|  |
| --- |
| CPSC 488 – Software Engineering |
| Requirements Document |
| Multi Modal Routing Project |
|  |
| **Miller, Daniel V.;Petrusch, Zachary P.;Solomon, Christopher J., Schiller, Jordan M.** |
| **5/8/2014** |



|  |
| --- |
|  |

# Revision Sheet

|  |  |  |
| --- | --- | --- |
| Release Number | **Date** | **Description** |
| Revision 0 | 28 Jan 2014 | Rough draft of FSD, due to Project Supervisor |
| Revision 1 | 24 Feb 2014 | Added Class Structure, UML Diagrams |
| Revision 2 | 27 Mar 2014 | Updated |
| Revision 3 | 30 Apr 2014 | Various updates |
|  |  |  |
|  |  |  |
|  |  |  |

# Table Of Contents

[Revision Sheet i](#_Toc381021822)

[Table Of Contents ii](#_Toc381021823)

[1. General Information 1](#_Toc381021824)

[1.1 Purpose 1](#_Toc381021825)

[1.2 Scope 1](#_Toc381021826)

[1.2.1 High Level 1](#_Toc381021827)

[1.2.2 Low Level 1](#_Toc381021828)

1.3 Project References ………………………………………………………………………………………………………………………..1

[1.4 Glossary 2](#_Toc381021830)

[2.0 Project Background 2](#_Toc381021832)

[2.1 Current Functionality 2](#_Toc381021833)

[2.2 Current Methods and Procedures 2](#_Toc381021834)

[2.2.1 Input and Output 2](#_Toc381021835)

[2.3 Deficiencies 2](#_Toc381021836)

[3.0 High Level Application Specifications 3](#_Toc381021837)

[3.1 Project Overview 3](#_Toc381021838)

[3.2 Use Case Diagram 4](#_Toc381021839)

[3.3 Graphical User Interface 5](#_Toc381021840)

[4.0 Low Level Application Specifications 5](#_Toc381021841)

[4.1 Project Overview 5](#_Toc381021842)

[4.2 Use Case Diagram 8](#_Toc381021843)

[APPENDIX a](#_Toc381021844)

Flow Chart ………………………………………………………………………………………………………………………………………………b

Class Structure ………………………………………………………………………………………………………………………………………..c

Database Diagram …………………………………………………………………………………………………………………………………..d

Glossary ………………………………………………………………………………………………………………………………………………….e

# General Information

## 1.1 Purpose

The purpose of this document is to specify the functional requirements for the Multi Modal Routing Project [MMRP] that is being completed by students in Software Engineering [SE] during the spring semester, 21 January 2014 through 9 May 2014.

## 1.2 Scope

This section outlines the scope of the project and breaks the problem down into three distinct sections. Different students will be working on each section and the division of work will be listed with each section.

### 1.2.1 High Level

The scope of the high level is to design and implement a graphical user interface [GUI] for MMRP that allows user functionality with the existing MMRP code base. The GUI will allow a user to perform the following items:

1. Add a new route to the existing routes of the program.
2. Delete a route from the existing routes of the program.
3. Delete all routes from the program.
4. Create a shipment.
5. Query routes based upon multiple selection criteria.
6. Interface with the Low Level part of the application and Zeus[[1]](#footnote-1), an existing framework for shipments.

### 1.2.2 Low Level

The scope of the low level is to document and improve on the existing MMRP code base and implement the following new items:

1. Interface with the Low Level Part of the application and Zeus.
2. Obtain and implement real data into the system
3. Implement Error checking to ensure that vehicle assets are paired with the correct types of route, ensuring, for example, that a truck cannot be used to move a shipment from San Francisco to Tokyo

## 1.3 Project References

Below is a list of all references and external sources that were used throughout the course of the semester.

* Solomon C, Schiller J. Heuristics for Solving the Multi-Modal Routing Problem. Poster session presented at: Undergraduate Research in Science, Technology, Engineering, and Mathematics . 1st Annual Conference of the Pennsylvania State System of Higher Education; 2013 November 15-16; Slippery Rock, PA.
* Reference 2

# 1.4 Glossary

A complete glossary of all terms can be found starting on page e of the Appendix. Readers should refer here as needed.

## 2.0 Project Background

MMRP was started by two undergraduate students, Christopher Solomon and Jordan Schiller, during the preceding semester, 28 August 2013 through 13 December 2013. The work was presented Pennsylvania State System of Higher Education [PASSHE] Undergraduate Research in Science, Technology, Engineering, and Mathematics [STEM] conference poster session on 15 November 2013.

## 2.1 Current Functionality

MMRP currently generates multiple routes from one destination to another while keeping track of the quickest, shortest, and cheapest routes it has generated. All routes that are generated however can not be checked for correctness do the randomness of the data set.

## 2.2 Current Methods and Procedures

Once given a starting and ending location. A series of algorithms are run, including A\*, FirstFind, BestFirstFind, and NodeCrawler. When a route is generated it is displayed in a panel for the user to see. This route is also compared against the current cheapest, quickest, and shortest routes. If the current route is better than one of the previous best, it replaces it and is displayed in the appropriate panel on the form.

### 2.2.1 Input and Output

User input is obtained via two drop down lists that select the starting and ending locations for the route. The best routes are displayed in a tab container which will draw and print out the details of the corresponding route. Also the most recent route generated is displayed alongside the tab container.

## 2. Deficiencies

Lack of real world data makes the routes hard to check. Also some of the current algorithms run for a long amount of time. This needs to be cut down so we can quickly generate enough routes to use in a Genetic Algorithm. The current data structure may not best represent a real world model and needs changed.

# 3.0 High Level Application Specifications

The purpose of this section is to cover the high level end of the application. As previously stated the high level of the project will be handled primarily by students Dan Miller and Zach Petrusch. This section will cover the general layout of the GUI and provide rough ideas as to the look of the application.

## 3.1 Project Overview

The following list provides an overview of the High Level of the project. The list can be matched to the flow chart found in the Appendix at the end of this document.

1. Shipments
   1. Create Shipment
      1. Get Shipment Information
      2. Number of Vehicles Required For Shipment
      3. Options [Recieves Info From Low Level of Application]
         1. Schedule A Shipment
            1. Schedule For Immediate Delivery
            2. Schedule For Delivery At Later Time
         2. Send Scheduled Shipment To Low Level For Processing
   2. Delete/Cancel Shipment
2. Routes
   1. Add A Route
      1. Check For Duplicates In Database
   2. Delete A Route
      1. Delete One Route
      2. Delete All Routes
   3. Import
      1. Overwrite Existing Routes
      2. Append To Existing Routes
         1. Check Duplicates
   4. Modify Route
      1. View Route
      2. Change Route
3. Vehicles
   1. Add A Vehicle
      1. Check For Duplicates In Database
   2. Delete A Vehicle
      1. Delete One Vehicle
      2. Delete All Vehicles
   3. Import
      1. Overwrite Existing Vehicles
      2. Append To Existing Vehicles
         1. Check Duplicates
   4. Modify Vehicle
      1. View Vehicle
      2. Change Vehicle

## 3.2 Use Case Diagram

The High Level Application needs to allow the user to interact with shipments and vehicles. It also needs to allow the user to exit the program cleanly. Both shipments and vehicles break down into subcategories. Shipments interact with the user, allowing the user to create shipments, delete shipments, and modify shipments. In similar fashion vehicles interact with the user, allowing the user to create vehicles, delete vehicles, modify vehicles, and import vehicles from an external source. Vehicles and shipments both use the Low Level Application to perform communicate the results of the low level algorithms back to the user.



Figure 1 - High Level Use Case

## 3.3 Graphical User Interface

The goal of the GUI is to create a simple and intuitive interface for MMRP. The GUI needs to support all the functionality of the Low Level of the application while maintaining simplicity so that non-technical users are able to use the application with no problems. As such the following mock-up is the proposed home screen for the application.



Figure 2 - Wireframe Diagram Application

# 4.0 Low Level Application Specifications

The purpose of this section is to cover the high level end of the application. As previously stated the high level of the project will be handled primarily by students Jordan Schiller and Chris Solomon.

## 4.1 Project Overview

The following list provides an overview of the Low Level of the project. The list can be matched to the flow chart found in the Appendix at the end of this document.

1. Algorithms
   1. Create Algorithms
      1. Vehicle-Based Algorithms

How Vehicle-Based Search works

1) See if both starting and ending points can be reached by this mode of travel, if not we are done

2) 30% of the time try to find a direct path from start to end using this mode of travel, then pick the one with the lowest metric cost.

3)70% of the time pick a segment using this travel type 50% of the time we will choose the lowest metric cost and 50% of the time we will use a random segment that uses this travel type.

4)See if we can reach the end from this new location, if not repeat from step 3

* + - 1. Rail Only
      2. Plane Only
      3. Train Only
      4. Truck Only
    1. Cost-Based Algorithms
    2. Time-Based Algorithms
    3. Weighted Algorithms
    4. Random Algorithms
       1. NodeCrawler

How Node Crawler Works

1) Grab the segments connected to the start node

2) Pick a random segment to go down

3) Add this segment to the path

4) Check to see if we can now connect to the end node

5) If we have reached the end node we are done, else repeat 3

* + - 1. NodeCrawlerFirst

How nodeCrawlerFirst Works

1) Grab the segments connected to the start and end nodes

2) While we arent connected to the end node do the following

3) Pick a random segment to go down

4) Check to see if we can now connect to the end node

5) Add this segment to the path

6) If we have reached the end node we are done, else repeat 3

* + 1. Guided Algorithms
       1. A Star
       2. BestFirstFind

How bestFirstFind Works

1) Grab the segments connected to the start and end nodes

2) While we arent connected to the end node do the following

3) Pick the best segment, using the metric to go down

4) Check to see if we can now connect to the end node

5) Add this segment to the path

6) If we have reached the end node we are done, else repeat 3

* 1. Choose Algorithms
  2. Balance Algorithms
     1. Tracking Algorithm Performance

1. Implementing Real World Data
   1. Collecting Relevant Real World Data
      1. Collect location, vehicle, and company data.
   2. Incorporate Data
      1. Add database information for locations, vehicles and companies based on real world data.
2. Error Checking
   1. Check Routes
      1. Ensure the routes that are generated by the algorithms are logically viable and are possible.
   2. Log Errors
      1. Track the errors, to find out where they occur, either in the algorithms or in the code itself.
   3. Correct Errors
      1. Fix the bugs, or place restrictions on algorithms so errors cannot occur.
3. Zeus Interface
   1. Incorporate Existing Zeus Structure
4. API Document To Interface With High Level
5. Genetic Algorithms
   1. Algorithm Modification Through GA
   2. Algorithm Selection Through GA
   3. Environment Modification Through GA
6. Dynamic Support
   1. Real-Time Agents
      1. Shipment Constraint Agent
      2. Location Constraint Agent
      3. Vehicle Constraint Agent
      4. Route Constraint Agent
      5. Random Agents
   2. Route Repair
      1. Repair Algorithms

## 4.2 Use Case Diagram

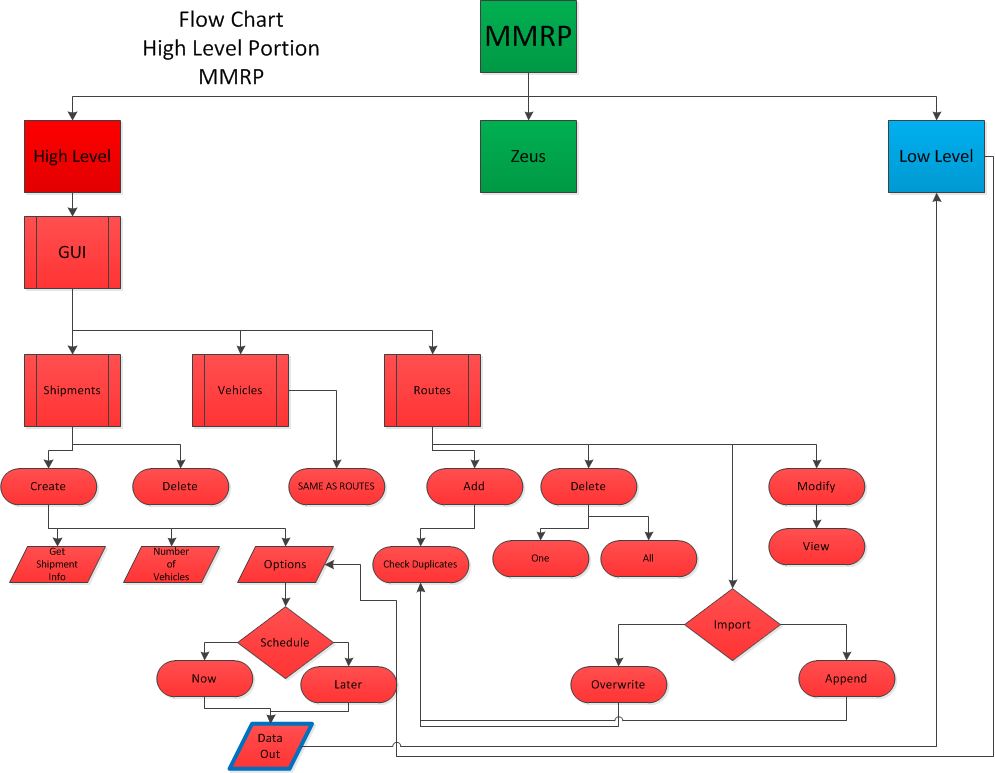
The Low Level Application needs to process the data passed in by the User via the graphical user interface. It needs to process shipment requests passed in as well as vehicle data. After performing the desired computations on the data it needs to return the data to the graphical user interface for the user.



Figure 3 - Low Level Use Case

# APPENDIX

This Page Intentionally Left Blank







**Glossary of Terms**

Vehicle- an abstract thought to represent any mode of transportation

Travel Type – a more specific describer of a vehicle mode

Truck- a vehicle used on road ways

Rail- a vehicle used on a pre-laid track such as a train or trolley

Cargo- a vehicle used to transport across waterways

Plane- a vehicle used to transport through the air

Bike – a man powered vehicle with limited capacity. Travels along roads and usually shorter distances

Location- a point of interest with latitude, longitude and name values

Segment- represents a path between two locations used by a specific vehicle. Attributes include time to travel, distance and cost

Shipment- a package to be shipped from one location to another based on its size, and priority

Shipment History- a list of segments a shipment has/will travel along with a number indicating the order in which the segments are traversed

Schedule- The list of segments corresponding with a given vehicle

Cost- monetary value to ship across a segment

Priority- High = fastest possible route; Medium = Balanced mix between cost and time; Low = cheapest possible route

1. Zeus is an existing piece of Vehicle Routing software written and maintained by Dr. Sam Thangiah provided for use. [↑](#footnote-ref-1)