**Requirement Document**

CIS 4911 – Senior Project

Virtual Queue

**Member:**

Kenneth Kon

Michael Lazo

**Instructor**

Masoud Sadjadi

**Mentor**

[Bernard Parenteau](http://spws-dev.cis.fiu.edu/senior-project-website-v4/user/231)

**Date**

February 13th 2015

Copyright © Florida International University 2015. All Rights Reserved. No part of the Virtual Queue Project or documentation may be reproduced or modified without the express consent of Florida International University, School of Computing & Information Sciences. Redistribution or commercial use is strictly prohibited. The Software is provided “AS IS”, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose, if no event shall the authors or copyright holders be liable for any claim, damage or other liability.

**ABSTRACT**

The Virtual Queue System will be designed for theme parks and other businesses that have multiple rides or events for which patrons typically wait in line. The idea is that both the theme park and the patron would benefit by the patrons walking around the park (and maybe spending money) rather than standing in line.

The Feasibility Study and Project Plan document gives an introduction to the Virtual Queue System. Chapter 1 gives a basic introduction about the Virtual Queue System, including the problem definition, background on the problem, definition of important terms, and an overview of the document. Chapter 2 introduces the purpose of our system since there is not an actual system, and it will also list the high-level user requirements along with an analysis of alternative solutions to the problem. Chapter 3 includes the Project Plan, which contains project management concepts of the project, hardware and software resources used, and mention the tasks, milestones, and deliverables. Chapter 4 covers different charts and information of the project, a feasibility and cost matrix, and a diary of meetings. Finally, Chapter 5 contains references to any other documents that have been used for reference.

**CONTEXT**

1. Introduction
   1. Problem Definition.
   2. Scope of system.
   3. Terminology - Definitions, acronyms, and abbreviations.
   4. Overview of document
2. Current System
3. Proposed System Requirements
   1. Functional Requirements
   2. Analysis of System Requirements
      1. Scenarios
      2. Use case model
      3. Static model e.g., object diagrams, class diagram
      4. Dynamic model e.g., sequence diagrams or state machines
4. Glossary - define terms used in document, especially domain specific terms.
5. Appendix
   1. Appendix A - Complete use cases
   2. Appendix B - Use case diagram using UML
   3. Appendix C - Static UML diagram
   4. Appendix D - Dynamic UML diagrams
   5. Appendix E - User Interface designs.
   6. Appendix F - Diary of meeting and tasks.
6. References

# 1. Introduction

The introductory chapter gives some background information about the Virtual Queue system. In the following sections, the problem definition, and scope of the system will be described. Following, the design methodology used is identified. In addition, definitions, acronyms, and abbreviations of terms used in this deliverable will be provided and explained on this chapter. Finally, the chapter will conclude with a brief explanation of what to expect from the following chapters of the document.

## Problem definition

Theme parks today as they grow in popularity have one fundamental flaw, the constant waiting in line for each ride. With each of the top tier rides constantly growing in wait time, this only makes matters worse for the average visitor. Standing in line begins to spend the majority of their time rather than sightseeing, exploring, or enjoying some tasty snacks at any store within the park and buying souvenirs. To make matters worse this is not taking into consideration weather conditions the standard waiting line may be under. As for the business aspect, it would be to the theme parks benefit to have more guests spending their time browsing the souvenir shops or food stands rather than spending their time idle while waiting for their turn to get on an attraction. Every minute longer the guests spend active can potentially increase park revenue.

Thus was the creation of the Virtual Queue System, it is the proposed solution to the problem. The system provides the visitors the freedom and flexibility to spend their time doing exactly as they wish while simultaneously waiting in line to ride the attraction they want. As for the Theme Park, it is increasing the traffic flow in between souvenir shops and food courts. Potentially increasing sales and making the theme park overall a more enjoyable place to vacation with. In addition to all of this, the system will also keep track of user information as well as aid in the operation of each ride.

## 1.2. Scope of System

This project (Virtual Queue) will consist of an online alternative to the standard waiting lines that currently exist within theme parks today. By allowing visitors the ability to queue up for attractions online rather than standing idle waiting their turn. In addition, this system allows administrators the ability to keep track of visitor informations and aid in the automatic run time per attraction.

## 1.3. Terminology

**Definitions**

* **Guest Users**: Anyone who wants to browse through the site and view offered deals before placing an order.
* **Registered Users**: Users that have already created an online account and can place orders and view previous orders they have requested.
* **System:** The system itself.
* **Theme park/Event**: Amusement park with a unifying setting or idea.

**Acronyms**

* **VQ**: Virtual Queue
* **FIU:** Florida International University
* **SCIS:** School of Computing & Information Sciences

**Abbreviations**

As of right now, there are no abbreviations for this project.

# 2. Current System (limitation and problems)

The current system requires the customers to buy their tickets and wait in line at the venue until it’s their turn for their event or ride. There is no mechanism to allow the user to register and sign in to see the available rides, and see the waiting time per each ride available and/or select the rides they want; or to logout when they decide to do so. It lacks of a system that allows the user the ability to be added to their selected rides, to see all the queues he/she registered for. There is no mechanism for adding a ride and automatically remove registered users from the queue once their time for a rides comes up, or to allow users to remove themselves if they decide to do so. There is no system with the capability of notifying the users before their selected rides/events starts.

# 3. Proposed System Requirements

The proposed system (Virtual Queue v2.0) requires the visitors log in the VQ in order to select the desired attractions. In turn it also allows the visitors the privilege to remove themselves from any given attraction if they no longer desire to be. The VQ grants administrative rights to admin users; Allowing them to automatically run attractions that remove visitor records from any given attraction while simultaneously sending notifications to each visitor whose turn is approaching via email. Administrators can also manipulate any ride instance in order to account for changes to any particular ride if need be, Administrators can also add visitor records to any particular ride. This chapter will include the functional and non-functional requirements of the system and the requirements analysis phase of the system.

## 3.1 Functional and Non-Functional Requirements

Allow unregistered users to register.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow users to validate their account.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow registered users to delete their accounts.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow registered users to sign in to their accounts.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow registered users to logout of their account.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow users to reset their password.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow registered users to have access to the available rides/events.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within less than a second.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow registered users to be added to their selected rides/events.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within less than a second.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow registered users to see all the rides/events they signed on for.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within less than a second.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow registered users to delete themselves from registered rides/events.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the user to be notified before their rides/events time occurs.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the system to add and/or remove registered users from their selected rides.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the system to send notification to registered user before ride time added approaches.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the system to store and retrieve information regarding to the rides/events.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the system to check for duplicates registration or multiple registrations for same user.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the system to handle network connectivity issues.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the user to be notified as soon as they register for their rides/events time occurs.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the admin to remove visitors from the queue as soon as they finish their events time occurs.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the admin to add visitors to the queue for a specific event.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the admin to add visitors to the queue for a specific event.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

Allow the admin to edit ride instances to manipulate time calculations for a specific event.

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

## 3.2 Analysis of System Requirements

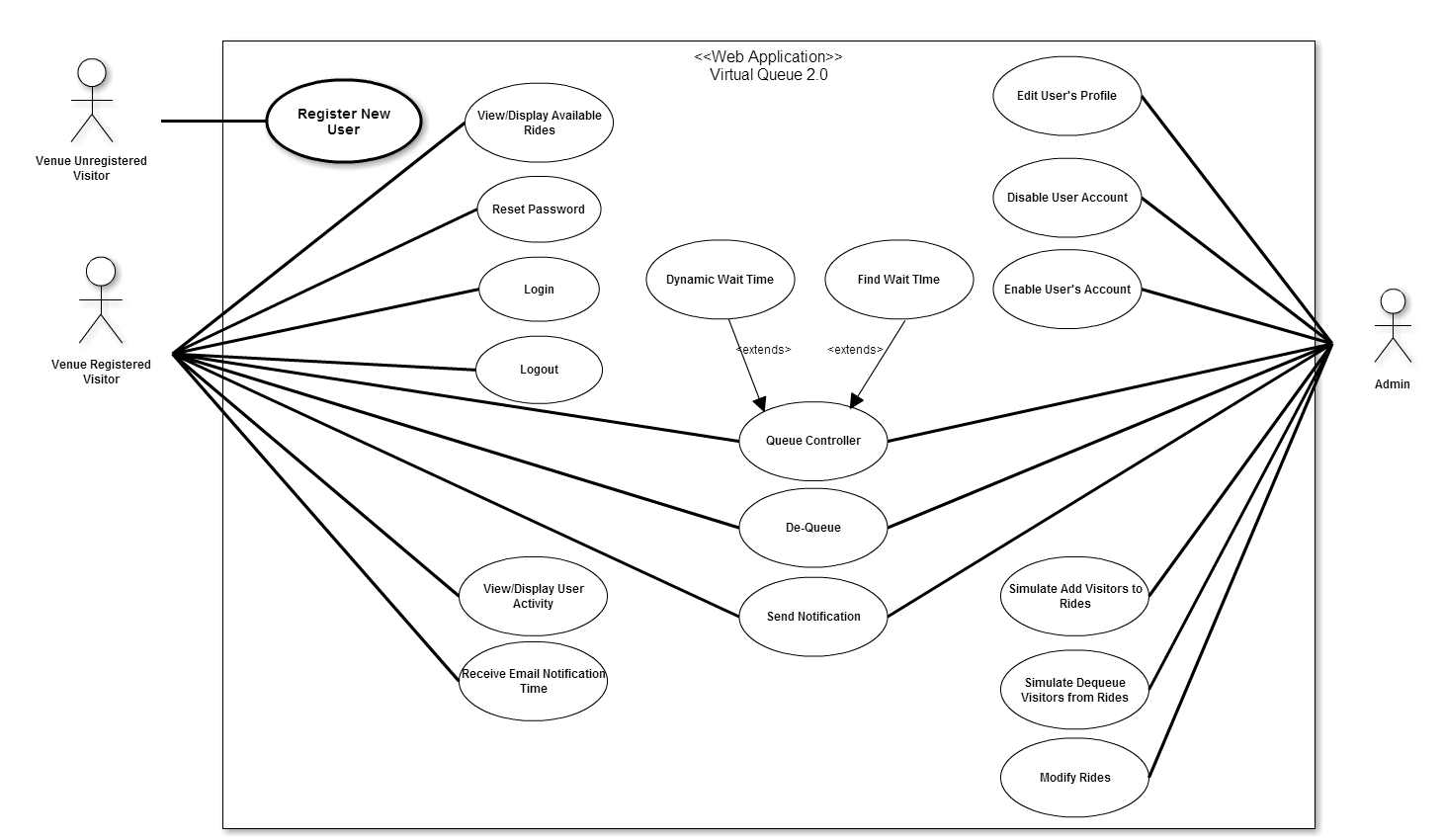
This section includes subsections that present the use case model of the VQ system, the static model, and the dynamic model consecutively.

### 3.2.1 Scenarios

The VQ allows for any individual person to vacation over to a Theme Park and take his/her entire family. While maximizing the time they can spend together by not having to spend all those long hours waiting to get on an attraction. A family of four can simultaneously register together for up to three attractions while relaxing in the food court over lunch. The VQ also allows the Theme Park to handle larger traffic flow of visitors maximize profits and create a better park that everyone can enjoy.

### 3.2.1 Use case model

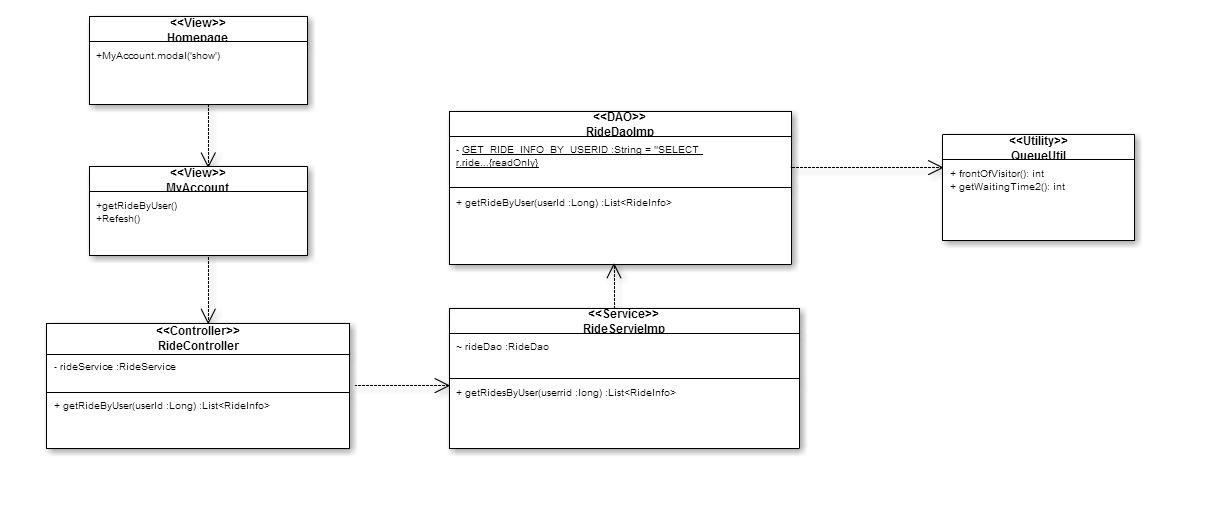
The use case diagram describes the list of steps that defines the interaction between the three types of users displayed in the diagram: venue guest user, venue registered user, and single venue admin and the system. They all work together to accomplish the goal of the proposed system. Below is the Use Cases Diagrams using UML for specific details with the specific functionalities that were implemented.



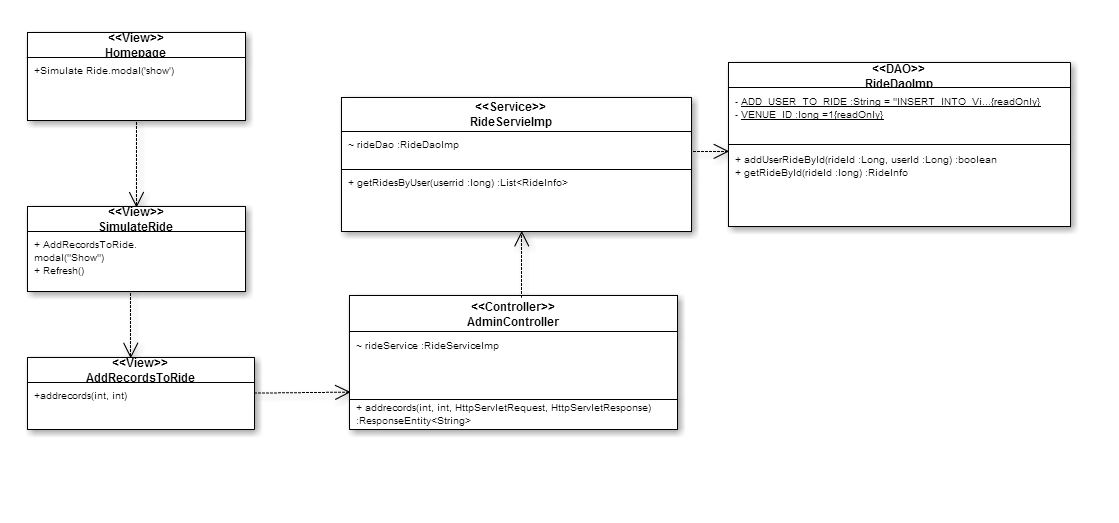
### 3.2.3 Static model

A static model expresses the system and does not account for sequence of events or time. For the VQ system, a class diagram will be included. The diagram will display the structure of system by showing the classes, attributes, methods, and the relationship between these classes. On Appendix D, the static diagram will be shown.

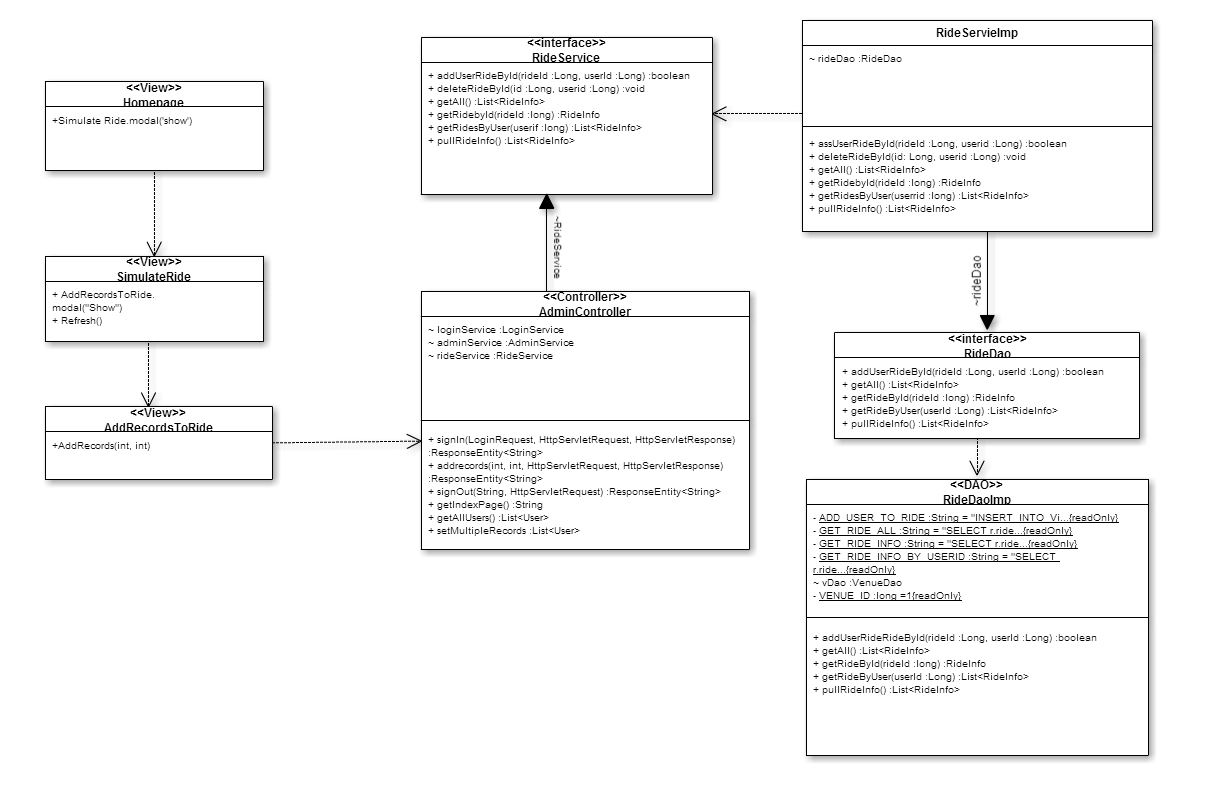
**VQ15 – Find Wait Times**

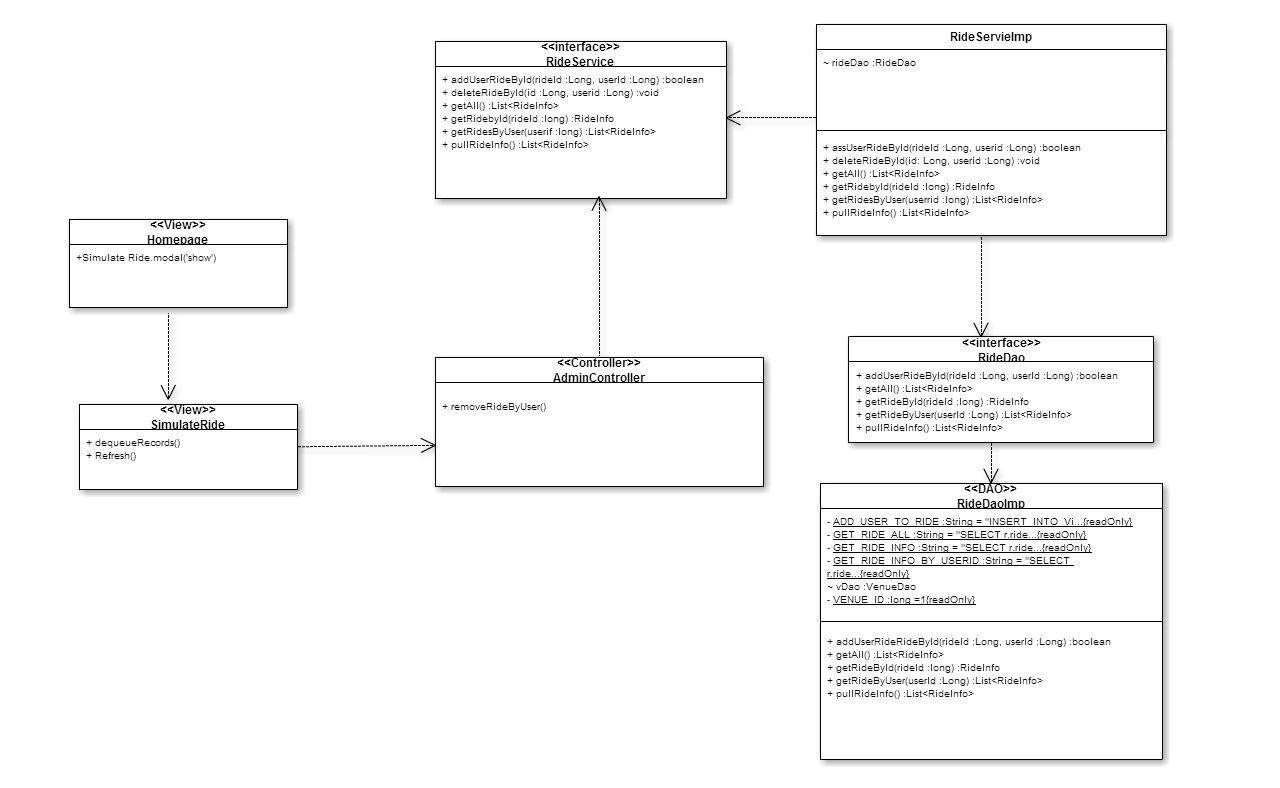


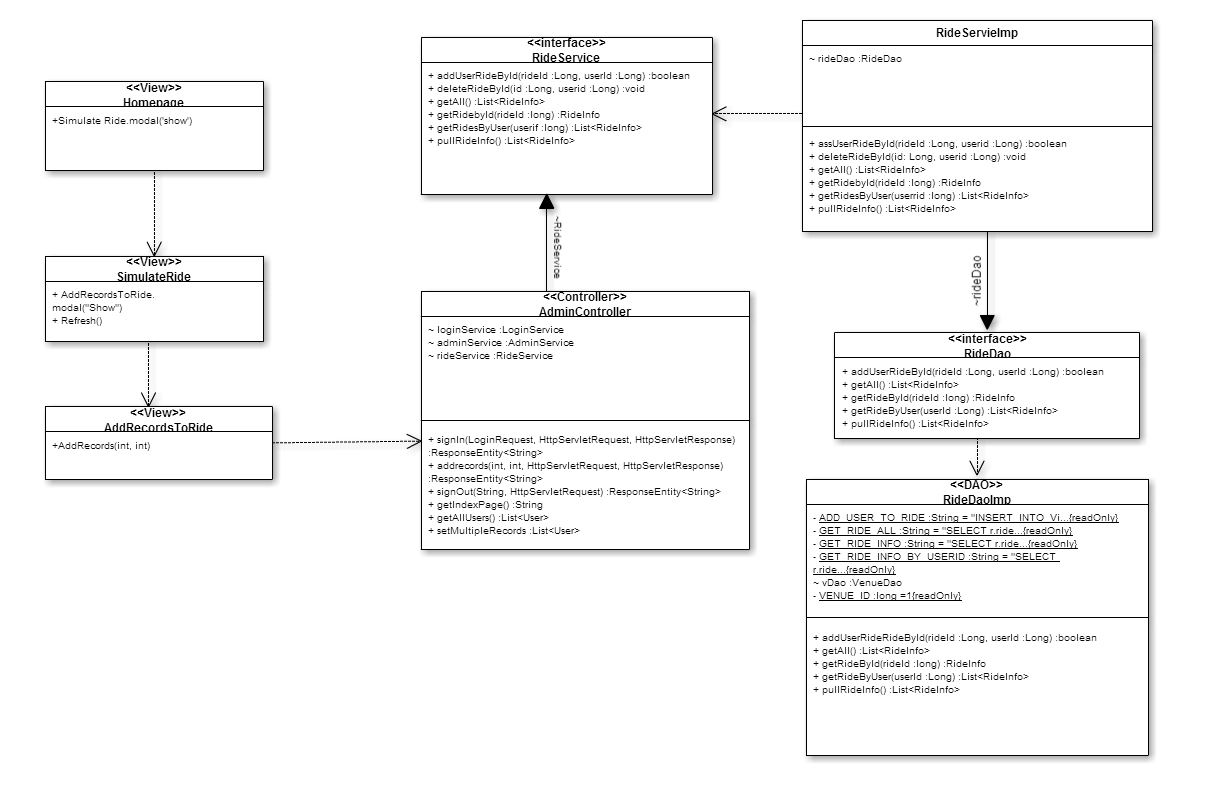
**VQ16 – Add Queue**



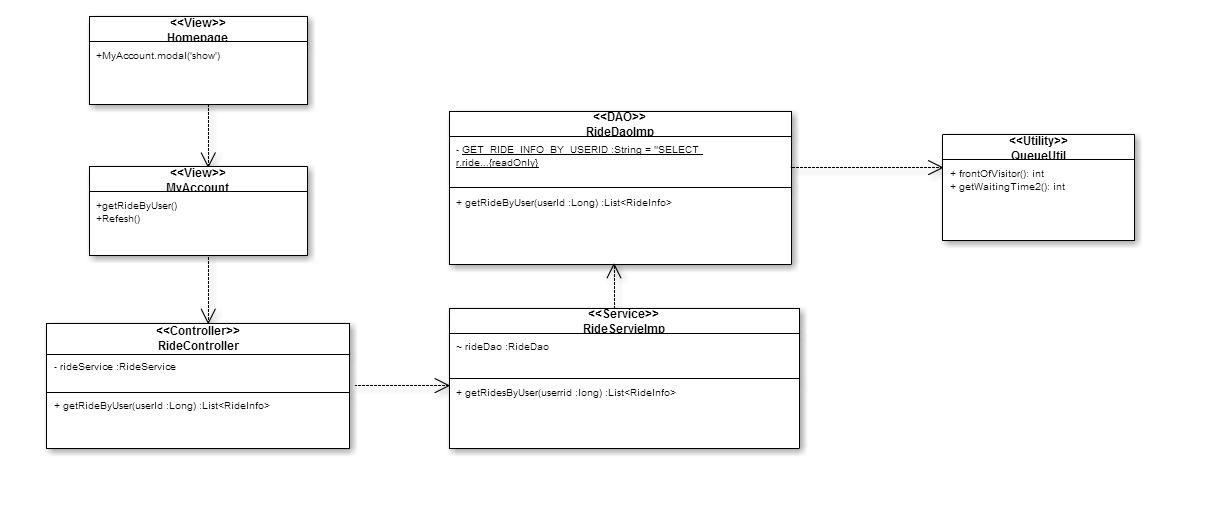
**VQ17 – Visitor DeQueue**



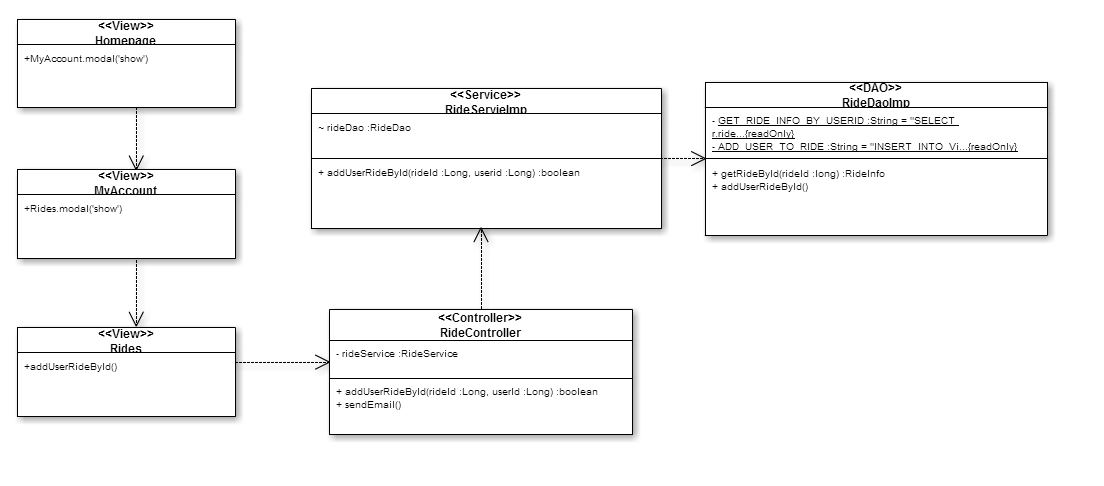
**VQ18 – Simulate Add Ride**

**VQ19 – Admin Simulate Dequeue for Rides**

**VQ20 – Dynamic Find Wait Time**



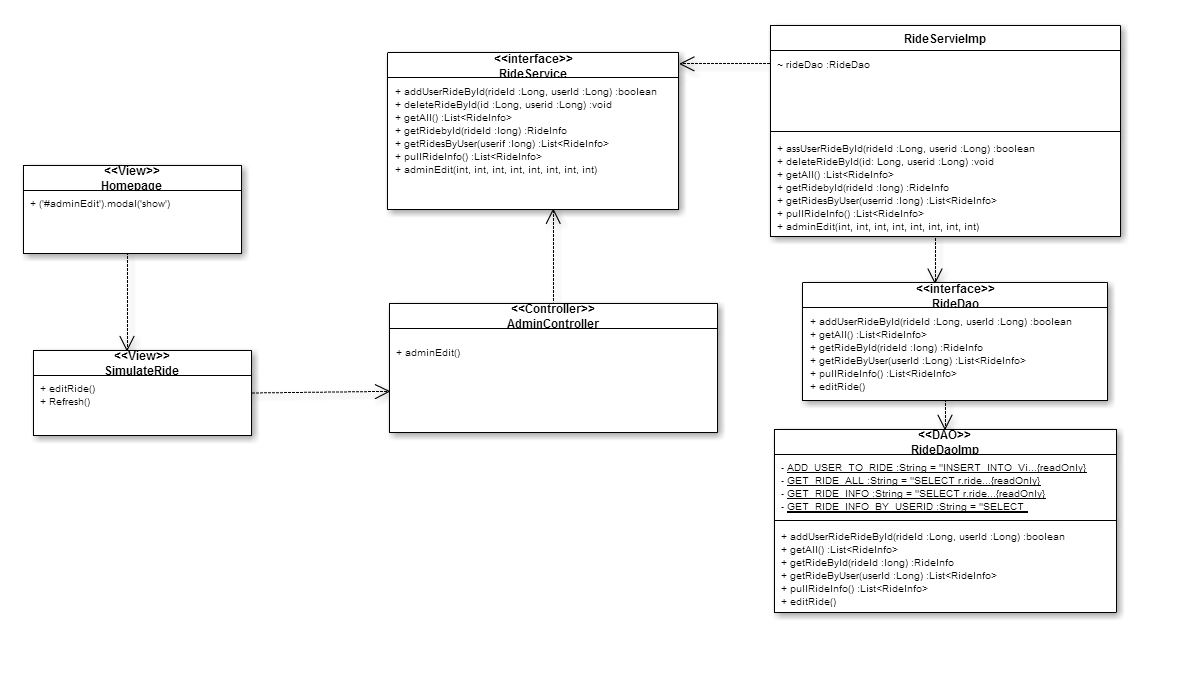
**VQ22 – Send Notification for adding ride**



**VQ23 – Send Notification for time warning**



**VQ24 – Admin Modify Rides**



### 3.2.4 Dynamic model

On the other hand, the dynamic model does account for time. For the VQ system, sequence diagrams will be included. These will show objects and class interactions in a sequence of events arranged in a time line that displays functionality in order to allow the developers and programmers to view how the users should transition based on these actions. On Appendix D, the dynamic diagram will be shown.

# 4. Glossary

UML: Stands for Unified Modeling Language. It is a standardized language that is used to model various things within the field of software engineering.

Theme park: An amusement park with a unifying setting or idea.

User Interface (UI): The way through which a user interacts with the computer system.

Class Diagram: A pictorial representation of all the classes in the system

Object Diagram: A pictorial representation of an instance of a class with example of how the data of the class will be populated.

Sequence Diagram: A pictorial representation of how processes operate with one another and the user during the course of a specific piece of functionality.

Task: A piece of job that serves as a unit of work.

# 5.Appendix

**5.1. Appendix A - Complete use cases**

*Use Case ID:* **VQ01 – User Login**

*Actors:* VenueRegistered Visitor, Single Venue Admin.

*Pre-conditions:*

1. Web page has been activated
2. Participants must have a web username and password.

*Description:*

1. Use case begins when the user accesses the login option.
2. The user will be prompted with a data entry template for username, password, and code if he/she is a Venue Registered Visitor.
3. The user will be prompted with a data entry template for username, and password if he/she is a Single Venue Admin.
4. The user provides a previously registered user name, password, and code if he/she is a Venue Registered Visitor.
5. The user provides a previously registered user name, and password if he/she is a Single Venue Admin.
6. User shall continue request by accessing the Login.
7. If the credentials are valid, the system will log the user into the corresponding page.
8. Use case ends when access is granted to respective user.

*Relevant requirements:* A user will only be allowed into the system if he/she has a valid username and password.

*Post-conditions:*

1. Access is granted.

*Alternative Courses of Action for Venue Registered Visitor:*

1. In step 2 of Description section there is an option that allows the Venue Registered Visitor to reset the password if he/she forgot. (See use Case **VQ03 – Reset Password**)
2. In step 6 of Description section users have the option to click cancel and close the data entry form.

*Exceptions:*

1. The login option on the Website is not active.
2. The cancel option on the Website is not active.
3. The option to reset password is not active.
4. The database is not active.

*Related Use Cases: Related Use Cases:* **VQ04 – Confirm Identity.**

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ02 – User Logout**

*Actors:* VenueRegistered Visitor, Single Venue Admin.

*Pre-conditions:*

1. Web page has been activated.
2. User must have been previously logged in.

*Description:*

1. Use case begins when the user accesses the logout option.
2. The system logs the user out.
3. Use case ends when the system logs user out and displays the home screen.

*Relevant requirements:* A user will only be logged out if he/she has previously logged in.

*Post-conditions:*

1. User gets successfully logged out of the system.

*Alternative Courses of Action:*

1. The user closes the webpage.

*Exceptions:*

1. The logout option on the webpage is not active.

*Related Use Cases:* None.

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ03 – Reset Password**

*Actors:* VenueRegistered Visitor.

*Pre-conditions:*

1. User must have previously registered into the system.
2. User must be logged out of the system.

*Description:*

1. Use case begins when the user accesses the option to reset password on the login screen.
2. The system will prompt the user with a data entry template.
3. The user should enter username, which will be their registered email, choose their security question, and write their security answer, new password and password confirmation.
4. The user shall complete the form and submit the changes.
5. If the username is on record, and security question, and security answer are correct, the system will update the password and notify the user if the request was submitted successfully. (See Use case **VQ04 – Confirm Identity**).
6. Use case ends when the request is stored in the system.

*Relevant requirements:* A user will only be allowed to reset password if he/she has previously registered.

*Post-conditions:*

1. The password gets successfully changed in the database.

*Alternative Courses of Action:*

1. In step 3 of Description section the username could not be found.
2. In step 3 of Description section the security question chosen does not match the records.
3. In step 3 of Description section the answer to the security answer does not match the records.
4. In step 4 of Description section the user has the option to cancel the request.

*Extensions:*

1**. VQ04 – Confirm Identity**

*Exceptions****:***

1. The option to reset the password is inactive.
2. The user table in the database is inactive.

*Related Use Cases:* **VQ04 – Confirm Identity.**

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ04 – Confirm Identity**

*Actors:* VenueRegistered Visitor, Single Venue Admin.

*Pre-conditions:*

1. Web Page has been activated.
2. User already has an active access account.
3. User accesses the login option.
4. User accesses the forget password option.

*Description:*

1. Use case begins when the user is prompted with a template for data entry either for login or reset password.
2. The user shall enter his username which is the email, password correctly, and code if she/he is a Venue Registered User trying to login.
3. The user shall enter his username which is the email, password correctly if she/he is a Single Venue Admin trying to login.
4. The user shall enter his username, which is the email, and finish the rest of the reset password form correctly if she/he is a Venue Registered User trying to login.
5. Use case ends when user clicks on submit button and is presented with the login form.

*Relevant requirements:* none.

*Post-conditions:* The Venue Registered User is taken back to Login if she/he is trying to reset the password, or presented the login form again if it is a Venue Registered User or a Single Venue Admin trying to login.

*Alternative Courses of Action:*

1. In step 1 of the Description section the user can cancel the request.
2. In step 2 of the Description section the username, password, or code are not found.
3. In step 3 of the Description section the username or password are not found.

*Exceptions****:***

1. The system is unavailable.

*Related Use Cases:* VQ01 Login.

*Related Use Cases:* VQ03 Reset Password.

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

*Use Case ID:* **VQ05 - Register New User**

*Actors:* Venue Unregistered Visitor

*Pre-conditions:*

1. Web page has been activated

*Description:*

1. Use case begins when guest user accesses the register option on the Website.

2. The system shall provide the guest user or new administrator with a template for data entry.

3. The guest user should enter the following data: first name, last name, password, security question, security answer, phone number, age, and code (for confirmation).

4. The guest user should send the request by accessing the submit option.

5. The system shall then notify the guest user if the request was submitted successfully.

6. When the request is received the system shall update the record for that new user.

7. Use case ends when the record is updated.

*Relevant requirements:*

1. A guest user will only be able to register if they do not have a profile created from before.
2. Venue Registered users will be rejected.

*Post-conditions:*

1. The record for that user has been created.

*Alternative Courses of Action:*

1. In step 4 o the f Description section the user has the option to cancel the request.
2. In step 5 o the f Description section, if any of the required fields are left blank the system shall notify the user of the missing fields, and how to fill the fields properly.

*Exceptions:*

1. The register option on the webpage is not active.
2. After the user enters the required information into the system the user gets rejected.

*Use Cases:* None.

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ06 – Display Available Rides and their Waiting Time.**

*Actors:* VenueRegistered Visitor.

*Pre-condition:*

1.      Web page has been activated.

*Description:*

1.      Use case begins when user accesses the option Rides from the home page or from his account page.

2.      The system shall get the list of the rides available at the venue along with each ride actual waiting time.

3.      Use case ends when the system displays the list of the rides available for the venue user along with each ride actual waiting time.

*Alternative Courses of Action:*

1. In Step 1 of Description section the Venue Registered User request could not be completed.
2. In step 3 of the Description section the user has the option to cancel the form.
3. In Step 3 of Description section the list of venue available rides and their respective waiting time could not be display.
4. In Step 3 of Description section the Venue Registered User can use the search option to search for a specific ride name or other criteria.
5. In Step 3 of Description section the Venue Registered User can switch the views of the table presented.
6. In Step 3 of Description section the Venue Registered User can update the complete table by clicking the update symbol.
7. In Step 3 of Description section the Venue Registered User can choose which columns they want to see from the table presented.

*Exceptions:*

1. The option to select ride/event is not active.
2. The system could not continue after submitting the selection.

*Related Uses case:* **VQ07 – Add (Queue) for a Ride**

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ07 – Add (Queue) User for a Ride**

*Actors:* VenueRegistered Visitor.

*Pre-condition:*

1. Web page has been activated.
2. Venue Registered User has an active account within the system.
3. Venue Registered User is already on his account page.

*Description:*

1. Use case begins when user clicks on Rides from his account page, and a list with available rides is presented.
2. User shall continue when she/he has already decided which ride she/he wants to add (queue).
3. User shall continue by clicking the “+” symbol to add their preferred ride.
4. The system shall notify the user if ride was successfully added.
5. Use case ends when customer accepts the notification displayed and she/he is taken to his/her account page where the ride added is displayed along with his/her waiting time.

*Post-conditions:*

1. The system shall update the changes made on the database.
2. The number of rides for the customer is increased by one in the database.
3. User is added to UserQueue table.
4. The waiting time for that specific ride is increased.

*Alternative Courses of Action:*

1. In step 3 of the Description section the user has the option to cancel the ride from his page.
2. In Step 4 of Description section the system notifies that the request could not be completed.

*Exceptions:*

1.      The option to add ride is not active.

2.      The option to accept the notification is not active.

3.      The system could not submit the request.

*Related Uses case:* **VQ06 – Display Available Rides and Their Waiting Time.**

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ08 - Edit User's Profile**

*Actors:* Single Venue Admin.

*Pre-conditions:*

1. User (administrator) has an existing profile.
2. Web page has been activated.
3. User (administrator) has successfully logged onto the system.

*Description:*

1. Use case begins when *Admin* selects the Search under Admin tab from the home page.
2. *Admin* search the user to be edited.
3. Admin finds the user and select Edit option for that specific user.
4. Admin is prompted with a data entry template with already loaded user info.
5. Admin can change any field they want (making sure all fields have an entry).
6. Admin shall accept “Submit”.
7. The system shall notify the admin if the request was submitted successfully.
8. Admin is taken back to the search page.
9. Use case ends when the system saves the updated information.

*Relevant requirements:* None.

*Post-conditions:*

1. User's profile is updated in the database.

*Alternative Courses of Action:*

1. In Step 4 of Description section the user has the option to cancel the request.
2. In Step 5 of Description section the system prompt the user to enter more data if a required field is left blank.
3. In Step 7 of Description section the system notifies that the request could not be completed.

*Exceptions:*

1. The edit option link is not active.
2. The cancel option link is not active.
3. The submit option is not active.

*Related Use Cases:* None

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ09 - Disable User's Account**

*Actors:* Single Venue Admin.

*Pre-conditions:*

1. User (administrator) has an existing profile.

2. Web page has been activated.

3. User has successfully logged onto the system.

*Description:*

1. Use case begins when *Admin* selects Search under Admin tab from the home page.
2. *Admin* search the user to be disabled.
3. Admin finds the user and selects disabled option for that specific user.
4. The system shall ask the *Admin* if they are sure they want to proceed.
5. The *Admin* shall confirm by accepting the “Ok” option.
6. The system shall notify the *Admin* if the request was submitted successfully.
7. *Admin* is taken back to the search page.
8. Use case ends when the system saves the updated information.

*Relevant requirements:* None.

*Post-conditions:*

1. Record for user is updated in the database.

*Alternative Courses of Action:*

1. In step 4 of Description section the *Admin* has the option to cancel the request.

*Exceptions:*

1. The system is unavailable.
2. The “Cancel” option is unavailable.
3. The confirmation dialog is not displayed.
4. The “Disable” option is unavailable.

*Related Use Cases:* None

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ10 - Enable User's Account**

*Actors:* Single Venue Admin.

*Pre-conditions:*

1. User (administrator) has an existing profile.

2. Web page has been activated.

3. User has successfully logged onto the system.

*Description:*

1. Use case begins when *Admin* selects Search under Admin tab from the home page.
2. *Admin* search the user to be enabled.
3. Admin finds the user and selects enabled option for that specific user.
4. The system shall ask the *Admin* if they are sure they want to proceed.
5. The *Admin* shall confirm by accepting the “Ok” option.
6. The system shall notify the *Admin* if the request was submitted successfully.
7. *Admin* is taken back to the search page.
8. Use case ends when the system saves the updated information.

*Relevant requirements:* None.

*Post-conditions:*

1. Record for user is updated in the database.

*Alternative Courses of Action:*

1. In step 4 of Description section the *Admin* has the option to cancel the request.

*Exceptions:*

1. The system is unavailable.
2. The “Cancel” option is unavailable.
3. The confirmation dialog is not displayed.
4. The “enable” option is unavailable.

*Related Use Cases:* None

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ11 – Delete (Dequeue) User from a Registered Ride**

*Actors:* VenueRegistered Visitor.

*Pre-condition:*

1. Web page has been activated.
2. Venue Registered User has successfully logged onto the system.
3. User is on his/her account page.
4. There is a user request to remove an already ride he/she queued for from his/her account.

*Description:*

1. Use case begins when user accesses MyAccount link from the home page, and the corresponding page is displayed.
2. User shall continue when she/he has already decided which ride she/he wants to delete (dequeue).
3. User shall continue by clicking the “X” symbol to dequeue from that specific ride.
4. The system shall ask the *user* if they are sure they want to proceed.
5. The *user* shall confirm by accepting the “OK” option.
6. Use case ends when customer accepts the notification displayed and she/he is taken to his/her account page where the ride delete is not displayed anymore on his account page.

*Post-conditions:*

1. The system shall update the changes made on the database.
2. The system shall dequeue user from ride when the time for that ride comes up, even if he/she have not select de “delete” option.
3. The number of rides for the customer is decreased by one in the database.
4. User is deleted to UserQueue table.
5. The waiting time for that specific ride decreases.

*Alternative Courses of Action:*

1. In step 3 of the Description section the user has the option to cancel the confirmation.
2. In Step 4 of Description section the system notifies that the request could not be completed.

*Exceptions:*

1.      The option to delete ride is not active.

2.      The option to accept the notification is not active.

3.      The system could not submit the request.

*Related Uses case:* **VQ07 – Add (Queue) for a Ride**

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ12 – View/Display User Activity**

*Actors:* VenueRegistered Visitor.

*Pre-condition:*

1.      Web page has been activated.

*Description:*

1.      Use case begins when user accesses the option MyAccount from the home page.

2.      The system shall get the list of the rides available for that specific user along with each ride actual waiting time.

3.      Use case ends when the system displays the list of the rides available for that user along with each ride actual waiting time.

*Alternative Courses of Action:*

1. In Step 1 of Description section the Venue Registered User request could not be completed.
2. In step 3 of the Description section the user has the option to close the modal.
3. In Step 3 of Description section the list of venue available rides and their respective waiting time could not be display.
4. In Step 3 of Description section the Venue Registered User can use the search option to search for a specific ride name or other criteria.
5. In Step 3 of Description section the Venue Registered User can switch the views of the table presented.
6. In Step 3 of Description section the Venue Registered User can update the complete table by clicking the update symbol.
7. In Step 3 of Description section the Venue Registered User can choose which columns they want to see from the table presented.

*Exceptions:*

1. The system could not display the form.

*Related Uses case:* None

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ13 – Send/Received Notification Time**

*Actors:* Venue Registered User.

*Pre-condition:*

1. Web page has been activated.
2. The user has successfully logged onto the system.
3. The user has successfully selected the ride preferred.

*Description:*

1. Use case begins when user has already added the ride requested to his/her account page.
2. User shall be kept in the ride queue until the time for his/her turn approaches.
3. System should create a job and send notification to each user that time for rides are coming up next on queue.
4. Use case ends when user receives the notification email with the name and time for the specific ride before the time approaches.

*Post-conditions:*

1. The system shall update the changes made on the database to update the jobs.

*Alternative Courses of Action:*

1. In step 2 of the Description section the user has the option to cancel their ride from their account page.
2. In step 3 of the Description section the could not create a job and send the notification.
3. In Step 4 of Description section the system notifies that the notification could not be sent.

*Exceptions*

1. The system could not create the job.
2. The system could not send the request.

*Related Uses case:* **VQ07 – Add (Queue) for a Ride**

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ14 – Start-Stop Jobs to Send Notifications and Dequeue users automatically from Rides**

*Actors:* Single Venue Admin.

*Pre-condition:*

1. Venue is going to be opened soon.

*Description:*

1. Use case begins when admin is decides is time to start the scheduler to send notifications and delete users from rides when is time to do so.
2. Admin shall type the appropriate URL path with the right command to start the scheduler jobs that will send the notification first and then dequeue user from ride on specific time.
3. Use case ends when system creates the appropriate jobs and execute in in the corresponding time.

*Post-conditions:*

1. The system shall update the users account once they are dequeue from ride.
2. The system shall update the changes made on the database to update the jobs.
3. The system should continue the jobs until the administrator decides to stop it.

*Alternative Courses of Action:*

1. In step 2 of the Description section the admin does not type the URL path correctly.
2. In step 3 of the Description section the system could not create a job and send the notification.
3. In step 3 of the Description section the system could not create a job to dequeue user from ride.
4. In Step 4 of Description section the system notifies that the notification could not be sent.

*Exceptions*

1. The system could not create the job.
2. The system could not send the request.

*Related Uses case:* **VQ13 – Send/Receive Email Notification**

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 3 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ15 – Find Wait Time**

*Descriptions:* Calculate how many riders per car to find appropriate wait times to populate lists

*User Story:* #109 Find Wait Time

*Actors:* Single Venue Admin and Venue Visitor/Non Visitor.

*Pre-condition:*

1. Admin or visitor logged in. Non Visitor does not need to be.

*Description:*

1. Use case begins when admin or visitor clicks on Ride Tab above the VQ home webpage.
2. Use case ends when the Ride's list populated with the ride names and their concurrent waiting times.

*Post-conditions:*

1. The system shall update the waiting time as users queue.
2. The system shall update the waiting time when users dequeue from the rides.

*Alternative Courses of Action:*

1. In Step 1 Users can click on My Account and click on Ride tab within to access the Ride Find Wait Time.

*Exceptions*

1. The system could not find wait time.
2. The system could not find activities.

*Related Uses case:* **N/A**

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 2 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ16 – AddQueue**

*Descriptions:* ‘Controller that allows finding wait time and queuing and dequeuing to the queue.

*User Story:* #108 Implementing the Queue

*Actors:* Single Venue Admin and Venue Visitor/Non Visitor.

*Pre-condition:*

1. Admin or visitor logged in. Non Visitor does not have access.

*Description:*

1. Use case begins when admin or visitor user clicks on Ride Tab on the VQ home webpage.
2. The User or Administrator will have a freshly updated list of Rides and their concurrent waiting times.
3. The admin or Visitor can click on the “Add Ride” button located to the right of each of the wait times.
4. Use case ends when the selected ride becomes added to the Queue and displayed within the user account under “My Account”.

*Post-conditions:*

1. The system shall update the waiting time as users queue.
2. The system shall update the admin or Visitor account with newly selected rides.

*Alternative Courses of Action:*

1. In Step 1 Users can click on My Account and click on Ride tab within to access the Ride List.

*Exceptions*

1. The system could not find wait time.
2. The system could not add into queue.
3. The system could not find activities.

*Related Uses case:* **VQ15 – Find Wait Time,****VQ17 – Visitor DeQueue**

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 2 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ17 – Visitor DeQueue**

*Descriptions:* If a user does not want to go on a particular ride they have queued up for. This allows them to remove themselves from the queue for that ride.

*User Story:* #121 Visitor Dequeue for rides

*Actors:* Single Venue Admin and Venue Visitor/Non Visitor.

*Pre-condition:*

1. Admin or visitor logged in. Non Visitor does not have access.

*Description:*

1. Use case begins when a visitor user clicks on MyAccount Tab on the VQ home webpage.

2. The User will have a freshly updated list of Rides and their concurrent waiting times.

3. The Visitor can click on the “Remove Ride” button located to the right of each of the wait times.

4. Use case ends when the selected ride becomes removed from the Queue as well as the users account and displayed within the user account under “My Account”.

*Post-conditions:*

1. The system shall update the waiting time as users queue.

2. The system shall update the admin or Visitor account with newly selected rides.

*Alternative Courses of Action:*

1. In Step 1 Users can click on My Account and click on Ride tab within to access the Ride List.

*Exceptions*

1. The system could not find wait time.

2. The system could not remove from queue.

3. The system could not find activities.

*Related Uses case:* **VQ15 – Find Wait Time,****VQ16 – Add Queue**

**Special Requirements:**

· **Usability**: No previous training time. System is simple and easy to follow.

· **Reliability**: The system should perform correctly 99% of the time.

· **Performance**: The system should be sent and saved within 2 seconds.

· **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ18 – Simulate Add Visitors to Rides**

* *Descriptions:* As an admin, I should have the controllers be able to add random visitor records to a particular ride, so to compare wait times for a ride and also to increase the wait time.

*User Story:* #111 **Simulate Add Visitors to Rides**

*Actors:*  Admin

*Pre-condition:*

1. Admin is logged in.

*Steps:*

1. Use case begins when admin clicks on Simulate Ride tab within the admin tab in the homepage.

2. system will display a pop up and the table within is already populated Then in the Simulate Ride Widow will pop up and the table within is already populated with all active rides and total # of records queued in that venue.

3. The actor clicks on add records button in simulate section of a table to the rides they want to add records to.

4. The system will display the Add Records to Ride window will open where the admin enters number of dummy records they want to add in the text box.

5. The admin clicks on Submit button in the Add Records to Ride window.

6. Use case ends the Admin is taken back to the Simulate Ride Window.

*Post-conditions:*

1. The system shall update the queue, by adding # of dummy records to the queue to the specific ride selected.

2. The system shall update the number of queue records on the Admin Ride Table for that specific ride selected.

*Alternative Courses of Action:*

1. In step 2 admin can enter value 0 (in Admin Queues text box) and press Admin Add Queue button for a ride, then no records will be added.

*Exceptions*

1. The system could not find list of rides.

2. Invalid value entered in the Admin Queues text box.

3. The system could not add # of records to that ride.

4. Admin cannot add more then 100 records at a time.

*Related Uses case:* **VQ19 – Simulate Dequeue Vistors from Rides**

· **Usability**: No previous training time. System is simple and easy to follow.

· **Reliability**: The system should perform correctly 99% of the time.

· **Performance**: The system should be sent and saved within 1 seconds.

· **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ19 – Simulate Dequeue Visitors from Rides***Descriptions:* Admin has ability to remove queues records for particular rides.

*User Story:* #90 Simulate Queue for admin: Add Multiple Visitors to Rides

*Actors:* Single Venue Admin

*Pre-condition:*

1. Admin is logged in.

*Steps:*

1. Use case begins when admin clicks on admin ride tab.

2. Then the Admin Ride Table will be populated with all active rides and total # of records queued in that venue.

3. Use case ends the Admin presses the Admin Remove Queue button for specific ride from the table.

*Post-conditions:*

1. The system shall update the queue, by removing # equal to the rides max capacity.

2. The system shall update the number of queue records on the Admin Ride Table for that specific ride.

*Alternative Courses of Action:*

1. In step 3, User does not press the Admin Remove Queue button.

*Exceptions*

1. The system could not find list of rides.

2. The system could not remove # equal to the rides max capacity.

3. The system does not remove the correct number of records from that queue.

4. The system removes some other rides that was not selected.

*Related Uses case:* **VQ18 – Simulate Add Visitors to Rides**

· **Usability**: No previous training time. System is simple and easy to follow.

· **Reliability**: The system should perform correctly 99% of the time.

· **Performance**: The system should be sent and deleted within 1 seconds.

· **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ20 – Dynamic Find Wait Time**

*Descriptions:* As an user and admin, I should be able to see that the wait time for the rides that I am queued up to change depending on how many visitors is queued up in front of me.

*User Story:* #97 Dynamic Find Wait Time

*Actors:* Admin and Visitor

*Pre-condition:*

1. Admin/Visitor is logged in. Admin/Visitor have already queued up to a ride.

*Description:*

1. Use case begins when admin/visitor clicks on My Account tab above the VQ home webpage.
2. The window will already populated with whatever ride admin/ride is queued up to.
3. Use case ends when admin/visitor clicks on refresh button and the ride wait time decreases only if the ride was dequeued and updated the table with new information.

*Post-conditions:*

1. The system shall update the wait time for admin/visitor when the ride is dequeued.

*Alternative Courses of Action:*

1. In Step 3 the admin/visitor does not click the refresh button and wait time does not get updated.

*Exceptions*

1. The system could not find the dynamic wait time.
2. The system could not find visitor queued to any records.

*Related Uses case:* **VQ15 – Find Wait Time**

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 2 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ22 – Send Notifications for adding ride**

*Descriptions:* As a Visitor or an Admin, I should be notified when I queued myself to a ride and when its almost time to get onto the ride.

*User Story:* #112 Send Notifications

*Actors:* Admin & Visitor

*Pre-condition:*

1. Admin & Visitor is logged in.

*Description:*

1. Use case begins when admin/visitor clicks on My Account tab above the VQ home webpage.
2. Then the admin/visitor clicks on the Ride Tab within the My account tab, which Ride window will open.
3. Admin/visitor then queue themselves on the rides they want to get onto, by pressing Add Ride button.
4. Admin/visitor should exit out of the ride window by pressing the cancel button or x button.
5. Use case ends when the admin/visitor receives the email that they are queued up to a Ride. The email provides what the wait time and what position the Visitor is in the line.

*Post-conditions:*

1. System shall send another email to Visitors when they are almost ready to get onto a ride.

*Alternative Courses of Action:*

1. In Step 3, Visitor does not queue up to a ride.
2. The Admin/Visitor does not follow 2-5 steps. Instead they are already queued to ride they want to. When the ride dequeues as times goes on, and that user finally reaches 10 minute mark in that queue of the ride. The admin/visitor will receive email notifying that they have less than 10 minutes to go on the ride.

*Exceptions*

1. The system could not add admin/visitor to the Ride.
2. The system could not send email to the admin/visitor.

*Related Uses case:* VQ23 – Send Notifications for time warning

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 2 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

*Use Case ID:* **VQ23 – Send Notifications for time warning**

*Descriptions:* As a Visitor or an Admin, I should be notified when I queued myself to a ride and when its almost time to get onto the ride.

*User Story:* #112 Send Notifications

*Actors:* Admin & Visitor

*Pre-condition:*

1. Admin & Visitor is logged in and already queued up to a ride.

*Description:*

1. Use case begins when admin/visitor clicks on My Account tab above the VQ home webpage.
2. Use case ends when one of the rides that the admin/visitor is waiting for is dequeued and that users wait time becomes 10 minutes or less. At that point the system will send email notifications to each admin/visitor who has 10 minutes or less wait time for that ride.

*Post-conditions:*

1. System shall send another email to Visitors when they are almost ready to get onto a ride.

*Alternative Courses of Action:*

1. In Step 1 does not click on My Account rab to check his/her wait time.

*Exceptions*

1. Admin/Visitor is not queued up to the ride.
2. The system could not send email to the admin/visitor.

*Related Uses case:* VQ22 – Send Notifications for adding ride

**Special Requirements:**

* **Usability**: No previous training time. System is simple and easy to follow.
* **Reliability**: The system should perform correctly 99% of the time.
* **Performance**: The system should be sent and saved within 2 seconds.
* **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

* *Use Case ID:* **VQ24 – Admin Modify Rides**
* *Descriptions:* As an Admin, I should be able to modify the ride information to accompany the many changes that may occur through the life of the venue

*User Story:* #119 Simulate Queue for admin: Add Multiple Visitors to Rides

*Actors:* Single Venue Admin

*Pre-condition:*

1. Admin is logged in.

*Steps:*

1. Use case begins when admin clicks on admin tab.

2. Then the Admin Ride Table will be populated with all active rides and total # of records queued in that venue.

3 The Admin presses the Edit ride button for specific ride from the table.

4. The admin user must than fill out the pop up window with the Interval Time, Entry Time, Exit Time, Max Capacity, and Concurrency ride.

5. The use case ends when the admin hits the submit button from this window and accepts the changes made.

*Post-conditions:*

1. The system shall update the ride table, updating the corresponding elements in the table according to what was added.

*Alternative Courses of Action:*

1. N/A

*Exceptions*

1. The system could not find list of rides.

2. The system could not update the Interval Time .

3. The system could not update the Entry Time. 4. The system could not update the Exit Time.

5. The system could not update the Max Capacity.

6. The system could not update the Concurrency ride.

*Related Uses case:* **N/A**

· **Usability**: No previous training time. System is simple and easy to follow.

· **Reliability**: The system should perform correctly 99% of the time.

· **Performance**: The system should be sent and deleted within 1 seconds.

· **Supportability**: The system should be easy to maintain, make appropriate changes, and be correctly handled by IE, Mozilla, Chrome and Safari.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**5.2. Appendix B - Use case diagram using UML**

**5.3. Appendix C - Static UML diagram**

Class Diagram

**5.4. Appendix D - Dynamic UML diagrams**

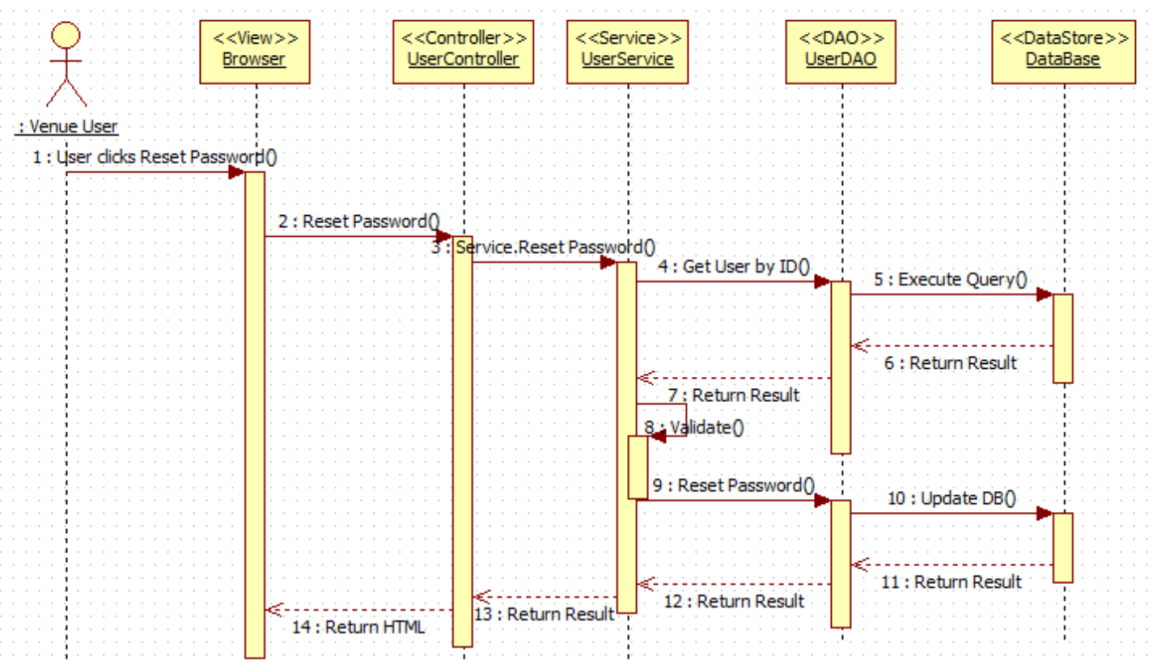
VQ01-Login:



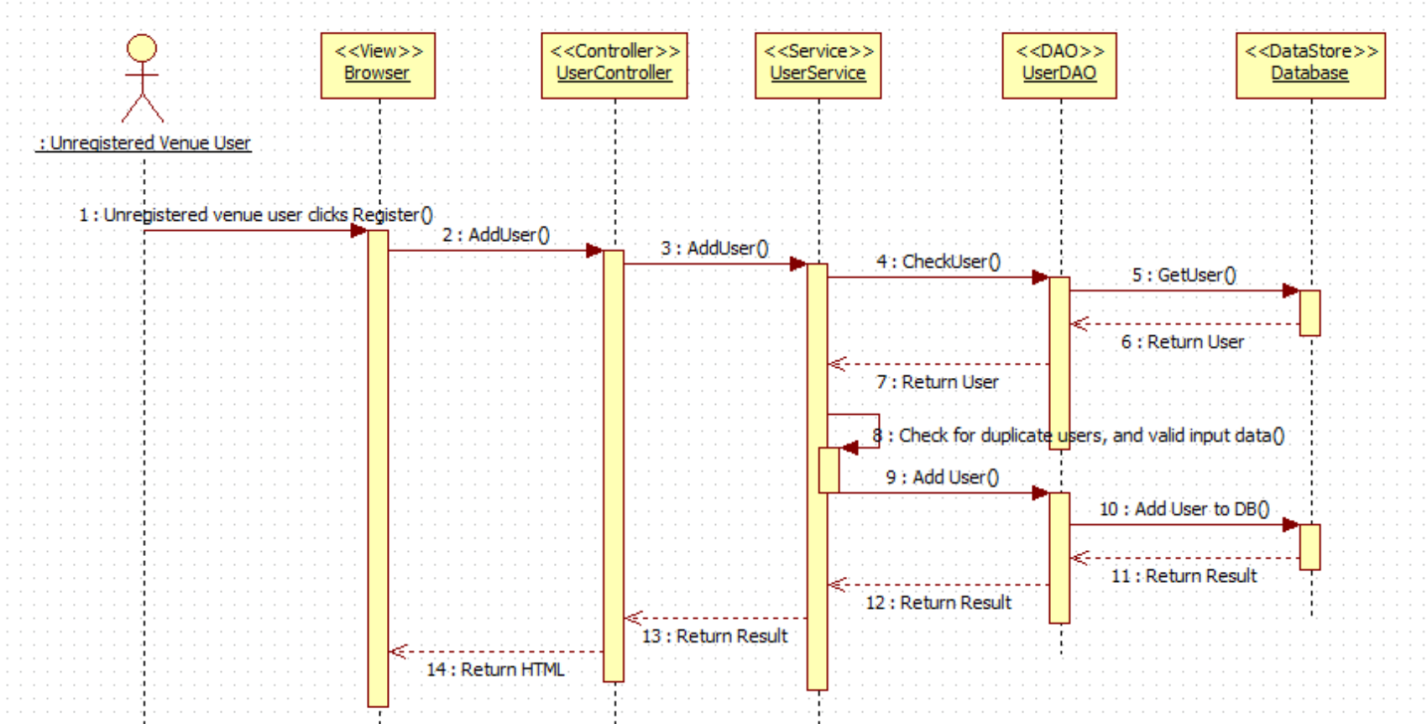
VQ02-Logout:



VQ03-Reset Password:



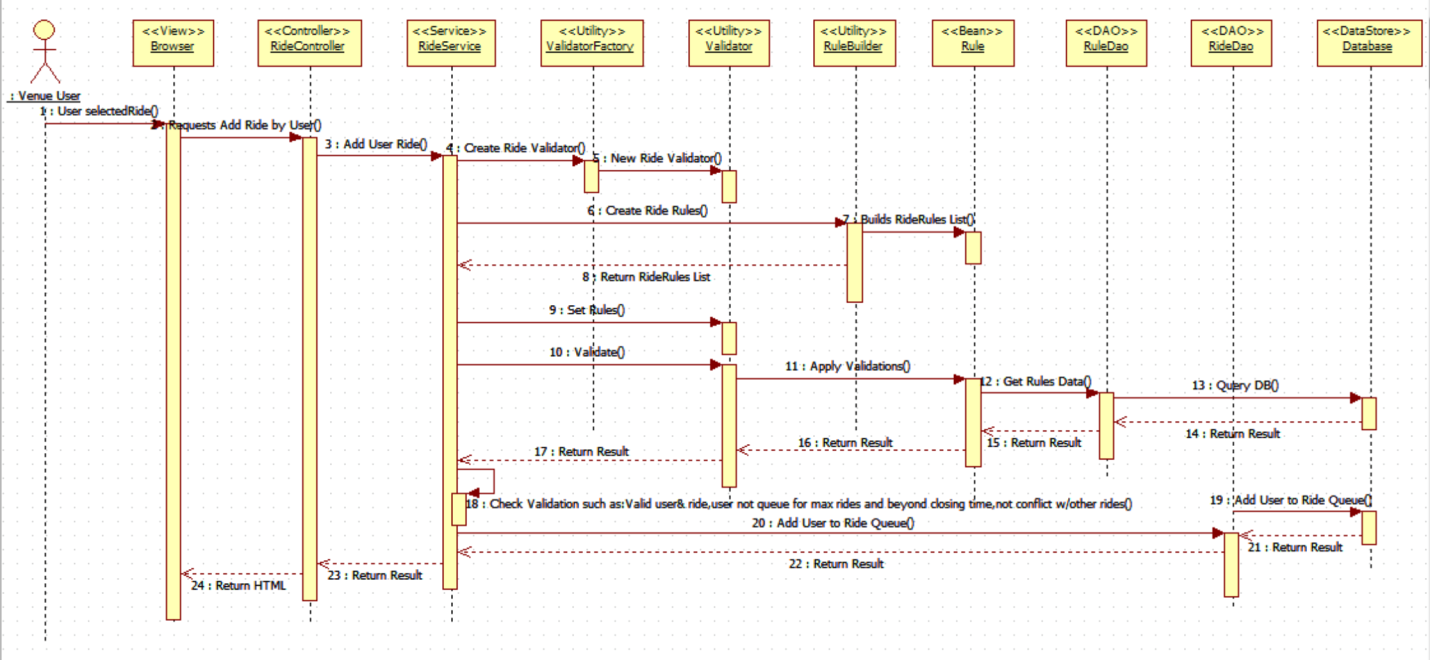
VQ05 - Register New User

****

VQ06 – Select Ride



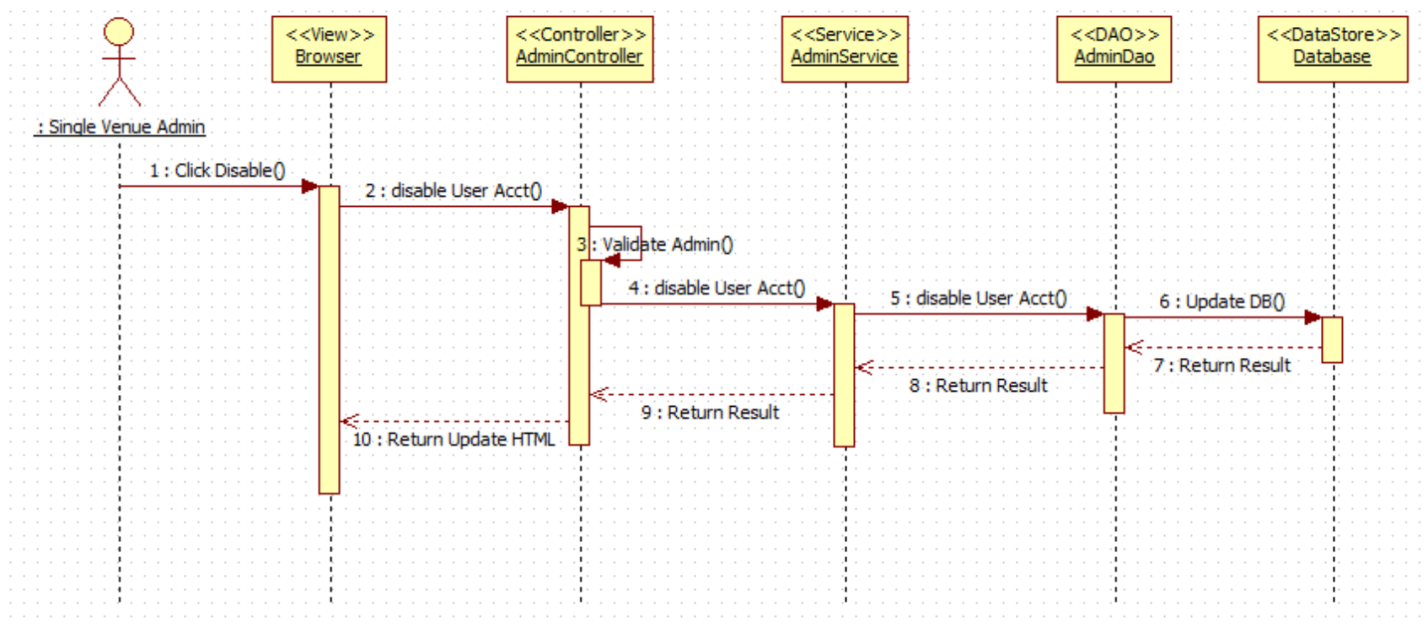
VQ07 – Add Ride (Queue User for Ride)



VQ08 - Edit User's Profile



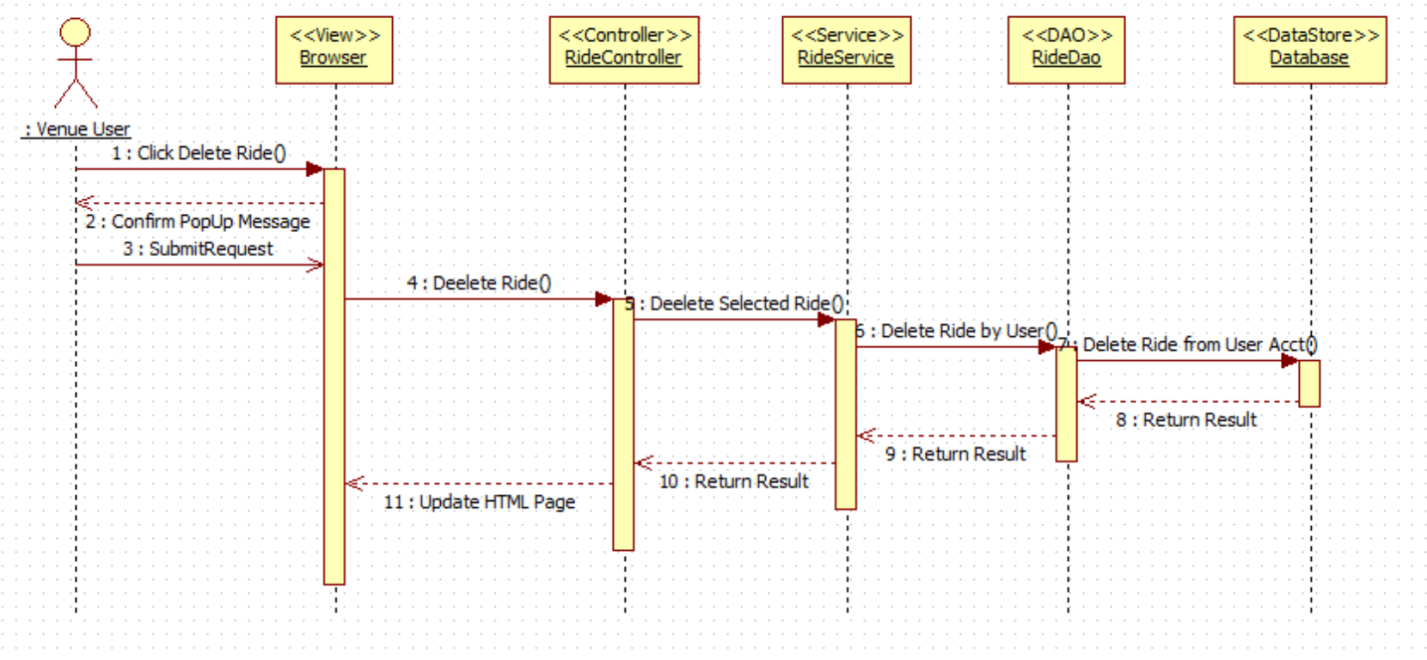
VQ09 - Disable User's Account



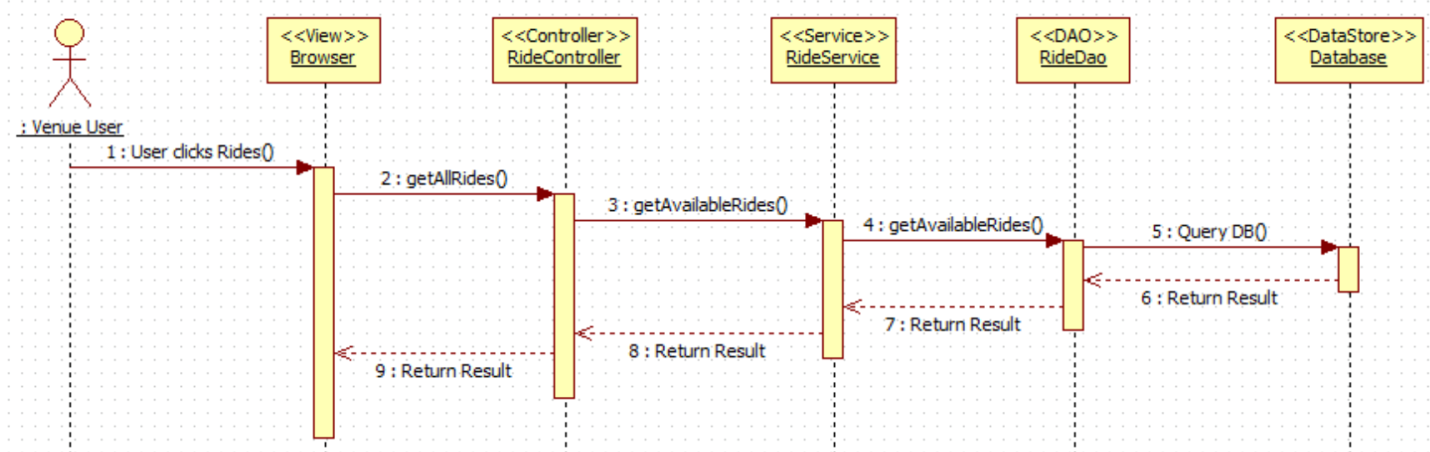
VQ10 - Enable User's Account

****

VQ11 – Delete a Registered Ride (De-queue User from Ride)



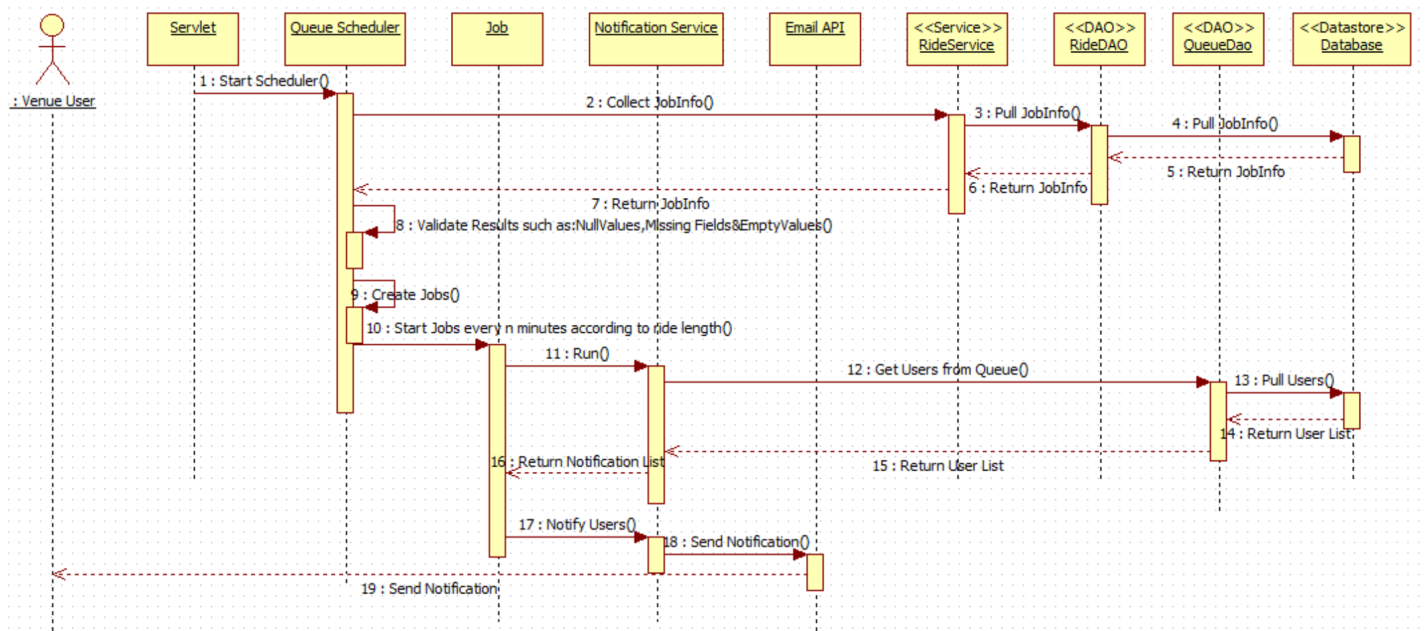
VQ12 – View Available Rides



VQ13 – View User Activity



VQ14 – Receive Notification (before ride time approaches)



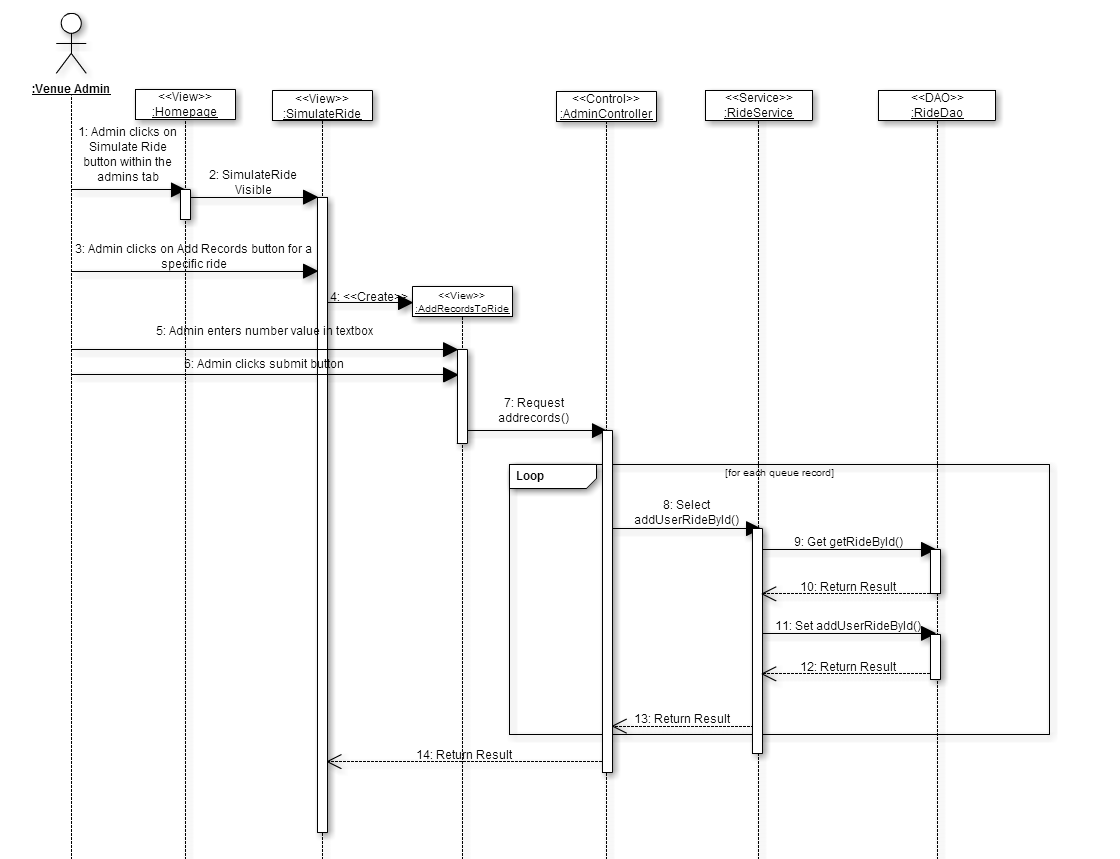
**VQ15 – Find Wait Times**

## VQ16 – Add Queue

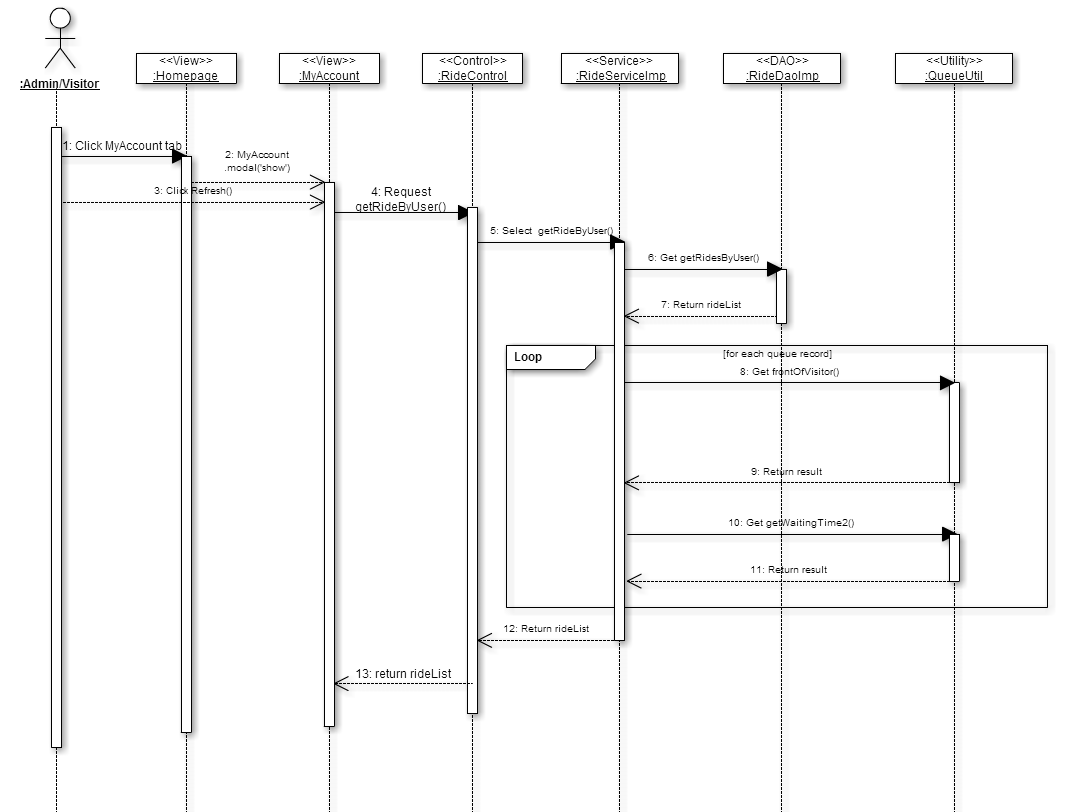


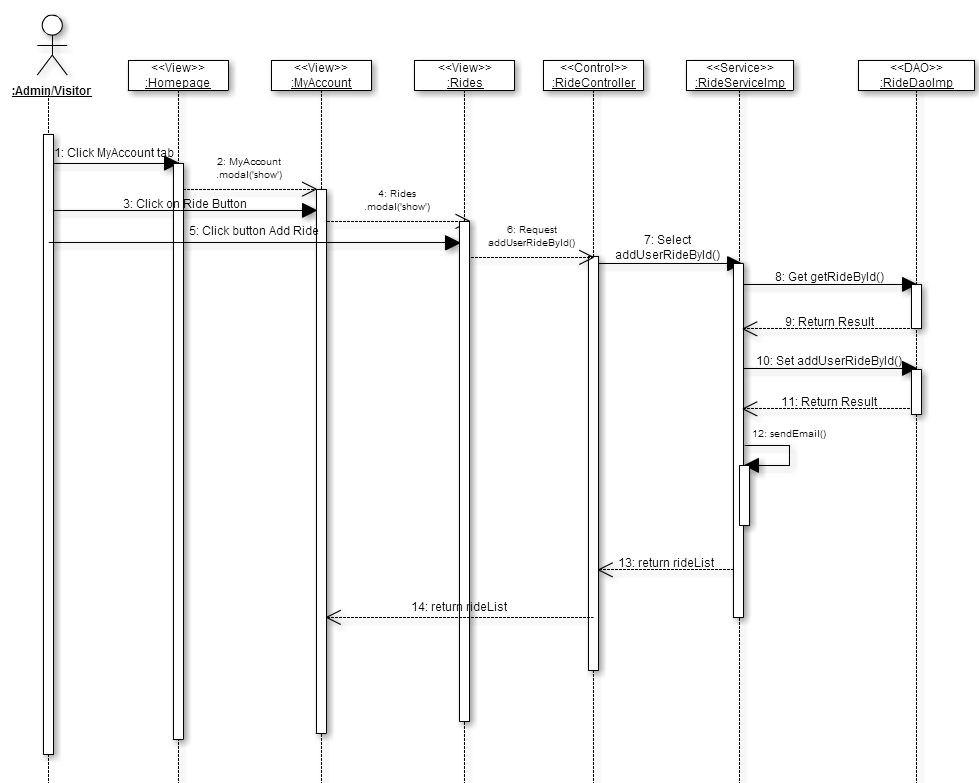
**VQ17 – Visitor DeQueue** 

**VQ18 – Simulate Add Ride**



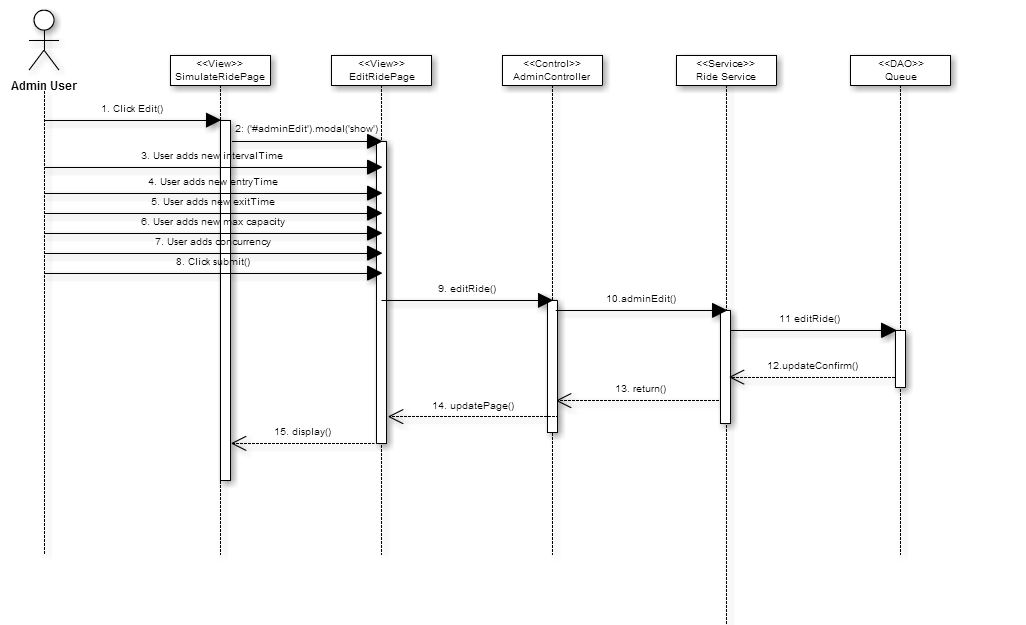
**VQ19 – Admin Simulate Dequeue for Rides**

**VQ20 – Dynamic Find Wait Time**

**VQ22 – Send Notification for adding ride**

**VQ23 – Send Notification for time warning**

**VQ24 – Admin Modify Rides**



**5.5. Appendix E - User Interface designs.**

Class Interfaces (code) for the subsystem Login Operation:

package com.virtual.queue.dao;

import com.virtual.queue.beans.User;

public interface LoginDao {

public Long validLogin(String user,String password, String code);

public User signIn(String userName,String password, String code);

public boolean signOut(String userName);

}

package com.virtual.queue.service;

import com.virtual.queue.beans.User;

public interface LoginService {

public User signIn(String userName, String password, String code);

public boolean signOut(String userName);

}

Class Interfaces (code) for the subsystem Notification API:

package com.virtual.queue.service;

import java.util.List;

import com.virtual.queue.beans.UserQueueInfo;

public interface NotificationService {

public List<UserQueueInfo> pullNotInfo(Integer rideId);

public List<UserQueueInfo> pullAllNotInfo();

public void notifyUser(Integer rideId) throws Exception;

public void notifyAllUsers() throws Exception;

}

package com.virtual.queue.handler;

import com.virtual.queue.beans.NotificationInfo;

import com.virtual.queue.exception.NotificationException;

public interface NotificationHandler {

public void notifiyUser(NotificationInfo info) throws NotificationException;

}

Class Interfaces (code) for the subsystem User Operation:

package com.virtual.queue.dao;

import java.util.List;

import com.virtual.queue.beans.User;

public interface UserDao {

public User getUser(String username, String passwd);

public String getCurrentlyAuthenticatedUserName();

public User getUserByToken(String token);

public String storeToken(Long userId);

public List<User> getAll();

public void addUser(User user);

public void updateUser(User user);

public void deleteUserById(Long id);

public User getUserByUserName(String userName);

public Boolean resetPassword(String userName, String securityAnswer,

String securityQuestion, String newPassword) throws Exception;

public User authenticateUser(String userName,String securityQuestion,String securityAnwser);

public User getUserById(long userId);

public boolean removeUserFromQueue(long userId,long rideId);

}

package com.virtual.queue.service;

import java.util.List;

import com.virtual.queue.beans.User;

import com.virtual.queue.request.UserPasswordResetRequest;

public interface UserService {

public User getUser(String username, String passwd);

public String getCurrentlyAuthenticatedUserName();

public User getUserByToken(String token);

public String storeToken(long userId);

public List<User> getAll();

public void addUser(User user);

public void updateUser(User user);

public void deleteUserById(Long id);

public User getUserByUserName(String userName);

public void resetPassword(UserPasswordResetRequest passwordReset) throws Exception;

boolean removeUserFromQueue(long rideId, long userId);

}

Class Interfaces (code) for the subsystem Queue Scheduler:

package com.virtual.queue.service;

public interface QueueService {

public boolean removeUserFromQueue(long rideId,long userid);

public boolean removeAllUsersFromQueue(long rideId);

}

package com.virtual.queue.dao;

import java.util.LinkedList;

import java.util.List;

import com.virtual.queue.beans.QueueInfo;

import com.virtual.queue.beans.UserQueueInfo;

import com.virtual.queue.beans.RideInfo;

import com.virtual.queue.beans.User;

public interface QueueDao {

public List<UserQueueInfo> pullInfo(Integer rideId);

public List<UserQueueInfo> pullAllInfo();

public LinkedList<User> getAllUserQueueForRide(long rideId);

public QueueInfo getQueueInfoByRideId(long rideId);

public boolean removeUserFromQueue(long rideId, long userid);

public boolean removeAllUsersFromQueue(long rideId);

public LinkedList<RideInfo> getRideListByUser(long userId) throws Exception;

}

Class Interfaces (code) for the subsystem Ride Operation:

package com.virtual.queue.dao;

import java.util.List;

import com.virtual.queue.beans.RideInfo;

import com.virtual.queue.exception.NotificationException;

public interface RideDao {

public List<RideInfo> pullRideInfo() throws NotificationException ;

public List<RideInfo> getRideByUser(Long userId) throws NotificationException;

public RideInfo getRideById(long rideId) throws NotificationException;

public boolean addUserRideById(Long rideId, Long userId);

public List<RideInfo> getAll();

}

package com.virtual.queue.builder;

import java.util.List;

import com.virtual.queue.rule.Rule;

public interface RuleBuilder {

public List<Rule> buildRules();

}

package com.virtual.queue.validator;

import java.util.List;

import com.virtual.queue.rule.Rule;

public interface Validator {

public void setRules(final List<Rule> rules) throws Exception;

public boolean validate(long userId,long rideId);

}

package com.virtual.queue.service;

import java.util.List;

import org.springframework.stereotype.Service;

import com.virtual.queue.beans.Ride;

import com.virtual.queue.beans.RideInfo;

import com.virtual.queue.beans.User;

import com.virtual.queue.exception.NotificationException;

public interface RideService {

public List<RideInfo> getAll();

public void addRide(Ride ride);

public void updateRide(Ride ride);

public void deleteRideById(Long id, Long userid);

public boolean removeRidebyId(String id);

public boolean addUserRideById(Long rideId, Long userid) throws Exception;

public List<RideInfo> pullRideInfo();

public RideInfo getRidebyId(long rideId) throws NotificationException;

public List<RideInfo> getRidesByUser(long userId) throws NotificationException;

}

package com.virtual.queue.rule;

public interface Rule {

public void loadData(long userId,long rideId) throws Exception;

public boolean apply();

void loadData();

}

**5.6. Appendix F - Diary of meeting and tasks.**

**Meeting 1**

**Time:** 6:15PM-7:15PM

**Date:** 1/22/2015

**Members Participated:**

Michael Lazo - scrum master

Kenneth Kon

Bernard Parenteau

**Topic**:

Talking about the requirements with Bernard for Virtual Queue 2.0

**Meeting 2**

**Time:** 6:15PM-7:15PM

**Date:** 1/29/2015

**Members Participated:**

Michael Lazo

Kenneth Kon - scrum master

Bernard Parenteau

**Topic**:

Discussed about the Database layout and the fields of Virtual Queue 2.0

**Meeting 3**

**Time:** 6:15PM-7:15PM

**Date:** 2/03/2015

**Members Participated:**

Michael Lazo - scrum master

Kenneth Kon

Bernard Parenteau

Topic:

Discussed on what User Story we should work on with the Product Owner/Mentor, for Sprint 1.

**Meeting 4**

**Time:** 6:15PM-7:15PM

**Date:** 2/09/2015

**Members Participated:**

Michael Lazo

Kenneth Kon - scrum master

Bernard Parenteau

**Topic**:

Discussed our progress on the Sprint 1, discussed any impediments.

**Meeting 5**

**Time:** 6:15PM-7:15PM

**Date:** 2/13/2015

**Members Participated:**

Michael Lazo

Kenneth Kon - scrum master

Bernard Parenteau

**Topic**:

Discussed Sprint 1 Review, if User Story was satisfy the requirements.

Discussed also discussed the impediments and need to refactor the previous design.

**Meeting 6**

**Time:** 6:15PM-7:15PM

**Date:** 2/17/2015

**Members Participated:**

Michael Lazo

Kenneth Kon

Bernard Parenteau - scrum master

**Topic**:

Discussed Sprint 2 Planning, convince Product owner Team Story has higher priority.

**Meeting 7**

**Time:** 6:15PM-7:15PM

**Date:** 2/28/2015

**Members Participated:**

Michael Lazo

Kenneth Kon - scrum master

Bernard Parenteau

**Topic:**

Discussed Sprint 2 Review, display the new design of the Refactored product.

Discussed also discussed the impediments and need to Refactor the previous design.

**Meeting  8**

**Time:** 6:15PM-7:15PM

**Date:** 3/3/2015

**Members Participated:**

Michael Lazo - Scrum master

Kenneth Kon

Bernard Parenteau - Product Owner

**Topic:**

Get more information on the stories we will be working on for Sprint 3 from our product owner.

**Meeting  9**

**Time:** 6:15PM-7:15PM

**Date:** 3/17/2015

**Members Participated:**

Michael Lazo

Kenneth Kon - Scrum master

Bernard Parenteau - Product Owner

**Topic:**

Talked about the progress of sprint 3. Clarified on Simulate Add Ride user story.

**Meeting  10**

**Time:** 6:15PM-7:15PM

**Date:** 3/22/2015

**Members Participated:**

Michael Lazo

Kenneth Kon - Scrum master

Bernard Parenteau - Product Owner

**Topic:**

Sprint 3 Review, showcased User Story Simulate Add Ride, Simulate Dequeue and Visitor Dequeue user story. Discussed what we would need to work on for Sprint 4.

**Meeting  11**

**Time:** 6:15PM-7:15PM

**Date:** 3/24/2015

**Members Participated:**

Michael Lazo - Scrum master

Kenneth Kon

Bernard Parenteau - Product Owner

**Topic:**

Sprint 4 requirement elicitation on the user stories assigned to us.

Mostly on Dynamic Find wait time and Edit Rides as Admin

**Meeting  12**

**Time:** 6:15PM-7:15PM

**Date:** 3/29/2015

**Members Participated:**

Michael Lazo

Kenneth Kon - Scrum master

Bernard Parenteau - Product Owner

**Topic:**

Sprint 4 review, both of our user stories were accepted. After that Product owner assigned us sprint 5 user stories.

**Meeting  13**

**Time:** 6:15PM-7:15PM

**Date:** 4/2/2015

**Members Participated:**

Michael Lazo - Scrum master

Kenneth Kon

Bernard Parenteau - Product Owner

**Topic:**

Sprint 5 requirement elicitation on the user stories assigned to us.

**Meeting  14**

**Time:** 6:15PM-7:15PM

**Date:** 4/10/2015

**Members Participated:**

Michael Lazo - Scrum master

Kenneth Kon

Bernard Parenteau - Product Owner

**Topic:**

Sprint 5 reviews, both of our user stories were accepted.

# 7.References