**NOVA**

**Software Requirements Specification (SRS)**

**Version: 1.0**

**April 16, 2018**

**Mark Kirschenman  
Nathaniel Fleet  
John Hedman  
Jacob Gordon**

**TabContents**

[1. Introduction……………………………………………………………………………………... 3](#_gjdgxs)

[1.1](#_30j0zll) Purpose………………………………………………………………………………….. 3

[1.2](#_1fob9te) Scope……………………………………………………………………………………. 3

[1.3](#_3znysh7) Definitions, acronyms, and Abbreviations……………………………………………… 3

[1.4](#_2et92p0) References Written…………………………………………………………………….... 4

[1.5](#_tyjcwt) Overview……………………………………………………………………………….. 4

[2. Overall Description……………………………………………………………………………... 5](#_3dy6vkm)

[2.1](#_1t3h5sf) Product Perspective……………………………………………………………………... 5

[2.1.1](#_4d34og8) System Interfaces………………………………………………………………….. 5

[2.1.2](#_2s8eyo1) User Interfaces…………………………………………………………………….. 5

[2.1.3](#_17dp8vu) Hardware Interfaces……………………………………………………………….. 5

[2.1.4](#_3rdcrjn) Software Interfaces………………………………………………………………... 5

[2.1.5](#_26in1rg) Communications Interfaces………………………………………………………... 6

[2.1.6](#_lnxbz9) Memory Constraints………………………………………………………………..6

[2.1.7](#_35nkun2) Operations…………………………………………………………………………. 6

[2.1.8](#_1ksv4uv) Site Adaptation Requirements…………………………………………………….. 6

[2.2](#_44sinio) Product Functions………………………………………………………………………. 6

[2.3](#_2jxsxqh) User Characteristics…………………………………………………………………… 6

[2.4](#_z337ya) Constraints…………………………………………………………………………….. 7

[2.5](#_3j2qqm3) Assumptions and Dependencies………………………………………………………... 7

[3. Specific Requirements………………………………………………………………………….. 8](#_2xcytpi)

[3.1 External Interface Requirements……………………………………………………………. 8](#_1ci93xb)

[3.1.1](#_3whwml4) User Interfaces…………………………………………………………………….. 8

[3.1.2](#_2bn6wsx) Hardware Interfaces……………………………………………………………….. 9

[3.1.3](#_qsh70q) Software Interfaces………………………………………………………………... 9

[3.1.4](#_3as4poj) Communications Interfaces……………………………………………………….. 9

[3.3.2](#_147n2zr) Cancel Order……………………………………………………………………... 12

[3.3.3](#_23ckvvd) Add Billing Information………………………………………………………….. 13

[3.3.4](#_32hioqz) Edit Billing Information………………………………………………………….. 13

[3.3.5](#_41mghml) Edit Customer Profile…………………………………………………………….. 14

[3.3.6](#_vx1227) View Customer Profile…………………………………………………………... 15

[3.3.7](#_1v1yuxt) Create Customer Profile/Account………………………………………………... 15

[3.3.8](#_2u6wntf) View Previous Trip History………………………………………………………. 16

[3.3.9](#_3tbugp1) Message Customer……………………………………………………………….. 16

[3.3.10](#_nmf14n) View Work Schedule/Hours…………………………………………………….... 17

[3.3.11](#_1mrcu09) Place Order……………………………………………………………………….. 18

[3.3.12](#_2lwamvv) Edit Preferred Availability……………………………………………………….. 19

[3.3.13](#_3l18frh) Submit Shift Switch Form……………………………………………………….. 19

[3.3.14](#_4k668n3) View Pickup Schedule………………………………………………………….... 20

[3.3.15](#_1egqt2p) Notify System/Manager that customer arrived………………………………….. 20

[3.3.16](#_2dlolyb) Notify System of Departure and Arrival…………………………………………. 21

[3.3.17](#_sqyw64) Edit Driver Profile……………………………………………………………….. 21

[3.3.18](#_3cqmetx) Edit Employee Pay Rates………………………………………………………... 22

[3.3.19](#_1rvwp1q) View Employee Pay Rates……………………………………………………….. 23

[3.3.20](#_4bvk7pj) View Driver Profile………………………………………………………………. 23

[3.3.21](#_2r0uhxc) Terminate Drivers………………………………………………………………... 24

[3.3.22](#_1664s55) Adjust Schedule Manually………………………………………………………. 24

[3.3.23](#_3q5sasy) View Bus Rates………………………………………………………………….. 25

[3.3.24](#_25b2l0r) Place Buses in Maintenance Mode………………………………………………. 25

[3.3.25](#_kgcv8k) Place Order (from Phone Orders).......................................................................... 25

[3.3.26](#_34g0dwd) Edit Orders (from Phone Orders)............................................................................ 26

[3.3.27](#_1jlao46) Request Report (Daily, Weekly, Monthly, Yearly)................................................. 26

[3.3.28](#_43ky6rz) Edit Bus Rates……………………………………………………………………. 27

[3.3.29](#_2iq8gzs) Create Driver Profile……………………………………………………………... 28

[3.4](#_xvir7l) Performance Requirements……………………………………………………………. 29

[3.5](#_3hv69ve) Design Constraints……………………………………………………………………. 29

[3.6](#_1x0gk37) Software System Attributes………………………………………………………….... 29

[3.61](#_4h042r0) Reliability…………………………………………………………………………... 29

[3.6.2](#_2w5ecyt) Availability………………………………………………………………………. 29

[3.6.3](#_1baon6m) Security ………………………………………………………………………….. 29

[3.6.4](#_3vac5uf) Maintainability…………………………………………………………………... 30

[3.6.5](#_2afmg28) Portability………………………………………………………………………... 30

[3.7](#_pkwqa1) Logical Database Requirements………………………………………………………. 30

[3.8](#_39kk8xu) Other Requirements………………………………………………………………….... 31

# 1. Introduction

## Purpose Written by: Nathaniel Fleet, Proof Read by: Jonathan Hedman

The purpose of this document is to give a detailed outline of the specifications and requirements of Nova Bus Booker. It will list dependencies, scope and functionality of the project, user interface and external-system interactions, and serve as a reference for customer approval during the initial development.

## Scope Written by: Nathaniel Fleet, Proof Read by: Jonathan Hedman

Nova Bus Booker is a web-based application that serves as a management system for a bus transit company. Its primary function is to take orders from customers, and then create a pickup/drop-off schedule that the drivers will use.

The system will utilize a database to store customer, manager, and driver information. The system, being web-based, will require an internet connection to function for all users. It incorporates third-party GPS technology in addition to the database. All users will create an account that will be stored on the database, with the exception of “Guest” users. Information associated with an account includes name, phone number, address, and billing information will also be stored on the database.

Nova Bus Booker also provides useful managerial tools, such as logging employee hours, editing driver records, managing bus maintenance, and report generation. Additionally, manager functions will include the ability to manually edit the schedule provided by the system, and input orders taken from phone calls.

## Definitions, acronyms, and Abbreviations Written by: Nathaniel Fleet, Proof Read by: Jonathan Hedman

|  |  |
| --- | --- |
| User | Any human being that interacts with the system |
| Manager | An administrator account, person in charge of running the system |
| Driver/Employee | Bus operator who uses the schedule |
| GPS | Global Positioning System |
| Stakeholder or Customer | Person involved with product that is not part of the development team |
| DBMS | Database Management System |
| Guest | A account (used for order-placing) that does not require login credentials |
| RAM | Random Access Memory, used by computer to store frequent memory |

## References Written by: Nathaniel Fleet, Proof Read by: Jonathan Hedman

[1] IEEE Software Engineering Standards Committee, “IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications”, October 20, 1998.

## Overview Written by: Nathaniel Fleet, Proof Read by: Jonathan Hedman The rest of this document contains 2 chapters. The second chapter will provide an overview of the entire system, and describe its intended functionality as well as its interactions with other systems. It will describe user characteristics, different levels of interface, and design constraints.

The third chapter will describe the system requirements and interfaces in greater detail. It will provide use-case descriptions and diagrams for the functionality, performance requirements and description. This chapter will also describe all of the system functions in great detail, including functionality and proper use.

# 2. Overall Description

This section will provide an overview of the entire system, and explain how the system created will interact with other external systems. It will also introduce the functionality of the system at a basic level. Finally, it will describe constraints as well as assumptions made and dependencies.

## Product Perspective Written by: Nathaniel Fleet, Proof Read by: Jonathan Hedman

This system is built to be a web-based application, meant to be able to work on most platforms that have access to a web browser. Its primary job will be to take orders from customers, and use them to create a bus schedule that the drivers will implement. The system will require interaction with a GPS system as well as a database.

### System Interfaces Written by: Nathaniel Fleet, Proof Read by: Jonathan Hedman, Revised by: Jacob Gordon

The system is a web-based application, so it is made to run in a variety of environments, as long as this environment is able to reach the application through a web browser. Google Maps will be used as a 3rd party software to aid in the use of the application when placing orders and running routes. The system will be be created to interact with Google Maps in order to create the best route for the buses. This means that the algorithm will use Google Maps as an input and create an optimized route for an output.

### User Interfaces Written by: Nathaniel Fleet, Proof Read by: Jonathan Hedman

The system will have four different main interfaces. First of the four is the login screen. Here the user will enter in their credentials to start using the system, or create an account for future use. From this login screen, the system will direct the user to the appropriate page based on their account privileges.

Customers will be directed to the customer user interface. Here they can make an order, change their account information such as billing information, and see their previous trips they have taken. They will also be able to view any current trip information such as confirmations or messages from the bus driver.

Drivers will be directed to the driver user interface. Here they can view schedule and routes, contact customers, see pay, and edit and update their personal information. They also have an option to contact the manager to request hours and update manager on routes.

Managers will be directed to the manager user interface. Here they will be able to perform any of the manager duties. These include creating an order, generating reports, editing or viewing driver information, adjust scheduling, change rates, and placing buses into maintenance mode.

### Hardware Interfaces Written by: Jacob Gordon, Proof Read by: Jonathan Hedman

The system will display each user interface to the users monitor or mobile phone screen. When using the application on a computer, it will recognize mouse and keyboard inputs. On a mobile phone, the phone's built in keyboard will be recognized, and touching and dragging will be recognized as replacement for mouse inputs.

### Software Interfaces Written by: Mark Kirschenman, Proof Read and Revised by: Jonathan Hedman

The customer should be able to easily interact with the system using most available operating systems. The system application should be compatible with Windows 7 and up, Mac OS X and up, as well as Android Lolipop and up, and Apple IOS OS X and up. The system should also be able to interact with google maps in order to locate a customer's position of pickup and whether they are within the bounds of the services.

The purpose for interacting with Windows and Mac OS X and up is that this will enable customers, drivers and managers to interact with the system online. This is crucial because some people may not be able to interact with the system using a smartphone or tablet.

The app will also work well on Android and IOS because many people use their phones while they are away from home and would prefer to interact with our application via a mobile device. We need to consider both desktops and mobile devices in order to not alienate potential customers.

The app will run on a web application that will be supported on most modern web browsers that support the latest versions of HTML and Javascript. Though most of these modern web browsers will work, the application will be guaranteed to work on Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari. The application will work best on the latest version of these browsers but the application will work on any version newer than the following: Chrome 62.0.3202, Firefox 55.0.1, Microsoft Edge 17035, Safari 11.0.1. It is also important to specify that these versions apply to Windows, Mac OS, Android, and IOS and all the operating systems will be able to run the application on all of the web browsers.

The application will use MySQL as the Database Management System. MySQL version 5.7 will be used as the supported version for the web application, this is the newest version of MySQL. This version will allow the team to build the database from the ground up to support the needs of the customer. MySQL will allow the team to control the security and reliability of the system. The customer’s data is very important to keep secure and the uptime of the database is very important as well.

### Communications Interfaces Written by: Mark Kirschenman, Proof Read and Revised by: Jonathan Hedman

The users must be able to connect to the web application through the internet. Once connected to the website, the server will then be able to communicate with the user through a client-server design. The server will then be able to complete any data lookups by communicating with the Database Management System (MySQL). The DBMS will be in charge of keeping data integrity and data security and responding back to the server in a timely manner. While the users may not physically see this process, it is important to make sure that the communications between the server and the database are done in a fast, yet safe manner.

### Memory Constraints Written by: Jonathan Hedman, Proof Read: Jacob Gordon

The system will require a minimum of one gigabyte of RAM to operate appropriately. Currently, no memory will be required to be stored on user’s system.

### Operations Written by: Mark Kirschenman and Jonathan Hedman, Proof Read: Nathaniel Fleet

No special operations will be required for the customer. They should access the webpage to utilize application functionality. Follow the instructions given to create an account, request a ride, and contact us if necessary. After user is done user will exit the web app.

For the Driver, they need to be able to login, and be able to access all the available driver features.

For the Manager, they should be able to login, and access the manager features.

### Site Adaptation Requirements Written by: Jonathan Hedman, Proof Read; Nathaniel Fleet

Requirements for data or initialization sequences are not necessary. Features that will be modified will due to a particular installation will be decided once the different installations are made. Customer features that would be affected include but are not limited to: making an order, changing an order, canceling an order, editing billing information, and creating an account, or modifying one. The features for the drivers and managers will also be also be limited or modified.

## Product Functions Written by: Jacob Gordon, Revised by: Mark Kirschenman, Proof Read by: Jonathan Hedman

The software is designed for the following functions:

* Customers to create an order, through a registered account or a guest account
* Customers to modify or cancel an existing order
* Customers to edit and receive notifications from the server and from the driver based on their currently placed orders
* Customers to create their own user profile for the system to remember the by
* Customer to edit their current user profile as well as add payment methods to the the account
* Customers to view and edit current billing information as well as view trip history
* Drivers to view their hours and weekly schedule created by the manager
* Drivers to notify customers and managers of arrival and departure
* Drivers to edit their availability of schedule
* Drivers to edit and view their profile
* Managers to view reports that are generated by the system
* Managers to view driver's availability times
* Managers to create a schedule for drivers or allow the system to automatically compile one
* Managers to place buses in maintenance mode
* Managers to add or delete existing drivers and their information
* Managers to adjust employee pay rates and bus fare rates
* Managers to manually create orders from the phone

## User Characteristics Written by: Mark Kirschenman and Jonathan Hedman, Proof Read by: Nathaniel Fleet

Customers may have minimal computer experience, but should at least have the ability to access a website and basic follow instructions. User-training should not be necessary, as long as the customer is capable of following on-screen instructions. The system should take care of most user interactions in a friendly way, so the user does not need extensive expertise. If the customer can access the internet and the site, they should be able to use it.

Drivers may also have minimal computer experience, but a small amount of training is necessary to ensure that employees are proficient in the system’s intended use. The system will handle the main part of the driver’s functions, driver-users just need to know how to interact with it properly.

Managers should have a baseline experience level with computers. As the administrator of the system, the extensive training should be provided to the assigned manager in the use of customer, driver, and the manager’s own functions. As many of the function augment and streamline standard managerial operations, manager-users should be aware of how these work manually to understand the system.

## Constraints Written by: Mark Kirschenmen, Proof Read by: Jonathan Hedman

No current restraints on the system currently, because the customer had no specific required constraints. Although, constraints may show up further down the project and programmers should be aware of potential changes due to constraints.

## Assumptions and Dependencies Written by: Jonathan Hedman, Proof Read and Revised by: Jacob Gordon

Assumptions:

* Users can navigate through the simple web application user interface
* The database will be backed up daily during slower operations of business
* The users are in an area with good internet signal strength in order to receive and processes data

Dependencies:

* The areas that the service can be used may change in the future
* Needed software updates in the future

# 

# 3.

# 3. Specific Requirements

## 3.1 External Interface Requirements Written by: Jacob Gordon, Proof Read by: Jonathan Hedman

**Inputs:**

* Login Credentials
  + Purpose: logging customer, driver, manager into the system. Guards the system from malicious users or other systems.
  + Source: Customer, driver, or manager
  + Relationships: The value of this input changes the output from the system by specifying if the user is a customer, driver, or manager. It also allows for the system to identify if the user is available for certain permissions.
* Order Info
  + Purpose: To place an order.
  + Source: Customer or manager
  + Relationships: The value of this input sets an order to be added to the bus route
* Billing Info
  + Purpose: To provide payment information for an order.
  + Source: Customer
  + Relationships: This information will be used to complete the transaction for the services provided, and also provides security to the company from fraudulent orders.
* Function Selection
  + Purpose: For the user to select what functionality of the software they would like to utilize.
  + Source: Customer, driver, or manager
  + Relationships: Based on this click input from the user, the interface for the functionality selected will be displayed.

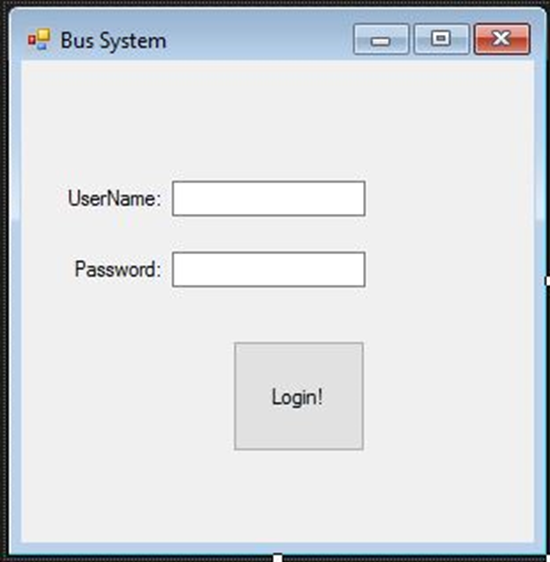
**Outputs:**

* Function Output
  + Purpose: Displaying output of the selected function to the user.
  + Source: Customer, driver, or manager
  + Relationships: This output is based on previous input from the user, mainly function selection and previous inputs to the database.

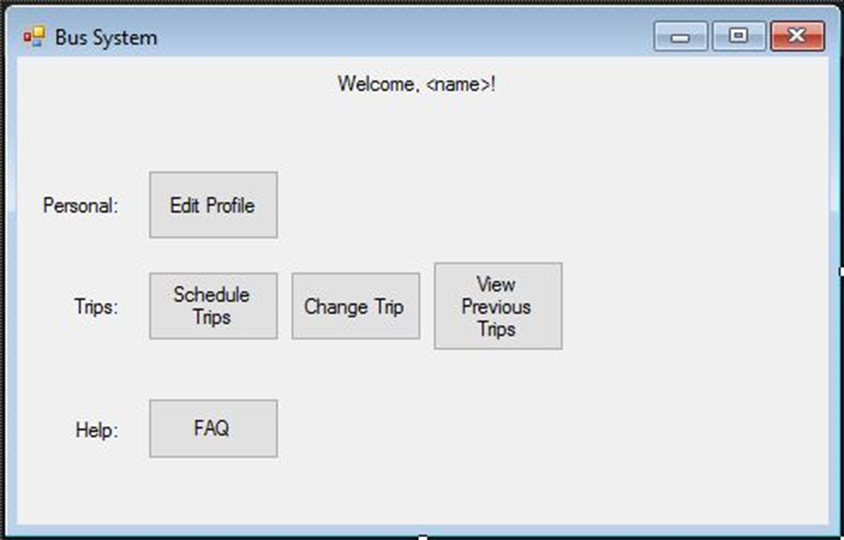
### User Interfaces Written by: Jacob Gordon, Proof Read by: Jonathan Hedman, Revision: Mark Kirschenmann

The user will interact with the system via a mobile device or computer. The interface will require input from a keyboard, mouse, and will be visible on computer screen. While using mobile app the user should be able to use the system. The user interface should be user friendly and simple.

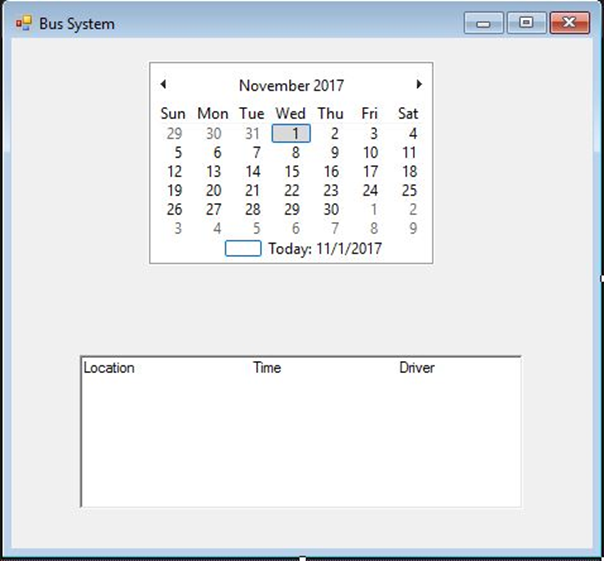
* Login Credentials
  + Purpose: logging customer, driver, manager into the system. Guards the system from malicious users or other systems.
  + Source: Customer, driver, or manager
  + Relationships: The value of this input changes the output from the system by specifying if the user is a customer, driver, or manager. It also allows for the system to identify if the user is available for certain permissions.



* Function Selection
  + Purpose: For the user to select what functionality of the software they would like to utilize.
  + Source: Customer, driver, or manager
  + Relationships: Based on this click input from the user, the interface for the functionality selected will be displayed.



* Function Output
  + Purpose: Displaying output of the selected function to the user.
  + Source: Customer, driver, or manager
  + Relationships: This output is based on previous input from the user, mainly function selection and previous inputs to the database.



### 3.1.2 Hardware Interfaces Written and Revised by: Jacob Gordon, Proof Read by: Jonathan Hedman

* Mouse and Keyboard/Touch Screens
  + Purpose: Receive input from the users to advance the program
  + Source: Customer, driver, or manager
  + Relationships: Based on this input the program will navigate the user to a different part of the program

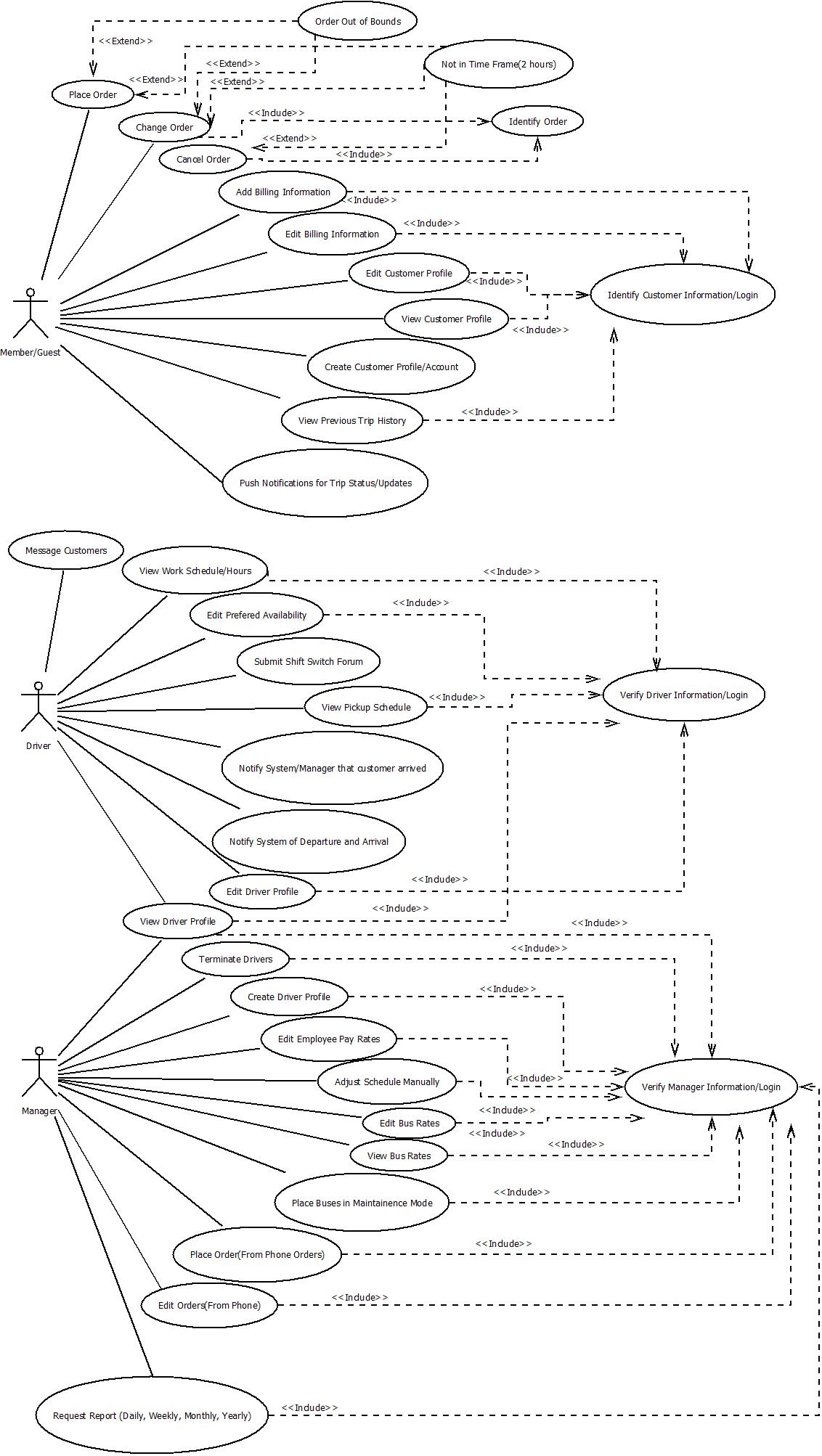
### 3.1.3 Software Interfaces Written and Revised by: Jacob Gordon, Proof Read by: Jonathan Hedman

* Operating System
  + Purpose: Have reliable system to communicate with multiple differing operating systems
  + Source: Customer, driver, or manager
  + Relationships: The operating system may affect how the program interacts with the user of the program
* Web Browser
  + Purpose: Have a uniform way to communicate with multiple users over the internet
  + Source: Customer, driver, or manager
  + Relationships: The web browser may affect how the program interacts with the user of the program

### 3.1.4 Communications Interfaces Written and Revised by: Jacob Gordon, Proof Read by: Jonathan Hedman

* Client/Server Design
  + Purpose: Have a system that can communicate with the users and get their input while also running processes in the background to not have to rely on their computers computing power
  + Source: Client and our server
  + Relationships: The program relies heavily on this design. This allows the users to give input and it also is the way for the program to complete tasks in the background while the customer is doing their own tasks. This is also the main source of output to the users.
* Database Management System
  + Purpose: Have a system that keeps track of data and also sustains data integrity, security, and reliability of the data base.
  + Source: Our servers/database
  + Relationships: Our server is the only thing that will be able to access the database. The users can request information from the database but the server will actually fetch the information and relay it to the user. This allows the data to be secure. The database is also essential to run the program. All of the information that is needed for the program to run will be located on the database.

**Functional Requirements (Software Product Features)**

**3.2 Use Case Model** Jonathan Hedman and Mark Kirschenman created **Use Case Model**

### 

### 

### 

### Member/Guest descriptions: Jonathan Hedman, Driver: Mark Kirschenman, Manager: Jacob Gordon. Sequence Diagrams made by whole team. Revisions were made by Nate Fleet and Jacob Gordon

### 3.3.1 Use Case 1: Change Order

#### 3.3.1.1 Use case description

Name: Change Order

Initiator: Member/Guest

Goal: Change an order that a customer made previously

Main Success Scenario

1. Customer requests for the system to change an order

2. Include Identify Order

3. System checks if the order is valid to change

4. Customer changes the needed information

5. System checks if information is valid to process

6. System changes information in database

7. System re-runs algorithm

8. System notifies customer that the order has been placed and gives details on changed order

Extensions

3. Not within time frame (2 hours)

a. System notifies that the pick-up times is within 2 hours and can’t be changed

b. fail

5. Order out of bounds

a. System notifies customer that the address they provided is out of bounds

b. System brings customer back to step 3

5. Not within time frame (more than 2 hours)

a. System notifies customer that the time they provided is in less than 2 hours

b. System brings customer back to step 3

### Use Case 2: Cancel Order

#### 3.3.2.1 Use case description

Name: Cancel Order

Initiator: Member/Guest

Goal: Cancel an order that a customer has made previously

Main Success Scenario

1. Customer requests for the system to cancel an order

2. Include Identify Order

3. System checks if the order is valid to cancel

4. System deletes the order from the database

5. System notifies the customer that the order was successfully canceled

6. System re-runs algorithm for new schedule

Extensions

3. Not within time frame (2 hours)

a. System notifies that the pick-up times is within 2 hours and can’t be changed

b. System returns customer back to the main menu

### Use Case 3: Add Billing Information

#### 3.3.3.1 Use case description

Name: Add Billing Information

Initiator: Member/Guest

Goal: Add billing information to member/guest account

Main Success Scenario

1. Include Identify Customer Information/Login

2. Customer requests to add billing information to the system

3. System asks customer to fill out forum for new billing information

4. Customer fills out forum

5. Customer submits forum to system

6. System checks if the information is valid

7. System saves information into database

8. System notifies the customer that the information was successfully saved

Extensions

6. Information entered was invalid

a. System notifies customer that the information was invalid

b. System bring customer back to step 3

### Use Case 4: Edit Billing Information

#### Use case description

Name: Edit Billing Information

Initiator: Member/Guest

Goal: Edit customer billing information that was previously saved in the database

Main Success Scenario

1. Include Identify Customer Information/Login

2. Customer requests to edit billing information to the system

3. System asks customer to fill out forum for changing billing information

4. Customer fills out forum

5. Customer submits forum to system

6. System checks if the information entered is valid

7. System save changed information to the database

8. System notifies customer that the information was successfully changed

Extensions

6. Information entered was invalid

a. System notifies customer that the information was invalid

b. System bring customer back to step 3

### Use Case 5: Edit Customer Profile

#### Use case description

Name: Edit Customer Profile

Initiator: Member/Guest

Goal: Edit current customer profile

Main Success Scenario

1. Include Identify Customer Information/Login

2. Customer requests to edit their profile

3. System requests customer to fill out forum to change profile information

4. Customer fills out forum and submits it to the system

5. System checks if the information is valid

6. System saves edited information into the database

7. System notifies customer that the information was successfully edited

Extensions

5. Information entered was invalid

a. System notifies customer that the information was invalid

b. System bring customer back to step 3

### Use Case 6: View Customer Profile

#### Use case description

Name: View Customer Profile

Initiator: Member/Guest

Goal: Allow customer to view their profile

Main Success Scenario

1. Include Identify Customer Information/Login

2. Customer requests to view their profile

3. System retrieves customer information from the database

4. System presents the customer their profile in a new page

5. Customer views the information and selects when to close the information

### Use Case 7: Create Customer Profile/Account

#### Use case description

Name: Create Customer Profile/Account

Initiator: Member/Guest

Goal: Create a customer account for customer

Main Success Scenario

1. Customer requests to create a new account

2. System provides the customer with a forum to fill out for new accounts

3. Customer fills out the forum and submits it to the system

4. System verifies if the information is valid

5. System stores new account information inside of the database

6. System notifies customer that the new account was created

Extensions

4. Information entered was invalid

a. System notifies customer that the information was invalid

b. System bring customer back to step 2

### Use Case 8: View Previous Trip History

#### Use case description

**Name:** View Previous Trip History

**Initiator:** Member

**Goal:** Allow member to see their previous trip history

Main Success Scenario

1. Include Identify Customer Information/Login

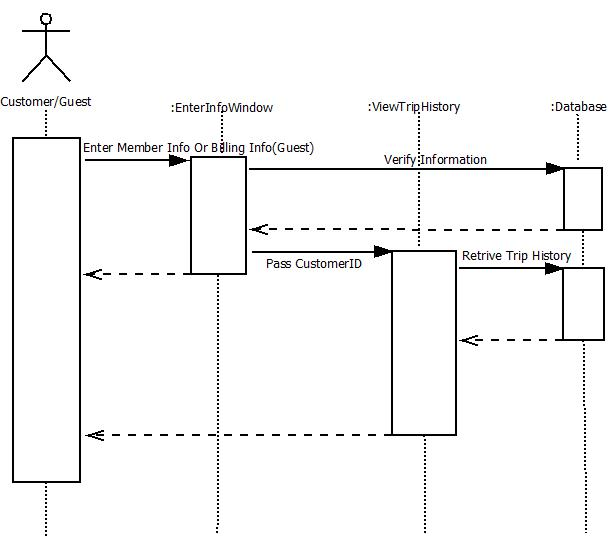
2. Customer requests to view previous trip history

3. System fetches trip history from database

4. System displays trip history to the customer

5. Customer views the information and selects when to close the information

**3.3.8.3 Sequence Diagram**

****

### Use Case 9: Message Customer

#### Use case description

Name: Message customer

Initiator: Driver

Goal: send a simple message to the customer

Main Success Scenario

1. Driver requests to message the customer

2. System confirms that the customer exists in customer accounts

3. Driver enters a message

4. System relays that message to the customer using customer contact information

### Use Case 10: View Work Schedule/Hours

#### Use case description

Name: View Work Schedule/Hours

Initiator: Driver

Goal: Access the work schedule to view the hours

Main Success Scenario

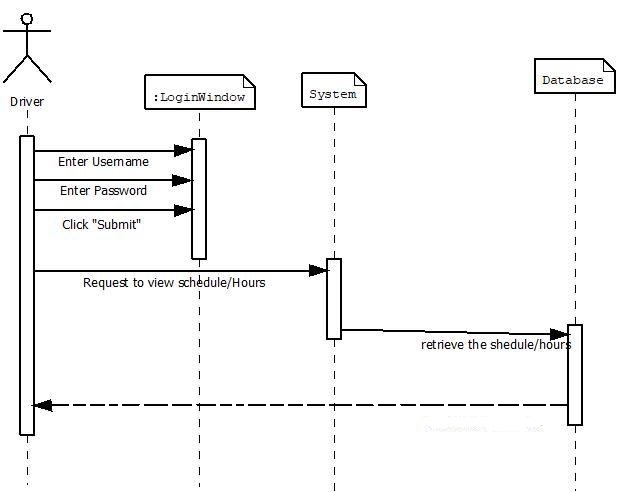
1. Include Driver Information/Login

2. Driver requests to view schedule/Hours

3. System approves request

4. System displays the drivers schedule from schedule data

**3.3.10.3 Sequence Diagram**

****

### Use Case 11: Place Order

#### Use case description

Name: Place orders

Initiator: Customer

Goal: Customer places an order and the system adds it to the routes

Main success scenario

1. Include Identify Customer Information/Login

2. Customer requests to place an order

3. System asks for customer to fill out information about the order

4. Customer enters location, date, and time for the wanted order

5. System verifies the time is two hours away

6. System verifies the location is available

7. System adds the order to the current schedule

8. System sends a confirmation email to the customer

Extensions

5. Order is not in time frame(2 hours)

a) The system tells the user that the order must be placed at least two hours in advanced

b) The system brings the customer back to step 3

6. Order is out of bounds

a) The system tells the user that the order must be in Brookings, to Sioux Falls, or coming from Sioux Falls

### Use Case 12: Edit Preferred Availability

#### Use case description

Name: Edit Preferred Availability

Initiator: Driver

Goal: Submit preferred availability

Main Success Scenario

1. Include Driver Information/Login

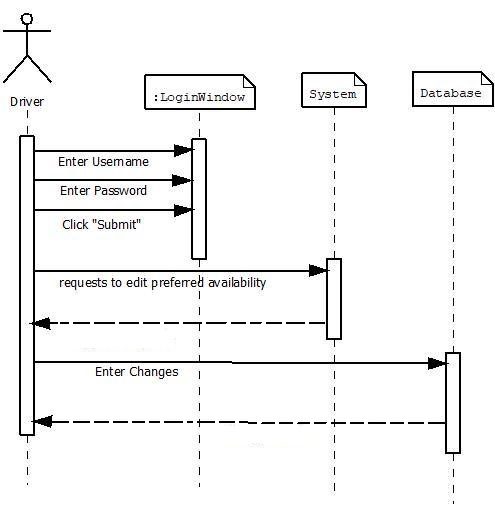
2. Driver requests to edit preferred availability

3. System approves request

4. Driver enters changes

5. System changes the information

**3.3.13.2 Sequence Diagram**

****

### Use Case 13: Submit Shift Switch Form

#### Use case description

Name: Submit Shift Switch Form

Initiator: Driver

Goal: Submit a shift switch form and send the message to the manager

Main Success Scenario

1. Include Identify Customer Information/Login

2. Driver requests to submit a shift switch form

3. System approves request

4. Driver enters what they want to switch

5. System relays a message to manager of switch

### Use Case 14: View Pickup Schedule

#### Use case description

Name: View Pickup Schedule

Initiator: Driver

Goal: View route and pickup schedule

Main Success Scenario

1. Include Driver Information/Login

2. Driver requests to view the pickup schedule

3. System retrieves driver’s information from the database

4. System displays the schedule

### Use Case 15: Notify System/Manager that customer arrived

#### Use case description

Name: Notify System/Manager that customer arrived  
Initiator: Driver  
Goal: to notify the manager that the customer has arrived

Main Success Scenario  
1. Driver requests to notify manager that customer arrived for current order  
2. System approves request  
3. System sends message to the manager

### Use Case 16: Notify System of Departure and Arrival

**3.3.16.1 Use case description**

Name: Notify System of Departure and Arrival

Initiator: Driver

Goal: notify the system that we arrived, notify the system we departed

Main Success Scenario

1. Driver notifies system we arrived for the current order

3. System changes a status value to show we arrived

4. User notifies the system we departed

6. System changes a status value to show we departed

### Use Case 17: Edit Driver Profile

**3.3.17.1 Use case description**

Name: Edit Driver Profile

Initiator: Driver

Goal: to edit the driver profile

Main Success Scenario

1. Include Driver Information/Login

2. Driver requests to edit their profile

3. System approves editing the profile

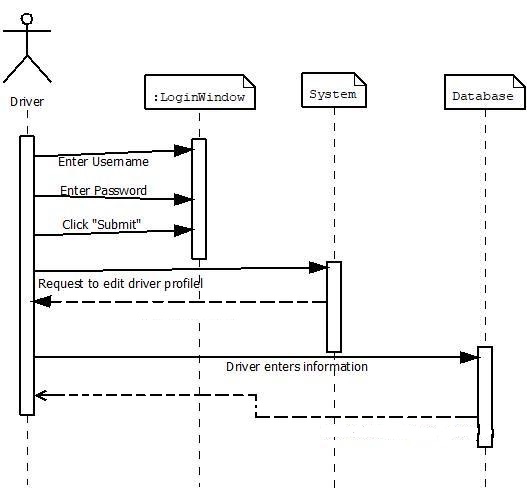
4. Driver chooses information to edit

5. Driver enters new information

6. System changes the information in the profile

7. System sends a success reply

**3.3.17.2 Sequence Diagram**

****

### Use Case 18: Edit Employee Pay Rates

**3.3.18.1 Use case description**

Name: Edit Employee Pay Rates

Initiator: Manager

Goal: Edit the employee Pay Rates

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager requests to edit pay rates

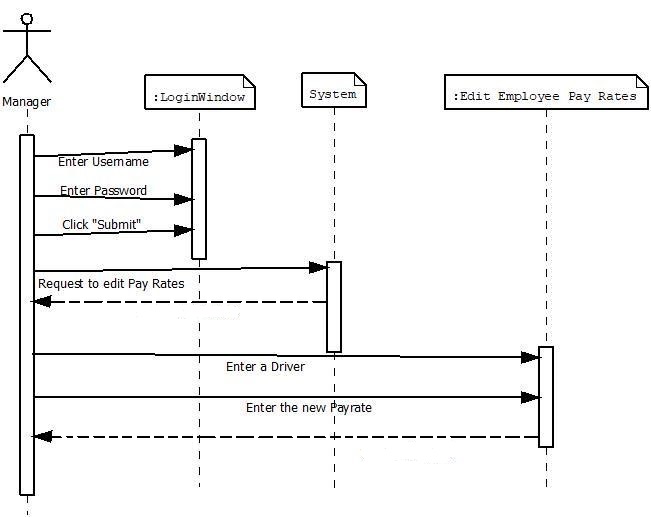
3. System approves

4. Manager enters a driver

5. Manager enters a payrate

6. System changes the payrate in the database

**3.3.18.2 Sequence Diagram**

****

### Use Case 19: View Driver Profile

**3.3.19.1 Use case description**  
Name: View Driver Profile

Initiator: Manager

Goal: View the information on a driver’s profile

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager selects View Driver Profile

3. Manager enters desired drivers name

4. System retrieves the entered driver’s information from the database

5. System displays the driver’s information

### Use Case 20: Terminate Drivers

**3.3.20.1 Use case description**

Name: Terminate Drivers

Initiator: Manager

Goal: Remove a driver from the database

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager selects Terminate Driver

3. Manager enters desired drivers number

4. System finds the entered driver’s information in the database

5. System removes information from the main database

### Use Case 21: Adjust Schedule Manually

**3.3.21.1 Use case description**

Name: Adjust Schedule Manually

Initiator: Manager

Goal: Alter the schedule of routes manually if needed

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager selects Adjust Schedule

3. Manager changes desired route

4. System updates altered route from the manager

5. System updates the customers affected by the changed routes

### Use Case 22: View Bus Rates

**3.3.22.1 Use case description**

Name: View Bus Rates

Initiator: Manager

Goal: View the current bus rates

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager selects View Bus Rates

3. System fetches current bus rates from the database

4. System displays the current bus rates to the manager

### Use Case 23: Place Buses in Maintenance Mode

**3.3.23.1 Use case description**

Name: Place Buses in Maintenance Mode

Initiator: Manager

Goal: Set a bus to be under maintenance

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager selects Place Bus in Maintenance

3. Manager selects the bus to put in maintenance

4. System sets selected bus to maintenance mode

5. System removes the bus from the schedule and updates as needed

### Use Case 24: Place Order (from Phone Orders)

**3.3.24.1 Use case description**

Name: Place Order (From Phone Orders)

Initiator: Manager

Goal: Add an order to the system from an order taken over the phone

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager selects Add Order

3. System asks manager to fill out information on the order

4. Manager enters order details

5. System adds the order to the schedule

### Use Case 25: Edit Orders (from Phone Orders)

**3.3.25.1 Use case description**

Name: Edit Orders (From Phone)

Initiator: Manager

Goal: Edit an order that was placed over the phone

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager selects Edit Order

3. Manager enters identifying order details

4. System retrieves order details from database

5. Manager alters order

6. System updates the order in the database

### Use Case 26: Request Report (Daily, Weekly, Monthly, Yearly)

**3.3.26.1 Use case description**

Name: Request Report(Daily, Weekly, Monthly, Yearly)

Initiator: Manager

Goal: Generates a report from the given time period

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager selects Generate Report

3. Manager selects desired time period

4. System retrieves the information from time period from the database

5. System generates report

6. System displays the report to the manager

### Use Case 27: Edit Bus Rates

**3.3.27.1 Use case description**

Name: Edit Bus Rates

Initiator: Manager

Goal: Change the Current Bus Rate

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager requests to edit current bus rate

3. System fetches current bus rate information from the database

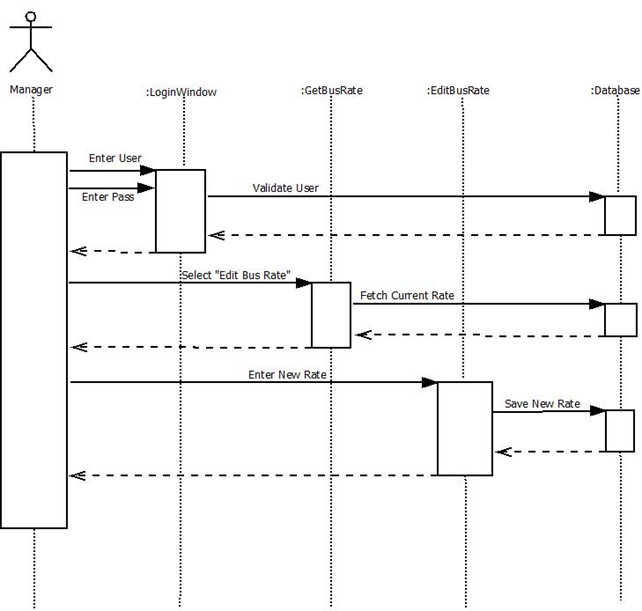
4. System provides the current bus rate information

5. System asks for the new bus rate

6. Manager provides new bus rate

7. System saves the new bus rate information into the database

8. System notifies manager that the bus rate has been updated

**3.3.27.2 Sequence Diagram**

### Use Case 28: Create Driver Profile

**3.3.28.1 Use case description**

Name: Create Driver Profile

Initiator: Manager

Goal: Create a profile for a new driver

Main Success Scenario

1. Include Verify Manager Information/Login

2. Manager selects Create Driver Profile

3. Manager enters new driver’s information

4. System store the entered driver’s information in the database

### 3.3.29 Use Case 29: Identify Customer Information/Login

**3.3.29.1 Use case description**

Name: Identify Customer Information/Login

Initiator: Member

Goal: Log customer into the system to allow for more features

Main Success Scenario

1. Customer requests to log into the system

2. System asks customer to provide Login Username and Password

3. Customer provides the information and submits it to the System

4. The system verifies the information with the database

5. System tells the customer that the login was successful

6. System brings customer to the member main menu

Extensions

4. System can’t find customer information in the database

1. System tells customer that his/her information was not found
2. System brings the customer back to step 2

### 3.3.30 Use Case 30: Verify Driver Information/Login

**3.3.30.1 Use case description**

Name: Verify Driver Information/Login

Initiator: Driver

Goal: Log driver into the system to allow for him/her to access driver’s features

Main Success Scenario

1. Driver requests to log into the system

2. System asks driver to provide Login Username and Password

3. Driver provides the information and submits it to the System

4. The system verifies the information with the database

5. System tells the driver that the login was successful

6. System brings driver to the driver main menu

Extensions

4. System can’t find driver information in the database

1. System tells driver that his/her information was not found
2. System brings the driver back to step 2

### 3.3.31 Use Case 31: Verify Manager Information/Login

**3.3.31.1 Use case description**

Name: Verify Manager Information/Login

Initiator: Manager

Goal: Log manager into the system to allow for him/her to access the manager’s features

Main Success Scenario

1. Manager requests to log into the system

2. System asks manager to provide Login Username and Password

3. manager provides the information and submits it to the System

4. The system verifies the information with the database

5. System tells the manager that the login was successful

6. System brings manager to the manager main menu

Extensions

4. System can’t find manager information in the database

1. System tells manager that his/her information was not found
2. System brings the manager back to step 2

### 3.3.32 Use Case 32: Identify Order

**3.3.32.1 Use case description**

Name: Identify Order

Initiator: Guest/Member

Goal: Guest/Member specifies a specific order for the system to fetch

Main Success Scenario

1. System asks the guest/member to specify the order to fetch from the database

2. Guest/Member provides a order number or last name and time for order

3. System looks for the provided information in the database

4. System then returns the order details to the function that called it (this is never called by itself)

## Performance Requirements Written by:Nathaniel Fleet, Proof Read by: Mark Kirschenman

The software will support enough clients at once to where an overload issue should never occur. Should an overload occur, the software will shutdown and reboot, kicking all current users off temporarily. The exact amount of users supported at once is 500. The amount of transactions the software can handle within a 1 second interval is 10. If this number is exceeded, any transactions past the tenth will be delayed for 10 extra seconds before processing. This should rarely happen, but from the user’s perspective, this will just be a delay in a confirmation message and the software will not become unresponsive. If this delay is applied to the same transaction more than 3 times, an error will be displayed to the user and they will have to re-enter the transaction information.

## Design Constraints Written by: Mark Kirschenman

No design constraints required by customer.

## Software System Attributes Written by: Jacob Gordon, Proof Read by: Nathaniel Fleet

The following is a list of system attributes that can serve as objectively verifiable requirements.

### 3.61 Reliability Written by: Jacob Gordon, Proof Read by: Nathaniel Fleet

The software will be expected to have a maximum of one data error per week of operation. These errors could include not finding a search item on the first attempt, failing to save an item that should have been successful, or showing different data than the user requested on the first attempt.

### Availability Written by: Jacob Gordon, Proof Read by: Mark Kirschenman

Of the 168 hours in a week, the software will be available for 167 of them. One hour will be reserved for maintenance and possible updates. This would be performed at the usual lowest traffic time of the week. Although, this maintenance hour need not be used every week if no maintenance or updates are required.

### Security Written by: Mark Kirschenman, Proof Ready by: Jacob Gordon

The system should securely store the user's account and password through encryption. The system should be aware if someone is trying to access a user’s account without success multiple times within a short period of time. After a user logs in there may be the necessity for a second level of checking, secondary questions to confirm user identity. This is especially necessary for the manager, anyone who can access a manager account could be a big threat to the system. User activity logging should be kept track of and secured. The system should notice if customer places excessive orders in a day, week or month, as this could negatively impact bus service for other customers. This would only be a problem if the customer is scheduling large amounts of orders and not showing up. The modules within the system should only be able to interact with other appropriate connected modules. There should not at any time be a module that can interact with another module incorrectly, this could cause security issues.

### Maintainability Written by: Jonathan Hedman and Mark Kirschenman, Proof Read by: Jacob Gordon

The program will be tested before release to make sure that no critical updates will be needed at the beginning release. If updates and changes are needed, the software will be written in a modular fashion where as each different function is a separate function within the code. So if a change is needed or desired, the code for a particular function will be easy to find and changing this code will not affect the functionality of other functions in the software.

### Portability Written by: Jonathan Hedman, Proof Read by: Mark Kirschenman

The program will be made to with portability in mind. The program should be available on any operating system that is able to open up any of the supporting web browsers. The difficulty of portability will come from porting the desktop web application version to the phone web application version. The phone web application will have to be specifically modified to each mobile operating system.

## Logical Database Requirements Written by: Jonathan Hedman and Mark Kirschenman, Proof Read by: Nathaniel Fleet

The database will include information on:

* Customer, driver, and manager account information such as login and password.
* Customer phone number, address, billing information.
* Customer order history
* Current unfulfilled orders
* Customer status, such as member or guest
* Driver information, such as pay rate, number of hours worked, address, contact information, schedule, and prefered availability
* Driver shift switch forms
* Manager information, such as contact, address, pay rate, hours worked, and schedule
* System generated reports
* Current service locations
* Times of order arrival and departures
* Current Bus Information, if they are in service or unavailable, how many customers it can hold, if it is a handicap bus
* Rates for specific trips and other fees/discounts that can be charged

The database will be frequently used. It will be available to be accessed 100 times a second and it will be designed to give information contained inside of the database instantly as long as the database isn’t accessed more than 100 times in a second, if it is then some delay will be expected. If the information is spread across multiple tables and a data item is updated in one table the data will be updated in all other tables containing that data.

Managers will be related to drivers inside of the database, multiple managers can be in charge of many drivers while the drivers can be managed by one or more managers. Customer data will be related to orders data, customers can have zero to many orders that they are related to, but the orders must be related to only one customer. The manager entity will be related to the report entity and the manager can be related to multiple reports and the reports can be related to multiple managers.

## Other Requirements

None.