



AEHelper

Project Proposal

Conor Thompson

National College of Ireland

BScH4SD Computing

X13466188

Table of Contents

1. Introduction.....	2
1.1 Purpose.....	2
1.2 Project Scope.....	2
1.3 Definitions, Acronyms, and Abbreviations.....	2
2. User Requirements Definition.....	3
3. Requirements Specification.....	3
3.1 Functional Requirements.....	3
3.1.1 User Registration/Login.....	3
3.1.2 User Emergency Call.....	6
3.2 Non-Functional Requirements.....	8
3.2.i Android Mobile Phone.....	8
3.2.ii Internet Connection.....	8
3.2.iii GPS Location.....	8
3.2.1 Performance/Response Time Requirement.....	8
3.2.2 Availability Requirement.....	8
3.2.3 Security Requirement.....	8
3.2.4 Reliability Requirement.....	8
3.2.5 Extensibility Requirement.....	8
3.3 Interface Requirements.....	9
4. GUI.....	10
4.1 Web Application.....	10
4.2 Android Application.....	10
5. System Architecture.....	11
6. QuickBlox Android Wrapper.....	13
7. Code Snippets.....	14
8. Testing.....	18
9. System Evolution.....	19
10. References.....	21
11. Monthly Journals.....	22

Introduction

Purpose

The purpose of this document is to lay out the requirements and specifications of this project, to better give an understanding of it's design and creation.

The intended users for this software is virtually anyone with an android device, as this is a utility used for emergency situation, which can happen to anyone at any time.

Project Scope

The scope of this project is to develop a system that allows users in an accident or emergency situation, to be directly in video and audio contact with an emergency consultant, who in their position, is more experienced in accidents and emergencies, and can advise the user how best to handle this situation, while the emergency services are on the way.

Many emergency services boast an 8 minute maximum time of arrival for accidents and emergencies. This, in many cases has been proved incredibly hard to achieve due to distance, weather, location etc. This software will help give the user the best advice and mindset to better the situation as much as possible while the services are en route.

Definitions, Acronyms, and Abbreviations

WebRTC	Web Real Time Communication (Google)
GUI	Graphical User Interface
WSS	Websocket Server

User Requirements Specification

An application to talk to someone who knows what to do in certain situations, whilst also showing them via video what is happening in this situation. The user will be looking for a way to communicate with a consultant, who will tell them what to do, in the instance that they are unsure what should be done.

Video Communication

The video communication will be achieved by utilising the Google WebRTC api, which allows for both video and audio communication between any amount of people at any given time.

Requirements Specification

Functional Requirements

User Registration/Login - Admin

3.1.1.1 Description & Priority

This function is one of the main pillars of the system architecture, as it allows the consultants and admins to log in and use the system to receive calls from the user. This also will allow admins to log in to use the admin section for analytics and adding new consultants. This is a high priority.

3.1.1.2 Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope

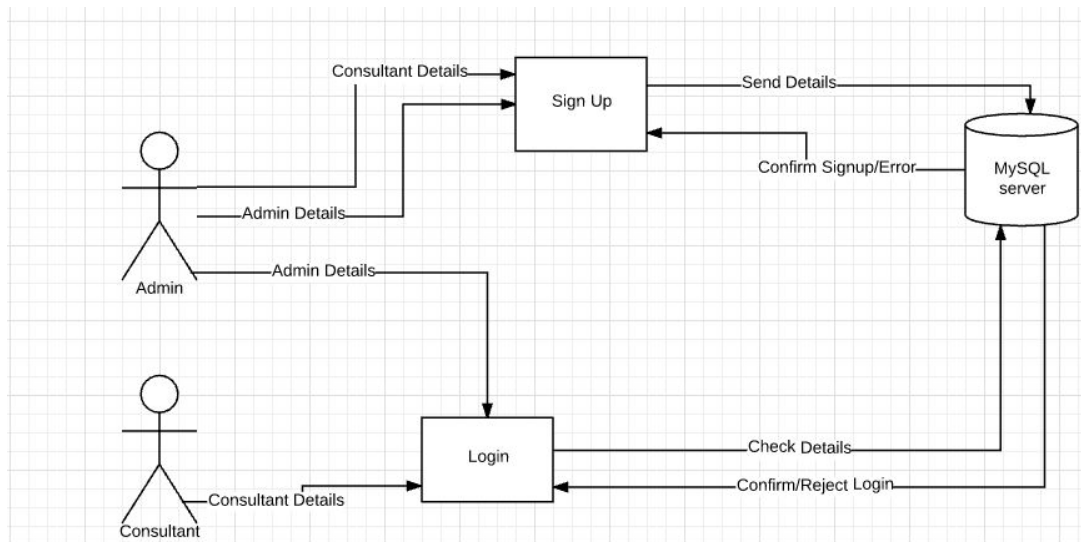
The scope of this use case is to demonstrate how the consultant and admin users accounts are made, and how their system works with logging in.

Description

This use case describes the process of signing up a Consultant or Admin and logging as each of them.

The Master Admin is giving the original login script, and they can then sign up other admins, which will then sign up new consultants.

Use Case Diagram



Flow Description

Precondition

The system is given a master admin script which will have the details for logging the original admin in.

Activation

This use case starts when an admin signs up another admin or consultant.

Main Flow

1. The system identifies the user details, and checks if they're valid
2. The admin is then signed up, and can then sign up a consultant.
3. The admin then signs up a new consultant.
4. The Consultant may now sign in.

Alternative Flow

A1 : Admin log in, analytic check

1. The Admin logs in
2. The Admin checks the statistics on the admin home page

Exceptional Flow

E1 : Incorrect credentials

1. The system checks for valid details,
2. The Admin/consultant types them in wrong
3. The System throws back an error, and doesn't log them in.

Termination

The system logs in, and the consultant can now see their homepage, as well as the admin.

Post Condition

The system goes into a wait state.

User Emergency Call

Description & Priority

This is when the user will click the emergency button, which will enable the WebRTC api to send the video and microphone input from the users device, and connect them to the WSS and peer them with a consultant.

Use Case

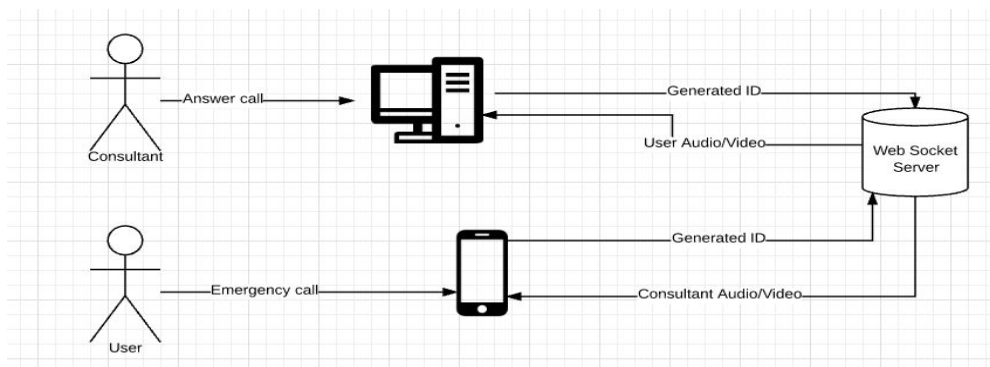
Scope

The scope of this use case is to show how the user will utilise the main function of this application, calling and initiating contact with the consultant.

Description

This use case describes the architecture used from the mobile app connecting to the WSS to the receiver on the consultants end, displaying one another's video and mic input.

Use Case Diagram



Flow Description

Precondition

The system is in a blank idle mode, awaiting the user to press the emergency call button.

Activation

This use case starts when a user presses the emergency call button.

Main Flow

1. The user presses the Emergency Call button
2. The system generates a new peer ID to the WSS
3. The WSS connects to the registered Consultant peerID
4. The Consultant answers a call
5. The pair are now connected via audio and video.

Alternative Flow

A1 : Consultant calls back

1. The Consultant presses the call back button
2. The System contacts the user peerID location
3. The user receives and answers call

Exceptional Flow

E1 : User hangs up

1. The User presses the hangup button
2. The WSS connection shuts off and is closed
3. The app tells you how long the call was.

Termination

The system closes the app.

Post Condition

The system goes into a wait state.

Non-Functional Requirements

Android Mobile Phone

- The minimum Android version is honeycomb [8].
- A Flashlight may also be useful, in the case that the situation is dark and the mobile doesn't have a flashlight.

Internet Connection

Unfortunately the user must have an internet connection, which limits the application in certain situations where the user may be out of reception. This is also a hindrance to using emergency services

GPS Location

Though not necessary, it is helpful in most situations to enable your GPS location, to allow the consultant to know exactly where you are, where they may guide emergency services.

Performance/Response Time Requirement

WebRTC Latency (Good connection)

Though WebRTC is a renowned API for having fantastic latency times, it is best if the user is in a situation where they have a decent internet connection, for clearer and more detailed video communication.

Availability Requirement

GPS and Mobile Data connection

As previously mentioned, the user is in the best use situation for this app when they have a decent connection to the internet.

Security Requirement

Encrypted device (optional)

For best security practices, the user should (optionally) encrypt their device, as sharing video and audio and GPS location can be subject to man-in-the-middle attacks, depending on the location.

Extensibility Requirement

WebRTC extension

The WebRTC api also allows for text communication, which may be a great addition in the case of certain dangerous emergency situations.

Interface Requirements

Webpage:

-WebRTC

The webpage will make use of the WebRTC api, allowing for peer-to-peer communication with users on the mobile app. This will serve as the consultants homepage, and the admin's analytics tools.

-Bootstrap [9]

The CSS will mainly be handled by the Bootstrap framework.

-PHP

The server-side logic will be handled by PHP, as the host is using apache2, and serves as the simplest language for the job it performs, which is to handle the admin and consultant login, and transfer data to the data warehouse.

App:

-WebRTC

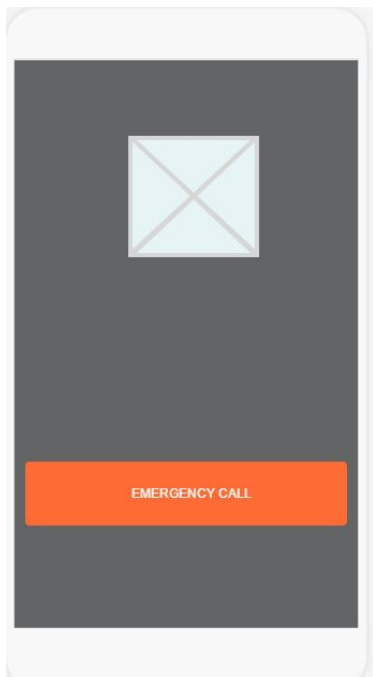
The android application will also use the webrtc api, as it needs to utilise the websocket server interaction to send the audio and video over to the consultant on the opposite end.

-PHP

There will be a php page for the android app to communicate with the hosted database, to send specific details about each call.

GUI - Android

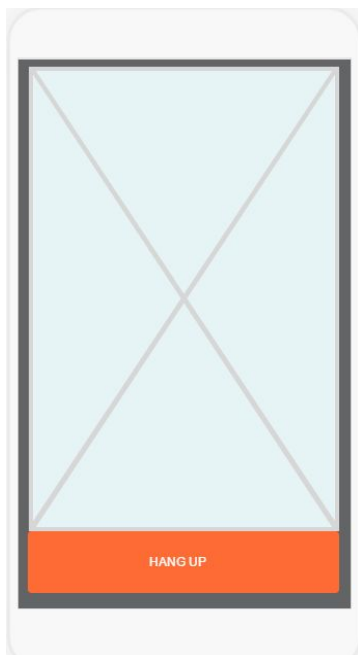
User Quick Homepage



This will be as simplistic as possible, to save for any unnecessary GUI features, the user will have a large button that is easily visible.

A small logo will also be there to serve as another button to press for the emergency button.

Camera Activity



The Camera activity is launched when connected to the consultant, this blank space will be where the consultants feed will be sent to you, so as to better communicate with peer-to-peer communication.

This will also include a hangup button, which will cancel out the connection.

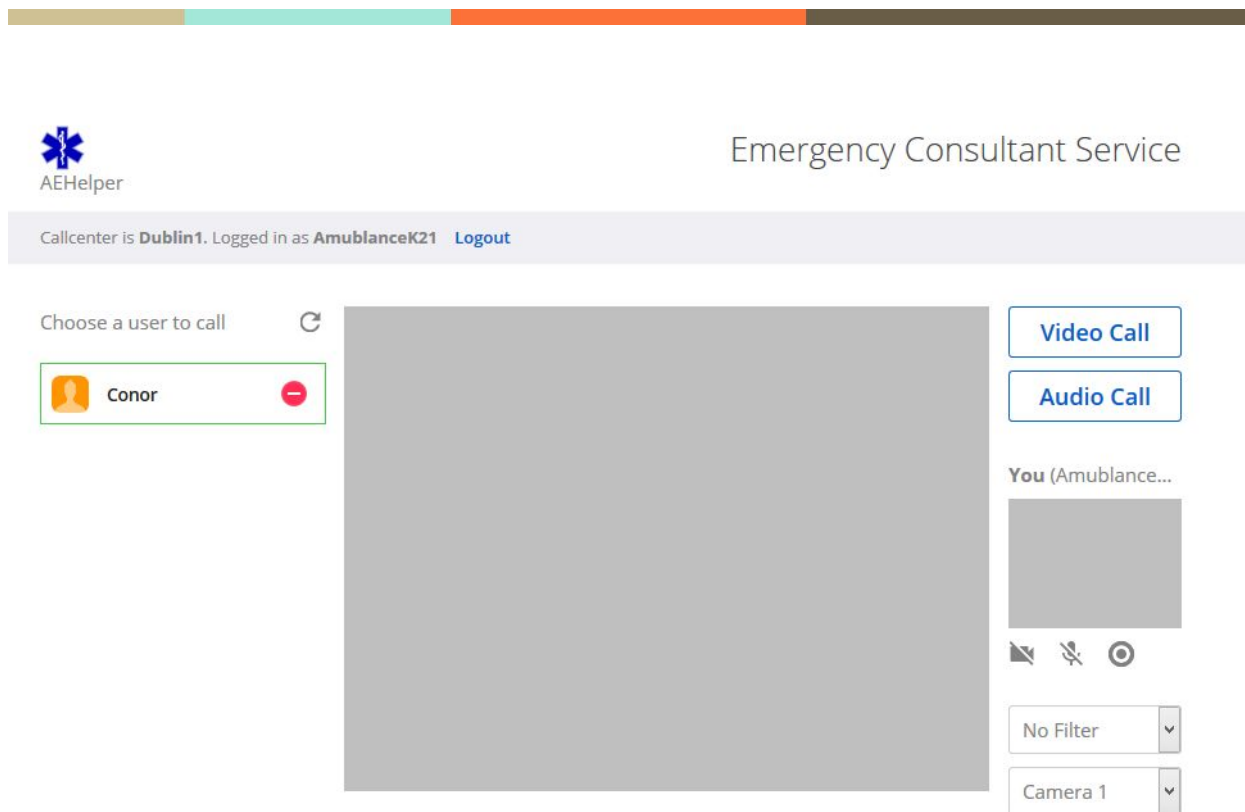
GUI - Web App



Emergency Consultant Service

Please enter your username and call centre name.

The Admin section will hold a login function, for the call centre the consultant or paramedic is residing in, and the username they have been given. Prior to this, the javascript will quickly grant them access to the WSS functionality, giving them access to anyone trying to call in that area, and the paramedic that may possibly be dispatched to the user of the application.



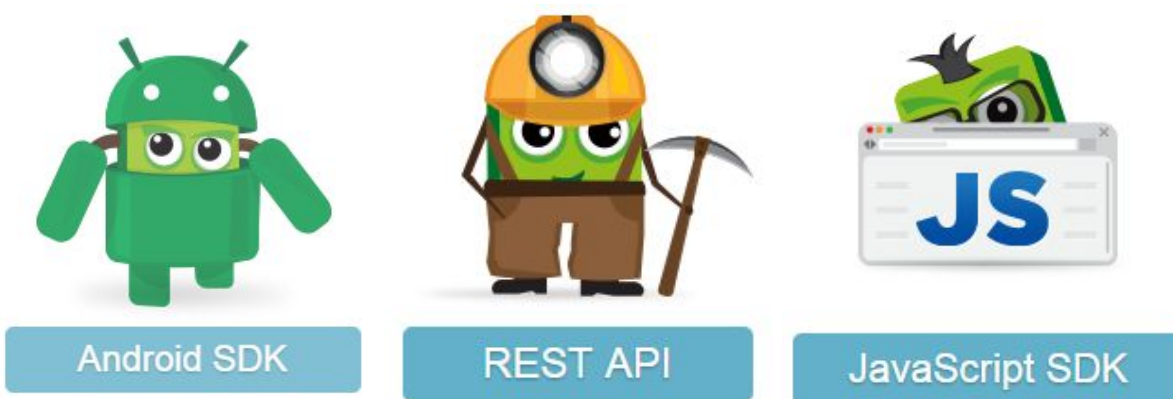
Once the admin has logged in, they are then shown the interface for the main application, this allows them to receive calls from users, paramedics and consultants alike, again, keeping a minimalist design as to not clutter anything important. The layout was achieved using mainly Bootstrap, Favicons and template images.

QuickBlox Android WebRTC

Due to the architecture of the WebRTC framework, Google have not provided adequate documentation for Android development with WebRTC api's.

This was partly resolved by a company named "QuickBlox" [1], who built an Android implementation of this framework.

They have provided their own documentation and development cycle for building a mobile application with this framework.



This provides a large amount of support for the developers to integrate the API into a fully functional Java application on Android.

Code snippets

In this section, we will be looking into some of the main methods and code applications in this project, and what they do.

Permissions

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

This snippet is quite self explanatory, the application will be asking the Android Operating System for access to the following hardware functionalities, including the Camera, Audio Settings, Microphone Recording, Internet access, and Write to external storage (SD card). Without these permissions, the application simply cannot carry out the main functions it is intended for.

WebRTC API access

```
static final String APP_ID = "961";
static final String AUTH_KEY = "PBZxXW3WgGZtFZv";
static final String AUTH_SECRET = "vvHjRbVFF6mmeyJ";
static final String ACCOUNT_KEY = "961";
```

In order to gain access to a lot of the features of the WebRTC api, we need a small amount of authentication in to utilise these features in an external application. This simply requires you to make your own account and you are then given randomly generated keys and secret tokens to gain access.

Android noteworthy methods

```
void onUserNotAnswer(RTCSession session, Integer userID);
```

This function is called when the user has tried calling the consultant, or vice versa, and the opposite end of the call has not answered within a specified time limit. The logic behind this will throw a pop-up explaining the user on the other end did not answer.

```
void onCallAcceptByUser(RTCSession session, Integer userID, Map<String, String> userInfo);
```

This is one of the main calls that will be used in the application, as it is called when the consultant, user, or paramedic, has pressed a button to answer the call. This function will be the gateway to the media exchange, and will hold the information of the caller, including user details (if available) and the geolocation.

```
void onReceiveHangUpFromUser(QBRTCSession session, Integer userID);
```

This function will be called when a user of the call session hangs up using the hangup button. This will end the session held between the consultant, paramedic, and user.

```
void onSessionStart(QBRTCSession session);
```

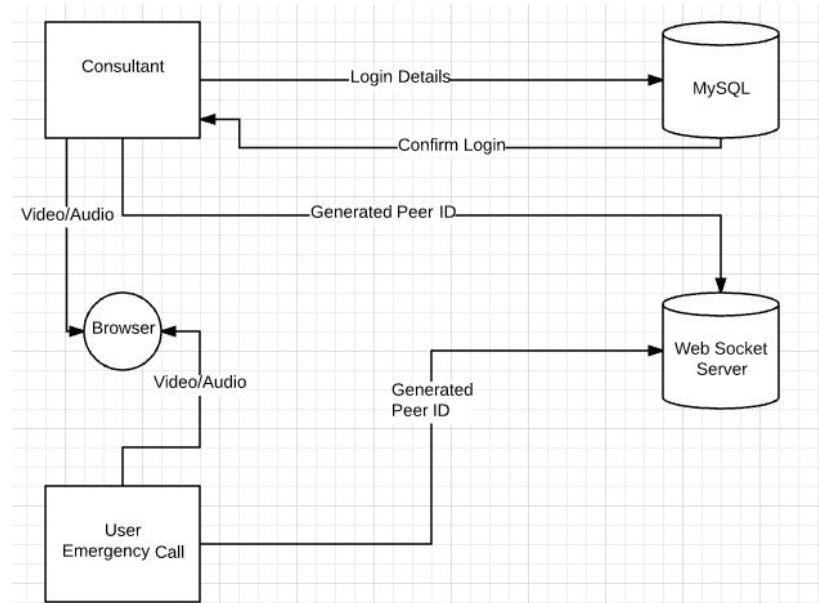
This function is called when the user has started a call, and directs the user to the id of the call centre in use, and the consultant will then answer the call,

```
void onSessionClosed(QBRTCSession session);
```

This function is called upon closing the session of the call between the user and consultant, this can also be overridden to hold user information.

System Architecture

System Diagram



The current system architecture consists of the mobile application and admin webpages connecting to the websocket server, and peering with one another to allow a socket stream of media to be transferred between one another.

This is quite minimal architecture, so as for extra challenge there is an analytics data warehouse included which will store all of the statistics gained from the calls and consultant inputs.

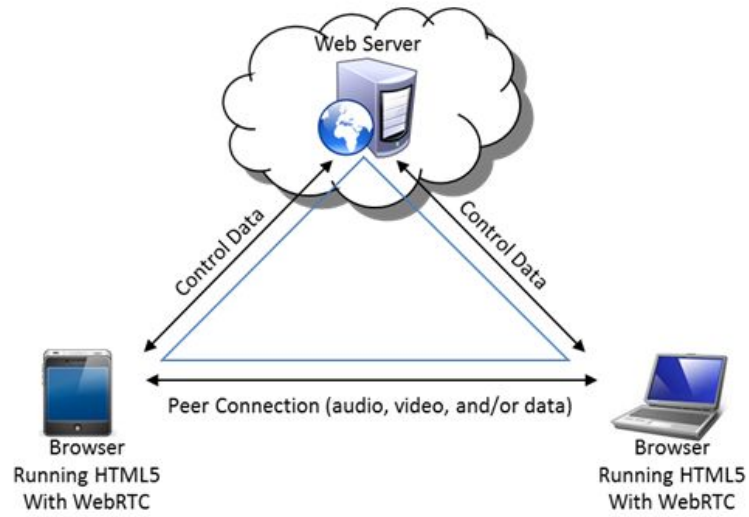
Application Programming Interfaces

One of the main utilities of this project is sampled off the architecture of Google's Real Time communication api, webRTC [3].

"WebRTC is a free, open project that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs. The WebRTC components have been optimized to best serve this purpose."

-[<https://webrtc.org/>]

This api allows us to remotely connect devices to a websocket server, and transmit media data between all of these devices. Including text, video, audio and files.



Testing

Unit Testing

For the unit testing, I tested the login, registration and camera data transfer in both the web application and mobile application. This was done using PHPUnit [6][7] and JUnit [5][11].

```
public class CallAcceptByUserTest {  
    private CallActivity callActivity = new CallActivity();  
  
    @Test  
    public void testAcceptByUser() throws Exception {  
        assertEquals("Test accept call failed", true, callActivity.onConnectedToUser(session,12));  
    }  
}
```

```
final class LoginTest extends TestCase  
{  
    public function testLoginForAdmin(): void  
    {  
        $this->assertInstanceOf(  
            DBHandler::class,  
            DBHandler::Login('123@user.com', 'password123')  
        );  
    }  
}
```

End-to-End Testing

The end-to-end testing was accomplished using the Selenium tool and utilising ProtractorJS [2], this allowed me to build scripts that acted as a user and test out the user experience, and to check how many holes or faults I could find.

Testing the login looks similar to this in ProtractorJS:

```
spec.js
describe('login page', function() {
  browser.driver.get('localhost/Login/');
  it('should render login page', function() {
    // Checking the current url
    var currentUrl = browser.driver.getCurrentUrl();
    expect(currentUrl).toMatch('/Login/');
  });
  it('should sign in', function() {
    // Find page elements
    var userNameField = browser.driver.findElement(By.id('username'));
    var userPassField = browser.driver.findElement(By.id('password'));
    var userLoginBtn = browser.driver.findElement(By.id('loginbtn'));

    // Fill input fields
    userNameField.sendKeys('test@user.com');
    userPassField.sendKeys('1234');

    // Ensure fields contain what we've entered
    expect(userNameField.getAttribute('value')).toEqual('test@user.com');
    expect(userPassField.getAttribute('value')).toEqual('1234');

    // Click to sign in - waiting for Angular as it is manually bootstrapped.
    userLoginBtn.click().then(function() {
      browser.waitForAngular();
      expect(browser.driver.getCurrentUrl()).toMatch('/Admin/Caller/');
    }, 10000);
  });
});
```

In this script, we are simply informing the browser to fill in the login forms with test user data, that will then be sent to the server and checked. Upon this being the correct credentials, the script expects the browser to redirect to the Caller App.

Usability Testing

To ensure the usability of this application, I had a number of different potential users, of various ages, backgrounds, and technical competence, to use the main features of this application. This was monitored in a “Think out loud” dialogue recount.

Colleen O’Neill - EMT Trainee

“As someone in this area of work, I can see this application being a tremendous help, we stay in contact as much as possible, but have no solid contact with the caller themselves. This will make a massive difference, as we can advise what to do while we are on the way to the accident area”

Mary-Kate Findon - Journalist

"I can see this application being a great idea, but too hard to actually set up and have people use regularly, people are too used to dialing 911, adding an app will just confuse things"

Eoin Daly - Secondary Level Student

"I would probably use it if I saw something happen and we already called the ambulance, my neighbour had a heart attack last year and it took the ambulance 25 mins to get here, none of us really knew CPR, so having this [application] would have really helped"

Jean O'Keeffe - OAP

When asked would she be comfortable using it:

"I don't own a smart-phone, but from using it so far it seems as easy as dialling 999, you just press a button and someone is now talking you through your problem"

System Evolution

Scalability

This app has the potential to be used in a wide amount of situations, with regards emergencies. The standard communication in these services is always audio, though video is an important structure in the emergency service sector. As it allows admins to analyze and validate the emergency responders activity.

Evolution of communication

As the developments in human communication over the coming decade, so shall the use of RTC, as it is now becoming the main form of structural communication.

System evolution

The current architecture is small, simple, and useful. Anything to be added would follow the same suit, as such text communication would be the next step, or simply mobile-esque communication like that of Viber or Google Hangouts [4].

References

- [1] "Quickblox Developers (API Docs, Code Samples, SDK)". Quickblox.com. N.p., 2017. Web. 4 Apr. 2017.
- [2] "Protractor - End-To-End Testing For Angularjs". Protractortest.org. N.p., 2017. Web. 10 May 2017.
- [3] "WebRTC Home | WebRTC". WebRTC.org. N.p., 2017. Web. 10 May 2017.
- [4] "Google Hangouts". Hangouts.google.com. N.p., 2017. Web. 10 May 2017.
- [5] "What Is Java And Why Do I Need It?". Java.com. N.p., 2017. Web. 10 May 2017.
- [6] "Phpunit – The PHP Testing Framework". Phpunit.de. N.p., 2017. Web. 10 May 2017.
- [7] "PHP: What Is PHP? - Manual". Php.net. N.p., 2017. Web. 10 May 2017.
- [8] "Android". Android. N.p., 2017. Web. 10 May 2017.
- [9] Mark Otto, and Bootstrap contributors. "Bootstrap · The World's Most Popular Mobile-First And Responsive Front-End Framework.". Getbootstrap.com. N.p., 2017. Web. 10 May 2017.
- [10] "What Is Latency? - Definition From WhatIs.Com". WhatIs.com. N.p., 2017. Web. 10 May 2017.
- [11] "JUnit - About". Junit.org. N.p., 2017. Web. 10 May 2017.

Monthly Journals

Month: September

My Achievements

This month, I started working a small amount on the architecture of my project. Getting a overview of what technologies may be needed on top of what I have already planned.

My contributions to the projects included: Decided on MySQL for the database, WebRTC for the socket layer programming, and android for the mobile application.

My Reflection

I felt that the decisions I made were the right ones.

Intended Changes

Next month, I will try to have a working prototype of the mobile app, before working on the desktop application.

I realised that I need to find time to plan and develop the project.

Month: October

My Achievements



This month I put a lot of time and resources into studying into media streaming with sockets, and how I can best layout the architecture of the android project.

I was able to nearly complete the web site for the consultant to use, with some remaining interface and graphing left to do. I have a much further understanding now of how the project timing and development will play out.

My Reflection

Working on the architecture was a better decision instead of running head first into programming the features one after the other.

Though planning has gone well, I still need to manage my time with the other module projects better.

Intended Changes

Next month I intend on having the website finished, and a good stab at the base gui for the android project, then start working on the WebRTC and socket interfaces.


I understand that I have other projects and study to do with other modules, this will be helped when I start having meetings with Lisa, my supervisor.

Month: November

My Achievements

This month, I researched into compression algorithms to better exchange the media data between peers. This is an important feature as the visibility needs to be optimal for the consultant to give educated and clear advice.

My Reflection



From looking into compression algorithms I found a number of sources dealing with the Websocket Server, that may help in more optimal data exchange. Much of it comes down already to internet speed and android version. Which is unchangeable on my side.

Intended Changes

Next Month I intend on having a fully functioning admin section and GUI, this will be done using a bootstrap layout, and the WebRTC framework.

Month: December

My Achievements

This month, I was able to deploy a working admin section, and have most of the GUI completed. The WSS is working and now all I need is a working mobile application.

My Reflection

I now have my documentation in better order, and have a developed on the admin section, this will now be extended with more features as the months go on, after discussion with my supervisor.


Intended Changes

Next Month I intend to have an online meeting with a member of the WebRTC framework development team, and have a discussion about the development cycle and path in the Android world.

I realise the short time I have in developing a fully functioning peer-to-peer application on mobile to my standard, so I will have to prioritise it this semester.

Month: January

My Achievements



This month, I was able to build an android application that has the capabilities of peer to peer communication via webRTC.

My contributions to the projects included building the android application, reformatting certain aesthetics in the browser application.

My Reflection

After getting in contact with a WebRTC dev, I have clairvoyance on how I will finish this project efficiently.

Intended Changes

Next month, I will try to refine the websocket server, usability of the app, design, and documentation.

Month: February

My Achievements

This month, I was mostly over encumbered with other projects, but managed to work a good bit on the mobile application. The application is now on the way to being finished as of the end of last month I had a chat with one of the developers of the WebRTC framework, who helped me in grasping the Android section of my project.

My Reflection

I need to work more on my documentation and testing.

Intended Changes

Next month I hope to have the testing and the documentation fully completed, in both the usability, frontend, backend and manual.

Month: March



My Achievements

This month, I began reformatting the layout of the project, including the Android and Browser application. I have also begun using a new logo for the main splash of the applications.

My Reflection

There has been a number of changes in the WebRTC api that I now have to implement into the application, this caused a minor hiccup in the development time schedule.

Intended Changes

Next Month, I hope to have the entire application fully functioning and to my standard of aesthetics and completion.