Institiúid Teicneolaíochta Cheatharlach

INSTITUTE *of* TECHNOLOGY CARLOW

At the Heart of South Leinster

LifeBuoy Monitor Functional Specification

BSc (Hons) in Software Development

Name: Garry Byrne

Student ID: C00120055

Year: 4th Year

Supervisor: Dr. Oisin Cawley

Due date: 18-04-2018

Table Of Contents

Abstract

The purpose of this document is to set out the functional and nonfunctional requirements for the project. It will also cover other topics such as iterations, System architecture and brief use cases among others.

Introduction

Introduction

The purpose of this project is to create a system to allow a user the ability to monitor lifebuoys remotely. The user could be either a local authority employee or a private individual.
The system will allow the the user to log into the system, view their registered devices, this will also show them the status of the devices(red or green). The user will also be able to update the location of their registered devices and to reset any active alarms.

The system will also be designed to be simplistic and intuitive to use.

Target Market

The main target market of this system will be Local Authorities such as county councils who look after the waterways within their area in Ireland. Local councils face the issue of their lifebuoys been vandalised and misused. This system will give the user a instant notification if the lifebuoy is used, be it correctly or incorrectly thus allowing the user to check on the device themselves. Lifebuoys are required to me checked if:

- located within lifeguarded zones then should be checked daily by lifeguards when on duty.
- located in non-lifeguarded zones, should be checked on a fortnightly basis during the summer season (May – September) and checked monthly in wintertime.
- located within cities, major towns, harbours should be checked weekly.
- located in minor towns, villages and rural areas should be checked every two months.

So as can be seen there is a benefit for a local council to purchase and use this device. If a device placed in a rural area is vandalised, it might not be noticed until the bi-monthly inspection takes place, with this device, if it is vandalised a week after its inspection the user gets an instant notification and they can investigate immediately compared to not knowing for another 7 weeks when the next inspection will occur.

Functional Requirements

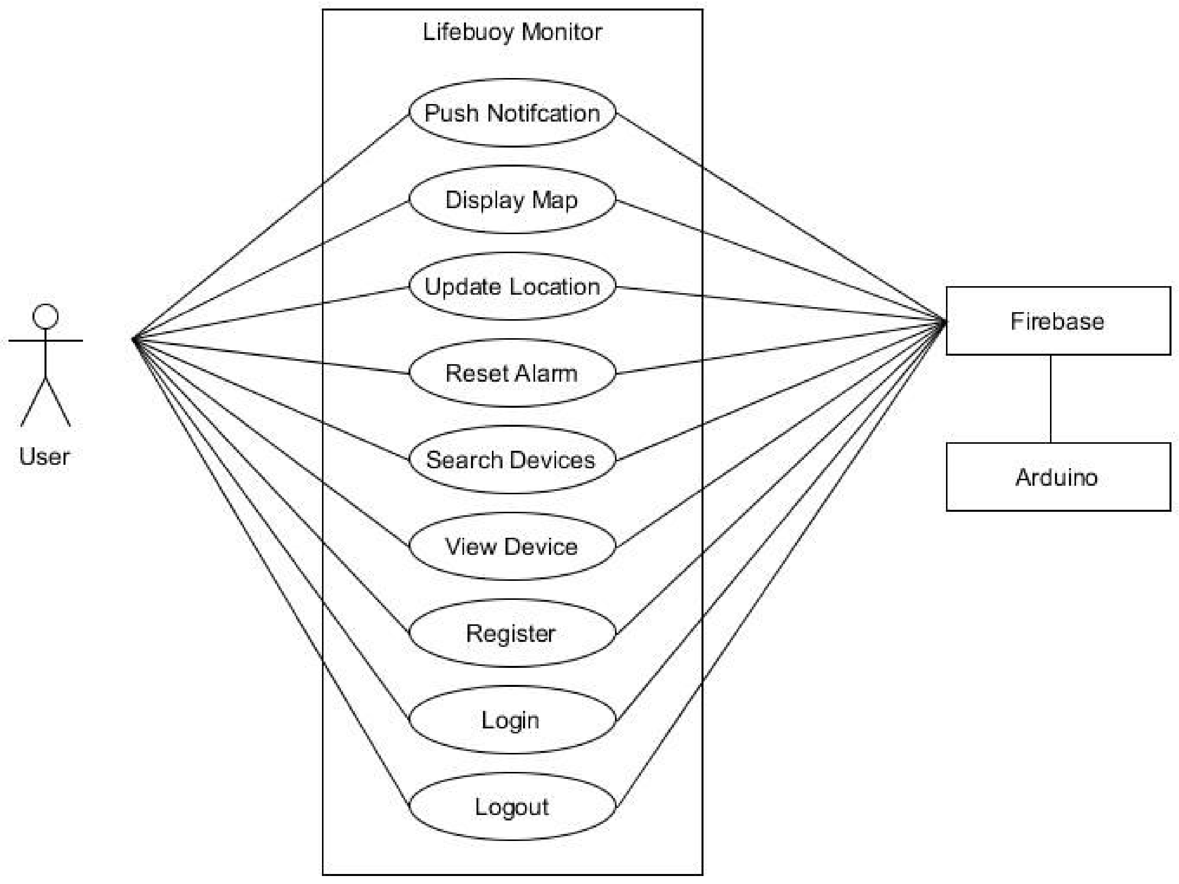
Software Requirements

- C++ - The program that will run on the arduino MKRFOX 1200 will be written in C++.
- HTML,CSS,Javascript - The app will be written in HTML,CSS and Javascript.
- NoSQL database - A NoSQL will be used to store all data related to the system.
- Sigfox network - The Sigfox network will be used to transmit a message from the device to the sigfox backend and from their it will be forwarded to firebase.
- Cordova Angular - Cordova provides a wrapper around HTML5, which is written in native platform languages like Java (for android), objective-C, swift (for iOS) which then allows access to device specific features like camera, accelerometer and contacts.
- Arduino IDE - Needed to write programs for the device.
- Visual Code - Needed to write code for the hybrid app.

Hardware Requirements

- Smartphone - Needed to install and run the app. Needs access to wifi or 3/4G with location services turned on.
- Arduino mkrfox 1200 - Needed for connecting to the Sigfox network and for allowing the connection of needed sensors.
- Altimeter/Accelerometer - Needed for sensing if the device has moved and for waking the mkr fox 1200 from sleep.

Use Case Diagram



Brief use cases

1: Login

Actors: User, Database
Brief Description: This use case begins when a user wishes to log into the app. The user with be prompted to enter a username and password. If the user enters their details and they get the login details correct they will be allowed to progress to the next page relevant to that user.
If the user enter their details and they get the login details incorrect they will not be allowed to progress to the next page instead they will be asked to re-enter their details.
This use case ends when the user has either successfully logged in to the app or has given up trying to log in. If the user has logged in successfully, details of the user's login (time, date) will be stored in the relevant database.

2: Logout

Actors: User, Database
Brief Description: This use case begins when the user wants to logout of the app. The user clicks the logout button. This use case ends when the user has successfully logged out and the login screen is displayed.
.

3: Register

Actors: User, Database
Brief Description: This use case begins when the user wishes to Log in but has yet to create an account. The user will get to the Login page where there will be an option to create an account if the user does not currently have one. The user must enter the required details. This use case ends when the user has successfully entered the required details and created an account. These user details will be stored in the relevant database.

4: View Devices

Actors: User, Database
Brief Description: This use case begins when the user wishes to view a device that have occurred. The user will be able to the click the ID of the desired Lifebuoy monitor device.The user will then be presented the details of the device, and will also be able to see the status of the device.

5: Search Devices

Actors: User, Database
Brief Description: This use case begins when the user wishes to search for a device. The user will be able enter the full/partial ID of the device .The user will then be presented the details of the device/devices matching the ID. The user will be click on the wanted device, they will then be able to see the details of the device.

6: Reset Alarms

Actors: User, Database
Brief Description: This use case begins when the user wishes to reset a alarm. The user will search for the desired device, once the device is found the user will click the device and be able to view all relevant device information. On this page the user will be able to click the reset alarm button. This will then reset the alarm and google maps marker. The use case ends when the user has successfully reset the alarm.

7: Update Location

Actors: User, Database
Brief Description: This use case begins when the user wishes to update the location of a device. The user will search for the desired device, once the device is found the user will click the device and be able to view all relevant device information. On this page the user will be able to click the update location. This will update the longitude and latitude of the device and then reposition the relevant google maps marker. The use case ends when the user has successfully updated the alarm.

8: Display Map

Actors: User, Database
Brief Description: This use case begins when the user wishes to view all devices(represented on a map).The user logs in, on the redirected page the user will be shown a google map with all their markers displayed through use of markers. The user will be able to click on a marker, this will display some basic information about the device to the user. The use case ends when the user has viewed the map.

9: Push Notifications

Actors: User, Database, Arduino
Brief Description: This use case begins when an event has occurred with the arduino. This event will be if the accelerometer/altimeter senses any movement. If an event has occurred, the arduino will send a message to both the user through push notification and tothe database. This

use case ends when the user has received the push notification and the details of the alert have been saved to the database.

Non-Functional Requirements

Non-Functional requirements are any other requirements that are not functional requirements. Where functional requirements are what a system should do, non-functional requirements are what can be used to judge the behavior of the system under certain conditions.

FURPS

Functionality

- An internet connection is required to access the app as no offline functionality is planned.
- Users must be logged in to view the app..

Usability

- App will only work on an android device.
- The UI should be intuitive enough for beginners and advanced users to use.

Reliability

- Should be little system downtime(>95% uptime).
- Battery should last at least 2 months.
- Arduino will send a message once movement has been detected over a certain threshold.

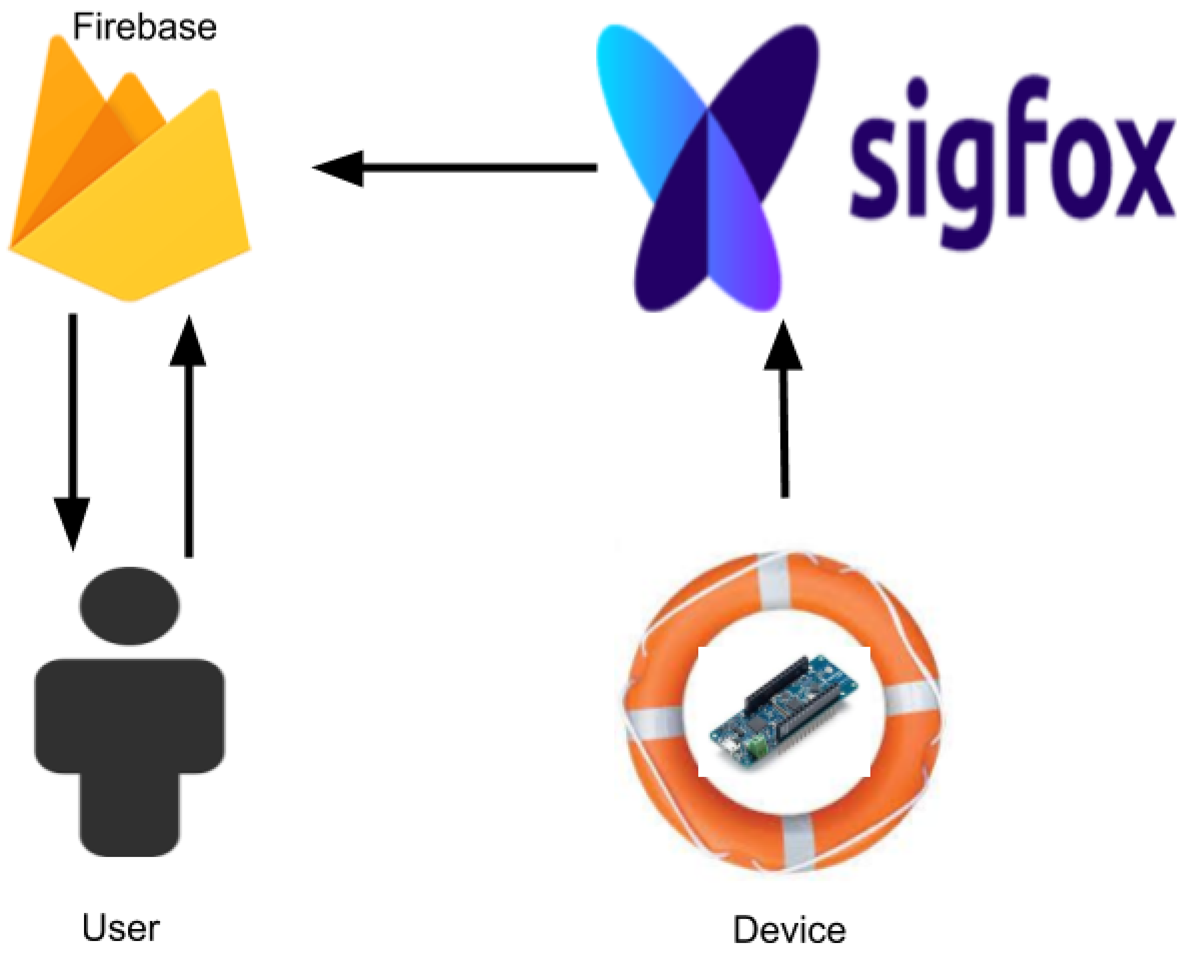
Performance

- The system response time should be as fast as possible ” According to Kissmetrics, 47 percent of visitors expect a website to load in less than 2 seconds, and 40 percent of visitors will leave the website if the loading process takes more than 3 seconds.”**[1]** Even though only admins will be accessing the app, the point still holds, the response time will have to be fast.
- The status of the lifebuoys should be updated immediately after an event has occurred.

Supportability

- The system should be easy to maintain and well tested (little bugs in the final product).
- The system should have good scalability in case the organisation/business adds extra lifebuoys in the coming years.

System Architecture



Iteration

Iteration 1:

- Have the arduino board connecting and transmitting data across the sigfox network.
- Have the accelerometer/altimeter sensing movement over a certain threshold and sending data back to a database.
- Get feedback.

Iteration 2:

- Implement feedback.
- Create a app that allows the user to login\create an account.
- Allow the user to create a new tracker and assign that tracer a location(maybe a pin on a map).
- Get feedback

Iteration 3:

- Implement feedback.
- Get push notifications sending to the user when an event happens.
- Create cloud function for sorting duplicate devices.
- Clean up any UI problems or bugs.

References

1. Exemplar Clinic Management System David Bryan[Online]Available at: https://docs.google.com/document/d/1pbnp3TnSPRBO7ReDHYZfZr9mW0qnjbbgNdwJD5iV_Vs/edit **[Accessed 08/01/17]**