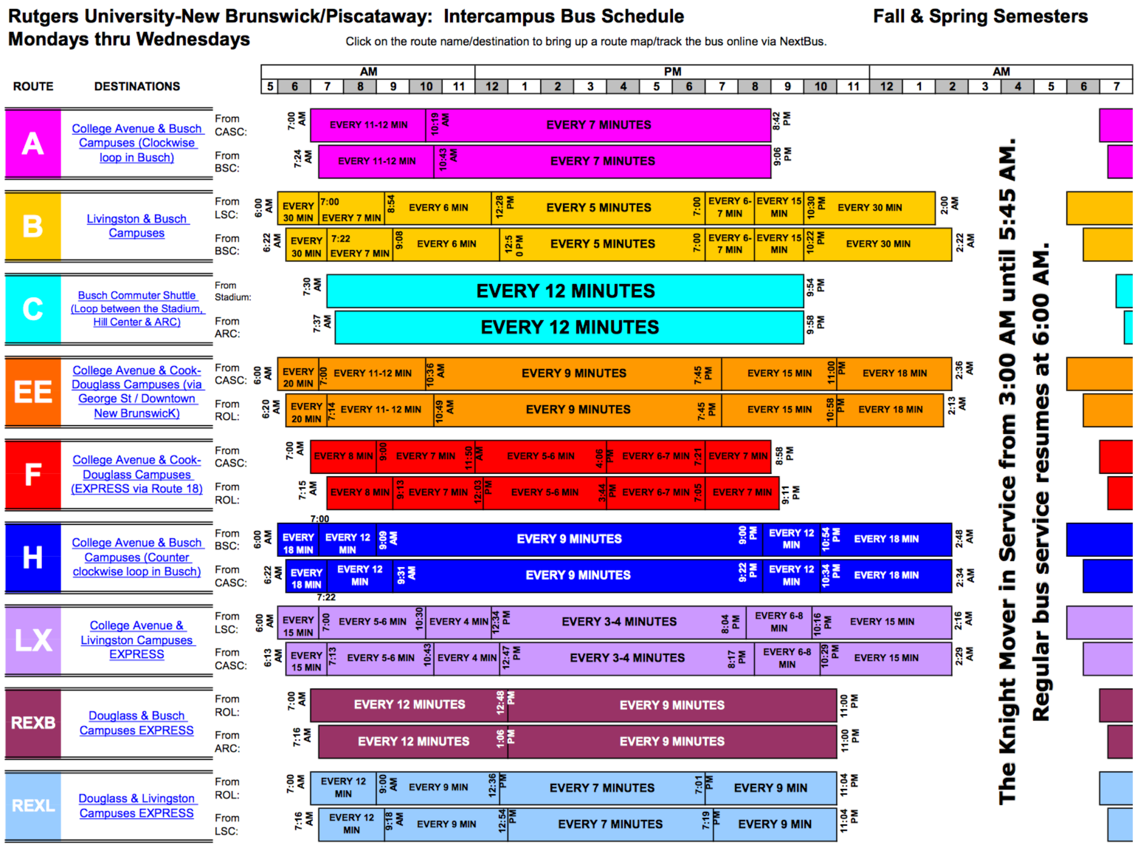

Real-time Request total (for all stops)
Time: 11:15am
Average
A
B
C
EE
F
LX
REXB
REXL
H
(when real time request reaches average of peaks, it triggers the alert).

Example:
Time: Monday Feb 12/ 2018
This interface shows the total requests of each routes, and shows the average of peak of last Monday Feb 04/2018. If the number of real time request reaches a limit(peak) for 2 minutes, it will trigger the alert which will be shown below.


Alert
May need to be adjusted:
LX: 392
XX XX XX XX XX XX XX
REXL: 264
XX XX XX XX XX XX XX
when real time request exceeded average for more than 2 minutes,
alarm will be triggered and report number of requests)

This is the alert interface. Will display the routes which exceed the average needs, there also have listed number of requests and statistics of each stops.

Bus driver version



Rutgers University-New Brunswick/Piscataway: Intercampus Bus Schedule **Fall & Spring Semesters**
Mondays thru Wednesdays
Click on the route name/destination to bring up a route map/track the bus online via NextBus.

ROUTE	DESTINATIONS		Schedule (AM 5–11, PM 12–11, AM 12–7)
A	College Avenue & Busch Campuses (Clockwise loop in Busch)	From CASC:	7:00 AM · EVERY 11-12 MIN · 10:19 AM · EVERY 7 MINUTES · 8:42 PM
		From BSC:	7:24 AM · EVERY 11-12 MIN · 10:43 AM · EVERY 7 MINUTES · 9:06 PM
B	Livingston & Busch Campuses	From LSC:	6:00 AM · EVERY 30 MIN · 7:00 · EVERY 7 MIN · 8:54 · EVERY 6 MIN · 12:28 PM · EVERY 5 MINUTES · 7:00 · EVERY 6-7 MIN · EVERY 15 MIN · 10:30 PM · EVERY 30 MIN · 2:00 AM
		From BSC:	6:22 AM · EVERY 30 MIN · 7:22 · EVERY 7 MIN · 9:08 · EVERY 6 MIN · 12:50 PM · EVERY 5 MINUTES · 7:00 · EVERY 6-7 MIN · EVERY 15 MIN · 10:22 PM · EVERY 30 MIN · 2:22 AM
C	Busch Commuter Shuttle (Loop between the Stadium, Hill Center & ARC)	From Stadium:	7:30 AM · EVERY 12 MINUTES · 9:54 PM
		From ARC:	7:37 AM · EVERY 12 MINUTES · 9:58 PM
EE	College Avenue & Cook-Douglass Campuses (via George St / Downtown New BrunswicK)	From CASC:	6:00 AM · EVERY 20 MIN · 7:00 · EVERY 11-12 MIN · 10:36 AM · EVERY 9 MINUTES · 7:45 PM · EVERY 15 MIN · 11:00 PM · EVERY 18 MIN · 2:36 AM
		From ROL:	6:20 AM · EVERY 20 MIN · 7:14 · EVERY 11- 12 MIN · 10:49 AM · EVERY 9 MINUTES · 7:45 PM · EVERY 15 MIN · 10:58 PM · EVERY 18 MIN · 2:13 AM
F	College Avenue & Cook-Douglass Campuses (EXPRESS via Route 18)	From CASC:	7:00 AM · EVERY 8 MIN · 9:00 · EVERY 7 MIN · 11:50 AM · EVERY 5-6 MIN · 4:06 PM · EVERY 6-7 MIN · 7:21 · EVERY 7 MIN · 8:58 PM
		From ROL:	7:15 AM · EVERY 8 MIN · 9:13 · EVERY 7 MIN · 12:03 PM · EVERY 5-6 MIN · 3:44 PM · EVERY 6-7 MIN · 7:05 · EVERY 7 MIN · 9:11 PM
H	College Avenue & Busch Campuses (Counter clockwise loop in Busch)	From BSC:	6:00 AM · EVERY 18 MIN · 7:00 · EVERY 12 MIN · 9:09 AM · EVERY 9 MINUTES · 9:00 PM · EVERY 12 MIN · 10:54 PM · EVERY 18 MIN · 2:48 AM
		From CASC:	6:22 AM · EVERY 18 MIN · 7:22 · EVERY 12 MIN · 9:31 AM · EVERY 9 MINUTES · 9:22 PM · EVERY 12 MIN · 10:34 PM · EVERY 18 MIN · 2:34 AM
LX	College Avenue & Livingston Campuses EXPRESS	From LSC:	6:00 AM · EVERY 15 MIN · 7:00 · EVERY 5-6 MIN · 10:30 · EVERY 4 MIN · 12:34 PM · EVERY 3-4 MINUTES · 8:04 PM · EVERY 6-8 MIN · 10:16 PM · EVERY 15 MIN · 2:16 AM
		From CASC:	6:13 AM · EVERY 15 MIN · 7:13 · EVERY 5-6 MIN · 10:43 · EVERY 4 MIN · 12:47 PM · EVERY 3-4 MINUTES · 8:17 PM · EVERY 6-8 MIN · 10:29 PM · EVERY 15 MIN · 2:29 AM
REXB	Douglass & Busch Campuses EXPRESS	From ROL:	7:00 AM · EVERY 12 MINUTES · 12:48 PM · EVERY 9 MINUTES · 11:00 PM
		From ARC:	7:16 AM · EVERY 12 MINUTES · 1:06 PM · EVERY 9 MINUTES · 11:00 PM
REXL	Douglass & Livingston Campuses EXPRESS	From ROL:	7:00 AM · EVERY 12 MIN · 9:00 AM · EVERY 9 MIN · 12:36 PM · EVERY 7 MINUTES · 7:01 PM · EVERY 9 MIN · 11:04 PM
		From LSC:	7:16 AM · EVERY 12 MIN · 9:18 AM · EVERY 9 MIN · 12:54 PM · EVERY 7 MINUTES · 7:19 PM · EVERY 9 MIN · 11:04 PM

The Knight Mover in Service from 3:00 AM until 5:45 AM.
Regular bus service resumes at 6:00 AM.

Courtesy of the Rutgers Transportation Department Website
<http://rudots.rutgers.edu/campusbuses.shtml>

B. User Effort Estimation

Student's Mobile Application:

1. Authentication (1 click and keystrokes)

a. Press the keyboards in mobile application in order to input Rutgers NetID and Password.
b. Click the "login" button to access application

2. Navigation of Campus Bus Stops (3 clicks)

a. Application Auto-confirms the current bus stop, but the user can click their "current bus stop" in order to avoid system/GPS error (not necessary to count in clicks).
b. Click the "destination bus stop"
c. Click "bus route" in the Google Map interface with suggested route showed on screen.
d. Click "Next" in order to send their requests

3. Confirmation (0 clicks)

a. If the student feels they have their selected the wrong options, they can click "Go back" in order to change their request (not necessary to count in clicks).

4. Feedbacks (1 click)

a. User can Click "On bus" or "Not on bus" in order to send feedbacks to the server to notify our system.

Manager’s Web Application:

1. Authentication (1 click and keystrokes)

a. Press the keyboard in web application to input username and Password.
b. Click the “login” button to access application

2. Operation of bus route(T2 clicks)

a. Press “Change route” by system to automatically notify the manager to change the bus route.
b. Press “Confirm changing” to confirm the click before.

Bus drivers:

Drivers simply need to check their static schedule and listen to the manager instructions.

Section 5. Domain Analysis

A. Domain Model

<u>Use Case 1: App Interface</u>

- Concept definitions:

Responsibility Description	Type	Concept Name
Once the app is opened, display login page	D	App Interface
After logging in, display the bus request page and options	D	App Interface
Display feedback options when user finish their request	D	App Interface

- Association Definitions:

Concept pair name	Associated Definition	Associated Name
User↔Identity Check	In order to use the request function, users should login to the app successfully first (precondition). After identity check, users can request or search based on the display by interface.	Identity check

- Attribute Definitions:

Concept	Attributes	Attribute Description
App Interface	All Options button App Menu	Display space to login and options menu, help users to

		visualize the app
Database	Bus route data	Associated with the request page, display how much time left for bus routes
Database	Identity check	Associate with login page, display space for users to login and check if they used a valid username and password

Use Case 2: UserAccess

- Concept definitions

Responsibility Description	Type	Concept Name
Check the current App user's identity	D	Database
Once verify user's identity, show "GetRequest" page	D	AppInterface

- Association Definitions

Concept Pair name	Associated Definition	Associated Name
User↔Identity Check	User fill out the NetID and password information. Using Rutgers authentication API check user's identity	Identity check
UserAccess↔GetRequest	Once confirm user's identity, link to "GetRequest" page	Linking

- Attribute Definitions

Concept	Attributes	Attribute Description

Application	Rutgers authentication	A system to check the user’s identity
	Forget password	A Rutgers website that will help find password
Database	Rutgers authentication API	Table of valid NetID information

<u>Use case 3: GetRequest</u>

- Concept Definition

Responsibility Description	**Type**	**Concept Name**
Users enter information into AppInterface	D	AppInterface
AppInterface sends demand to database	D	Database
AppInterface receives demand and show “GetResponse” page.	D	AppInterface

- Association Definitions

Concept Pair Name	**Associated Definition**	**Associated Name**
User↔AppInterface	User enter their information into AppInterface including original bus station, bus and destination.	DataCollection
AppInterface↔Database	After users enter information, AppInterface sends demand to Database and Database will write down the information.	DataCollection
AppInterface↔GetResponse	AppInterface receives	Linking

	demand and builds respond, then links user into next page, GetResponse.	

- Attribute Definitions

Concept	Attributes	Attribute Description
AppInterface	All information button	Display all of demands that users need and receive information from users.
Database	Demands data	AppInterface receives information from users and sends to Database.

Use Case 4: GetResponse

- Concept definitions

Responsibility Description	Type	Concept Name
Uses the NextBus API to display the waiting time for selected bus	D	Tracking Service
Take in user input to find which bus the user is requesting. Output selected the bus route.	D	Application
Contains the bus GPS coordinates and Those data can be accessed by the application.	K	Database

- Association Definitions

Concept Pair name	Associated Definition	Associated Name
Application↔Tracking Service	Application sends user selected bus to the Tracking Service. The Tracking Service returns the waiting time	Provide data

Tracking Service↔Database	The Tracking Service requests Database information to be used in the function and display the correct estimate waiting time.	Provide data

- Attribute Definitions

Concept	**Attributes**	**Attribute Description**
Application	Bus Routes	Different bus routes the user requirements
	Time	Display the waiting time
Database	Bus Routes Data	Different buses routes
	Gps Coordinates Data	Show the current buses coordinates

<u>Use Case 5: Feedback</u>

Responsibility Description	**Type**	**Concept Name**
Uses the NextBus API to determine the bus is arriving or not	D	Application/Controller
Check users get on the bus or not	D	MessageDevice
Collect users' responds to decide next step	D	CheckDevice
Recalculate the next bus arriving time if user click	D	Timer

“I’m not on the bus”		
Database collect completed request and failure request to help analyzing the future schedule	K	Database
Sending a notification to the users who do not get on bus about the next bus arriving time	K	Application/Controller

- Association Definitions

Concept Pair name	Associated Definition	Associated Name
Application↔MessageDevice	Application remind users about the arriving bus and ask for user status(on bus or not)	Notification Services
MessageDevice↔CheckDevice	Check user status and make next decision	Provide data
CheckDevice↔Database	Collect requests feedback for future analyzing and resend the request for failure case.	Provide data
Database↔Timer	Database sends information about the next bus and estimate waiting time.	Provide data

- Attribute Definitions

Concept	Attributes	Attribute Description
Application	Notifications	Remind for arriving bus
MessageDevice	Check user status	Get on bus or not
CheckDevice	Make decision for database	Determine if resend the request ot not

Database	Collect cases Gps Coordinates Data	Completed and failure request Show the current buses coordinates
Timer	Display the bus status	Show users who do not get on bus about the next bus information

Use Case 6: GetNotification

- Concept definitions

Responsibility Description	**Type**	**Concept Name**
Request current bus conditions and disruptions	D	Database
Push notification with a summary of the requested information	D	AppInterface

- Association Definitions

Concept Pair name	**Associated Definition**	**Associated Name**
Application↔Database	The application will request data from the database to notify the user of current transportation service conditions	Notification Service

- Attribute Definitions

Concept	**Attributes**	**Attribute Description**
Application	Rutgers authentication	A system to check the user's identity
	Forget password	A Rutgers website that will help find pa
Database	Rutgers authentication API	Table of valid NetID information

<u>Use Case 7: ManagerAuthenticator</u>

- Concept Definitions:

Responsibility Description	Type	Concept Name
Shows the login page to the manager	D	AppInterface
Container for manager's authentication data, such as identification key	K	Key
Verify whether or not the identification key entered by the manager is valid	D	KeyChecker
Container for the collection of valid key(s) associated with manager(s)	K	KeyStorage
Send text message to retrieve key	D	KeyRetriever

- Association Definitions:

Concept Pair Name	Associated Definition	Associated Name
AppInterface ↔Key	The manager enters identification key into the login page of the web interface	Identity Check
Key↔KeyChecker	Key is handed off the KeyChecker to verify its validity	Check Key
KeyChecker↔KeyStorage	KeyChecker will check if the provided key is contained in the KeyStorage	Check in KeyStorage
AppInterface↔KeyRetriever	Sends text message to manager containing the ID key	Retrieve Key

- Attribute Definitions:

Concept	Attributes	Attribute Description
AppInterface	Authentication	A system to verify manager’s identity
	Retrieve Key	Interfaces with KeyRetriever for manager to retrieve the identification key
Key	Provided key	The key that the manager provides
KeyChecker	Check if Key is valid	Interfaces with the KeyStorage to check if

		key is valid
KeyStorage	Valid Key Container	Contains the set of valid keys
KeyRetriever	Retrieve key	Text message service (for example, Twilio) to send a text message to manager, containing the valid key

Use Case 8: SendMessageToDriver

Includes Concept Definitions, Association Definitions, and Attribute Definitions from UC-7:ManagerAutherntіcator

- Concept Definitions:

Responsibility Description	Type	Concept Name
Shows a menu/user interface with functions available to manager	D	AppInterface
MessageInterface	K	A part of the AppInterface: User Interface to select recipients and type a message to send to bus drivers
MessageSender	D	A part of the MessageInterface, connects to text message service (such as Twilio) to send text message to the bus drivers

- Association Definitions:

Concept Pair Name	Associated Definition	Associated Name
AppInterface↔MessageInterface	AppInterface shows MessageInterface as a function available to the manager	Display function
MessageInterface↔MessageSender	MessageInterface passes off the message typed by the manager to the MessageSender which connects to a text message service to send the message	Send message

- Attribute Definitions:

Concept	Attributes	Attribute Description
AppInterface	Available Functions	Displays a menu or user interface with functions available to the manager
MessageInterface	Select Recipients	Gives the option of either selecting a specific bus driver to send a message to, or broadcast a message to all the bus drivers
	Type Message	An interface to type a message in
MessageSender	Send Message	Interfaces with MessageInterface to get the message typed by the manager and connects to a text message service to

		send that message

Domain Traceability Matrix

	Application	**MapService**	**Database**	**ManagerApplication**	**TextMessage Service**
UC-1	Display interface				
UC-2	Receive user's information and link to other pages		Check user's input		
UC-3	Receive user's demand and link to other pages		Remember user's demand		
UC-4	Waiting time is shown on Application		Check the NextBus API		
UC-5	Reminding user about the arriving bus	Displaying the information about the bus for users who do not get on bus	Collecting completed and failure request for future analyzing		
UC-6	Application requests transportation service status and sends a		Checks status of bus routes		

	push notification				
UC-7				Provides the log in page for the manager to access the bus management system	Retrieve keys for the manager
UC-8				Provides the user interface with all the functions that the manager can use	Sends message to the bus driver

B. System Operation Contracts

Name	UC-1 App interface
Preconditions	● For Request page and feedback page, it may require users to log in successfully by entering correct username and password.
Postconditions	● Display correct page when requested

Name	UC-2 UserAccess
Preconditions	● The system can recognize the user's identity ● System ask users for NetID and password
Postconditions	● system allow users who enter the correct information to another page ● system make users who enter the wrong information stay at this page

Name	UC-3 GetRequest
Preconditions	-Users know which buses they want to take and where they want to go. -App can correctly receive information and can specify information from different users.
Postconditions	-App receive users' information and specify their demands. -After users enter information, AppInterface will link users to next page, "GetResponse".

Name	UC-4 Get Response

Preconditions	Application is available Database is not empty
Postconditions	1.Students know the bus route they are waiting. 2.Students know the waiting time of selected bus to their current bus stop.

Name	UC-5 Feedback
Preconditions	A route of bus is requested by users
Postconditions	A bus has passed the requested station

Name	UC-6 GetNotification
Preconditions	● Has set in the options what they want to receive notifications on and at what time intervals
Postconditions	● The timer for notifications is reset

Name	UC-7 ManagerAuthenticator
Preconditions	● The set of valid keys stored in the system
Postconditions	● The manager is authenticated and logged into the web interface for managing the buses/drivers

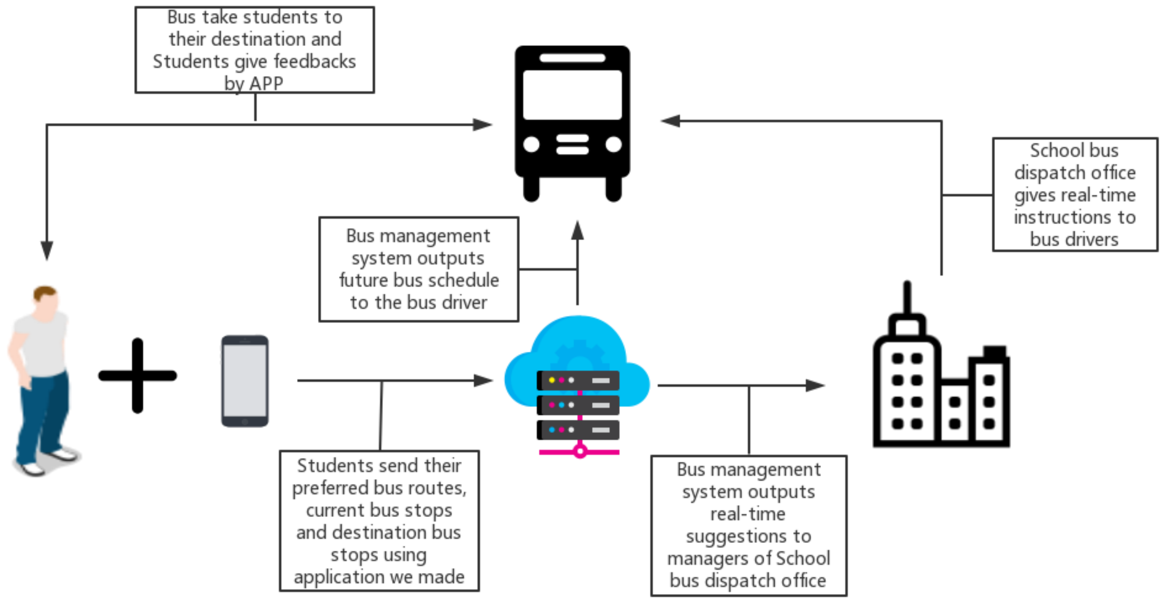
Name:	UC-8 SendMessageToDriver
Preconditions	● The system displays the menu of

	available functions to the manager ● The bus driver(s) is/are able to receive notifications from the manager, i.e., cell phone switched on and active mobile signal
Postconditions	● The message that the manager intended to send is received by the bus driver(s).

C. Mathematical Model

We do not use any mathematical model in this project.

Section 6. Plan of Work


Bus take students to their destination and Students give feedbacks by APP
School bus dispatch office gives real-time instructions to bus drivers
Bus management system outputs future bus schedule to the bus driver
Students send their preferred bus routes, current bus stops and destination bus stops using application we made
Bus management system outputs real-time suggestions to managers of School bus dispatch office

Our goal is to solve the issues of low transportation capacity at peak-hour time lots and reduce the waste during off-hour transportation. The estimated cost for our project should be no more than $100 for a basic implementation.

What circumstances might we need to think about?

- A regular weekday
- A regular weekend
- Occasional heavy traffic during a regular weekday
- Occasional heavy traffic during a regular weekend
- Planned Rutgers events during a regular weekday.
- Planned Rutgers events during a regular weekend.
- Midterm weeks
- Finals weeks

To coordinate with the bus schedules during these conditions, some preliminary work is required

1. We are planning to develop a mobile App for Rutgers students.

- How does this App work?
- Students first select a station that they are willing to take bus(e.g., stadium west lot and livingston student center).
- Then, they choose the bus(e.g., B bus and C bus).
- The students will also select the destination stop.
- The App will send the bus requestments to the control center.
- The control center needs to check the camera in the station which has the heavy demands of bus(This can prevent deliberately tricksy request)
- After verifying the high volumes of students, the control center schedules temporary bus to ease the traffic pressure.
- Sending a notification of extra bus schedule and estimated time to students who request bus.

2. Collecting requesments data in trial period(probably two weeks to a month), or analysing existing data from Rutgers bus control center.

Why we need this step?

- We need to know that each station is busy at which time.
- We need to know that each routing of bus is highly demanded at which time.
- We can forecast a heavy period and schedule more buses in advance to avoid congestions and reduce student waiting time.
- We can plot the graph of busy period and send notifications to students.

How can we complete this step?

- Questionnaire
- Recording student flow rate at each stations by some instructors(e.g., gopro)
- Asking Rutgers for previous records.
- Communicating with students who did similar researches.

3. Applying data to different circumstances to validate our App's feasibility.

How can we do it?

- We input our analysis into the regular weekday and weekend (predictable busy period with scheduled bus system).
- If the occasional heavy traffic happens during regular weekday and weekend, which means that our App receives great demands of bus, our App will work as

described before---verifying the student volume, scheduling extra buses and sending notifications.

- We will send notifications one day before holding Rutgers planned events(e.g., football game and Mega fair) to remind student to avoid some stations. Also, our App will coordinate buses according to the temporal circumstances.
- More students will be taking buses during the exam week. Due to this, we will schedule more buses in certain stations(e.g., Hill center station). For example, most student need to drop off at Hill center station if they take exams in Busch campus.
- In final week, we will focus on the time before final exams start and after final exams end.For example, final exams are usually held during 4pm-7pm and 8pm-11pm. We will schedule more buses during 7pm-8pm to make sure all students can leave the stations and arrive on next exam time.

4. Taking care users' feedback

Why we need feedback?

- Feedback can guide us to improve our App, especially in trial period.
- Bug reports from users may help us to be more precise and resolve the errors that are experienced in the app.
- Great ideas from users may help us develop a better interface.
- Reminders from user may lead us to satisfy their demands which we do not consider.

5. All the team members are proficient in C++ and Java, which will be required to develop the application and the backend components. We have eight students working on our team. Any fewer number of students would not be efficient for this project because it will not be feasible to develop different components of the program and communicate its purpose and functionality with fewer than eight students.

- Our App offers manager a histogram which updates every minute.
  1. This histogram accumulates students' demands of each buses in every minute.
  2. Each station should have a histogram.
  3. Our app will analyze the demands and send manager notifications.
  4. We assume a bus can take 40 students

     Articulated Bus Seating Capacity: 62 (New Flyer D60LF)

Normal Bus Seating Capacity: 41 (Eldorado-ca Axess)

5. If more than 40 students request a certain bus route, the histogram will turn to yellow(warming).

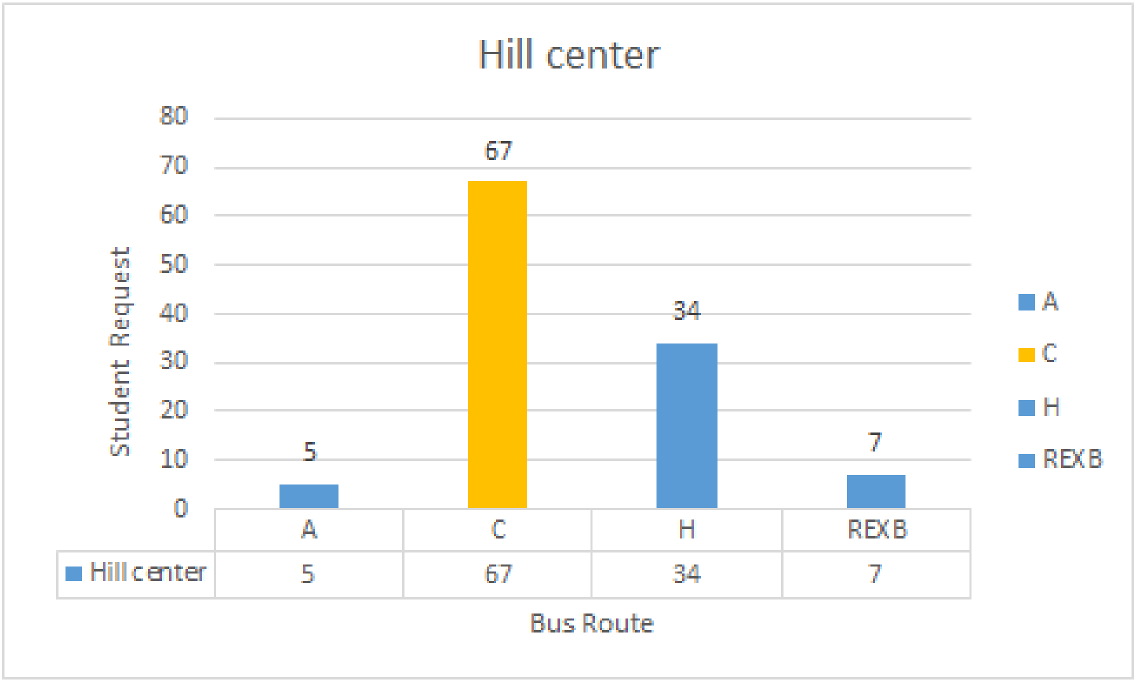
6. If the histogram maintains yellow for 10 mins, our app will send manager a notification about considering sending extra buses.

7. If more than 80 students request a certain bus route, the histogram will turn to red(urgent), which means that manager need to send extra bus immediately.

- Each student is allowed to send request one time.
  1. Students can resend the request if they do not get on the bus.
  2. Students can cancel or change the request any time.

For example:
The histogram of Hill center for manager.

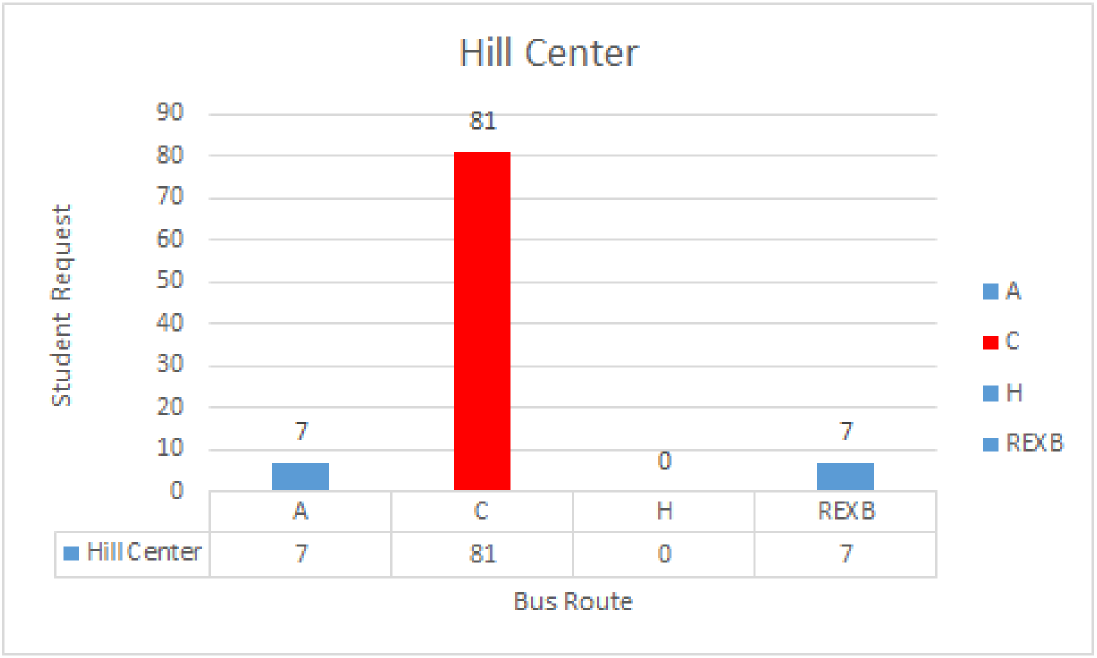
Time 2:59 pm



	A	C	H	REX B
Hill center	5	67	34	7

- According to the histogram, the manager notices that many students request C and H bus because their classes are almost over and they try to go to stadium west lot and College ave. campus.
- The C bus is shown in yellow.

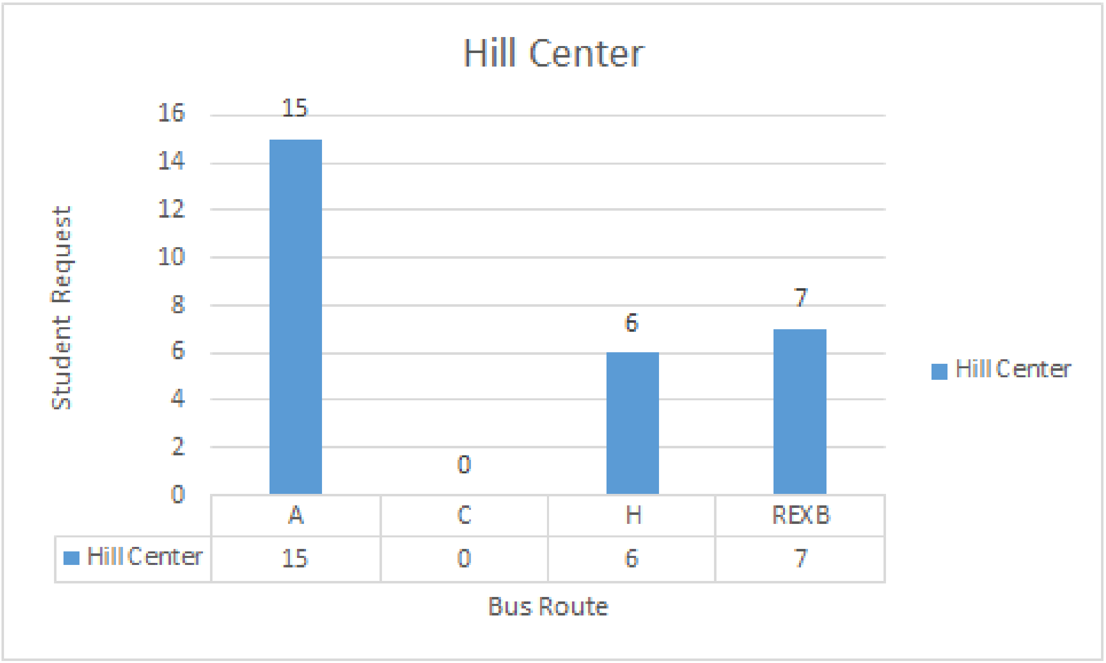
Time 3:00 pm



	A	C	H	REXB
Hill Center	7	81	0	7

- After 1 min, more and more students request C bus and it becomes red(urgent).
- Our app will remind manager to send extra bus immediately.
- A H bus just arrive and our app automatically clear all the request of H bus.

Time 3:01 pm

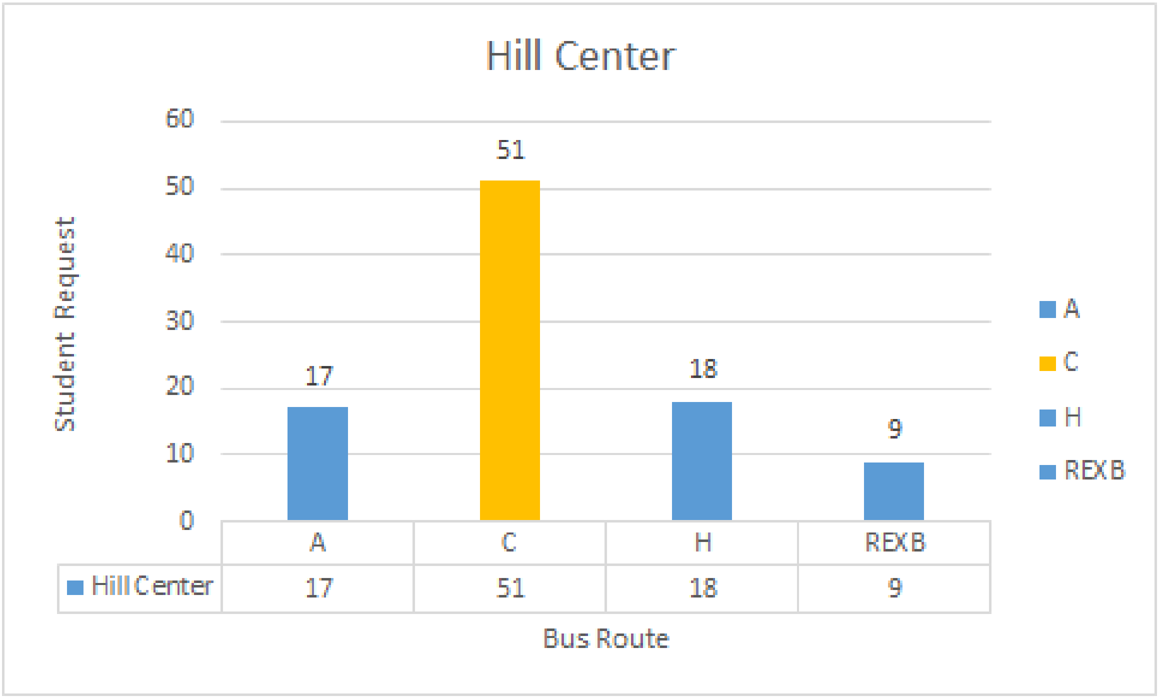


	A	C	H	REXB
Hill Center	15	0	6	7

- When it is 3:01 pm, there are several students who do not get on bus due to the full bus capacity.
- Therefore, they are allowed to resend the request.

- Our app can approach this by combining real-time bus schedule. After the system locates that H bus leaves Hill Center, it will allow all students to send request of H bus at Hill Center station.
- Also, the a C bus is arriving and all the demands are cleared.

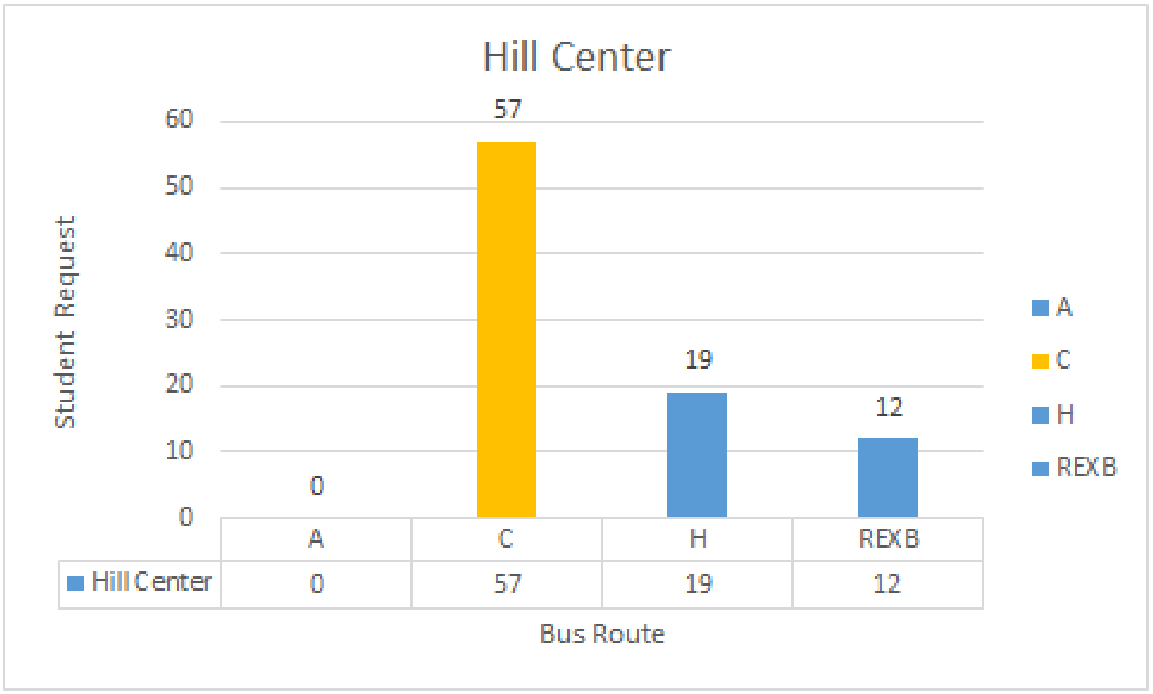
Time 3:02 pm



	A	C	H	REX B
Hill Center	17	51	18	9

- Apparently, one C bus can not take all the students.
- The students who do not get on C bus resend the request.
- There are some students request other buses, but it is not a high demand.

Time 3:03 pm

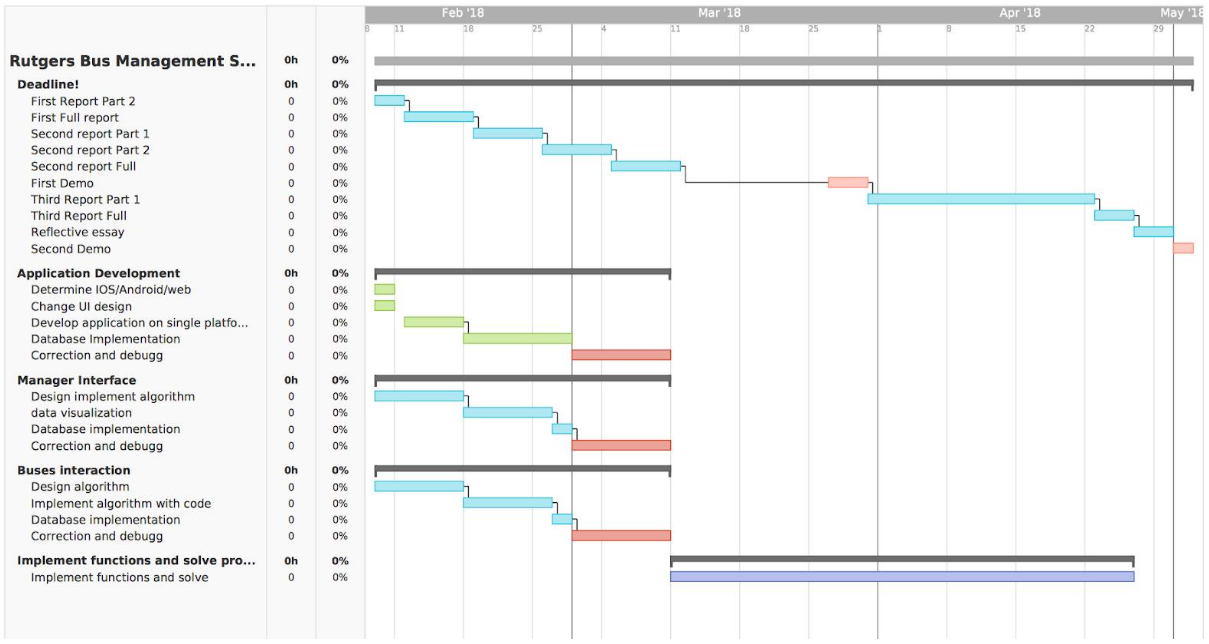


	A	C	H	REXB
Hill Center	0	57	19	12

- At 3:03 pm, A bus arrives and all demands are cleared.
- The C bus is still highly demanded, the manager should make decision of sending extra buses by combing the bus schedules and students request.

Project Management

Yaocheng Tong are responsible and the only person in this group to submit reports to Sakai. All team members' name,email and Github pages are listed at the end of report. Github project link is attached at the cover page. Team members are communicating with "Wechat" application, and we have in-person meeting once a week, "Wechat" voice group meeting twice a week.



Rutgers Bus Management S...	**0h**	**0%**
Deadline!	**0h**	**0%**
First Report Part 2	0	0%
First Full report	0	0%
Second report Part 1	0	0%
Second report Part 2	0	0%
Second report Full	0	0%
First Demo	0	0%
Third Report Part 1	0	0%
Third Report Full	0	0%
Reflective essay	0	0%
Second Demo	0	0%
Application Development	**0h**	**0%**
Determine IOS/Android/web	0	0%
Change UI design	0	0%
Develop application on single platfo...	0	0%
Database Implementation	0	0%
Correction and debugg	0	0%
Manager Interface	**0h**	**0%**
Design implement algorithm	0	0%
data visualization	0	0%
Database implementation	0	0%
Correction and debugg	0	0%
Buses interaction	**0h**	**0%**
Design algorithm	0	0%
Implement algorithm with code	0	0%
Database implementation	0	0%
Correction and debugg	0	0%
Implement functions and solve pro...	**0h**	**0%**
Implement functions and solve	0	0%

References

1. Marsic, Ivan. 2012. *Software Engineering*.
2. *Google Maps*. Feb. 4, 2018. Retrieved from: https://www.google.com/maps
3. *Rutgers University-New Brunswick/Piscataway: Intercampus Bus Schedule.* Feb. 4, 2018. Retrieved from: rudots.rutgers.edu/campusbuses.shtml
4. *Nextbus API*. Feb. 4, 2018. Retrieved from: http://api.rutgers.edu
5. *Rutgers Campus Buses.* Feb. 5, 2018. Retrieved from: https://en.wikipedia.org/wiki/Rutgers_Campus_Buses

Member List

1. Yaocheng Tong

tongyaocheng@gmail.com

https://github.com/YaochengTong

2. Yufeng Lin

jimlin1996@gmail.com

https://github.com/YufengL

3. Viraj Patel

viraj.patel321@gmail.com

https://github.com/virajpatel321

4. Minghao Qin

qmh603311680@icloud.com

https://github.com/MinghaoQin

5. Yuhai Zhang

stevenzhang97@gmail.com

https://github.com/Stevenzyh

6. Lu Jin

lulufixedgear@gmail.com

https://github.com/lulufixedgear

7. Haowei Li

hwli12138@gmail.com

https://github.com/HaoweiL

8. Aaron Hu

ajh193@scarletmail.rutgers.edu

https://github.com/Melancholiax