2011

System Requirement Specification

Group I

Ali

Group i

2011-01-01

Revision History

- 28/03-11: changed front page.
- 28/03-11: Assumptions and dependencies updated.
- 28/03-11: Software interfaces updated
- 29/04-11: added some more requirements in secondary requirements field, and updated the software interfaces field.

1.0 Introduction

1.1. Purpose with the system and the document.

a) Functionality: The system will basically show the user the path to the user's desired destination. It will help the user along the way to be on the right path to his destination.

b) External interfaces:

c) Performance: the performance of the software will be of great value in this project. Therefore the developers will strive to use efficient API's and efficient algorithms in order to achieve as high performance of the software as possible

d) Attributes:

e) Design constraints imposed on an implementation: the developers will restrict the user by designing a user interface that easy enough for any user to understand.

This document will show the requirements for the GPS project of Group I. The requirements will be divided into three sub sections *Essential*, *Conditional*, *Optional.* The people working and being affected by the project are following:

Dmitry Igoshin
Johan Wikström
Kirill Blazhko
Mahsa Mirtalebi
Sebastian Hansson
Ali Nazar

The purpose of this document is to provide developers and stakeholders with the chance to review the system and its functions required to build this system. This document will serve a development guide for developers and a list of functionality and requirements for the stakeholders that can be changed by the stakeholders.

1.2 Scope

a) Description: This software is going to be developed by the purpose of allowing users to travel easily and effectively. The software will also be able to read a set of coordinates called geocache and the software will provide the necessary and fastest ways towards that geocache.

The team's goal is to provide the user with a program which will be fast and efficient in terms of usage.

1.3 Definitions, acronyms, and abbreviations.

1.4 References

1.5 Overview

The SRS is organized with a table of contents and will show you the requirements based on the sub headers that will be provided while explaining the requirements.

In section 2.0 you will be familiarized with the background of the requirements so that you can have an easier time understanding the requirements and also their specification implied in section 3.0.

The document also provides a glossary section which will help you to understand various terms specified in the document. A reference list is also provided so that you can account for where the details described in this document has been taken from.

2.0 Overall Description

2.1 Product perspective

This software is similar to a GPS navigation system that you can easily buy nowadays in electronic stores. The idea behind developing this software is basically the same as the navigation systems in the stores. However this system will not be as full of handy functions as provided by many navigation systems in the market today. This is because of the fact that this project is a university related project containing students as developers and also that the development team is not that large.
The system will behave exactly like a navigation system, providing the user with the ability to go to a desired place without any difficulties and as efficiently as possible. The system will contain a somewhat intuitive user interface and also contains a hardware component in the shape of a GPS receiver which provides GPS signals with coordinates that will be handled by the program being developed.

The user interface will provide the user with options like zoom in/out functions to see the map more clearly. The interface shall also provide with functions as enabling the user to see his/her current speed and will also provide an option for finding the fastest path to the user's destination. The user will also be able to import a geocache to the software which will be read by the program and used to show the coordinates provided in the geocache file. The user will also be shown a fastest path towards that Geocache.

2.1.1 User Interface

The design of the user interface will be very simple and therefore also constraining the user to do anything nonrelated to the program leading to errors. However if the user provides for example some wrong Geocache with invalid coordinates, the system will detect that and show that to the user using a message dialog.

2.1.2 Hardware Interfaces

The hardware unit e.g. the GPS device will provide the program with the essential coordinates so that the program can easily show the user where he is located on the map. The GPS device is an essential part of the software and the user will need it every time he/she wants to use the system. Since the system will be used on a computer, the hardware device will need some serial ports e.g. USB ports so that it can be connected to the computer and then read by the program. The program will use necessary libraries that can communicate with comport/serial ports in order to get the information from the GPS device.

2.1.3 Software interfaces

The following will be used as software interfaces for our system.

- Name: RXTX API
- Mnemonic:
- Specification number:
- Version nr: RXTX-2-2-20081207
- Source: http://www.cloudhopper.com/opensource/rxtx/
- Was used for the purposes of communicating with the GPS device.

- Name: Java communication API
- Mnemonic:
- Specification number:
- Version nr: 3.0
- Source: https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_SMI-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=JAVACOMM-3.0.1-LX-SP-G-F@CDS-CDS_SMI

- Name: JMapViewer API
- Mnemonic:
- Specification number:
- Version: 2011-02-19
- Source: http://svn.openstreetmap.org/applications/viewer/jmapviewer
- Used to implement the map.

Name: facebook-java-api

- Dec 2 - Release: 3.0.2 Lots Of Fixes, and changes to be in sync with api (deprecating methods, adding methods, etc)
- Used for the implementation of the non-functional requirement #15.

- Name: Twitter4J
- Description: an API to connect with java, which allows you to access your twitter account and post tweets.
- Version: 2.2.2
- Source: http://twitter4j.org/en/index.html
- Used for the implementation of the non-functional requirement #15.

- Name: Smack API
- Description: This application is a networking API, easy to use, fast and efficient API for sending messages between computers
- Version 3.2.0
- Source: http://www.igniterealtime.org/projects/smack/index.jsp
- Used to be able to implement the non-functional requirement #10

2.2 Product functions

- The software should have cross-platform support. It should be compatible on Windows and Linux or Mac operative systems.
- It should be able to get the current location of the user. The user will be able to see his location on the map.
- It should be able to handle LOC files for coordinates, to get the target location. That is the geocache feature should be implemented.
- The system needs to show a calibrated map with N/W/S/E and necessary graphics to make it easy to read.
- The map should be able to viewed at any size you want and it should able to manipulated so that the user can choose between deeper terrain details or lighter with a zoom function.
- The map should include a scrolling function

- The program should include an intuitive GUI comprising at least buttons for manipulating the map and other buttons for the necessary features.
- The system needs to specify to the user somehow, either through voice function or message function that the target area has been achieved.
- It should show where the current location and target location is on the map, graphically.
- The program should notify the user if the user is heading in a wrong direction rather than the one been specified for him by the system.
- It should update the users current location at a regular basis.
- The guidance provided to a destination should be graphically expressed in the system, somehow by line.
- With the updated current location, it should update the best suited way to the target location

2.3 User Characteristics

The user needs to be aware that the system is developed for purpose of more efficient navigation either while car driving or hiking etc. The user must know the English language to be able to follow the instructions the device will provide. The user needs to be familiar with systems like navigation systems in order use this program or computers in general.

2.4 Constraints

2.4.1 Hardware limitations

The GPS device will provide with signal data and that data will transmitted by the GPS device according to the signal strength. The signal strength is affected by the environment the user is currently attending. If the signal strength is low, the data will not be flow from The GPS device will not be as fast and this may affect the performance of the software.

This signal transmitting is also a parallel activity along with the software, and the software will not function without the GPS device, and therefore the system is constrained by the function of first checking if the software is connected to GPS device. If this is not the case, the user will not be able to do anything with software.

2.4.2 Reliability requirements

The program should be able to run on any computer with any platform, so the development of the software must be implemented by taking this into consideration.

The software should not, in any case, crash because of errors in internal logic and code. Therefore the software needs to be tested multiple times in order to assure that it is safe for a user to use this software without facing any anomalies.

2.4.3 Performance requirements

The developers will strive to make the software as fast as possible. This will be achieved by creating the system in a sense that the time to load the map should not take a lot of time. The GPS device that provides the map with signals/ coordinates therefore needs to fast in response so that the map can calibrated quickly. So the developers will focus on the map quality and the performance of showing the map to the user. The developers will also chose efficient algorithms and methods in order for the software to take as minimum of space memory as possible, therefore making it much faster.

2.5 Assumptions and dependencies

1) In one of the requirements, the user is intended to be highlighted somehow that he/she has reached the destination. This has been specified with two implementation ways, either through voice messaging or normal text messaging. However the latter case is that the developers may prefer to use the text messaging implementation.

2) The library used for the GPS device provided some external .so files to make it convertible for the Linux platform, and for the GPS system to work, these extra files will be provided so that the user can put in them in the correct folder as specified in the HOW-TO-INSTALL readme file.

3) we are assuming that we can achieve all of the listed requirements, including the non-functional requirements, however, the project has a dead line of two months, and while looking forward and also looking at all of the requirements, the developers may prefer to take the requirements that are necessary and leave the rest for later implementation if more time is available.

2.6 Apportioning of requirements

1) The developers would like to implement a function that can calculate the distance of a given set of coordinates and display this on the interface.
2) The developers would like to implement a function that will give the instructions to the user along the way to his/her destination through voice.
3) The developers would like to create a GUI that can be transparent.
4) The developers would also like to implement a function that can calculate the fastest way, the most efficient way and the simplest way in terms of destination traveling.
5) Enabling the user to input a target location of his/her own choice.
6) The developers would like to create a function for enabling the user to save routes into the memory of the software and then access that route at any time.

7) To implement a functions called Multilanguage function, so that the program can run on different languages. Since the developers are from different countries, this function can be implemented very easily.

3.0 Specific Requirements

In this section, all of the requirements that were created during the requirements elicitation process are listed.

1. The software should have cross-platform support
2. It should be able to get the current location
3. It should be able to handle LOC files for coordinates, to get the target location
4. It should show a map, N/S W/E calibrated
5. The map should include a zooming function
6. The map should include a scrolling function
7. It should include a GUI with at least zooming and scrolling buttons
8. It should give a notification when the target is reached
9. It should show where the current location and target location
10. is on the map, graphically
11. It should give a notification when the user is heading in the
12. wrong direction
13. It should update the users current location at a regular basis
14. The guidance should be one line from current location to target location.
15. With the updated current location, it should update the
16. suited way to the target location

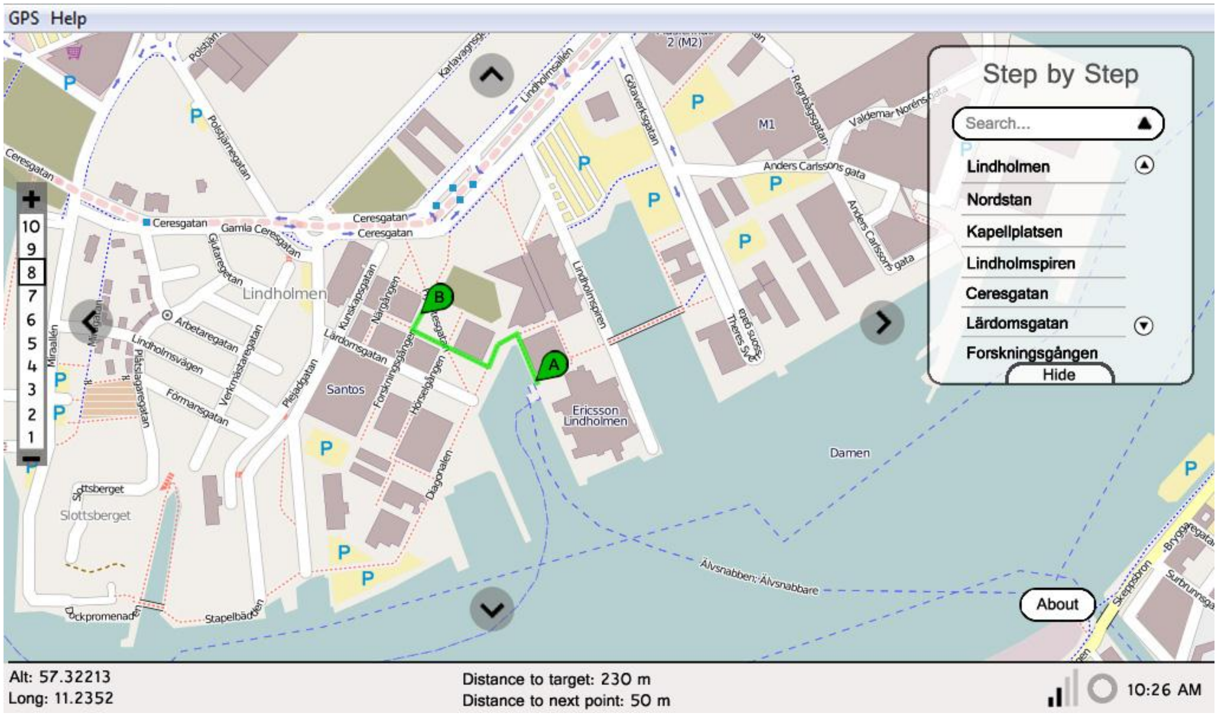
Non-functional requirements:

1. ***It should show the distance***
2. ***It might have sound indications***
3. **It should have a Colorful and transparent GUI**

4. ***It should have different modes (fastest way, nicest way, easiest way***
5. ***It should have a save/load function for routes***
6. ***It should be able to estimate the time needed to get to the target***
7. ***The user should be able to input a target location***
8. **Multilanguage interface**
9. **Compass**
10. **Networking between two computers so that both can see coordinations on the map. Chat function/guidance function.**
11. **Showing some areas of interest on the map by a video, the video will be showing the surrounding area and the area of interest**
12. **caloric burnage (using speed parameter).**
13. **Implement a self-rendered map.**
14. **Connect an XBOX joystick, so that the maps position can be changed using the joystick.**
15. **The application should be connected with twitter and facebook, you should be able to update your wall/tweet your location. This is to make the program more personal.**

Assumed design of the software.

The following picture will show you a design constructed in order to have a basic vision of how the software will look like at the end of implementation. This design was specified by all of the developer together.



Glossary

1) GPS: Global Positioning System
2) API :application programming interface
3)

References:

1) Software Engineering Standards Committee of the IEEE Computer Society , *IEEE Recommended Practice for Software Requirements Specifications,* Approved 25 June 1998.