SOFTWARE REQUIREMENTS SPECIFICATIONS REPORT

CS 2XB3

The following report covers the Software Requirements Specifications of the No Left Turn Route application.

Group 19: Hassaan Malik, Trevor Rae and Paul Warnick

TABLE OF CONTENTS

Contents

1. INTRODUCTION

1. Introduction

1.1 PURPOSE

The purpose of this SRS (Software Requirements Specifications) is to give any user or developer an idea of the functionality of the application created in this project. By the end of the SRS the reader should have a general knowledge of the application.

1.2 SCOPE

(1) The software application of discussion that will produced in this project is: No Left Turn Route

(2) This final intent of the application will be to mirror services like Google Maps by creating an optimal route from one distinct location on a map to another. The key difference is that the application will run an algorithm to avoid as many left turn as possible.

(3) There are a number of benefits associated with creating an application to function in this manner for example:

- Reduced traffic due to the reduced number of people turn left at intersections
- A greater efficiency in delivery truck routes due to the speed up of driving
- Fewer car accidents due the lessened number of vehicles crossing traffic

1.3 OVERVIEW

(1) The following sections of the SRS contain a general description of the project (i.e. its functions, its types of users, etc.) and a section describing all requirements (function and non-function) used in creating the software project.

(2) The SRS is organized in a manner that is easy to follow. Thus making specific portions easy to find and update if need be. All sections are listed in the table of contents with appropriate page numbers.

2. GENERAL DESCRIPTION

2. General Description

2.1 PRODUCT FUNCTIONS

A number of function will be preform by the software including the ability to:

- Find an optimal route between two distinct locations on a map
- Find the a route between two distinct locations on a map replacing any left turns with a number of right turns to return you to the original intersection
- Calculate the difference in time between taking a route with no left turns or a regular route based on average waiting times of left turns during rush hour
- Zoom and pan through the map

2.2 USER CHARACTERISTIC

The intend users of this software can range from large delivery based companies such as UPS and FedEx to the average commuter trying to find the quickest way to get to work. The potential users here are unlimited due to the fact that the application assist in the everyday task of driving from one place to another.

2.3 GENERAL CONSTRAINTS

The constraints limiting certain development aspects of creating a mapping software are as follows:

- The size of the map that can be navigated is directly proportional to the power and size of the host computer running the software
- Without using complex data structures such as a Geodatabase (used in GIS mapping software) the realism of the map can only reach a certain extent. (curved streets are represented as a number of straight lines)

2.4 ASSUMPTIONS AND DEPENDENCIES

The software product will require the host machine to have the newest version of Java installed along with the ability to run .jar files.

3. SPECIFIC REQUIREMENTS

3. Specific Requirements

3.1 FUNCTIONAL REQUIREMENTS

3.1.1 Reading Data

Introduction:

All information used in the program is given in the form of two .txt files. One containing a list of all intersections (vertices) and another containing all streets (edges) in the map.

Inputs:

The first of the text files (containing the intersections) is formatted to have a (in order):

- *Node ID* to later identify that intersection
- *X Coordinate* for placing the intersection on a map
- *Y Coordinate* for placing the intersection on a map

The second text file (streets) is formatted to have a (in order):

- *Edge ID* to later identify the street
- *Start Node ID* to show where the street starts
- *End Node ID* to show where the street ends
- *Distance* to give a length to the street

Processing:

All the data will be read into arrays, to allow for quick access.

Outputs:

The array indices will be added one at a type to an edge weighted graph data structure.

3.1.2 Constructing the Map

Introduction:

An edge weighted graph is required to be constructed in order to represent the map

3. SPECIFIC REQUIREMENTS

Inputs:

The output of the previous requirement will be used as inputs here (intersections and streets) which will create a necessary dependency.

Processing:

Each input will be added to the map in the appropriate location (based on the data of the input).

Outputs:

A map representing a city based on the input files with all appropriate streets and intersections.

3.1.3 Finding the SPT (Shortest Path Tree)

Introduction:

Once the map has been constructed a SPT must be constructed from the users given start location.

Inputs:

The map constructed in the previous requirement along with the user given starting location.

Processing:

The use of a modified version of Dijkstra's algorithm will be used to calculate the tree with the give inputs.

Outputs:

A tree containing the shortest path to any intersection given the users inputted starting location.

3.1.4 Removing Left Turns

Introduction:

Once the SPT requirement has been fulfilled, the route can be modified to eliminate all left turns.

Inputs:

The SPT generated in the previous step will be used as the input

Processing:

Every time a left is encountered, a series of right turns will be added to eliminate it.

Outputs:

A route from one intersection to another (both specified by the user) containing as few left turns as possible.

3.2 NON-FUNCTIONAL REQUIREMENTS

3.2.1 Performance

Performance plays a key role in the application's overall usability. For this reason Dijkstra's algorithm was used in order to minimize runtime and allow for the most efficient finding of the shortest path. These design choice allow for runtime proportional to $S \log I$ in the worst case (where S is the number of streets and I is the number of intersections) to find the SPT. On top of this in order to optimize the application Dijkstra's algorithm terminates once the desired path is found (i.e. shortest path from start point to end point of route).

3.2.2 Reliability

The application is able to maintain reliability due to the use of Dijkstra's algorithm. This algorithm has been thoroughly tested time and time again and is known as the best choice of algorithm is finding SPT's.

3.2.3 Availability

Due to the application not requiring an internet connection to run (compared to most other mapping applications) it can be used anywhere as long as the device running it has the proper data files.

3.2.4 Maintainability

The applications source code is very lightweight meaning it is updates (if needed) would be easy to implement. On top of this due to the use of well-known algorithms that have been created in hopes of few changes being needed, (i.e. Quicksort, Dijkstra's, etc.) the applications code itself will likely not need to be updated as well.