Final DOCUMENT

Pinecrest People Mover

Members:

Ricardo Martinez

Maurice Pruna

Mentor:

Gabriela Wilson

Instructor:

Masoud Sadjadi

December 8, 2014

Florida International University School of Computing and Information Sciences

CIS 4911 U01

© 2014 by Senior Project – People Mover.

All rights reserved. No part of this document may be reproduced or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of Senior Project – People Mover.

Abstract

This document is divided into ten sections: Introduction, Feasibility study, Project Plan, System Requirements, System Design, Detailed Design, System Validation, Glossary, the Appendix, and References. The Introduction provides the information in regards to the project. This information consists of explaining the problem definition, which includes the desire, expectation, and goal provided by the client, gives a description of the design methodology used, and provides a list of definitions and acronyms used in the document.

The Feasibility Study provides information about the current system, the descripition of the alternative solutions considered and the recommendations with explanation of why the solution was selected. The project plan goes into the project organization, shows the identifications of tasks, milestones, and deliverables, and lastly the cost of the project. The system requirements gives information in regards to the functional and nonfunctional requirements as well as the requirement analysis.

The system design section provides a high level explanation of the system with the use of diagrams, descriptions of the hardware, software, data storage and security as well. The Detailed Design gives the description of the structure for each subsystem using static models and dynamic models. The System Validation provides the test cases used to validate the system. The test cases were used to test the system and subsystems of the application.

Table of Contents

[1. Introduction 5](#_Toc405830351)

[1.1. Problem definition. 5](#_Toc405830352)

[1.2. Scope of system 5](#_Toc405830353)

[1.3. Overall development methodology 5](#_Toc405830354)

[1.4. Definitions, acronyms, and abbreviations 5](#_Toc405830355)

[1.5. Overview of document 5](#_Toc405830356)

[2. Feasibility Study 7](#_Toc405830357)

[2.1. Current System 7](#_Toc405830358)

[2.2. Description of alternative solutions considered. 7](#_Toc405830359)

[2.3. Recommendation with explanation of why the solution was selected. 7](#_Toc405830360)

[3. Project Plan 8](#_Toc405830361)

[3.1. Project Organization 8](#_Toc405830362)

[3.1.1. Project Personnel 8](#_Toc405830363)

[3.1.2. Hardware and Software Resources 8](#_Toc405830364)

[3.2. Identification of Tasks, Milestones and Deliverables 9](#_Toc405830365)

[3.3. Cost of the Project 10](#_Toc405830366)

[4. System Requirements 11](#_Toc405830367)

[4.1. Functional and Nonfunctional Requirements 11](#_Toc405830368)

[4.2. Requirements Analysis 14](#_Toc405830369)

[4.2.1 Scenarios 14](#_Toc405830370)

[4.2.2 Use case model 17](#_Toc405830371)

[5. System Design 0](#_Toc405830372)

[5.1. Overview 0](#_Toc405830373)

[5.2. Subsystem Decomposition 0](#_Toc405830374)

[5.3. Hardware and Software Mapping 1](#_Toc405830375)

[5.4. Persistent Data Management 2](#_Toc405830376)

[5.5. Security/Privacy 3](#_Toc405830377)

[6. Detailed Design 4](#_Toc405830378)

[6.1. Overview 4](#_Toc405830379)

[6.2. Static model 5](#_Toc405830380)

[6.3. Dynamic model 5](#_Toc405830381)

[7. System Validation 6](#_Toc405830382)

[7.1. Subsystem Tests 6](#_Toc405830383)

[7.2. System Tests 8](#_Toc405830384)

[7.3. Evaluation of Tests 37](#_Toc405830385)

[Subsystem Test Results 37](#_Toc405830386)

[System Test Results – Sunny Day 38](#_Toc405830387)

[System Test Results - Rainy Day 39](#_Toc405830388)

[8. Glossary 41](#_Toc405830389)

[9. Appendix 42](#_Toc405830390)

[9.1. Appendix A - Project schedule (Gantt chart or PERT chart) 42](#_Toc405830391)

[9.2. Appendix B – All use cases with nonfunctional requirements. 45](#_Toc405830392)

[9.3. Appendix C – User Interface designs. 53](#_Toc405830393)

[9.4. Appendix D – Analysis models (static and dynamic) 58](#_Toc405830394)

[9.5. Appendix E – Design models (static and dynamic) 59](#_Toc405830395)

[9.6. Appendix F – Documented Class interfaces (code) and constraints. 76](#_Toc405830396)

[Class Hierarchy 76](#_Toc405830397)

[Interface Hierarchy 77](#_Toc405830398)

[9.7. Appendix H – Diary of meeting and tasks for the entire semester. 78](#_Toc405830399)

[10. References 87](#_Toc405830400)

# Introduction

This section gives a brief indication about what is the problem presented by the community of Pinecrest in the person of our mentor and client. The have been using their trolley system called People Mover (PPM)

## Problem definition.

Pinecrest government would like to design a Web tracker and a Mobile tracker to show residents routes, hours of operation, real-time trolley location (as a list and as an interactive map) and allow for automatic notifications for arrival at user’s favorite stops. The mobile tracker should work well on iPhone and android devices.

## Scope of system

The Pinecrest People Mover is a free transit bus service operated by the Village of Pinecrest connecting their neighborhoods and schools. Middle and high school students who do not qualify for bus service from the school district mostly use it

## Overall development methodology

The agile software development model is what is being used to develop the system. We specified the scope of the system, analyzed the use cases needed, and defined the functional and non-functional requirements in the Requirement Document (RD). In this Design Document, we will use the same analysis model from the Requirement Document to show the design objects, system architecture, and subsystem decomposition that will be part of the process. The agile system will guarantee that the documentation process is well designed but at the same time we are going to be involved in the developing process to get a fast delivery of the system to the client. The UML diagrams as part of the analysis model are required to aid in designing the system. The use case diagrams from the RD makes the creation of the class diagrams and sequence diagrams easier to define.

## Definitions, acronyms, and abbreviations

PPM: Pinecrest People Mover

SRS: Software Requirements Specification

UML: Unified Modeling Language

## Overview of document

* Chapter 2 describes the Feasibility Study, which includes:
  + Description of the Current System
  + Description of alternative solutions considered
  + Recommendation with explanation of why the solution was selected
* Chapter 3 describes the Project Plan, which includes:
  + Project Organization
  + Identification of Tasks, Milestones, and Deliverables,
  + Cost of the Project
* Chapter 4 describes the System Requirements, which includes:
  + Functional and Nonfunctional Requirements
  + Requirements Analysis
* Chapter 5 describes the System Design, which includes:
  + Overview of the system
  + Subsystem Decomposition
  + Hardware and Software Mapping
  + Persistent Data Management
  + Security/Privacy
* Chapter 6 describes the Detailed Design, which includes:
  + Overview of the design
  + Static Model
  + Dynamic Model
  + Code Specification
* Chapter 7 describes the test cases used for validation, which includes:
  + Subsystem Tests
  + System Tests
  + Evaluation of Tests
* Chapter 8 contains the glossary
* Chapter 9 contains the appendices
* Chapter 10 contains any references used

# Feasibility Study

## Current System

The village of Pinecrest has two mover routes that are currently operating as a free service (Palmetto Middle School and Palmetto Senior High School). Each one has a north and a south path. Pinecrest citizens mainly use the movers as alternative to get to the local schools. They have no way to know with precision the wait time for next mover while are waiting on a stop, or the nearest stop to their actual position. For that reason Pinecrest government is currently involved on the installation of GPS devices in the movers. A new system will be created to serve the clients desired purposes.

## Description of alternative solutions considered.

The first alternative is geared to implement the same kind of application for each of the platforms listed. In this case we could have a personalized design taking to account the specifications for each platform. The interfaces would be different, creating some difficulties in the end user. The speed is important in order to get a proper tracking response, depending too much of the phones hardware requirements.

The second alternative would be one web application with responsive design that will work on all the platforms. After analyzing different frameworks we found given the real-time characteristic of this project we should select a framework that allow continuous calls without losing in functionality. Hence, to address this feature we found Sencha Touch framework that uses JavaScript, HTML5 and CSS3. Sencha Touch takes advantage of hardware acceleration to deliver an incredible application.

## Recommendation with explanation of why the solution was selected.

Given the characteristics of the problem, we wanted to handle a solution easy to implement, and easy to maintain. Designing the web application only, could reduce the time where the system could be functional for all the platforms. The maintenance time and cost would be reduced having a web only application. Also the interface will maintain the same format making easy to reproduce the same functionalities in any platform.

Analyzing the characteristics of this project we recommend the use of Sencha Touch. The use of this technology will decrease the development time, also providing the possibility of future maintenance in a centralized system.

# Project Plan

This chapter will explain the schedule for the development plan. The roles of each team member are outlined in this section including their assigned roles. The hardware and software resources needed for the development of this project are also provided. Tasks, milestones, and deliverables are also outlined.

## Project Organization

The team implementing this project consists of two students. Being the fact that the team is small, each team member will require having multiple roles, which include project manager, developer, tester, etc. The team members will have to use the resources allocated to them and must be approved by the client in order for the development of project to be a success

### Project Personnel

The structure of the project organization can be considered hierarchal. Each student will be in constant communication with each other during the development process. The students will be presenting their progress to Masoud Sadjadi, the professor, and Gabriela Wilson, the client. Masoud Sadjadi will be providing feedback to Ricardo Martinez and Maurice Pruna in how to improve project.



|  |  |  |
| --- | --- | --- |
| Name | Roles | Contact |
| Gabriela Wilson | Client/Mentor | gwilson@pinecrest-fl.gov |
| Masoud Sadjadi | Professor | sadjadi@cs.fiu.edu |
| Ricardo Martinez | Front-End Developer, Tester | rmart071@fiu.edu |
| Maurice Pruna | Back-End Developer, Tester | mprun003@fiu.edu |

### Hardware and Software Resources

**Required Hardware**

* iOS device
* Android device
* Server

**Required Software**

* Microsoft Word
* Sencha Touch
* Cacoo
* Mockup Builder
* Microsoft Project
* GitHub
* Trello
* SourceTree
* FIU Moodle

## Identification of Tasks, Milestones and Deliverables

|  |  |
| --- | --- |
| ID | Task Name |
| 1 | Meet with Partner |
| 2 | Meet with mentor |
| 3 | Create high level requirements |
| 4 | UI Mockup Designs |
| 5 | Create Use Cases |
| 6 | Create Schedule |
| 7 | Use Case Diagrams |
| 8 | Requirements Document |
| 9 | Project Plan Document |
| 10 | Feasibility Study |
| 11 | Retrieval of client server and database information |
| 12 | Map layouts with routes and trolley tracking |
| 13 | Begin Implementations |
| 14 | Design Test Cases based on use case criteria |
| 15 | Present to client product |
| 16 | Present project to class |
|  |  |

|  |  |
| --- | --- |
| Milestones | Projected Date |
| Finalized Requirements Document | 10/5/14 |
| Finalized Design Document | 10/13/14 |
| Completion of Implementation | 11/28/14 |
| Testing Phase Completed | 12/4/14 |
| Final Deliverable | 12/11/14 |
| Provide finished product to client | 12/12/14 |

## Cost of the Project

|  |  |
| --- | --- |
| Human resource | Work(hours) |
| Ricardo Martinez | 360 |
| Maurice Pruna | 360 |

|  |  |  |
| --- | --- | --- |
| Non-human Resources | Quantity | Cost |
| Computers | 2 | $0.00 |
| Server | 1 | $0.00 |
| Apache Server | 1 | $0.00 |
| Apache Tomcat | 1 | $0.00 |
| Spring Framework | 1 | $0.00 |
| Spring Tool Suite | 2 | $0.00 |
| MYSQL | 1 | $0.00 |
| Sencha Architect, Sencha Cmd | 1 | $0.00 |

# System Requirements

This chapter will contain the detailed requirements for the application. Also there will be information about the different scenarios, uses cases model, class diagram, and sequential diagram.

## Functional and Nonfunctional Requirements

1. **PPM\_01: Display Hours of Operation**

The system shall provide the user with the information in regards to the hours of operation.

* Usability: The user will access this function in 1 click from home.
* Reliability: Low mean time to failure since this function would be one of the main buttons on the menu
* Performance: Data will load in less than 2 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_02: Trolley Alerts**

The system shall provide the user with any updates in regards to the trolleys. This can be a malfunction to a trolley, delayed departure, and not-in-service for that day

* Usability: This data doesn’t require any action of the user to get activated.
* Reliability: Low mean time to failure as this functionality would be very beneficial to the users
* Performance: Notifications should be sent as soon as information is entered by admin
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_03: Estimated Time for a Favorite Stop**

The system shall provide the estimated time for a selected stop marked as favorite.

* Usability: The user will access this function in 1 click from home.
* Reliability: Low mean time to failure since this is a main functionality of the app
* Performance: Data will load in less than 2 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_04: Set Favorite Stop**

The system shall allow the user set a stop as a favorite stop.

* Usability: The user will access this function in the list stop view.
* Reliability: Low mean time to failure as user should be able to see estimated time for arrivla
* Performance: Information would be available within 2 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_05: Remove Favorite Stop**

The system shall allow the user remove the favorite stops entered.

* Usability: Easy as using pressing a button. No training required
* Reliability: Low mean time to failure since user should be able to set and remove favorite stops.
* Performance: Information would be deleted within 2 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_06: List Stops**

The system shall provide the user a list of all the stops given a route and direction.

* Usability: No training required. Can be seen in two clicks from the home screen.
* Reliability: Low mean time to failure since this information is retrieved from the database.
* Performance: Data will load in less than 2 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_07: Map Routes**

The system shall provide the user a map given a route and direction.

* Usability: No training required. App opens up to map view.
* Reliability: Low mean time to failure since this is one of the main functionalities of the app
* Performance: Data will load in less than 2 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_08: Estimated Arrival Time given Stop**

The system shall provide the estimated time for a selected stop.

* Usability: The user will access this function in 1 click from home.
* Reliability: Low mean time to failure as this is a main functionality of the app.
* Performance: Data will load in less than 2 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_09: Find Nearest Stop**

The system shall provide the nearest stop to the user’s geolocation.

* Usability: The user will access this function in 1 click from home.
* Reliability: Low mean time to failure since this depends on the information tracked by the GPS
* Performance: Data will load in less than 2 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_10: Arrived to Location Alert**

The system shall send an email notification to user once trolley has arrived or is near to arrive (given the time defined in for a favorite stop) to selected location

* Usability: No training required.
* Reliability: Low mean time to failure since this depends on the information tracked by the GPS
* Performance: Notification should be sent no more than 1 minute.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_11: Login**

The system shall check if the username and password of a user are correct and show favorite stops.

* Usability: No training required. Login will contain input boxes.
* Reliability: Low mean time to failure as this functionality will allow user many functionalities.
* Performance: Login should take no longer than 10 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_12: Register**

The system shall save the username and password of a user.

* Usability: No training required. Registration view will contain input boxes.
* Reliability: Low mean time to failure as this functionality allows system to have users.
* Performance: Registration should take no longer than 20 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_13: Request Stop**

The system shall provide the user with a form view allowing the user to request a stop. They system should send an email to the pertinent authority and reply with an alert box saying ‘Email Sent’. The Village of Pinecrest reviews routes twice a year (Aug and Jan) based on suggestions from interested residents. Please review route information regularly to see if we were able to accommodate your request’

* Usability: No Training required. The view will contain input boxes where the user can type in information.
* Reliability: Low mean time to failure as this functionality allows interaction from the user.
* Performance: Request should take no longer than 20 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_14: Report Problem**

The system shall provide the user with a form view allowing the user to report a problem. They system should send an email with the problem message in question and/or a picture documenting the problem to the pertinent authority and reply with an alert box saying ‘Email Sent’ .

* Usability: No Training required. The view will contain input boxes where the user can type in information.
* Reliability: Low mean time to failure as this functionality allows interaction from the user.
* Performance: Request should take no longer than 20 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

1. **PPM\_15: Send Feedback**

The system shall provide the user with a form view allowing the user to send feedback. They system should send an email with the feedback message in question to the pertinent authority and reply with an alert box saying ‘Email Sent’ .

* Usability: No Training required. The view will contain input boxes where the user can type in information.
* Reliability: Low mean time to failure as this functionality allows interaction from the user.
* Performance: Request should take no longer than 20 seconds.
* Supportability: This functionality should be supported by any internet browser, iOS and Android Devices

## Requirements Analysis

Analysis models – contains the complete functional specification and is mainly for the designers and programmers. This section describes the diagrams in the Appendices B - D and validates the models against the use cases.

### 4.2.1 Scenarios

1. **Scenario - PPM\_01: Display Hours of Operation**
   1. The user has downloaded the app and wants to check what time the Pinecrest People Mover is operating. The user clicks on the Service Time button located in a tab panel on the Main Menu. The system retrieves the information from the database and displays information on the interface. The user will not be able to see at what time the trolley is working.
2. **Scenario – PPM\_02: Trolley Alerts**
   1. The user has the app downloaded on the phone device and wants to verify that the trolleys are operating on schedule. The user clicks on the ‘More’ button and when system has redirected to another view, he/she presses the Trolley Alerts button. The system retrieves any information saved in the database and displays on the interface that the trolley for Palmetto High Route is not functioning.
   2. The system will send notifications to all users in the app indicating that the movers are out-of-service. When the user opens the app a message will appear indicating that there has been a problem with the movers. If the user has provided an email or cell phone, a notification will be sent to them ahead of time
3. **Scenario – PPM\_03: Estimated Time for Favorite Stop**
   1. The user has selected a certain stop. The system should provide the user with information about the closets bus with that stop in its route. Such information should include the ID number, route information, the mover’s next stop, and estimated time of arrival to the user’s selected stop. The estimated time of arrival should be in minute format.
4. **Scenario – PPM\_04: Set Favorite Stop**
   1. The user knows of someone using the north route for the Palmetto Senior High mover. Once the mover, has arrived to the Palmetto Senior High stop, the system will send a notification to the user indicating that the mover has arrived safely to the location. This will require email and cell phone information so the notifications can be sent.
5. **Scenario – PPM\_05: Remove Favorite Stop**
   1. The user wants to remove their current favorite stop from the list. They have realized that they no longer us that stop, so they decided to click on the favorites button on the home page. They click on their favorite stop and a view is showing, where an option appears and displays ‘Remove from Favorites’. The user selects the option and the stop no longer appears on the list.
6. **Scenario –PPM\_06: List Stops**
   1. The user is using the application and wants to see all the stops a mover will make in a certain route. In the menu there will be a ‘List route’ button which will direct the user to another page indicating to choose which destination, Palmetto Middle School or Palmetto High School. He/she then chooses the route. Once the route has been selected, a list of all the stops is now being shown to the user
7. **Scenario – PPM\_07: Map Routes**
   1. The user is using the application and clicks on the view map button. The system will direct the user to a site with a Google map implemented. This map will have four different colors, which will be used to identify the different routes, and a People Mover icon, which will represent the actual location of the bus. The map will be considered real-time by showing the current location of the mover, so if the mover is driving to the next stop, the icon will be moving as well.
8. **Scenario – PPM\_08: Estimated Arrival Time Given Stop**
   1. The user is currently looking at the stop list for a given route. The list will also display the estimated arrival time for each stop. The user now knows the status of the trolley and around what location the trolley is currently located.
9. **Scenario – PPM\_09: Find the Nearest Stop**
   1. The user is on the Pinecrest People Mover application and clicks on the ‘Near Me’ button. The system should use the GPS feature to determine the user’s location and direct the user to a pin indicating the closest stop to their current location. They system will provide the user with the address of the stop and its route information.
10. **Scenario – PPM\_10: Arrived to Location Alert**
    1. The user has selected their favorite stop and the menu for time of alert appears. This menu give you options to either get instant notification when the trolley has arrived and ranges to up to 15 minutes before trolley arrives. The user selects the instant notification. Once the trolley has arrived at the favorite spot, they receive a SMS text notification and email notification indicating the trolley has arrived.
11. **Scenario – PPM\_11: Login**
    1. The user has already registered for the app. They want the app to retrieve all of their information and allow alert functionalities for their favorite spots. They click on the ‘More’ button and select the Sign in button. They enter their username and password and now the system will allow the alerts functionalities and their selected favorite spots will now appear.
12. **Scenario – PPM\_12: Register**
    1. The user wants to have the alert functionalities and keep the list of the favorite spots. They click on the ‘More’ button and select the Register button. They will enter their username, password, email, and phone number. Once they have selected the register button they will receive an email confirmation and can now use that information to sign in.
13. **Scenario – PPM\_13: Request a Stop**
    1. The user has moved location but there is no stop for the trolley near by. He/she wants to send a request so when they click on the ‘More’ button, they then click on the request a stop button. They will fill out the needed fields and submit the request. They will then see a confirmation message.
14. **Scenario – PPM\_14: Report Problem**
    1. The user see a tree blocking the way. He/she wants to report the problem to the transportation department. He/she click on the ‘More’ button, they then click on the report problem button. They will fill out the needed fields and submit the report . They will then see a confirmation message.
15. **Scenario – PPM\_15: Send Feedback**
    1. The user wants to state how great the app is. He/she wants to send the feedback to the transportation department. He/she click on the ‘More’ button, they then click on the send feedback button. They will fill out the needed fields and submit the feedback . They will then see a confirmation message.

### 4.2.2 Use case model

This sections is geared to point out the different steps within our engineering process where there is interaction between the system and the different actors that use the system. It provides with an overall idea of what the system needs to meet the requirements. The use case model gives a high level outlook of the system.

Once finalized, this system will provide a solution for Pinecrest’s citizens to get information about the town trolley system. The Use Case Diagram shown in Appendix B provides with all the functions that an actor could perform to get the requirements that are part of the system. For this scheme we identified two actors and 15 use cases that will be implemented. The interaction between the actors and the uses is cases is defined as follow:

Actor 1 Regular User:

* **Display Hours of Operation**
* **Trolley Alerts**
* **List Stops**
* **Map Routes**
* **Estimated Arrival Time Given Stop**
* **Find the Nearest Stop**
* **Request a Stop**
* **Report Problem**
* **Send Feedback**

Actor 2 Registered User:

* **Display Hours of Operation**
* **Trolley Alerts**
* **Estimated Time for Favorite Stop**
* **Set Favorite Stop**
* **Remove Favorite Stop**
* **List Stops**
* **Map Routes**
* **Estimated Arrival Time Given Stop**
* **Find the Nearest Stop**
* **Arrived to Location Alert**
* **Login**
* **Register**
* **Request a Stop**
* **Report Problem**
* **Send Feedback**

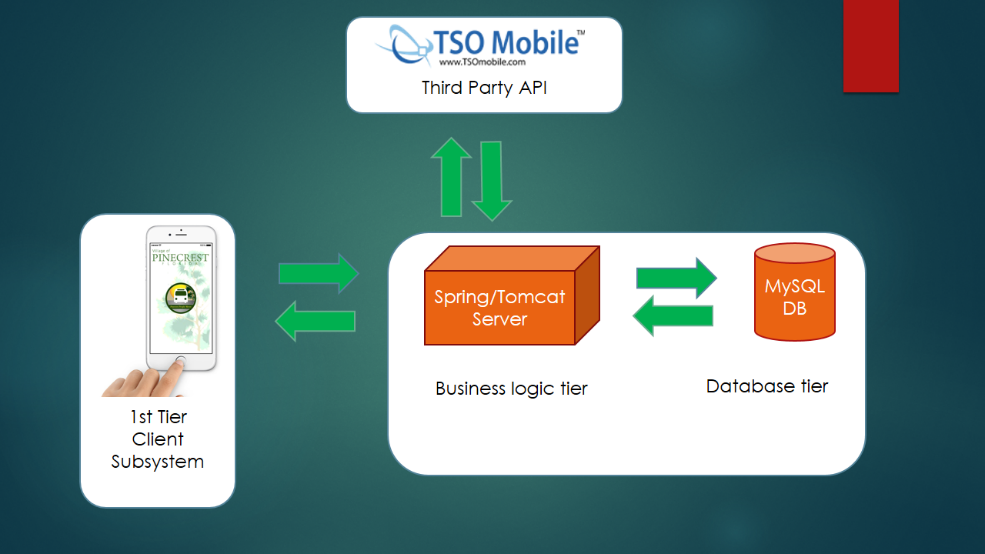
# System Design

Along this chapter we will cover our system and subsystem design. We will present:

* High-level description of the system design.
* Detail description of each of the major subsystems.
* Maps of the interaction between the hardware and the subsystems.
* Structure of the data to be stored.
* Authentication processes and security in general.

## Overview

The design chosen to be used in our system is based in a three tier architecture. Given that our system has an important part mobile oriented, we thought that a light weight data processing would be beneficial, also this represents easier scalability and better handling of the security. Our system also implements a client server architecture given that our business logic and database reside in the same server and they exchange resources and services. The subsystem defined as client side is an application developed using Sencha Touch framework. The server subsystem is composed by a tomcat server with a resident web app using Spring Framework over Java and MySQL as database server The server subsystem takes care of the resources requested from the client subsystem, database processing and requests to a third party API that provide data of the trolleys as well.

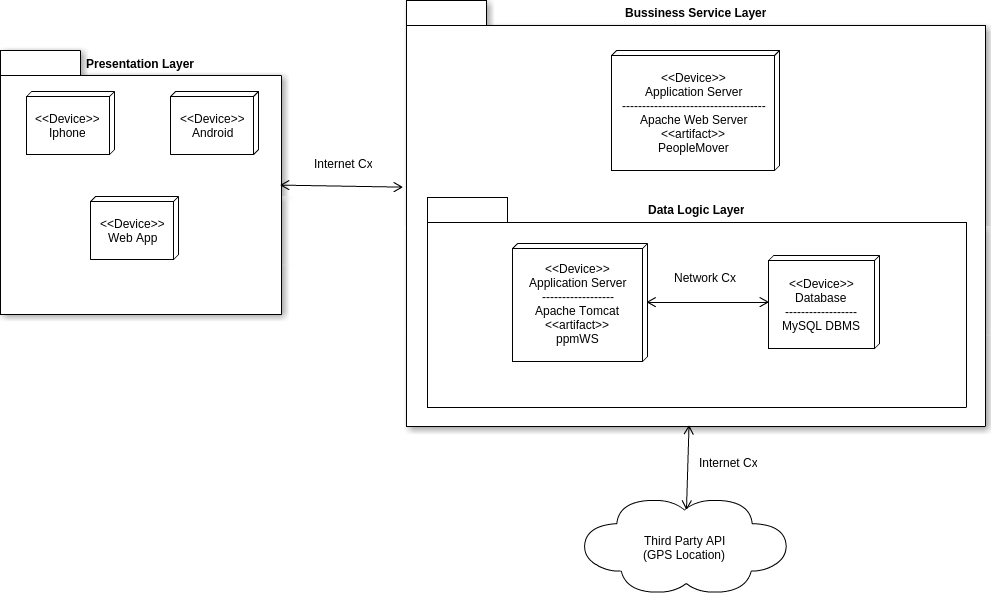


## Subsystem Decomposition

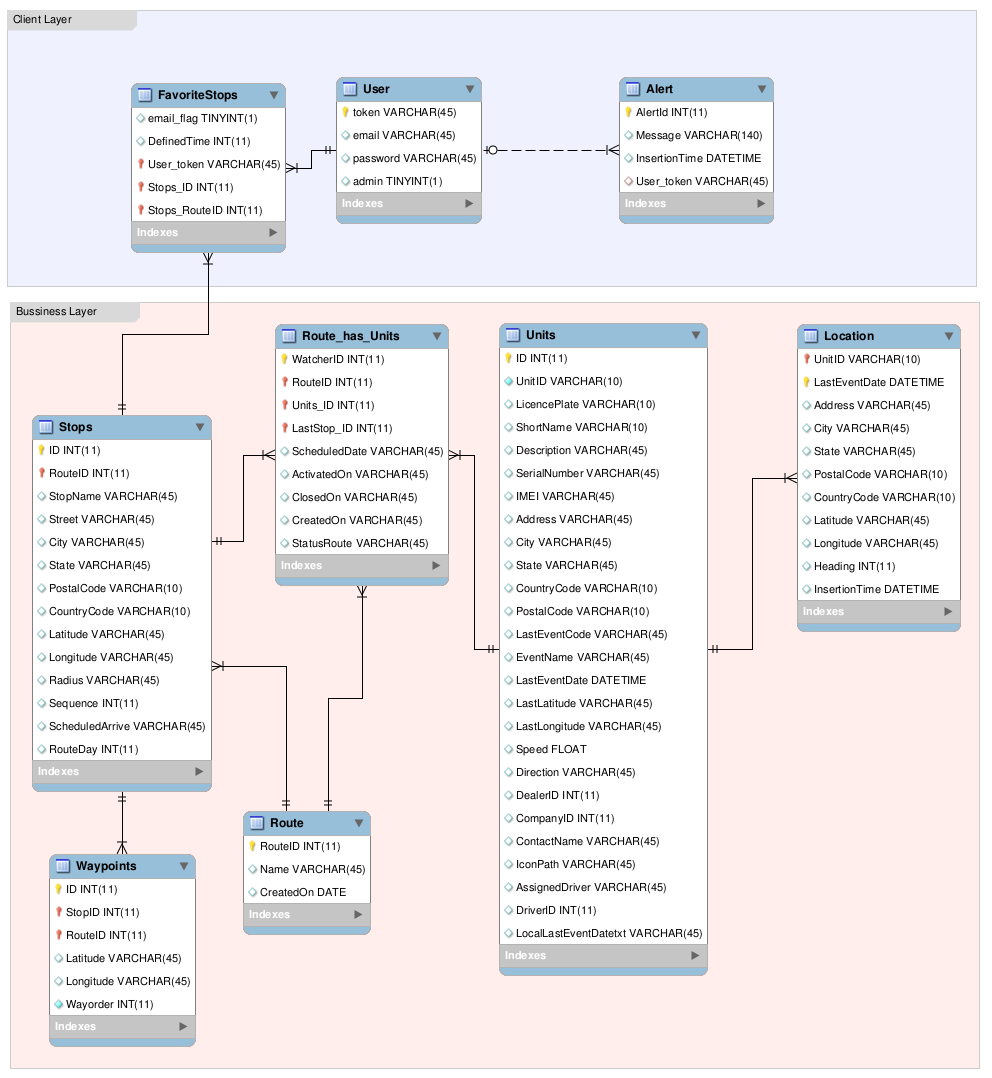
This system is composed by 4 subsystems. The first one is the Client subsystem which will manage how the interface is delivered to the user. This subsystem will provide to regular and registered users with features to allow tracking of trolleys, find near stops and send message to the system. The Authentication subsystem will provide with the functionality to perform specific task to registered users (like define favorite stops, define alerts). This subsystem resides part in the client side and part in the business logic tier. The part in the business logic tier will perform most of the validation process and will manage the information request corresponding to registered users. The Web Service subsystem will manage the interaction with the third party API that will provide the system with basic trolley information, will store and retrieve information from the database, and will deliver information to the Client Subsystem in the form of web service. The Email subsystem will take care of process all the email transmission requested from the Client subsystem and will be used as method to manage the stops alerts.

## Hardware and Software Mapping

The hardware and software mapping is shown below:



## Persistent Data Management



## Security/Privacy

The user’s passwords will be encrypted using a hashing function. This password will be encrypted before being sent to the Database. In case of an intruder get to the database it will not be able to retrieve the original password. The regular users will not be able to save any favorite stop and/or define an alert for a favorite stop. All the authentication process from the client side is validated through the People Mover API. This API is the one that connects to the database retrieves the necessary information and send it back to the client side.

# Detailed Design

The detail design of the system is the step prior to the implementation process. It will a more detailed specification in a level that is easier enough to get into the implementation process in a straight way. All the modules present on the system are specified pursuing as much detail as possible. It will be used different modeling procedures that are part of the detail design

## Overview

The application we wish to create is composed of four subsystems. The first subsystem is the client subsystem. It has the overall forms and visual information needed on the client side. Users can check the trolley location on the map, also list stops and see the estimated time to arrival, check trolley alerts and send feedback, problems and requests to the system. The business layer contains the other three subsystems. The Web service subsystem is the one that process all the data communication between the client side and the business layer. It also contains classes to process the information retrieves from the TSO Api regarding the trolleys. All this information is stored and retrieves from a MySql database. . The Favorite subsystems allows users to create, remove favorite stops. In this subsystem users can also activate and deactivate alarms and define a time when they want to receive those. The Alarm subsystem is in charge of sending the emails at the time defined by the users that registered their favorite stops to receive alarms.

## Static model

A static model is a representation of the system itself. It doesn’t represent the time or the sequence of events involved on the system. The object diagram is geared to display a set of objects with attributes using the relation among them. By the other hand a class diagram show the overall structure of the system using representation of the classes, attributes and methods and the relation among the classes as well. All these diagrams are displayed on the appendix C of this document.

The diagrams in appendix C are part of the business logic tier. The User Authentication class diagram is the subsystem that manages the user login, registration, and the information that will be provided to the Client subsystem. The Web Service subsystem is defined by the information requested to the external API, it process that information and is send it back to the client Subsystem.

## Dynamic model

Dynamics models are a representation of the system taking to account the time. The most used ones are the sequence diagrams and states machines. The sequence diagrams are representations of the interaction between objects and classes given by a sequence of events along a time line which helps to display the functionality. By the other hand states machines displays a class object and the iteration of this object through all the system. The principal use of these diagrams is to display the transition events of the objects through the system in a manner that developers could understand easily how the interaction of the user using with the system takes place. These types of diagrams could be found in the Appendix D of this document.

The email subsystem main functionality is set on the filter. As the users interacts with the filters by enabling checkboxes and selecting items in the lists, flags are enabled for each interaction. At the time of displaying the recipients list the setJudgeFilters() and setStudentFilters() functions build another function that is then passed to the store for judges and students respectively. This function will be using the flags previously enabled and state each record in the store to determine if that record has to be included in the filter. When the user deselects either student or judge from the list in the recipient lists step, the sent flag for that record is disabled. Finally on the send step the subsystem fetches the filtered data for students and judges and builds the email bodies by replacing the fields on the email template with the record values. Also any record with the sent flag disabled is ignored. A simple loop iterates over the built emails and calls the PHP API to send the email.

The Live Stats subsystem makes extensive use of the Sencha Touch’s built in data binding features to populate the list, rendered graph, and apply the corresponding grouping. The popup that appear when a user selects an element in the list or graph call on the itemtap() function. This function will map the element chosen with the corresponding record from the store and process the record for the adequate information to display in the popup.

# System Validation

This chapter coves our system validation, implicating all the testing process of the system. This chapter is divided into two sections. Section 7.1 Subsystem Tests, which covers all the testing done for each subsystem in specific preceding the integration process. Section 7.2 System Tests, covering all the system tests directed after integrating the modules of the system.

## Subsystem Tests

|  |  |
| --- | --- |
| Subsystems groups | Test Description |
| Frontend UI (Client Subsystem) and backend Web Services | This tests that all the request generated on the frontend are interpreted in the backend webservice returning the result expected. This will only validate that the string to perform the request is correct and that the return is the json object projected. |
| Backend Web Services and backend Favorite Alarms | This tests that a favorite alarm can be added, removed, or set defined time to trigger through the Web Service, and that favorite alarm triggers a message. This Email message is sent to the user that registered that favorite alarm. |
| Backend Web Services and backend Email Service | This tests checks that Emails generated are being received by the correct recipients with the defined template and attachments if so. |

The system is comprised of four subsystems. Among these four subsystems there were three groups that needed to have subsystem testing done on them. These three groups are as follows, the frontend UI (client subsystem) and the backend Web services, the backend Web services and the backend Favorite Alarms, the backend Web services and backend Email service. We will now identify the subsystem testing that took place for each of these groups.

This project had very little subsystem testing that required to be completed. During the planning stage of designing our application we did it in a way that the exchange of information between subsystems were as minimal as possible. We decided to do this so that we would have very little connection between subsystems. In that way our system will become more consistent.

For the first subsystem group, we created the Web services taking to account that few information will be passed as parameters by the client subsystem to each of them and sent in the form of Post request. JSON format was returned with minimum information possible. Continues testing was accomplished using browser Rest tools, given the fact that the frontend and backend interaction is only through web services requests. We conducted manual testing of the interaction of this method between the two subsystems

For the second subsystem group, we used the Web Service to make possible that a favorite alarm can be added, removed, or set defined time to trigger through the Web Service, and that favorite alarm triggers a message in the form of Post request. JSON format was returned with minimum information possible stating the status of that favorite alarm. To check this first part the testing was accomplished using browser Rest tools like in the first subsystem group testing. The second part was checked that the Email message is sent to the user that registered that favorite alarm during the time defined.

For the third subsystem group, we created the Email services using Gmail as SMTP client so our testing focused in check that Emails generated were received by the correct recipients with the defined template and attachments if the message was generated with them.

## System Tests

|  |  |
| --- | --- |
| Test Case ID: | PPM-001-1 Display Hours of Operation |
| Purpose: | To validate that they system displays the hours of operation specified by the client |
| Preconditions: | The user has downloaded the app |
| Inputs | 1. User clicks on ‘More ‘ button 2. User presses the ‘Hours of Operation’ button |
| Expected Output: | The system retrieves the information and displays the hours of operation to the user |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-001-2 Display Hours of Operation |
| Purpose: | To validate that they system displays the hours of operation specified by the client |
| Preconditions: | The user has downloaded the app |
| Inputs | 1. User clicks on ‘More ‘ button 2. User select another button 3. User clicks back to ‘More’ menu 4. User presses the ‘Hours of Operation’ button |
| Expected Output: | The system retrieves the information and displays the hours of operation to the user |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-001-3 Display Hours of Operation |
| Purpose: | To validate that they system does not display hours of operation if database is not populated |
| Preconditions: | The user has downloaded the app |
| Inputs | 1. User clicks on ‘More ‘ button 2. User select another button 3. User clicks back to ‘More’ menu 4. User presses the ‘Hours of Operation’ button |
| Expected Output: | The system does not show any information because no data is stored in database |
| Actual Output: | Hours of operation are still be shown |

|  |  |
| --- | --- |
| Test Case ID: | PPM-002-1 Trolley Alerts |
| Purpose: | To validate that they system allows the user to see any alerts pertaining to the trolleys |
| Preconditions: | * The user has downloaded the app * The admin has typed in one alert |
| Inputs | 1. User clicks on ‘Trolley Alerts ‘ button |
| Expected Output: | The system retrieves the information from the database and now displays information to the user |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-002-2 Trolley Alerts |
| Purpose: | To validate that they system allows the user to see any alerts pertaining to the trolleys |
| Preconditions: | * The user has downloaded the app * The admin has typed in multiple alerts |
| Inputs | 1. User clicks on ‘Trolley Alerts ‘ button |
| Expected Output: | The system retrieves the information from the database and now displays information to the user |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-002-3 Trolley Alerts |
| Purpose: | To validate that the system does not display any information if admin has removed data |
| Preconditions: | * The user has downloaded the app * The admin has typed in multiple alerts |
| Inputs | 1. User clicks on ‘Trolley Alerts ‘ button |
| Expected Output: | They system displays blank page and an alert message says there are no alerts |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-003-1 Show Estimated Time to Favorite Stop |
| Purpose: | To validate that they system calculates the estimated time from trolley location to set favorite stop |
| Preconditions: | * The user has selected the favorite stop * The trolley is being operated |
| Inputs | 1. User opens up app 2. User clicks on ‘Favorites’ tab |
| Expected Output: | The system retrieves the location of the favorite stop from the database, gets the location of the trolley from the TSO API, calculates the time and displays to user |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-003-2 Show Estimated Time to Favorite Stop |
| Purpose: | To validate that they system shows message if trolley has passed favorite stop on the route |
| Preconditions: | * The user has selected the favorite stop * The trolley is being operated |
| Inputs | 1. User opens up app 2. User clicks on ‘Favorites’ tab |
| Expected Output: | The system retrieves the location of the favorite stop from the database, gets the location of the trolley from the TSO API, calculates the time and displays to use a ‘N/A’ message |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-003-3 Show Estimated Time to Favorite Stop |
| Purpose: | To validate that the system does not calculate time if trolley is not on the route for that favorite stop |
| Preconditions: | * The user has selected the favorite stop * The trolley is being operated |
| Inputs | 1. User opens up app 2. User clicks on ‘Favorites’ tab |
| Expected Output: | The system acknowledges that stop is not on current route and displays a notification message |
| Actual Output: | The favorite stops continue to show estimated time even when the trolley is not on the current route |

|  |  |
| --- | --- |
| Test Case ID: | PPM-004-1 Set Favorite Stop |
| Purpose: | To validate that they system allows the user to save a favorite stop |
| Preconditions: | * The user has signed in to the app |
| Inputs | 1. User opens up app 2. User clicks on ‘Routes’ tab 3. User selects a route 4. User selects a stop 5. User clicks on save stop |
| Expected Output: | The system retrieves the user’s token and stop information. The system then saves the information in the database |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-004-2 Set Favorite Stop |
| Purpose: | To validate that they system allows the user to save multiple favorite stops |
| Preconditions: | * The user has signed in to the app * The user already has a stop saved in the database |
| Inputs | 1. User opens up app 2. User clicks on ‘Routes’ tab 3. User selects a route 4. User selects a stop 5. User clicks on save stop |
| Expected Output: | The system retrieves the user’s token and stop information. The system then saves the information in the database for the second favorite stop |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-004-3 Set Favorite Stop |
| Purpose: | To validate that they system does not save information if no token is available |
| Preconditions: | * The user is not signed in to the app |
| Inputs | 1. User opens up app 2. User clicks on ‘Routes’ tab 3. User selects a route 4. User selects a stop 5. User clicks on save stop |
| Expected Output: | The system displays a notification if the user is not signed in |
| Actual Output: | System states for function to work user must be signed in |

|  |  |
| --- | --- |
| Test Case ID: | PPM-005-1 Remove Favorite Stop |
| Purpose: | To validate that they system allows the user to remove a favorite stop |
| Preconditions: | * The user has signed in to the app * User has set favorite stop |
| Inputs | 1. User opens up app 2. User clicks on ‘Favorites’ tab 3. User selects a favorite stop 4. User selects Delete button and confirms it |
| Expected Output: | The system retrieves the user’s token and stop information and removes information from the database. System refreshes view |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-005-2 Remove Favorite Stop |
| Purpose: | To validate that they system allows the user to remove a favorite stop |
| Preconditions: | * The user has signed in to the app * User has more than one favorite stop |
| Inputs | 1. User opens up app 2. User clicks on ‘Favorites’ tab 3. User selects a favorite stop 4. User selects Delete button and confirms it. Does it twice |
| Expected Output: | The system retrieves the user’s token and stop information and removes information from the database. System refreshes view |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-005-3 Remove Favorite Stop |
| Purpose: | To validate that the system does not remove stop if user does not confirm |
| Preconditions: | * The user has signed in to the app * User has saved favorite stop |
| Inputs | 1. User opens up app 2. User clicks on ‘Favorites’ tab 3. User selects a favorite stop 4. User selects Delete button but does not confirm it. |
| Expected Output: | The system acknowledges user did not confirm and keeps information stored |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-006-1 List Stops |
| Purpose: | To validate that the system displays all the stops for a given route |
| Preconditions: | * The user has opened the application |
| Inputs | 1. User clicks on ‘Routes’ tab 2. User selects a route |
| Expected Output: | The system retrieves the stops from that given route from the database and displays to user as a list |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-006-2 List Stops |
| Purpose: | To validate that the system displays all the stops for a given route |
| Preconditions: | * The user has opened the application |
| Inputs | 1. User clicks on ‘Routes’ tab 2. User selects a route 3. User clicks back and selects another route |
| Expected Output: | The system retrieves the stops from that given route from the database and displays to user as a list |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-006-3 List Stops |
| Purpose: | To validate that the system hides the list route when user clicks back button |
| Preconditions: | * The user has opened the application |
| Inputs | 1. User clicks on ‘Routes’ tab 2. User selects a route 3. User clicks back |
| Expected Output: | The system acknowledges view change and hides route lst |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-007-1 Map Routes |
| Purpose: | To validate that the system displays all of the routes for the trolley |
| Preconditions: | * The user has opened the application |
| Inputs |  |
| Expected Output: | The system retrieves the waypoints from the database for each route and displays in color coated fashion representing two different routes |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-007-2 Map Routes |
| Purpose: | To validate that the system displays all of the routes for the trolley |
| Preconditions: | * The user has opened the application |
| Inputs | 1. User clicks on ‘More’ tab 2. User clicks on ‘Map’ tab |
| Expected Output: | The system retrieves the waypoints from the database for each route and displays in color coated fashion representing two different routes |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-007-3 Map Routes |
| Purpose: | To validate that the system does not display route if no information is stored in database |
| Preconditions: | * The user has opened the application |
| Inputs |  |
| Expected Output: | The system only displays route that information is present in database |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-008-1 Show Estimated Time to Stop |
| Purpose: | To validate that the system displays estimated time for a clicked stop |
| Preconditions: | * The user has opened the application |
| Inputs | 1. User has clicked on a stop |
| Expected Output: | The system retrieves the stop information and the trolley location. System calculates and displays estimated time |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-008-2 Show Estimated Time to Stop |
| Purpose: | To validate that the system displays notification that trolley has passed particular stop |
| Preconditions: | * The user has opened the application |
| Inputs | 1. User has clicked on a stop |
| Expected Output: | The system retrieves the stop information and the trolley location. System calculates and displays a ‘N/A’ for estimated time |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-008-3 Show Estimated Time to Stop |
| Purpose: | To validate that the system displays a notification for estimated time for the stop not on route |
| Preconditions: | * The user has opened the application |
| Inputs | 1. User has clicked on a stop |
| Expected Output: | The system retrieves the stop information and the trolley location. System sends message indicating stop is not on current route |
| Actual Output: | System calculates distance from trolley location |

|  |  |
| --- | --- |
| Test Case ID: | PPM-009-1 Find Nearest Stop |
| Purpose: | To validate that the system retrieves the user’s location and returns the nearest stop given the route. |
| Preconditions: | Requires the GPS setting enabled on the user’s device or accept localization when using a desktop web browser |
| Inputs | 1. User clicks on Map view 2. User presses the ‘Near Me’ button 3. User selects route: **Palmetto Middle North Route** |
| Expected Output: | System finds current location of user and returns: 9398 Ludlam Road stop as closest stop |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-009–2 Find Nearest Stop |
| Purpose: | To validate that the system retrieves the user’s location and returns the nearest stop given the route. |
| Preconditions: | Requires the GPS setting enabled on the user’s device or accept localization when using a desktop web browser |
| Inputs | 1. User clicks on Map view 2. User presses the ‘Near Me’ button 3. User selects route: **Palmetto Middle North Route** |
| Expected Output: | System finds current location of user and returns: 9800 SW 73rd Ct as closest stop |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-009–3 Find Nearest Stop |
| Purpose: | To validate that the system does not show closest stop when the user doesn’t select a route |
| Preconditions: | Requires the GPS setting enabled on the user’s device or accept localization when using a desktop web browser |
| Inputs | 1. User clicks on Map view 2. User presses the ‘Near Me’ button 3. User presses outside the popup menu |
| Expected Output: | System returns to the map view with no new marker indicating closest stop. |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-010-1 Arrived to Location Alert |
| Purpose: | To validate that the system sends an email to the user the trolley has arrived to their favorite stop |
| Preconditions: | The user has set a alert time notification for a favorite stop |
| Inputs |  |
| Expected Output: | System retrieves trolley and favorite stop information. System sends the user an email indicating trolley has arrived |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-010-2 Arrived to Location Alert |
| Purpose: | To validate that the system sends an email to the user the trolley has arrived to their multiple favorite stop |
| Preconditions: | The user has set a alert time notification for a favorite stop |
| Inputs |  |
| Expected Output: | System retrieves trolley and favorite stop information. System sends the user an email indicating trolley has arrived |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-010-3 Arrived to Location Alert |
| Purpose: | To validate that the system does not send an email to the user the trolley has arrived to their favorite stop if not alert was select |
| Preconditions: | The user has saved a favorite stop |
| Inputs |  |
| Expected Output: | System does not send email |
| Actual Output: | System sent the email |

|  |  |
| --- | --- |
| Test Case ID: | PPM-011-1 Login |
| Purpose: | To validate that the system allows the user to login with correct credentials |
| Preconditions: | User has already registered an account |
| Inputs | 1. User clicks on ‘More’ tab 2. User selects ‘Login’ button 3. User types in username: [mauricepruna@gmail.com](mailto:mauricepruna@gmail.com) 4. User types in password: Cuba1234 |
| Expected Output: | System acknowledges user credentials and logins with favorites information |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-011-2 Login |
| Purpose: | To validate that the system allows the user to login with correct credentials |
| Preconditions: | User has already registered an account |
| Inputs | 1. User clicks on ‘More’ tab 2. User selects ‘Login’ button 3. User types in username: [rmart071@fiu.edu](mailto:mauricepruna@gmail.com) 4. User types in password: Admin1234 |
| Expected Output: | System acknowledges user credentials and logins with favorites information |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-011-3 Login |
| Purpose: | To validate that the system does not allows the user to login with incorrect credentials |
| Preconditions: | User has already registered an account |
| Inputs | 1. User clicks on ‘More’ tab 2. User selects ‘Login’ button 3. User types in username: masoud@fiu.edu 4. User types in password: SeniorProject |
| Expected Output: | System acknowledges user credentials are incorrect, displays message, and does not let user sign in |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-012-1 Register User |
| Purpose: | To validate that the system stores user information when user registers with the application |
| Preconditions: | User has not created an account |
| Inputs | 1. User clicks on ‘ More’ tab 2. User clicks on ‘Register’ button 3. User types in user name: gabrielawilson@gmail.com 4. User types in password: lovepinecrest |
| Expected Output: | System stores credentials in database and system logins in user |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-012-2 Register User |
| Purpose: | To validate that the system stores user information when user registers with the application |
| Preconditions: | User has not created an account |
| Inputs | 1. User clicks on ‘ More’ tab 2. User clicks on ‘Register’ button 3. User types in user name: pepe@gmail.com 4. User types in password: billete |
| Expected Output: | System stores credentials in database and system logins in user |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-012-3 Register User |
| Purpose: | To validate that the system does not let user register if account is already preset |
| Preconditions: |  |
| Inputs | 1. User clicks on ‘ More’ tab 2. User clicks on ‘Register’ button 3. User types in user name: [mauricepruna@gmail.com](mailto:mauricepruna@gmail.com) 4. User types in password: Cuba1234 |
| Expected Output: | System shows notification and does not save information |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-013-1 Request a stop |
| Purpose: | To validate that the system allows the user to send an email to request for a new stop |
| Preconditions: | User has opened app |
| Inputs | 1. User clicks on ‘More’ tab 2. User clicks on ‘Request a Stop’ button 3. User Enters name: Maurice 4. User enters email: [mauricepruna@gmail.com](mailto:mauricepruna@gmail.com) 5. User enters location: 1111 SW 111 St 6. User clicks send |
| Expected Output: | Email is sent to system admin with text information |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-013-2 Request a stop |
| Purpose: | To validate that the system allows the user to send an email to request for a new stop |
| Preconditions: | User has opened app |
| Inputs | 1. User clicks on ‘More’ tab 2. User clicks on ‘Request a Stop’ button 3. User enters name: Ricky 4. User enters email: [rmart071@fiu.edu](mailto:rmart071@fiu.edu) 5. User enters location: 9400 SW 63 Ct 6. User clicks send |
| Expected Output: | Email is sent to system admin with text information |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-013-3 Request a stop |
| Purpose: | To validate that the system recognizes that email input is correct format |
| Preconditions: | User has opened app |
| Inputs | 1. User clicks on ‘More’ tab 2. User clicks on ‘Request a Stop’ button 3. User enters Name: test 4. User enters email: [select \* from users](mailto:rmart071@fiu.edu) 5. User enters location: 1111 SW 111 St 6. User clicks send |
| Expected Output: | System acknowledges email is incorrect format and does not send email |
| Actual Output: |  |

|  |  |
| --- | --- |
| Test Case ID: | PPM-014-1 Report a Problem |
| Purpose: | To validate that the system allows the user to send an email to report a problem |
| Preconditions: | User has opened app |
| Inputs | 1. User clicks on ‘More’ tab 2. User clicks on ‘Report a Problem’ button 3. User enters their email: [mauricepruna@gmail.com](mailto:mauricepruna@gmail.com) 4. User enters problem: tree is in the middle of the road 5. User clicks send |
| Expected Output: | Email is sent with problem in body |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-014-2 Report a Problem |
| Purpose: | To validate that the system allows the user to send an email to report a problem |
| Preconditions: | User has opened app |
| Inputs | 1. User clicks on ‘More’ tab 2. User clicks on ‘Report a Problem’ button 3. User enters their email: [mauricepruna@gmail.com](mailto:mauricepruna@gmail.com) 4. User takes picture 5. User enters problem: tree is in the middle of the road 6. User clicks send |
| Expected Output: | Email is sent with problem in body along with photo as attachement |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-014-3 Report a Problem |
| Purpose: | To validate that the system does not allow the user to send an email to report a problem if all blank |
| Preconditions: | User has opened app |
| Inputs | 1. User clicks on ‘More’ tab 2. User clicks on ‘Report a Problem’ button 3. User leaves information blank 4. User clicks send |
| Expected Output: | System acknowledges fields are blank and does not send email |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-015-1 Send Feedback |
| Purpose: | To validate that the system allows the user to send an email to send feedback |
| Preconditions: | User has opened app |
| Inputs | 1. User clicks on ‘More’ tab 2. User clicks on ‘Send Feedback’ button 3. User types in name: Maurice 4. User types in email: [mauricepruna@gmail.com](mailto:mauricepruna@gmail.com) 5. User types in: This app is the best!!!! 6. User clicks send |
| Expected Output: | System sends email to system admin with text added by the user the body |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-015-2 Send Feedback |
| Purpose: | To validate that the system allows the user to send an email to send feedback |
| Preconditions: | User has opened app |
| Inputs | 1. User clicks on ‘More’ tab 2. User clicks on ‘Send Feedback’ button 3. User enters name: Ricky 4. User types in email: rmart071@fiu.edu 5. User types in: This app has a lot of meat! 6. User clicks send |
| Expected Output: | System sends email to system admin with text added by the user the body |
| Actual Output: | Same as expected |

|  |  |
| --- | --- |
| Test Case ID: | PPM-015-3 Send Feedback |
| Purpose: | To validate that the system does not allow the user to send an email to send feedback if incorrect email format |
| Preconditions: | User has opened app |
| Inputs | 1. User clicks on ‘More’ tab 2. User clicks on ‘Enter Feedback’ button 3. User types in username: Ricky 4. User enters email: Ricky 5. User types in: This app has a lot of meat! 6. User clicks send |
| Expected Output: | System acknowledges incorrect format and does not send email |
| Actual Output: | Same as expected |

## Evaluation of Tests

### Subsystem Test Results

|  |  |  |
| --- | --- | --- |
| Test Case ID | Description | Result |
|  | Fetch Known Key | PASSED |
| M011 | Fetch Unknown Key | PASSED |
| M012 | ValidCMSUpdate | PASSED |
| M013 | InValidCMSUpdate | PASSED |
| M014 | SendValidEmail | PASSED |
| M015 | DontSendInvalidEmail | PASSED |

### System Test Results – Sunny Day

|  |  |
| --- | --- |
| Test Case ID | Test Result |
| PPM-001-1 | PASSED |
| PPM-001-2 | PASSED |
| PPM-002-1 | PASSED |
| PPM-002-2 | PASSED |
| PPM-003-1 | PASSED |
| PPM-003-2 | PASSED |
| PPM-004-1 | PASSED |
| PPM-004-2 | PASSED |
| PPM-005-1 | PASSED |
| PPM-005-2 | PASSED |
| PPM-006-1 | PASSED |
| PPM-006-2 | PASSED |
| PPM-007-1 | PASSED |
| PPM-007-2 | PASSED |
| PPM-008-1 | PASSED |
| PPM-008-2 | PASSED |
| PPM-009-1 | PASSED |
| PPM-009-2 | PASSED |
| PPM-010-1 | PASSED |
| PPM-010-2 | PASSED |
| PPM-011-1 | PASSED |
| PPM-011-2 | PASSED |
| PPM-012-1 | PASSED |
| PPM-012-2 | PASSED |
| PPM-013-1 | PASSED |
| PPM-013-2 | PASSED |
| PPM-014-1 | PASSED |
| PPM-014-2 | PASSED |
| PPM-015-1 | PASSED |
| PPM-015-2 | PASSED |

### System Test Results - Rainy Day

|  |  |
| --- | --- |
| Test Case ID | Test Result |
| PPM-001-3 | FAILED |
| PPM-002-3 | PASSED |
| PPM-003-3 | FAILED |
| PPM-004-3 | PASSED |
| PPM-005-3 | PASSED |
| PPM-006-3 | PASSED |
| PPM-007-3 | PASSED |
| PPM-008-3 | FAILED |
| PPM-009-3 | PASSED |
| PPM-010-3 | PASSED |
| PPM-011-3 | PASSED |
| PPM-012-3 | PASSED |
| PPM-013-3 | PASSED |
| PPM-014-3 | PASSED |
| PPM-015-3 | PASSED |

# Glossary

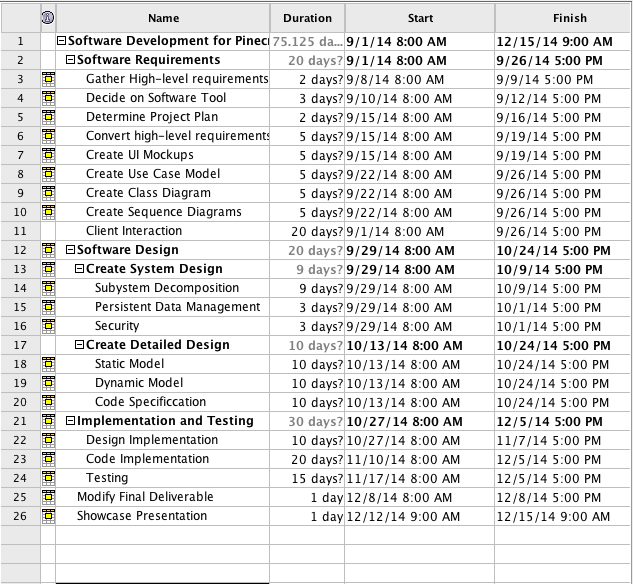
**Sencha Touch**: Sencha Touch, a high-performance HTML5 mobile application framework, is the cornerstone of the Sencha HTML5 platform. Built for enabling world-class user experiences, Sencha Touch is the only framework that enables developers to build powerful apps that work on iOS, Android, BlackBerry, Windows Phone, and more.

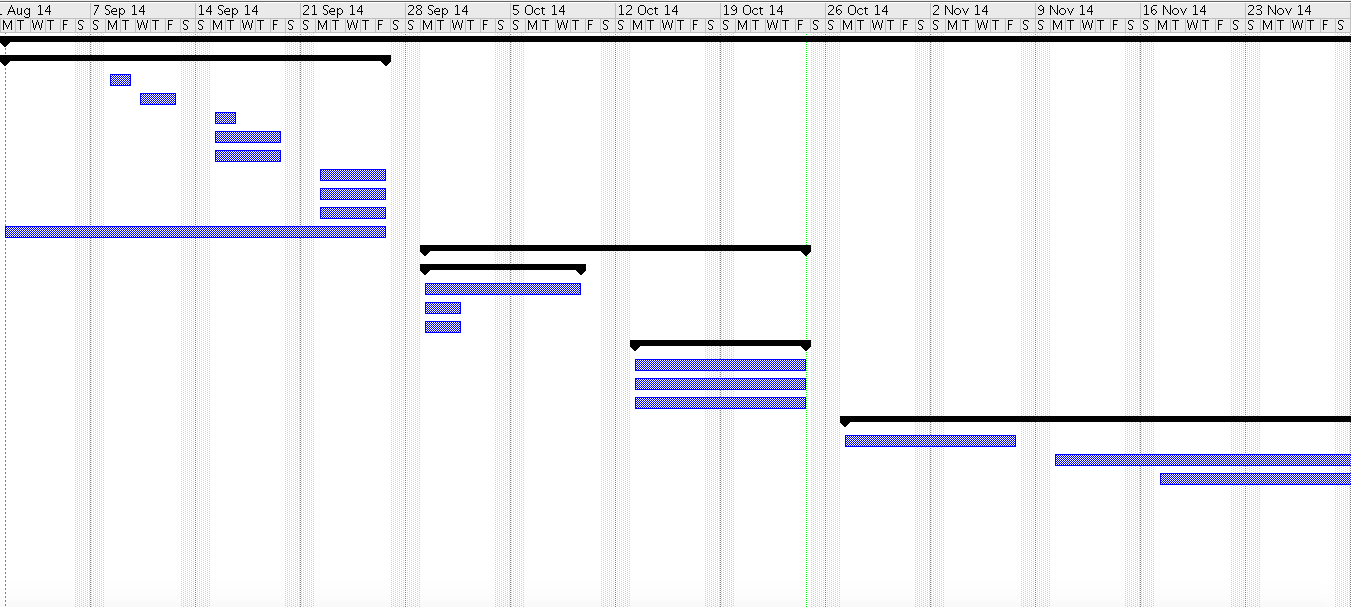
**Spring**: Spring helps development teams everywhere build simple, portable, fast and flexible JVM-based systems and applications

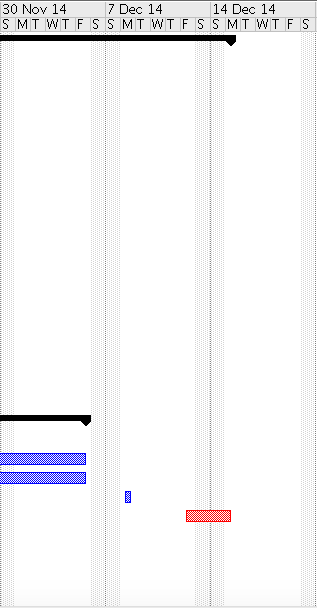
**Cordova**: free and open source framework that allows you to create mobile apps using standardized web APIs for the platforms you care about.

# Appendix

## Appendix A - Project schedule (Gantt chart or PERT chart)







## Appendix B – All use cases with nonfunctional requirements.

|  |  |
| --- | --- |
| Use Case ID | PPM-01 Display Hours of Operation |
| Description | Allow users to view the hours of operation specified by the client |
| Actor | User |
| Pre-conditions | 1. The hours of operations have been inputted into the database by the client 2. The user has downloaded and opened the app 3. The user has clicked the more button, and clicks on ‘Hours of Operations’ button |
| Main success scenario | 1. The system retrieves the information from the database 2. The system sends the information to the controller 3. The controller sends information to the view and displays information to user |
| Post-conditions | 1. The view now contains the hours of operations |
| Exceptions | If any error has occurred, the system will display a message indicating the error |

|  |  |
| --- | --- |
| Use Case ID | PPM-02 Trolley Alerts |
| Description | Allow user to receive alerts about any issues or problems in regards to the trolleys |
| Actor | User |
| Pre-conditions | 1. The user has opened the app |
| Main success scenario | 1. The system checks in database if any current issue is present 2. System shows alert box with alert message |
| Post-conditions | 1. The user can now make changes to plans |
| Exceptions |  |

|  |  |
| --- | --- |
| Use Case ID | PPM-03 Show Estimated Time to Favorite Stop |
| Description | The system updates the user with the estimated time of the user’s favorite stop |
| Actor | User |
| Pre-conditions | 1. The user has set a favorite stop in the application |
| Main success scenario | 1. The user clicks on the favorites stop tab 2. The system retrieves the location of the trolley through the API. 3. The system retrieves the user’s favorite stops information. 4. The system sends the information to the view 5. The user can now see the estimated time of arrival for the stop(s) |
| Post-conditions | The user can now plan accordingly |
| Exceptions |  |

|  |  |
| --- | --- |
| Use Case ID | PPM-04 Set Favorite Stop |
| Description | Allows user to define their favorite stop |
| Actor | User |
| Pre-conditions | 1. User is logged on the system |
| Main success scenario | 1. The user has chosen a route 2. The system displays all the stops for that given route 3. The user clicks the add to favorites button 4. The system asks for the timer notification for that stop 5. The user selects the timer option 6. System sends confirmation alert |
| Post-conditions | The user will now receive alerts for their selected stop(s) |
| Exceptions |  |

|  |  |
| --- | --- |
| Use Case ID | PPM-05 Remove Favorite Stop |
| Description | Allows user to remove their favorite stop |
| Actor | User |
| Pre-conditions | 1. User is logged on the system |
| Main success scenario | 1. The user has selected the favorite stops tab 2. The system retrieves the user’s favorites information 3. The system displays the stop(s) to the user in the view 4. The user selects the ‘Remove Stop’ button 5. System sends a an alert 6. User acknowledges actions 7. System removes stop from the database |
| Post-conditions | The stop will longer exist for the user’s favorite stop(s) list |
| Exceptions |  |

|  |  |
| --- | --- |
| Use Case ID | PPM-06 List Stops |
| Description | Allows user to remove their favorite stop |
| Actor | User |
| Pre-conditions | 1. User is logged on the system |
| Main success scenario | 1. The user has selected the route 2. The system retrieves the stops for that route in the database 3. The system displays all of the stops for that route |
| Post-conditions | 1. The user can now set stop(s) as favorites 2. The user can see the stops for the routes |
| Exceptions |  |

|  |  |
| --- | --- |
| Use Case ID | PPM-07 Map Routes |
| Description | Allows the user to see all of the routes for the trolley |
| Actor | User |
| Pre-conditions | 1. User has selected the map button |
| Main success scenario | 1. The system retrieves the routes information 2. The map displays color coated routes and stops |
| Post-conditions | The user can now see all of the routes and stops on a map |
| Exceptions |  |

|  |  |
| --- | --- |
| Use Case ID | PPM-08 Show Estimated Time to Stop |
| Description | The user can see the estimated time of arrival for each stop |
| Actor | User |
| Pre-conditions | 1. The user has selected a route |
| Main success scenario | 1. The system retrieves the stops for the given route 2. The user is looking at the list 3. The system displays the estimated time of arrival for each stop |
| Post-conditions | The user can now plan accordingly |
| Exceptions |  |

|  |  |
| --- | --- |
| Use Case ID | PPM-09 Find Nearest Stop |
| Description | Allows user to find the closest stop to their location in the Village of Pinecrest |
| Actor | User |
| Pre-conditions | 1. The user has opened the app |
| Main success scenario | 1. The user clicks on the ‘Near Me’ button 2. The system retrieves the user’s location 3. The system find the closest location using the API 4. The application shows the user where the closest stop is |
| Post-conditions | The user can now go to that stop |
| Exceptions | The user is now in the Village of Pinecrest |

|  |  |
| --- | --- |
| Use Case ID | PPM-10 Arrived to Location Alert |
| Description | The system sends an alert when the trolley has arrived to users favorite stop |
| Actor | User |
| Pre-conditions | 1. User is logged on the system 2. User has set the favorite stops with the alert timer |
| Main success scenario | 1. The system checks the database to see if the user has any stops chosen 2. The system retrieves the location of the trolley through the API 3. The system sends the user a notification that the trolley has arrived to their selected stop |
| Post-conditions | 1. The user is now notified of the arrival of the trolley to the favorite stop |
| Exceptions | The user is not logged on the system. |

|  |  |
| --- | --- |
| Use Case ID | PPM-11 Login |
| Description | The system allows the user to login to retrieve information |
| Actor | User |
| Pre-conditions | 1. User has opened the app 2. User selects the login option |
| Main success scenario | 1. The system displays the login panel 2. The user enters the username and password 3. The system retrieves all of the favorite stop information 4. The system retrieves all of the alerts information |
| Post-conditions | 1. The user can now receive alerts |
| Exceptions | The user types in wrong credentials |

|  |  |
| --- | --- |
| Use Case ID | PPM-12 Register User |
| Description | Allows user to Register on the system |
| Actor | User |
| Pre-conditions | 1. The user has opened the app 2. The user selects the register button |
| Main success scenario | 1. The system validates that the filed username is populated with an email. 2. The system validates that the password and confirm password entered are equal. 3. If validations are correct the system save the email address and password introduced. |
| Post-conditions | The system enabled options related to the management use cases |
| Exceptions |  |

|  |  |
| --- | --- |
| Use Case ID | PPM-13 Request a stop |
| Description | Allow user to send messages to the system. In this case to request a stop |
| Actor | User |
| Pre-conditions | 1. User is logged on the system to automatically capture sender email 2. User enters email to the system. |
| Main success scenario | 1. The system creates a message with a formatted subject with one of the stated topics 2. Send an email to the corresponding authority. |
| Post-conditions | 1. The email is sent to the corresponding authority |
| Exceptions | The user has typed in wrong email or incorrect email format |

|  |  |
| --- | --- |
| Use Case ID | PPM-14 Report a Problem |
| Description | Allow user to send messages to the system. In this case to report a problem |
| Actor | User |
| Pre-conditions | 1. User is logged on the system to automatically capture sender email 2. User enters email to the system. |
| Main success scenario | 1. The system creates a message with a formatted subject with one of the stated topics 2. Send an email to the corresponding authority. |
| Post-conditions | 1. The email is sent to the corresponding authority |
| Exceptions | The user has typed in wrong email or incorrect email format |

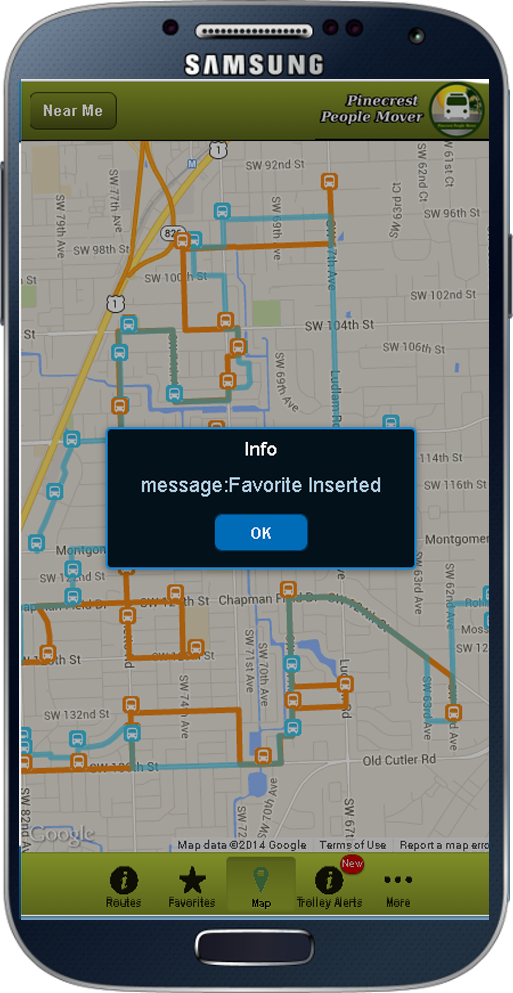
|  |  |
| --- | --- |
| Use Case ID | PPM-15 Send Feedback |
| Description | Allow user to send messages to the system. In this case to send feedback |
| Actor | User |
| Pre-conditions | 1. User is logged on the system to automatically capture sender email 2. User enters email to the system. |
| Main success scenario | 1. The system creates a message with a formatted subject with one of the stated topics 2. Send an email to the corresponding authority. |
| Post-conditions | 1. The email is sent to the corresponding authority |
| Exceptions | The user has typed in wrong email or incorrect email format |

## Appendix C – User Interface designs.

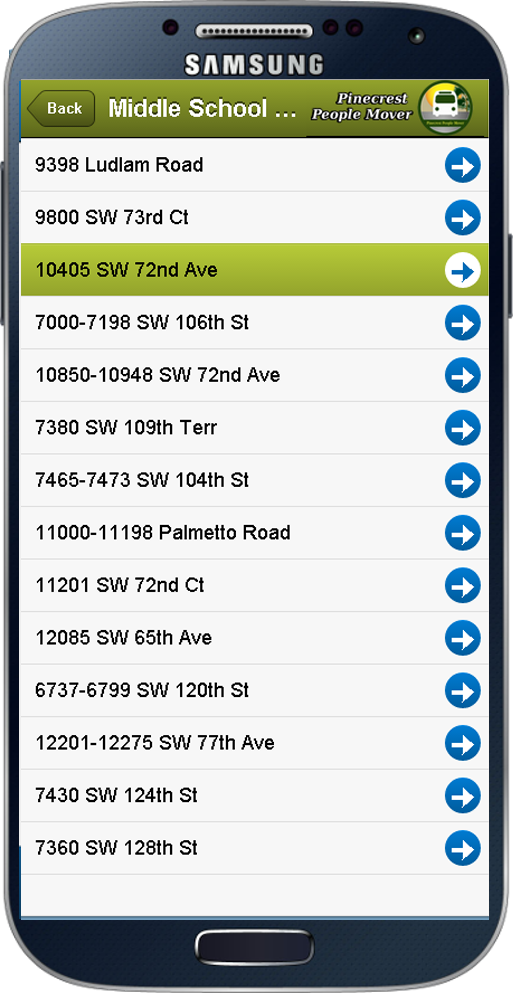
##### Figure E-1: Loading Image



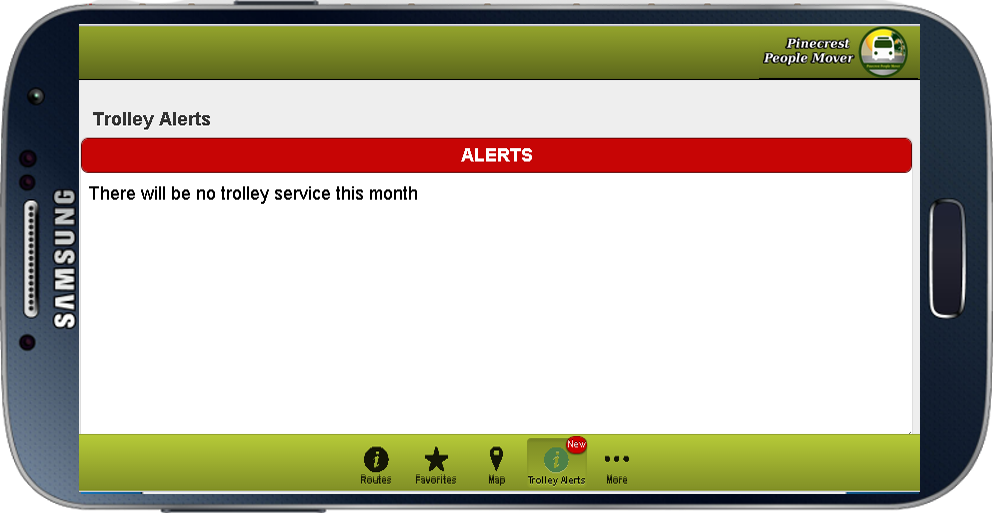
##### Figure E-2: Main Menu and Insert Favorite



##### Figure E-3: Stop List



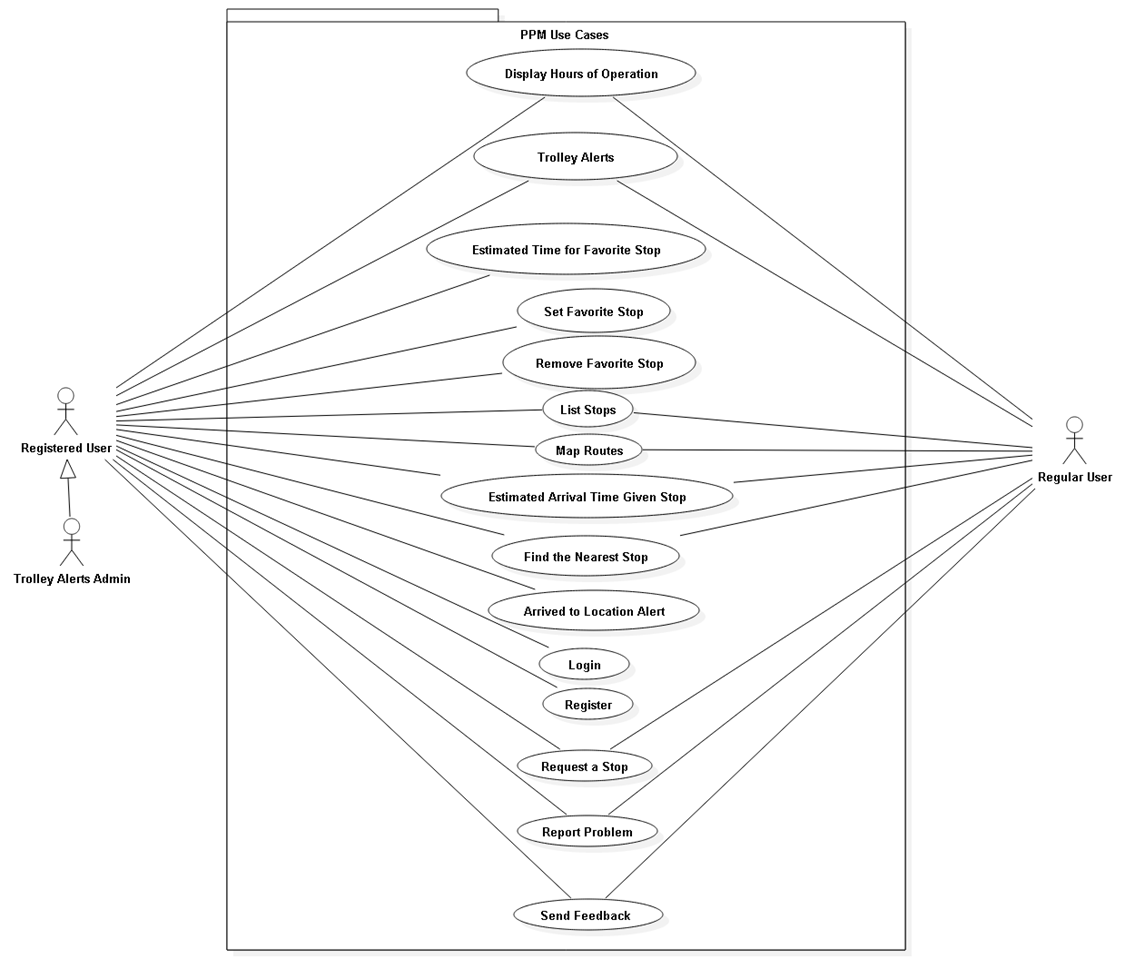
##### Figure E-4: Trolley Alerts



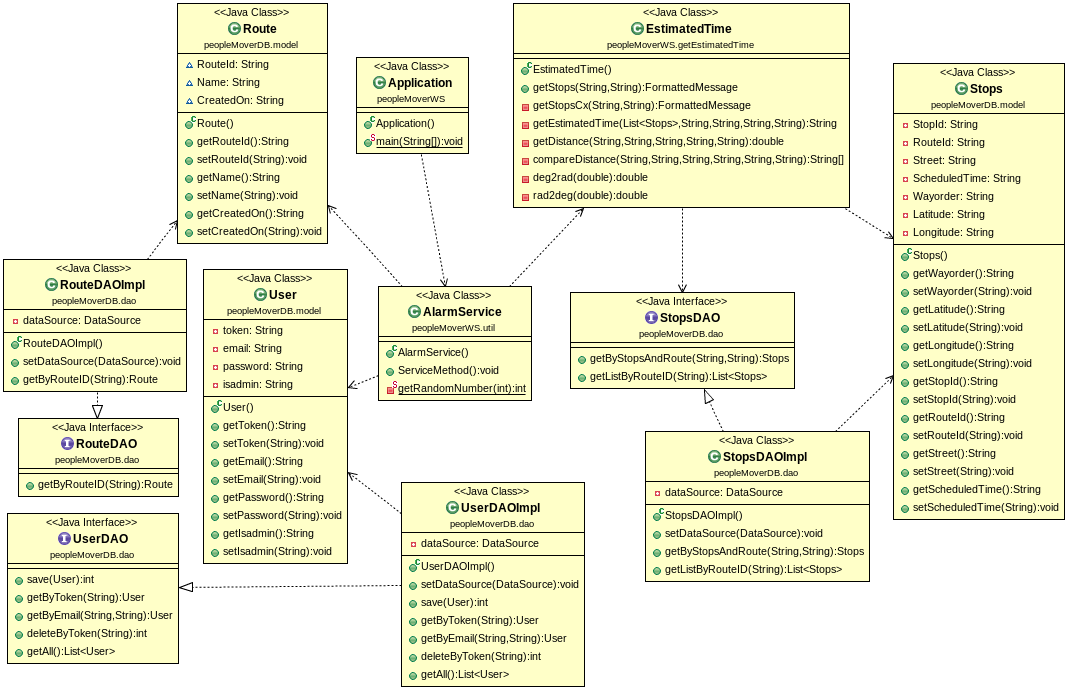
##### Figure E-5: Login View

##### C:\Users\maurice\OneDrive\Public\login.png

## Appendix D – Analysis models (static and dynamic)

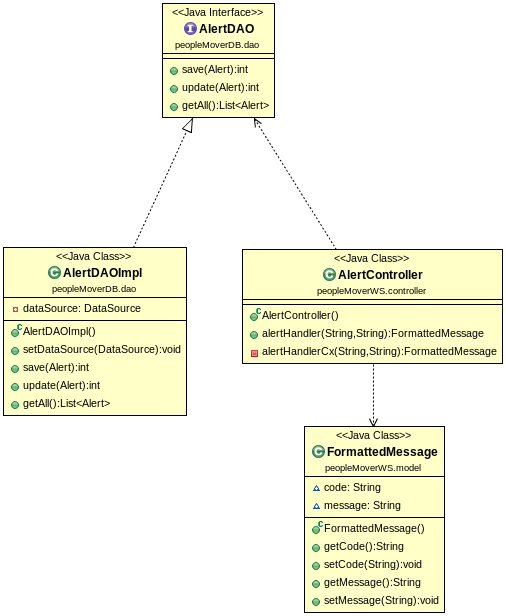


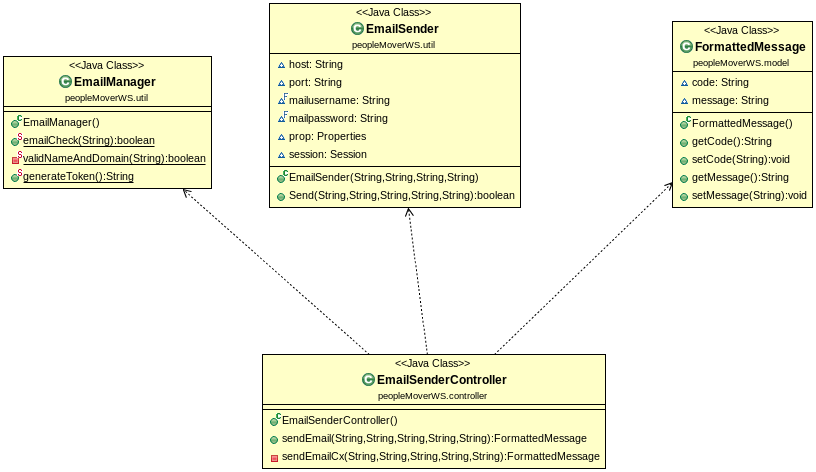
## Appendix E – Design models (static and dynamic)



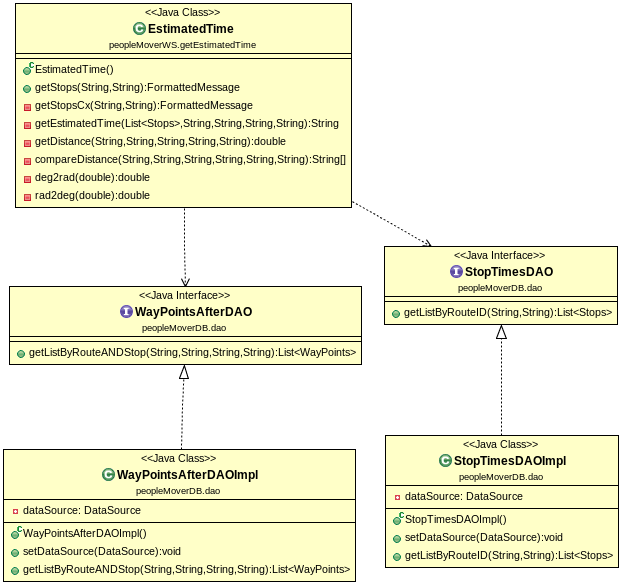
**Alarm Service**

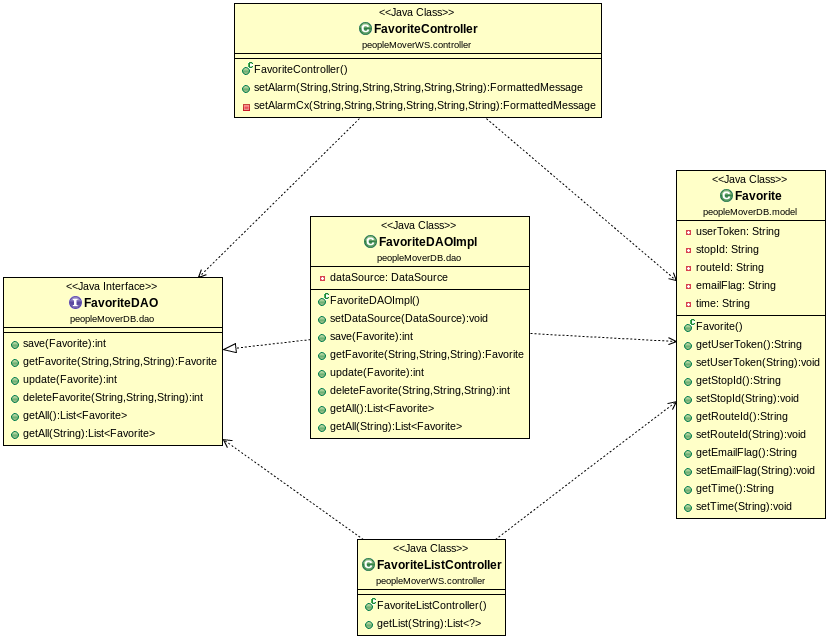
**Alert Subsystem**

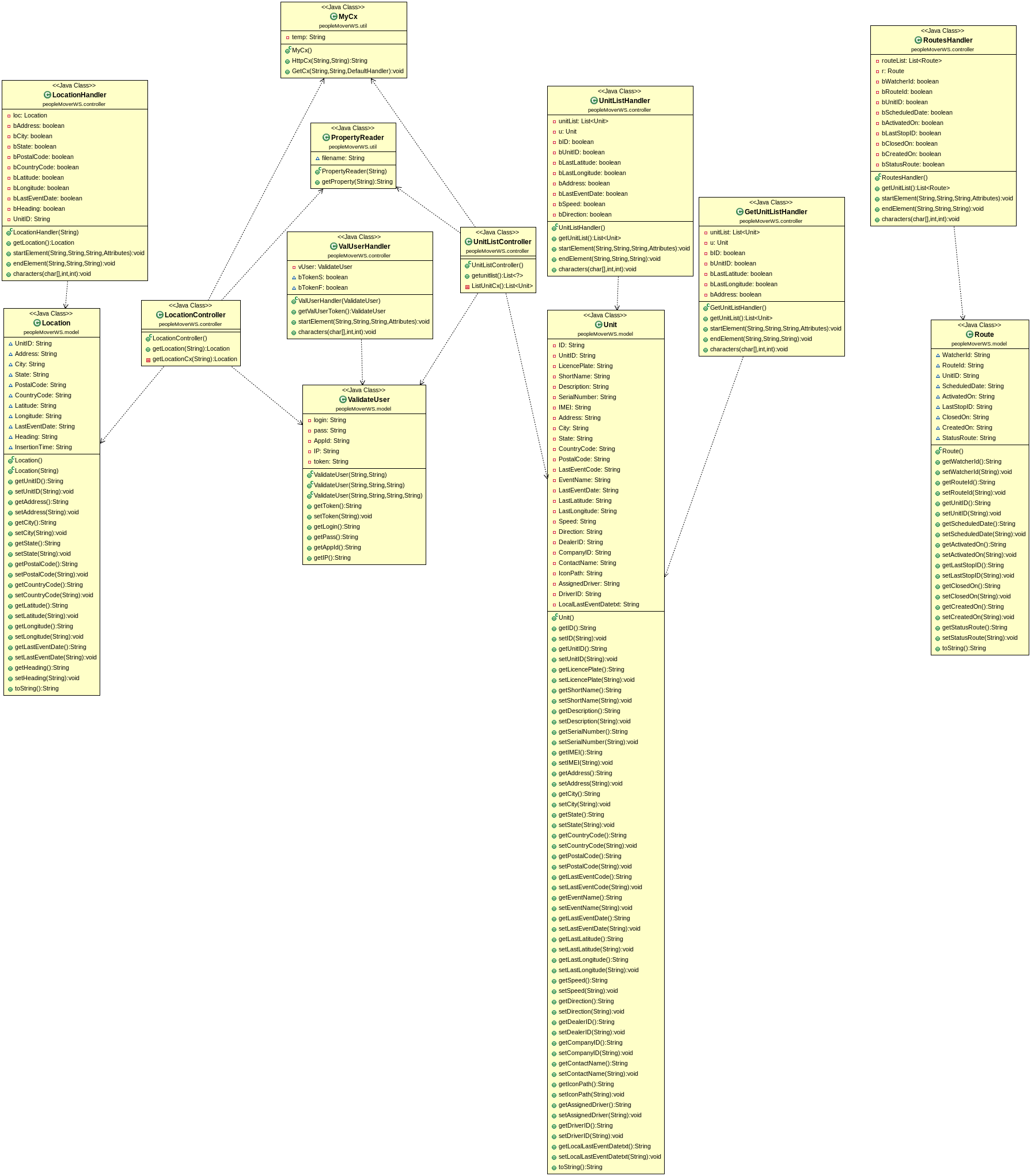


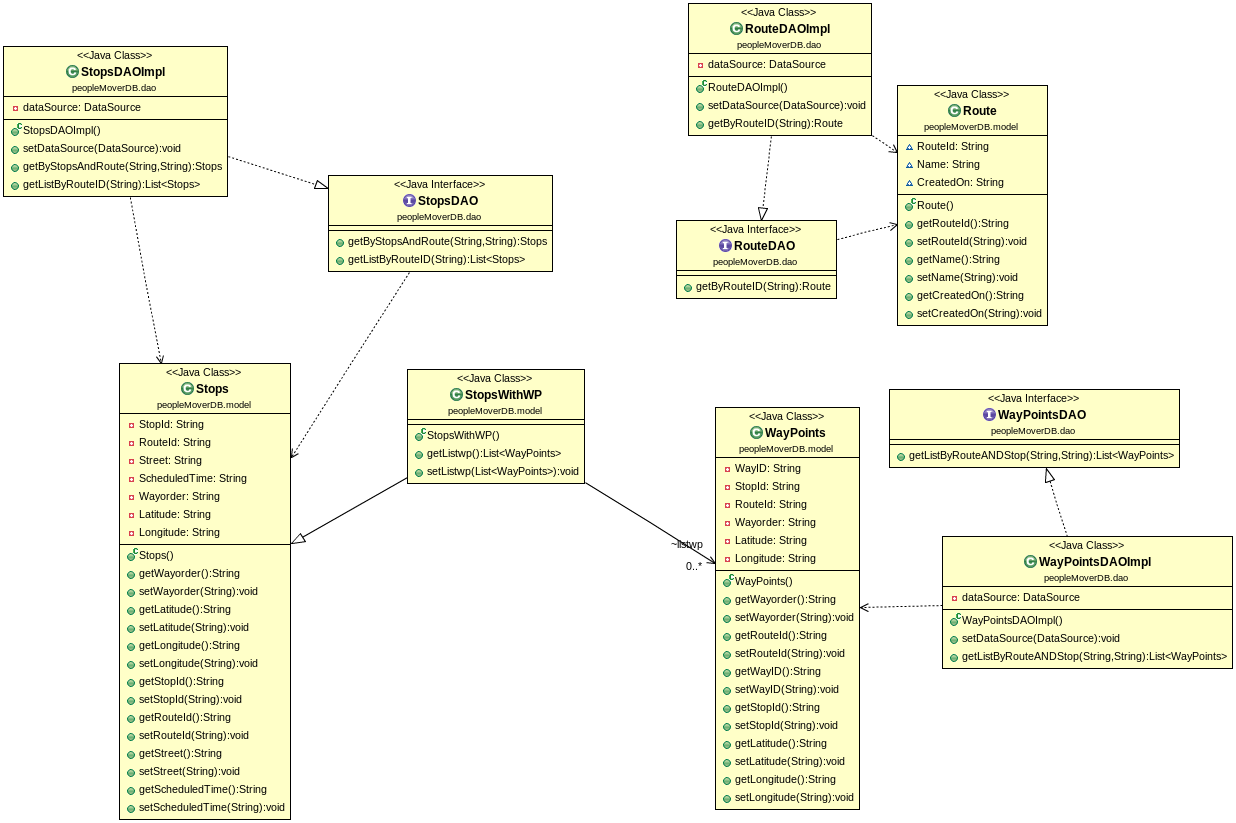
**Email Subsystem**

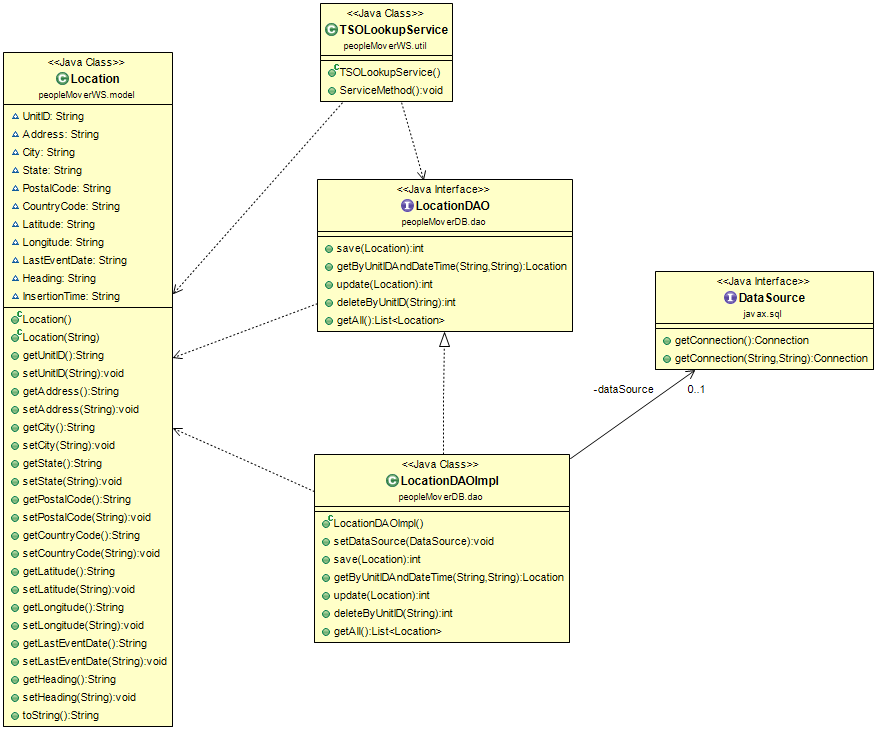
**Estimated Time**



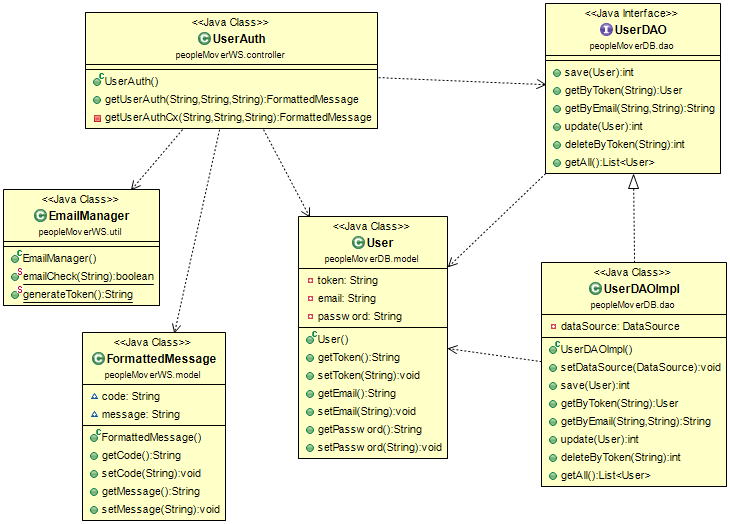
**Favorite Definition**

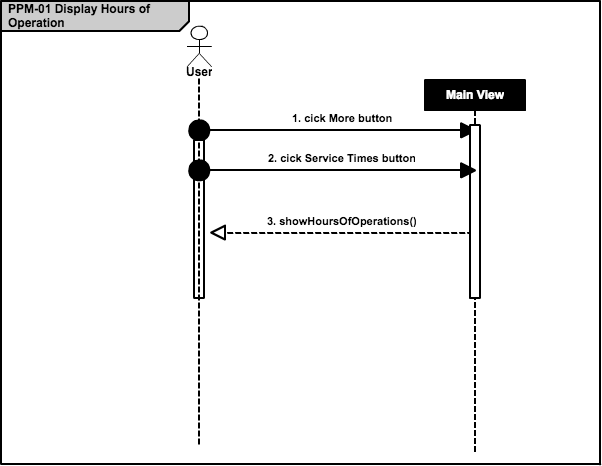
**Location, Unit and Route**

**Stops, Route, and Waypoints**

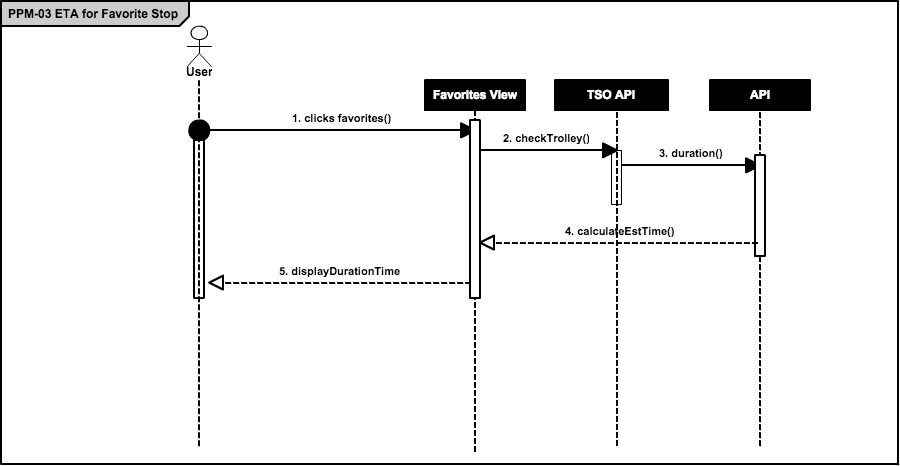
**TSOLookUpService (Historical Location)**

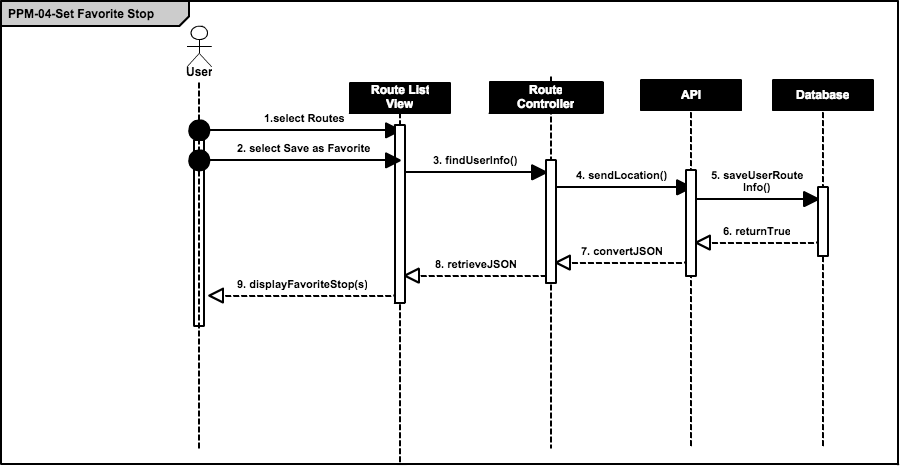
**Authentication Service**

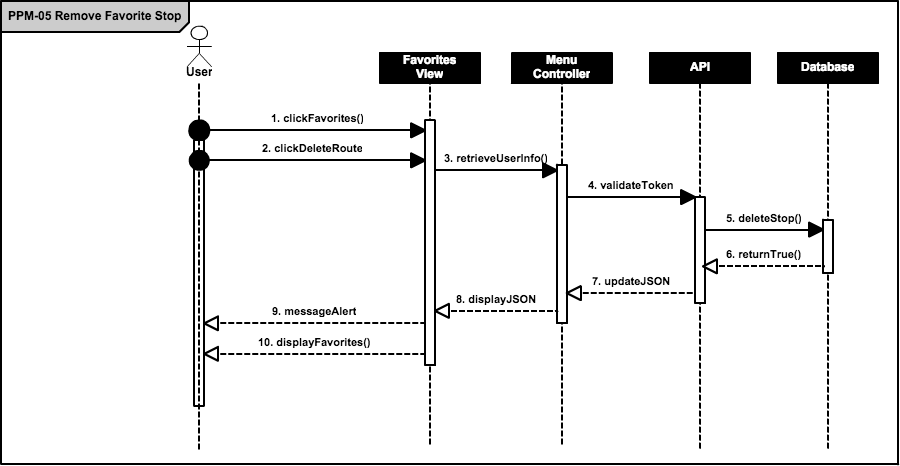


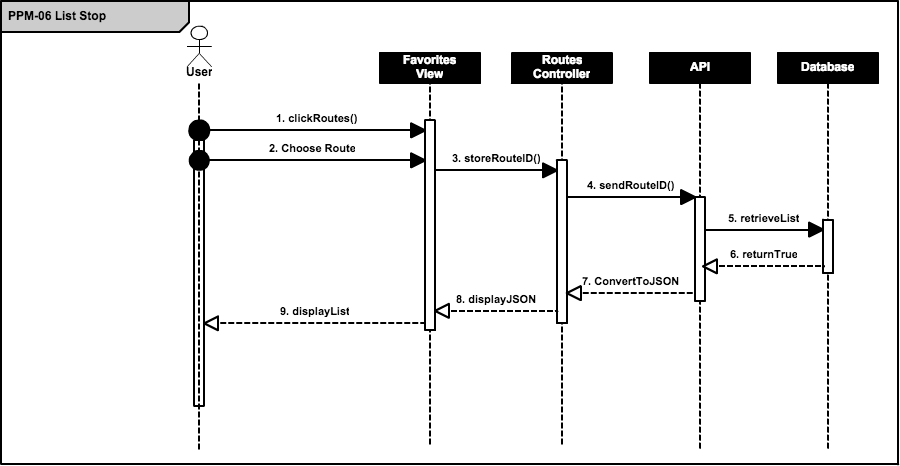


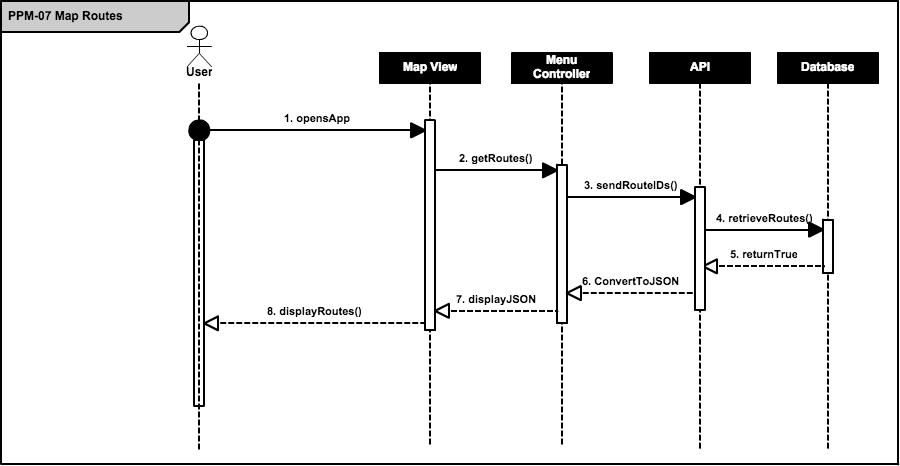


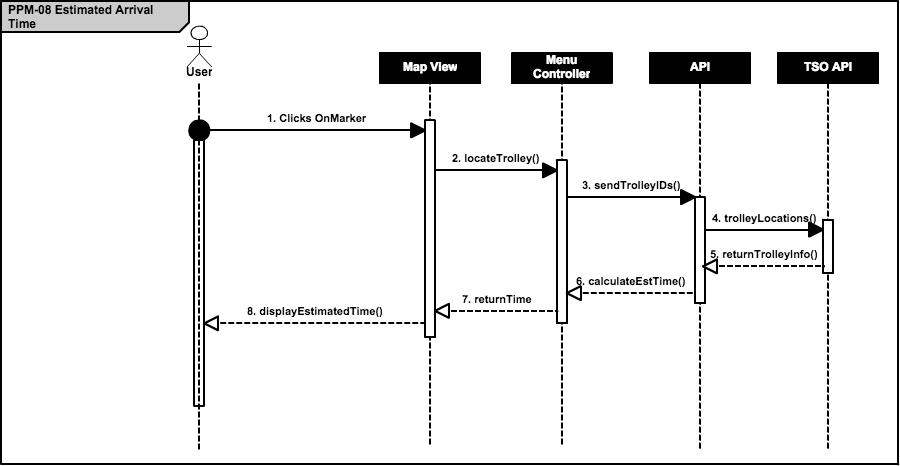


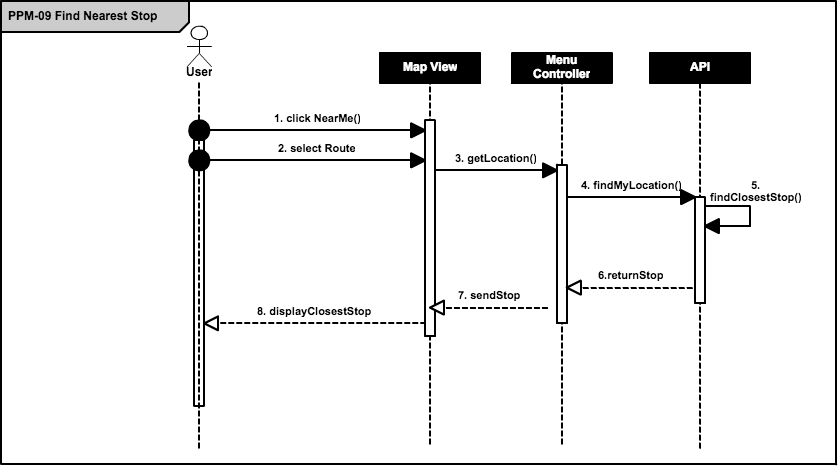


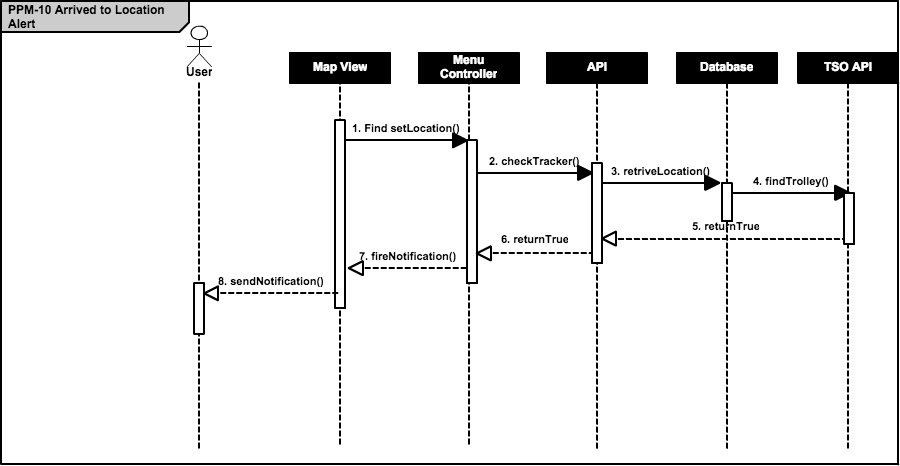


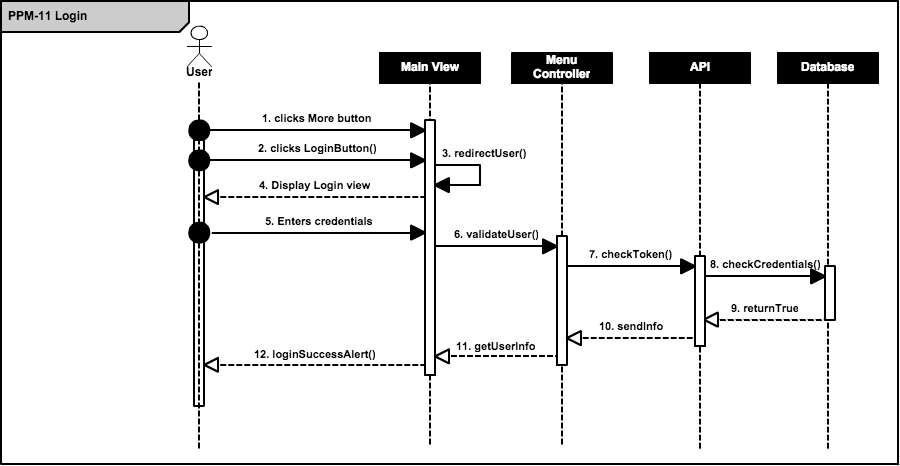


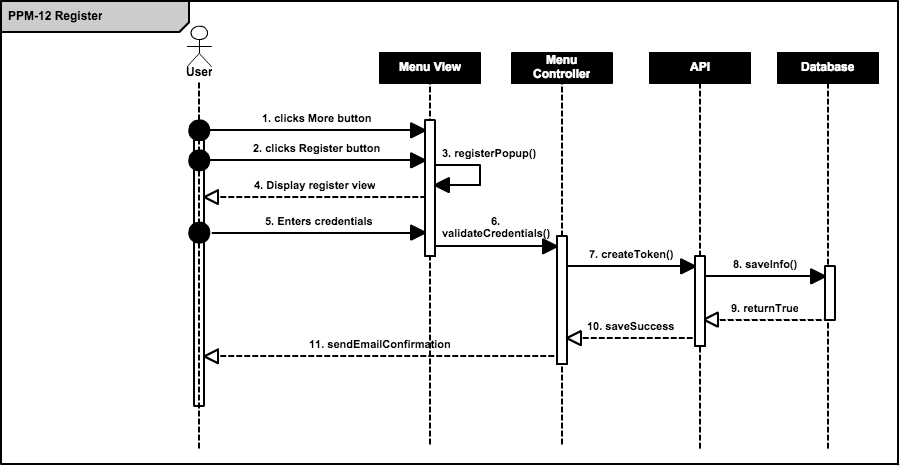


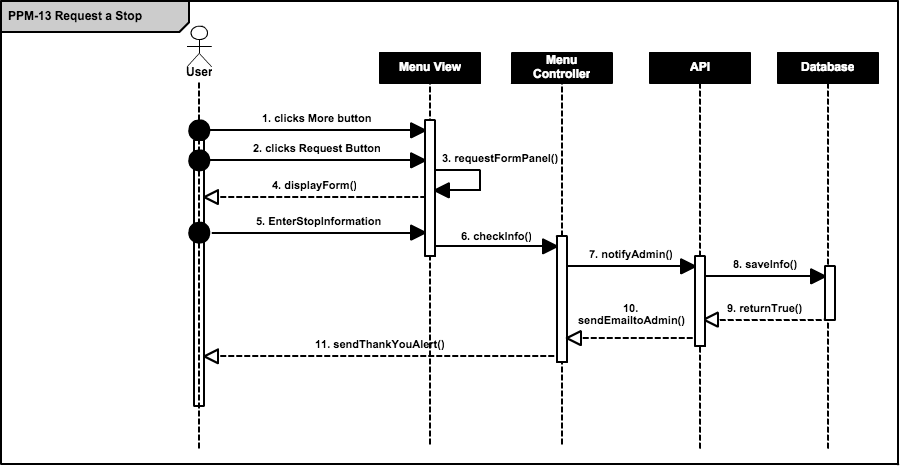


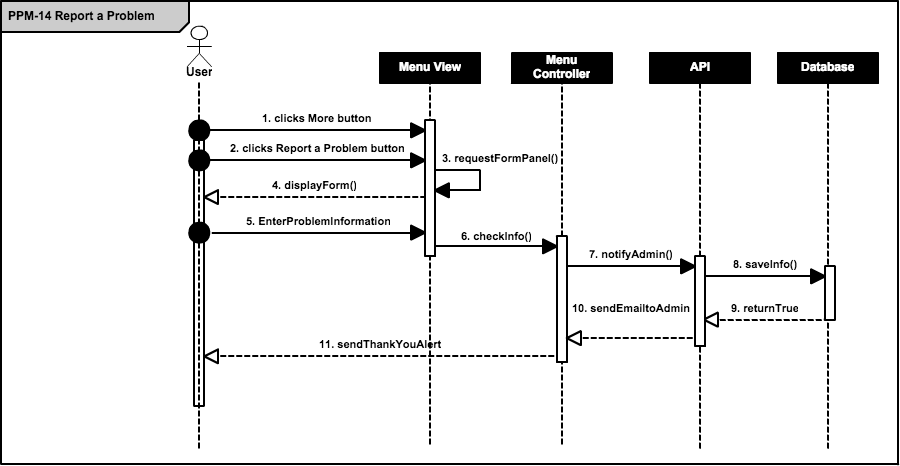


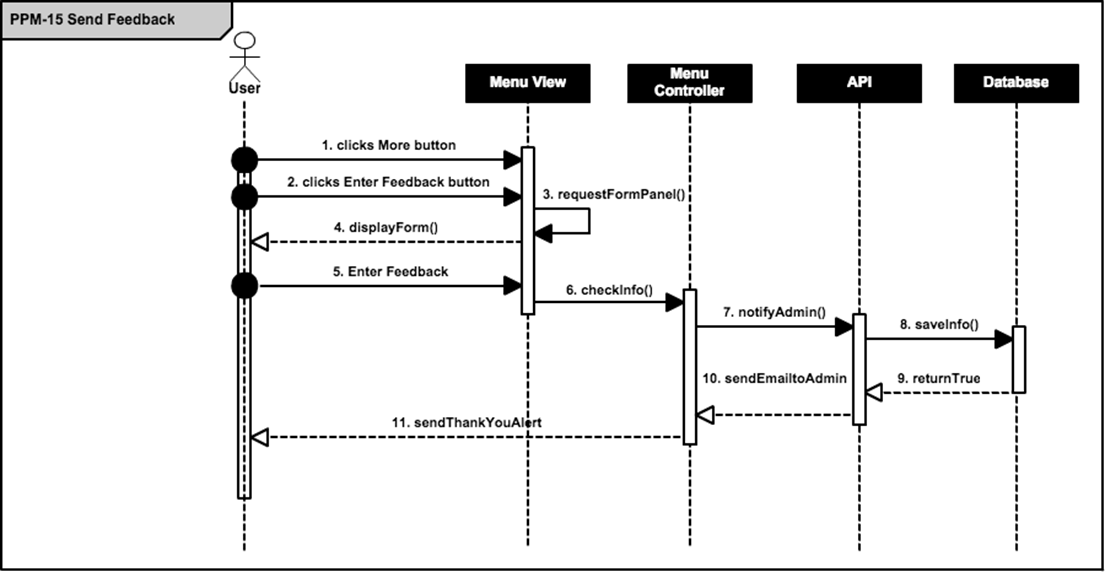












## Appendix F – Documented Class interfaces (code) and constraints.

Package Hierarchies:

* [peopleMoverDB.dao](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\package-tree.html),
* [peopleMoverDB.model](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\model\package-tree.html),
* [peopleMoverWS](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\package-tree.html),
* [peopleMoverWS.controller](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\package-tree.html),
* [peopleMoverWS.getEstimatedTime](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\getEstimatedTime\package-tree.html),
* [peopleMoverWS.model](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\model\package-tree.html),
* [peopleMoverWS.util](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\util\package-tree.html)

## Class Hierarchy

* java.lang.Object
  + peopleMoverDB.model.[Alert](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\model\Alert.html)
  + peopleMoverWS.controller.[AlertController](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\AlertController.html)
  + peopleMoverDB.dao.[AlertDAOImpl](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\AlertDAOImpl.html) (implements peopleMoverDB.dao.[AlertDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\AlertDAO.html))
  + peopleMoverWS.[Application](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\Application.html)
  + org.xml.sax.helpers.DefaultHandler (implements org.xml.sax.ContentHandler, org.xml.sax.DTDHandler, org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler)
    - peopleMoverWS.controller.[GetUnitListHandler](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\GetUnitListHandler.html)
    - peopleMoverWS.controller.[LocationHandler](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\LocationHandler.html)
    - peopleMoverWS.controller.[RoutesHandler](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\RoutesHandler.html)
    - peopleMoverWS.controller.[UnitListHandler](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\UnitListHandler.html)
    - peopleMoverWS.controller.[ValUserHandler](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\ValUserHandler.html)
  + peopleMoverWS.util.[EmailManager](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\util\EmailManager.html)
  + peopleMoverWS.util.[EmailSender](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\util\EmailSender.html)
  + peopleMoverWS.controller.[EmailSenderController](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\EmailSenderController.html)
  + peopleMoverWS.getEstimatedTime.[EstimatedTime](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\getEstimatedTime\EstimatedTime.html)
  + peopleMoverDB.model.[Favorite](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\model\Favorite.html)
  + peopleMoverWS.controller.[FavoriteController](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\FavoriteController.html)
  + peopleMoverDB.dao.[FavoriteDAOImpl](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\FavoriteDAOImpl.html) (implements peopleMoverDB.dao.[FavoriteDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\FavoriteDAO.html))
  + peopleMoverWS.controller.[FavoriteListController](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\FavoriteListController.html)
  + peopleMoverDB.model.[FavoritesList](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\model\FavoritesList.html)
  + peopleMoverWS.model.[FormattedMessage](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\model\FormattedMessage.html)
  + peopleMoverWS.model.[Location](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\model\Location.html)
  + peopleMoverWS.controller.[LocationController](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\LocationController.html)
  + peopleMoverDB.dao.[LocationDAOImpl](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\LocationDAOImpl.html) (implements peopleMoverDB.dao.[LocationDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\LocationDAO.html))
  + peopleMoverWS.util.[MyCx](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\util\MyCx.html)
  + peopleMoverWS.util.[PropertyReader](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\util\PropertyReader.html)
  + peopleMoverWS.model.[Route](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\model\Route.html)
  + peopleMoverDB.model.[Route](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\model\Route.html)
  + peopleMoverDB.dao.[RouteDAOImpl](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\RouteDAOImpl.html) (implements peopleMoverDB.dao.[RouteDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\RouteDAO.html))
  + peopleMoverWS.util.[Service](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\util\Service.html)
    - peopleMoverWS.util.[AlarmService](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\util\AlarmService.html)
    - peopleMoverWS.util.[TSOLookupService](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\util\TSOLookupService.html)
  + peopleMoverWS.[SimpleCORSFilter](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\SimpleCORSFilter.html) (implements javax.servlet.Filter)
  + org.springframework.boot.context.web.SpringBootServletInitializer (implements org.springframework.web.WebApplicationInitializer)
    - peopleMoverWS.[PMWebXml](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\PMWebXml.html)
  + peopleMoverWS.controller.[StopController](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\StopController.html)
  + peopleMoverDB.model.[Stops](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\model\Stops.html)
    - peopleMoverDB.model.[StopsWithWP](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\model\StopsWithWP.html)
  + peopleMoverDB.dao.[StopsDAOImpl](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\StopsDAOImpl.html) (implements peopleMoverDB.dao.[StopsDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\StopsDAO.html))
  + peopleMoverWS.controller.[StopsWithWPController](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\StopsWithWPController.html)
  + peopleMoverDB.dao.[StopTimesDAOImpl](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\StopTimesDAOImpl.html) (implements peopleMoverDB.dao.[StopTimesDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\StopTimesDAO.html))
  + peopleMoverWS.model.[Unit](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\model\Unit.html)
  + peopleMoverDB.dao.[UnitDAOImpl](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\UnitDAOImpl.html) (implements peopleMoverDB.dao.[UnitDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\UnitDAO.html))
  + peopleMoverWS.controller.[UnitListController](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\UnitListController.html)
  + peopleMoverDB.model.[User](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\model\User.html)
  + peopleMoverWS.controller.[UserAuth](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\UserAuth.html)
  + peopleMoverDB.dao.[UserDAOImpl](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\UserDAOImpl.html) (implements peopleMoverDB.dao.[UserDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\UserDAO.html))
  + peopleMoverWS.model.[ValidateUser](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\model\ValidateUser.html)
  + peopleMoverWS.controller.[WayPointController](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverWS\controller\WayPointController.html)
  + peopleMoverDB.model.[WayPoints](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\model\WayPoints.html)
  + peopleMoverDB.dao.[WayPointsAfterDAOImpl](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\WayPointsAfterDAOImpl.html) (implements peopleMoverDB.dao.[WayPointsAfterDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\WayPointsAfterDAO.html))
  + peopleMoverDB.dao.[WayPointsDAOImpl](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\WayPointsDAOImpl.html) (implements peopleMoverDB.dao.[WayPointsDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\WayPointsDAO.html))

## Interface Hierarchy

* peopleMoverDB.dao.[AlertDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\AlertDAO.html)
* peopleMoverDB.dao.[FavoriteDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\FavoriteDAO.html)
* peopleMoverDB.dao.[LocationDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\LocationDAO.html)
* peopleMoverDB.dao.[RouteDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\RouteDAO.html)
* peopleMoverDB.dao.[StopsDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\StopsDAO.html)
* peopleMoverDB.dao.[StopTimesDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\StopTimesDAO.html)
* peopleMoverDB.dao.[UnitDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\UnitDAO.html)
* peopleMoverDB.dao.[UserDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\UserDAO.html)
* peopleMoverDB.dao.[WayPointsAfterDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\WayPointsAfterDAO.html)
* peopleMoverDB.dao.[WayPointsDAO](file:///C:\Users\maurice\Documents\workspace-sts-3.6.3.RELEASE\ppmWS\doc\peopleMoverDB\dao\WayPointsDAO.html)

## Appendix H – Diary of meeting and tasks for the entire semester.

Meeting 1

|  |  |
| --- | --- |
| Date: | Wednesday, September 3, 2014 |
| Location: | Google Hangouts |
| Start time: | 8:00 pm |
| End time: | 8:30 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Introduction to Teammates |
|  | 2. Discuss ideas and gather information of what may be needed for the project. (use Miami-dade transit website as reference) |
|  | 3. Go over information in regards to PineCrest Mover |
| Assigned Tasks: |  |
|  | 1. Meet September 4, 2014 at FIU |
|  |  |

Meeting 2

|  |  |
| --- | --- |
| Date: | Wednesday, September 4, 2014 |
| Location: | FIU, JCCL Lab |
| Start time: | 8:15 pm |
| End time: | 9:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Gather questions for tomorrow’s conference call with mentor |
|  | 2.Share bootstrap and Google maps API resources |
|  | 3. Go over what is needed in document and feasibility matrix |
| Assigned Tasks: |  |
|  | 1. Meet September 5, 2014 for conference call 12:30pm – 1:30 pm |
| Duration: | 45 minutes |

Meeting 3

|  |  |
| --- | --- |
| Date: | Friday, September 5, 2014 |
| Location: | Conference Call |
| Start time: | 12:30 am |
| End time: | 1:30 pm |
| In Attendance: | Gabriela Wilson, Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Problem discussion and identification of possible users   * Agreed to start a Web/Mobil app * Defined guest user for general interaction * Defined registered user for “favorite bus stop” * Extra tool requested to manage routes and buses. |
|  | 2. Defined status of the GPS devices on the movers.   * Identified Trolley Company as the source of GPS data. |
|  | 3. Identified similar applications from other local governments. |
|  |  |
| Assigned Tasks: | 1. Create mock ups design 2. Define use cases 3. Contact the trolley company to define the data retrieval |
| Duration: | 60 minutes |

Meeting 4

|  |  |
| --- | --- |
| Date: | Saturday, September 6, 2014 |
| Location: | FIU, JCCL Lab |
| Start time: | 10:30 pm |
| End time: | 4:30 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Begin Feasibility Study document |
|  | 2.Begin Project Plant document |
|  | 3.Discuss best approach for implementation of project |
| Assigned Tasks: |  |
|  | 1. Meet September 7, 2014 10:30 AM at FIU |
| Duration: | 6 hours |

Meeting 5

|  |  |
| --- | --- |
| Date: | Sunday, September 7, 2014 |
| Location: | FIU, JCCL Lab |
| Start time: | 11:00 pm |
| End time: | 4:45 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Think of specifications |
|  | 2.Begin Specifications doc |
|  | 3.Discuss best approach for implementation of project |
| Assigned Tasks: |  |
|  | Submit first revision documents |
| Duration: | 5 hours 45 minutes |

Meeting 6

|  |  |
| --- | --- |
| Date: | Tuesday, September 9, 2014 |
| Location: | FIU, JCCL Lab |
| Start time: | 6:30 pm |
| End time: | 8:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Check Requirements Feedback |
|  | 2. Work on the mockups design |
|  |  |
| Assigned Tasks: | Continue working on the mockups |
|  | Improve use cases model |
| Duration: | 90 min |

Meeting 7

|  |  |
| --- | --- |
| Date: | Wednesday, September 10, 2014 |
| Location: | Conference Call |
| Start time: | 12:00 am |
| End time: | 12:30 am |
| In Attendance: | Ricardo Martinez, Maurice Pruna, Gabriela Wilson |
| Agenda: | 1. Check Requirements |
|  | 2. Check mockups design updates |
|  | 3. Check business workflow |
|  | 4. Status of the TSO api and GPS devices |
| Assigned Tasks: | Continue working on the mockups |
|  | Improve use cases model |
| Duration: | 30 min |

Meeting 8

|  |  |
| --- | --- |
| Date: | Thursday, September 11, 2014 |
| Location: | Starbucks |
| Start time: | 7:30 pm |
| End time: | 9:30 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Read documentation for Sencha |
|  | 2. Did some research for MVC architecture |
|  |  |
| Assigned Tasks: | Continue working on the mockups |
|  | Begin with interfaces |
| Duration: | 2 hours (120 minu) |
|  |  |

Meeting 9

|  |  |
| --- | --- |
| Date: | Sunday, September 14, 2014 |
| Location: | FIU JCCL Labs |
| Start time: | 3:30 pm |
| End time: | 7:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Continue with Interface implementation |
|  | 2. Provide feedback and assign roles |
| Assigned Tasks: | Continue with Interface Implementation |
|  | Improve use cases model |
| Duration: | 3hours 30 min |
|  |  |
|  |  |

Meeting 10

|  |  |
| --- | --- |
| Date: | Wednesday, September 17, 2014 |
| Location: | Conference Call |
| Start time: | 12:00 pm |
| End time: | 1:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna, Gabriela Wilson, David Gonzalez (TSO) |
| Agenda: | 1. Ask questions in regards to usage of API |
|  | 2. Database integration |
| Assigned Tasks: | Look over API Documenation |
|  |  |
| Duration: | 60 min |

Meeting 11

|  |  |
| --- | --- |
| Date: | Sunday, September 21, 2014 |
| Location: | FIU JCCL Lab |
| Start time: | 3:30 pm |
| End time: | 7:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Go over API Documentation |
|  | 2. Continue with interface implementation |
| Assigned Tasks: |  |
|  |  |
| Duration: | 3hour 30 min |
|  |  |
|  |  |

Meeting 13

|  |  |
| --- | --- |
| Date: | Thursday, September 25, 2014 |
| Location: | FIU JCCL Lab |
| Start time: | 7:30 pm |
| End time: | 9:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. API Connection |
|  | 2. Continue with interface implementation |
| Assigned Tasks: |  |
|  |  |
| Duration: | 1hour 30 min |
|  |  |
|  |  |

Meeting 14

|  |  |
| --- | --- |
| Date: | Saturday, September 27, 2014 |
| Location: | Starbucks |
| Start time: | 3:00 pm |
| End time: | 6:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. API Connection |
|  | 2. Continue with interface implementation |
| Assigned Tasks: |  |
|  |  |
| Duration: | 3 hours |
|  |  |
|  |  |

Meeting 15

|  |  |
| --- | --- |
| Date: | Sunday, September 28, 2014 |
| Location: | Starbucks |
| Start time: | 3:00 pm |
| End time: | 7:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Prepare Documentations |
|  |  |
| Assigned Tasks: | Ready to present for tomorrow’s class presentation |
|  |  |
| Duration: | 4 hours |
|  |  |
|  |  |

Meeting 16

|  |  |
| --- | --- |
| Date: | Thursday, October 2, 2014 |
| Location: | Conference Call |
| Start time: | 2:00 pm |
| End time: | 3:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. API Connection lag time |
|  | 2. Presentation of interface implementation |
| Assigned Tasks: |  |
|  |  |
| Duration: | 1 hour |
|  |  |
|  |  |

Meeting 17

|  |  |
| --- | --- |
| Date: | Thursday, October 8, 2014 |
| Location: | Starbucks |
| Start time: | 7:30 pm |
| End time: | 10:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | 1. Show Markers on Map |
|  | 2. Show routes on Map |
| Assigned Tasks: |  |
|  |  |
| Duration: | 2 hours 30 minutes |
|  |  |

Meeting 18

|  |  |
| --- | --- |
| Date: | Thursday, October 8, 2014 |
| Location: | Starbucks |
| Start time: | 12:00 pm |
| End time: | 1:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna, Gabriela Wilson, David (TSO developer) |
| Agenda: | 1. Figure out lag time for trolley |
|  | 2. Update Gabriela with progress |
| Assigned Tasks: |  |
|  |  |
| Duration: | 1 hour |
|  |  |
|  |  |

Meeting 19

|  |  |
| --- | --- |
| Date: | Monday, October 13 2014 |
| Location: | Starbucks |
| Start time: | 7:00 pm |
| End time: | 10:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | Analysis of the DB design |
|  | Work on the automatic insertion on the historical table |
| Assigned Tasks: | Look information to create the native build  Interact with Google Maps Api to draw an asynchrony marker |
|  |  |
| Duration: | 3 hours |
|  |  |
|  |  |

Meeting 20

|  |  |
| --- | --- |
| Date: | Saturday, October 18 2014 |
| Location: | Starbucks |
| Start time: | 5:00 pm |
| End time: | 10:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | DB adjustments  Work on the connection between Sencha and the PPM API |
|  | Redesign of the States diagrams |
| Assigned Tasks: | Create the Model and Stores needed to retrieve information from the PPM API |
|  |  |
| Duration: | 5 hours |
|  |  |
|  |  |

Meeting 21

|  |  |
| --- | --- |
| Date: | Tuesday, October 21 2014 |
| Location: | Starbucks |
| Start time: | 7:00 pm |
| End time: | 10:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | Analysis of the DB design changes requested by Gabriela |
|  | Work on the synchronic calls to for the Trolley position within Sencha |
| Assigned Tasks: | using Google Maps Api to draw an asynchrony marker |
|  | Start working on the Login and Registration modules |
| Duration: | 3 hours |
|  |  |
|  |  |

Meeting 22

|  |  |
| --- | --- |
| Date: | Tuesday, October 23 2014 |
| Location: | FIU CS Lab |
| Start time: | 7:00 pm |
| End time: | 10:00 pm |
| In Attendance: | Ricardo Martinez, Maurice Pruna |
| Agenda: | Adjust classes on a MVC fashion on Sencha |
|  | Adjust the synchronic calls to for the Trolley position within Sencha |
| Assigned Tasks: | Testing of the drawing of asynchrony marker |
|  | Testing on the Login and Registration modules |
| Duration: | 3 hours |
|  |  |
|  |  |

# References

<http://www.sencha.com/>

<http://www.sencha.com/forum/>

<http://spring.io/>

<http://www.sublimetext.com/>

<https://cacoo.com/diagrams/>

<https://developers.google.com/maps/documentation/javascript/tutorial>

<http://phonegap.com/blog/2013/11/20/SenchaPhoneGap/>