Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: Sam Gormley
Student ID: x12467312
Supervisor: Arghir Moldovan

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: Sam Gormley

Date: 11/05/16

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Complete the sections above and attach it to the front of one of the copies of your assignment,

What constitutes plagiarism or cheating?

The following is extracted from the college's formal statement on plagiarism as quoted in the Student Handbooks. References to "assignments" should be taken to include any piece of work submitted for assessment.

Paraphrasing refers to taking the ideas, words or work of another, putting it into your own words and crediting the source. This is acceptable academic practice provided you ensure that credit is given to the author. Plagiarism refers to copying the ideas and work of another and misrepresenting it as your own. This is completely unacceptable and is prohibited in all academic institutions. It is a serious offence and may result in a fail grade and/or disciplinary action. All sources that you use in your writing must be acknowledged and included in the reference or bibliography section. If a particular piece of writing proves difficult to paraphrase, or you want to include it in its original form, it must be enclosed in quotation marks

and credit given to the author.

When referring to the work of another author within the text of your project you must give the author's surname and the date the work was published. Full details for each source must then be given in the bibliography at the end of the project

Penalties for Plagiarism

If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the college's Disciplinary Committee. Where the Disciplinary Committee makes a finding that there has been plagiarism, the Disciplinary Committee may recommend

- that a student's marks shall be reduced

- that the student be deemed not to have passed the assignment
- that other forms of assessment undertaken in that   academic year by the same student be declared void
- that other examinations sat by the same student at the same sitting be declared void

Further penalties are also possible including

- suspending a student college for a specified time,
- expelling a student from college,
- prohibiting a student from sitting any examination or assessment.,
- the imposition of a fine and
- the requirement that a student to attend additional or other lectures or courses or undertake additional academic work.

National College of Ireland

BSc in Computing

2015/2016

Sam Gormley

X12467312

X12467312@student.ncirl.ie

Yellow Umbrella Tours app

Technical Report

National College of Ireland

Table of Contents

Executive Summary

This project, the Yellow Umbrella Tours app, will be an Android app initially. It will make use of an online database in order to facilitate guide registration and log in, as well as allow guides to enter notes and data concerning their tours, and will also allow tourists to book tours, give feedback and get in contact with tour guides at Yellow Umbrella Tours in order to receive recommendations, directions or just general help without forcing one time tourists to create accounts and log in for what could only be a weekend trip.

1 Introduction

1.1 Background

The idea for this project came from a previous project that I had done involving a tourism app, and noticing a pattern in general tourism apps. I noticed that, in my own app and others, the entire experience was centred on the tourist, with little to no thought given to the guide. For example, Tripadvisor allows tourists to rate their experience and give some feedback, while allowing tour operators some contact with potential customers. I feel that tripadvisor is a good planner and can give you some background on where you’re going and what to do, but is a planner and nothing more, and I would not call it a travel companion app. Yelp is much the same. Another issue I find with these apps is that while they may mark out certain things to do, they fail to immerse you in these activities, providing you with only a waypoint. From this, I began to research the various tourism based apps in order to ascertain what was missing from them, and what I could do to improve them. I decided to model the app on a single tour company, for the sake of simplicity.

My idea for this app was also inspired by a tour company operating in the City Centre, currently called Dublin Free Walking Tours, but currently transitioning and rebranding to Yellow Umbrella Tours, which is the name of this App. I worked with 2 of the guides initially to form the requirements, then began talking to other guides once I began development in order to ascertain what they would like to see and began asking for their feedback on the app once some of them were asked to test beta versions of the app. I also asked the guides about the non-functional requirements of the app, and the things that could make their lives a bit easier and make some of their processes more efficient, for example, having either a contact form or some kind of messaging function, which would allow tourists to contact guides outside of tour time. Through my conversations with the guides, we were able to map out what features would be added to the app, what kinds of users we would have and how they could benefit from the different features of the app. From here, I began development of the app, checking back in with the guides from time to time in order to demo whatever I had done for the app and to ascertain the next step of the development.

1.2 Aims

My aim for the project is to create an Android app that will be useful for tourists to find free tours and points of interest, and for tour guides to assist other guides by leaving helpful notes about tours, route changes or traffic issues etc. by sending the information back to a database. By the end of this project, I hope to have a fully functioning Android app that will have map functionality, user registration and a contact form or message function which will allow tourists to get in contact with a guide who may be able to answer their queries. I also wanted to allow tourists to book tours using the app. My main aim is to create an immersive, interactive tourist app that lends itself to informative trips and can also assist the tour guide.

1.3 Technologies

This project makes use of java for Android, as I felt that it would be easy enough to work with as well as providing the functionality required for the project. The Google Maps API will be used in the app in order to provide waypoints and directions to users. I chose the Maps API as it is easy to integrate, work with and personalise. The Maps API is implemented using the API key and a class that will call in the map and specify any locations or markers. This will be discussed in more detail in the implementation section. I also decided to use MySQL with PHP files, as well as creating an online database using 000Webhost, in order to get a log in database working in my app. I used MySQL and PHP because I felt that they would be easy to integrate into an Android app, and MySQL would allow me to edit and work on tables without much difficulty. For the chat function, I will be attempting to implement the support function from Intercom.io. Intercom makes use of Google Cloud Messaging(GCM) in order to send push notifications to the users phone, and would allow the user to get in contact with a guide directly, in real time. I had considered an alternative to Intercom, known as hotspot.io, but decided to go with intercom, both for the helpfulness of their documentation, and also as a financial measure. Intercom.io offers the package that would be relevant for this project for $49 a month, whereas the same product from hotline.io would cost $300 a month. In terms of services offered, there was not a great deal that hotline.io provided that intercom did not, especially in relation to my project and what I would need, so spending the extra $251 just seemed unnecessary. If intercom proves to be too troublesome to implement, I will instead implement a contact form as a sort of ticketing system, in which tourists will fill out a contact form of name, email, phone number and query, which will then be sent into the database and followed up by a guide who will regularly check this list and provide support to the tourists who need it.

In order to facilitate registrations, log ins and any other functions that require any networking functions, e.g sending data to databases or receiving any information from the server, I will be making use of the http library **Volley**. This can be seen in any classes that involve data being sent to or retrieved from the database, in the final method called **RequestQueue**. This allows the class in which it is implemented to call on another java class in order to interact with the database(for example, within my project, the class with the request queue method may be called GuideRegister, and the class being called in the method would be called GuideRegisterRequest). Volley allows you to send information to and from databases via your app by linking to the relevant .php file and sending the information to it first, before passing it off to the database.

In order to handle the dependencies and libraries, I used the module level build.gradle and the app level build.gradle in order to import any libraries or dependencies and repositories that I would need in my project, and then enabled the use of these libraries by importing them into each individual class. This saves the trouble of manually downloading whatever libraries you need and then importing them as modules in Android Studio in order to make use of them. As an aside, I used cloud storage sites like Google Drive and Dropbox in order to store versions of my project during development, both as a backup and in order to integrate certain features into the app that I wasn’t too confident in, so as not to ruin the entire project and have to start over.

1.4 Structure

Brief overview of each chapter

2 System

2.1 Requirements

2.1.1 Functional requirements

- App must have login and Database functions
- App must have a map interface detailing route of tours.
- Guide must be able to record notes concerning the tour in the database.
- User must be able to select their preferred tour
- User must be able to give feedback on their chosen tour.
- Users must be able to book tours using the app.

2.1.2 App must have database and login function

2.1.2.1 Description & Priority

The app must allow the user to create an account and login in order to store favourite tours, stops, etc. Most important requirement.

2.1.2.2 Use Case

Scope

This will be the

Description

This use case describes the process involved in the login and database saving.

Use Case Diagram

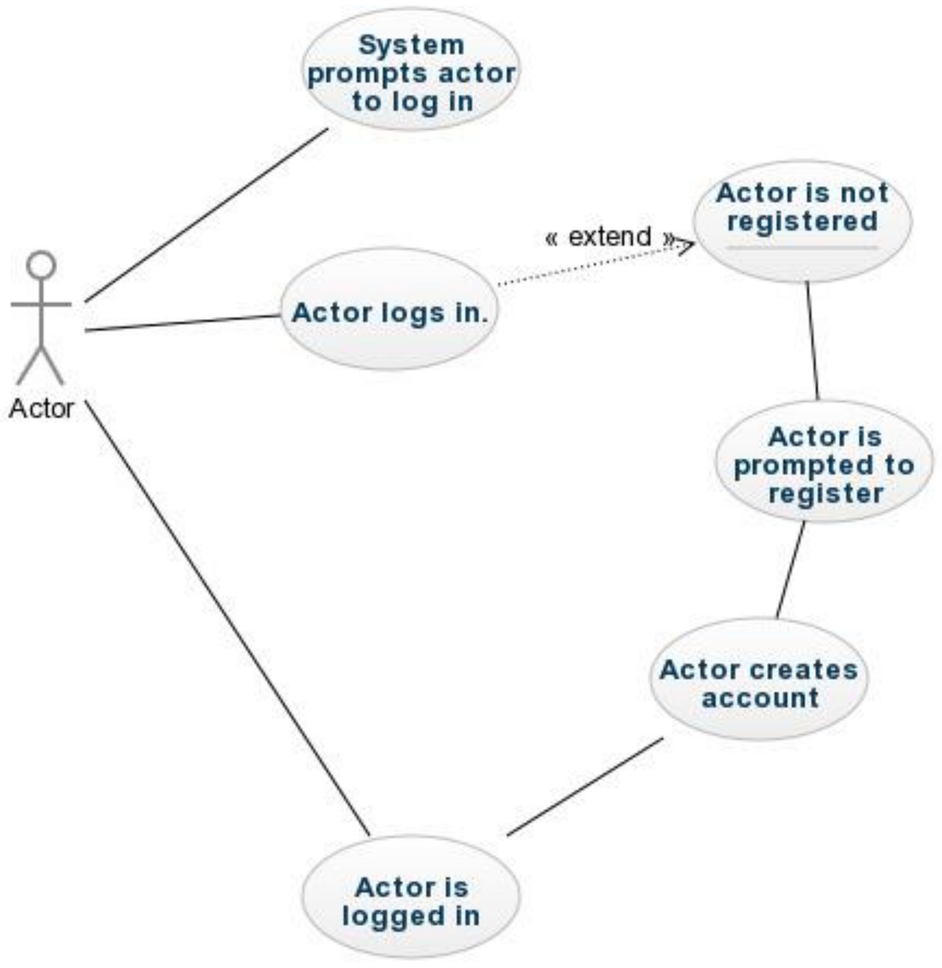


Figure 1 Login Usecase

Flow Description

Precondition

The system is in initialisation mode.

Activation

The use case starts when the actor starts up the app.

Main flow

1. The actor activates the app
2. The system prompts user to log in
3. The Actor Logs in

Alternate flow

A1 :

1. The User is not registered
2. The system prompts user to register
3. User Registers

Exceptional flow

E1 : The system prompts user to log in

4. The actor enters their username and password
5. The password is incorrect
6. The system refuses log in.

E2: The user is a tourist and does not have to log in

1. The actor opens the app on the home page
2. The actor selects continue as tourist
3. The actor is taken to

Termination

The user is brought to the home screen.

Post condition

The system goes into a wait state

2.1.3 App must allow user to choose route

2.1.3.1 Description & Priority

The app must have a map to illustrate the different tours and routes. As the pre-tense of this app is about the tours, this is very important.

2.1.3.2 Use Case

Scope

The scope of this use case is to show the required functionality of the map interface

Description

This use case describes the process of the actor using the map interface

Use Case Diagram

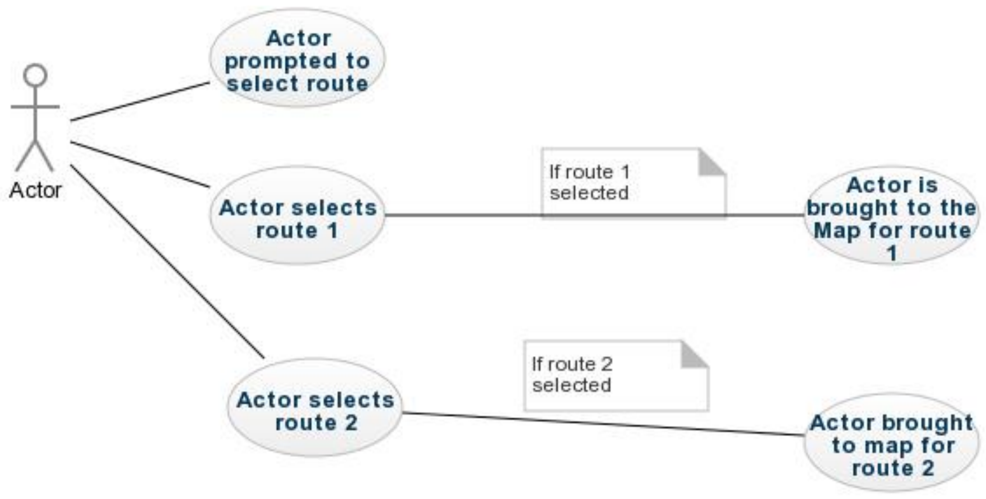

Actor
Actor prompted to select route
Actor selects route 1
If route 1 selected
Actor is brought to the Map for route 1
Actor selects route 2
If route 2 selected
Actor brought to map for route 2

Figure 2 Route select Use Case

Flow Description

Precondition

The actor is logged into the system and at the home screen.

Activation

This use case starts when the actor tries to activate the map function from the homescreen

Main flow

4. The system prompts the user to choose a route.
5. The <Actor> chooses route 1

6. The system brings the actor to the map page
7. The <Actor> views the route on the map

Alternate flow

A1 :

The actor chooses route 2

7. The system brings user to the map screen for the second route
8. The use case continues at position 3 of the main flow

Termination

The Actor is at the map screen, viewing the route.

Post condition

The system goes into a wait state

2.1.4 Guide must be able to record information about the tour in the databse

2.1.4.1 Description & Priority

The app must allow the guide to send useful information about the tour back to the database.

2.1.4.2 Use Case

Scope

This scope of this use case is to show the required functionality for this requirement.

Description

This use case describes the process involved in the guide recording the information.

Use Case Diagram

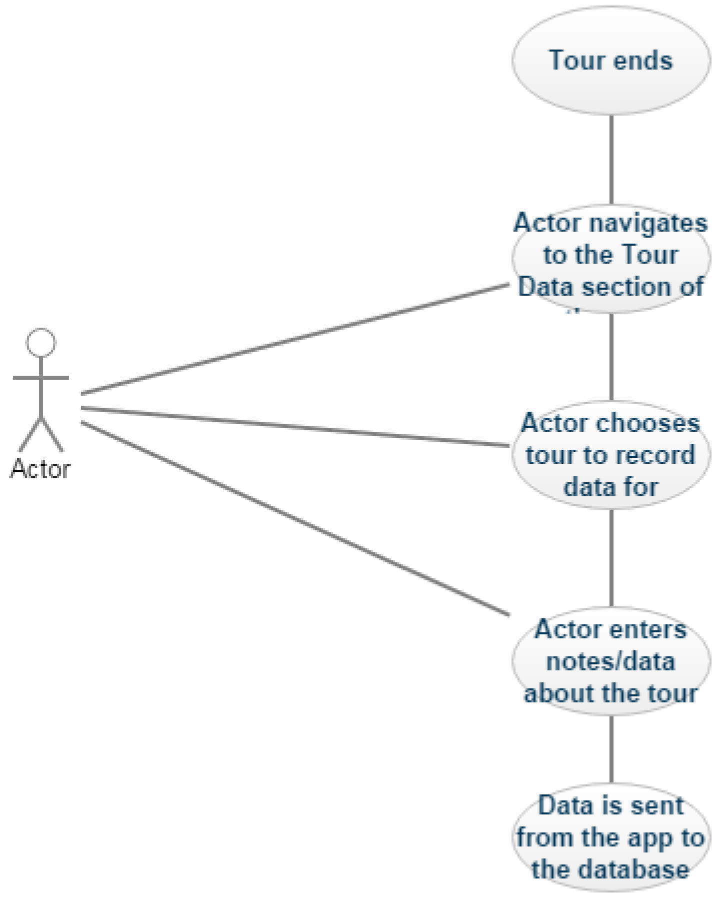

Tour ends
Actor navigates to the Tour Data section of
Actor chooses tour to record data for
Actor enters notes/data about the tour
Data is sent from the app to the database
Actor

Figure 3 Guide data recording use case

Flow Description

Precondition

The system is in initialisation mode.

Activation

The use case starts when the actor finishes a tour.

Main flow

8. The actor activates the part of the app that allows the guide to record details of the tour

9. The system prompts actor to record details of the tour
10. The Actor enters the details of the tour
11. The details of the tour are sent to the database

Termination

A thank you message is shown

Post condition

The system goes into a wait state

2.1.5 User must be able to provide feedback on their tour

2.1.5.1 Description & Priority

The app must allow the user to vote for their favourite tour. Add interactivity. Mildly important.

2.1.5.2 Use Case

Scope

This scope of this use case is to show the required functionality for this requirement.

Description

This use case describes the process involved in the user voting for their favourite tour.

Use Case Diagram

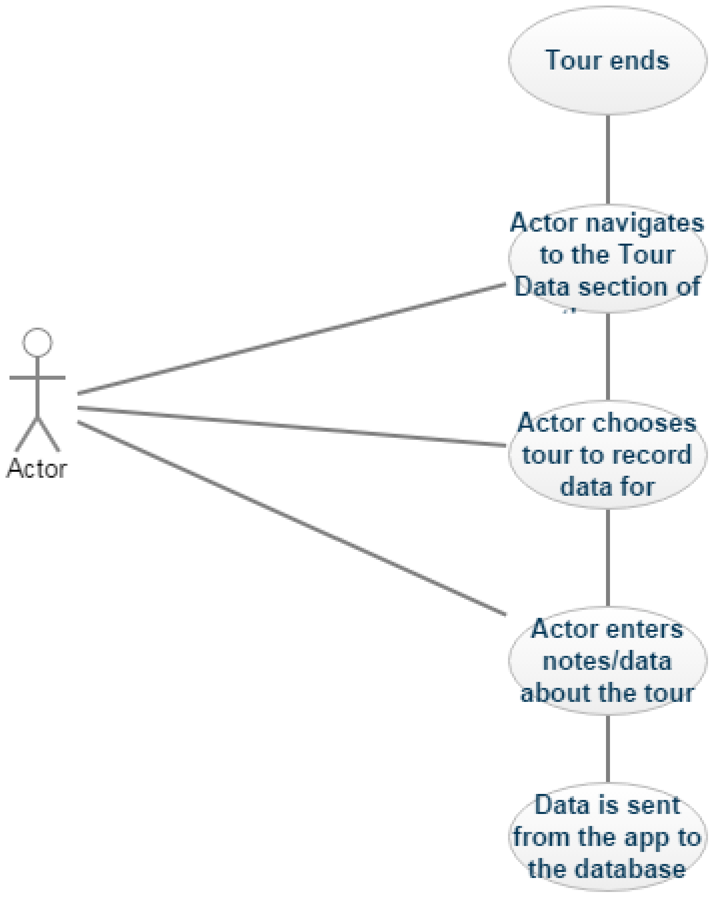

Tour ends
Actor navigates to the Tour Data section of
Actor
Actor chooses tour to record data for
Actor enters notes/data about the tour
Data is sent from the app to the database

Figure 4 Tour feedback usecase

Flow Description

Precondition

The system is in initialisation mode.

Activation

The use case starts when the actor navigates to the feedback section of the app.

Main flow

12. The actor navigates to the feedback section of the app.
13. The system prompts user to feedback for their tour
14. The Actor records their feedback for the tour.

Alternate flow

A1 :

9. The Actor chooses not to vote at this time.
10. The system takes the user back to the home page

Termination

The user is brought to the home screen.

Post condition

The system goes into a wait state

2.1.6 User must be able to book tours using the app.

2.1.6.1 Description & Priority

The app must allow the user to book tours while using the app. While useful for the tourist, it is not a main functionality for the App.

2.1.6.2 Use Case

Scope

This scope of this use case is to show the required functionality for this requirement.

Description

This use case describes the process involved in the user booking a tour.

Use Case Diagram

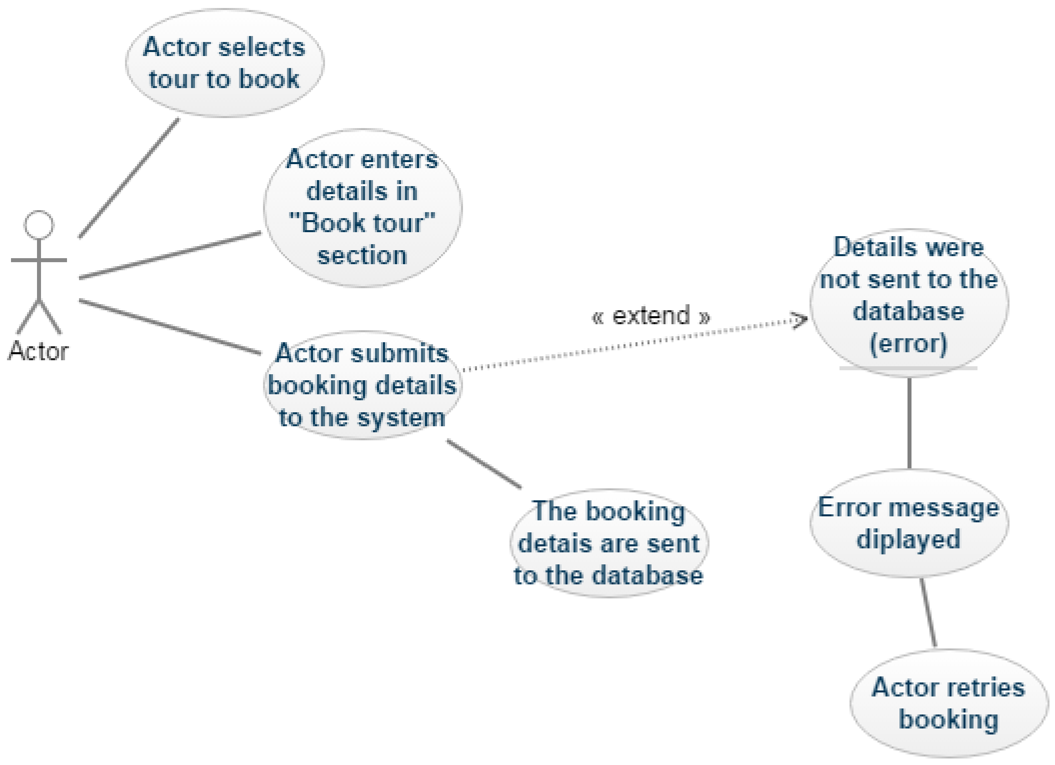

Actor
Actor selects tour to book
Actor enters details in "Book tour" section
Actor submits booking details to the system
« extend »
Details were not sent to the database (error)
The booking detais are sent to the database
Error message diplayed
Actor retries booking

Figure 5 Booking use case

Flow Description

Precondition

The system is in initialisation mode.

Activation

The use case starts when the actor selects the book tour option

Main flow

15. The actor selects book tour option.
16. The system prompts user to select what tour they want to book
17. The Actor selects tour
18. The system prompts for details
19. The Actor tour details.
20. The System send the data to the database

21. A confirmation message is displayed

Exceptional flow

E1 : The booking cannot be processed

11. The actor enters their details
12. The info is not sent to the database
13. An error message is displayed
14. The actor retries their booking.

Termination

The actor is shown a success message.

Post condition

The system goes into a wait state

2.2 Non-Functional Requirements

2.2.1 Performance/Response time requirement

App must be quick and responsive, with very little lag time when navigating pages, etc.

2.2.2 Security requirement

App must be secured by passwords and require log in to begin session. Payment processing must be secure.

2.2.3 Reliability requirement

App must attain above 95% uptime.

2.2.4 Extendibility requirement

App will be ported to iPhone in the future

2.2.5 Resource utilization requirement

App will access all data from the code or the database

2.2.6 Data requirements

The app must be able to process user input, such as login details, booking details and messages, such as the intercom messages.

2.2.7 User requirements

The user (client) feels that this app should have a map interface, a feedback mechanism and a function that will allow users to book tours, as well as a feature that will assist his guides. The map interface is dealt with by using the Google maps API, the feedback is answered by the features that will allow the users/tourists to rate their tour guide and their favourite tour route, and the assistance for the guides is provided via the contact form within the app, which will allow the guide to answer any queries from tourists outside of tour time. The guides will also have a section where they can enter notes and details about their tours to the database for the benefit of the other guides. For example, a guide could remark that a particular stop may be overcrowded due to an event, or that traffic could be heavy at one of the stops near the road. This can then be seen by the guide who will also be looking after support, who can then disseminate the information to the other guides.

2.3 Design and Architecture

This app is made up of a main central Android App, an online hosted database from 000Webhost and fragments within the app itself. This will be accessed by an external user who has the app on their device.

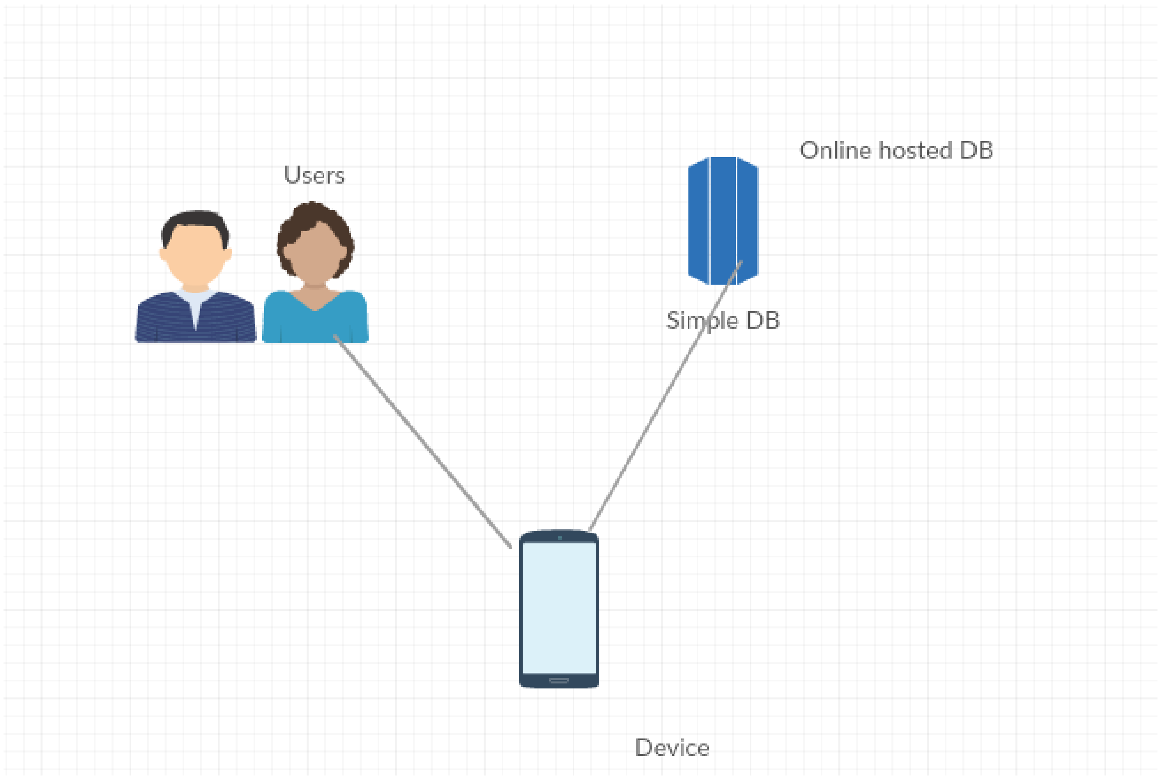


Figure 6 High Level Design and Architecture

2.4 Implementation

One of the main classes in this app will be the class which contains and adds detail to the map to be used by tourists. This class will contain the code which calls the map and places a specific location or marker on the map, and uses another class in which the API key is used, allowing me to use the Google Maps API within my project. Below is the code for the class that will call in the map and specify the location of a marker on the map, along with a custom image for the marker:

```
public class Map extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_map);
        // Obtain the SupportMapFragment and get notified when the map is ready
to be used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
    @Override
    public void onMapReady(GoogleMap map) {
        LatLng dublin = new LatLng(53.349, -6.260 );

        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
&& ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

            return;
        }
        map.setMyLocationEnabled(true);
        map.moveCamera(CameraUpdateFactory.newLatLngZoom(dublin, 14));

        map.addMarker(new MarkerOptions()
                .title("The Spire")
                .snippet("The starting point for all of our tours!")
                .position(dublin)
                .icon(BitmapDescriptorFactory.fromResource(R.drawable.marker)));
```

https://developers.google.com/maps/documentation/android-api/

The class containing the API key has very little, other than the key itself, and is shown below:

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    <string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">
        AIzaSyDcSr5AsoqnlM3qRUeO1zktUzAufEe2y60
    </string>
    </resources>
```

The next thing to implement would be the login/register database for guides, which will use MySQL, PHP files and be integrated into the app using java code. MySQL is used to edit and access the tables of the database, while the PHP files are used to send the data to the database. An example of one PHP file, named ‘GuideRegister.php’, which sets up the basic connection to the database, is shown below:

```
<?php
    $connect = mysqli_connect("mysql9.000webhost.com", "a8066174_SamG",
"B3taBas3", "a8066174_YellowB");

    $gName = $_POST["gName"];
    $gEmail = $_POST["gEmail"];
    $gPassword = $_POST["gPassword"];
     function registerGuide() {
         $connect, $gName, $gEmail, $gPassword;
        $statement = mysqli_prepare($connect, "INSERT INTO guide (gName,
gEmail, password) VALUES (?, ?, ?)");
        mysqli_stmt_bind_param($statement, "sss", $gName, $gEmail, $gPassword);
        mysqli_stmt_execute($statement);
        mysqli_stmt_close($statement);
    }
    function gEmailAvailable() {
        global $connect, $gEmail;
        $statement = mysqli_prepare($connect, "SELECT * FROM guide WHERE gEmail
= ?");
        mysqli_stmt_bind_param($statement, "s", $gEmail);
        mysqli_stmt_execute($statement);
        mysqli_stmt_store_result($statement);
        $count = mysqli_stmt_num_rows($statement);
        mysqli_stmt_close($statement);
        if ($count < 1){
            return true;
        }else {
            return false;
        }
    }
    $response = array();
    $response["success"] = false;
    if (gEmailAvailable()){
        registerGuide();
        $response["success"] = true;
    }

    echo json_encode($response);
?>
```

In the above code snippet, the name, user of and password for the database that contains the “guide” table are at the top of the code, and within the code there are the methods to send the information back to the database to register a user.

Below, you will see some of the java code from the class where the Registration takes place:

```
gRegisterBTN.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        final String gName = gNameRegET.getText().toString();
        final String gEmail = gEmailRegET.getText().toString();
        final String gPassword = gPasswordRegET.getText().toString();

        Response.Listener<String> responseListener = new
Response.Listener<String>() {
            @Override
            public void onResponse(String response) {

                try {
                    JSONObject jsonResponseT = new JSONObject(response);
                    boolean success = jsonResponseT.getBoolean("success");

                    if (success) {
                        Intent intent = new Intent(GuideRegister.this,
GuideLogin.class);
                        GuideRegister.this.startActivity(intent);
                    } else {
                        AlertDialog.Builder builder = new
AlertDialog.Builder(GuideRegister.this);
                        builder.setMessage("Registration Failed")
                                .setNegativeButton("Retry", null)
                                .create()
                                .show();

                    }


                } catch (JSONException e) {
                    e.printStackTrace();
                }
```

At the end of this code, there is a small section that calls in the **registration request**:

```
GuideRegisterRequest gRegReq = new GuideRegisterRequest(gName, gEmail,
gPassword, responseListener);
RequestQueue queue = Volley.newRequestQueue(GuideRegister.this);
queue.add(gRegReq);
```

What this request does is call in a java class called GuideRegisterRequest, which contains the information needed to send the data captured from my EditText fields and converted into strings to be sent to the database in order to register the user. The code for GuideRegisterRequest is below:

```
public class GuideRegisterRequest extends StringRequest {
    private static final String GUIDE_REGISTER_REQUEST_URL =
"http://yellowumbrellatours.comxa.com/GuideRegister.php";
    private java.util.Map<String, String> params;

    public GuideRegisterRequest(String gName, String gEmail, String gPassword,
Response.Listener<String> listener){
        super(Method.POST,GUIDE_REGISTER_REQUEST_URL, listener, null);
        params = new HashMap<>();
        params.put("gName", gName);
        params.put("gEmail", gEmail);
        params.put("gPassword", gPassword);
    }
    public java.util.Map<String, String> getParams() {
        return params;
    }
}
```

The next PHP file, GuideLogin.php, checks that all parameters entered in GuideLogin.php are correct before allowing the user to log in:

```
<?php
```

```
    $connect = mysqli_connect("mysql9.000webhost.com", "a8066174_SamG",
"B3taBas3", "a8066174_YellowB");

    $gEmail = $_POST["gEmail"];
    $gPassword = $_POST["gPassword"];

    $statement = mysqli_prepare($connect, "SELECT * FROM guide WHERE gEmail = ?
AND gPassword = ?");
    mysqli_stmt_bind_param($statement, "ss", $gEmail, $gPassword);
    mysqli_stmt_execute($statement);

    mysqli_stmt_store_result($statement);
         $returnName = '';
         $returnEmail = '';
         $returnPassword = '';
    mysqli_stmt_bind_result($statement, $returnName, $returnEmail,
$returnPassword);

    //echo("Name : $returnName , Email : $returnEmail , Password :
$returnPassword");

    $response = array();
    $response["success"] = false;

    while(mysqli_stmt_fetch($statement)){
        $response["success"] = true;
        $response["gEmail"] = $returnEmail;
        $response["gPassword"] = $returnPassword;
                 $response["gName"] = $returnName;
    }

    echo json_encode($response);
?>
```

Similarly to the registration, the Login is also implemented in its own class and has a request class, calling for the return of the details to make sure that they match credentials already in the database. Firstly, the GuideLogin class:

```
gLoginBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String gEmail = gEmailET.getText().toString();
                String gPassword = gPasswordET.getText().toString();

                // Response received from the server
                Response.Listener<String> responseListener = new
Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        try {
                            JSONObject jsonResponse = new JSONObject(response);
                            boolean success =
jsonResponse.getBoolean("success");

                            if (success) {
                                Intent intent = new Intent(GuideLogin.this,
GuideHome.class);

                                GuideLogin.this.startActivity(intent);
                            } else {
                                AlertDialog.Builder builder = new
AlertDialog.Builder(GuideLogin.this);
                                builder.setMessage("Login Failed")
                                        .setNegativeButton("Retry", null)
                                        .create()
                                        .show();
                            }

                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                };

                GuideLoginRequest gLoginRequest = new GuideLoginRequest(gEmail,
gPassword, responseListener);
                RequestQueue queue = Volley.newRequestQueue(GuideLogin.this);
                queue.add(gLoginRequest);
            }
        });
    }
}
```

The LoginRequest class:

```
public class GuideLoginRequest extends StringRequest {
    private static final String GUIDE_LOGIN_REQUEST_URL =
"http://yellowumbrellatours.comxa.com/GuideLogin.php";
    private java.util.Map<String, String> params;

    public GuideLoginRequest(String gEmail, String gPassword,
Response.Listener<String> listener) {
        super(Method.POST, GUIDE_LOGIN_REQUEST_URL, listener, null);
        params = new HashMap<>();
        params.put("gEmail", gEmail);
        params.put("gPassword", gPassword);
    }

    @Override
    public java.util.Map<String, String> getParams() {
        return params;
    }
}
```

The next thing that must be implemented is the function which will allow the user to book a tour of their choice:

```
tBookingBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final String name = tBookingNameET.getText().toString();
                final String email = tBookingEmailET.getText().toString();
                final int people =
Integer.parseInt(tBookingNumET.getText().toString());

                Response.Listener<String> responseListener = new
Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {

                        try {
                            JSONObject jsonResponseT = new JSONObject(response);
                            boolean success =
jsonResponseT.getBoolean("success");

                            if (success) {
                                AlertDialog.Builder builder = new
AlertDialog.Builder(NorthsideBook.this);
                                builder.setMessage("Tour Booked")
                                        .setPositiveButton("Okay", null)
                                        .create()
                                        .show();

                            } else {
                                AlertDialog.Builder builder = new
AlertDialog.Builder(NorthsideBook.this);
                                builder.setMessage("Booking Failed")
                                        .setNegativeButton("Retry", null)
                                        .create()
                                        .show();

                            }


                        } catch (JSONException e) {
                            e.printStackTrace();
                        }

                    }
                };

                NorthsideBookRequest nBookReq = new NorthsideBookRequest(name,
email, people, responseListener);
                RequestQueue queue = Volley.newRequestQueue(NorthsideBook.this);
                queue.add(nBookReq);
            }
        });
    }
}
```

This is the code to book the Northside Tour. As with the previous java snippets, this class calls in the Request class in order to send the details serialized as JSON objects to the database. The NorthsideBookRequest class is shown below:

```
public class NorthsideBookRequest extends StringRequest {
    private static final String NORTHSIDE_BOOK_REQUEST_URL =
"http://yellowumbrellatours.comxa.com/BookNorth.php";
    private java.util.Map<String, String> params;

    public NorthsideBookRequest(String name, String email, int people,
Response.Listener<String> listener) {
        super(Method.POST, NORTHSIDE_BOOK_REQUEST_URL, listener, null);
        params = new HashMap<>();
        params.put("name", name);
        params.put("people", people + "");
        params.put("email", email);

    }

    @Override
    public java.util.Map<String, String> getParams() {
        return params;
    }
}
```

This is the class that specifies the .php file that will be handling the information to be sent to the database. That .php file, BookNorth.php, is shown below:

```
<?php
    $connect = mysqli_connect("mysql9.000webhost.com", "a8066174_SamG",
"B3taBas3", "a8066174_YellowB");

    $name = $_POST["name"];
    $people = $_POST["people"];
    $email = $_POST["email"];
    $statement = mysqli_prepare($connect, "INSERT INTO northsideBooked (name,
email, people) VALUES (?, ?, ?)");
    mysqli_stmt_bind_param($statement, "ssi", $name, $email, $people);
    mysqli_stmt_execute($statement);

    $response = array();
    $response["success"] = true;

    echo json_encode($response);
?>
```

I will also be implementing a chat function within the app so that tourists can communicate with a tour guide that will be monitoring the support messages. Tourists will be able to use this messaging function in order to communicate with a guide from within the app in order to ask questions or direct any queries at the guide who will be monitoring the support system. The messaging system will make use of intercom.io and GCM. The app will then send push notifications to any users who have received a message, but in order to use push notifications, I will have to register my App for GCM (Google Cloud Messaging) before adding in that functionality. GCM works by importing the google-services.json file to your android project. After this, we can begin coding

Below, we have the code that will listen for the user I.D assigned by intercom to the user, by which they will be identified by the guide providing support:

```
public class IntercomIdListenerService extends InstanceIDListenerService {
    @Override
    public void onTokenRefresh() {
        Intent intent = new Intent(this, RegistrationIntentService.class);
        startService(intent);
    }
}
```

In this class, we see an intent referencing RegistrationIntentService, which is the class that assigns a token to the user in order to send push notifications to the user. This class is shown below:

```
public class RegistrationIntentService extends IntentService {
    private static final String TAG = "RegIntentService";
    public RegistrationIntentService() {
        super(TAG);
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        try {
            InstanceID instanceID = InstanceID.getInstance(this);
            String token =
instanceID.getToken(getString(R.string.gcm_defaultSenderId),
GoogleCloudMessaging.INSTANCE_ID_SCOPE, null);
            Log.i(TAG, "GCM Registration Token: " + token);

            sendRegistrationToServer(token);
        } catch (Exception e) {
            Log.d(TAG, "Failed to complete token refresh", e);
        }
    }

    private void sendRegistrationToServer(String token) {
        Intercom.client().setupGCM(token, R.drawable.intercomsdk_default_push);
    }
}
```

Intercom itself requires some initialization code:

```
public class CustonClass extends Application {
    //CHANGE THESE VALUES
    private static final String YOUR_API_KEY = "<INTERCOM_API_KEY>";
    private static final String YOUR_APP_ID = "<INTERCOM_APP_ID>";

    @Override public void onCreate() {
        super.onCreate();
        Intercom.initialize(this, YOUR_API_KEY, YOUR_APP_ID);
    }
```

From here, you will need to add the function to allow the user to send a support message:

```
Intercom.client().displayMessageComposer();
```

As well as a button to allow users to view previous conversations:

```
Intercom.client().displayConversationsList();
```

You also need to add some code to your main activity to make sure that Intercom and GCM are working together:

```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener {

    private static final String YOUR_EMAIL = "";
    private static final String YOUR_USER_ID = "";


    private static final String YOUR_HMAC = "";
    private static final String YOUR_DATA = "";

    private Button registerButton;

    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        registerButton = (Button) findViewById(R.id.register_button);
        registerButton.setOnClickListener(this);

        //if you have provided a hmac and data try begin secure session
        if (!TextUtils.isEmpty(YOUR_HMAC) && !TextUtils.isEmpty(YOUR_DATA)) {
            Intercom.client().setSecureMode(YOUR_HMAC, YOUR_DATA);
        }


        Intercom.client().openGCMMessage(getIntent());
    }

    @Override public void onClick(View v) {
        Registration registration = Registration.create();
        if (!TextUtils.isEmpty(YOUR_USER_ID)) {
            registration.withUserId(YOUR_USER_ID);
        }
        if (!TextUtils.isEmpty(YOUR_EMAIL)) {
            registration.withEmail(YOUR_EMAIL);
        }
        Intercom.client().registerIdentifiedUser(registration);

        registerButton.setEnabled(false);
        setUpPush();
    }

    @Override public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_main, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action_intercom) {
            Intercom.client().displayConversationsList();
        }
        return super.onOptionsItemSelected(item);
    }


    private void setUpPush() {

        if (checkPlayServices()) {
            // Start IntentService to register this application with GCM.
            Intent intent = new Intent(this, RegistrationIntentService.class);
            startService(intent);
        }
    }

    public boolean checkPlayServices() {
        int resultCode =
GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);
        return resultCode == ConnectionResult.SUCCESS;
    }
}
```

In the end, the intercom integration didn't go to plan, and I ended up creating a contact form that would allow the tourist to fill out a contact form in order to be contacted by a guide. The form would work by collecting the users data entered into the 3 text fields (Name, email and query), converting them into strings, as seen in some of the classes above, and sending them to the database, where they will be monitored by a guide who will look after support. This will work in a similar fashion to the feedback and booking functions, with the contact form class also having a request class that will send the data to the PHP file, to be inserted into the relevant table in the database.

Some of the code that will enable the contact form is shown below:

```
contactBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final String name = touristNameET.getText().toString();
                final String email = touristEmailET.getText().toString();
                final String question = questionET.getText().toString();

                Response.Listener<String> responseListener = new
Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {

                        try {
                            JSONObject jsonResponseT = new JSONObject(response);
                            boolean success =
jsonResponseT.getBoolean("success");

                            if (success) {
                                AlertDialog.Builder builder = new
AlertDialog.Builder(ContactUs.this);
                                builder.setMessage("Your question has been
asked. A guide will contact you in relation to this shortly.")
                                        .setPositiveButton("Okay", null)
                                        .create()
                                        .show();

                            } else {
                                AlertDialog.Builder builder = new
AlertDialog.Builder(ContactUs.this);
                                builder.setMessage("Question not submitted,
please try again")
                                        .setNegativeButton("Retry", null)
                                        .create()
                                        .show();

                            }


                        } catch (JSONException e) {
                            e.printStackTrace();
                        }

                    }
                };

                ContactUsRequest contactReq = new ContactRequest(name, email,
question, responseListener);
                RequestQueue queue = Volley.newRequestQueue(ContactUs.this);
                queue.add(contactReq);
            }
        });
    }
}
```

2.5 *Testing*

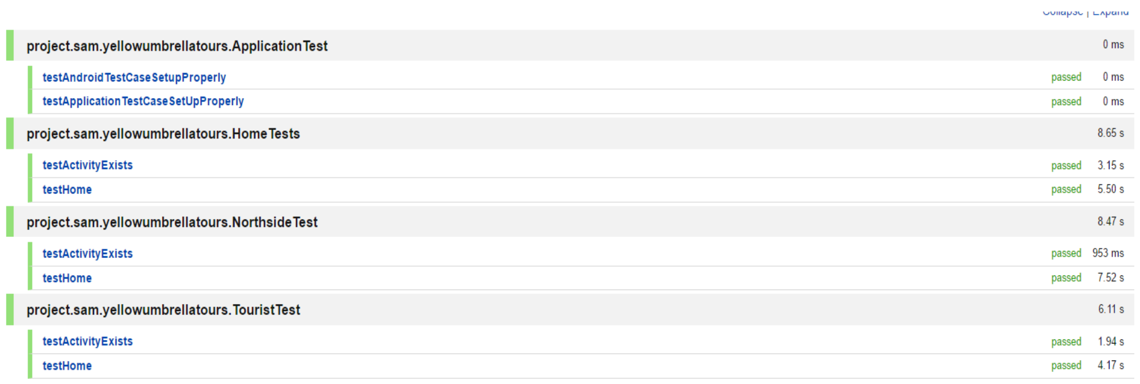
For this project, I will be using Junit testing in order to carry out my technical testing, and will be making use of testing libraries in Android Studio to make sure that my app is performing as expected under the hood. I will also be carrying out practical testing, which will entail myself and one guide from Yellow Umbrella Tours. Practical testing will be discussed later on, under the **Customer Evaluation** heading, so for now we will discuss JUnit testing. JUnit is a framework which allows you to write testing methods and organize these methods into test classes, and use these to test the functionality of your app. In order to test my app, I constructed a number of simple tests to test the staple functions of the app, i.e making sure that buttons were working and leading to the correct pages, etc. One such class that I wrote in order to test my project was a testing class called HomeTests, which you can see below:

```
public class HomeTests extends ActivityInstrumentationTestCase2<Home> {

    public HomeTests() {
        super(Home.class);
    }

    public void testActivityExists() {
        Home activity = getActivity();
        assertNotNull(activity);
    }

    public void testHome() {
        Home activity = getActivity();


        // Tap "tourist" button
        // ----------------------

        Button touristButton =
                (Button) activity.findViewById(R.id.tBTN);

        TouchUtils.clickView(this, touristButton);



    }
}
```

The above test ensures that the activity exists within the project, and carries out a button click in order to make sure that the button is working and leading to the right page. Another class that I created, which carries out a similar job to this is TouristTest, which makes sure that the buttons on the TouristHome page function properly.

```
extends ActivityInstrumentationTestCase2<TouristHome> {

    public TouristTest() {
        super(TouristHome.class);
    }

    public void testActivityExists() {
        TouristHome activity = getActivity();
        assertNotNull(activity);
    }
    public void testHome() {
        TouristHome activity = getActivity();

        Button northButton =
                (Button) activity.findViewById(R.id.nSideTourBTN);

        TouchUtils.clickView(this, northButton);

        Button southButton =
                (Button) activity.findViewById(R.id.sSideTourBTN);

        TouchUtils.clickView(this, southButton);



    }
}
```

This test checks the functionality of both buttons on the page. After writing a couple of more tests, I decided to check the results of these tests by exporting them as an html file, and the results can be seen below:



project.sam.yellowumbrellatours.ApplicationTest		0 ms
testAndroidTestCaseSetupProperly	passed	0 ms
testApplicationTestCaseSetUpProperly	passed	0 ms
project.sam.yellowumbrellatours.HomeTests		8.65 s
testActivityExists	passed	3.15 s
testHome	passed	5.50 s
project.sam.yellowumbrellatours.NorthsideTest		8.47 s
testActivityExists	passed	953 ms
testHome	passed	7.52 s
project.sam.yellowumbrellatours.TouristTest		6.11 s
testActivityExists	passed	1.94 s
testHome	passed	4.17 s

Figure 7 Test Run times

The above results show how long each test took and whether or not it passed. This allowed me to analyse not only the success of the tests, but how they were performing and if they would need to be changed, and in making sure that no function dropped below an acceptable level of performance.

2.6 Graphical User Interface (GUI) Layout

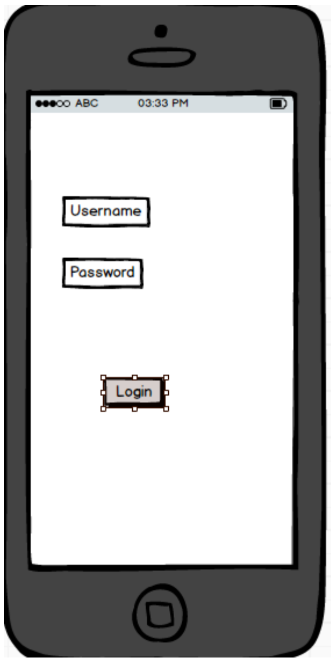


Figure 8 Login screen

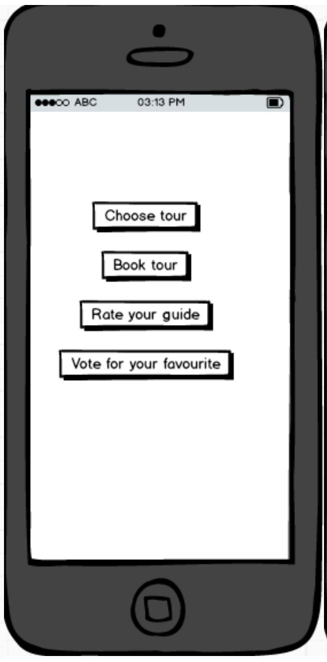


Figure 9 Home screen

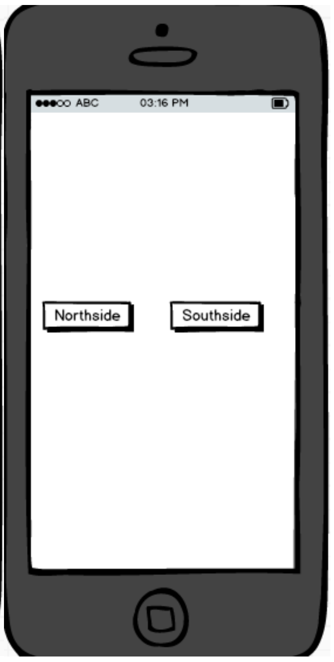
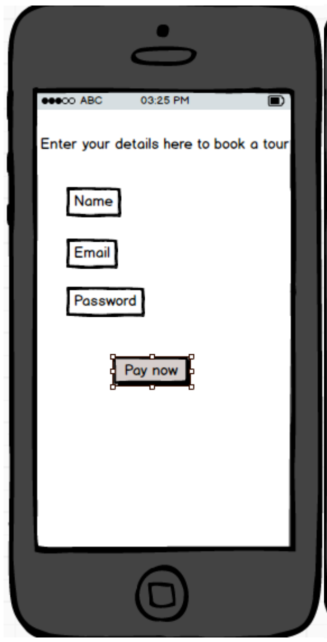


Figure 10 Choosing your tour



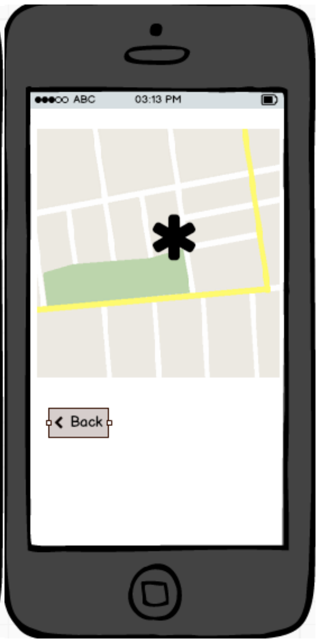


Figure 11 Payment screen   Figure 12 Page containing the map.

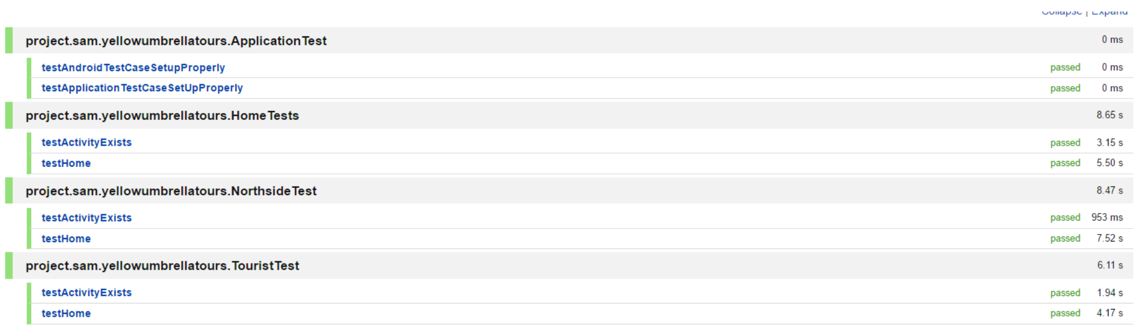
2.7 Customer testing

In order to carry out customer testing, I installed the app on one of the guides' Android phone. From here, I asked him to try out the different sections of the app and give me his feedback for the different sections and functions. This was a process carried out over a long period of time, throughout the development of the project .The main points that the guide, Peter, touched on initially were that system seemed a bit slow at first, and that the interface was a bit clunky and not too user friendly. At this stage, I had only developed a very rough draft of what the app would look like. After spending another couple of weeks working at the app, I brought it back to him, installed it and asked him to evaluate it once more. He noted the improvement of the interface, remarking that it was much more user friendly, with sections being labelled in a much more recognizable fashion, and the system seemed to be a little quicker, due to a cleanup of the code that I had carried out between review sessions. Overall, after our second review, Peter seemed pleased with the progress that had been made, and advised me on a couple of changes that he thought would add to the app. At this stage, he suggested I add markers for the start and finish points for the tours on the map, and add the companies soon to be logo, the yellow umbrella, as a custom marker. The next time that we reviewed the app's usage, Peter noted that the app was still running smoothly and that the main features of the app were working well. In regards to my contact with Peter and other guides currently working with him, I also spoke to the guides during my requirements gathering phase, asking the guides what they felt the app needed and also what they felt would improve the app for both the tourists and the guides.

2.8 Evaluation

The system was evaluated using a combination of customer evaluation, as detailed above, and through JUnit testing. Throughout the customer evaluation phase, it became evident that the code would need to be cleaned up in order to keep the app efficient. I also needed to construct my Junit testing classes in order to make sure that my classes were all running as expected, and those tests have been discussed in more detail under the **Testing** heading in this report.

One of the factors discussed above was the tests themselves and how quickly they were running, as shown below:



```
project.sam.yellowumbrellatours.ApplicationTest                 0 ms
    testAndroidTestCaseSetupProperly                  passed    0 ms
    testApplicationTestCaseSetUpProperly              passed    0 ms
project.sam.yellowumbrellatours.HomeTests                     8.65 s
    testActivityExists                                passed  3.15 s
    testHome                                          passed  5.50 s
project.sam.yellowumbrellatours.NorthsideTest                 8.47 s
    testActivityExists                                passed  953 ms
    testHome                                          passed  7.52 s
project.sam.yellowumbrellatours.TouristTest                   6.11 s
    testActivityExists                                passed  1.94 s
    testHome                                          passed  4.17 s
```

Figure 13 Test run times

3 Conclusions

In conclusion, from working on this app, I feel that this app has an advantage over other tourism apps in that it takes guides into consideration and allows them to make their own lives easier through the tour data system, which allows guides to submit notes about their tour back to the database, i.e changes to routes, issues at certain stops, etc. In the context of the apps practical use, it also helps the guides and the company itself to see the number of people attending the tours and what time they finish, allowing the company to better organise tours around finishing times and attendance. I feel that this would be especially helpful if the app were to expand and include a whole network of tour companies, instead of just one. This would lead to greater co-operation between tour companies, as well as allowing tourists to book tours in cities that they haven't visited yet. A disadvantage of the app is that I could not facilitate a guide support mailbox within the app itself, which would have led to a much more interactive and immersive experience. The opportunites offered by the app here include the potential for expansion (mentioned above), the opportunity to "onboard", if you like, incoming tourists and leave them with a good experience of the city and of Yellow Umbrella Tours especially. One limit of this project is getting it out there. It may be difficult to get people who may only be in the city for a few days to find this app and use it in order to book tours and give feedback, especially considering that the company originally did most of its business through flyers and relying on tourists to come to the start point of the tour, but I feel that this could be counter-acted with changes to the marketing structure of Yellow Umbrella Tours in order to try and push people towards using the app while they are in Dublin. Another limit that could be an issue in the early phase is not only getting the app out there and spreading knowledge of it, but encouraging use, especially if it is just based off Dublin. Without the expansion and addition of other cities, people may not be keen on downloading an app that will only be useful for a couple of days. This however, is only a potential drawback, as it is impossible to tell how people will take to it before it has been put on the market.

4 Further development or research

With more time and resources, I feel I could scale this app out to include an entire network of tour companies and cities, each linked through suggestions and based off of where the user has been and what tours they have enjoyed. This would address one of the limits that I had mentioned previously, and would probably be received more positively by tourists if the app were to include a network of tour companies around Europe as opposed to just the one. Another possibility for the future, if the app were to include a network of tour comapanies around Europe, would be to create something of a tourist social network and have tourists share pictures, experiences and recommendations and use guides as admins, making sure that all content is appropriate and also offering their own contributions to pictures, experiences and reccomendations. The app could also be ported over to other devices such as iPhones or Windows phones.

5 References

How to Accept Payments in an Android App Using MPL - PayPal Developer. 2016. *How to Accept Payments in an Android App Using MPL - PayPal Developer*. [ONLINE] Available at:https://developer.paypal.com/docs/classic/mobile/ht_mpl-itemPayment-Android/. [Accessed 04 February 2016].

Google Maps Android API | Google Developers. 2016. *Google Maps Android API | Google Developers*. [ONLINE] Available at: https://developers.google.com/maps/documentation/android-api/. [Accessed 04 February 2016].

Testing Concepts | Android Developers. 2016. *Testing Concepts | Android Developers*. [ONLINE] Available at: http://developer.android.com/tools/testing/testing_android.html. [Accessed 08 May 2016].

Testing activity in Android Studio. Part 1.. 2016. *Testing activity in Android Studio. Part 1.*. [ONLINE] Available at: http://evgenii.com/blog/testing-activity-in-android-studio-tutorial-part-1/. [Accessed 08 May 2016].

Testing activity in Android Studio. Part 2 . 2016. *Testing activity in Android Studio. Part 2.*. [ONLINE] Available at: http://evgenii.com/blog/testing-activity-in-android-studio-tutorial-part-2/. [Accessed 08 May 2016].

Testing activity in Android Studio. Part 3.. 2016. *Testing activity in Android Studio. Part 3.*. [ONLINE] Available at: http://evgenii.com/blog/testing-activity-in-android-studio-tutorial-part-3/. [Accessed 08 May 2016].

Intercom. 2016. *Intercom*. [ONLINE] Available at: https://app.intercom.io/a/apps/od2x3tpl/guide. [Accessed 09 May 2016].

GitHub. 2016. *GitHub - tonikami/NEWLoginRegister: Tutorieal for creating an android app which allows user to login and register.*. [ONLINE] Available at: https://github.com/tonikami/NEWLoginRegister. [Accessed 09 May 2016].

GitHub. 2016. *intercom-android/samples/intercom-gcm-sample/gcmsample/src/main/java/io/intercom/gcmsample at master · intercom/intercom-android · GitHub*. [ONLINE] Available at: https://github.com/intercom/intercom-android/tree/master/samples/intercom-gcm-sample/gcmsample/src/main/java/io/intercom/gcmsample. [Accessed 09 May 2016].

Hotline.io - Native in-app chat | Deep links | mobile app support |. 2016. *Hotline.io - Native in-app chat | Deep links | mobile app support |*. [ONLINE] Available at: https://hotline.io/in-app-chat. [Accessed 09 May 2016].

Google Developers. 2016. *Set up a GCM Client App on Android | Cloud Messaging | Google Developers*. [ONLINE] Available at: https://developers.google.com/cloud-messaging/android/client. [Accessed 09 May 2016].

Transmitting Network Data Using Volley | Android Developers. 2016. *Transmitting Network Data Using Volley | Android Developers*. [ONLINE] Available at:http://developer.android.com/training/volley/index.html. [Accessed 09 May 2016].

6 Appendix

6.1 Project Proposal

7 Objectives

This app is aimed towards tourism companies who would like more exposure and be able to offer more to customers than they can through sites like TripAdvisor. While we all have trip advisor and its different sections and its helpful reviews and suggestions, there is only so much that you can take from it. My goal is to create a tourism app (which will be based on one company initially) where you can view the available tours, the times they depart and even book and pay for your place on the paid tours. By the time I have developed this app, there should also be a page on the app where you can rate your tour guide and an average rating will appear, and even send a tip for your tour guide to the company through the paypal functionality within the app. My main objective is to create a stylish, well designed app that can perform all of its functions properly and with minimal faults, as well as making these tours and info regarding them accessible to all who may wish to go on them. It is also imperative that the security around the paypal API and the entering of all personal details is tight and all users of the app can be ensured that their details are safe and secure. The overall user experience must be pleasant and not too difficult, even for those who may not be all that technical. The app will also let users “bookmark” certain tours, sending them push notifications an hour or so before their tour is due to begin. There will be a section within the app that will also have info on the main stops on the tour, and allow you to choose your favourite, which will be saved. There will be a basic login feature and database to make sure that everything is stored and loaded correctly for each registered user. A facebook Login is also a possibility, with the option to share your tour experience and favourite stop through Facebook.

8 Background

The idea for this project came from a previous project that I had done as part of a group in the beginning of 3rd year. A similar concept (a tourism app, some points of interest), but I felt that it was sorely missing a lot of functionality. The ability to rate and save certain stops, booking tours, setting reminders for tours you'd like to go on and rating and reviewing your guide (arguably the most important part of the tour) were all missing. I wanted create an app with these features and improve on the existing features in order to create an app that will encourage people to get out and see Dublin City and what it has to offer, as well as creating an easy to use, immersive app that allows you to give your opinion on the experience. There is also a small walking tour company that I know of in the city centre, and looking at their business model and the structure of their tours also made me want to make this app, as everything nowadays is digital, whether it be marketing, sales or education, and I thought this app could really help this company in particular to grow a huge amount, allowing them to gain exposure that they otherwise would not have seen. I also wanted to show that I could handle the more complex issues that will no doubt arise while I am working on this project in order to prove my capabilities as a developer.

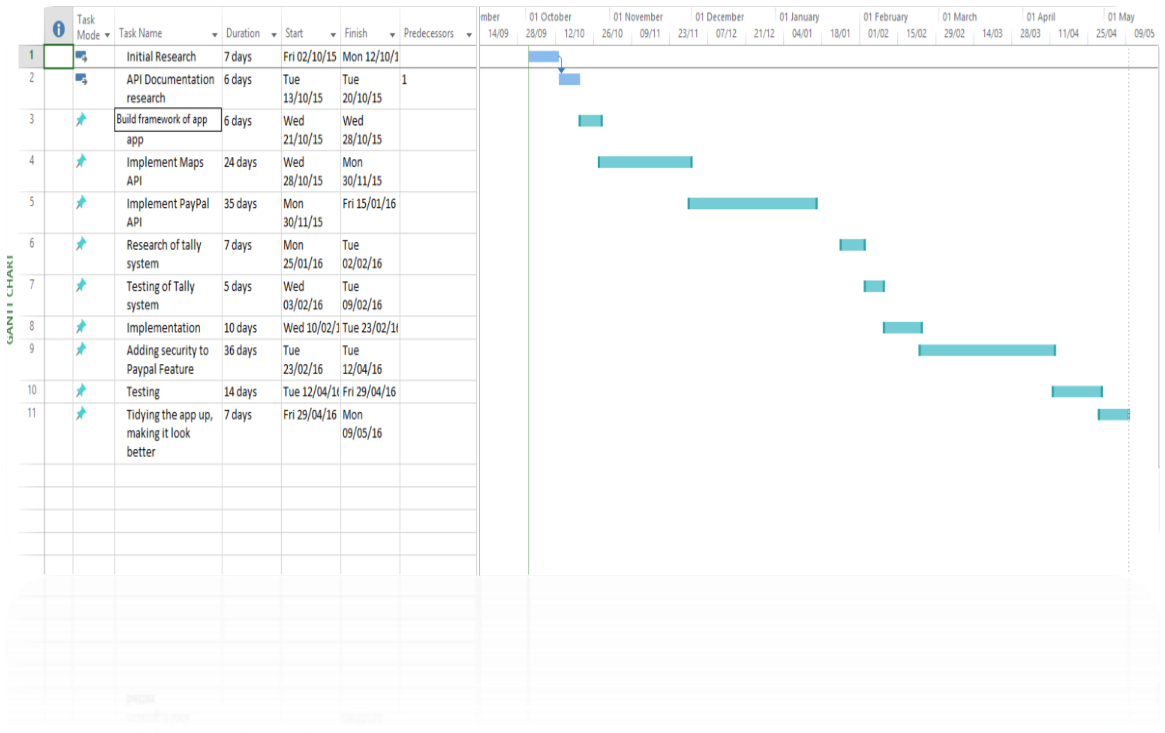
9 Technical Approach

For this project, I will be working from a basic Android framework with Map Activities as well as separate pages and added API's to give the functionality needed. I will also need to add in a database to store login details as well as people's ratings and reviews. This of course means that the database will have to be hosted in order for it to run consistently on every device that it's on. The use of the MPL for PayPal will likely give me the most trouble, as I will have to make sure the security around that is up to scratch. The tally system for the tour guide ratings will also be a bit difficult, and I will have to do more research on this before I dive in and start coding that. In regards to the PayPal MPL, PayPal themselves have a lot of documentation on their website to do with its implementation and compatability. The Google Maps API will be simple enough to implement, as there is plenty of documentation and even the option to put in a map activity in Eclipse JDE.

10Special resources required

Resource-wise, all I will need are android devices to test on and a few reference websites, such as the PayPal and Google developer sites for the API's, and other sites for tutorials on Android programming.

10.1Project Plan



	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Initial Research	7 days	Fri 02/10/15	Mon 12/10/1	
2		API Documentation research	6 days	Tue 13/10/15	Tue 20/10/15	1
3		Build framework of app app	6 days	Wed 21/10/15	Wed 28/10/15	
4		Implement Maps API	24 days	Wed 28/10/15	Mon 30/11/15	
5		Implement PayPal API	35 days	Mon 30/11/15	Fri 15/01/16	
6		Research of tally system	7 days	Mon 25/01/16	Tue 02/02/16	
7		Testing of Tally system	5 days	Wed 03/02/16	Tue 09/02/16	
8		Implementation	10 days	Wed 10/02/1	Tue 23/02/1	
9		Adding security to Paypal Feature	36 days	Tue 23/02/16	Tue 12/04/16	
10		Testing	14 days	Tue 12/04/1	Fri 29/04/16	
11		Tidying the app up, making it look better	7 days	Fri 29/04/16	Mon 09/05/16	

10.2Monthly Journals

Reflective Journal

Student name: Sam Gormley

Programme BSc in Computing

Month: September

11 My Achievements

During the month of September, much of my time was spent researching the ins and outs of my project. I researched the use of the PayPal API as well as researching a few different databases which I could use and incorporate into my project. I was also able to look at a previous project that may help me to build on this project, as the framework is similar. This helped me to see what I needed to do in relation to what I can already do and what I need to learn. My research took me to other APIs as well, such as the Google Maps API and the proper implementation of the features that I plan to use within my project, such as the geo-location and marking stops on the tours.

12 My Reflection

I felt it was good to look at an old project in order to work out where I was in relation to my ability and what I needed to learn and work on. As a whole, my research gave me a good idea of what I need to do, how long it will take and what I need to give more time to. I felt that I could have benefitted from some more practical work on the project, perhaps towards making a prototype.

13 Intended Changes

Next month I will devote more time to playing around with the practical elements and try to implement some of the smaller features into the project, such as the map and some markers, just to test them out before I dive in with the more complex features, as well as trying out the compatibility of some features together to make sure that one won’t break the other.

14 Supervisor Meetings (Not assigned supervisor yet)

Date of Meeting: N/A

Items discussed: N/A

Action Items: N/A

Reflective Journal

Student name: Sam Gormley

Programme: BSc in Computing

Month: October

15 My Achievements

This month, I was able to meet with my project supervisor and discuss the in’s and outs of my project, as well as come up with a couple of ideas for additions to my project in order to increase the complexity and come up with something innovative.

16 My Reflection

I was glad that I got to meet with my supervisor so that I could vocalise some of my ideas and get some feedback on them, and I felt that it was good to get more assured feedback, rather just my own research.

17 Intended Changes

Implement the more basic parts of my project so that I can devote more time to the innovative features of the app and try to test and implement them.

18 Supervisor Meetings

Date of Meeting: 06/11/15

Items discussed: Project structure, innovation.

Action Items: Work on the basic aspects of the project

Reflective Journal

Student name: Sam Gormley

Programme: BSc in Computing

Month: November

19 My Achievements

This month, I was able to begin my documentation, and I completed my requirements specification at the beginning of the month. Unfortunately, other assignments took up a lot of my time, so my contribution to the project was minimal this month.

20

21 Intended Changes

Next month, I will try to get more work done on the app and try to implement some featured. I realised that I need to try to divide my time out more evenly between subjects and projects.

22 Supervisor Meetings

Date of Meeting:

Items discussed:

Action Items:

Reflective Journal

Student name: Sam Gormley

Programme: BSc in Computing

Month: December

23 My Achievements

This month, I was able to begin some of the coding for my project, and decided to use Android Studio rather than the Android plugins for Eclipse, as Android studio is easier to use and more up to date than Eclipse. I created the shell for the app as well as attempting some tutorials on implementing some of my functionalities.

24

25 Intended Changes

Next month, I intend to begin the implementation of the aforementioned functionalities into my app instead of just following tutorials and making sure that I can understand the methods behind these functions.

26 Supervisor Meetings

Date of Meeting: 11/12/15

Items discussed: Functions of the project, documentation

Action Items: Work on diagrams, improve documentation

Reflective Journal

Student name: Sam Gormley

Programme: BSc in Computing

Month: January

27 My Achievements

This month, a lot of my time was consumed with the exams and preparing for them, so my time to work on the app was limited. However, I did manage to begin implementing some of the features into my app, such as the map and the login and registration function.

28

29 Intended Changes

With the exams out of the way, I will have a lot more time to work on my project, both the documentation and the app itself, so in the coming month I hope to make a lot of progress with my project.

30 Supervisor Meetings

Date of Meeting: 28/01/16

Items discussed: Implementation, Mid Point Presentation

Action Items: Implement one feature into the project for the presentation, get documentation up to scratch.

Reflective Journal

Student name: Sam Gormley

Programme BSc in Computing

Month: February

31 My Achievements

Throughout the month of February, I was able to focus a bit more on the research of other technologies for my project, such as hotline.io, an alternative to Intercom, different libraries that I could use for the project, etc. I was also able to begin some of the more basic implementations for my project, and began to get a better picture of how the app would look once completed.

32 My Reflection

I was happy to be able to start some of the more basic implementations this month, as I felt they gave me a better understanding of what would be needed and how best to Implement these features. However, I felt that I could have carried out a bit more research this month regarding some of the more difficult implementations.

33 Intended Changes

Next month, I will try to work on some more features of the app, as well as working a little bit on the design as well. I also realised that I needed to make some of the interfaces more user friendly and a bit more simplistic in order to appeal to potential users.

34 Supervisor Meetings

Date of Meeting: 22/02/16

Items discussed: Progress of the app, feedback from the mid-point presentation

Action Items: Work on Documentation, fix points brought up from mid-point presentation, continue to work on implementation

Reflective Journal

Student name: Sam Gormley

Programme BSc in Computing

Month: March

35 My Achievements

Throughout the month of March, I continued to try and balance my project with the rest of my workload. When I got the chance to work on my project, I focused mainly on the database interactions between the app and database, such as registration, logging in or sending feedback, etc. I also got the chance to work on the messaging function of the app, which will be provided by intercom.io.

36 My Reflection

It was good to get started on some of the more complex functions within the app this month. Getting a chance to look at intercom gave me a chance to see what I will need to do and what other steps I would need to take in order to get the app working the way I would like it to.

37 Intended Changes

Next month, I will try to work on the design of the app and try to perfect the GUI and get one of the guides to test it, in order to obtain some feedback.

38 Supervisor Meetings

Date of Meeting: 22/04/16

Items discussed: Progress of the app, advice on the next steps to take, project deliverables

Action Items: Work on Documentation, continue to work on implementation, work on poster and start thinking about the presentation.