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 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.

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 SOS 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.

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 use 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 DynamoDB as the database tier and Lambda and API Gateway as the application platform; formally known as a 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.

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. DocumentDB - 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

This product has two key contexts.

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:**
Customer will click a register button on the front-end of the web interface for the platform.
• **Response:**
The system shall direct the user to the account registration screen.

• **Stimulus:**
Customer will submit a registration form on the front-end of the web interface for the platform.
• **Response:**
The system shall create an initial account for the customer.

• **Stimulus:**
Customer will click a button in the options section of the platform to request an API key .
• **Response:**
The system shall generate, store, and display a client ID and API key for the customer.

• **Stimulus:**
Customer will click a login button on the front-end of the web interface for the platform.
• **Response:**
The system shall prompt the customer for their registered email and password and log the customer in to access services.

• **Stimulus:**
Customer has forgotten their password. The customer shall click a forgotten password option on the front-end of the web interface for the platform and enter their email.
• **Response:**
The system shall email the customer a link to reset their password.

• **Stimulus:**
Customer has an API key and makes an HTTP request using an interface external to the web interface for the platform.
• **Response:**
The system shall respond with the requested data for the specific target micro-service of the HTTP request.

Functional Requirements

1. The system shall utilize standard HTTP encryption practices.
2. The system shall not result in any encryption issues, after an infinite number of request-response transactions.
3. The system shall provide a database of all registered users.
4. The system shall provide a database of all subscribed users.
5. The system shall keep a log of all generated API keys.
6. The system shall track the number of authenticated requests made by an API key.
7. The system shall require customers to be logged in to use any micro-service.

Non-function Requirements

1. Customers shall be able to make accounts accounts in less than 1 minute with no training.
2. Customers shall be able to verify the email address associated with their account in less than 1 minute with no training.
3. Customers shall be able to generate API keys for the platform in less than 10 seconds with no training.
4. Customers shall be able to login to their account in less than 1 minute with no training.

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

4 Appendices