SOFTWARE REQUIREMENTS SPECIFICATION

Includes Summarization of Interview Results

EmergenSeek
Finding Assistance When You Need It

Prepared for

Texas Tech University - CS 4366
Senior Capstone Project

Professor: Dr. Sunlim Ho
Students: Suhas Bacchu, Derek Fritz, Kevon Manahan, Annie Vo, Simon Woldemichael

Wednesday, February 13, 2019

Contents

Revision History

Date	Reason For Changes	Delivery Date
February 5, 2019	Initial Creation	February 13, 2019

This LaTeX document is under source control management using Git.

The initial template for this LaTeX document was found on GitHub.
Source: https://github.com/jpeisenbarth/SRS-Tex

This LaTeX document follows the IEEE Recommended Practice for Software Requirements Specifications per IEEE Std 830™-1998(R2009).
Reference: http://www.cse.msu.edu/~cse870/IEEEXplore-SRS-template.pdf

1 Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the "EmergenSeek" mobile application. It will illustrate the purpose and complete declaration for the development of system. It will explain system constraints and interfaces. EmergenSeek will assist users in gaining access to emergency information and notifications. The application will also provide friends and family with realtime location-based information. The application will serve as a go-to utility for those that are traveling, both locally and abroad in the event of an emergency.

The intended audience of the document will be advisor's and stakeholders who will have procedural responsibilities throughout the development of this application. This document should be read in the order that it is presented. At the end of this document, intended readers should evaluate summarized interviews and take into account how system features and details of this specification are correlated to said summaries.

1.2 Project Scope

As the timeframe for this project is 3 months, developers shall consider the scope of this project to produce a functional, minimum viable product (MVP) which covers important usecases, functional requirements, and non-functional requirements.

These MVP components include, (1) functional use of an S.O.S. button which will automatically notify family members and/or emergency services depending on 2 degrees of emergency severity, (2) the ability to search for nearby hospitals, pharmacies, and other facilities that will assist the user in a time of emergency or distress, (3) location based polling which will broadcast metadata regarding the users current location to their primary contacts, friends, and family members, and (4), paired with (3), the ability to granularly define what permissions contacts have for the purpose of giving the user control over what private information they choose to share and with whom. Additionally (5), health and wellness information pertaining to the user shall be display on the lock screen of the device for the convenience of first responders and family members.

Assumptions made in creating the MVP will ensure that the scope and size of the project is not too broad. Allowing the application to be generalized to only the specifics of emergency situations and notifications provides the developers with a much more specific set of features to implement.

Software products used include the Go programming language, which will serve as a backend API, React Native, which will serve as our single source, cross-platform mobile

application framework, and a suite of Amazon Web Services to be used as utilities in driving the development of the program, as well as providing necessary tiers for the application. For instance, we will utilize DynamoDB as the database tier and Lambda and API Gateway as the application platform; formally known as a Functions-as-a-Service (Faas) offering.

1.3 Definitions, acronyms, and abbreviations

Definitions, acronyms, and abbreviations will be added to this section as they arise. Single use acronyms and abbreviations will be enumerated in-line, within the context that they are used.

1. API - Application Programming Interface - An abstraction, provided by a developer or exposed to a developer for the sake of simplifying front-end and downstream programming and implementation.
2. S.O.S. - A common distress code usually mistaken for "Save Our Souls." While the letters have no meaning, they are an alphabetical representation of the distress code's Morse code equivalent.
3. MVP - Minimum Viable Product - The bare minimum system implementation required for the application to be released and usable by end-users.

1.4 References

Note: This is not a comprehensive list, but should be review in pair to section 2.1.4 Software Interfaces. Amendments to this section will be made by due diligence as additional, paramount sources of reference are utilized, as necessary.

a. Golang - Google's statically typed, compiled programming language
   i. Language Specification https://golang.org/ref/spec
   ii. Documentation Generation - GoDoc - https://godoc.org/-/about
b. React Native - A JavaScript, cross-platform mobile application framework
   i. Framework Specification - https://facebook.github.io/react-native/docs/getting-started.html
   ii. ECMAScript Specification - http://www.ecma-international.org/ecma-262/6.0/1
c. Amazon Web Services
   i. Documentation https://docs.aws.amazon.com/index.html
   ii. DynamoDB - https://aws.amazon.com/dynamodb/
d. API architecture

i. OpenAPI Specification - https://github.com/OAI/OpenAPI-Specification

ii. HTTP/2 - Version 2 of the Hypertext Transfer Protocol per RFC7540 - https://tools.ietf.org/html/rfc7540

iii. gRPC - Google's custom Remote Procedure Call framework - https://grpc.io/

iv. Protocol Buffers - Fast binary serialization and program definition language - https://developers.google.com/protocol-buffers/

1.5 Overview

The rest of this Software Requirements Specification document will describe functional and non-functional requirements of the platform, constraints of various interfaces, and stimulus-response assumptions. Later revisions of this document may include pricing and marketing strategies to monetize and scale the application. This document will not contain design documents, as Project 3 is defined as a Software Design Specification. API specifics and implementation details will also be left out of this document.

Finally, the latter end of this document shall include summarized results of user interviews that were conducted. Data for these interviews was collected using a collaboratively generated Google Form.

2 Overall Description

2.1 Product Perspective

Similar implementations of this application that we have researched are *Life360* https://www.life360.com/, *Guardly* https://en.wikipedia.org/wiki/Guardly, and *In Case of Emergency* https://en.wikipedia.org/wiki/In_Case_of_Emergency. In comparison to these applications, our application serves a very similar purpose, but will integrate a wider range of emergency assistance features. For instance, the latter application is tailored more towards storing contact information, whereas our application will also provide that feature, in addition to several dynamic, location-based additions which are enumerated in Section 3 : Specific Requirements.

2.1.1 System Interfaces

Because this is a mobile application, our application will conform to the requirements of running mobile applications through interfaces that have already been well defined and exposed for us as developers. Put clearly, because we are using popular mobile development frameworks and would like to develop and application that will run, cross-platform, on the top 2 most popular mobile operating systems (Android and iOS), we need not take any additional precautions or development considerations in order to ensure that we are able to run our application on modern day, mobile devices.

As our backend API is a separate entity from our mobile client, it shall too need to conform to some standards in-order to be hosted in a production state. Again, because we are provided friendly interfaces and abstractions on top of our deployment provider, Amazon Web Services' API's in this case, no additional consideration outside of the bare minimum requirements to run our projects on Amazon's platform needs to be taken.

2.1.2 User Interfaces

Below, the word "training" in the context of the user shall refer to the reading of user documentation or demonstration videos, by the user. A general rule of them when developing application interfaces and mobile applications is to follow expected design schemes and practices. For instance, the first screen that the user shall see when loading the app will be a login/authentication component. Clicking on a `x` button within the application shall close or remove a component. Following best practices will ensure that everything within the application is intuitive.

1. Interfaces of this platform shall be intuitive and familiar to the user.

i. The user shall be able to login and logout in less than 2 seconds with no training.

ii. The user shall be able to recover their account in less than 10 seconds with no training.

iii. The user shall be able to setup their personal privacy filters and primary contacts in less than 1 minute with no training.

It is also important to note, because of time constraints we will not take into consideration accessibility necessities and features which will assist end-users with visual or motor impairment. While this is not an assumption of the target demographic, as we did note that anyone would be able to use this application, it is something that would most definitely be implemented, outside of the scope of our plan for this course. For instance, visual accessibility considerations may be implemented by holding physical buttons on the device to perform automated actions (i.e. hold the volume-up button for n seconds to trigger the S.O.S. feature instead of opening the application and navigating to the feature).

2.1.3 Hardware Interfaces

This subsection shall correspond to the same specifications as the System Interfaces section. Because the implementation of the project involves no external hardware interfaces (i.e. single board computers or microprocessors) we may develop all software and system interfaces without thinking about low-level programming languages or hardware details.

2.1.4 Software Interfaces

All third-party software interfaces used for development and implementation shall not be be in an End-of-life (EOL) state. This means, they shall actively supported by their original developing organization and in a Long Term Support (LTS) state.

While all additional dependencies used for specific software interfaces (i.e. programming languages) will not be documented, their usage shall conform to the open-source licenses that they are packaged with. For instance, in ordinance with the specifications of The GNU General Public License v3.0 (GNU GPLv3 - https://www.gnu.org/licenses/gpl-3.0.en.html) there shall be no closed-source or internal distributions of amendments to third-party software dependencies which are packaged with this license.

Name	Version Number	Source
Golang	v1.11	https://github.com/golang/go
AWS SDK for Go	v1.16.3X	https://aws.amazon.com/sdk-for-go/
Node.js	v10.15.1	https://nodejs.org/en/
React Native	v0.58.X	https://github.com/facebook/react-native
Twilio API	Managed	https://www.twilio.com/docs/libraries
Google Location Services	Managed	https://cloud.google.com/maps-platform/
AWS CodeBuild CI/CD	Managed	https://aws.amazon.com/codebuild/

2.1.5 Communications Interfaces

As this application is intended to be a network connected mobile application, it shall communicate over HTTP/2 (HyperText Transfer Protocol version 2), and WSP (The WebSocket protocol). Protocols utilized in underlying dependencies, libraries, or by cloud providers (AWS) will not be documented in this SRS.

2.2 Product Functions

The product will provide the following functions. Functions are listed in order of priority and correspond to the system features that are discussed in Section 3 : Specific Requirements.

1. S.O.S. button emergency broadcast
2. Emergency service locator
3. Periodic notifications to contacts (location-based polling)
4. Granular permission definitions for contacts
5. Lock screen display of health information for emergency services

2.3 User Characteristics

Users may be anyone of any age, educational level, or technical background. As the application is meant for the betterment and piece of mind of all, there are no assumptions or minimizations on the demographics or target audience of the application.

2.4 Constraints

Given that time is the largest constraint on our project, the only additional constraints that will come into consideration are those that abide by formally defined security and privacy definitions for the sake of end users. Regulatory policies such as the General Data Protection Regulation (GDPR) will ensure that the application and its components do not retain sensitive location information, that can be easily related a user, for "no longer than necessary". Under article 12.3 of the GDPR, the application shall also, in application development phases beyond the scope of this course and MVP, recognize legitimate erasure requests within the specified 30 day time limit. An official, sectioned copy of the regulation documentation may be found at the following url: https://gdpr-info.eu/.

Because our application utilizes modern, performant software interfaces, we will be able to achieve our goals of speed, reliability, and security without much additional work. See sections following section 3.1 on specific specifications of these goals.

2.5 Assumptions and Dependencies

Changes to this SRS document are dependent on the requirements or desires of stakeholders, customers, developers, and advisors.

3 Specific Requirements

Part 2 : Functions of this section shall detail, through grouping, *general* stimuli and responses that the application and developers will expect. Following, will be stimuli-response pairings specific to individual system functions. It is important to note that trivial, expected features (features which are a given) are not considered features specific to our system. For instance, registering and logging in users is a trivial, and obvious necessity.

3.1 External Interfaces

1. Android and iOS Mobile Device
    i. *Description of purpose;* Operating systems which our cross-platform application will compile runnable binaries for. The main interface in which the user will interact with.
    ii. *Source of input or destination of output;* Devices running these operating systems are assumed to have touch screen interfaces. As such, inputs will be read from the touch screen, and outputs displayed to the touch screen.
    iii. *Valid range, accuracy, and/or tolerance;* No consideration will be made to account for the sensitivity, working state, or reliability of the mobile device. It is the responsibility of the user to have a device which conforms to the hardware requirements necessary for running these operating systems. (i.e. compatible processors with ample RAM and disk space).
    iv. *Units of measure;* Seconds; used to determine the responsiveness of the interface components maintained and developed by project developers. For instance, no blocking actions shall occur to ensure that the interface may accept and display inputs and outputs with minimal latency.
    v. *Timing;* Frames-per-second (FPS); Cross-platform mobile components maintained and developed by project developers and any additional animations shall conform to the standard 60 FPS supported by modern smartphones running these operating systems.
    vi. *Data Formats;* The application shall, through the abstracted implementation of our cross-platform application's compiled, runnable binaries, conform to the operating system communication protocols necessary to make the app functional. This is another implementation detail that project developers
2. Amazon Web Services (AWS)

i. *Description of purpose;* AWS shall be the platform, storage tier, and database tier for our application.

ii. *Source of input or destination of output;* The mobile client shall send JSON messages to AWS when performing API requests and receive JSON messages to display the results of said requests.

iii. *Valid range, accuracy, and/or tolerance;* Consideration for timing, accuracy, tolerance, and reliability of AWS shall not be considered as developers are not responsible for maintaining the underlying interfaces that power AWS.

iv. *Units of measure;* Milliseconds; used to determine the speed at which data is exchanged (request-response feedback). For instance, all JSON requests shall be responded too with the proper JSON response within 10 milliseconds.

v. *Timing;* Uptime of AWS will again not be taken into consideration as the reliability of AWS can be further mitigated by the replication of application instances across several regions within the Virtual Private Cloud.

3.2 Functions

General Stimulus-Response sequences

• **Stimulus:**
The user will open the application for the first time.
• **Response:**
The system shall direct the user to the account registration screen.

• **Stimulus:**
The user will select either the Google or Facebook log-in buttons.
• **Response:**
The system shall communicate with the Google or Facebook API to retrieve and store user information.

• **Stimulus:**
The user will select the emergency service locator.
• **Response:**
The system communicate with the Google Maps API to retrieve and display the locations of nearby emergency services.

• **Stimulus:**
The user will change the filter options for the emergency service locator.
• **Response:**
The system shall make a new API call to reflect the modified search criteria.

• **Stimulus:**
The user will select the contacts list.
• **Response:**
The system shall request the user to import contacts from Google, or to manually create a new contact.

• **Stimulus:**
The user will select a contact to import.
• **Response:**
The system shall communicate with the Google Contacts API to parse the desired contact's information and store it as a new contact associated with the user.

• **Stimulus:**
The user will select manual contact creation.
• **Response:**
The system shall redirect the user to the contact creation page with the required fields to be completed.

• **Stimulus:**
The user will select the S.O.S. alert.
• **Response:**
The system shall prompt the user to specify the level of the emergency situation.

• **Stimulus:**
The user will select the mild emergency option.
• **Response:**
The system shall broadcast an alert to the contacts registered to receive minor emergency alerts.

• **Stimulus:**
The user will select the severe emergency option.
• **Response:**
The system shall broadcast an alert to the contacts registered to receive severe emergency alerts, as well as communicate with the Twilio API to send an automated call to emergency services.

• **Stimulus:**
The user will select the location update options.
• **Response:**
The system shall redirect the user to the options page for customizing location update notifications.

• **Stimulus:**
The user will enable location updates.
• **Response:**
The system shall periodically send location updates to the contacts registered to receive the notifications.

• **Stimulus:**
The user will select the emergency information page.
• **Response:**
The system shall retrieve the user's location via the device's GPS and display stored emergency information associated with the user's location.

Functional Requirements

1. The system shall allow users to sign-in using their Google account.
2. The system shall allow users to manually create contacts.
3. The system shall allow users to import contacts from their Google account.
4. The system shall display the locations of emergency services within a 5 to 10 mile radius of the user.
5. The system shall allow users to filter the selection of nearby emergency services.
6. The system shall send periodic location updates in accordance to the user's settings.
7. The system shall broadcast emergency alerts upon S.O.S. activation in accordance to the user's settings.

Non-function Requirements

1. The system shall transition between menus in under 5 seconds.
2. The system shall allow users to log in in under 10 seconds.
3. The system shall continue to function without being impacted by API communication failures.
4. The system shall handle system failures through multi-instance replication via AWS.

3.2.1 System Feature 1 - S.O.S. Button

Introduction

This is a very high priority feature. This feature will provide users, depending on the severity of their emergency, the ability to notify emergency services and/or their primary contacts.

Stimulus/Response sequence

• **Stimulus:**
The user is in an emergency situation.
• **Response:**
The user shall open the application and determine if they are in a `Severe` or `Mild` emergency.

• **Stimulus:**
The user shall select the `Severe` or `Mild` button.
• **Response:**
If the user selected `Severe`, then the system shall automatically call emergency services and notify the users primary contacts via SMS. If the user selected `Mild` then the system shall notify the users contacts via SMS and display emergency service locations (i.e. Hospital's and pharmacies) that are within a 5 to 10 mile range.

3.3 Performance Requirements

1. The server of any component for the platform shall be able to receive at least 20,000 requests per second.

2. The server of any component for the platform shall be able to send at least 20,000 valid responses to the client per second.

3. The system shall be fully concurrent, performing no blocking I/O tasks.

3.4 Design constraints

1. As the system relies on frequent communication with various API's, the system's responsiveness and reliability depends heavily on the device's connectivity to the internet.

2. Per the previous bullet, consideration should be taken for realtime updates and offline state management on the mobile client. This way, the mobile client can still queue data that it would like to send as soon as it is reconnected to a reliable network and can additionally receive updates in realtime.

4 Interview Summarization

4.1 Introduction

In this additional section 4, originally not apart of the recommended IEEE Std 830 specification, we shall summarize the findings of our analyses of potential target users. To perform this, we used a Google Form. The form contained a total of 3 sections and received a total of 30 responses. Physical quantities and graphs are left out of this document for brevity, but may be delivered upon request.

The first section contained questions which would allow us to analyze the user being studied. This includes information such as their age, level of technical ability, and other demographical information. Of the 30 users, we found the following characteristics present in the data collected:

1. More than half of the users were within the 18 to 30 age group.
2. There was an almost even distribution in the gender, with slightly more males.
3. Only a quarter of participants had children.
4. More than 3 quarters of participants have been abroad.
5. More than 90% of users consider themselves technologically literate.
6. **100%** of participants *regularly* carry a smartphone around with them.

Section's two and three of the form were used primarily to asks questions regarding the application. We split these sections into topical and practical analyses.

4.2 Topical Analysis - Emergency Preparedness

In this section we analysed the emergency preparedness amongst participants, both local and abroad.

From the answers collected addressing what the first thing is that people would do in an emergency situation, we have determined that the SOS button feature of our application would be of use to 90% of survey participants.

Most of the individuals we interviewed do not travel on a regular basis, but rather travel occasionally.

90% of users also chose, before traveling, to keep important information regarding their trip or travel destination on their mobile phone. The feature of providing all of the

location-based emergency information for a user would be of great use to address this need.

We have also found the need for an additional feature, an emergency service page. This feature would include information searched before-hand, emergency phone numbers, but may not be included in the MVP of version of our application.

4.2.1 Practical Analysis - Application Features

In this section, we asked the participant questions regarding their sentiment and opinion on the usefulness and demand for emergency-assistance-centric applications.

What is the thing(s) you think of when people introduce you an "Emergency App"? Select all that apply.

Heavy emphasis was placed on calling emergency services than we had previously anticipated

If you were in an emergency situation how far would you be willing to go to find the assistance and resources that you need?

Most users prioritized traveling a shorter distance, as it would not be logical to travel a far distance when in an emergency situation. This fell in line with our expectations.

If your loved one was traveling abroad or very far from home, how often would you expect them to contact you or check-in with you?

We anticipated that most people would want to hear from their loved ones more often than once a day. This did not fall in line with our expectations as people would be okay with keeping contact as infrequent as once a day. Although most people said once a day, the results were still somewhat skewed with large portions also wanting more frequent and less frequent locations. This justifies a feature which allows users to set different tiers of location polling.

If you could know, automatically, where your loved one was, how frequently would you want an up-to-date location?

Upon specifying that location polling would be performed automatically on behalf of the user, more people expressed an interest in receiving more frequent location updates. There was also a presence of consideration on how intrusive the method of automated notification would be. We will incorporate this into our design choices when developing location-based updates. Overall, the answers were consistent with how often users would expect their loved ones to manually contact them.

What information would you like your loved ones to know in an emergency situation? Select all that apply.

The selected information that the user's loved ones would need to know coincided with our expectation that location is the most pressing information that one would want to receive, followed by blood type.

What travel information would you like to see about the country you are visiting? Select all that apply.

Again, this section's selections support the need for our feature of nearby emergency services as the distribution of answers is almost equal. The need for including the location of one's home country's embassy could also be included, given the location is

within the range of the user;s current location. With this, we found that most people would make use of a comprehensive list of emergency numbers for foreign locations (i.e. police, hospitals, embassies) as we cannot assume that all of these services are accessible through the same emergency phone number.