SFWRENG 2XB3 Final Project

Requirements Specification

Department of Computing and Software
McMaster University

Group 8
Project Name: NavSafe
Version Number: 1.0

Arkin Modi - 400142497
Benson Hall - 400129627
Joy Xiao - 400125285
Leon So - 400127468
Timothy Choy - 400135272

Last updated: April 12, 2019

Contents

1 Project Domain

1.1 Purpose and Objective of the Project

The purpose of NavSafe is to help drivers and pedestrians travel from one destination to the next in the shortest and safest route evaluated based on past collision data.

1.2 General Description

NavSafe is a mobile application brings a new, innovative navigation system which focuses on safety. Vehicle collisions are major concern, especially those which result in injury or death. There are many high-risk intersections which collisions are more likely to occur due to the design of the intersection, as well as other factors out of the traveler's control. This application allows both drivers and pedestrians, such as foot-pedestrians and bicyclists, to travel safer through mapping a navigation route which reduces the likelihood of collisions based on past collision data obtained from Seattle GIS Open Data. The application also considers specific travelling conditions such as weather, road, and daylight conditions to assign a specific weight to a route for a specific time and location.

1.3 Stakeholders

1.3.1 The Client

The clients of NavSafe consist of the Professor, Dr. R. Samavi, and the teaching assistants of SFWRENG 2XB3.

1.3.2 The Users

The targeted users of NavSafe consist of drivers, cyclists, and pedestrians who seek a safer way to navigate from one destination to another.

1.3.3 Other Stakeholders

The other stakeholders this application will impact are the general population of the area where this app is used.

2 Functional Requirements

2.1 Read Module

The read module is designed to take in the data-set (csv file) and parse the data into a usable form. This is done by reading in the file, one line at a time, and using the data to create a collision object. The object is then stored in an array.

2.2 Collision ADT Module

The collisions ADT module is responsible for providing a data structure for which the rest the program will use to perform its various tasks. The data structure represents a collision. The data structure contains a field for all data provided in the csv file.

2.3 Intersection ADT Module

The intersection ADT module is responsible for providing a data structure for which the rest of the program will use to perform its various tasks. The data structure represents an intersection between two roads. The data structure contains a field for all data provided in the csv file.

2.4 Street ADT Module

The street ADT module is responsible for providing a data structure for which the rest of the program will use to perform its various tasks. The data structure represents a road between two intersections. The data structure contains a field for all data provided in the csv file.

2.5 Graphing Module

The graphing module will be responsible for creating a weighted edge graph out of the intersections. The nodes are the intersections while the edges are the streets. The weights will be determined by the distance between two connecting intersections and the number of collisions. The shortest safest path will be implemented using Dijkstra's algorithm.

2.6 Sort Module

The sort module will be responsible for sorting the collision or the intersections. Collisions will be sorted based on the severity of the collision. Intersections will be sorted in alphabetical order. The sorting module will be implemented with merge sort (top-down).

2.7 Search Module

The search module will be responsible for searching for a given intersection or collision. This will be done with linear search as the result will be based on multiple factors.

3 Non-Functional Requirements

3.1 Reliability

The program should always return the safest path from the start location to the destination (within the city of Seattle, Washington). Safety of a path is determined by the collision data which is comprised of the collision history for Seattle, Washington.

3.2 Accuracy of Results

The program's results should be high in accuracy. The outputted result will be a path. This path must be following roads and should not take the user through places where a vehicle cannot go (both legally and physically). The path should also have the least amount of collisions possible.

3.3 Performance

The performance of the program must be relatively quick. Typically when the user will use this program, they will be ready to leave and will want to know the way immediately. For this reason the program should be fast and responsive.

3.4 Human-computer Interface Issues

The interface needs to be simple and easy to understand. As mentioned in the Performance section above, users will typically be using this program when they are ready to leave. This implies a sense of urgency and the user will need to be able to use the program in a quick manner. Having a simple user interface will enable faster operations and a smaller learning curve. Also, with the large target user based, people who drive in Seattle, a simple and easy to understand user interface will encourage more people to use the program.

3.5 Scalability

The data set for this program has the potential to grow as new roads are added and more collision happen. The current implementation uses data from 2004 to 2018. As the data set grows with more data becoming available, older years should no longer be considered. Since technology is always improving and change the automobile industry and new safety technology is being added to the streets, hot spots for collision will move. Therefore the application should only consider a select specific time period to focus on when calculating the route. This will allow scalability even as the data sets continue to grow.

3.6 Constraints

3.6.1 Operating Constraints

The program will be made with the Java SE Development Kit 11 in Android Studio and exported as an apk to be installed and run on an Android device. The program will need to run on Android 4.0 or newer (recommended OS support from Google).

3.6.2 Physical Constraints

The program will need to be able to run on an Android smartphone.

4 Development and Maintenance Process Requirements

4.1 Quality Control Procedures

4.1.1 Unit Testing

Individual units/components of the program will be tested for correctness and reliability. The purpose of this testing to validate that the software is operating as needed to be.

4.1.2 Continuous Integration Testing

Individual modules will be tested for compatibility with existing modules. The purpose of this testing is to prevent incompatible modules from merging into the master branch and causing problems for all members.

4.1.3 System Testing

The completely integrated system will be tested against the specification. The purpose of this test is to verify the program and specification are aligned.

4.2 System Maintenance Procedures

A likely changes to the system maintenance procedures is to focus on providing one path as appose to multiple paths. The change would improve run-time and development time.