6-15-2018

SpotMe Emergency Location Service

Arya Faili
Santa Clara University, afaili@scu.edu

Zain Umerani
Santa Clara University, zumerani@scu.edu

Kunal Bhimjiyani
Santa Clara University, kbhimjiyani@scu.edu

Santa Clara University
DEPARTMENT of COMPUTER ENGINEERING

Date: June 15, 2018

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

Arya Faili, Zain Umerani, Kunal Bhimjiyani

ENTITLED

SpotMe Emergency Location Service

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

AND

BACHELOR OF SCIENCE IN WEB DESIGN AND ENGINEERING

[illegible]

THESIS ADVISOR

[illegible]

DEPARTMENTCHAIR

SPOTME EMERGENCY LOCATION SERVICE

by

Arya Faili, Zain Umerani, Kunal Bhimjiyani

SENIOR DESIGN PROJECT REPORT

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
and
Bachelor of Science in Web Design and Engineering
School of Engineering
Santa Clara University

Santa Clara, California

June 15, 2018

SpotMe Emergency Location Service

Arya Faili
Zain Umerani
Kunal Bhimjiyani

Department of Computer Engineering
Santa Clara University
June 15, 2018

ABSTRACT

This document delves into our disaster relief application that allows people who are helpless due to a natural disaster to find a way out and get help. The purpose of this document is to explain how the application works, but more specifically the design of the application, use cases, and conceptual models. Starting with a brief introduction, this paper will dive into the necessary requirements needed to build an application at this scale while presenting several use cases. To help the reader understand the application at a finer detail, activity diagrams will be shown along with models. Lastly, the document will cover what technologies will need to be used as well as a test plan and risk analysis.

Acknowledgements

We would like to thank the following individuals for their help in building SpotMe Emergency Locator:

- Dr. Silvia Figueira of Santa Clara University for her guidance, support, and advising
- Mr. Allan Baez Morales of Santa Clara University for his guidance, mentorship, and enthusiasm for our project

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Natural disasters often leave devastation that no one can prepare for. Hurricanes bring powerful winds that can level buildings, topple infrastructure, or cause catastrophic flooding. During these events, communities are often told to evacuate and while that may be the safest option, sometimes it is not possible for certain individuals or households to leave their homes behind. Road congestion, fear of property damage, health issues, or lack of resources can force people to stay behind and endure. Island communities are in even more danger as they may not even have the option to evacuate. In these situations, emergency responders will often be overwhelmed or unable to help until after the storm has passed, leaving people stranded or trapped in their homes, shelters, or other locations. Electricity, internet, and other communication infrastructure often fail during natural disasters, and those who are trapped following a disaster may have no way of communicating with rescuers in order to inform them of their critical situation.

1.2 Current Solutions

Current solutions do not offer an intuitive method for users to rapidly reveal their whereabouts to first responders prior to a large natural disaster. Dialing 911 is usually seen as the only option, but with a tremendous number of inquiries following a disaster, help may take hours if not days to arrive. Good samaritans are also left out of the loop when using 911. Many times after natural disasters, citizens not involved with law enforcement or search and rescue lend a hand to find trapped or stranded families. Reporting a location to 911 alone will not inform good samaritans as to a location of as stranded individual or family. Apps exist that allow users to report emergency situations without the need to dial 911, but their user interface is often complicated and requires many steps in order to actually use the app. The Guardly app, for example, is not designed with urgency in mind. The user is required to log in and is then presented with a screen to view global emergencies around the world, and then further presented with more options of the severity of the emergency, etc. As the user requires immediate attention in or following the event of a natural disaster, the user should not have to input many options or spend valuable time navigating within the app before asking for help.

There is no app designed explicitly for natural disasters which also provides a one touch solution to enable stranded individuals to ask for help within seconds. Our solution is the simple one touch solution. Upon tapping ”I need help”, a virtual pin is dropped on a map which would automatically contain the user’s information including name, address, phone number, and IMEI (obtained through his or her contact application on the phone) and location (obtained through GPS). The IMEI number is essential as that would enable mobile service providers to further track the phone during the disaster. This one touch rescue feature exposes all the necessary information of the user in order to attend to his or her emergency immediately. This is all done without asking the user to input unnecessary options beforehand and confusing him or her with irrelevant views within the application.

1.3 Our Solutions

Our solution involves both a mobile as well as a web component. For the mobile part, users can post their locations (”hotspots”) which will then be displayed on a map (where the disaster is). The location along with some metadata will be sent to a database where the web component will pull that data and display the various hotspots as well as a severity level (denoted by color). However, before the user opens the app, the app will check whether the user is within a disaster zone (a set radius). If the user is within the radius, only then will he/she be able to post their location. Anyone who is willing to help (citizens, rescuers, etc.) can view the locations of those in need and plan accordingly. Lastly, if no cell coverage is present, the user may also default to sending text messages to nearby rescuers. This allows the user to post their location in any situation where they are able to acquire some sort of data connection.

2 Requirements

2.1 Functional Requirements

- The system will allow the users to drop their personalized pin on the disaster map to enable others to determine their location and rescue them.
- The system will allow the users to customize their details (phone number, location, medical card, etc.) as they would like on their dropped ”pins” on the disaster map.
- The system will allow the users to view all the pins dropped by other users within their area during the time of emergency.
- The system should only allow the user to drop a personalized pin if an emergency has been declared in their area.

2.2 Non-functional Requirements

- The system will have a visually appealing UI which is simple to navigate.
- The system will be easily maintainable with modular components.

2.3 Design Constraints

- The solution will be a web based component as well as a mobile based component (running on both iOS and Android).
- The solution should work in areas of low network connectivity.

3 Use Cases

Figure 1 and the descriptions following explain how first responders and victims will use the application.

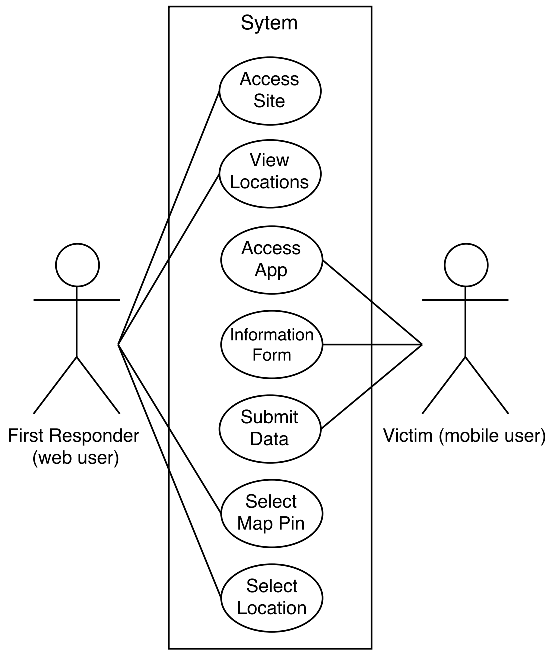


Figure 1: Use Case Diagram

1. **Access Site**

   Goal: To successfully access the website

   Actors involved: First responders

   Pre-condition: Have Internet access

   Steps: Enter URL for the website

   Post-conditions: User is now able to access the website data to locate victims

   Exceptions: None

2. **View Locations**

   Goal: Gain access to the list of locations currently under a state of emergency

   Actors involved: First responders

   Pre-conditions: Have access to the website

   Steps: Navigate to link on navigation pane labeled ’locations’

   Post-conditions: User can now view areas under a state of emergency

Exceptions: None

3. **Select Location**

   Goal: To allow the user to select a given area under a current state of emergency declaration

   Actors involved: First responders

   Pre-conditions: User is on the locations page showing impacted areas

   Steps: Click on the desired location (link) to a map page

   Post-conditions: User is now shown a map of the inflicted region, pins represent victims who have shared their location

   Exceptions: Certain locations that are not under state of emergency will not be available to the user

4. **Select map pin**

   Goal: To give the user information about a the selected victim(s)

   Actors involved: First responders

   Pre-condition: User must have navigated to a valid location on the website that is under an active state of emergency

   Steps: Select a pin to reveal more detailed information such as directions and contact information

   Post-conditions: The user can now see key data such as the victims phone number, GPS coordinates, description of the location (if available), severity of the situation, and the number of people in need of assistance

   Exceptions: None

5. **Access App**

   Goal: To give the user (potential victim) the ability to input their information and location in the system in the event of an natural disaster/emergency

   Actors involved: Victims

   Pre-condition: User must have a compatible mobile device (android/iOS)

   Steps: Using the app store, search for the mobile app and install

   Post-conditions: The user now has the ability to share their location in the event of a natural disaster

   Exceptions: None

6. **Input Location/Submit**

Goal: Allow the user to share their location with first responders and good samaritans following a natural disaster.

Actors involved: Victims

Pre-condition: User must have the mobile app installed on their device.

Steps: Open the app, the app will collect the user's location and if a state of emergency has been declared in the area, the user will have the option of inputting their location and details about their situation in the app for first responders to see.

Post-conditions: The user has now shared their location with good samaritans and other rescuers.

Exceptions: State of emergency has not been declared

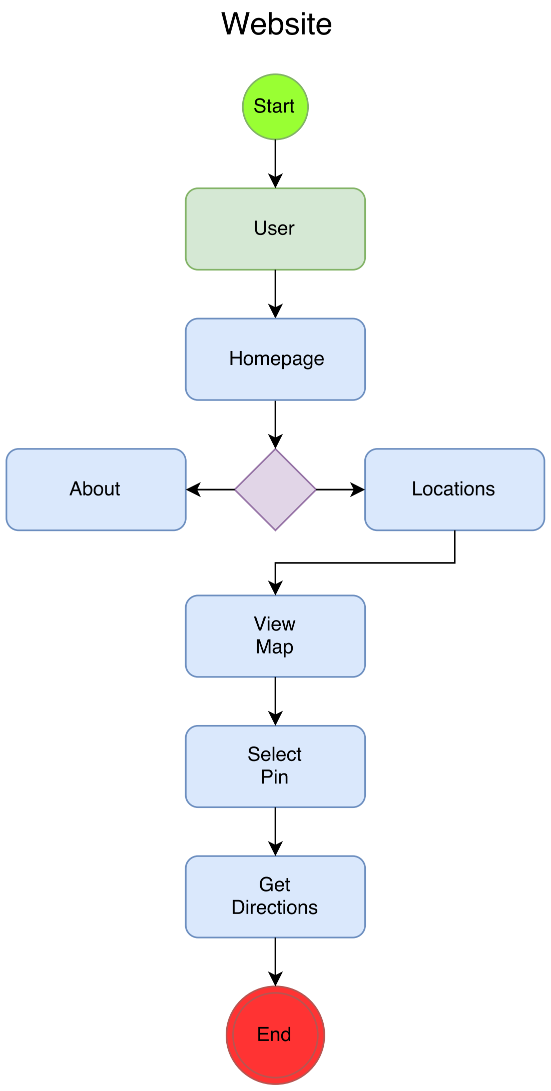
4 Activity Diagram

4.1 Website

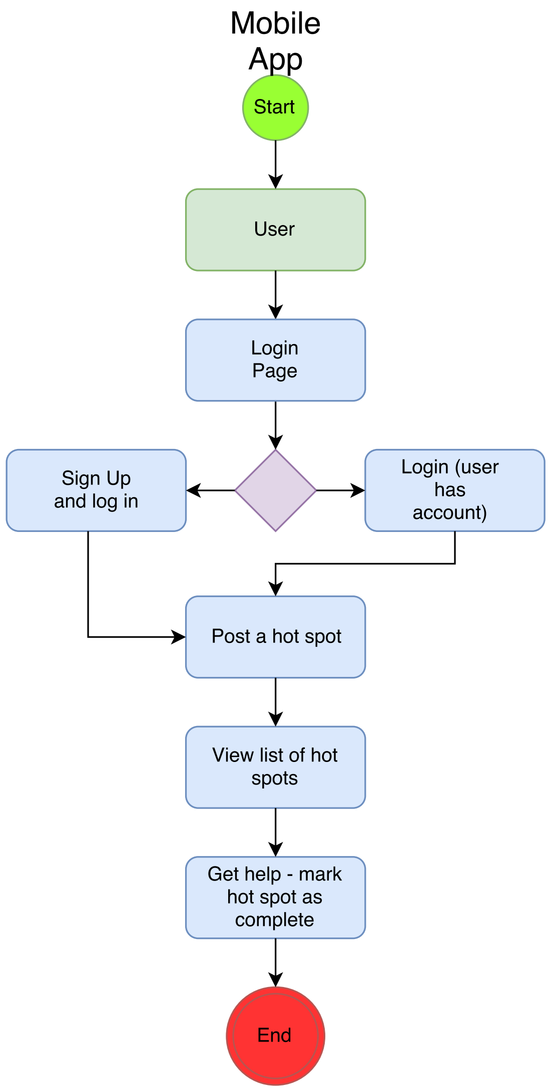
Figure 2a includes steps on how a first responder will use the website in order to view stranded victims. The user will land on the homepage, where they can then navigate to the locations page using the navigation bar. After the user has selected a valid location (state of emergency must be declared), the user can then view the map of the region. If there are any victims who have submitted their information to through the mobile app, they will appear on the map and the user can select their pin to view additional information, as well as get directions to the victims location.

4.2 Mobile App

Figure 2b includes steps on how a mobile user (the victim) will use the application. He or she will first need to authenticate themselves. After that, the user can post a hot spot and view a list of hot spots within their region. Once the user gets help, the hot spot is then removed from the region.



(a) Activity Diagram Web

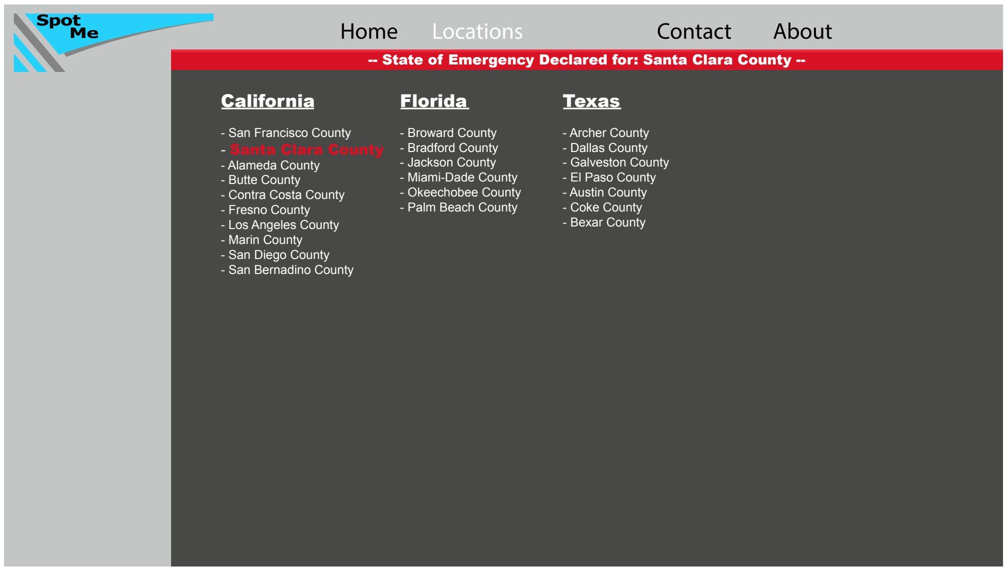


(b) Activity Diagram for Mobile App

Figure 2: Activity Diagrams

5 Conceptual Model

Our model shows the basic structure of the system. The user lands on the locations page in order to minimize the amount of time it takes to access vital information. As you can see in Figure 3, the user can view a list of locations, those under a state of emergency are highlighted in red and can be selected by the user. They are then taken to the map view where the user can view more detailed information about stranded victims.



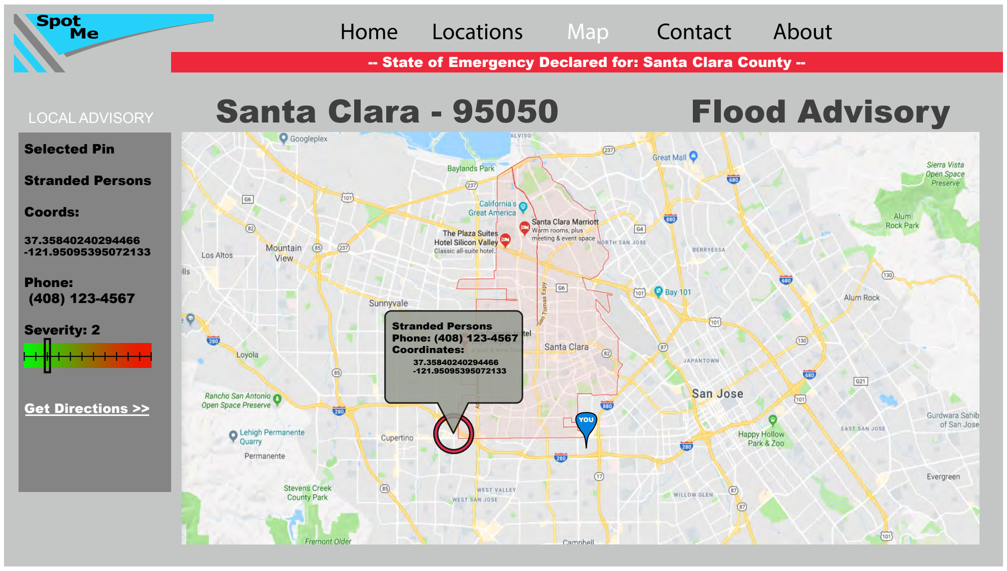
-- State of Emergency Declared for: Santa Clara County --

California
- San Francisco County
- Santa Clara County
- Alameda County
- Butte County
- Contra Costa County
- Fresno County
- Los Angeles County
- Marin County
- San Diego County
- San Bernadino County

Florida
- Broward County
- Bradford County
- Jackson County
- Miami-Dade County
- Okeechobee County
- Palm Beach County

Texas
- Archer County
- Dallas County
- Galveston County
- El Paso County
- Austin County
- Coke County
- Bexar County

Figure 3: Location View Model



-- State of Emergency Declared for: Santa Clara County --

LOCAL ADVISORY

Santa Clara - 95050 **Flood Advisory**

Selected Pin

Stranded Persons

Coords:

37.35840240294466
-121.95095395072133

Phone:
(408) 123-4567

Severity: 2

Get Directions >>

Stranded Persons
Phone: (408) 123-4567
Coordinates:
37.35840240294466
-121.95095395072133

Figure 4: Map View Model

6 System Design

In this section, we will describe the design of our system as well as the rationale behind our design decisions. We built two applications which comprised our entire system. Our mobile application will be used by individuals stranded in an emergency situation to drop a personalized pin on the map which includes their severity level, name, location, and phone number. The web application will display a map to enable first responders and rescue parties to determine the location of the stranded individuals and send help where required. Our mobile application will be available on both iOS and Android platforms and our web application will work on all modern browsers and mobile devices. The web and mobile applications are connected to our backend services which will serve the applications requests and storage systems which will house our application data. We have one Firebase Database and one PostgreSQL database. Firebase handles immediate data that is being sent from our mobile applications when the user drops a personalized pin on the map. The reason we use Firebase is because of its real time nature - this can be an advantage in a disaster-like situation. The PostgreSQL database was used for testing because Heroku and PostgreSQL have teamed up in providing a fully functional backend system. In this manner, we were able to have the app up and running in minutes and debug all backend issues on one platform: Heroku.

6.1 SpotMe Mobile Application

The users of our mobile applications will have the ability to drop a personalized pin on the map to display critical information for the rescuers to utilize. Upon downloading our application, the user will be prompted to enter their phone number and name which will be saved locally on the device. After successfully completing this step, they will be taken to a separate view in which they can drop a pin if a state of emergency is declared in their area. We utilize multiple government API's to determine if a state of emergency is declared near the user's location. If a state of emergency has been declared in their area, the user will be given a form to fill out their information which will be condensed into a virtual pin on the map. The pins on the map will consist of the stranded individuals name, location, phone number, and severity. Since time is a critical factor in an emergency situation, we want to make the process of dropping pins effortless and quick for our users. To do this, we will pre-populate 3 fields, the name, location, and phone number fields, and only ask the user to enter their severity. We will have access to the user's name and phone number since he or she would have already entered this information when they first opened our mobile application. Regarding the location, we can simply take the devices location from the GPS chip present in the device.

6.2 SpotMe Web Application

The users of our web application will be able to see a global view of every stranded individual in various emergency areas. The users of our web application will be subject to a background check to ensure that they are not viewing the pins of our users for malicious reasons. We will mainly reach out to individuals part of rescue programs, good samaritans, and law officers to ensure that the users pin information is safe and not in reach of individuals with wrong intentions. The web application users will be presented with a map which will show red pins of the users who are stranded. Upon clicking on one of the pins, the information about the user including their name, location, phone number, and severity will be displayed on a sidebar. The individual viewing the pin then has the option to either call or text the user right from the web application or further notify other first responders in the area where the user is stranded.

6.3 Technologies Used

- Front End:
  - HTML5/CSS3/BootStrap
    - We used HTML5 to build the user interface of our web application.
    - CSS3 enabled us to style our HTML5 elements to design a visually appealing and easy to use user interface.
    - BootStrap enabled us to quickly build out user interfaces which worked on all platforms including mobile, desktop, and tablet.
  - JavaScript / jQuery:
    - Used to make the application interactive. Such as click events, pop ups, etc.
  - React Native:
    - A free and open source framework used for cross platform development for iOS and Android apps.
    - We write our code in pure JavaScript and React Native will automatically compile the JavaScript code into the respective mobile platform code (Objective-C for iOS and Java for Android).
    - This saves us time from having to maintain two different code bases: one for iOS and one more Android.
- Back End:
  - Flask
    - We used Flask to develop an API which our mobile and web applications make requests to in order to save data into our data stores.
    - Flask is a lightweight python framework which handles everything from I/O handling to routing, and is quick, easy to use, and robust.
    - It also has a extensive developer community online and is widely used in the production grade applications.
  - Firebase

    * Used to store the pin informations that the user submits from the mobile application.
    * Firebase is a real time store, which allows our web application to listen to any changes within the store and update immediately after a change has been detected.

  - PostgreSQL
    * Used for testing and jump starting our application.
    * Used to store user information such as name and phone number which the user enters when first opening our mobile application.

7 Design Rationale

In this section, we outline our design decisions on the technologies we utilized to create the product as well as the UI features we utilized in our application.

7.1 UI Justification

- The rescuers as well as the individuals being rescued should have the same intuitive layout - with all the pins of the people who need help on the disaster map.
- The layout for both the mobile application and website were designed with simplicity and usability in mind.

7.2 Technology Justification

- We used HTML since that is the most widely used markup language for creating websites.
- We utilized JavaScript to enable us to manipulate the DOM in the HTML document and make the website interactive through animations.
- We used CSS since it enabled us to create a visually appealing website.
- We used React Native to build our mobile application as it will help us develop for both codebases (iOS and Android) at a much faster rate.
- We used PostgreSQL for storing metadata (user information, session information, etc.) and use NoSQL to track in real all of the pins that have dropped on the disaster map.

7.3 Technologies we did not use

- We didn’t use Swift or Kotlin to develop out iOS and Android apps respectively since we can use React Native to develop both codebases using JavaScript.

8 Technology Used

For an application at this scale, it is important to discuss the different components. This project is divided into two sectors: frontend and backend. Both components are important for the application to work seamlessly.

8.1 Frontend Technologies

The frontend portion of this application were split into two: web and mobile.

For the mobile portion, we used React-Native, a mobile framework that allows developers to build native applications using React. This was done by dividing up the UI into components. For example, in React-Native there is a ”View” tag which is similar to ”div” in HTML and an exact native correlation to ”UIView” in iOS and ”View” in Android.

For the web portion, we used React again and AngularJS as an MVC framework. It’s important to have an MVC framework so we can appropriately label our models and talk to our server in order to provide dynamic content to our users. Using React was a clear option for us because of its’ ability to create a Virtual DOM and apply changes to it only by changing the necessary portions - as opposed to re-rendering the entire DOM altogether. An application with users constantly adding and removing information will cause the DOM to render quite often, and so it would be great to have the DOM efficiently re-render.

8.2 Backend Technologies

For this application, the backend will be used by both the mobile and web components. We split up the backend into two components: server and database.

The server is responsible for handling all incoming requests such as creating an account, posting a ”hot spot”, etc. While there are many technologies out there that help in building servers, we stuck with a lightweight framework called Flask. Flask is a server-side framework that allows server-side code to be written in Python. Flask handles everything from I/O handling to routing, and is quick, easy to use, and robust. To host the server, we used use Heroku, which is a PaaS (platform as a service) as well as a subsidiary of Salesforce. Heroku has proved to be fault tolerant and is capable of handling large loads. Scaling horizontally on Heroku is also really easy and simple.

For our database, we used SQL since all of our relationships are strict. For testing purposes, we used PostgreSQL since Heroku and PostgreSQL have teamed up in providing a cohesive backend. That way, we could have the app up and running in minutes and could debug any backend issues on one platform: Heroku. We also used a NoSQL database, Firebase, to handle immediate data such as storing hotspots and severity of issues. The reason we used Firebase is because of its real time nature - this can be an advantage in a disaster-like situation.

9 Architectural Diagram

9.1 Structure

Our system used a client-server architecture. Clients will interact with the server in one of two ways. The first way is via a web client. Here, the user will pull all the hotspots in the region and can act accordingly. The second way is through a mobile client. Here, the user can either view the hotspots or post a hotspot. The user can also create an account and securely login through our server.

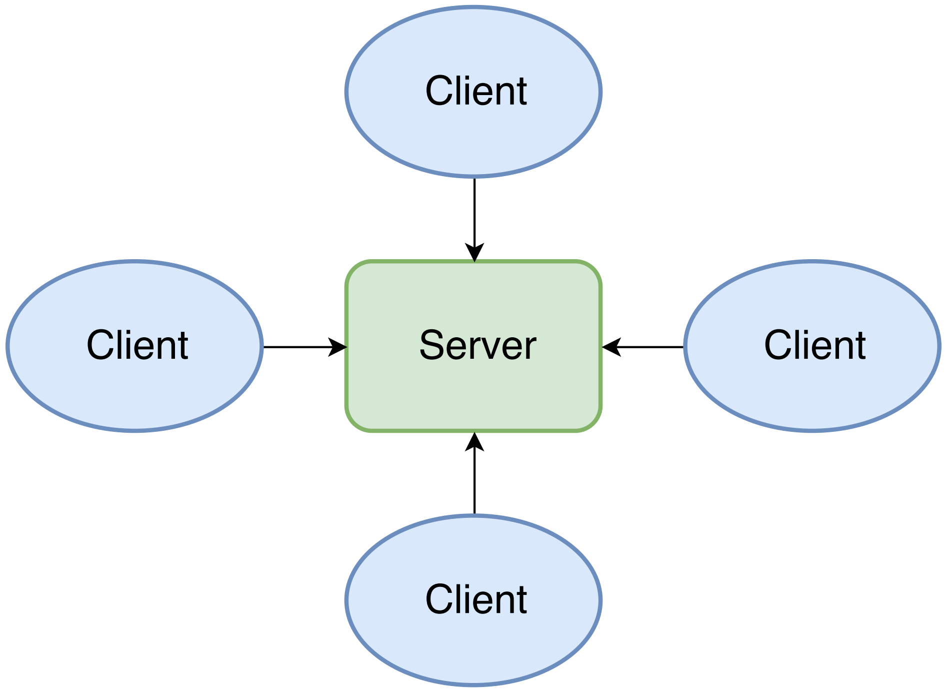


Figure 5: Client-Server Architecture Diagram

10 Test Plan

In order to test the application, we broke the process down to three steps: testing for resiliency, testing with large amounts of people, and testing for correctness.

10.1 Resiliency Testing

To test for resiliency, we will run many requests (side-by-side) to make sure that the server can handle the requests in a timely fashion and produce the right output. Both time and producing the right output are important factors here since this application is a disaster-relief application. The idea here is to test for both durability as well as availability of the overall system. The test for resilience is not only a test for the server component but the frontend component as well. This involves making sure that the frontend does not crash, lag, or present the wrong UI.

10.2 Stress Testing

To test with large amounts of people, we will group people together into different categories: those with the same emergency, those with the same severity, and those in the same region. We also had a group in a different region to test that the regions are split up properly on the server side. Splitting into groups is important so that one can test proper outputs given large and similar groups of people.

10.3 Testing for Correctness

Lastly, we tested for correctness. To do this, we made sure that the web component was actually pulling in data from the region that the user is in. We also made sure that the mobile component was able to push up a hot spot in the correct region.

11 Risk Analysis

We created a risk analysis table to determine the potential risks that can occur during our development process. The table shown below lists the risk, consequences, probability, severity, impact, and mitigation.

Table 1: Risk Analysis

Risk	Consequence	Probability	Severity	Impact	Mitigation
Bugs in application	Delay in release and the system would not be in working condition.	1	4	3	Test each major change thoroughly. Modularize the code base (loose coupling high cohesion).
Group member getting sick	Specific components of the system might be delayed along with deadlines.	0.4	8	4	Rest well eat healthy and have a good balance between work and personal life.
Not understanding the requirements	May cause extra work to be done and time would be wasted.	0.1	8	2	Have weekly or daily stand ups to make sure everyone is working on the correct requirements.
Learning Curves for new skills	Productivity will be low since most of the time will be consumed in effectively learning the new skill.	0.6	4	3	Pick technologies that we are familiar with and share information amongst team members to speed up the learning process.

12 Societal Impact

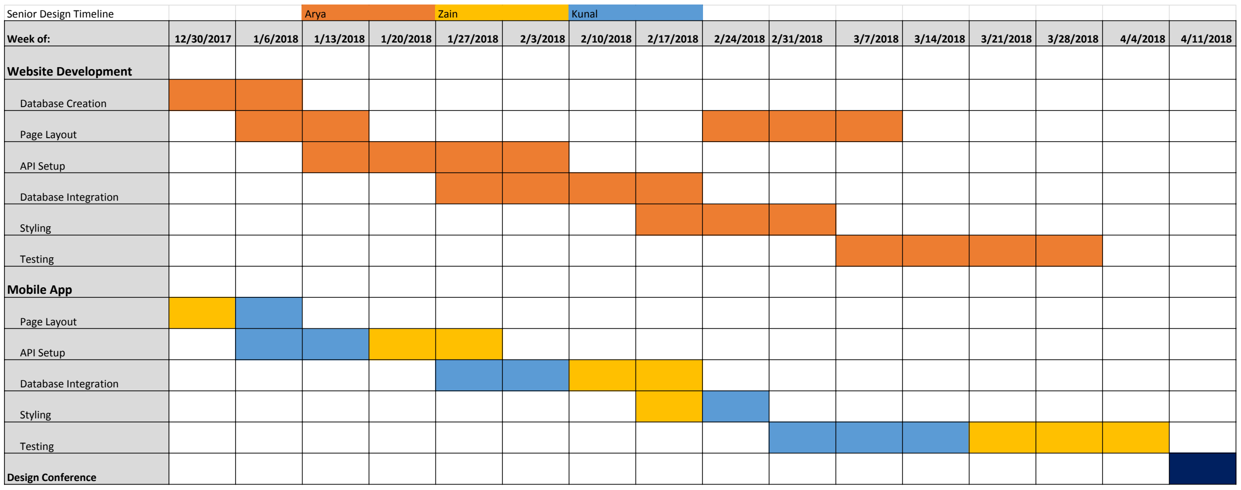
12.1 Impact on Society

SpotMe is an application designed to help those who are in trouble due to a natural disaster, general statewide emergency, or any severe situation. Every aspect of the application was built with the thought in mind that it must be quick and easy to use. During an emergency, people do not have much time to think or even plan out an escape route due to different variables in the situation. Emergency responders can also be very slow in responding depending on the severity of the situation (as experienced in Hurricane Harvey). We believe that the application can potentially help thousands of people that fall into helpless situations due to natural disasters or any statewide/national emergencies.

12.2 Ethical Analysis

When sitting down with Dr. Figueria to build SpotMe, the plan was to build an application with an ethical purpose. After witnessing the number of people stranded because of Hurricane Harvey, we realized that responders being busy poses a huge ethical concern for the on-call emergency service in this country. We built SpotMe to address that exact ethical concern and to assuage people's fears during a severe situation.

13 Development Timeline



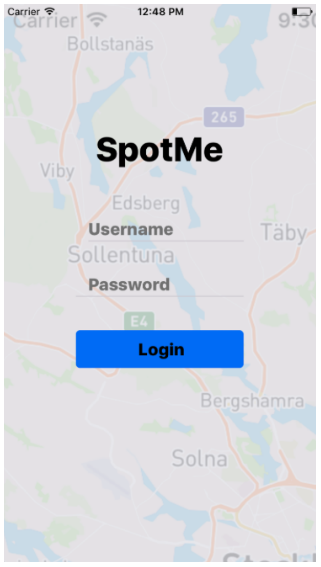
Senior Design Timeline			🟧 Arya	🟧	🟨 Zain	🟨	🟦 Kunal	🟦								
Week of:	**12/30/2017**	**1/6/2018**	**1/13/2018**	**1/20/2018**	**1/27/2018**	**2/3/2018**	**2/10/2018**	**2/17/2018**	**2/24/2018**	**2/31/2018**	**3/7/2018**	**3/14/2018**	**3/21/2018**	**3/28/2018**	**4/4/2018**	**4/11/2018**
Website Development																
Database Creation	🟧	🟧														
Page Layout		🟧	🟧						🟧	🟧	🟧					
API Setup			🟧	🟧	🟧	🟧										
Database Integration					🟧	🟧	🟧	🟧								
Styling								🟧	🟧	🟧						
Testing											🟧	🟧	🟧	🟧		
Mobile App																
Page Layout	🟨	🟦														
API Setup		🟦	🟦	🟨	🟨											
Database Integration					🟦	🟦	🟨	🟨								
Styling								🟨	🟦							
Testing										🟦	🟦	🟦	🟨	🟨	🟨	
Design Conference																⬛

Figure 6: Development Timeline

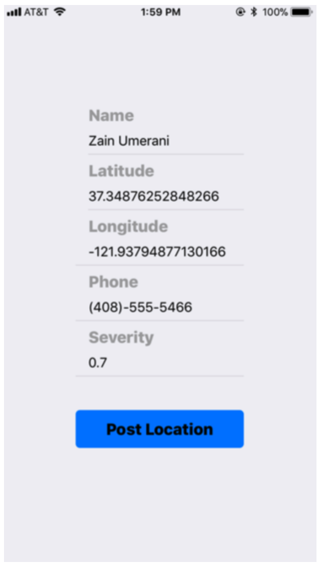
14 Results

14.1 Test Results

Our team was able to develop and implement a working system that performed the actions specified in the design review. The mobile application we developed is able to determine whether a state of emergency has been declared in a specified area, and activate the hotspot generator in order for the user to input their location and situational information into the database. The user is prompted to log in so that we can have some control over the user's actions and be able to verify the validity of the user's intentions. After logging in, the application can access the geo-location information from the mobile device and input the coordinates into the form as seen in Figure 7a.



(a) Mobile Login Form

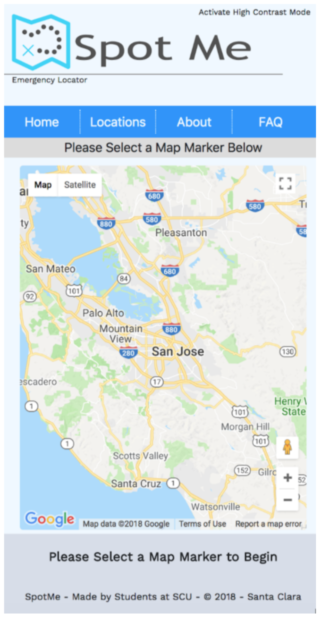


Name
Zain Umerani
Latitude
37.34876252848266
Longitude
-121.93794877130166
Phone
(408)-555-5466
Severity
0.7

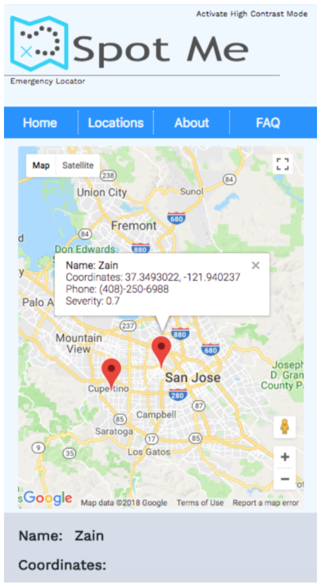
(b) Mobile Submit Form

Figure 7: Mobile Interface

The web application was developed successfully as well, able to display multiple markers after being submitted through the application. By integrating a Google Maps API, we were are able to provide an accurate and reliable interface that can cater to users world wide. The website is available to all users in its current state in an effort to get survivor information out to as many people as possible, as quickly as possible following a disaster. Many of the good Samaritans or search and rescuers are looking to access this information as quickly as possible, so the use of a login form may not play in their favor. All of the stranded party's information is displayed on the site when selecting a specific pin, so rescuers are able to get into contact and gather as much information as they can before mobilizing.


Activate High Contrast Mode
Spot Me
Emergency Locator
Home
Locations
About
FAQ
Please Select a Map Marker Below
Please Select a Map Marker to Begin
SpotMe - Made by Students at SCU - © 2018 - Santa Clara

(a) Responsive Web Model

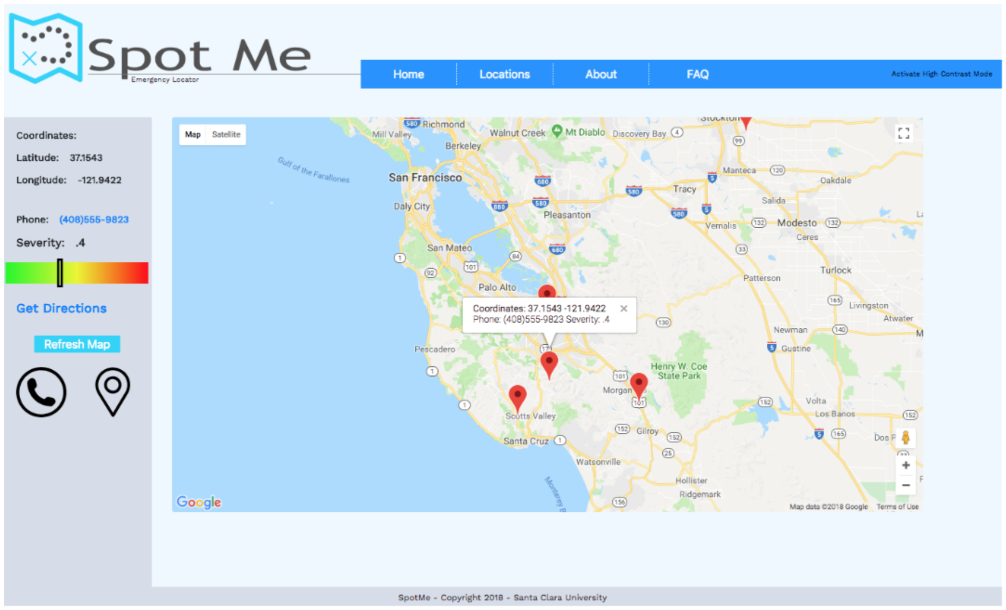


Name: Zain
Coordinates: 37.3493022, -121.940237
Phone: (408)-250-6988
Severity: 0.7

Name: Zain
Coordinates:

(b) Selected Marker Mobile

Figure 8: Web Interface


Spot Me
Emergency Locator
Home
Locations
About
FAQ
Activate High Contrast Mode
Get Directions
Refresh Map
SpotMe - Copyright 2018 - Santa Clara University

Coordinates:
Latitude: 37.1543
Longitude: -121.9422
Phone: (408)555-9823
Severity: .4

Coordinates: 37.1543 -121.9422
Phone: (408)555-9823 Severity: .4

Figure 9: Desktop Web Interface

14.2 Deployment

Currently we are in talks with FEMA to see whether they are interested in deploying our application. There is a tremendous need for an application such as this as we have seen time and time again that EMS first responders are often overwhelmed following natural disasters. In the event that an organization requires security or verification for the first responder party, it can be added to the application. As of now, we are still waiting for a response as to whether FEMA is interested in adopting our application, and we have reached out to other similar organizations that could also benefit from this service. Our goal is to implement our application with backing from an organization so that real-world needs can be better met, and the service can be tailored for various organizations and their specific needs.

14.3 Lessons Learned

Throughout the design process, we understood that there might be particular elements of the system that we may had overlooked that proved to be important for the finished product. Some of the features we wanted to initially implement were not completed by the design conference, and the feedback from the audience reflected that. As security is a crucial component of user data, we should have discussed the security aspects of our platform more thoroughly during the presentation. Implementing a secure website and application is becoming more and more difficult with new technologies coming out that hackers and malicious actors are utilizing, so staying ahead of the game is very important in a publicly released application, especially one that deals with user locations. Above all, the most important lesson we learned is the necessity to communicate with a client or potential client throughout the design process, as their real-world needs will differ from the theoretical needs thought of by a developer. Having a mentor or guide experienced in the field or subject in which the application is catered towards will tremendously help the end product, take some guess-work out of the design process, and ultimately lead to a more impactful finished product.

15 Conclusion

This project was an eye-opening experience that allowed us to develop a product that can be implemented in a real-world environment and provide a large number of people with help. Victims stranded after a natural disaster are very vulnerable, and in a mode of survival that many of us have not experienced. Some victims may have disabilities and might require a quicker rescue as they may not be able to provide aid to themselves or those around them. Floods can quickly render families or individuals helpless so developing a system that can share their location with as many people as possible in this time of need can provide immense improvements to disaster response times, lift some of the burden from public EMS services, and streamline the efforts of private citizen search and rescue teams. Our hopes are that this application can be implemented and used in disaster situations, providing a much needed, new way to alert first responders of a given victim situation.

Appendices

A Appendix A - Web Source Code

A.1 Back-End

The source code below is used to select the SQL database and parse the data to an XML document where the Google Maps API is able to access the document.

```
<?php
require("dbconnect.php");

function parseToXML($htmlStr)
{
$xmlStr=str_replace('<','&lt;',$htmlStr);
$xmlStr=str_replace('>','&gt;',$xmlStr);
$xmlStr=str_replace('"','&quot;',$xmlStr);
$xmlStr=str_replace("'",'&#39;',$xmlStr);
$xmlStr=str_replace("&",'&amp;',$xmlStr);
return $xmlStr;
}


$connection=mysqli_connect(DBHOST, DBUSER, DBPASS);
if (!$connection) {
  die('Not connected : ' . mysql_error());
}

// Set the active MySQL database
$db_selected = mysqli_select_db($connection, DBNAME);
if (!$db_selected) {
  die ('Can\'t use db : ' . mysql_error());
}

// Select all the rows in the markers table
$query = "SELECT * FROM hotspots WHERE 1";
$result = mysqli_query($connection, $query);
if (!$result) {
  die('Invalid query: ' . mysql_error());
}

header("Content-type: text/xml");

// Start XML file, echo parent node
echo '<markers>';

// Iterate through the rows, printing XML nodes for each
while ($row = @mysqli_fetch_assoc($result)){
  // Add to XML document node
  echo '<marker ';
  echo 'id="' . $row . '" ';
//  echo 'name="' . parseToXML($row['name']) . '" ';
//  echo 'address="' . parseToXML($row['address']) . '" ';
  echo 'lat="' . $row['lat'] . '" ';
  echo 'lng="' . $row['lng'] . '" ';
  echo 'phone="' . $row['phone'] . '" ';
    echo 'severity="' . $row['severity'] . '" ';
  echo '/>';
}

// End XML file
echo '</markers>';
?>
```

A.2 Front-End

The source code below consists of the HTML and JS files that construct the front end layout of the site

```
<?php
    session_start();
?>
    <!DOCTYPE html>
    <html>
    <head>
        <meta charset="UTF-8" />
        <link href="https://fonts.googleapis.com/css?family=Work+Sans" rel="stylesheet" />
        <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
        <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
        <title>SpotMe</title>
        <link rel="stylesheet" href="css/bootstrap.css" />
        <link rel="stylesheet" type="text/css" title="default" href="style.css" />
        <link rel="alternate stylesheet" type="text/css" title="contrast" href="contrast.css" />
    </head>

    <body>
        <div class="top_banner">
            <span id="motto">Emergency Locator</span>
            <img id="logo" src="assets/Asset2.svg" alt="spotme">
        </div>

        <div id="border"></div>

        <nav class="nav">
            <div id="nav_wrapper">
                <a href="index.php">Home</a>
                <a href="locations.php">Locations</a>
                <a href="about.php">About</a>
                <a id="faq" href="faq.php">FAQ</a>
            </div>
        </nav>
```

```
<span id="contrast" class="contrast" onclick="switch_style('contrast');document.getElementById(
<span id="default" class="contrast" onclick="switch_style('default');document.getElementById('d

<div id="small_banner">
    <span id="to_hide">Please Select a Map Marker Below</span>
</div>

<script>
    function reload() {
        location.reload();
    }

    var customLabel = {
        restaurant: {
            label: 'SEVERE'
        },
        house: {
            label: 'B'
        }
    };

    function initMap() {
        var map = new google.maps.Map(document.getElementById('map'), {
            center: {
                lat: 37.3488,
                lng: -121.9380
            },
            zoom: 9
        });

        var infoWindow = new google.maps.InfoWindow;

        if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(function(position) {
                var pos = {
                    lat: position.coords.latitude,
                    lng: position.coords.longitude

                };

                // infoWindow.setPosition(pos);
                //  infoWindow.setContent('Your Location');
                infoWindow.open(map);
                map.setCenter(pos);
            }, function() {
                handleLocationError(true, infoWindow, map.getCenter());
            });
        } else {
            // Browser doesn't support Geolocation
            handleLocationError(false, infoWindow, map.getCenter());
        }


        // Change this depending on the name of your PHP or XML file
```

```
downloadUrl('https://bb426599.ngrok.io/api/v1/getLocations', function(data) {
    var xml = data.responseXML;
    var markers = xml.documentElement.getElementsByTagName('marker');
    Array.prototype.forEach.call(markers, function(markerElem) {
        var id = markerElem.getAttribute('id');
        //  var name = markerElem.getAttribute('name');
        //  var address = markerElem.getAttribute('address');
        var lat = markerElem.getAttribute('lat');
        var long = markerElem.getAttribute('lng');
        var phone = markerElem.getAttribute('phone');
        var severity = markerElem.getAttribute('severity');
        var name = markerElem.getAttribute("name");
        var point = new google.maps.LatLng(parseFloat(markerElem.getAttribute('lat')),pa

        var infowincontent = document.createElement('div');
        var strong = document.createElement('strong');
        var coordinates = document.createElement('coordinates');
        var text = document.createElement('text');
        var sev = document.createElement('severity');

        //  HEADER FOR DISPLAYING COORDINATES
        strong.textContent = "Name: " + name;
        infowincontent.appendChild(strong);

        infowincontent.appendChild(document.createElement('br'));

        coordinates.textContent = "Coordinates: " + lat + ", " + long;
        infowincontent.appendChild(coordinates);

        infowincontent.appendChild(document.createElement('br'));

        text.textContent = "Phone: " + phone ;
        infowincontent.appendChild(text);

        infowincontent.appendChild(document.createElement('br'));

        sev.textContent = "Severity: " + severity;
        infowincontent.appendChild(sev);


        var icon = customLabel[severity] || {};
        var marker = new google.maps.Marker({
            map: map,
            position: point,
            label: icon.label
        });
        marker.addListener('click', function() {
            infoWindow.setContent(infowincontent);
            infoWindow.open(map, marker);

            var tell = "tel:";
            var bracket = ">";

            //Methods for altering the left pane information
```

```
                    document.getElementById("span_name").innerHTML = name;
                    document.getElementById("span1").innerHTML = lat;
                    document.getElementById("span2").innerHTML = long;
                    document.getElementById("span3").innerHTML = "<a href='tel:" + phone + "'>"
                    document.getElementById("span4").innerHTML = severity;
                    document.getElementById("phone_link").href = "tel:" + phone;


                    var slider = severity * 100 - 4;
                    var slide = slider + "%";
                    //Javascript slider setting
                    //  document.getElementById("severity_slider").style.marginLeft = slider +

                    //Jquery animation of slider
                    $("#severity_slider").animate({
                        marginLeft: slide
                    });
                    document.getElementById("severity_container").style.display = "block";

                    elements = document.getElementsByClassName("info");
                    for (var i = 0; i < elements.length; i++) {
                        elements[i].style.display = "inline-block";
                    }

                    //Dynamic link for directions
                    var link;
                    if (lat) {
                        link = lat + "," + long;
                        document.getElementById("directions").style.display = "inline";
                        document.getElementById("directions").href = "https://www.google.com/ma
                        document.getElementById("map_link").href = "https://www.google.com/maps
                        document.getElementById("message").style.display = "none";
                        // document.getElementById("small_banner").style.height = "0px";
                        $("#small_banner").animate({
                            height: "0px"
                        });
                        document.getElementById("to_hide").style.display = "none";
                        document.getElementById("phone_icon").style.display = "block";
                        document.getElementById("map_icon").style.display = "block";
                    }
                });
            });
        });
    }

    function downloadUrl(url, callback) {
        var request = window.ActiveXObject ?
            new ActiveXObject('Microsoft.XMLHTTP') :
            new XMLHttpRequest;

        request.onreadystatechange = function() {
            if (request.readyState == 4) {
                request.onreadystatechange = doNothing;
                callback(request, request.status);
                }
            };
            request.open('GET', url, true);
            request.send(null);
        }

        function doNothing() {}
    </script>


    <div class="left_pane">
        <span class="info">Name:</span><span id="span_name"></span><br>
        <span class="info">Coordinates:</span><br>
        <span class="info">Latitude: </span><span id="span1"></span><br>
        <span class="info">Longitude: </span><span id="span2"></span><br><br>
        <span class="info">Phone: </span><span id="span3"></span><br>
        <span class="info" id="sev_span">Severity: </span><span id="span4"></span><br>
        <div id="severity_container">
            <div id="severity_slider"></div>
        </div>
        <span><a id="directions" href="">Get Directions</a></span><br>
        <span id="message"><strong>Please Select a Map Marker to Begin</strong></span>
        <div id="refresh" onclick="reload()">Refresh Map</div>
        <a id="phone_link"><img id="phone_icon" src="assets/phone.svg" alt="call_number"></a>
        <a id="map_link"><img id="map_icon" src="assets/map.svg" alt="get_directions"></a>
    </div>



    <div id="map">
    </div>
    <script>
        var style_cookie_name = "style";
        var style_cookie_duration = 30;
        var style_domain = "spotme.com";

        function switch_style(css_title) {
            var i, link_tag;
            for (i = 0, link_tag = document.getElementsByTagName("link"); i < link_tag.length; i++)
                if ((link_tag[i].rel.indexOf("stylesheet") != -1) &&
                    link_tag[i].title) {
                    link_tag[i].disabled = true;
                    if (link_tag[i].title == css_title) {
                        link_tag[i].disabled = false;
                    }
                }
                set_cookie(style_cookie_name, css_title,
                    style_cookie_duration, style_domain);
            }
        }

        function set_style_from_cookie() {
            var css_title = get_cookie(style_cookie_name);
            if (css_title.length) {
                    switch_style(css_title);
                }
            }

            function set_cookie(cookie_name, cookie_value, lifespan_in_days, valid_domain) {
                var domain_string = valid_domain ? ("; domain=" + valid_domain) : '';
                document.cookie = cookie_name +
                    "=" + encodeURIComponent(cookie_value) +
                    "; max-age=" + 60 * 60 *
                    24 * lifespan_in_days +
                    "; path=/" + domain_string;
            }

            function get_cookie(cookie_name) {
                var cookie_string = document.cookie;
                if (cookie_string.length != 0) {
                    var cookie_value = cookie_string.match(
                        '(^|;)[\s]*' +
                        cookie_name +
                        '=([^;]*)');
                    return decodeURIComponent(cookie_value[2]);
                }
                return '';
            }

        </script>
        <script async defer src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBcLNeVLpSQKcv2Bh_XGL

        </body>

    <footer>
        SpotMe - Made by Students at SCU - &copy; 2018 - Santa Clara University
    </footer>

    </html>
    <?php
    session_destroy();
?>
```

B Appendix B - Mobile App Source Code

B.1 Back-End

The following code is the API for SpotMe. Both the mobile and web components use the endpoint to post and pull data, respectively.

```
     import os # Incase we need to mess with Heroku Dyno binaries/packages.

from flask import Flask, make_response, jsonify, Response, render_template
from flask_restful import reqparse, Resource
from flask_cors import CORS, cross_origin
from database import Database
from database import CursorFromConnectionFromPool
from urllib.parse import urlparse # To parse Heroku DB URL.
import psycopg2

import xml.etree.cElementTree as ET

app = Flask(__name__)
CORS(app)

url = urlparse('postgres://ucyroaeunxxsrn:e042657f9582726c420bf5bc72987796f029d915b78a756b3fa93c0d86032
Database.initialize(database=url.path[1:], user=url.username, password=url.password, host=url.hostname,

''' GET '''
@app.route('/api/v1/getLocations', methods=['GET'])
def hello_world():
with CursorFromConnectionFromPool() as cursor:
try:
sql_string = 'SELECT * from locations'
cursor.execute(sql_string)
result = cursor.fetchall()

root = ET.Element("markers")

for res in result:
marker = ET.SubElement(root, "marker", id="Array", lat='{}'.format(res[1]), lng='{}'.format(res[2]),
phone='{}'.format(res[3]), severity='{}'.format(res[4]), name='{}'.format(res[5]))

tree = ET.ElementTree(root)

tree.write('templates/file.xml')
data = ET.dump(root)
return render_template('file.xml'), 201, {'Content-Type': 'text/xml'}
except psycopg2.IntegrityError:
result = {'message' : 'Resources not found.', 'status' : 404}
jsonify(result)


''' POST '''
@app.route('/api/v1/postLocation', methods=['POST'])
def postLocation():
parser = reqparse.RequestParser(bundle_errors=True)
parser.add_argument('lat', required=True, help="You need latitude.")
parser.add_argument('lng', required=True, help="You need longitude.")
parser.add_argument('phone', required=True, help="You need a phone number.")
parser.add_argument('severity', required=True, help="You need severity.")
parser.add_argument('name', required=True, help="You need a name.")

args = parser.parse_args()

response = None
with CursorFromConnectionFromPool() as cursor:
try:
```

```
sql_string = 'INSERT INTO locations (lat, lng, phone, severity, name) VALUES (%s, %s, %s, %s, %s) RETUR
cursor.execute(sql_string, (args['lat'], args['lng'], args['phone'], args['severity'], args['name']))
id_of_new_row = cursor.fetchone()
response = {
'message:' : 'location created successfully',
'status' : 201,
'id' : id_of_new_row[0],
'lat' : id_of_new_row[1],
'lng' : id_of_new_row[2],
'phone' : id_of_new_row[3],
'severity' : id_of_new_row[4],
'name' : id_of_new_row[5]
}
return jsonify(response)
except psycopg2.IntegrityError:
# We already have an entry with the same email address (return HTTP code 409 - conflict).
response = {'message:' :'Duplicate Location' , 'status' : 409}
return jsonify(response)



if __name__ == "__main__":
app.run(port=5000, debug=True)
```

B.2 Front-End

The code below is for the mobile portion of SpotMe - written with the React-Native platform in JavaScript.

```
     import React from 'react';
import { StyleSheet, Text, View, TextInput, Image } from 'react-native';
import { StackNavigator } from 'react-navigation';
import firebase from 'firebase';
import { Container, Header, Content, Form, Item, Input, Label, Button } from 'native-base';

class PostLocation extends React.Component {

  state = {
      latitude: '',
      longitude: '',
      phone: '',
      severity: '',
      name: ''
    };

  static navigationOptions = {
    header: null
  }

  componentDidMount()
  {
```

```
  navigator.geolocation.getCurrentPosition(
          (position) => {
              var lat = position.coords.latitude.toString();
              var lng = position.coords.longitude.toString();
              this.setState({latitude: lat})
              this.setState({longitude: lng})
          }
  );

}

  postLocation() {
    const {latitude, longitude, phone, severity} = this.state;

    fetch('https://bb426599.ngrok.io/api/v1/postLocation', {
        method: 'POST',
        headers: {
          'Accept': 'application/json',
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          lat: this.state.latitude,
          lng: this.state.longitude,
          phone: this.state.phone,
          severity: this.state.severity,
          name: this.state.name
        })
    }).then(() => {
      alert('Success!');
    })
}

render()
{
    return (
      <Container style={{marginTop: '25%', flex: 1, flexDirection: 'column', alignItems: 'center', ju
        <Content>
          <Form style={styles.formContainer}>
              <Item floatingLabel>
                <Label style={{fontSize: 20, color: 'gray', fontWeight: '800', justifyContent: 'cente
                <Input
      onChangeText={(value) => this.setState({name: value})}
      value={this.state.name}
    />
              </Item>

              <Item floatingLabel>
                <Label style={{fontSize: 20, color: 'gray', fontWeight: '800', justifyContent: 'cente
                <Input
      onChangeText={(value) => this.setState({latitude: value})}
      value={this.state.latitude}
    />
              </Item>
              <Item floatingLabel last>
```

```
                <Label style={{fontSize: 20, color: 'gray', fontWeight: '800'}}>Longitude</Label>
                <Input
      onChangeText={(value) => this.setState({longitude: value})}
      value={this.state.longitude}

    />
              </Item>

              <Item floatingLabel last>
                <Label style={{fontSize: 20, color: 'gray', fontWeight: '800'}}>Phone</Label>
                <Input
      onChangeText={(value) => this.setState({phone: value})}
      value={this.state.phone}

    />
              </Item>

              <Item floatingLabel last>
                <Label style={{fontSize: 20, color: 'gray', fontWeight: '800'}}>Severity</Label>
                <Input
      onChangeText={(value) => this.setState({severity: value})}
      value={this.state.severity}

    />
              </Item>
              <View style={{flexDirection: 'row', justifyContent: 'center', alignItems: 'center'}}>
                <Button onPress={this.postLocation.bind(this)} primary style={styles.blueButton}><Tex
              </View>
          </Form>
        </Content>
      </Container>
    );
  }
}

class MainScreen extends React.Component {

  static navigationOptions = {
    header: null
  }

  state = {
      username: '',
      password: ''
    };

  signUpButtonPressed() {
   const {username, password} = this.state;

   firebase.auth().signInWithEmailAndPassword(username, password)
   .then(() => {
         // Do something ...
      })
   .catch((err) => {
```

```
   //alert('Please sign up first.');
        this.props.navigation.navigate('PostLocation')
   })
  }

  render() {
    return (
        <Container style={{flex: 1, flexDirection: 'column', alignItems: 'center', justifyContent: 'cen
          <Image source={require('./images/simple-map-view.png')} style={{flex:1, position: 'absolute',
          <Text style={{marginTop: '40%', fontSize: 40, color: 'black', fontWeight: '800'}}>SpotMe</Tex
          <Content>
            <Form style={styles.formContainer}>
              <Item floatingLabel>
                <Label style={{fontSize: 20, color: 'gray', fontWeight: '800', justifyContent: 'center'
                <Input
onChangeText={(value) => this.setState({username: value})}
value={this.state.username}
/>
              </Item>
              <Item floatingLabel last>
                <Label style={{fontSize: 20, color: 'gray', fontWeight: '800'}}>Password</Label>
                <Input
secureTextEntry={true}
onChangeText={(value) => this.setState({password: value})}
value={this.state.password}

/>
              </Item>
              <Button onPress={this.signUpButtonPressed.bind(this)} primary style={styles.blueButton}><
            </Form>
          </Content>
        </Container>
    );
  }
}

const App = StackNavigator({
    Main: {
      screen: MainScreen
    },

    PostLocation: {
      screen: PostLocation
    }
})

const styles = StyleSheet.create({
  formContainer: {
    marginTop: '10%',
    justifyContent: 'center',
    alignItems: 'center'
  },
  blueButton: {
    marginTop: 40,
    width: 200,
    justifyContent: 'center'
  }
});

export default App;
```

16 References

1. Zhang, F., R.E. Morss, J.A. Sippel, T.K. Beckman, N.C. Clements, N.L. Hampshire, J.N. Harvey, J.M. Hernandez, Z.C. Morgan, R.M. Mosier, S. Wang, and S.D. Winkley, 2007: An In-Person Survey Investigating Public Perceptions of and Responses to Hurricane Rita Forecasts along the Texas Coast. Wea. Forecasting, 22, 11771190, https://doi.org/10.1175/2007WAF2006118.1

2. Marsha L. Vanderford, Teresa Nastoff, Jana L. Telfer Sandra E. Bonzo (2007) Emergency Communication Challenges in Response to Hurricane Katrina: Lessons from the Centers for Disease Control and Prevention, Journal of Applied Communication Research, 35:1, 9-25, DOI: 10.1080/00909880601065649

3. Jeannette Sutton, Cedar League, Timothy L. Sellnow, Deanna D. Sellnow. (2015) Terse Messaging and Public Health in the Midst of Natural Disasters: The Case of the Boulder Floods. Health Communication 30:2, pages 135-143.

4. Waugh, W. L. and Streib, G. (2006), Collaboration and Leadership for Effective Emergency Management. Public Administration Review, 66: 131-140. doi:10.1111/j.1540-6210.2006.00673.x

5. Perry, R. W. and Lindell, M. K. (2003), Preparedness for Emergency Response: Guidelines for the Emergency Planning Process. Disasters, 27: 336-350. doi:10.1111/j.0361-3666.2003.00237.x