Software Requirements Specification (SRS)

Group 1: Travel Safe

SFWRENG 2XB3: Software Engineering Practice and Experience: Binding Theory to Practice

Name	MacID	Email Address
Abeer Al-Yasiri	alyasira	alyasira@mcmaster.ca
Gary Gong	gongc12	gongc12@mcmaster.ca
Almen Ng	nga18	nga18@mcmaster.ca
Veerash Palanichamy	palanicv	palanicv@mcmaster.ca
David Yao	yaod9	yaod9@mcmaster.ca

Revision Page

Revision History

Date	Version	Description	Author
March 6 2020	1.0	Initial SRS	David Yao
April 12	1.2	Changed few requirements	Veerash Palanichamy

Table of Contents

1. Introduction

1.1 Purpose

This document outlines the intended functionality and externalities of the Android application project, Travel Safe. It elaborates on what the software is supposed to do and how it is supposed to perform. In addition to defining the behavior, the specification serves as a contract between the user and the software developing team and a confirmation of what the software is trying to accomplish for the software developers.

1.2 Scope

Travel Safe aims to help with travel decisions with the aid of weather predicting software application. The objective is to use a detailed database to draw relations between the input statistics that will deliver a system to assist the user with choosing their future travel location. The system will offer a user-friendly interface which highlights the benefits and risks of the location in interest. The project will adopt several software qualities and principles to achieve the intended behaviour outlined in the requirements.

1.3 Definitions, Acronyms, and Abbreviations

Android: A mobile operating system (OS) developed by Google.

Application Programming Interface (API): Set of definitions and protocols for integrating application software.

Database: An organized collection of data.

Eclipse IDE: An Integrated Development Environment (IDE) that enables many customizations and extensions. It is known for its Java IDE.

FR: Functional Requirement

ID: Identification

IEEE (Institute of Electrical and Electronics Engineers): A professional electrical and electronics engineering association dedicated to technological innovation and advancement.

Graphical User Interface (GUI): The visual output displayed by an application to the user to aid user interaction.

Software Requirements Specification (SRS): A document that defines functions and behavior of a system or software application as well as the constraints to operate by.

User: A person who uses the software.

1.4 References

Michigan State University. (1998). *IEEE STD 830-1998* [PDF file]. Retrieved from http://www.cse.msu.edu/~cse870/IEEEXplore-SRS-template.pdf

National Centers for Environmental Information. (n.d.). *Storm Events Database.* Retrieved from ftp://ftp.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/

1.5 Overview

In the following section, Overall Description gives insight on the functionality of the software. It outlines the factors that will affect the application and serves the purpose of establishing context for the more technical requirements in the following section, Specific Requirements. The Overall Description section is primarily intended for the user to read which means that the writing should be easier to understand, thus the language is more informal.

The Specific Requirements section is written for the developers and details the technical aspects of the functionality of the software. In this section, the software requirements are listed and sufficiently elaborated on to enable the developing team to satisfy the requirements and for testers to test the application to ensure that it meets these requirements. There would be descriptions of the input and output of the system and all the functions that are to be performed by the system in response to the input or in support of the output. We based this section of the document on the “Template of SRS Section 3 organized by object” in the IEEE SRS 1998 PDF document in Annex A.

The third and final section of the SRS is the Appendix in which potential features that may be implemented in future versions of the software are listed and described. The Appendix is not to be considered part of the requirements.

2. Overall Description

2.1 Product Perspective

2.1.1 System Interfaces

The software is intended to be run entirely on the Android platform, through which all user interactions will also be processed. The application will be run on the latest version of Android that is supported on Android Studio.

2.1.2 User Interfaces

The application will include a GUI incorporating menus, input text boxes, icons, and buttons. Users will interact with the application via touch by inputting their desired locations, viewing data representations on the map, and scrolling through output data summaries. The GUI can be divided into three main conceptual scenes - the map, the location input, and the data summary for the given location. The data summary will be given largely in text, with automated messages depending on the program's conclusion about the given location.

2.1.3 Hardware Interfaces

There is no immediate plan to make use of the phone's built in location-finding hardware at this time. Its intended use is for individuals to plan their trips in advance, so little benefit is derived from allowing the user to automatically view information about their present location.

2.1.4 Software Interfaces

The application will make use of Google's map API to display data for the given location as part of the wider United States. The API is used mainly as a visual representation for the user. Drive Safe will be developed using Google's Android Studio IDE for full support on their Android mobile operating system across multiple models of Android mobile devices.

2.1.5 Memory

Garbage collection will be a necessity to prevent memory leaks. Android itself incorporates memory management functionality but its use should be kept to a minimum.

2.2 Product Functions

Summary of the product features and functions accessible to the user:

I. Weather report covering weather risks that includes information on:
   a. Types of hazards prominent in location
   b. Number of hazards happening in the inputted location at the given time
   c. Calculated overall risk factor

II. Status diagram representation of the location's weather severity .

2.3 User Characteristics

Users that interact with this software are people with an Android device looking to make strategic decisions based on the weather prediction data that will result from searching a specific location. They should have general knowledge of city names and how to read maps. The application is designed to have a simple UI (User Interface) design allowing for users with various levels of technical expertise to operate the software with ease.

2.4 Constraints

I. **Regulatory policies**
   a) GPS modeling framework for displaying locations on the map. This ensures familiar user interface experience and clear borderlines for legal uses.
   b) Credible data set and citations for copyright and accurate display of information.

II. **Hardware limitation**
   a) Must meet the absolute minimum requirements of a 200 MHz processor, 32 MB of RAM, and 32 MB of storage to operate on Android
   b) Any version of Android above 4.4 (KitKat, Lollipop, Marshmallow, Nougat, Oreo, and Pie) requires an ARMv7 processor.

III. **Parallel operation**
   a) Operable under the availability of Internet connection due to the reliance on parallel searching and data fetching over online network connection.

IV. **Audit functions**

a) Not a revenue intended software.
b) No external audit will be used, financial statements will be solely produced by Travel Safe.
c) Evaluate the quality of internal controls and software management.
d) Review the effectiveness and security of information technology systems.

V. **Control functions**

a) The use of version control system through the project's development stage.
b) Quality control through code documentation and maintainability checks.
c) Control modules for data sets functionality and interfaces.
d) Monitoring software issues and version consistency.

VI. **Higher-order language requirements**

a) SQL commands for above queries and application.
b) Client/server system.
c) Application programs to construct a stable store that use procedures to implement the required tasks.
d) Textual user interface for displaying constraint violations.

VII. **Signal handshake protocols**

a) Supports an Android coding alphabet.
b) Medium fast data transfer rate.
c) Transport Layer Security to prevent shareable data.

VIII. **Reliability requirements**

a) Built in test and system diagnostics.
b) Design analysis of extreme usage conditions in the testing stage.
c) Data integrity checks.
d) Input and output validity testing.
e) Consistent maintainability schedules.
f) Records for testing failures and successes.

IX. **Criticality of the application**

a) Operational availability of the software is low.
b) Quality requirements fully satisfied.

X. **Safety and security considerations**

a) User notification upon detection of software error and network error.

b) System will record daily logs on a weekly basis.
c) Immediate system restart and no back up information with detection of illegal activities.

2.5 Assumptions and Dependencies

I. The software is used on the Android operating system.
II. It will be an app interface that is best used on mobile phones.
III. The software application will be used for personal travel of a single user.
IV. User understands how to represent locations geographically. Therefore, expect proper inputs from the user.
V. The user is familiar with using an Android app.
VI. Data set provides full coverage of the needed weather analysis results.
VII. Development process will follow a systemized timeline.
VIII. Internet connection is available.
IX. Low maintenance and testing records required after the development stage.

3. Specific Requirements

3.1 External Interfaces

3.1.1 User Interfaces
3.1.2 Hardware Interfaces
3.1.3 Software Interfaces
3.1.4 Communication Interfaces

3.2 Functional Requirements

3.2.1 User Class 1 - The User

3.2.1.1 Functional requirement 1.1

ID: FR1

TITLE: Download mobile application

Description: The user should be able to download the mobile application through Google Playstore or similar services.

3.2.1.2 Functional requirement 1.2

ID: FR2

TITLE: Search hazards by time and month

Description:

- Given a specific time and location, find all past hazards from the database for the given time period and display them in a map view.

3.2.1.3 Functional requirement 1.3

ID: FR3

TITLE: Rank times to when a specific location is less hazardous

Description:

- Given a specific location, find the top 3 best times when that location is less hazardous.
- The weight used to compare time periods is based on the number of total past hazards at a given month.
- Display the top 10 time periods in a list view.

3.2.1.4 Functional requirement 1.4

ID: FR4

TITLE: Provide the least hazardous path from one state to another.

Description:

- Given a specific month, source and destination states, find the safest path between them. The path will not reflect the actual physical routes but rather show which states to go through.

3.3 Performance Requirements

- *Response time Requirements*. The front-page load time of the app must be no more than 3 seconds and a loading icon should be provided indicating the software is not down. As users input their travel specifics (location, date, etc), the response of the software should be instantaneous under decent internet quality. The suggested travel routes and danger zone icon should be displayed at the same time with no delay from one to another and the whole process should be finished in approximately 1.0 second. A loading icon with a progress bar will be shown on the screen to provide the similar functionality as mentioned above. Moreover, there should not be a significant variation in response time when users input different travel specifics.
- *Memory Requirements*. It should take up no more than 10% of the RAM space of a typical android cell phone with 8GB memory.
- *Accuracy Requirements*. The output displayed through the Google Map API should follow a logical sense, meaning that markers should be displayed at the appropriate coordinates based on the inputted data.

3.4 Design Constraints

- *Language Constraints*. The software backend should be written in Java and the front end can be chosen from other different programming languages
- *Size Constraints*. The size of the software should be less than 80MB.
- *OS Constraints*. The software should run in any distribution (EMUI for instance) of the Android operating system.

3.5 Software System Attributes

Reliability: The software should complete 99% of its operations correctly without failure. When software failure happens, however, an error message would be displayed and the application continues from before the error occurred.

Availability: The deployment of a new module has a minimum effect on the software itself. The front page will always stay up and the updating process will not take longer than 5 minutes.

Maintainability: The section, 3.2 Functional Requirements, of this document demonstrates the great maintainability of the software, as the algorithms being implemented and modules are separated clearly. Any changes on algorithm or data structure can be handled easily.

Portability: The software has limited portability as it only correctly runs on Android-based devices and it also requires the device to have access over Google map services. Regions that can not access Google services would not be able to run this software correctly.

3.6 Other Requirements

3.6.1 Maintenance

In the interest of keeping the software simple to update and test developers will need to adhere to the following guidelines.

3.6.1.1 Documentation

All main code, excluding test drivers, will be documented per the Javadoc standard as provided in the Eclipse IDE. The goal is for colleagues to be able to make important changes to code modules without the need for input from the original author. Full functionality should be attainable without referring to code - documentation should be complete and thorough.

3.6.1.2 Version control

Code will be uploaded to a central repository using Git to facilitate collaboration within the group, as well as to support rollbacks if a faulty update is released. Group members will need to communicate with the rest of the group what they are currently working on, in order to prevent merge conflicts. This repository will also facilitate flexibility in work environments as programmers are free to download to whatever machine they choose.

3.6.1.3 Testing

The program will be tested using the Junit unit testing framework. Modules and their interactions with other modules will need to be tested thoroughly whenever they are added, or updated. Errors must be kept to a minimum in order to meet specified requirements.

4. Appendixes

Appendix A - Potential Features To Implement

GPS Integration - Include GPS connection to enable access to current location and display current relevant weather information of the area.

Favourite Locations - Have a menu of favourite locations so users do not have to input their desired location every time.