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

1 Introduction **4**
- 1.1 Purpose . . . 4
- 1.2 Project Scope . . . 4
- 1.3 Definitions, acronyms, and abbreviations . . . 5
- 1.4 References . . . 5
- 1.5 Overview . . . 6

2 Overall Description **7**
- 2.1 Product Perspective . . . 7
  - 2.1.1 System interfaces . . . 7
  - 2.1.2 User Interfaces . . . 7
  - 2.1.3 Hardware Interfaces . . . 7
  - 2.1.4 Software Interfaces . . . 8
- 2.2 Product Functions . . . 8
- 2.3 User Characteristics . . . 8
- 2.4 Constraints . . . 8
- 2.5 Assumptions and Dependencies . . . 8

3 Specific Requirements **9**
- 3.1 System features . . . 9
  - 3.1.1 System Feature 1 - SOS Button . . . 11
- 3.2 Performance Requirements . . . 12
- 3.3 Design constraints . . . 12

4 Appendices **13**

5 Index **14**

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 application will be to develop a cross-platform mobile application which will assist users in gaining access to emergency information and notifications. The application will 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.

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 those 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.

i. 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.

ii. 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.

1.4 References

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-262/6.0/0

b. 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/

c. File encoding

   i. UTF-8 - https://www.fileformat.info/info/unicode/utf8.htm

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

2.1.1 System interfaces

Because this is a web based platform, the system shall operate following standard web usage specification. There should be no limitation in what web browser is used or how powerful the system accessing the service is.

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.

2.1.3 Hardware Interfaces

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.
All third-party software interfaces.

Name	Version Number	Source

2.2 Product Functions

The product will provide the following functions. Functions will be added or removed as development progresses. Functions are listed in order of priority. Functions are enumerated in detail in section 3 of this document.

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

2.5 Assumptions and Dependencies

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

3 Specific Requirements

3.1 System features

Stimulus/Response sequence

• **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 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 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.

3.1.1 System Feature 1 - SOS 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

Functional Requirements

1. The system shall display the status of all of it's child micro-services.
2. The system shall keep a log of system uptime.
3. The system shall notifiy developers and support staff via email and SMS when any child micro-service goes offline, within 5 seconds of the service going offline.

Non-functional Requirements

1. Customers shall be able view if a micro-service is online or offline in less than 1 second with no training.
2. Customers shall be able to notified by support staff about system downtime or lack of speed in less than 1 hour with no training.
3. Customers shall be compensated within 24 hours, if the platform is offline for more than 4 hours.

3.2 Performance Requirements

- The server of any component for the platform shall be able to receive at least 20,000 requests per second.
- The server of any component for the platform shall be able to send at least 20,000 valid responses to the client per second.
- The system shall be fully concurrent, performing no blocking I/O tasks.

3.3 Design constraints

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.

4 Appendices

5 Index