vSafe

Requirement Specifications Document

Andrew Balmakund, Namit Chopra, Brian Le, Brandon Duong
balmakua@mcmaster.ca, choprn9@mcmaster.ca, leb7@mcmaster.ca, duongb9@mcmaster.ca

McMaster University
March 17th, 2020

Table of Contents

Abstract

vSafe is an application that provides users with information for safer travel. The following document presents information regarding the requirements of the program. The domain of the program is discussed first. Details such as the stakeholders, their goals, etc, are all presented in section "The Domain." Functional requirements discuss the behaviour of the program. Short descriptions are provided underneath different sections of the program. The non-functional requirements section lists the qualities the team expects from the program. The requirements for the development and maintenance process are also provided.

The Domain

Application Domain

The goal of vSafe is to create a danger rating for all cities in North America, calculated through each city's history of natural, dangerous disasters. The plan is to at least cover earthquakes and storms in these danger rating calculations; however, the inclusion of others is not out of the question, as described in the specification for the Assign Danger Rating module. As a result of the implementation, it is expected for users to be able to search for any city located in Canada for an accurate estimation of whether or not the area is comfortably safe. This product is also striving for a clean and easy-to-use interface, and nearly 24/7 uptime, so that any everyday customer can easily get the most out of the product.

Another related goal intended to be reached is to appeal to a wide variety of stakeholders, including local residents, tourists, vacation-planning businesses, and travel providers (along with essentially the entire travel industry as a whole). Therefore, with this sleek interface, it is believed that heavy focus on the product's adaptability is also an important aspect to achieve as when it comes to having many users, it is inevitable they use many different points of access, such as smartphones, computers, etc. Vacation-planning businesses and travel providers may even want to import this product for use in the back-end of their websites. This is one good example of how these four stakeholders have their own goals and expectations of the final product, making it crucial that careful consideration is taken for each and every one of them.

Stakeholders

Local Residents: The goals of the locals of a specific city, or all Canadian residents in general, are all inherently the same. They can expect to have a reliable information source pertaining to their own area about whether or not an unfortunate disaster is likely. With this product, and of course, its accurate calculations, they can also expect to experience an increase in their comfortability and confidence in not only their own safety but their loved ones as well.

Tourists: For tourists, their goals also have a clear core focus on comfortability and confidence in their safety. Through the use of vSafe, tourists, and any other person looking to travel, can plan their trip appropriately without worry for any unexpected or life-changing events to occur. Another fact to consider is that the use of vSafe may actually create a boost in tourism due to the fact that many may be actively avoiding a well-deserved vacation due to the enormous planning hurdles, therefore tourists can expect to see a spike in this area as well.

Vacation-planning Businesses / Travel Providers: Vacation-planning businesses and travel providers mostly overlap each other in terms of their goals and expectations of vSafe. With this product, not only can they expect an influx of tourists for reasons mentioned before, but also expect safer returns as tourists can now more appropriately choose and plan their trips. However, their main goal is hopefully to implement vSafe into the back-end of their websites to be able to close off areas and their services for durations in which they have a high danger rating. Although it goes against profits, this would garner tons of respect from not only their customers but everyone around the world, making this a viable option to consider.

Functional Requirements

vSafe requires a user-interface, a controller to communicate the front end and back end, and needs to be able to search and sort out the data sets based on multiple fields at a quick rate. For the project domain, additional storage space is not a concern as the backend will be running off a desktop. Once the data can be searched efficiently, it will be used to make calculations of natural disasters occurring which will help create a Heat Map to display safe and dangerous neighbouring cities based on the user's input location.

User-interface

This module will display the input for the user to provide as well as the output.

Front-Back End controller

This module will be effectively communicating with the user interface and the back end system.

Visual Heat Map

This module will plot a heat map that showcases all neighbouring cities of the user's location input, and display these cities with coloured circles that represent the safety of them.

Search For All Safe Nearby Cities

This module will search find all connected components to a city and return it as a list. Each Node will have a weighted edge that will help determine the distance between nearby cities. Then it will process this list to determine which cities are safe based on their assigned Danger Ratings.

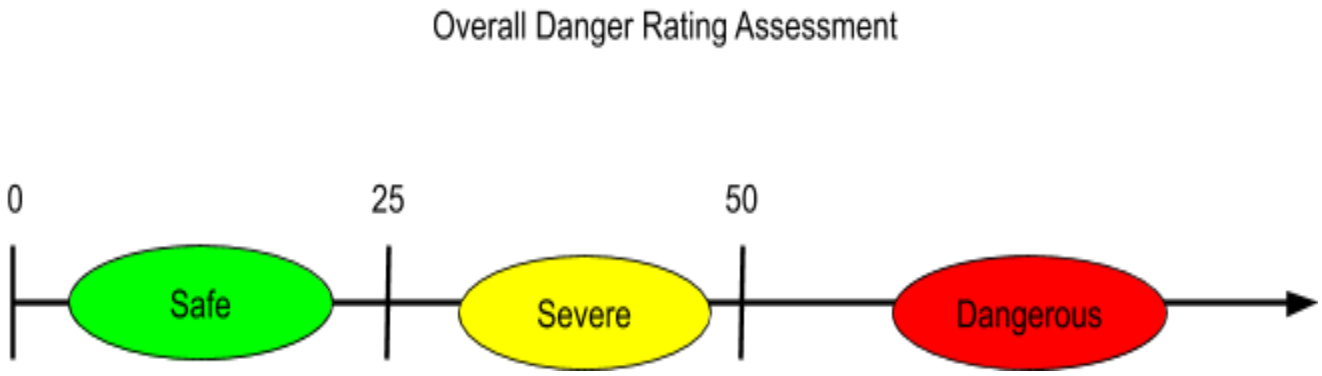
Graph Cities Construction (Linked-List)

This module will construct a linked-list representation of a map connecting nearby cities together with weighted edges. To achieve this construction, a data set containing the latitude and longitude positions of all cities in North America will be used to compute distance between nearby cities.

Assign Danger Ratings

Based on the calculated probabilities and magnitude measures of each natural disaster (which will vary from disaster to disaster), assign a specific danger rating to a specific city. This danger will be a cumulative value representing $\mathbb{N}$. Assessing individual disaster and accumulating a global danger rating for a city will go as follows:

Figure 1:



The following diagram demonstrates a cumulative global rating for a city and categorizes them either safe, severe or dangerous.

Calculate Probabilities

This module will use the search module to find all recurring natural disasters at a specific location and date, overall several years to determine the likelihood of it occurring again; calculating the average of repeating natural disasters.

Search Data Sets

This module will use binary search over a 2D string array that is sorted row-wise to search through all rows of a specific column for a string.

Sort Data Sets

This module will use sort on a 2D matrix array to sort through a specific column and return back a sorted 2D matrix. As mentioned before, storage space will not be an issue for this project because the backend will be running on a desktop.

Read & Store Data sets

This module will read any CSV files and store this data set by rows and columns in a 2D String Matrix.

Non-Functional Requirements

Along with meeting the functional requirements, some requirements ensure that the program meets the user's expectations. The requirements verify that the program obtains the quality set by the developers. The following non-functional requirements provide criteria for measuring the quality of the program:

Accurate Results

When the user enters the required information, city and date, it is expected of the program to produce correct results. As products in software engineering are different to assess, correctness is very hard to achieve. Despite the challenge, the developers expect and require the program produces the final result correctly 95% of the time.

Easy to use Interface

The user needs to input information regarding their trip to obtain the danger rating from the program. Users should not have a difficult time navigating through the software to obtain the information required. Making an interface that is intuitive and easy to navigate is necessary to increase the usability of the program by decreasing the effort required to use the program.

Portable across different machines and operating systems

There are a variety of different machines and operating systems used today. Having the users change their machine to run the program would not only decrease the portability but also the usability of the program as well. The software must run on a variety of machines and operating systems to maximize users. Increased portability allows the program to exist for a longer period due to its ability to run on different and newer machines.

24/7 Availability

The program is expected to be running and functional 24/7 unless there is an update or bugs that need to be fixed. Although the program only provides the danger rating for North American cities, those who are travelling to the continent from around the world should also be able to use the program. Due to the time difference around the world, it is logical to have the program available at all times of the day. To increase usability and maximize users, it is logical to have the program available all the time.

Strong Performance

The performance of the program is one of the biggest aspects that users look for. To ensure that the program meets the user's standards, the program must not crash and give an error to the user. If the user enters a misspelled city name or the name of a city is not in the database, it is not logical to output the error it produces in the code. It is required that the program must produce outputs that can be understood by all the users. Furthermore, users are more inclined to use a program that is fast and does not take most of their time computing the results. The program must output the results, including danger rating and graph, at most 5 seconds after the input is entered.

Requirements on the development and maintenance process

One of the team's goals is to ensure that the system makes the right decisions in response to various human inputs and makes correct judgements to simulate the decision making and recognition of weather conditions like a human would. To attain this final goal we can classify the requirements in the following way:

- *System test procedures.* The core of vSafe comes from the scale of decision making (assign danger rating module) as this is what will help users identify how they choose to go about their vacations. Through the test procedures, vSafe plans on calibrating the system's core functionality so that it best resembles the decision-making process of us and future users. From a series of test cases, vSafe plans on generating an ideal spectrum of what is considered to be safe and dangerous for vacation destinations that the average person would agree on. As shown in the overall danger rating assessment diagram and values above, developers plan on tweaking these ratings so that the system becomes scalable across the entire decision-making spectrum. Building on this, various test procedures will be testing the accuracy and consistency of the system's decision making by introducing different inputs and variables such as location, types of weather conditions, and user preferences.

- *Priorities of the required functions.* Core processes such as sorting, searching and data processing will be a major area of development for the system as this will act as the backbone for other features down the line which would help project information to the users. From there, priorities shift to user feedback. This includes any means of converting data that has been processed by vSafe's decision-making system to useful information that can be easily interpreted by the end-user. Such means of useful information would include worked numbers, graph implementation and descriptive responses by the system. Knowing the team's time and labour constraints, vSafe will prioritize graphing implementations and then transition to other various platforms of data representation should the team need to with the goal of developing a product which provides the right amount of information back to the user.

- *System maintenance procedures.* For vSafe, the team's biggest priority for maintenance is to refresh the data set on a consistent basis with new statistics and datasets which better suits the world as it is today. Factors such as global warming will result in statistics for weather conditions to constantly change and overtime vSafe's dataset will not properly reflect current weather conditions. Maintenance also revolves around user input/feedback as vSafe's decision-making scale will constantly be tweaked. People will also have different perspectives on specific weather conditions (i.e. Some wouldn't mind a location that has a lot of rainstorms for their vacation destination) and vSafe's system needs to account for this constant change.

Priority	Tasks	Description	EST. Date to Finish	Who
1	Find the probability of a natural disaster occurrence	To do this: Once the data has been stored locally and sorted, search for each instance of a year for a typical natural disaster along with its date. There can be instances where the data set will store data that cover several years. Might need separate modules for these	March 10th, 2020	Namit
2	Implementing Merge Sort	Modified merge sort that will read a 2D String matrix and only sort a specific column of that matrix (i.e Sort by location). Since Merge is an Inplace sort, sorting by location and sorting by date preserves the location and only sorts the date.	March 1st, 2020	Andrew
3	Download all data sets and upload to Gitlab REPO	Collect data sets and upload them to the repo.	February 27th,2020	Brandon
4	Reading CSV files	Read from a CSV file and store this in a 2D string array.	March 1st, 2020	Brian
5	Read JSON Files	Read from a JSON file and store this in a 2D string array.	March 1st, 2020	Brian
6	Construct a graph of cities	Create a Linked-List representation representing connected cities with weighted edges.	March 16th,2020	Brandon
7	Search for all nearby cities of a graph (Linked List)	Use a depth-first search to search for all connected neighbouring components of a city and return a list of all connected cities. The weights of the tree will help determine which city would be the closest.	March 20th, 2020	Brian
8	Implement Binary Search	Normal binary search.	February 27th, 2020	Brian

9	Assign Danger Rating	These danger ratings will be assumed by us to determine what we believe would be safe or not. Each disaster rating would vary from one to another. Once each city has accumulated a danger rating.	March 14th, 2020	Andrew
10	Plot the results from the graphing algorithm to create a “HeatMap”	This graph will graph all connected components into a heat map that will display a map focusing on the given city along with several nearby cities that will display circles surrounding the cities ranging from colours green, yellow, and red.	March 24th, 2020	Namit
11	Create a Front-Back end controller module	The controller will communicate with the user interface and the backend (see lecture week7)	March 26th, 2020	Namit
12	Create a FrontEnd-User Interface	Provides an interface for the user consisting at least three textboxes for entering location, date (dd/mm/yyyy to dd/mm/yyyy) and filters option	March 27th, 2020	Andrew