# Software Engineering Group Project
# Producing a Final Report

Author: Christopher Edwards;Dillon Cuffe;Douglas Gardner;Jostein Kristansen;Ben Rainbow; Ashley Smith;James Woodside;Luke Horwood
Config Ref: SE_02_Fr_001
Date: 2014-02-10
Version: 1.2
Status: Released

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

# Contents

# 1 Introduction

## 1.1 Purpose of this Document

The purpose of this document is to set out all the necessary information and required documentation for the delivery of the final project within the Software Engineering Group Project. This final report will consist of all the required documents that will be needed for the delivery of the final report to the department of Computer Science, Aberystwyth University (AU).
As well as this it will give a full description of the roles and responsibilities of individuals within the project plus the areas of the project that where most difficult and what areas where most beneficial
Within this document are the essential parts of the project and specific pieces of information that shall be marked and form the grade of our CS22120 assignment.
These documents were developed and follow the requirements and specification of the University requirements specification.

## 1.2 Scope

This document is going to be used to establish what will be needed to be prepared and developed for the delivery of the final report. This document will offer all the developments and information within the group project. It will offer the alternative solutions to problem solving and will suggest how different tasks could have been completed differently.

## 1.3 Objectives

The aim of this document is to collate all the documents and information that we have gathered throughout the development of the group project. This document is to be shown to the markers as the final document that they will receive from our group and will form the basis of the mark that we shall receive.

# 2 Documentation produced for the final Delivery

## 2.1 The End of Project Report

### 2.1.1 Management Summary

***What the project achieved***

I feel that the project met all the requirements that were set out at the beginning of the project. I feel that we, as a group, have developed a fully functioning Android application that allows the user to record walks and upload them to a server to share with other people.
As well as this, we were also able to meet the functional requirements of the project. This shows that the application has the ability to record the walking tour, allowing for the GPS location of the tour. The application also has the ability to attach an image to the tour and add a description. In addition we have the ability to send the walks to the server where they will be stored in the database, which we set up. Then we display the walks on the website we created, so people can view them.

***What parts of the project work or don't work***

Within the creation of the Android application we aimed to get all areas of the application to be fully functional and have no errors. I believe that we have done this and we found the Android side of the project the easiest to use and develop the functionality for.
However there are also areas that caused issues to the project and do not work as intended. This can be seen on the server side part of the assignment. This part of the project is where we found most difficulties; we struggled with the uploading of an image from the tour to the database and getting the images and posts on to our website.

***What documents are in a good state and which are not***

When referring to the documents that have been produced, and looking at the feedback that was provided I feel that the documents were of a high quality and in a good state. The quality of our documentation can be seen in the Project Plan. This is because I felt that the document has a large amount of necessary and informative content that is required for the project, there seemed to be minimal need to improve this document when looking at the feedback provided, however to perfect this document we altered parts of it.

However looking at the other side I feel that the documents that were of a lower standard can be seen in the Test Procedure. This is because this document is the document that we received the lowest mark for.
Responding to the low marks received for this document, the QA manager looked in-depth at this document and made numerous changes to it. The changes that where made where in conjunction with the feedback provided and I believe we will achieve a higher mark for this document.

Other documents that we felt are in a good state would be the Design Specification, I feel that we worked together as a group on this document with each individual having a specific role and all collating together to gain a high level of quality within this document.

***How well the team preformed***

As a whole the group worked very well, we performed all the tasked that were asked of us. There were minor issues with individuals not wanting to do specific pieces of the work; however we managed to work our way through these issues and work as a team to complete the project. Throughout the whole project there was good leadership and every individual know there role within the development of the application. If there were any issues within the project we knew we could talk to others in the group or go to the Project Manager.

***Difficulties that stood in the way of the project completion***

Below are the problems that the group faced through the development of the Android application.

**New to Android development:**

First of all we were all new to developing Android applications, however our previous experiences with Java helped with structuring the code but Android code is completely new in terms on syntax. The main problem being new to the Android development process was learning about Activities and the Activity lifecycle. These are important steps in understanding how the application will present itself to the user, in terms of its user interface and the transition between activities.

In all honesty this was not too difficult as there were many resources online that provided simple, but in depth analysis of how to effectively develop an Android application sticking to the conventional standards in regards to the Activity life cycle.

**Passing information between Activities:**

Another Activity related problem we faced during the development process was how to pass the information in this Activity to another. Initially we had no idea how to go about this. After a quick bit of research online using resources such as StackExchange.com and the Android API's, we found the solution we were looking for.

This solution uses Bundles and Intent Extra's to pass in extra information to another Activity. This information can then be retrieved using specific String keys in the onCreate () method in the Activity, where the extras are required.

Also we came across another method which you can start an Activity in order to return its result, this is using the startActivityForResult (), setResult () and finish () methods in combination with the onActivityResult () method to produce the desired outcome.

Overall this problem seemed very daunting at first; however, once you start to research the problem and understand how certain methods work it is fairly trivial to pass information around your application's many Activities.

**Accessing and displaying Google Maps**

This is a major part of the Android walking tour application. Initially I thought this would be a fairly simple task. I originally thought it may be a couple of imports here and there and then create a new instance of a Google Map Object. We were not too far off with this assumption, but the technical process was confusing and tricky to debug at times.

In order to use the Google Maps for Android API version 2, you need to locate the SHA1 key for your Android installation and access the GoogleAPI cloud console. Once you have accessed this console, you are required to generate an API key. This is inserted into your metadata in the Android manifest.xml in order to be granted authority for your application package.

After following some guides on the internet we finally managed to understand the whole process. We now have a fully functioning Google Map Object with the ability to add markers to a map overlay.

**Retrieving the devices GPS location**

Since none of the group had undergone a task like this before this was pretty much new territory to us all. We knew that somehow we would need to access the devices network state in order to retrieve information about our current position in terms of Latitude and Longitude.

Firstly, we were required to add permissions to the Android manifest that allowed the application to access the Wi-Fi state/network states on the device running the application. Secondly, you needed to important the relevant Android libraries, which in this case were LocationManager and Location.

The LocationManager Class made the whole process relatively simple, as you could create a new Location Object based off the information stored in the Location Manager, and from this Location you can access the methods provided such as Location.getLongitude() and Location.getLatitude() in order to achieve the desired outcome.

**Uploading a walk/images/locations to the server**

Again none of us had a real hand on experience in terms on uploading to a server from an Android application but we knew what techniques we were to use. We all understood that we were to use the HTTP POST protocol and agreed on using PHP as the server side language, which made the whole process slightly easier as it narrowed down the research scope.

To start off we looked up sending information over HTTP POST in Android and discovered that although it is relatively simple in coding terms, you cannot perform network operations in the Main Thread of an Android application so we had to find a work around. In order to prevent this problem, we found that you should us an ASyncTask and its doInBackground () method to perform such activities. These ASyncTasks are quite easy to implement and run, which saved us a lot of hassle and potential refactoring.

Once we had managed to POST to the target server we then had to think about how we were going to send the information. Sending a JSON Object composed of several JSON Objects/Arrays appealed to be the simplest way to achieve this. To do this, the walks information and locations are encoded into JSON Objects and the PHP parses the Objects storing them in the correct manner in the database, this wasn't too difficult to achieve.

Arguably the main problem in this task was sending the images across the server as part of the JSON. After some heavy research and several failed attempts we decided that encoding the image Bitmap into a base 64 string then sending that to the server which can then re encode this image fairly simply, was the best approach to tackle the problem. There are potentially more elegant ways, but this can be part of maintenance process.

**Achieving a consistent layout for the Activities using the XML**

The problem was not too hard for the vast majority of the time, but sometime when you needed to position something in a very specific location using relative layout; it would automatically add the attribute such as alignBottom or alignRight of a particular element, which would cause mayhem if you tried to move things around.

In terms of simplicity it was not technically difficult as there is a WYSIWYG editor available through the Eclipse interface; it was quite tricky to get the layout exactly as intended. Not all of the Activities are perfect but after numerous attempts, we decided we had achieved a professional finish to our applications Activities.

### 2.1.2 A historical account of the project

Following the Gantt chart that was developed at the beginning of the project, we aimed to work to the dates with the dependencies on each step. We also wanted to develop full functionality and ensure we were working to the deadline. The events are as follows:

Thurs 10/10/13: Our first weekly group-meeting (With Lynda).
We went through strengths and weaknesses and previous experience.
Nice spread of skills - some not confident in coding, some not confident in documentation.

Wed 16/10/13: Android workshop.
Christopher and Luke were our two volunteers for this event - however did not go well. Some trouble with organisation of the workshop, and software that did not work properly.

Mon 21/10/13: Third group-meeting.
Roles fully established
Leader-James (jaw57), Documentation, Version Control
Deputy leader- Dillon (dic6), Documentation
QA Manager- Ben (bar5), Documentation, minutes
QA team- Jostein (jok13), Testing
QA team- Ashley (ays8), Documentation
Tech team- Luke (luh11), Coding, Android
Tech team- Chris (che16), Coding, Android
Tech team- Douglas (dog2), PHP, GitHub

Thurs 25/10/13: No meeting with Lynda.
Lynda was sick, so we could not have our weekly group-meeting with her. We also decided that, due to many assignments being due soon, we would not hold any meetings this week.

Wed 30/10/13: GitHub practical and meeting with Lynda.
Ashley and James volunteered to attend a GitHub lesson; they felt capable to use GitHub afterwards. Lynda attended our self-organised group-meeting today. We discussed risk-analysis, UI-design, Gantt-chart, and having another group-meeting on Tuesdays. Lynda seemed pleased with our progress.

Sun 03/11/13: Finishing the project plan.
The QA team (Ben, Ashley, and Jostein) and Dillon worked on finishing the content for the project plan together.

Mon 04/11/13: Final touches to the project plan.
We were missing a little bit of work on the project plan, but these parts have been added to the document now. Douglas decided to take it upon himself to convert our entire word-document into Latex, but we decided to discard his document in our final hand in.

Wed 06/11/13: First deliverable – Project plan including interaction design for the system.
We handed in the final Word document, and updated the title page, table of contents, and document history to finish it off.

Fri 15/11/13: Second deliverable – Test specification for the final system.
We handed in our finished version of the test specification, containing our complete list of tests to be run on the system once it is ready to be tested.

Wed 20/11/13: Design meeting.
James, Christopher, Luke, and Douglas met in a computer room and worked together to produce class diagrams for the different parts of our program, as part of our third deliverable: The Design specification.

Thurs 05/12/13: Weekly group meeting (with Lynda).
We talked about how to set up a database on the university network, and found that another group had registered a database for our group by mistake; preventing us from getting a new one or accessing the one they had registered.

Fri 06/12/13: Third deliverable – Design specification for the final system.

We handed in our completed version of the design specification; we only had one issue during the development of this document: the sequence diagram. The sequence diagram is the reason why the document was delivered at the end of the week; the rest of the document was already finished on Tuesday the same week.

Thurs 12/12/13: Fourth deliverable – prototype demo.

Christopher had prepared a demo for us to show to Lynda in our weekly group meeting, and she seemed pleased at the progress we had made. We did not yet have a working application, but we had some example screens of the Android and web pages to show, simulating planned functionality.

Sun 15/12/13: Database.

We now have a working MySQL database. The issue with the databases from earlier (Thurs 05/12/13) is therefore resolved.

Mon 27/01/14: Work week.

During work week, the final changes for the Android application were rectified and perfected. Documentation was worked on and was checked over by the QA team and additional diagrams were completed. The website was worked on and the additional features were added to ensure we had all the requirements covered. The server side code was refactored and testing was carried out.

### 2.1.3 Final state of the project

From a programming perspective the project is completed with its limitations noted within our system level tests and within the acceptance testing.

FR1 - Proven to work via the passing of test SE-F-001 through to SE-F-007
FR2 – Proven to work via the passing of test SE-F-008 through to SE-F-015
FR3 - Proven to work via the passing of test SE-F-016 through to SE-F-024
FR4 - Proven to work via the passing of test SE-F-023 through to SE-F-028
FR5 - Proven to mostly work via the passing of test SE-F-029 and SE-F-031
FR6 - Proven to work via the passing of test SE-F-032
FR7 - Proven to work via the passing of test SE-F-034 and SE-F-035
FR8 - Proven to work via the passing of test SE-F-036
FR9 - Proven to work via the passing of test SE-F-037

There was only a few minor issues that we couldn't resolve in time. The first of which is if a user wanted to go back and didn't do it in the way specifically outlined in the functional requirements it would cause the program to stop working correctly.

When the back button is pressed, the onResume method for the Activity lift cycle is called which in turn calls the onCreate method. The Google Map is created in this method which means that when the button is pressed it overwrites the original map with all the tracking data on it.

As an oversight by the group, we didn't realise we had to calculate the length that a walk was until it was clarified in a group email so this function is also missing.

From a documentation stand point all of the amendments given to us from previous hand in dates have been made so initial issues with those documents should no longer be there.

### 2.1.4 Performance of each team member

**Ben Rainbow**

Ben's primary role as QA manager was to ensure that the documents and code produced were in line with the standards given to us for this project. He has also checked over all the documents for spelling and grammar and making them coherent. He had to collate the pieces of documentation that was distributed across the team into a singular document for each submission (when needed) and also been the one that has primarily done the minutes for any meetings we have had on and off timetable.

## Chris Edwards

Chris has been in charge of the coding for the Java application. Throughout the project he has had heavy involvement in the design aspects of the app and written bits of the documentation. This includes the basic design of the windows and their descriptions for the project plan and descriptions of the significant classes and how they map to the functional requirements of the project.

## Douglas Gardner

Douglas' primary responsibility has been the coding of the server side aspect of the project. Like Chris, Douglas has also had heavy involvement how the design of the project and has written detailed structure for the web side of the project in the design specification. In addition to this, Douglas has been the one that has been in contact with the timetabling office to find us a consistent room for our meetings and the one that set up our GitHub structure.

## Luke Horwood

Luke has been a coder that was meant to code the Java application and the server side, but has ended up being the primary coder for the website's functionality. Like the other team members, Luke has had a contribution to the design of this program and has written up the class interfaces and made contributions to the UML diagrams.

## Jostein Kristiansen

Jostein is primarily in charge of the testing aspects of the project. He has produced the detailed risk analysis in the project plan and produced the test specification and is also heavily involved with the carrying out of the tests on our final solution. Jostein is also writing up parts of the final report.

## Ashley Smith

Ashley's primary role has been to work on various parts of documentation. He made contributions to the project in the initial stages by making suggestions of ways to go about implementing some ideas and also written bits of documentation. He has also been one of the people to make amendments to our documents to fall in line with the feedback that's given.

## Dillon Cuffe

Dillon as deputy project leader has been a secondary point of call for the group. If I was not able to take a meeting then Dillon would instead head the meetings. Dillon has also been involved with the documentation for the project including helping Ashley make amendments to previous bits of documentation and is also a big part of the final report writing up various aspects of that.

## James Woodside

My role in this project was to be the project leader for the group. I've been responsible for the organisation and heading of our meetings both, on and off the timetable. Once we had gone over the tasks to be done by the

following meeting, I have had to give out new work for people to getting on with trying to keep the work load as evenly spread as possible.

### 2.1.5 Critical evaluation of the team and the project

#### *How did the team perform as a whole, and how could that have been improved?*

When referring to the evaluation of the group and how well the group worked together I feel that the group worked well, they were minimal issues with the project. At the beginning of the project there were issues with roles and what individual roles that each person had within the group project. However after a meeting for a number of times there was a clear understanding with the group leader that he would delegate the roles to the individuals that he felt where capable and who at the beginning stated which parts of the project that they as an individual would be confident in.

#### *How could the project have been improved?*

There are a number of ways that the project could have been improved, such as a more in depth Android lecture, offering a wider variety of knowledge and understanding for the coders. As well as this there were issues with the server side part of the project and so I felt that there could have been more information or guidance for this part of the project.
As well as this I also feel that the group could of benefited with additional meetings, as a group we did meet at frequent times and discuss the issues that were needed to be fixed, however I feel that the project could have benefited from additional meetings to allow for more interaction and flowing of ideas between everyone in the group.
However I do feel that as a whole the project worked well, the team worked well together and there seemed to be minimal issues or problems throughout the whole development of the project. I felt that we had a good structure and good leadership; all individuals knew their roles and followed these roles well.

#### *What were the most important lessons learned about software projects and about working in teams?*

Through the development of the application there were a number of lessons that I felt were important.
An important lesson that was learnt about a software project can be seen in the fact that each individual has their own skills and input into the project. Each person brings their own skills to the table and this was essential in the project. It allowed the individuals who were keen software developers to work on the script side of the project, this involved developing the code and implementing the website and server side of the project. Then individuals that believed that coding was their weak point worked on the documentation side as well as testing the code that the coders had developed.

Another important lesson when working in a team was to listen to your leader and follow the roles that he assigned to you, this allows for all team members to know what their roles were and not to have two people working on the same area of the project which could cause duplication.

Another lesson learnt when working in a team can be seen by the communication that we had within the group. The communication among group members is essential, this allows everyone to see where the group is and make sure we are heading in the right direction, as well as letting us know how much more work we need to do, if any. This communication also offered a large amount of feedback from the people in the group who reviewed each piece of document that was developed and so offered a higher chance of getting the problems and errors corrected before the uploading the documents.

Another lesson learnt working with the group and developing the application, was that everyone in the group had the same common goals, which are to achieve the highest grade possible. There may be issues in a group that some individuals are doing more work than others and this may cause tension, this may be true but if all individuals work together they saw that we all had the same goal and wanted to achieve this.

## 2.2 Appendices

### 2.2.1 The project Test Report

# 1. TEST LOG FORM

| | | |
|---|---|---|
| Test Log No: 2 | Group: 02 | Testers(s): jaw57, jok13 |
| Date: 30/01/14 | Tagged version ID: 1.4 | |

| Test ID | Pass / Fail | Fail description | CCF / issue # |
|---|---|---|---|
| SE-F-001 | Pass | N/A | |
| SE-F-002 | Pass | N/A | |
| SE-F-003 | Pass | N/A | |
| SE-F-004 | Pass | N/A | |
| SE-F-005 | Pass | N/A | |
| SE-F-006 | Pass | N/A | |
| SE-F-007 | Pass | N/A | |
| SE-F-008 | Pass | N/A | |
| SE-F-009 | Pass | N/A | |
| SE-F-010 | Pass | N/A | |
| SE-F-011 | Pass | N/A | |
| SE-F-012 | Pass | N/A | |
| SE-F-013 | Pass | N/A | |
| SE-F-014 | Pass | N/A | |
| SE-F-015 | Pass | N/A | |
| SE-F-016 | Pass | N/A | |
| SE-F-017 | Pass | N/A | |
| SE-F-018 | Pass | N/A | |
| SE-F-019 | Pass | N/A | |
| SE-F-020 | Pass | N/A | |
| SE-F-021 | Pass | N/A | |
| SE-F-022 | Pass | N/A | |
| SE-F-023 | Pass | N/A | |
| SE-F-024 | Pass | N/A | |
| SE-F-025 | N/A | Not a required test. Not tested | |
| SE-F-026 | Pass | N/A | |
| SE-F-027 | Pass | N/A | |

| Test ID | Pass / Fail | Fail description | CCF / issue # |
|---|---|---|---|
| SE-F-028 | Pass | N/A | |
| SE-F-029 | Pass | N/A | |
| SE-F-030 | Fail | Future maintenance task | 6 |
| SE-F-031 | Pass | N/A | |
| SE-F-032 | Pass | N/A | |
| SE-F-033 | N/A | Not a functional requirement so not tested | |
| SE-F-034 | Pass | N/A | |
| SE-F-035 | Pass | N/A | |
| SE-F-036 | Pass | N/A | |
| SE-F-037 | Pass | N/A | |

### 2.2.2 The project Maintenance manual

#### Program description

The WalkingTours application is designed to run on Android based mobile devices targeting the minimum API level 2.3 (also known as Gingerbread). The application is designed to provide the necessary interfaces/utilities which a user can use to record a walk.

The recording functionality of the application refers to the process of tracking the devices' GPS coordinates and displaying their current position on a localised Google Map (based on the GPS coordinates returned). Also in order to record a walk the application enables the ability to add a point of interest to the walk which in turn accesses the devices camera/gallery, this point of interest will be shown on the Google Map as a new Marker with custom details and icons.

The user can then upload or delete their walk depending on their needs. The walk recordings that get uploaded are viewable on the web client thanks to PHP being executed on the server side and a MySQL database.

#### Program structure

**Flow of control pseudo code for the Android application:**

application is launched and start screen is displayed
click the create new walk button
the CreateWalk Activity is displayed
enter the information requested and click record walk
    while walk name && walk short description && walk long description are not empty
        while walk name && short description && long walk validate
            progress to WalkRecording Activity
        else
            to ask user about incorrect input and why this is
WalkRecording Activity starts
    retrieve devices GPS and pass information to Google Map Object
        if GPS is available (is Internet connection ready)
            display local map
        else
            use last know location and display map based off that

display users' location on the Google Map
user clicks new POI button
CreatNewPOI Activity starts for result
user enters the information requested
click add image button
prompt user for gallery to camera selection
if user selects camera
launch camera intent and create a new File and store/return captured image in this file
if user selects gallery
launch gallery intent and return the absolute path of the image selected
click create button
while POI name && POI description are not empty
while POI name && POI description pass their validation
if image is not null
create a new PointOfInterest Object with the above information
parce this Object using the Parcelable interface
pass this parce Object back to the WalkRecording Activity
add this new PointOfInterest to the Vector of PointsOfInterest
Add a Marker to Google Map with this information
user clicks edit walk button
add extras to this intent
these extras will be the current walk name, walk short description and walk long description
start EditWalkActivity for result
retrieve the passed in intent Extras
set the EditText's on EditWalk Screen to the corresponding passed in values
walk name → passed in walk name extra
walk short description → passed in walk short description extra
walk long description → passed in walk long description
user can now edit the walks information
click update info button when happy with edits
set the result and put the new values as extras
retrieve these extras using onActivityResult ()
Update the walks information accordingly
user clicks upload Button
put the walks current information as intent extras
start ConfirmUploadActivity
retrieve the passed in walk details
display these to the user as a quick summary of the current walk

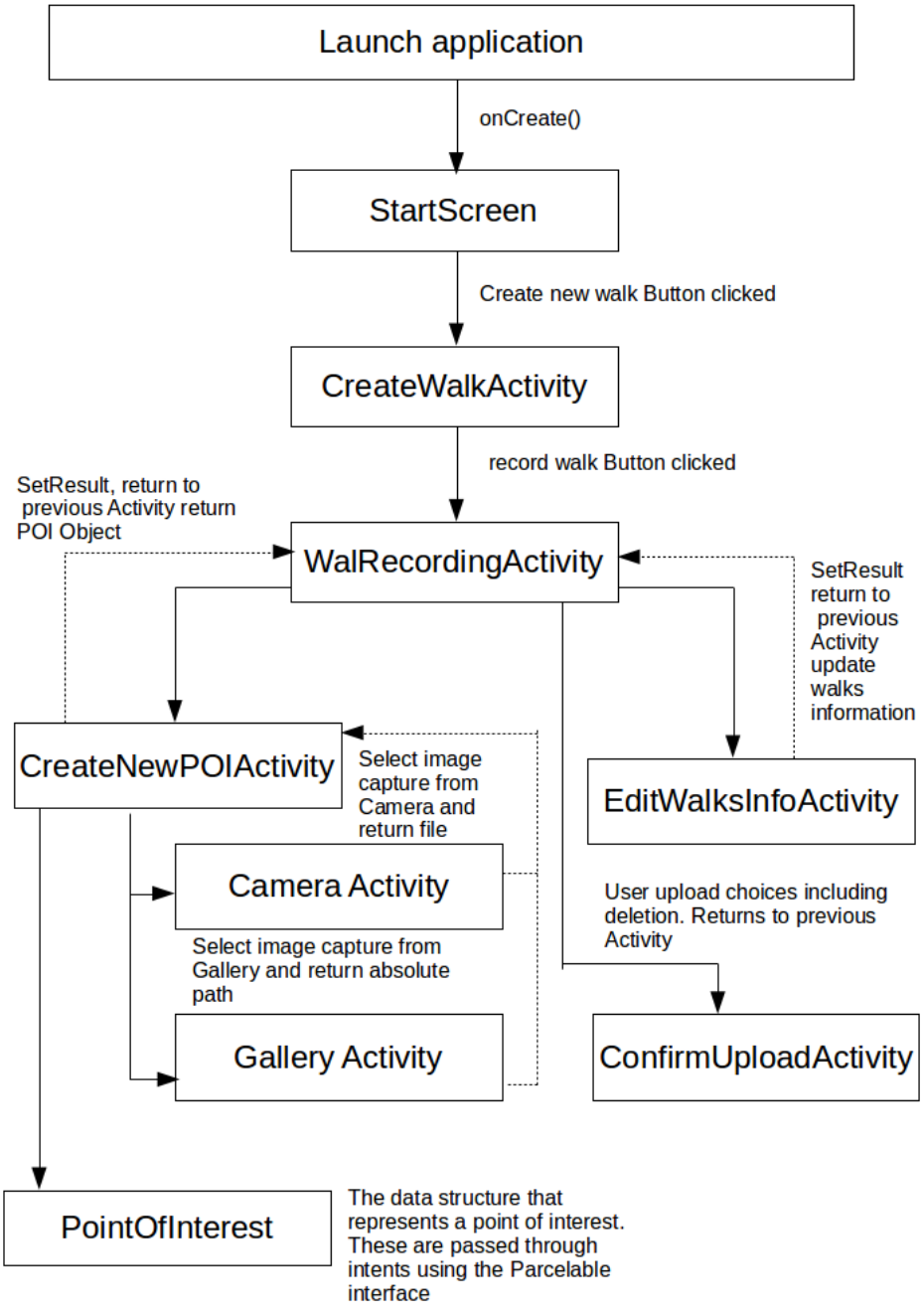
user click confirm button
prompt for upload confirmation
if user accepts upload
open URL connection to upload.php file
set up HTTP POST request
start AsyncTask and pass in POI Vector as argument
loop through the passed in Vector
Create JSON Object to replicate schema specified in design doc

section 5.4.3.4.1
populate JSON with
information from Vector
POST this JSON
to the sever

else

prompt the user about walk deletion
if user aspects deletion
prompt again reinforcing deletion policy
if user aspects deletion
delete walk and clear Google Map
finish Activity
else
return to ConfirmUploadActivity
finish
continue recording walk.

**Flow of Control Diagram:**



**Program modules:**

**Start Screen:**

This is the basic interface for executing the initial procedures which start the application. This module does not have an awful amount of functionality it is effectively a "welcome" screen so the user knows exactly what they are viewing as well as providing the ability for the user to open the application and then close it again without having to start a recording.

**Start Screen methods:**

**protected void onCreate (**Bundle**)**

This is part of the Android programming language/specification. It is called upon an Activities creation. This method takes a Bundle as a parameter argument which can be used to save the state of your Activity when switching between applications and or Activities.

**public void addListenerOnButtons ()**

This method enables the user interface to allow user interaction events. In this instance it enables the button clicks to execute a new intent and start the corresponding Activity.

**CreateWalkActivity**

This is the Activity which provides the user interface for setting up the walk correctly. This Activity holds the user interface elements that provide the opportunity to give the walk a name and both short and long descriptions.

**CreateWalkActivity methods:**

**protected void onCreate (**Bundle**)**

Please see previous method description for more detail.

**public void setEditTexts ()**

This method instantiates the EditText variables, which enables the content of these EditText variables to be set and or retrieved. This method also applies to correct content restrictions to each of the EditText to sanitise the data entry and prevent possible complications which can arise from malformed data entry.

**public void addListenerOnButtons ()**

This method enables the user interface to allow user interaction events. In this instance it enables the button clicks to execute the associated instruction(s).

**public** String **getWalkTitleText ()**

This method simply returns the value of the String value of the EditText Object which houses the walks name.

**public** String **getWalkShortDescText ()**

This method simply returns the value of the String value of the EditText Object which houses the walks short description.

**public** String **getWalkLongDescText ()**

This method simply returns the value of the String value of the EditText Object which houses the walks long description.

**public boolean checkInputLength (**String**)**

This method is used as part of the input validation for each of the EditText Objects. It returns a boolean value depending on the result of the validation check. If the passed in String has a length of greater than 0 then true is returned, else false is returned. This method prevents "null" Strings being set for the walks information.

**WalkRecording:**

This Activity provides the main interface and access to the main algorithms associated with the actual recording of the walk. This is where the GPS coordinates are recorded as tracked; also the Google Map Object is displayed to the user at this time. From this screen you can start the upload process, edit the walks information and also add new PointsOfInterest to the current walk. This Class extends FragmentActivity so a Google Map Object can be displayed.

**WalkRecording methods:**

**protected void onCreate (**Bundle**)**

Please see previous method description for more detail.

**Public void AddOnClickListeners ()**

This method enables the user interface to allow user interaction events. In this instance it enables the button clicks to execute the associated instruction(s).

**public static** Vector<PointOfInterest> **getPos ()**

Returns the current Vector<PointOfInterest> which is statically accessible and belongs to the WalkRecording Class. This is so a new Vector<PointOfInterest> is not accidently created.

**protected void onActivityResult (int** requestCode**, int** resultCode**,** Intent data**)**

The method is provided as part of the Android language specification. It is used to handle Activities initiate using StartActivityForResult (Intent). This method handles multiple Activity results which are segregated based on requestCode and resultCode. The Intent parameter argument is used to provide the data passed in from the Activity you are expecting the result from.

**public static** String **getWalkName ()**

Returns the current value of the String variable related to the walks name.

**public static void setWalkName (**String)

Set the current value of the String variable related to the walks name.

**public static** String **getWalkSDesc ()**

Returns the value of the String variable related to the walks short description.

**public static void setWalkSDesc (**String**)**

Set the current value of the String variable related to the walks short description.

**public** String **getWalkLdesc ()**

This method is used to return the value of the String variable related to the walks long description.

**public void setWalkLdesc (**String walkLdesc**)**

This method is used to set the value of the String variable related to the walks long description.

**public void addPOIToMap (**PointOfInterest**)**

This method is used to add a new PointOfInterest to the Google Map. The argument PointOfInterest will be used and the value returned from this Object will be represented as a marker on the Google Map Object using the MarkerOption () parameter of the Map.addMarker method. This method also called the getResizedBitmap method in order to provide a custom image to the marker.

**public** Bitmap **getResizedBitmap (**Bitmap**, int, int)**

This method is used to return a new Bitmap image that will be resized based on the value of the passed in Integer arguments. The first integer argument is the new Bitmap's height and then later is related to the new Bitmap's width. The first argument is the Bitmap you wish to resize, and it uses a matrix to resize the Bitmap image.

**public void startCountingTimer ()**

This method is used to display a timer and the current time stamp for each PointOfInterest and the walk itself. To do this it performs a calculation based on the system time in milliseconds and calculates the time difference between the two times. A conversion is then made to convert the milliseconds difference between the two times into an hh:mm:ss format.

**public** String **getTimeStamp ()**

This method returns the value of the String variable related to the time stamp.

**public void stopTimer ()**

This method sets the while loop condition within the startCountingTimer () method to true. This stops the while loop thus stopping the timer.

**public void deleteWalk ()**

This method is used to delete the walks information. During this deletion the following actions take place; first the Google Map object is cleared using Map.clear () then the Vector<PointOfInterest> is wiped using removeAllElements (). Then all the walk dependent variables are reset to default so all Strings (name and descriptions) are set to the value “” (empty String). Finally the time stamp String is reset to 00:00:00 and its text view is updated.

**public** GoogleMap **getMap ()**

This method returns the current Google Map Object being used to display the walk to the user.

**public void onDestroy ()**

This method is provided as part of the Android language specification. This is the final method call in the Activity life cycle and it is used to terminate and destroy the current Activity from the devices memory stack.

**CreateNewPOIActivity:**

This Class is designed to provide the interface required for the user to be able to create new PointOfInterest Objects. This Activity also has the option to launch the Camera or Gallery intent; this is required to progress through the creation process.

**CreateNewPOIActivity methods:**

**protected void onCreate (**Bundle**)**

Please see previous method description for more information.

**public void setEditText ()**

This method instantiates the EditText variables, which enables the content of these EditText to be set and or retrieved. This method also applies to correct content restrictions to each of the EditText to sanitise the data entry and prevent possible complications which can arise from malformed data entry.

**public void addOnClickListeners ()**

This method enables the user interface to allow user interaction events. In this instance it enables the button clicks to execute the associated instruction(s).

**public** String **getWPOINameText ()**

This method returns the value of the String variable related to the EditText for the PointOfInterest name data entry.

**public** String **getPOIDescText ()**

This method returns the value of the String variable related to the EditText for the PointOfInterest description data entry.

**public void addPointOfInterest (double**, double, String**)**

This method I used to actually create new PointOfInterest Objects with the specified values. The first double parameter argument refers to the latitude of the PointOfInterest, the second parameter is the longitude position of the PointOfInterest and the String parameter refers to the time stamp for the PointOfInterest on the current walk. This method then sets the result using setResult so the created PointOfInterest Object can be accessed by the WalkRecording Class.

**public void buildImageSelectionPrompt ()**

This displays a dialog to the user, prompting them for a choice. This choice refers to the method they want to use to attach an image to a PointOfInterest, Camera or Gallery. Depending on their choice the camera intent or the gallery intent is launched and the return values from these methods are handled elsewhere. This method simply handles the users' choice and not the data associated with the choice.

**private void dispatchTakePictureIntent ()**

The purpose of this method is to enable the user to switch the devices camera application in order to capture a fresh image. First of all a new file is created using createImageFile () so there is a location to store the image. Once the user is happy with the photo they have taken, the newly related path is stored. This path is passed around the application to allow for efficient manipulation of the image file at any time.

**private** File **createImageFile ()**

This method returns a file Object. It creates a new file and returns an object that represents that file. This file can be used to store images from the camera and maintain a location which can be accessed at any time to efficiently manipulate the image. The created file will be stored in the devices DIRECTORY_PICTURES.

**public void dispatchGalleryIntent ()**

This method switches the users focus and launches the devices media gallery. Once this intent has been launched the user can select an image which will be attached to the PointOfInterest. The Gallery intent is started using startActivityForResult.

**protected void onActivityResult (int** requestCode**, int** resultCode**,** Intent data**)**

The method is provided as part of the Android language specification. It is used to handle Activities initiate using StartActivityForResult (Intent). This method handles multiple Activity results which are segregated based on requestCode and resultCode. The Intent parameter argument is used to provide the data passed in from the Activity you are expecting the result from.

**public** String **getRealPathFromURI (**Context, Uri)

This method is required in order to retrieve the actual file path from a file (image) returned from gallery selection as the gallery will hide the real file path. In order to retrieve this file path a cursor is used in conjunction with a ContentResolver query in order to parse the URI parameter until the absolute path is located. This absolute path will then be returned as a String. The context parameter is simply the context in which this was called, e.g. the Activity you are currently in and the URI parameter is the URI of the file you wish to find the true path for.

**public boolean checkInputLength (**String**)**

This method is used as part of the input validation for each of the EditText Objects. It returns a boolean value depending on the result of the validation check. If the passed in String has a length greater than 0 then true is returned, else false is returned. This method prevents “null” Strings being set for the walks information.

**public void setFilters ()**

This method is used to apply input sanitation to the EditText's associated with setting the data required for every PointOfInterest. This input sanitation refers to character and size limit restrictions to prevent a breach of the functional requirements. Characters are limited to a-z, A-Z, 0-9 only, with the inclusion of the ' ' (space) character for the description EditText.

**ConfirmUploadActivity:**

This Activity is used to HTTP POST the walks information to the server in the form of a JSON Object with the scheme specific in section 5.4.3.4.1 of the design specification. This Activity uses an AsyncTask to perform said task because in Android you cannot perform network connections on the main or UI Threads.

**ConfirmUploadActivity methods:**

**protected void onCreate(**Bundle**)**

Please see previous method description for more information.

**public void buildUserPrompt()**

This method constructs a Dialog which prompts the user for input in regards to a YES/NO choice. These YES/NO values are represented as upload to server (YES) or delete walk (NO) depending on the users input a different route through the Activity is followed. If the user does in fact want to upload to the server, then the uploadToServer method is executed, else the deletion prompt is shown.

**public void buildDeletionPrompt()**

This method constructs a Dialog box which prompts the user for input in regards to a YES/NO choice. If the user selections YES, then the walk will be deleted and you will be returned to the WalkRecording, else the prompt will be removed from view and the user will remain with this Activity in focus.

**private void showUploadMessage()**

This method simply used the AlertDialog builder to display a message of upload confirmation to the user.

**public void locateUIElements()**

This method is used to locate the user interface elements for this Activity. This method enables the ability to add functionality to the Activity such as changing the context of the TextViews.

**public void addOnClickListeners()**

This method enables the user interface to allow user interaction events. In this instance it enables the button clicks to execute the associated instruction(s).

**public void uploadToServer(**Vector<PointOfInterest> **)**

This method handles the server upload algorithm(s). It takes the current Vector of PointOfInterest Objects (so all the PointOfInterest Objects associated with this walk currently) and parses them into a JSON Object with the scheme specified in 5.4.3.4.1 in the design specification. In order to parse this Vector this method makes use of the AsyncTask and its doInBackground method to prevent performance problems.

**private static** String **convertInputStreamToString(**InputStream **) throws** IOException

This InputStream converter is used to debug message received from the server. It will convert the server response (InputSream parameter) into a String which can easily be read/ displayed to help with the debugging process. An IOException can be thrown by this method is there is a problem with the associated BufferedReader.

**private class UploadASyncTask extends** AsyncTask<JSONObject**,** Void**,** Void>

This nested private Class is used to execute the HTTP POST request for the upload of the walks data. You have to use an interface such as the AsyncTask in order to perform these tasks in Android as you cannot do so on the main Thread. This method converts the final JSON Object into a String and that is the data that is sent to the server for parsing.

**protected** Void **doInBackground(**JSONObject**...)**

This is the "worker" method for the server upload algorithm. This is expected to run in the background thus, it does not free the application when performing intensive tasks. This method is what actually sends the JSON String and handled the server response.

**private** String **fileToBitmapAndEncode(**File**)**

This method is used to encode any file into a based 64 String which can then be sent to server for parsing. The file parameter is the file stored in each PointOfInterest Object, this file is then resized so it is easily sent across the server reducing the strain and the base 64 String is encoded and returned.

**private static** String **getStringFromBitmap(**Bitmap**)**

This method is used to convert a Bitmap image into a base 64 String encoded JPEG. In order to do this it first turns the image into a byte array, which is then encoded into this returned base 64 String representation of the JPEG image. The compression quality is set to 100, which provides a suitable level of image quality.

**public** Bitmap **getResizedBitmap(**Bitmap **, int , int)**

This method is used to return a new Bitmap image that will be resized based on the value of the passed in Integer arguments. The first integer argument is the new Bitmap's height and then later is related to the new Bitmap's width. The first argument is the Bitmap you wish to resize, and it uses a matrix to resize the Bitmap image.

**public int conertTimeStringToSeconds(**String timeString**)**

The time stamp is stored as a string and in order to send the time string and store it correctly in the database (see design specification 5.4.2.1) the String must be sent as a numerical representation. To do this efficiently the String is split base on the ':' character providing 3 String arrays which represent [hh][mm][ss] these can then be individually evaluated and turned into seconds which can be added together and sent as a total number of seconds to the server, this total can easily be transformed back into hh:mm:ss.

**EditWalksInfoActivity:**

The Activity is used to edit the walks information during the recording process. This Class provides the necessary methods and user interface elements required in order for the user to make run time adjustments to the walks name, short description and its long description. These updates are reflected immediately.

**EditWalksInfoActivity methods:**

**protected void onCreate(**Bundle**)**

Please see previous method descriptions for more information.

**public void locateUIElements()**

This method is used to locate the user interface elements for this Activity. This method enables the ability to add functionality to the Activity such as changing the context of the TextViews. This method also applies to the correct content restrictions to each of the EditText to sanitise the data entry and prevent possible complications which can arise from malformed data entry; these restrictions are identical to those in place in the CreateWalkActivity.

**public void addListenerOnButtons()**

This method enables the user interface to allow user interaction events. In this instance it enables the button clicks to execute a new intent and start the corresponding Activity.

**public** String **getWalkTitleText()**

This returns the contents of the EditText responsible for the data input for the walks title in String form.

**public** String **getWalkShortDescText()**

This returns the contents of the EditText responsible for the data input for the walks short description in String form.

**public** String **getWalkLongDescText()**

This returns the contents of the EditText responsible for the data input for the walks long description in String form.

**public void setEditTextValues()**

This method uses the information passed to the Activity using the WalkRecording Activity. The information passed to this Activity represents the walks current name and its short and long descriptions. If any of these passed in values are NULL then this method does nothing, if none of them are NULL then there EditText's on the screen will be updated to reflect the walks current information.

**public boolean checkInputLength(**String**)**

This method is used as part of the input validation for each of the EditText Objects. It returns a boolean value depending on the result of the validation check. If the passed in String has a length of greater than 0 then true is returned, else false is returned. This method prevents "null" Strings being set for the walks information.

**HelpScreen:**

This Activity is used to provide a new user with basic information to help them navigate and get the maximum amount of functionality from this application. It uses a very simple method of navigation to switch the help screen information, and can be accessed from every page.

**HelpScreen methods:**

**protected void onCreate(**Bundle**)**

For more information on this method please see the previous methods.

**public void setFirstScreen()**

This method located the first element in the help screen resources Vector<String> and sends the main content panel (TextView) to reflect this value.

**Public void AddOnClickListeners()**

This method enables the user interface to allow user interaction events. In this instance it enables the button clicks to execute the associated instruction(s).

**public void switchToNextHelpText ()**

This is used to update the main TextView on screen to reflect a new paragraph of help text. In order to do this the current help screen's information is tracked in the Vector<String> of resources, then this is incremented and the next index resource is selected and displayed to the user. This only happens if there is a next resource in the Vector<String> (current index +1 < Vector.size ()).

**public void switchToPrevHelpText ()**

This is used to update the main TextView on screen to reflect a new paragraph of help text. In order to do this the current help screen information's location in the Vector<String> of resources is tracked, then this is decremented and the previous index resource is selected and displayed to the user. This only happens if there is a previous resource in the Vector<String> (so if current index! = 0).

**private void populateHelpList ()**

This method is used to fill the Vector<String> with the necessary String resources which represent the help screen content for each page. Add any new help resources to this method body.

**PointOfInterest:**

This Class is used to represent exactly what every PointOfInterest should look like and what data is can contain. This Class also implements the Parcelable interface so these Objects can easily be passed between Activities efficiently at run time. This is one of the main data structures for this application.

**PointOfInterest methods:**

**public PointOfInterest (**String**,** String**, double** l**, double**, String)

Constructor for the PointOfInterest Objects. This creates a new PointOfInterest Object in memory with the specified values which refer to; name, description, longitude, latitude and the time stamp. This is called by the CreateNewPOIActivity.

**public PointOfInterest (**Parcel**)**

The Parcelable constructor. This is to decode the encoded Parcelable Objects back into their pre-encoded form. This is to again allow the application to easily manipulate these Objects between Classes/Activities.

**public** String **getImagePath ()**

Returns the file path for the image associated with this PointOfInterest Object, this will be in String form.

**public void addImage (**File)

This method provides the ability to attach a file which will point to the image you want associate with this PointOfInterest Object. The file parameter will be the file associated with this PointOfInterest Object.

**public** PointOfInterest **makeAndReturnPOI (**String, String, **double, double**, String)

Returns and creates a new PointOfInterest Object. This creates a new PointOfInterest Object in memory with the specified values which refer to; name, description, longitude, latitude and the time stamp. This is called by the CreateNewPOIActivity.

**public void writeToParcel (**Parcel, **int**)

This encoded the PointOfInterest Objects into a Parcel form which can then be decoded back into Object form using **PointOfInterest (**Parcel**)** method. The Parcel parameter is the Parcel destination for the encoded Object and the integer parameter is used to specify any flags/options you wish to use during the encoding.

**public** String **getName ()**

This returns the value of the variable associated with the PointOfInterests name in String form.

**public** String **getDescription ()**

This returns the value of the variable associated with the PointOfInterests description in String form.

**public double getLongitude ()**

This returns the PointOfInterests longitude which is represented as a double.

**public double getLattitude ()**

This returns the PointOfInterests latitude which is represented as a double.

**public** File **getImage ()**

This returns the file (image) associated with this PointOfInterest.

**public** String **getTimeStamp ()**

This returns the value of the variable associated with the time stamp for this PointOfInterest in String form.

**public void setLat (double)**

Specify the value for this PointOfInterest latitude; this must be presented as a double.

**public void setLng (double)**

Specify the value for this PointOfInterest longitude; this must be presented as a double.

**public void setName** (String)

Specify the value for this PointOfInterest name variable; this must be represented as a String.

**public void setDescription** (String)

Specify the value for this PointOfInterest description variable; this must be represented as a String.

**public void setTimeStamp** (String)

Specify the value for this PointOfInterest time stamp variable, this must be represented as a String.

## Algorithms

**Recording the walk/Adding points of interest:**

2. First of all the user starts the Walking Tours application.
3. Then user will now be represented with the Start Screen where the logo is show (a globe) as well as a simple welcome message, a brief message related to the help section and a create new walk button.
4. The user now clicks the "create new walk" button.
5. The createWalkActivity now launches as the user is show the screen titled "Create a new Walk"
6. On this screen there is 3 EditText Widgets all labeled to describe their purpose, they represent the walks name, and both the short and long descriptions. These Edit Texts have sanitation restrictions put in place. A walk name cannot contain ' ' (the space character) nor can it exceed 15 characters in length, the short description cannot exceed 100 characters and the long description cannot exceed 1000 characters. Also, none of the edit texts will accept any characters except for a-zA-Z0-9 and ' ' (excluding the name EditText).
7. To start Recording click the "Record Walk" button
    1. Once this is clicked the application will validate the data inputs to check their length (check they aren't null or length 0) as the other validation takes place as the user types. This size validation is preventing the application crashing due to null entries.
    2. If this validation passed the RecordWalk Activity will start.
8. The user will be greeted with a new screen with the walk name they entered in step 6.1 at the top of the screen above the displayed Google Map Object. When this Activity is created the devices GPS location is retrieved using NETWORK communications through the devices Internet connection (Wi-Fi) and the map Object's location reflects this by displaying a localised map, with a zoom level of 14 of the local area. The users' current location will be displayed on the map using the Google Map's API map**.**setMyLocationEnabled **(true)** the user will appear as a blue dot at their current location. This updates real time so they have a visual representation of where they currently are/heading. At this stage a static Vector<PointOfInterest> in created.
9. The user has 3 options at this screen:
    1. create a new PointOfInterest
        - Once the "new POI" button is clicked, the devices current GPS coordinates are again retrieved using LocationManager and these coordinate values for the latitude and longitude and added to an intent as extras with suitable String key values. The time stamp is also retrieved as that stage. The time stamp is calculated using the time the application was started based on the devices current time in milliseconds, you then subtract the current time in milliseconds away from this start time and get the absolute value. You now have the run time in milliseconds, to convert this to seconds divide by 1000. From this time in seconds you can convert to hh:mm:ss using hours = seconds / 3600, minutes = seconds / 60, seconds = seconds % 60.
        - The CreateNewPOIActivity is then started using startAcitivityForResult with the intent with the added extra data used as the starting intent. In the onCreate method for CreateNewPOIActivity these passed in extras as retrieved using the specifies String keys in step 8.1.1 and locally stored.
        - The user is now required to enter the information in the EditText' on screen, these relate to the

PointOfInterests name and description. These EditText's also have data input sanitation where the only values allowed are a-zA-Z0-9 and ' ' (' ' is not allowed in the name EditText). In addition the name cannot exceed 15 characters and the description cannot exceed 100.

- Once this information has been correctly entered the user must add an image, click the "add image" button.
    1. A prompt is now displayed asking the user if they want to use an existing image (from device gallery) or capture a new image (using device's camera).
        1. If the user opts to use the device gallery then the gallery intent is launch using the content filter of image/* limiting the selection to images so videos cannot break the application.
            1. One you have selected the image, you are returned to the Walking Tours application in the same position as you were before. You must now get the absolute path from the returned camera image so you can pass this through the application as a String and encode the image later on.
            2. In order to do this, see the details on the getRealPathFromURI method in the previous section.
    2. If the user opts to use the devices camera, you first need to create a File in which this new image will be stored; this file will be in the external media directory on the device. For more information on this see the method detail for createImageFile () in the above section.
        1. The camera intent will now launch. The user will take a photo as normal as when they are happy with the photograph they will be returned to the same location with no information loss.
            1. This newly taken photo is now stored in the external media storage on the device you retrieve the absolute file path for this and you can now access this image at any time.
- Once the image selection has taken place, the user clicks the "create" button.
    1. Validation I performed to check that:
        1. none of the data fields are null or of length 0 (name, description)
        2. check that there is a file associated with the PointOfInterest
    2. If 1.1 and 1.2 pass then a new PointOfInterest is created with the name and description specified in stage 3, the coordinate from stage 2 and the image file from stage 4. This new PointOfInterest Object is then encoded into its parcelable form and set as the result of this Activity.
- On return to the WalkRecording Class the result set in section 5 is evaluated and checked for its origin and if the result was OK.
    1. If these tests turn out to be in fact from stage 5.2 then the PointOfInterest is then decoded back into its Object form and added to the Vector<PointOfInterest>.
    2. Once it has been added to the Vector<PointOfInterest> it also needs to be added and displayed to the user on the Google Map. To do this, you must create a custom Marker using the MarkerOptions utility provided by the Google Maps API.
        1. To make add a custom icon to a Marker you need to convert the image to the Bitmap file format. To do this you retrieve the file path stored for the PointOfInterest you created in step 5.2. With this retrieve path you can pass it as a parameter to the BitmapFactory.decodeFile (file path) method which returns a Bitmap image.
            1. This image will be very large, so the image is now resized using getResizedBitMap method which resizes the image to 100x100 pixels.
                1. To do this it divides the full size Bitmap width/height by the new width/height and uses these values in a Matrix Object In order to achieve a scaled version of the Bitmap image.
        2. Once you have resized the image step 2.1.1.1 you can now add the rest of the data to the Marker
            1. To do this use the attributes provides by the Marker API and simply getters/setters for the encapsulated PointOfInterest Object(s)
                1. example:
                    1. .title( PointOfInterest.getName())
                    2. .snipper( PointOfInterest.getDescription() + " " + PointOfInterest.getTimeStamp())
                2. Steps 1.1 and 1.2 will create a marker with a title and a description which is visible when clicked. You then want to set the icon for this Marker to the

image created in step 2.1.1.1.
3. Once you have completed steps 2.1.1 the Marker will be shown on the Google Map.

- Repeats steps 1 through for an PointOfInterest you want to add.

2. Edit the walks information
   - in order to edits the walks information: the walk's name, and both its short and long descriptions the user clicks the “Edit walk” button
     1. Before the EditWalksInfo Activity is started the current walk information must be retrieved from the WalkRecording Activity (our current location) these variables will then be put into a new Intent using as intent data extras with corresponding sensible String keys so they can easily retrieved.
        1. Once step 1.1 has been completed the EditWalksActivity will be started using startActivityForResult with the new Intent with the added extras as the parameter.
     2. Once the EditWalksActivity has started the extras put into the intent will be retrieved and the values will be displayed in the corresponding EditText locations.
     3. The user can now edit these values:
        1. The edited values must meet the restrictions specified early in stage 5.
           1. if these validations pass the user can click the update info button
              1. A check is performed to check the updated values length (check it is large than 0 to prevent “”) and also check against NULL entities.
                 1. If stage 1.1.1 passed all OK, then create a new Intent and put the new updated values as extras and set this Intent as the result.
        2. On return to the WalkRecording Activity check the result set in stage 1.1.1.1 and check the result == RESULT_OK
           1. If so then updates the walk's information.
        3. Repeat step 2 as many times as you need to.
3. Upload the walk to the server
   - see Uploading to server below

**Uploading data to the sever:**

1. The user clicks to the “upload” button
   1. Before the ConfirmUploadActivity is started a new Intent is created. To this intent the walks current information is passed as separate intent extras with corresponding String keys, that is the walks name and both its short and long descriptions.
      1. The ConfirmUploadActivity is then started with the Intent created in stage 1.1 as the parameter.
2. Inside the onCreate method for the ConfirmUploadActivity the extras put in the Intent in stage 1.1 are retrieved and locally stored in variables. The walk name and walk short descriptions are then displayed to the user as a brief summary of the walk.
3. Once to user is ready to upload, click the “confirm” button
   1. A prompt will be displayed asking the user if they want to upload the walk or delete the walk
      1. if at stage 3.1 the user selects the want to upload the walk:
         1. The static Vector<PointOfInterest> is retrieved using the getter provided In the WalkRecording Class, this is then passed to the uploadToServer method.
         2. The upload to server method will generate the security hash and salt to pass server authentication the user does not know about these, nor do they matter to the user.
         3. A JSON Object is created using several nested JSON Objects/Arrays matching the schema in section 5.3.4.3.1. This JSON Object is then populated with the correct values by looping through the retrieve Vector<PointOfInterest>.
            1. for every element in Vector<PointOfInterest>
               1. Convert the current PointOfInterests time stamp to seconds using convertTimeStringToSeconds
               2. encode the current PointOfInterest image into base 64 String using fileToBitmapAndEncode
               3. Create a new JSON Array for the images.
               4. create a new JSON Object
                  1. put the current PointOfInterest's name into the stage 3.1.4 JSON Object with the key “name”

2. put the current PointOfInterest's latitude into the stage 3.1.4 JSON Object with the key “latitude”
3. put the current PointOfInterest's longitude into the stage 3.1.4 JSON Object with the key “longitude”
4. put the current PointOfInterest's time stamp (result of stage 1.1) into the stage 3.1.4 JSON Object with the key “timestamp”

5. create a new JSON Array
    1. add the current PointOfInterest's name into the JSON Array created in stage 3.1.5
    2. add the current PointOfInterest's description into the JSON Array created in stage 3.1.5
    3. add the JSON Array created in stage 3.1.5 to the JSON Object created in stage 3
6. add the images to the JSON Array created in stage 3.1.3
    1. Add the file name for the current PointOfInterest’s image with the ".jpeg" extension to the JSON Array referenced in stage 6
    2. add the 64 base encoded image String to the JSON Array referenced in stage 6
7. Add the JSON Array referenced in stage 6 to the JSON Object created in stage 3.
8. add the remaining information to the JSON Object created in stage 3:
    1. add the walk name with the key “title”
    2. add the walks short description with the key “shortDesc”
    3. add the walks long description with the key “longDesc”
    4. add the sever authorisation SHA1 from stage 1.1.2 with the key “auth”
9. Execute the AsyncTask

2. Pass the finalised JSON Object created after stage 1 is complete to the AsyncTask which is a private nested Class.
    1. Execute the doInBackground method using the passed in JSON Object from stage 2 as the parameter
        1. Create a new HttpParams Object
        2. Create a new HttpClient Object with the stage 1 HttpParams as the parameter
        3. Create a new HttpPost Object with the URL String for the upload.php code as the parameter
        4. Create a new List<NameValuePair>
            1. Add the passed in JSON Object's (from stage 2.1) toString to the List<NameValuePair> from stage 2.4 with “data” as the String key.
        5. Create a new UrlEncodedFormEntity with the List<NameValuePair> from stage 4.1 as the parameter
        6. set the UrlEncodedFormEntity (from stage 2.1.5) content encoding to HTTP.UTF_8 using setContentEncoding(HTTP.TUF_8)
        7. set the HttpPost's (from stage 2.1.3) entity as the entity created in stage 2.1.6
        8. Set the HttpPost's (from stage 2.1.3) header to be something meaningful for the server log.
        9. Execute the HttpPost request:
            1. create a HttpResponse Object with the HttpPost object referenced in stage 2.1.9 as the parameter

4. Server upload is now complete.

**The main data areas:**

There are two main data areas which are used and manipulated throughout the Walking Tours application. These two main data areas refer to the design a structure of a point of interest on the walk this is represented as the PointOfInterest Class. The second data structure is contained in WalkRecording Class. The data structure referenced here is a Vector which contains all the current points of interest on the current walk.

**PointOfInterest Object(s):**

A PointOfInterest Object is used to model and represent a location that user wants to add to the current walk. These Objects holds encapsulated data related to a PointOfInterest's name, description it’s GPS coordinated represented as separate longitude and latitude double variables as well as the File associated with the image

attached to this PointOfInterest.

The way in which these Objects are created is using the interface provided in the CreateNewPOIActivity. These Objects implement the Parcelable interface so these Objects can be encoded into Parcel form, which can then easily be passed around Activities in order to maintain references to each PointOfInterest in memory thus allowing them to be added to the Vector<PointOfInterest>. Once these Objects have been decoded from their Parcel form back into their Object form, and added to the Vector<PointOfInterest> the encapsulated data can easily be retrieved by accessing x index in the Vector and using the getter/setters provided as part of the PointOfInterest Class interface.

**Vector<PointOfInterest> in the WalkRecording Class:**

The WalkRecording Class houses a statically created Vector<PointOfInterest> which contains every "location" that has been added to the current walk. These locations are represented using the PointOfInterest Objects as discussed above.

The reason a Vector is used use as the data structure is because Vectors are very effectively Thread safe ArrayLists. The provides us with the ability to specify that a List is required but as opposed to an Array implementation the size does not have to specified. This allows the user to not be limited by a size restriction to the PointOfInterest List.

This Vector is declares as static because it belongs to the Class itself. Also the fact that it is static allows it to be manipulated and accessed by other Activities without having to create a new Vector in each Class that requires access, or indeed needing to pass the whole contents of the Vector through Intent Extras.

The elements in this Vector as accessed by the ConfirmUploadActivity and the uploadToServer method in order to populate the JSON Object specified earlier. Every PointOfInterest element in this Vector is parsed at this stage, and its encapsulated data is retrieved. This is required in order to complete the upload process and pass server validation.

**Files:**

**.java Files:**

Any and all Android source code must be contained in files suffixed with the .java notation and should be complied to produce .class files. These .java files should be placed in the relevant package location relevant to the applications structure. These .class files should be placed in the bin folder, all though this will normally be handled by the Java complier either through an IDE or using the javac command line tool.

**.xml files:**

XML files in Android are used to specify the system resources required. These resources range from the layout of each screen (Activity) to the permissions/hardware the application with need to interact with on the device as well as specifying all the Sting resources you will need to display in your TextView's, Buttons and EditText's.

These files should be place in the res/layout directory of the project (excluding manifest.xml and Strings.xml) and should be suffixed with the .xml extension.

**Android manifest XML file:**

This xml file is extremely important. This manifest file contains all the information required to execute your application as intended. This XML file contains all the permissions your application needs to run as well as the hardware and or the Android Operating System interfaces it will require access to in order to complete its objectives.

Also in this file your activities are specified and declared to be Activities. This tells the Android Operating System to treat them differently to standard Classes allowing them to be brought into focus. If you create an new Activity or to need access device hardware you must specify the permissions in this file or you cannot run/access what you require.

**Strings XML file:**

This file is created by Android for you. This being said you need to define any colours or static String resources in this file so you can access them at or before run time. It is bad practice and not advised in Android to manually define Strings for you Widgets you should always define them in this Strings.xml file.

This file is located in res/values do not move this file location or delete.

**Image file(s):**

Every PointOfInterest Object has an File associated with it. This File represents the image attached to the PointOfInterest. These files will be stored in the Android device's external storage; this will be handled by the Android Operating system. It must be said though, the file created when taking a new image using the devices camera are encoded as a JPEG. Take this into account if you make any modifications to the application.

**Interfaces:**

The application interacts with many externally provided interfaces. These interfaces enable the application to perform several tasks directly related to the functional requirements as specified in the requirements documentation. These tasks include accessing the devices media storage e.g. camera/gallery intents, accessing the devices network state to enable GPS retrieval and accessing the Google API's to display a Google Map Object to the user.

**Camera/gallery**

The Android application specifies in its manifest that it will require access to the camera via Android.hardware.camera inside a user's-feature tag. This is a requirement and without a camera this application cannot be started. Also the manifest specifies that this application will attempt to write to the devices external storage. This is required when you try to create a new image file just before the camera intent in started. Without this permission you have no way of saving the newly captured image, so you could no longer access it again once you have finished with the camera application.

**Accessing devices network connection:**

The manifest specifies that the application will try to access several sections of the Android language specification. These sections refer to the ability to first see if the device is currently connected to Internet and secondly retrieve the GPS coordinates based on the devices current location, the latter is done using the LocationManager Objects and interface provided as part of the Android libraries.

The permissions specified in the manifest are; ACCESS_NETWORK_STATE, ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, ACCESS_MOCK_LOCATION and ACCESSS_WIFI_STATE. These all relate and directly affect the ability to access/evaluate the devices' current network connection state. These permissions are required and should not be removed or changed.

**Using Google Maps for Android:**

In order to display a Google Map Object to the user there is several steps you need to follow:

In the manifest you need to specify the following permissions packageName.permission.MAPS_RECIEVE and com.google.Android.providers.gsf.permission.READ_GSERVICES. These allow the application to access the Google Maps for Android v2 API.

On top of the above, also in the manifest you need to make sure the device that want to run this application has open version 2 (minimum) this is a requirement and without this you cannot run the application, because if you did the Google Map Object cannot be displayed with this version.

The Google API Console is also used in order to register the application for the Google Maps for Android v2 API access. This key is based on your SHA1 key provided upon your Android install. If you do to register this SHA1 key and your package name in the Google API Console under the Google Maps for Android v2 section, you will receive and authentication error upon attempting to render the Google Map Object.

**Suggestions For improvement:**

**Walk will be deleted when the back button is pressed:**

The Android Operating system provides a back button on every screen that is displayed to the user. This button I located near another two buttons which are home and a button which displays all running applications. If the user is on the ConfirmUploadActivity screen and they press this back button either purposely or accidentally then the user will be returned the WalkRecording Activity but the currents walks information will be deleted.

This is obviously not an intended feature, and it happens because the Google Map is created in the onCreate method in the WalkRecording Activity. In order to prevent this the Google Map Object should only be created once and a check should be performed to see if the map has any Markers or if any PointsOfInterest are currently being stored. If either of these turns out to be true, then do no delete the walks information.

**Some minor display changes, some elements are incorrectly positioned:**

This is not related to the functionality of the application but it is related to the aesthetics. On the StartScreen there is a help text, Text Label which tells the user if they click the blue help icon at any time they will be taken to the help screen(s). This Text View should be centered when this applications design is revised as currently it looks out of place as it is shifted to the far left hand side of the screen.

This is very trivial is all honesty, it ranks lowest in order of importance in these discussed maintenance tasks.

**Add the ability to add multiple photos to a PointOfInterest:**

It may be the case that user wants to add multiple photos to a particular PointOfInterest Object. The sever is capable of handling this currently but there is no way to add multiple images to a PointOfInterest through the applications current interface. If the user has already attached an image and they go through the image adding process again then the most recent image will be associated with the PointOfInterest Object and the “old” image will be removed.

This will provide greater functionality to the application as it would be useful to both the application user a the web site user to view multiple images.

This should be fairly simple to implement and will involve a new data structure in the PointOfInterest model; this data structure should hold a list of all Files associated with this PointOfInterest.

**Prompt the user if and why the upload has failed:**

Currently the user is prompted with the same upload message. This message tells the user the that the upload has been completed. Sometimes the upload may fail, and the user will have no knowledge of this so when they upload and receive the upload complete message and go to view their walk on the via the web application they will be confused as to why it is not displaying.

The Android application already has measures in place to retrieve and parse the server's response but currently it does not an awful lot with the response. The server will return a keyword of true if the upload is success, and false if failed. You can then parse this response String and look for the keywords true or false and display an appropriate message to the user.

This should be fairly simple to implement, and improves the users performance significantly.

**Refactor the code:**

Code refactoring is one of the most important maintenance tasks. It is important to re factor the code in order to make other maintenance tasks easier. Also with “clean” re factored code, de bugging/making the existing code more efficiently become much, much easier as you can see what is happening without having “clouded” vision.

Potentially you could refactor this application's source code and extract the Google Map Object into its own Class and set of data structures as well as potentially handling the GPS tracking in a separate Class as well.

There are several other ways you can refactor, these are just two, potential the more useful examples.

**Updates the minimum API level at more versions are released:**

Currently the minimum SDK/API level targeted in Android 2.3 is also known as Gingerbread. This is because there are many devices running version 2.3 still so these users cannot be avoided. This being said the general application target is 4.4 (KitKat) this is the current latest release for the Android Operating System.

In the future it may be the case that 2.3 becomes obsolete so there is no point targeting this version anymore. The code should be updated to reflect the new “minimum” being widely used, and the old methods should be updated to reflect new principle in this newer release.

**Randomly general the authentication salt:**

The sever expects an encrypted String to be used as a method of authentication to check the validation of the HttpPost source entity. Currently this value is hard coded so it is a security flaw as anyone with access to this salt String can easily replicate the authentication.

In order to improve the overall security this String should be randomly generated. This can either be a random String made up on “nonsense” characters e.g. "jlnsabpnakhk" or it could be a word retrieved from a stored dictionary, all through the salt won’t be random itself the word selected will be chosen at random making it very difficult to replicate for every upload.

**Things to watch when making changes:**

**The static Vector<PointOfInterest> can be accessed from any Class:**

The fact that the Vector<PointOfInterest> is static and can get retrieved as any point means that if you are not careful you may accidentally overwrite the current Vector<PointOfInterest>. You must take care when adding to the Vector, you should minimalise the location in which you access the Vector<PointOfInterest> to avoid potential complications that may arise if you miss treat this static property.

**The current minimum targeted API:**

It must be noted when making changes to the application that the minimum target API (2.3 in this current version) may not have or support the features you want to implement. If this is the case, then don't simply change the target minimum version you have to either postpone the update or find a work around that will allow you to achieve the same results.

Adding unsupported features will render the application unusable for users with the minimum API level specified installed on their device this is obviously not want you want. Before adding functionality or updating the existing code take the time to read up about this current minimum API and see if what to want to implement is supported.

**Permissions must be specified in the Android manifest XML file:**

If you want to need to add access to a new external interface/API to the application or even a new Activity then you must specify these in the manifest XML file. If you do not specify these new additions your application will simply not function as you expected, thus rendering the whole process a bit redundant until you add the information to the manifest.

It is a good idea before you start writing code for these new addition to add them to the manifest straight away to avoid the possible complication discussed in the above paragraph.

**Any changes to the PointOfInterest Objects must be reflected in the JSON Object:**

Since the JSON Object houses the information that is sent to the server via a HTTP POST request, any updates to the PointOfInterest data structure must also be added to this JSON Object, and the server team should also be informed. For example if you want to add a new field to a PointOfInterest e.g. Date, then the JSON Object's scheme should have the ability to represent this new information otherwise it will not accurately represent the

full extent of the walks information.

**Physical limitations of the program:**

**Screen resolution:**

Because the layouts for every screen /every form widget (buttons, TextView's etc.) sizes are specified in terms of display pixels if a screen has an incredibly small or large screen resolution the application will look very different. If the device has a very small dpi (display pixels per pinch) then every form widget will appear very large and some elements may get hidden by others, effectively rendering the application unusable. Opposed to this is if a device has a very large dpi value then each form element will appear very small and maybe very difficult to interactive with, causing the same problem of the application being near to unsealable.

**Processing power:**

All though this application isn't very computationally intensive it is always a requirement for any system that it has a suitable amount of processing power. There isn't really a specified minimum for this application but the more power the better, As long as the device can run smooth when operating with the Android Operating system installed it should be able to run this Walking Tours application without any problems.

**No camera on the device:**

If the device running the Walking Tours application do not have a built in or externally attached camera then this application will not run on this device. This is because in the Android manifest the camera hardware is specified as a requirement, so this restriction is unavoidable.

**Lack of network connection ability:**

If the device running the walking tour application do not have the ability or loses Internet connectivity then the application will not function correctly. IF the device has no Internet connection then a localised Google Map will not be displayed instead the default location will be displayed.

If the device loses connection at run time, then the GPS coordinates will be retrieved using the devices late know location. This will obviously cause problems due to inaccuracies, how inaccurate these coordinates are will depends on how long the device loosed connectivity for.

**Rebuilding and Testing:**

**Rebuilding:**

In order to rebuild the application you will the files contained in the src folder and the res folder, you will also need the manifest XML file. These directories (src and res) are located in the Walking Tours root directory, so is the manifest.

First create a new Android project in your IDE (e.g. Eclipse) and replace the created src/res directories with the corresponding directories specified above, do the same for the manifest file. Alternatively if you are using Eclipse IDE you can simply import the project saving both time and hassle.

Regardless of which method you used to import your application you need to import the Google play services library into you workspace otherwise you will not be able to access the API's. To do this it will depend on your IDE, but using Eclipse go to file → import → existing Android code into workspace. From this menu you will need to locate your Google place services library, this is found in your Android SDK install directory → extras → Google → google_play_services and click import. You now need to “point” your project to this library by right clicking on the project and selecting add “external library”, when prompted select the Google play services library you added to you workspace. The two projects should now be linked.

Also on every new SDK install you perform the above steps with you will need to add the add the SHA1 key for the install to the Google API Console: Google Maps for Android v2 list of accepted keys, if you do not do this you cannot access Google Maps correctly.

**Testing:**

There are several Test Classes already set up inside the tests package. In order to run these test simply right click on any test Class or the test package and click run as → Android JUnit Test.

These tests should now run, and upon result your IDE or terminal window will now display the results of these tests. If these tests pass, then your IDE will display a green symbol next to the test(s) that pass and a grey symbol if there is a failure. A stack trace is available based on the message you provided to the test to help see you see the reasons of this failure.

**Adding new tests:**

If you want to add new tests to existing Classes simply import the necessary libraries and write your JUnit tests as normal, abiding by JUnit version 3 standards. You can no re run your tests and see if they pass or fail using the steps specified above.

Make sure if you edit the one of the provided test Classes then make you you're test conforms with the current test content, make sure you are testing the write Class and make sure you fully document what the test does and why you are performing the test(s).

**Adding a new test Class:**

If you want to add a new Test Class you need to create a new Class as normal in the test package, and give it a meaningful name. Before you starting writing tests you need to make this Class inherit from the correct super Class, either ActivityUnitTestCase<ActivityName> or ActivityInstrumentationTestCase2<ActivityName> depending on your test intentions.

You also need to set up a Constructor for this test Class. This Constructor is very situation and its contents can vary depending on what you are testing. Refer to the existing test Classes and their constructor to check the syntax and what you expected to include inside this method, this is very important and cannot be missed.

Finally when writing a test you can set up a “setUp” method. This method I executed before the tests so in this method you declare and instantiate any variables or instances of Objects/Activities that you need during your tests. It is important to note that when you are writing tests you need to add the correct annotation and use the correct naming convention or the tests will not run. The annotations you will need are “@SmallTest” or “@BigTest” depending on your test case's method body. The naming convention is all you test case method name have to begin with the word test (JUnit version 3 requirement).

## Website Maintenance Manual

**Program Description:**

The Walking Tour Displayer allows the user to access and view walks that have been created using the Walking Tour Creator (android application). The user chooses a route, and this is the details regarding this route are then accessed by the server and displayed on the web page in the form of both a map and a table of information. The user can navigate the map and select different points, all linked together, each of which displays a position number, name, description, timestamp and image when clicked.

**Program Structure:**

Modules:

map.js:
Function load (): The variables map, infoWindow and line are created. walkXml.php is then accessed to retrieve

information regarding the points of interest from the database. A marker is created for each point using a for loop and these are linked together using an array of line which is continuously added to within the loop.
Function bindInfoWindow(marker, map, infoWindow, html): This is called within the for loop and assigns information to each individual point of interest, including the name, description, timestamp and image. Marker is the current point, map is the map in use, infoWindow is the window to be altered and html is the information to add.
Function downloadUrl(url, callback): This function retrieves the information from the walkXml.php file and is called just before the for loop in load().

walkXml.php:
Function parseToXml($htmlstr): This function takes a string created in html and translates it to one usable in XML.
This file connects to the database and translates data from several database tables in to an xml format, allowing for it to be utilised by the map. All of the values are retrieved from their respective tables and are then written in a XML format via concatenation.

**Main Data Areas:**

XML: All of the data that comes from the database is utilised in an XML format. Data is read in from the database tables and then assigned to the correct object, allowing for the data to be quickly and easily reference when the data is needed by the map.

**Files:**
.XML: An XML file is accessed in the map.js file in order to display the markers on the map. This is done by using the current url, with '&xml' tagged on to the end' as a parameter in the downloadUrl() function. The XML file holds the different objects that are present in the tables.
.JS: The map is created using javascript. Map.js is located in the assets folder.
.PHP: The majority of the files on the server side have the .php format and are generally smaller files that are utilised via index.php using this includes function.

**Suggestions for Improvement:**
Allow for multiple images: Currently, the xml only allows for each marker to have just one image displayed. Whilst this was a design flaw, the android application also lacked the capability of uploading multiple images, so this was not an immediate issue. This would allow for users to upload multiple different pictures for each, each meaning that the user has more control over what happens and are less restricted by technical limitations.

Add new options: Currently, the map is lacking a few options and pieces of information that could greatly help the user when using the walking tour displayer. Fields could include more complex ones, such as having the android application record distance between points, allowing for users to see how long a walk is (perhaps even calculating an optimal route between points), to simpler changes such as linking the closest points rather than the order that they were created.

**Things to watch when making changes:**

**Physical Limitations of the Program:**
Screen resolution: The map should be sized so that it can fit comfortably on many different screens. As much of it is graphical rather than textual, it may be good to implement a percentage size to the map rather than giving it a pixel count. This would allow all screens to view the same amount of the map, preventing issues such as only half the map being shown. This would be especially prevalent on smaller devices such as phones and tablets and could potentially create problems for people using those devices. Despite this, some problems may still arise in that the text on the information windows may be too small to read.

Browser: The user may have an out-of-date or unsupported browser for certain features of the site. This would mean that certain areas may display incorrectly or may even just not appear at all. As such, up-to-date browsers such as Chrome and Firefox are required/recommended to utilise the Walking Tour Displayer.

- **Synopsis**

The "Walking Tour Displayer" is a simple PHP script that processes incoming route information from one or more "Walking Tour Creators" and stores them in a persistent manner, as well as displaying walks to a user that navigates to the script.

- **Definitions used in this document**

To make explanations and descriptions easier to understand, the following otherwise ambigious proper nouns have special meanings within this document. Non-proper nouns may not have the same meaning.

The proper nouns with special meanings are:

**Customer**: the entity that commissioned this project.

**Client:** the Walking Tour Creator.

**User:** the person using either the Client or the Server.

**Server:** the Walking Tour Displayer.

**Route:** a single walk created by the User on the Client and uploaded to the Server.

- **Basic installation**

The Server was developed with the LAMP stack in mind, so ensure that Apache, MySQL and PHP are installed on the (Linux) server.

Clone the required files:

```
$ git clone git://github.com/cs22120/server.git
```

Navigate to the resultant page using a web browser of your choice.

You should be redirected to a wizard: fill in the form and click the button. The server will attempt to install the database for you, but there are many things that prevent it from happening so it is likely you will have to override it manually – copy the generated PHP configuration into config.php

```
$ vi config.php
```

Run the install wizard again to attempt to populate the tables; if this doesn't work you can either run *createDatabase()* via command line, or you can copy/paste the SQL statement from **includes/database.php**. The now/heredoc should allow you to copy without needing to escape any delimiters.

- **Doxygen**

The PHP files are commented in such a way that Doxygen, a better version of Javadoc, will automatically generate documentation for you regarding the functionality of various methods.

You can find out more about Doxygen at http://doxygen.org.

- **Structure**

The server is made up of multiple interacting files:

| File path | Purpose of file |
|---|---|
| ./.gitignore | Internal Git configuration file. |
| ./README.md | A Markdown file briefly describing the server. |
| ./install.php | A surplus-to-requirements installation wizard designed to make installation easier for inexperienced system administrators or developers. |
| ./upload.php | The entry point for the Client to upload to the Server. |
| ./assets/map.js | A JavaScript file responsible for client-side rendering of the map. |
| ./includes/credits.php | A vanity page listing contributors and Git commit |

| | |
|---|---|
| | information. |
| ./includes/database.php | Methods for the Server interact with the MySQL database. |
| ./includes/images.php | Methods for the Server to interact with images. |
| ./includes/templates.php | Governs rendering of files in the ./templates directory. |
| ./includes/walkDetail.php | File run when the User requests to view the detail of a Route. |
| ./includes/walkList.php | File run when the User requests the index page (and tacitly requests a list of Routes stored in the database). |
| ./includes/walkXml.php | File run when a user requests a machine-readable version of a Route detail (generally via JS). |
| ./templates | A directory containing things to output to a User. |
| ./uploads | The directory in which uploaded images should appear. |

- **Index page**

The index page **index.php** does not (under normal circumstances) output anything: it instead delegates to other files. As such, the main purpose of **index.php** is to work out what the user is requesting, and require such code as may be necessary.

When a User requests the home page (*http://example.com*), the file **index.php** is run. The Server tries to find a configuration file containing database connection information at **config.php**. If it cannot find any configuration, the User is redirected to an installation wizard at **install.php**. Otherwise, it checks the GET parameters of the request and will run **includes/walkXml.php**, **includes/credits.php**, **includes/walkDetail.php** or **includes/walkList.php** as appropriate.

The index page is a valid entry point. Users may request example.com/index.php freely.

- **Walk Index**

If **index.php** is requested without any pertinent GET variables, **includes/walkList.php** is required and included into the script. **walkList.php** attempts to select all Routes from the database and displays them in a table. The table is rendered through copious echo() calls, passed into a variable and rendered accordingly using the Homepage template.

The walk list page is not a valid entry point. Users requesting example.com/includes/walkList.php will receive an error message.

- **XML API**

If **index.php** is requested with the GET variable "xml" set to any value (i.e. *example.com?xml*), **includes/walkXml.php** is run.

**TODO: fill in**
This file is not a valid entry point, and users trying to access it in a web browser will receive an error.

- **Walk Detail**

If **index.php** is requested without the XML variable outlined above, but does contain a GET variable named "walk", **includes/walkDetail.php** will be required and included.

If the contents of the walk variable is not numeric, a HTTP 400 response will be emitted and the script will stop execution.

The script will then issue an SQL query to the database. If there was a connection error, an appropriate message is echoed to the user. No abnormal HTTP header is sent (i.e. it sends 200 and not the more correct 500).

If no walks were found matching the walk ID specified in the GET, a different message is echoed to the user. Again, 200 is sent instead of the possibly more appropriate 404 – this is something that could be developed further by succeeding developer teams.

If a walk was found, the script calculates the correct HTML body, stores it in a variable, and passes it to the skeleton template. This HTML body includes a header taken from the walk title, a paragraph from the longer walk description (the shorter one being reserved for the index page), a placeholder for the map, and a table with relevant waypoint data.

This file is not a valid entry point, and users trying to access it in a browser will receive an error.

- **Image file**

The file **includes/images.php** contains a sole function, **processImage()**. It takes exactly two parameters: the name of the image to store, and a base64 encoded string containing the image to save. It will return FALSE on success, and will otherwise return a number pertaining to an error.

| | |
|---|---|
| FALSE (0) | Image saved successfully. |
| 1 | File name already exists. The function won't clobber an existing image: try again with a different name. |
| 2 | The base64 encoding was bad. Try again with valid base64. |
| 3 | The file-put failed. The Server couldn't save the image for some unspecified reason. This is likely to be a permissions issue; check them and try again. |

This file is not a valid entry point, but does not have an entry point check. Users trying to access it in a browser will get a blank screen.

- **Templates file**

The file **includes/templates.php** contains a single helper function designed as syntactic sugar to make the code more useable. The function **render()** takes either one or two parameters: the first is required, and specifies the template to load. The second parameter is optional and specifies any data to send to the template.

All the function does is require the **templates/xxx.php** file, where **xxx** is the contents of the first parameter. It is a hackish solution that could easily be fixed in future revisions of the software.

The file is not an entry point and thus users will be displayed a message telling them such if they view the page in a web browser.

The template files in **templates/** are essentially normal PHP files. Their sole purpose is to attempt to separate the cosmetic content (such as including stylesheets) from the functional content (generating data) into different files. Each template generally includes the **templates/header.php** and corresponding **templates/footer.php** and uses **$data** to render dynamic content to the User.

- **Credits page**

The result of a very quick five minute hack, the credits page displays the list of contributors to the project, and attempts to fetch the current HEAD of the git repository the Server is running on. This allows multiple developers to use different copies of the repository yet be sure that they are running the latest code without resorting to more "complicated" methods such as **$ git log**.

The **includes/credits.php** file is three lines long, and merely runs **$ git log** on the User's behalf before outputting to **templates/credits.php** which returns a list of project contributors as well as copyright information for third-party software used in the project (*vis*. Twitter Bootstrap and the Google Maps API).

- **Database file**

The file **includes/database.php** contains the bulk of the database-related functions.

The function **createDatabase()** can be run via command-line to create the tables required for the database to function correctly. It takes no arguments and returns TRUE on success.

The **inputWalk()** function takes a single argument (the walk to input) and attempts to store it in the database. On success it will literally output the string "SUCCESS" and will otherwise return a string describing the error encountered. Maintainers should be aware that PHP will treat all of these possible return values as TRUE.

The **openConnection()** and **closeDatabase()** functions take zero arguments and initiates or closes a database connection using parameters set in **config.php**. The former will return the open database connection.

The **executeSql()** function abstracts away the open and closing of a connection. It takes one argument, a string containing the SQL to output. It will return a **mysqli_result()** object if data is returned (i.e. a SELECT, SHOW, DESCRIBE or EXPLAIN) and will return TRUE on other successful queries. Failures will output a string describing the error. Maintainers should be advised that PHP will treat all of these values as TRUE, so one should use functions such as **is_object()** and **is_boolean()** to discern between an error string and a successful result.

- **Upload file**

The file **upload.php** is the end point for the Client to upload to the server. It accepts JSON input as a POST request only.

The main purpose of this file is to validate the JSON sent by the Client to the Server and put it into a variable suitable for the **inputWalk()** function of **includes/database.php**.

The Server will always respond with an *application/json* formatted response (encoded as UTF-8). Successful requests will output the **success=>true** key value pair. Failure will result in a "false" value, as well as an error number and a human readable error message. The following responses are able to be returned (note that the reasons below are modified for a technical audience; simpler messages may be returned and you should use the ID to refer to them):

| No. | Reason for error | Possible remedy |
|---|---|---|
| 01 | *Reserved for future use.* | |
| **02** | Wrong method used. | Use POST, not GET (etc.) |
| 03 | No POST variables were sent. | Ensure that data is being sent in a POST variable named "data". |
| 04 | POST vars were sent, but not the right ones. | |
| 05 | JSON decoding failed. | Check the syntax and encoding of the JSON. |
| 06 | No "authorization" object was found. | The JSON should contain an **authorization** object containing a **hash** and a **salt**. |
| 07 | No SHA1 hash present. | Compute and send a SHA1 hash under "authorization". |
| 08 | No salt present. | Ensure that the salt is being sent with the hash. |
| 09 | The hash isn't a hash. | Make sure it is SHA1. There should be 40 characters. |
| 10 | The hash didn't match with the salt. | Check your hash computation. The hash should be computed by concatenating the salt with the pre-shared secret using the SHA1 algorithm. |
| 11 | No **walk** object was found. | The JSON should contain a **walk** object. Add one. |
| 12 | The **walk** object does not have a **title** key. | Add the relevant key/value pair to the **walk** object. |
| 13 | The **walk** object has no **shortDesc** key. | |
| 14 | The **walk** object has no **longDesc** key. | |
| 15 | The **walk** object has no **locations** key. | |
| 16 | One of the **locations** has no **latitude** key. | |
| 17 | One of the **locations** has no **longitude** key. | |
| 18 | One of the **locations** has no **timestamp** key. | |

| | | |
|---|---|---|
| 19 | One of the **locations** has no **descriptions** key. | |
| 20 | One of the **locations** has no **images** key. | |
| 21 | One of the **latitude**s specified doesn't work. | Latitudes must be between -90 and 90 degrees north of the Equator. |
| 22 | One of the **longitude**s specified is invalid. | Longitudes must be between 180 and -180 degrees east of Greenwich. |
| 23 | One of the **timestamp**s isn't numeric. | Ensure that the timestamp is seconds from 0000 UTC on 1 January 1970. |
| 24 | One or more **descriptions** is not an array. | Ensure that the relevant keys have an array (square brackets) associated with them. |
| 25 | One or more **images** is not an array. | |
| **26** | Server error. Could not save to the database. | Check the error string for details. |
| 27 | Uploading is disabled in **config.php**. | Remove **$DISABLEUPLOADS = true**. |
| 27a | Non-arrays are in the **images** array. | The **images** array should in turn also contain arrays. |
| 28 | Wrong number of entries in one of the **images** arrays. | The **images** array should contain arrays exactly two values wide: the filename and the base64 string. |
| 29 | File name collision. | Try again with a different file name for one of the images. |
| 30 | Base64 decoding failed. | Fix the encoding on one or more of the image strings. |
| **31** | Server error: couldn't save file. | This is a wide error that could have a multitude of causes, but is often permissions based. |
| 32 | Server error (catch all) | This is a major bug in the code itself. |
| 33 | One or more **descriptions** are not an array. | The **descriptions** values should be an array of arrays (with two values in the latter: name and description). |
| 34 | One of the **descriptions** have the wrong number of entries in their arrays. | |
| 35 | *Reserved for future use.* | |

Error responses will return with a HTTP 400 Bad Request, with the exception of those in the table above with a bolded ID number.

Once validation is complete, **inputWalk()** from **includes/database.php** is called, which processes the data and stores it into the database.

The "authorization" object is there for rudimentary security: ideally the Client would generate a new random salt for every transaction, 'crypt the salt with a preshared key, and send both the salt and the resultant hash to the server, verifying that the Client is a trusted party (but without the difficulty of setting up TLS/SSL).

This file is a valid entry point.

### 2.2.3 Personal reflection report

Dillon Cuffe

As a group I found that we really worked well together through most of the development of the application everyone knew their roles and knew what problems and issues that they were responsible for. People where able to work on documents and have the checked by others in the group, which I personally found important as it offered essential feedback on each piece of work. This ensured that the work that we handed in to be reviewed and marked was of the highest standard and all the people in the group had an input into the document.
As well as this I did find there were problems whilst the application was being developed. I found that within the group there was a clash of personalities, this clash I believe was due to the roles that where allocated to one of the individuals, they felt that the work that were given was not from them and when it was allocated to them they did not do it, is caused a minor issue. However the leader discussed it with the individual and they understood that they needed to contribute to the assignment even if it wasn't the work that they necessarily did not want to do.
However as the project developed and we as a group became familiar with one another I found that we really know each other's strong points in each part of the project and we knew who would be best suited for different roles in the project. As well as knowing the areas that each individual felt that the struggled the most in.
In addition to this I found that the main problem for me was that I was not always sure of what work I needed to do if any, and so I found that i felt that I was not given enough work, as I am not a very competent coder I felt that I couldn't really put much input into that side of the assignment and I felt that this is why I may have had a lower timesheet then some of the coders as I was mainly working on the documentation side of the project.
Another positive about working in this group for me was that there was good communication between the group, which allowed the work that was required to be delivered on time and of a high standard. I feel that we as a group collaborate well and we had a team that had a good mix of individual's with different specialities and skills that helped the project move forward and helped it develop.

I personally found that the project offered a chance for me to work together with other computer scientist and helped me have a chance to work with them. It helped me build my team work skills and how to work with a large group.

James Woodside

In my opinion I feel the group as a whole was worked incredibly well together. At the start of the project I put my name forward to lead the group in the project as I felt that I would be able to allocate the work to people in a fair manner and I wanted to make sure that we approached the task and make decisions in a diplomatic way.
At the start of the project there was some disagreement on the best approach to the how the program should look and trying to balance our interpretations of what we needed to be doing against what we felt was the best way to do it. The big initial debate within the group was the best way to design the GUI elements of the project. There were two similar designs but we couldn't decide on the best one which resulted in the group voting on the design which our final app has.
Fairly allocating work to people I found to be a lot more difficult than I first thought it would be. For example, I feel that the coders in particular ended up doing more of the documentation than they probably should have been.
The group as a whole though has made my job a lot easier than it could have been and I feel as a result I have not done as much work as I could have ended up doing. Bits of work that's needed to be done was usually picked up by one of the 2 main coders of the project which would then be given to Ben to make sure it was up to QA standards, and that would be split across the QA team if it was a particularly large piece of documentation. If people had problems with bits of work then I would do my best to offer any suggestions as to how they can do it, but in particular with the web side of things, it did end up being mostly down to one person.
My contribution to the coding of the project has not been very good. I do not do PHP or understand the web elements necessary to make meaningful contributions to that side. As Chris was comfortable with the coding of the Java app, this meant there was nothing else I could have helped with coding wise and so helped out with the testing of the final project.

Ben Rainbow

The first task of choosing who was to fulfil each role within the group came together well, I became Quality Assurance Manager which I was happy with. Subsequently other positions were then quickly filled. The group got off to a shaky start with some disagreements on how the finished product should look and how the user should be able to navigate through the different screens. We decided in the end to have a democratic approach and we took a vote deciding whose ideas we thought were best. My duty as QA manager during all meetings was to send out the minutes to allow people who did not attend to keep up to date with what was happening. Attendance was never really an issue, as the Christmas holidays came closer people were needing to change the time and place of the meeting as lesson plans and time to complete other assignments came in. We decided together where people could attend meetings and where we should meet for the most convenient place. The rooms were booked by Douglas who ensured we always had somewhere private to hold our meetings.

The group project has gone well for me. I feel as a group we have really pulled together, this shows through the success of the project as a whole. At times we all struggled with the tasks we had been set, but depending on current workload I felt the tasks were spread evenly. Running up to Christmas in the first semester other work had to be included to ensure that people could take on extra work when others could not then when the other people had high demands on their time this was then passed to the people who had less time on their hands. This meant that we could keep a high quality of work as the work completed was done by people who had a clear head and enough time to complete it to the best of their ability.

During the coding week after the Christmas holidays our group was punctual and again continued to work well together. Some personality clashes were apparent but under the pressure of the project in hand and the amount of time spent working together some clashes were expected. People realised there were differences in opinion but these were overcome to make sure the group succeeded. What really helped the group concentrate on the task on hand was the fact we were able to work in a quiet working environment in an obscure computer room which very few people used. This allowed us to discuss ideas openly without the worry of other groups hearing us, or distracting us from our trail of thought. I also felt that with the good management of the team all our deadlines were met without any need for people to do excessive hours in our last week. The QA team was able to work on the documentation and scrupulously go through the documents without needing to rush. Our testing team had enough time to carry out some extraneous tests on our software to make sure that it was resistant to any bugs which may normally be missed. This could help with the maintenance of the software as we believe we have a resilient product. The test walk carried out went really well also, as the software worked on the first walk although some additional features were added afterwards, but it gave us good signs that as a team we were getting near to the goal of having a complete product.

Overall I feel the technical expertise of our group members and the good communication of the group helped us pull through and work through our problems together. I was pleased to see us complete the project the day before hand in which really took the pressure off. This meant we could submit well before the deadline which allowed for any technical issues with blackboard. I also feel that the sub teams within our group have worked well together, my team especially have pulled together to help with the checking of grammar and spelling of documents produced. If another computer scientist were to look at the code the comments on each class and method fully explain what is happening. I also feel we kept to all of our deadlines and work was completed in a professional manner.

Jostein Kristiansen

My job during the group project was to assist the QA-team in creating QA documents, and to perform the various forms of testing on our final application.

I feel that I wrote fairly good documents, and that those documents were as detailed as they needed to be. Although I must admit that one of my documents, the testing specification, was missing a few tests, it also contained a few tests that were not mentioned by the “client's” acceptance testing document. So overall I think that document was satisfactory as well.

The only work I did that I'm not too proud of, are the Sequence diagrams, and the server-side component-diagram. I tried to make something that looked acceptable, but they did not look very good in the end, and I'm not sure if they portray the correct things.

Personally, I don't feel like I had any clashes with my team-members during this period. Although I suppose my constant silence and lack of participation in discussions could have annoyed someone at some point, that's just how I am.
I have paid attention to everything that has been said during our meetings, and I have always shown up to them; so I hope nobody mistook my silence for: obliviousness or disgust towards any of my team-members, because that is really not what it is.

Finally, I would like to say that I think my group did a very good job at producing a final product, and I really believe we have a great chance at winning the prize for best group-project.

Luke Horwood

On the whole, I feel this project has gone fairly well and has enhanced my skills in several areas greatly. In my opinion, we worked well as a team and managed to get all of the work done for our respective roles, allowing us to complete the project to a good standard. As everyone in the group stuck to their roles, each deadline was met successfully and the group was able to follow an ordered and structured approach in the development of our application(s) and the documentation to correlate with that.
The group project not only allowed me to enhance my skills in programming for the web, but also in various aspects of the software development life cycle and as part of a team. This has involved constant communication and collaboration in order to manage the creation of our documents and our code repositories.
We had regular (often bi-weekly, barring the timetabled slot) meetings where we discussed and went through where we were in regards to our work and where we had to be. This allowed us to persistently reflect upon how our work was progressing and ensured that everyone stayed on track and did not fall behind with their assigned pieces of work. During our meetings, we kept a record of what was happening via minutes, ensuring that everyone knew what they and everyone else was tasked to deal with.

There were a few minor bumps in some areas of the project, such as the creation of the sequence diagrams and some problems encountered during the programming of the application. On these occasions, we were able to work together to help each other complete our tasks, allowing for much more efficient and higher quality work than if the work was done independently.

Christopher Edwards

During the last semester of work up to and including integration and testing work I have learned and developed a wide variety of skills and I now understand how to work both efficiently and effectively with a team environment.

First of all there was a small problem with the delegation of tasks and it took a little while to finally decide the roles which each member of the team would take. I wouldn't say this hindered the group and our final outcome but I think it slightly set us off on the first document. All though we didn't have the best of starts to this project, we quickly made up for lost time and produced a document which everyone had some involvement in but more importantly we produced a collaborative document that satisfied the whole team. This is where I first started to really understand how important communication between the group is when collaboratively working. Overall I think the first document went very well, everybody understood what went into the document and who had to provide which bit of information.

During the delivery of the test plan document I would say the group under performed as a hole.,The feedback we received reflected this. The main reason I feel as though we underperformed slightly on the testing document is because at the time we had several other assignments on going I know I personally prioritized the other assignments over the testing document mainly because I wasn't part of the testing team. I am by no means saying this is the case for everybody, but the fact that there may have been external interferences during this time may have impacted the final product in relation the the draft of this document.

The design specification and prototype went quite well in my personal opinion. We went in the the mentality that for the prototype we might as well do as much as we can in order to reduce the amount of work left to do in the remaining time and in implementation and testing week. This put us in a good position to focus on what really needed doing during the implementation and testing week, which wasn't really a huge amount in all honesty.

When implementation and testing week came around just prior to semester 2 the whole group new exactly what we had left to do. We had managed to locate a room which was rather quiet but spacious so we all had our own space and communicated with other effectively both via email and verbally. The reason I think both me and the team performed well during this week is because everybody new exactly what they had to do. James split the group up into the sections and each section targeted specific documents/tasks in order to bring each sub-module more in-line with the standard and quality expected at the end. Overall I think the team performed very well during this week.

Apart from the two minor hiccups mentioned in the above I feel that every member of pulled their weight fairly. This being said I gained a lot more working hours earlier on in the project than everyone else because I was the main Android developer. This soon evened out across the project because I pretty much finished the Android application early into the integration and testing week whilst everyone else had a lot of work to do. So whilst I was making small changes here and there to documents the rest of the team brought their hours up massively so in the end everybody put in a very fair and balanced effort.

There is always a risk when taking part in a group project related task such as this that there will be disagreements within the group, this maybe due to personality classes or another influence such as different ideas/methods. In our group we didn't really encounter any problems such as these everyone generally got along with each other and accepted each others methods and ways of doing things as long as it produced a high quality result. The only problem of this type we encountered was the initial debate of Latex Vs. Word/Open Office and which we would use for this project. We decided on using word processors, which isn't ideal when using git to its full potential I understand but we wanted to keep things simple as the majority of the group (me included) were not confident in producing our best documents through Latex. This issue was soon resolved.

Overall I have very happy with the way I performed in this project. I completed every task that I was assigned and I think the final product is very professional. In addition I feel that I had a strong group and each member was willing to assist the others in order to collaboratively achieve a common goal.

Ashley Smith

For the group project I was assigned to work on quality assurance with Ben as the QA leader. I believe this role suited me well as I didn't feel confident with the coding side of the project and was unsure about the database side of the project.

For the project plan, we split up the document into sections and assigned certain members of the group the task of completing the document. I was assigned the introduction, purpose, scope and objectives. I felt I fulfilled the duties asked of me for this task. During the project plans development, we met on a regular bases, this allowed us to start thinking about the other deliverables earlier.

To ensure that the coding and database team were able to focus their attention on beginning the design specification, the QA and testing teams worked on the test specification. This meant that we could approach the design specification with the correct specialist skills it required. I assisted the completion of the test specification.

The design specification was started by the coding and database team, I worked closely with one of these members to produce the component diagram of this document and worked on some of the general QA. This helped me to develop my understanding of component diagrams and what they are used for.

Coding week required our full time involvement in the project. Our group spent the day working on each part of the project. We used this time efficiently to work on code for the project, test the application and also work on other minor parts of the project. For some of this week I worked on updating documentation. I did this, ensuring to take into account the feedback that was provided to us.

In conclusion I believe I contributed adequately to this project. I believe I provided the time and effort that was expected of me. I have developed my skills as a group member and have learnt to understand the roles within a group and the expectations upon each member. I have also developed my technical skills from this project as I have learnt the way in which android applications are developed and the processes involved. I believe that if I had the confidence to push myself into a much more technical role in the group, or assisted the technical team a little more I would have been able to develop my skills further.

## 2.2.4 Revised Project and Design Plan

### Project Plan

# 1. Introduction

The below content is an overview of the proposed route planning system developed for the use on mobile devices running the Android Operating system.

This document covers all the aspects relating to the high level overview and abstract data types required in order to be able to create the final product.

This document also includes an overview of how the project will be completed, the tasks involved and how long each task will take.

# 2. Purpose of this Document

The purpose of this document is to show how we have created a basic set of objectives based on the client's requirements for a walking tour creator application.
This document will also inform the client of the proposed system's high-level plans of the proposed system, giving detail on how the system will function.
The document will show the basic navigation and visual appearance of the applications user interface.
A Gantt chart will be provided, this will show the tasks involved during the process of the group project, including the milestones linked to each task. This will also provide a timeline on the project and give details on separate deadlines for specific tasks.
Along with this will be a risk analysis, which will show possible problems that may be occur during the development of the system, it will also show how these risks can be avoided.

## 2.1 Scope

This document will take into account the client's requirements for the group project.
This document will look at the User Interface design, timetabling and the risks involved with this project.
The UI design of this document will include an overview of the system, detailing the high-level architecture, choice of platform and the target users of the proposed system. Use Case diagrams will be included to show interactions of user's with the proposed system. Designs of the user interface will be included to give an idea of how the application will look and give details on navigation.
To timetable this project a Gantt chart will be included to show the tasks involved in the systems development and the milestones of the development.
Finally a risk analysis will be included to give detail on potential problems we could face during the systems development, and how we may avoid or solve these problems.
This document will not include detailed design of any of the classes within the system neither will it look at any of the systems possible system or unit tests.

## 2.2 Objectives

The main objectives of this document are to:

10. Provide an overview of the system, giving detail of the target audience and the technologies that will be used
11. Specify the high-level architecture and platforms that will be used within the proposed system
12. Show the appearance and behaviour of the system from a user's perspective
13. Show the main interactions of the user's and the system
14. Show the basic navigation of the system and give an idea of how the finished product will look
15. Give a time line of all tasks that need to be completed
16. Define what tasks need to be completed
17. Identify any problems that may arise from the production of this system.
18. Provide information on how the problems discovered can be avoided or solved

# 3. Overview

## 3.1 Technologies

The following technologies (and platform) shall be used in this system:

### 3.1.1 Client

#### *3.1.2 Android*

The client's requirements specifically stated that the application shall be designed for Android phones. The target Android platform will be Android version 4.2 (Jelly Bean (1)) but will be backwards-compatible with Android 2.3 (Gingerbread (2)).

Android is a free operating system for mobile phones (3), in which the applications are written in Java (4). Many members of the development team are literate at Java programming, making Android a platform conducive to creating high-quality applications, such as the one sought by the client.

Android is the leading operating system for mobile phones, with almost 80% market share in the second quarter of 2013, with 187.4 million shipments (5). This makes it appealing over iOS, the operating system used by Apple's iPhones. Another compelling reason to use Android over iOS is that the development of Android applications is free: iOS licensing requires a minimum of a USD $99 *per annum* fee (6).

### 3.1.3 Server

#### *3.1.3.1 PHP*

PHP shall be used on the server to handle communication between the Android device and the database, as well as being used to create the website end-users access to view past walks.

PHP is widely used within server development, with usage on approximately 2.1 million devices (7), and is taught to second-year students (8), giving us an opportunity to gain skills through its application.

PHP has the benefit of being a pre-processor, meaning that PHP does not need its own service. It is also easy to learn, and easy to compose for, allowing for rapid development.

This will be hosted on a group member's personal IS account.

#### *3.1.3.2 MySQL*

MySQL will be used for our database to store all the user information from the application.

This will be hosted via db.dcs.aber.ac.uk. This will be requested from CS-Support for our group.

MySQL is a time-proven application used widely. It is well supported by the other server technologies we will use (in this case, PHP) (9), and is easy to administrate with graphical tools such as phpMyAdmin (10).

#### *3.1.3.3 Apache*

The Apache HTTP server shall be used in conjunction with PHP to create our web application. It is the most popular HTTP server in use today, with over 50% market penetration (11).

#### *3.1.3.4 Linux*

The "LAMP" stack is a common server application bundle (12), and will be used to create our server application. As such, we will be developing our server with a GNU/Linux operating system in mind.

## 3.2 Architecture

The high-level architecture will consist of the following elements:

### 3.2.1 Client

This describes the Android application. This will be shown in the screens described below in the user interface designs.

### 3.2.2 Map

The map will be displayed, server side, through the use of the Google Maps API, this allows us to have scrolling and zooming readily available for use. The map will show user's the route of the walk, and also allow us to show points of interest.

### 3.2.3 Photos

Users will be able to add Photos to points of interest within the application. GPS coordinates will tie a photo to a specific location.

### 3.2.4 Internet Connectivity

Users will require an internet connection to be able to upload a saved walk to the server. If the user fails to connect or connection is lost then they will be informed with a notification, they will not be able to upload to the server until they regain connection.

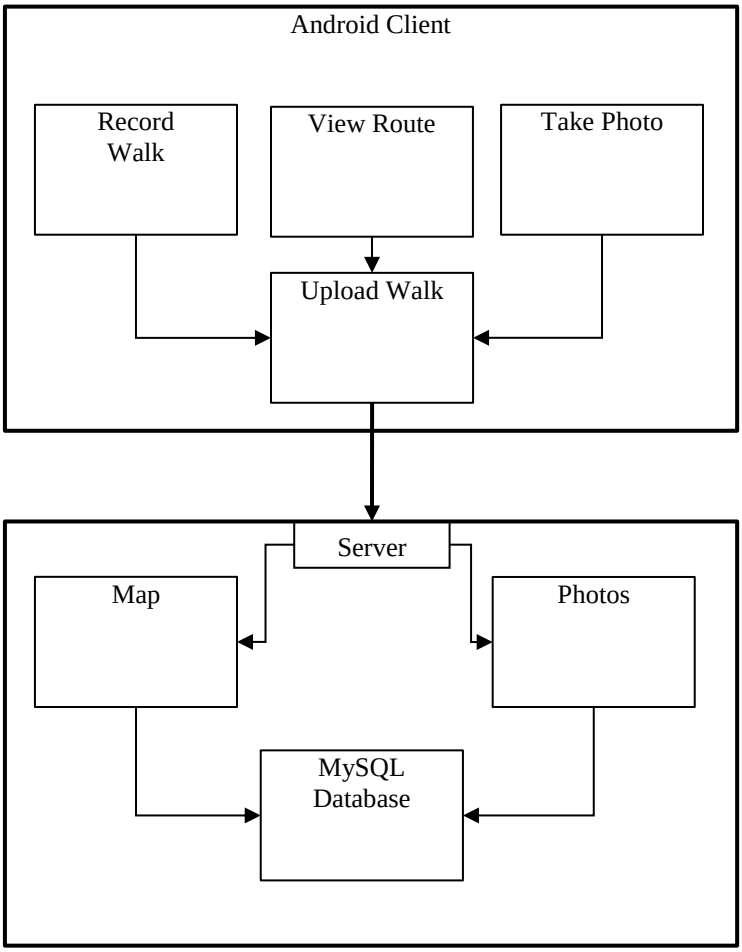



**Figure 1: Architecture Diagram**

## 3.3 Target Users

As specified in the requirements specification, the software is to be used by second year computer science students. The design of this system has taken into account the users knowledge of computer systems. We have tried to make the system as intuitive as we can, with the fewest user actions required.

# 4. USE CASE DIAGRAM

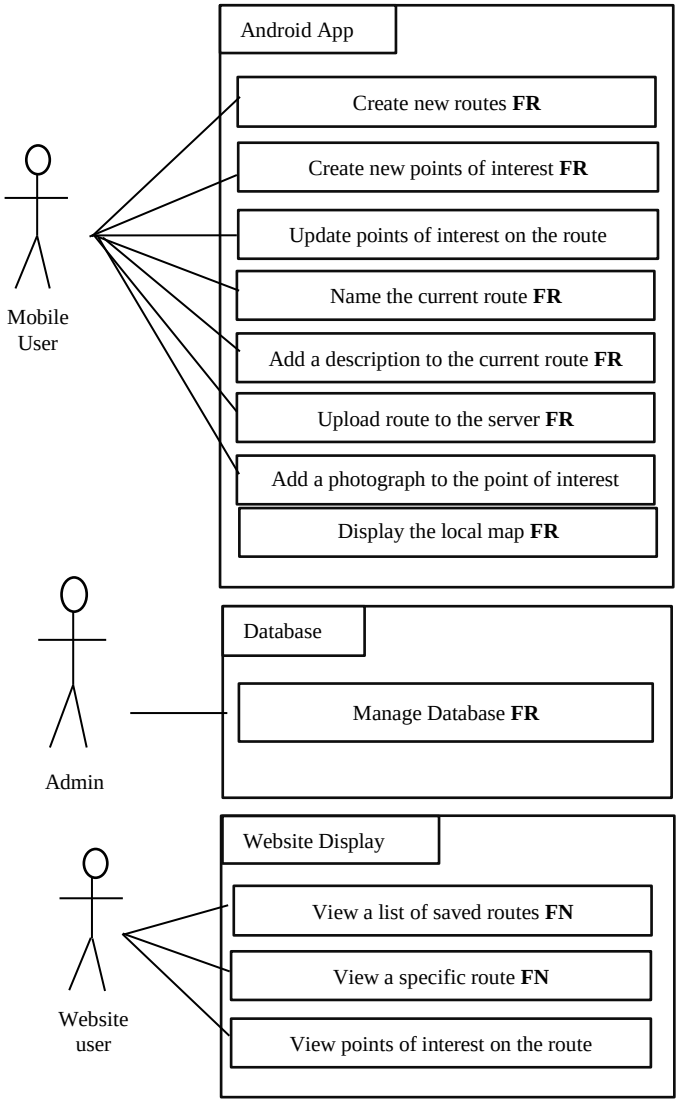



**Figure 2: Use Case Diagram**

***FR notation specifies that the task is a functional requirement and is needed in order to produce the specified output from the application.***

| **Web User** | |
|---|---|
| View a list of all stored routes | The website will display all routes currently listed in the database to the user. |
| View the information for a specific list | The website will allow the user to select a specific list. The specific information for the selected list will be returned to the user. |
| View the points of interest for a specified route | For any selected route the user will be able to view specific points of interest for said route. |

| **Database Administrator** | |
|---|---|
| Manage the database system | The administrator must be able to log into the database administrator facility to manage the database and collected data. |

| **Mobile user** | |
|---|---|
| Create new routes | The user of the Android application will be able to create new routes which will record their GPS coordinates and any desired points of interest. |
| Create points of interest on a route | The user of the application will be able to add points of interest with their current GPS coordinates and add it to the route. |
| Update points of interest on a route | The user can modify and or update the points of interest on the route. |
| Create a name for the route | A name can be specified for the route which can be used to identify the route at a later time period. |
| Add a description to the route | A description can be added to the route which will be displayed on the website. |
| Add a photograph to the point of interest | A photograph can be retrieved from the devices camera, which can then be applied to the point of interest as a visual aid. |
| Upload the completed route to the server | Once the application user is happy with the route they can upload it to the server which will handle the request and store the information in the database. |

# 5. User Interface Design

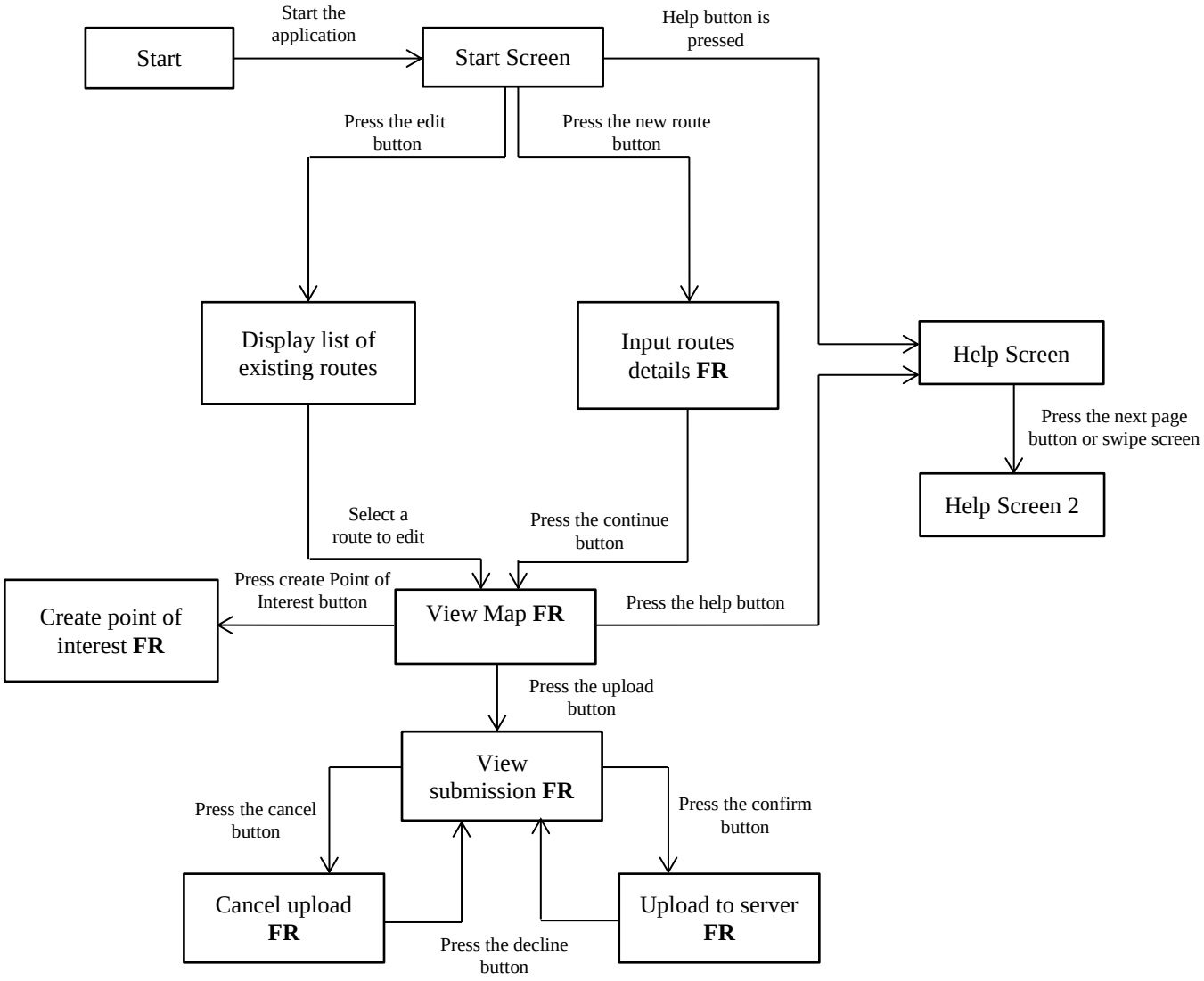



**Figure 3:Flow Chart of the screen navigation**

## 5.1 GUI design

Below are the designs for each element in the Graphical User Interface. These are simple representations of the proposed design for the Android application. These are designed to give a rough idea of the finished applications look, but are by no means finalized designs.

## 5.2 Start Screen

This is the screen that will greet users when they load the app. It has the name of the app and the logo, along with two buttons. The first button, 'New route' allows the user to start mapping a new walk and takes the user to the “Create new route” screen. The second button allows the user to modify existing routes and takes the user to the “Display all existing routes” interface.

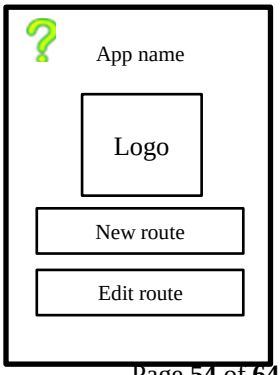



**Figure 4: Diagram showing the Start Screen**

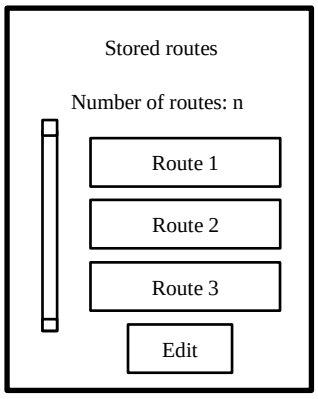



**Figure 5: The Display All Routes screen**

## 5.3 Displaying all existing routes Screen

This screen contains a list of all available routes that the user can modify. The user can scroll through the existing routes and select one they wish to make changes to by pressing the "Edit" button. The number of existing routes is placed above the list of routes.

## 5.4 Create new route screen

This screen allows the user to enter the details for their walk and has three text fields: "Route name", "Short description" and "Walk details". After inputting the required details, the user can then press the "Begin tracking" button to proceed to the map screen.

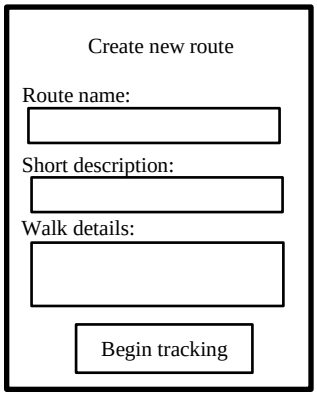



**Figure 6: Create New Route screen**

## 5.5 View map screen

This is the screen where users will create their walks. It has a button allowing the user to create a new point of interest, which takes them to the "Create new point of interest" screen. It also has "Help" and "Upload" buttons. The help button takes the user to the help interface where they can troubleshoot problems that they may be encountering. The upload button takes the user to the "Review submission" screen before sending the finished walk to the server and adding it to the list of editable walks. There are also two counters for the length (time taken) of the walk and the number of locations featured within the walk. A map is placed within the centre of the screen, which displays the points of interest that have already been placed on the map.

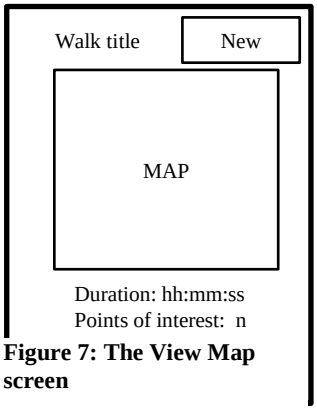



**Figure 7: The View Map screen**

## 5.6 Create new point of interest screen

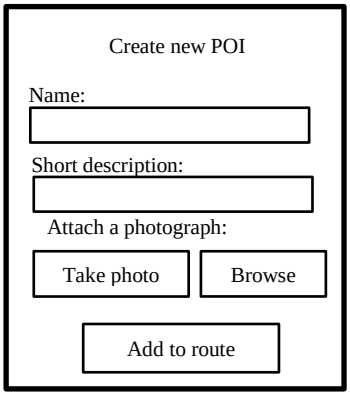



**Figure 8: Create New PoI screen**

The POI creation screen allows the user to designate locations that should be featured in the walk. There are two fields: "Name", where the user can input a name for the current location, and "Short description", where the user can write a short summary of the featured location. The user can also take pictures of the current point of interest using their phones camera by pressing the "Take photo" button and then attach the photo to the walk from the phone's memory. The "Add to route" button then adds the location to the map and takes the user back to the view map screen.

## 5.7 Review submission screen

The review submission screen allows the user to check that all details of the route are correct in order to stamp out any errors. The name, short description and walk details are displayed and there are two text fields allowing the user to modify the description and the details of the walk. Below the text fields, there is a scrollable box containing a list of all the points of interest within the walk. At the bottom of the screen, there are two buttons: cancel and confirm. Cancel takes the user back to the cancel upload screen and the confirm button takes the user to the upload confirmation screen.

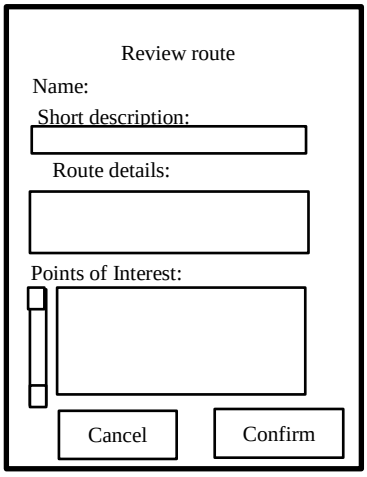



**Figure 9: The Review Route screen**

## 5.8 Cancel upload screen

This screen displays the details of the walk and at the bottom asks for confirmation if the user wants to cancel the walk. There are two buttons at the bottom: confirm and decline. If they press decline, the user is taken back to the previous screen (review submission). If they press confirm, the walk is deleted.

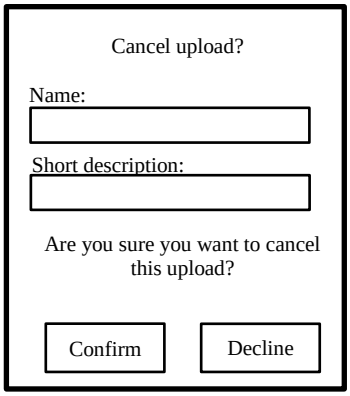



**Figure 10: The Cancel Upload screen**

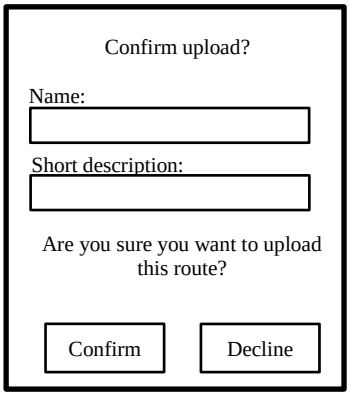



**Figure 11: The Upload Confirmation screen**

## 5.9 Upload confirmation screen

This screen displays the details of the walk and at the bottom asks for confirmation if the user wants to upload the walk to the server. There are two buttons at the bottom: confirm and decline. If they press confirm the walk is sent to the server and added to the list of walks available to edit. If they press decline, they are taken back to the review submission screen.

## 5.10 Help Screen

The help screens will all be created using a consistent theme and layout. Each help screen will display information on a specific topic (e.g. how to upload a route) and the user can navigate through these help pages using the "previous" and "next" buttons to either progress or back track through the pages. The screens themselves will display a basic text description of what the page is dedicated to (e.g. "How to upload :") as well as providing a more detailed body of text regarding the contents of the help method itself.

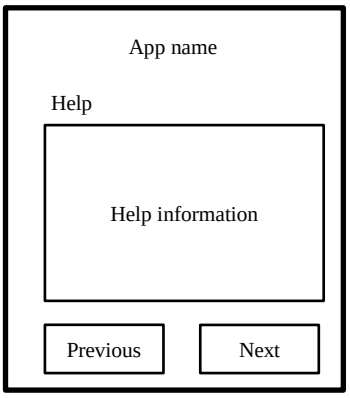
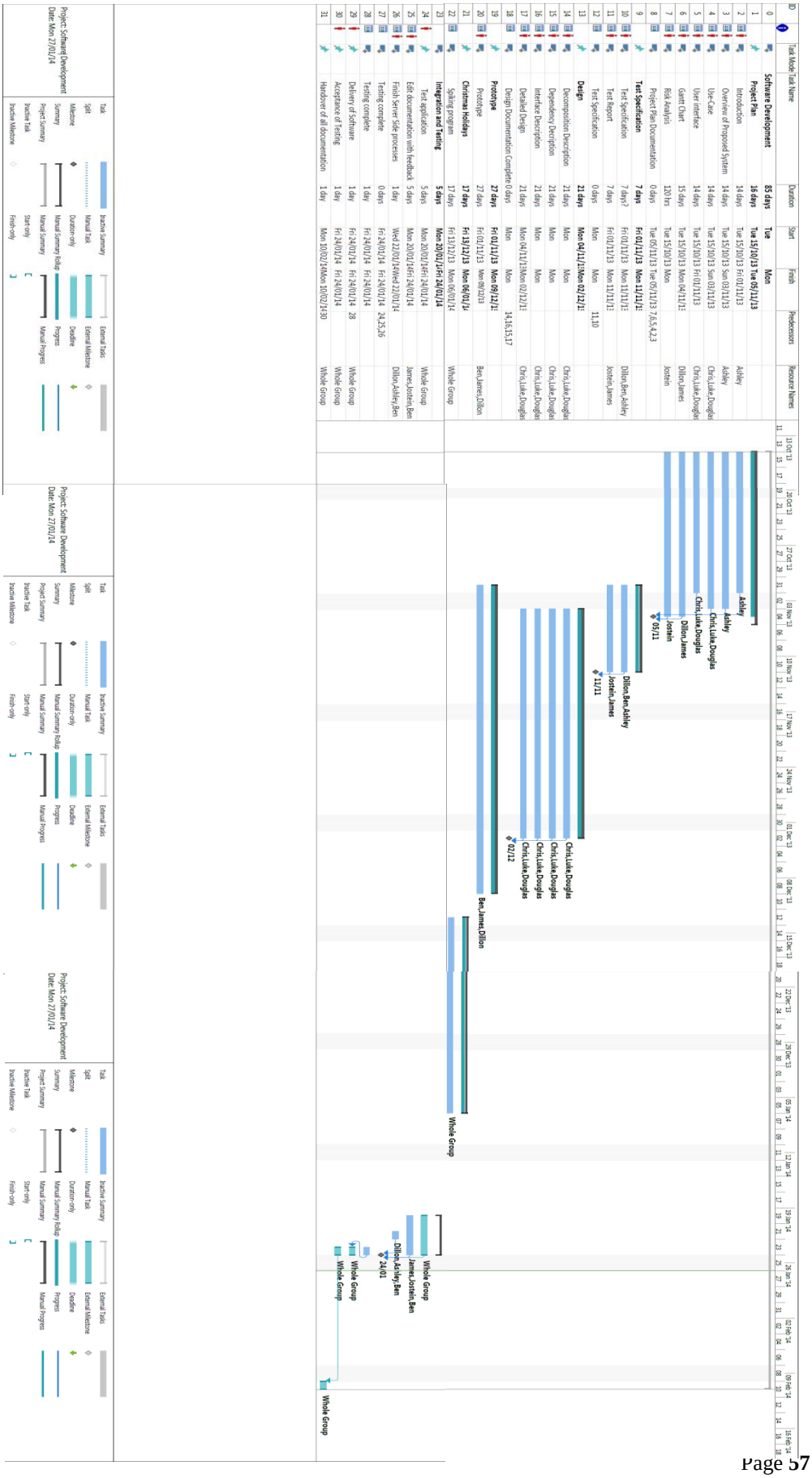



**Figure 11: The Help screen(s)**

# 6. Gantt chart



# 7. Risk Analysis

## 7.1 Constant Risks

| Task | Hazard | Risk level | How to deal with it |
|---|---|---|---|
| Scheduled meeting | Team member absent | low | Absent team member shall read the minutes of the missed meeting. Continued absence shall result in action being taken against the missing team member. |
| Scheduled meeting | Project leader absent | low | Meeting shall continue as normal, chaired by deputy leader. |
| Scheduled meeting | QA Manager absent | low | Meeting shall continue as normal; QA questions and decisions shall be handled by others in the QA team. |
| Scheduled meeting | Programmer absent | low | Meeting shall continue as normal; other programmers will raise any queries related to the program, on behalf of the missing programmer. |
| Informing the team whenever you complete a goal, and estimate how long it will take you to complete the next. | Lose team member contact | moderate | It is very important that every team member keeps updating the team on their progress via the group e-mail. If someone neglects their duty to do this, parts of documentation may become missing, and major setbacks in progress towards the next milestone may occur. |
| Handing work to the group | Local files lost | high | All work should be uploaded to GitHub. Team members shall ensure that copies of all work shall be backed up; either on the University files store or an external storage device. It is the sole responsibility of the team member to do so. |
| Handing over your work to the group | Illness or any other circumstance keeping you from completing your work | moderate / high | If you ever get the feeling that you are unable to complete any/all parts of your assigned work in time for hand-in; Inform the rest of the group immediately. We all rely on each other to make sure our own work is completed within the planned time; if you are falling behind, you should inform the others that you are struggling, and we will help you complete your work any way we can. |

## 7.2 Risks related to documentation

| Task | Hazard | Risk level | How to deal with it |
|---|---|---|---|
| Handing over your documentation to the group | Late submission by some team member(s) | low | Internal deadlines shall be set prior to the final, external deadlines, ensuring that there is a buffer period allowing team members to assist if issues arise. |
| Handing over your documentation to the group | Parts of documentation missing or incomplete | moderate | All documents will be handed over to the QA manager for approval. Documents shall be easy to read, and in need of minimal editing by the QA team. |
| Handing over your documentation via GitHub | Git | low | We might not be able to edit any .docx files directly on git-hub, but we should be able to download your documentation and edit it on our own machines, then re-upload the edited version to Git-hub. |

## 7.3 Risks related to development and delivery

| Task | Hazard | Risk level | How to deal with it |
|---|---|---|---|
| Keeping up with the project timeline (Gantt chart) | Not being able to complete all tasks by the time planned on the Gantt chart | moderate | The programming team shall update the QA manager and project leader on their progress, as often as possible; allowing the swift resolution of any problems that arise. |
| Following the project specification | Some parts of the project are missing, incomplete, or not what the specification asks for | moderate | All points in the project specification shall be implemented, with final responsibility for this lying with the project leader, who shall ensure that the project follows the specification.<br><br>Extensive testing of code shall take place to ensure that everything works as planned.<br><br>The QA manager shall ensure that all documentation, code, and other work are both high-quality and comprehensive. |
| Handing in the final version of the project | Downtime on, Blackboard / University File store / Git-hub / Aberystwyth University internet connection | moderate | The team shall ensure that each stage of the project is ready to be submitted at least one day before the deadline. This will ensure that disruption from any downtime on Blackboard, the University file store, GitHub, or the University's Internet connection.<br><br>Any downtime in these places is unlikely to last for longer than a day; so the project can be immediately submitted once services resume. |
| Storing parts of the project on an external storage device that you carry around with you | You might lose your storage device, it could be stolen, or damaged by water | moderate | All Files should be stored on Git-hub therefore External storage shall not be carried around without reason, and additional backups should be made on a regular basis (e.g., daily at 6 p.m.) |

## 7.4 Risks related to the usage of the program

| Task | Hazard | Risk level | How to deal with it |
|---|---|---|---|
| Loading the map | No connection to the internet, or too slow to load the entire map | low | If connection to the Internet is checked before the user gets the chance to start a new walk, one can make sure that no grey areas appear on the map.<br><br>Alternatively, if a version of the map of Ceredigion is available offline, the team shall only have to worry about the GPS reading coordinates correctly.<br><br>If the coordinates are correct, one should be able to switch from the offline-version of the map to the online when Internet connection is resumed.<br><br>(This should require the app to check for a new version of the offline map regularly, for maintainability purposes) |
| Displaying a list of existing routes | The map has changed after the route was made | low | If the map is different from when the route was made, it is possible that the resultant route will end up showing a route that cross over train tracks and through residential houses.<br><br>A possible response is to store the map along with a route. However, this may result in too much data being stored locally. |
| Using the UI | The application is run without a touch screen. | low | No certified Android phone lacks a touch screen (13). The Android developers recommend that one assumes that all Android phones have a touch-screen, making this risk a very low priority. |
| Using the help-screen(s) | The help-screen is too descriptive, or does not cover all problems | low | The QA manager shall take oversight of all help-screens. Usability tests shall take place to ensure that the help screens are helpful. |
| Creating a new point of interest | Help screen is not available | moderate | The developers shall ensure that the help screen is available.<br><br>If a set of help-screens are only displayed in this portion of the program, the other set of help-screens will feel less large and easier to navigate. |
| Viewing the map | The map is currently loading, and only displaying a grey area | low | The user must be aware of any delays in loading the map.<br><br>A small animation or textbox shall be added to inform the user that the map is loading; thereby avoiding any confusion as to why parts of the map are missing. |
| Cancelling / confirming the submission | The confirm – decline buttons have switched places from where they were on the previous screen | low | If one presses buttons without reading what they say, having the “cancel” / “confirm” buttons in different positions in the “Review submission screen”, “Cancel upload screen”, and “Upload confirmation screen”, could result in someone unintentionally deleting their walk.<br><br>Designs will ensure that the ‘default’ option is always to the right of the ‘cancel’ button, or equivalent. |
| Editing the walking tour | Should we have a separate “edit” | low | If it is made very clear that the user is editing the submission, by bringing up a new window when |

| | | | |
|---|---|---|---|
| before submitting it | button, to prevent someone from accidentally changing some information? | | you click on any of the information in the submission, accidental deletion of information can be avoided.<br><br>This is a small problem, and probably not a huge deal as you can edit the walking-tour after submission as well, although it might look better. |
| Cancelling any other parts of the program | As far as I can tell, there is no way to go back to the previous screen in: "Display existing routes", "Create new route", "create new point of interest", or the "Help" screens. | moderate | If one cannot cancel "create new route", then one would have to first create a route, and then cancel the route, and finally going back to the start screen.<br><br>If one cannot cancel "create new point of interest", then one would be forced to create a new POI, and then delete it again when you are submitting the walking-tour.<br><br>If there is no obvious way to exit the "Help-screen", some users might get stuck in there, and be forced to restart the entire program. |
| Removing a Point of interest | There is no way to remove a POI in the "View map screen" | low | Users should be able to remove any points of interest that they do not want from the map.<br><br>However, being able to remove them whilst reviewing the submission may not be enough; for example, placing POI in a slightly incorrect location, and want to relocate it.<br><br>(An example of this might be inaccurate GPS-coordinates.) |

# 8. References

1. Android 4.2 APIs. *Android Developers.* [Online] [Cited: 5 November 2013.] http://developer.Android.com/about/versions/Android-4.2.html.
2. Gingerbread. *Android Developers.* [Online] [Cited: 5 November 2013.] http://developer.Android.com/about/versions/Android-2.3-highlights.html.
3. **Alliance, Open Handset.** Android Overview. [Online] [Cited: 30 October 2013.] http://www.openhandsetalliance.com/Android_overview.html.
4. Application Fundamentals. *Android Developers.* [Online] [Cited: 30 October 2013.] http://developer.Android.com/guide/.
5. **Etherington, Darrell.** Android Nears 80% Market Share In Global Smartphone Shipments, As iOS And BlackBerry Share Slides, Per IDC. *TechCrunch.* [Online] 7 August 2013. [Cited: 4 November 2013.] http://techcrunch.com/2013/08/07/Android-nears-80-market-share-in-global-smartphone-shipments-as-ios-and-blackberry-share-slides-per-idc/.
6. **Apple, Inc.** Choosing an iOS Developer Program. *Apple Developer.* [Online] [Cited: 30 October 2013.] https://developer.apple.com/programs/start/ios.
7. **The PHP Group.** Usage Stats for January 2013. *php.net.* [Online] January 2013. [Cited: 4 November 2013.] http://php.net/usage.php.
8. **Aberystwyth University.** Module Information: CS25010. [Online] [Cited: 4 November 2013.] http://www.aber.ac.uk/en/modules/deptcurrent/?m=CS25010.
9. **The PHP Group.** Technology Overview. *PHP Manual.* [Online] 1 November 2013. [Cited: 4 November 2013.] http://us1.php.net/manual/en/mysqlinfo.terminology.php.
10. About. *phpMyAdmin.* [Online] [Cited: 4 November 2013.] http://www.phpmyadmin.net/home_page/index.php.
11. June 2013 Web Server Survey. *Netcraft.* [Online] June 2013. [Cited: 4 November 2013.] http://news.netcraft.com/archives/2013/06/06/june-2013-web-server-survey-3.html.
12. What is a LAMP stack? *Stack Overflow.* [Online] 8 April 2012. [Cited: 4 November 2013.] http://stackoverflow.com/questions/10060285/what-is-a-lamp-stack.
13. Compatibility Program Overview. *Android Source.* [Online] [Cited: 4 November 2013.] http://source.Android.com/compatibility/overview.html.
14. **C.J. Price, B.P.Tiddeman.** *Walking Tour Creator Requirements Specification.* [PDF Document] Aberystwyth : Aberystwyth University, 2013.
15. **C.J.Price, N.W.Hardy, B.P.Tiddeman.** *Design Specification Standards.* [PDF Document] Aberystwyth : Aberystwyth University, 29 September 2013.

# DOCUMENT HISTORY

| *Version* | *CCF No.* | *Date* | *Changes made to document* | *Changed by* |
|---|---|---|---|---|
| 0.1 | N/A | 2013-11-04 | Severely copyedit and collation | dog2 |
| 0.2 | N/A | 2013-11-04 | Added introduction and help screen info | che16 |
| 0.3 | N/A | 2013-11-05 | Collate and copy editing | bar5 |
| 0.4 | N/A | 2014-01-27 | Changes made in conjunction with feedback | ays8 |
| 0.5 | N/A | 2014-01-27 | Minor changes to introduction and use cases | ays8 |
| 0.6 | N/A | 2014-01-28 | Spell check and format alterations | bar5 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

References

[1] SE.QA.11-**Producing a Final Report**

[2] Software Engineering Group Projects: General Documentation Standards. C. J. Price, N. W. Hardy. SE.QA.03. 1.5 Release

Document History

| *Version* | *CCF No.* | *Date* | *Changes made to document* | *Changes made by* |
|---|---|---|---|---|
| 1.1 | N/A | 2014-01-27 | First version of the document | dic6 |
| *1.2* | N/A | *2014-02-17* | *QA and grammatical and spelling corrections proof reading and structure formatting.* | *bar5* |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# Software Engineering Group Project
# Project Plan


| | |
|---|---|
| Author: | Christopher Edwards;Douglas Gardner;Luke Horwood;Jostein Kristiansen;Ben Rainbow;Ashley Smith;James Woodside;Dillon Cuffe |


| | |
|---|---|
| Config Ref: | SE_02_PL_01 |
| Date: | 2014-01-28 |
| Version: | 0.6 |
| Status: | Released v1.1 |



Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

# CONTENTS

# TABLE OF FIGURES

# 1. INTRODUCTION

The below content is an overview of the proposed route planning system developed for the use on mobile devices running the Android Operating system.

This document covers all the aspects relating to the high level overview and abstract data types required in order to be able to create the final product.

This document also includes an overview of how the project will be completed, the tasks involved and how long each task will take.

# 2. PURPOSE OF THIS DOCUMENT

The purpose of this document is to show how we have created a basic set of objectives based on the client's requirements for a walking tour creator application.
This document will also inform the client of the proposed system's high-level plans of the proposed system, giving detail on how the system will function.
The document will show the basic navigation and visual appearance of the applications user interface.
A Gantt chart will be provided, this will show the tasks involved during the process of the group project, including the milestones linked to each task. This will also provide a timeline on the project and give details on separate deadlines for specific tasks.
Along with this will be a risk analysis, which will show possible problems that may be occur during the development of the system, it will also show how these risks can be avoided.

## 2.1 Scope

This document will take into account the client's requirements for the group project.
This document will look at the User Interface design, timetabling and the risks involved with this project.
The UI design of this document will include an overview of the system, detailing the high-level architecture, choice of platform and the target users of the proposed system. Use Case diagrams will be included to show interactions of user's with the proposed system. Designs of the user interface will be included to give an idea of how the application will look and give details on navigation.
To timetable this project a Gantt chart will be included to show the tasks involved in the systems development and the milestones of the development.
Finally a risk analysis will be included to give detail on potential problems we could face during the systems development, and how we may avoid or solve these problems.
This document will not include detailed design of any of the classes within the system neither will it look at any of the systems possible system or unit tests.

## 2.2 Objectives

The main objectives of this document are to:

1. Provide an overview of the system, giving detail of the target audience and the technologies that will be used
2. Specify the high-level architecture and platforms that will be used within the proposed system
3. Show the appearance and behaviour of the system from a user's perspective
4. Show the main interactions of the user's and the system
5. Show the basic navigation of the system and give an idea of how the finished product will look
6. Give a time line of all tasks that need to be completed
7. Define what tasks need to be completed
8. Identify any problems that may arise from the production of this system.
9. Provide information on how the problems discovered can be avoided or solved

# 3. OVERVIEW

## 3.1 Technologies

The following technologies (and platform) shall be used in this system:

### 3.1.1 Client

### *3.1.2 Android*

The client's requirements specifically stated that the application shall be designed for Android phones. The target Android platform will be Android version 4.2 (Jelly Bean (1)) but will be backwards-compatible with Android 2.3 (Gingerbread (2)).

Android is a free operating system for mobile phones (3), in which the applications are written in Java (4). Many members of the development team are literate at Java programming, making Android a platform conducive to creating high-quality applications, such as the one sought by the client.

Android is the leading operating system for mobile phones, with almost 80% market share in the second quarter of 2013, with 187.4 million shipments (5). This makes it appealing over iOS, the operating system used by Apple's iPhones. Another compelling reason to use Android over iOS is that the development of Android applications is free: iOS licensing requires a minimum of a USD $99 *per annum* fee (6).

### 3.1.3 Server

#### *3.1.3.1 PHP*

PHP shall be used on the server to handle communication between the Android device and the database, as well as being used to create the website end-users access to view past walks.

PHP is widely used within server development, with usage on approximately 2.1 million devices (7), and is taught to second-year students (8), giving us an opportunity to gain skills through its application.

PHP has the benefit of being a pre-processor, meaning that PHP does not need its own service. It is also easy to learn, and easy to compose for, allowing for rapid development.

This will be hosted on a group members personal IS account.

#### *3.1.3.2 MySQL*

MySQL will be used for our database to store all the user information from the application.

This will be hosted via db.dcs.aber.ac.uk. This will be requested from CS-Support for our group.

MySQL is a time-proven application used widely. It is well supported by the other server technologies we will use (in this case, PHP) (9), and is easy to administrate with graphical tools such as phpMyAdmin (10).

#### *3.1.3.3 Apache*

The Apache HTTP server shall be used in conjunction with PHP to create our web application. It is the most popular HTTP server in use today, with over 50% market penetration (11).

#### *3.1.3.4 Linux*

The "LAMP" stack is a common server application bundle (12), and will be used to create our server application. As such, we will be developing our server with a GNU/Linux operating system in mind.

## 3.2 Architecture

The high-level architecture will consist of the following elements:

### 3.2.1 Client

This describes the Android application. This will be shown in the screens described below in the user interface designs.

### 3.2.2 Map

The map will be displayed, server side, through the use of the Google Maps API, this allows us to have scrolling and zooming readily available for use. The map will show user's the route of the walk, and also allow us to show points of interest.

### 3.2.3 Photos

Users will be able to add Photo's to points of interest within the application. GPS coordinates will tie a photo to a specific location.

### 3.2.4 Internet Connectivity

Users will require an internet connection to be able to upload a saved walk to the server. If the user fails to connect or connection is lost then they will be informed with a notification, they will not be able to upload to the server until they regain connection.

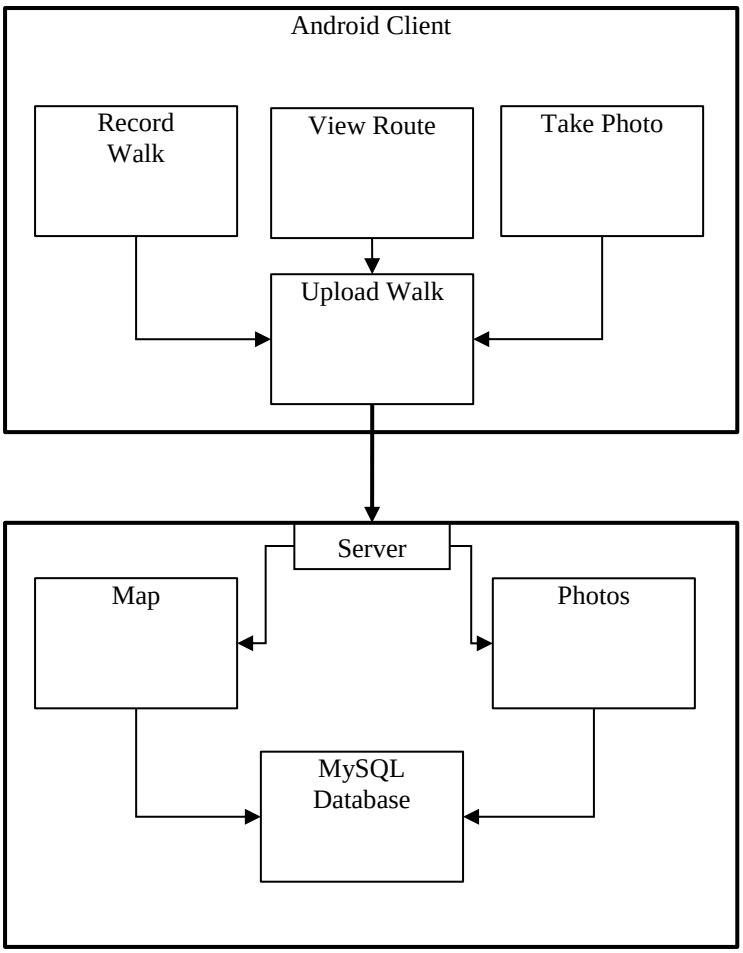



**Figure 1: Architecture Diagram**

## 3.3 Target Users

As specified in the requirements specification, the software is to be used by second year computer science students. The design of this system has taken into account the users knowledge of computer systems. We have tried to make the system as intuitive as we can, with the fewest user actions required.

# 4. USE CASE DIAGRAM

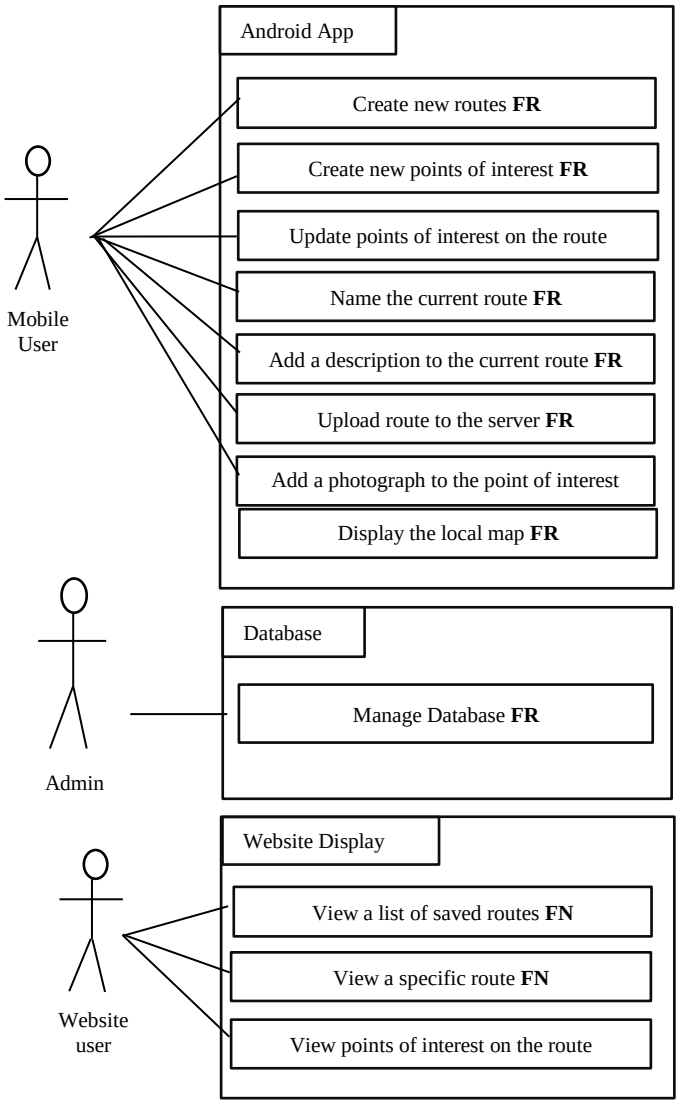



**Figure 2: Use Case Diagram**

***FR notation specifies that the task is a functional requirement and is needed in order to produce the specified output from the application.***

| **Web User** | |
|---|---|
| View a list of all stored routes | The website will display all routes currently listed in the database to the user. |
| View the information for a specific list | The website will allow the user to select a specific list.<br>The specific information for the selected list will be returned to the user. |
| View the points of interest for a specified route | For any selected route the user will be able to view specific points of interest for said route. |

| **Database Administrator** | |
|---|---|
| Manage the database system | The administrator must be able to log into the database administrator facility to manage the database and collected data. |

| **Mobile user** | |
|---|---|
| Create new routes | The user of the Android application will be able to create new routes which will record their GPS coordinates and any desired points of interest. |
| Create points of interest on a route | The user of the application will be able to add points of interest with their current GPS coordinates and add it to the route. |
| Update points of interest on a route | The user can modify and or update the points of interest on the route. |
| Create a name for the route | A name can be specified for the route which can be used to identify the route at a later time period. |
| Add a description to the route | A description can be added to the route which will be displayed on the website. |
| Add a photograph to the point of interest | A photograph can be retrieved from the devices camera, which can then be applied to the point of interest as a visual aid. |
| Upload the completed route to the server | Once the application user is happy with the route they can upload it to the server which will handle the request and store the information in the database. |

# 5. USER INTERFACE DESIGN

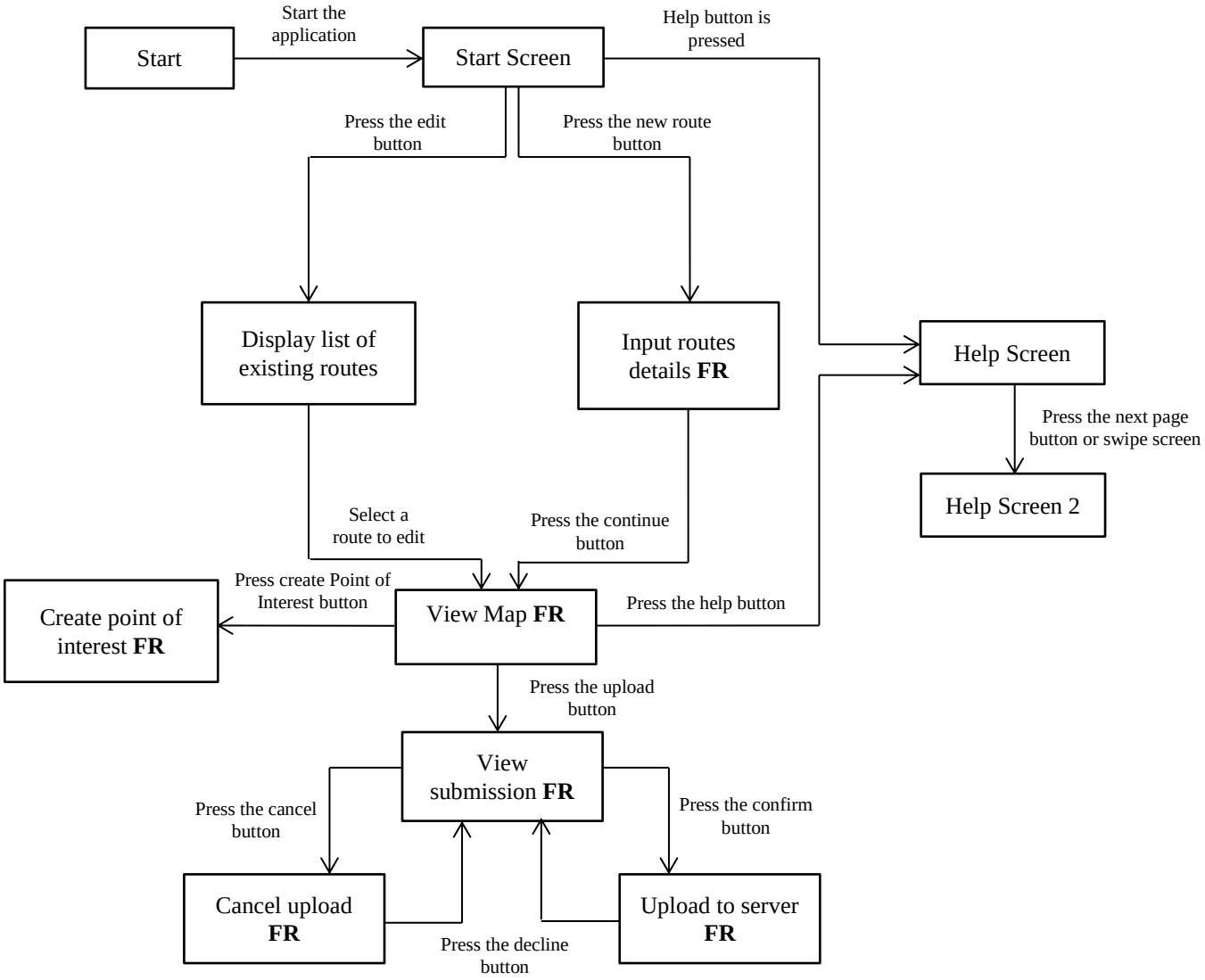



**Figure 3:Flow Chart of the screen navigation**

## 5.1 GUI design

Below are the designs for each element in the Graphical User Interface. These are simple representations of the proposed design for the Android application. These are designed to give a rough idea of the finished applications look, but are by no means finalized designs.

## 5.2 Start Screen

This is the screen that will greet users when they load the app. It has the name of the app and the logo, along with two buttons. The first button, 'New route' allows the user to start mapping a new walk and takes the user to the "Create new route" screen. The second button allows the user to modify existing routes and takes the user to the "Display all existing routes" interface.

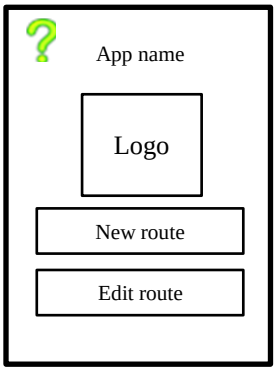



**Figure 4: Diagram showing the Start Screen**

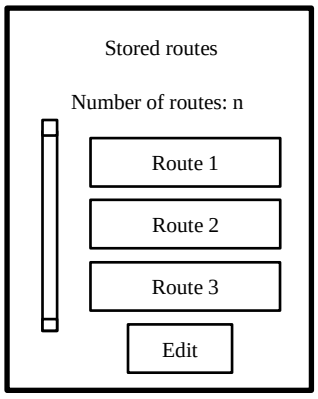



**Figure 5: The Display All Routes screen**

## 5.3 Displaying all existing routes Screen

This screen contains a list of all available routes that the user can modify. The user can scroll through the existing routes and select one they wish to make changes to by pressing the "Edit" button. The number of existing routes is placed above the list of routes.

## 5.4 Create new route screen

This screen allows the user to enter the details for their walk and has three text fields: "Route name", "Short description" and "Walk details". After inputting the required details, the user can then press the "Begin tracking" button to proceed to the map screen.

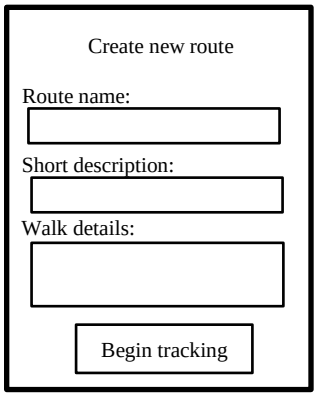



**Figure 6: Create New Route screen**

## 5.5 View map screen

This is the screen where users will create their walks. It has a button allowing the user to create a new point of interest, which takes them to the "Create new point of interest" screen. It also has "Help" and "Upload" buttons. The help button takes the user to the help interface where they can troubleshoot problems that they may be encountering. The upload button takes the user to the "Review submission" screen before sending the finished walk to the server and adding it to the list of editable walks. There are also two counters for the length (time taken) of the walk and the number of locations featured within the walk. A map is placed within the centre of the screen, which displays the points of interest that have already been placed on the map.

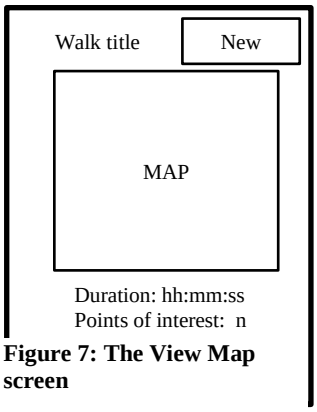



**Figure 7: The View Map screen**

## 5.6 Create new point of interest screen

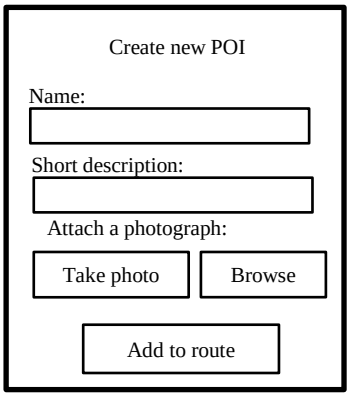



**Figure 8: Create New PoI screen**

The POI creation screen allows the user to designate locations that should be featured in the walk. There are two fields: "Name", where the user can input a name for the current location, and "Short description", where the user can write a short summary of the featured location. The user can also take pictures of the current point of interest using their phones camera by pressing the "Take photo" button and then attach the photo to the walk from the phone's memory. The "Add to route" button then adds the location to the map and takes the user back to the view map screen.

## 5.7 Review submission screen

The review submission screen allows the user to check that all details of the route are correct in order to stamp out any errors. The name, short description and walk details are displayed and there are two text fields allowing the user to modify the description and the details of the walk. Below the text fields, there is a scrollable box containing a list of all the points of interest within the walk. At the bottom of the screen, there are two buttons: cancel and confirm. Cancel takes the user back to the cancel upload screen and the confirm button takes the user to the upload confirmation screen.

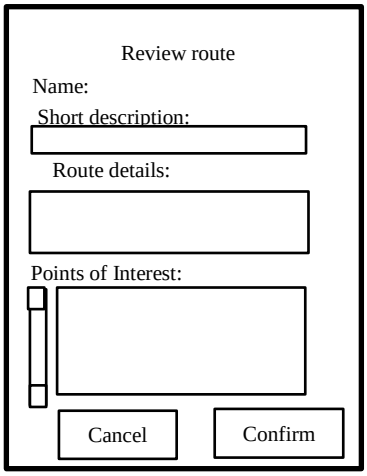



**Figure 9: The Review Route screen**

## 5.8 Cancel upload screen

This screen displays the details of the walk and at the bottom asks for confirmation if the user wants to cancel the walk. There are two buttons at the bottom: confirm and decline. If they press decline, the user is taken back to the previous screen (review submission). If they press confirm, the walk is deleted.

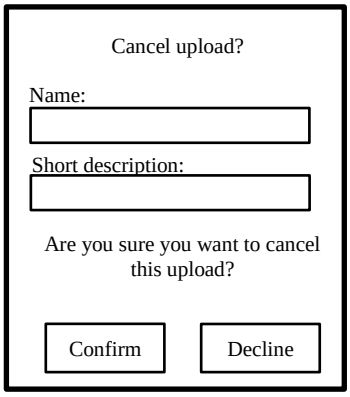



**Figure 10: The Cancel Upload screen**

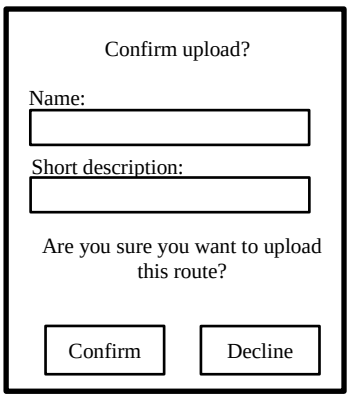



**Figure 11: The Upload Confirmation screen**

## 5.9 Upload confirmation screen

This screen displays the details of the walk and at the bottom asks for confirmation if the user wants to upload the walk to the server. There are two buttons at the bottom: confirm and decline. If they press confirm the walk is sent to the server and added to the list of walks available to edit. If they press decline, they are taken back to the review submission screen.

## 5.10 Help Screen

The helps screens will all be created using a consistent theme and layout. Each help screen will display information on a specific topic (e.g. how to upload a route) and the user can navigate through these help pages using the "previous" and "next" buttons to either progress or back track through the pages. The screens themselves will display a basic text description of what the page is dedicated to (e.g. "How to upload :") as well as providing a more detailed body of text regarding the contents of the help method itself.

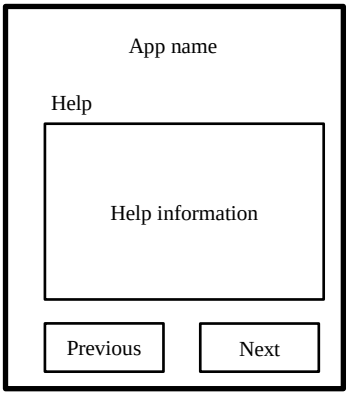
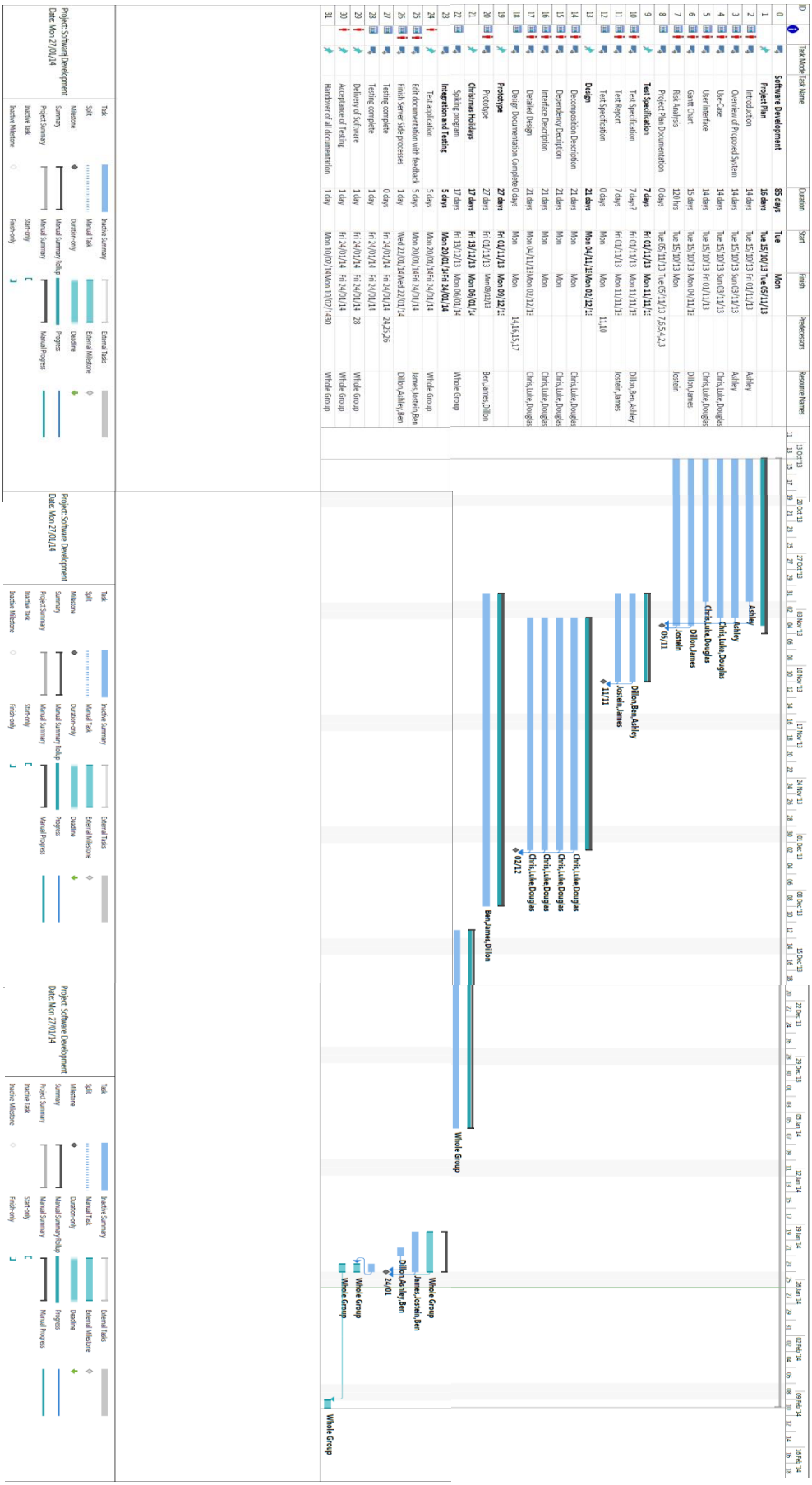



**Figure 11: The Help screen(s)**

# 6. GANTT CHART



# 7. RISK ANALYSIS

## 7.1 Constant Risks

| Task | Hazard | Risk level | How to deal with it |
|---|---|---|---|
| Scheduled meeting | Team member absent | low | Absent team member shall read the minutes of the missed meeting.<br>Continued absence shall result in action being taken against the missing team member. |
| Scheduled meeting | Project leader absent | low | Meeting shall continue as normal, chaired by deputy leader. |
| Scheduled meeting | QA Manager absent | low | Meeting shall continue as normal; QA questions and decisions shall be handled by others in the QA team. |
| Scheduled meeting | Programmer absent | low | Meeting shall continue as normal; other programmers will raise any queries related to the program, on behalf of the missing programmer. |
| Informing the team whenever you complete a goal, and estimate how long it will take you to complete the next. | Lose team member contact | moderate | It is very important that every team member keeps updating the team on their progress via the group e-mail. If someone neglects their duty to do this, parts of documentation may become missing, and major setbacks in progress towards the next milestone may occur. |
| Handing work to the group | Local files lost | high | All work should be uploaded to git-hub.<br>Team members shall ensure that copies of all work shall be backed up; either on the University files store or an external storage device. It is the sole responsibility of the team member to do so. |
| Handing over your work to the group | Illness or any other circumstance keeping you from completing your work | moderate / high | If you ever get the feeling that you are unable to complete any/all parts of your assigned work in time for hand-in; Inform the rest of the group immediately.<br>We all rely on each other to make sure our own work is completed within the planned time; if you are falling behind, you should inform the others that you are struggling, and we will help you complete your work any way we can. |

## 7.2 Risks related to documentation

| Task | Hazard | Risk level | How to deal with it |
|---|---|---|---|
| Handing over your documentation to the group | Late submission by some team member(s) | low | Internal deadlines shall be set prior to the final, external deadlines, ensuring that there is a buffer period allowing team members to assist if issues arise. |
| Handing over your documentation to the group | Parts of documentation missing or incomplete | moderate | All documents will be handed over to the QA manager for approval. Documents shall be easy to read, and in need of minimal editing by the QA team. |
| Handing over your documentation via GitHub | Git | low | We might not be able to edit any .docx files directly on git-hub, but we should be able to download your documentation and edit it on our own machines, then re-upload the edited version to Git-hub. |

### 7.3 Risks related to development and delivery

| Task | Hazard | Risk level | How to deal with it |
|---|---|---|---|
| Keeping up with the project timeline (Gantt chart) | Not being able to complete all tasks by the time planned on the Gantt chart | moderate | The programming team shall update the QA manager and project leader on their progress, as often as possible; allowing the swift resolution of any problems that arise. |
| Following the project specification | Some parts of the project are missing, incomplete, or not what the specification asks for | moderate | All points in the project specification shall be implemented, with final responsibility for this lying with the project leader, who shall ensure that the project follows the specification.<br><br>Extensive testing of code shall take place to ensure that everything works as planned.<br><br>The QA manager shall ensure that all documentation, code, and other work are both high-quality and comprehensive. |
| Handing in the final version of the project | Downtime on, Blackboard / University File store / Git-hub / Aberystwyth University internet connection | moderate | The team shall ensure that each stage of the project is ready to be submitted at least one day before the deadline. This will ensure that disruption from any downtime on Blackboard, the University file store, GitHub, or the University's Internet connection.<br><br>Any downtime in these places is unlikely to last for longer than a day; so the project can be immediately submitted once services resume. |
| Storing parts of the project on an external storage device that you carry around with you | You might lose your storage device, it could be stolen, or damaged by water | moderate | All Files should be stored on Git-hub therefore External storage shall not be carried around without reason, and additional backups should be made on a regular basis (e.g., daily at 6 p.m.) |

### 7.4 Risks related to the usage of the program

| Task | Hazard | Risk level | How to deal with it |
|---|---|---|---|
| Loading the map | No connection to the internet, or too slow to load the entire map | low | If connection to the Internet is checked before the user gets the chance to start a new walk, one can make sure that no grey areas appear on the map.<br><br>Alternatively, if a version of the map of Ceredigion is available offline, the team shall only have to worry about the GPS reading coordinates correctly.<br><br>If the coordinates are correct, one should be able to switch from the offline-version of the map to the online when Internet connection is resumed.<br><br>(This should require the app to check for a new version of the offline map regularly, for maintainability purposes) |
| Displaying a list of existing routes | The map has changed after the route was made | low | If the map is different from when the route was made, it is possible that the resultant route will end up showing a route that cross over train tracks and through residential houses.<br><br>A possible response is to store the map along with a route. However, this may result in too much data being stored locally. |
| Using the UI | The application is run without a touch screen. | low | No certified Android phone lacks a touch screen (13). The Android developers recommend that one assumes that all Android phones have a touch-screen, making this risk a very low priority. |
| Using the help-screen(s) | The help-screen is too descriptive, or does not cover all problems | low | The QA manager shall take oversight of all help-screens. Usability tests shall take place to ensure that the help screens are helpful. |
| Creating a new point of interest | Help screen is not available | moderate | The developers shall ensure that the help screen is available.<br><br>If a set of help-screens are only displayed in this portion of the program, the other set of help-screens will feel less large and easier to navigate. |
| Viewing the map | The map is currently loading, and only displaying a grey area | low | The user must be aware of any delays in loading the map.<br><br>A small animation or textbox shall be added to inform the user that the map is loading; thereby avoiding any confusion as to why parts of the map are missing. |
| Cancelling / confirming the submission | The confirm – decline buttons have switched places from where they were on the previous screen | low | If one presses buttons without reading what they say, having the “cancel” / “confirm” buttons in different positions in the “Review submission screen”, “Cancel upload screen”, and “Upload confirmation screen”, could result in someone unintentionally deleting their walk.<br><br>Designs will ensure that the ‘default’ option is always to the right of the ‘cancel’ button, or equivalent. |
| Editing the walking tour | Should we have a separate “edit” | low | If it is made very clear that the user is editing the submission, by bringing up a new window when |

| | | | |
|---|---|---|---|
| before submitting it | button, to prevent someone from accidentally changing some information? | | you click on any of the information in the submission, accidental deletion of information can be avoided.<br><br>This is a small problem, and probably not a huge deal as you can edit the walking-tour after submission as well, although it might look better. |
| Cancelling any other parts of the program | As far as I can tell, there is no way to go back to the previous screen in: "Display existing routes", "Create new route", "create new point of interest", or the "Help" screens. | moderate | If one cannot cancel "create new route", then one would have to first create a route, and then cancel the route, and finally going back to the start screen.<br><br>If one cannot cancel "create new point of interest", then one would be forced to create a new POI, and then delete it again when you are submitting the walking-tour.<br><br>If there is no obvious way to exit the "Help-screen", some users might get stuck in there, and be forced to restart the entire program. |
| Removing a Point of interest | There is no way to remove a POI in the "View map screen" | low | Users should be able to remove any points of interest that they do not want from the map.<br><br>However, being able to remove them whilst reviewing the submission may not be enough; for example, placing POI in a slightly incorrect location, and want to relocate it.<br><br>(An example of this might be inaccurate GPS-coordinates.) |

# 8. REFERENCES

1. Android 4.2 APIs. *Android Developers.* [Online] [Cited: 5 November 2013.] http://developer.android.com/about/versions/android-4.2.html.
2. Gingerbread. *Android Developers.* [Online] [Cited: 5 November 2013.] http://developer.android.com/about/versions/android-2.3-highlights.html.
3. **Alliance, Open Handset.** Android Overview. [Online] [Cited: 30 October 2013.] http://www.openhandsetalliance.com/android_overview.html.
4. Application Fundamentals. *Android Developers.* [Online] [Cited: 30 October 2013.] http://developer.android.com/guide/.
5. **Etherington, Darrell.** Android Nears 80% Market Share In Global Smartphone Shipments, As iOS And BlackBerry Share Slides, Per IDC. *TechCrunch.* [Online] 7 August 2013. [Cited: 4 November 2013.] http://techcrunch.com/2013/08/07/android-nears-80-market-share-in-global-smartphone-shipments-as-ios-and-blackberry-share-slides-per-idc/.
6. **Apple, Inc.** Choosing an iOS Developer Program. *Apple Developer.* [Online] [Cited: 30 October 2013.] https://developer.apple.com/programs/start/ios.
7. **The PHP Group.** Usage Stats for January 2013. *php.net.* [Online] January 2013. [Cited: 4 November 2013.] http://php.net/usage.php.
8. **Aberystwyth University.** Module Information: CS25010. [Online] [Cited: 4 November 2013.] http://www.aber.ac.uk/en/modules/deptcurrent/?m=CS25010.
9. **The PHP Group.** Technology Overview. *PHP Manual.* [Online] 1 November 2013. [Cited: 4 November 2013.] http://us1.php.net/manual/en/mysqlinfo.terminology.php.
10. About. *phpMyAdmin.* [Online] [Cited: 4 November 2013.] http://www.phpmyadmin.net/home_page/index.php.
11. June 2013 Web Server Survey. *Netcraft.* [Online] June 2013. [Cited: 4 November 2013.] http://news.netcraft.com/archives/2013/06/06/june-2013-web-server-survey-3.html.
12. What is a LAMP stack? *Stack Overflow.* [Online] 8 April 2012. [Cited: 4 November 2013.] http://stackoverflow.com/questions/10060285/what-is-a-lamp-stack.
13. Compatibility Program Overview. *Android Source.* [Online] [Cited: 4 November 2013.] http://source.android.com/compatibility/overview.html.
14. **C.J. Price, B.P.Tiddeman.** *Walking Tour Creator Requirements Specification.* [PDF Document] Aberystwyth : Aberystwyth University, 2013.
15. **C.J.Price, N.W.Hardy, B.P.Tiddeman.** *Design Specification Standards.* [PDF Document] Aberystwyth : Aberystwyth University, 29 September 2013.

# DOCUMENT HISTORY

| *Version* | *CCF No.* | *Date* | *Changes made to document* | *Changed by* |
|---|---|---|---|---|
| 0.1 | N/A | 2013-11-04 | Severely copyedit and collation | dog2 |
| 0.2 | N/A | 2013-11-04 | Added introduction and help screen info | che16 |
| 0.3 | N/A | 2013-11-05 | Collate and copy editing | bar5 |
| 0.4 | N/A | 2014-01-27 | Changes made in conjunction with feedback | ays8 |
| 0.5 | N/A | 2014-01-27 | Minor changes to introduction and use cases | ays8 |
| 0.6 | N/A | 2014-01-28 | Spell check and format alterations | bar5 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# Software Engineering Group Project
# Design Specification


| | |
|---|---|
| Author: | Christopher Edwards;Douglas Gardner;Luke Horwood;Jostein Kristiansen;Ben Rainbow;Ashley Smith;James Woodside;Dillon Cuffe |
| Config Ref: | SE_02_DS_001 |
| Date: | 2014-01-29 |
| Version: | 0.4 |
| Status: | Released v1.1 |

*Department of Computer Science*
*Aberystwyth University*
*Aberystwyth*
*Ceredigion*
*SY23 3DB*

# CONTENTS

# 1. INTRODUCTION

## 1.1 Purpose of this Document

The purpose of this document is to show the whole design specification aspect of the Android Application that is going to be created. Within this document there will be outlines for the design of the significant classes and detailed mapping of the requirements on the classes. As well as the defining dependency descriptions, manufacturing the interface description and having a detailed design for the entire system.

## 1.2 Scope

The design specification shows all the different components that will need to be implemented and will describe how components like the interface will work. In doing this it will allow for the group to access information and work in conjunction with the Requirements Specification.
All information on this document will need to be read and reviewed by all members of the group.

## 1.3 Objectives

The objectives of this document are to outline and define clearly the entire layout and design of the Walking Tour Android Application. The requirements of this document are set by the customer's standards and this document will take into consideration those points and address them by creating an application. This will clearly address and implement those requirements of the customer.
Below will be the complete design of the application in conjunction to these requirements, examples of this will be found in the form of how the software is structured, components of the software and how the interface works and reacts to user input.

# 2. DECOMPOSITION DESCRIPTION

## 2.1 Programs in system

The Walking Tours application will consist of two main components. The two applications are the Android application and the web processing application. The web application is dependent on the Android application, and in some sense the Android application depends on parts of the web application.

#### The Android application:

The Android application will be the main application in this project. The Android application will consist of several activities and classes, which will be used in conjunction with user interactions and Google API's to record a user's walk.

This application revolves around the process of initially creating a walk with a given name and descriptions. This information is then stored within the application (in an easily accessible location) and the users current GPS location is retrieved. This GPS location can then be used to provide the user with a localised map of the surrounding area, which can then be used to display the users movements (requires the ability to constantly record GPS location) and also allow for the addition of flagging up "Points of Interest" along the current route.

These "Points of Interest" will also prompt the user for a name, description and the option of taking a photo/selecting an image to be associated with said "Point of Interest". These "Points of Interest" will also be displayed on the current map and their location on this map will be determined by the GPS location where the

user creates the point of interest. At this point the coordinates of the user's location will be assigned to the "Point of Interest" which allows for placement upon the map.

Once the user is satisfied with the current walks situation (e.g. all "Points of Interest" added and walk finished) then they can upload the results to the server. The user will be prompted to make sure they do in fact want to upload the walk and unwanted clicks can happen accidentally on touch screen devices.

Uploading to the server will require a device (running Android) with network connectivity of some variation in order to access the server, the information will be sent to the server as a HTTP POST request. This is where the application is reliant on the web application, if the server is unavailable then the application cannot meet the functional requirements (FR 6).

This part of the application will run on any Android enabled device running Android version 2.3+ (Gingerbread onwards) and a minimum API level of 10.

**The web application:**

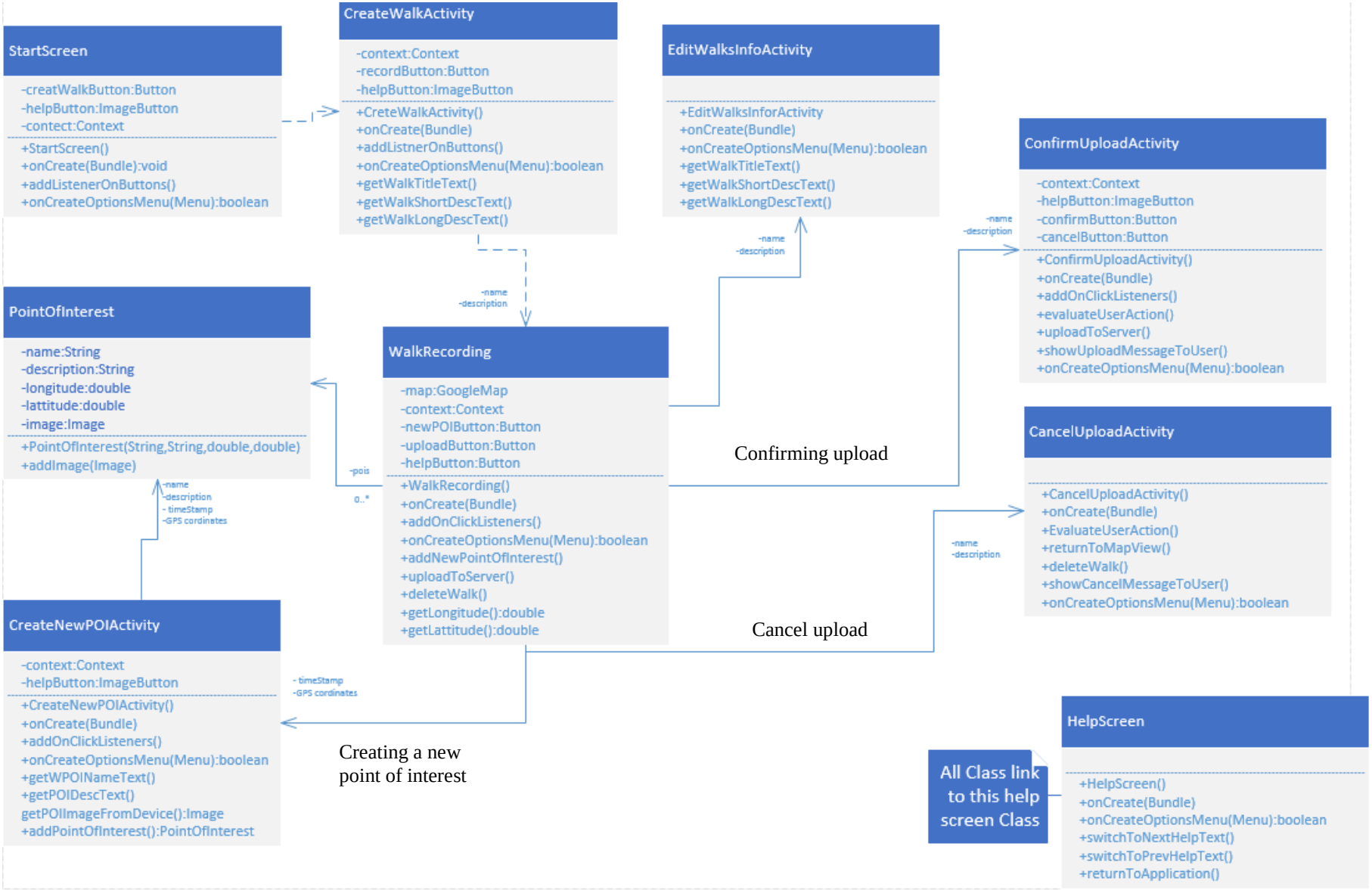
The web application refers to the website and the processing server required in order, producing a fully functional product.

On its own, the website will be able to retrieve information from a database; this information will be directly related to the number/details of currently saved walks sent from the android application. The information retrieved from the database will then be displayed initially as a list of all the stored walks (listed by title). Then a visitor to the website can click a walk and be shown the walks route, the information, the walks duration and all points of interest associated with the route.

The web application is heavily dependent on the Android application. If no information (e.g. walks) is being sent to the server from the application then the database will contain little or no entries resulting to none or few walks being displayed on the website. The main dependency is the web application with the Android application, as saved walks are needed to be on the web application to produce effective results.

This part of the application will be on the web server and any modern up to date browser.

## 2.2 Significant Classes:



**StartScreenActivity (FR 1, FR 7):**

The FrmHome class represents the initial screen presented to the user. This screen will be essential in the running of the application, as it houses a button which calls for the creation of the walk; until this button is clicked the user will not progress from this initial state, where only the FrmHome screen is displayed. In order to achieve this there will be a simple onClick listener attached to the button, which will simply call a different activity to be brought into focus, this activity will be CreateWalkActivity.

**CreateWalkActivity (FR 2, FR 7):**

The CreateWalkActivity class represents the screen where the user will be prompted for the information related to the walk. This information will include the walks name, short description and its longer more detailed description. This class also calls the creation of the WalkRecording class which will be the map/route display to the user.

**WalkRecording (FR3, FR 4, FR 5, FR 6, FR 7, FR 9):**

The WalkRecording class is used to model the current route using graphical representation. The map will display the user's current position and the current time elapsed along the route, as well as displaying all (if any) current Points of Interests along the current walk's path. The user can also create new Points of Interest and add them to the walk when viewing this screen.

**CreateNewPOIActivity(FR 3, FR 4, FR 7):**

The CreateNewPOIActivity class allows the user to specify the information required in order to create a new Point of Interest which can then be assigned to the current walk and stored locally ready for the server upload. This class is vital if the application is to achieve maximum functionality.

**PointOfInterest (FR 3, FR4):**

The PointOfInterest class is used to specify/hold the information related to each Point of Interest. The user can create multiple Points of Interests during one walk recording. It is essential to modulate the system, having a separate class to store/model the Points of Interest to allow this functionality.

**CancelUploadActivity (FR 5, FR 7):**

This class provides the user with the ability to cancel the upload to the server. This prevents the user accidentally uploading a walk that is unfinished or unwanted. The user will be prompted to make sure they understand that the walk will not be uploaded.

**ConfirmUploadActivity (FR 6, FR 7):**

This class provides the user with the ability to upload the walk to server. This allows the user to view the walk they have just recorded/uploaded via the web application.

## Main functions of the web application:

**Understanding HTTP POST requests (FR 6):**

The server will receive a POST request from the android application. It will parse this POST request and send the newly formatted information to the database system, which can then store the information in the correct tables in the database, ready for viewing on the website.

**Storing information in the database (FR 8, FR 9):**

There will be a DBMS (Database management system) in place to provide a suitable environment to externally store the information generated during the user's interaction with the Android application. This database can they be parsed by a script to produce the intended results when a person visits the website.

**The website (FR 8, FR 9):**

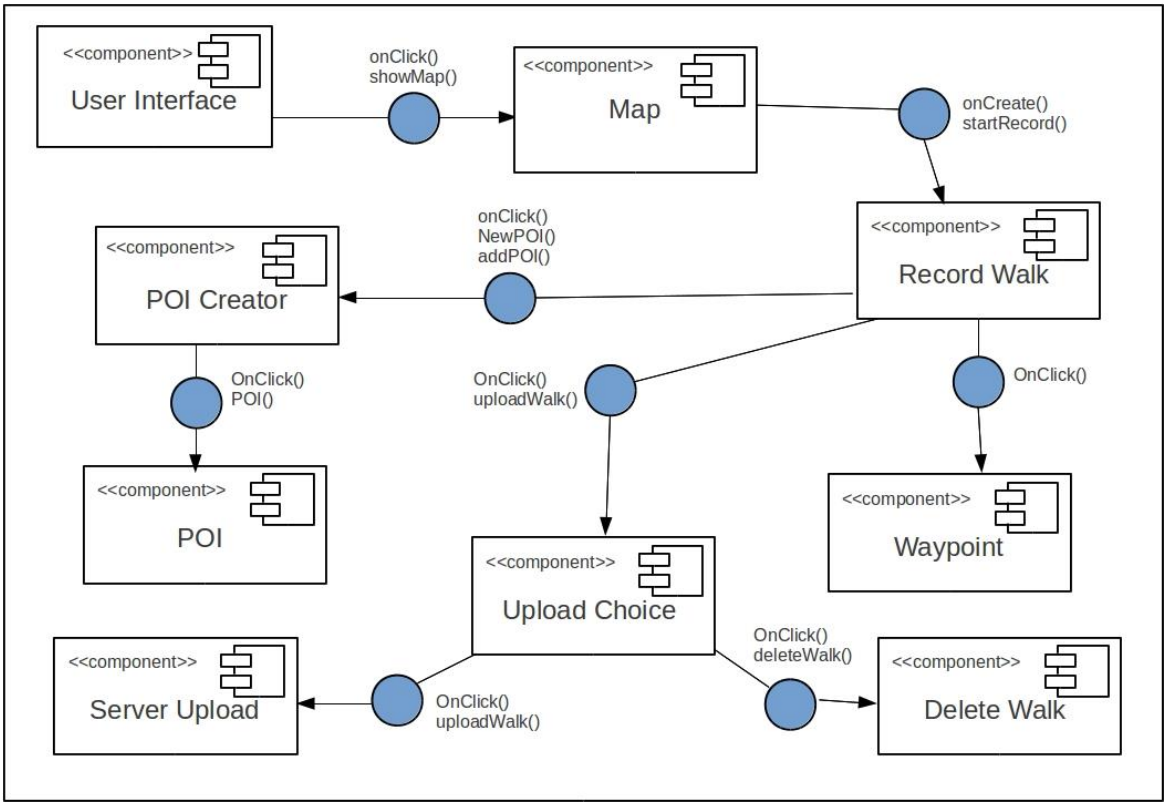
The web application will include a series of web pages which provide a suitable environment for the display of the stored walks information. This website will allow for user interaction and the user will be able to select which walk they want to view, this will directly reflect what information is retrieved from the database, thus producing different outputs for each walk.

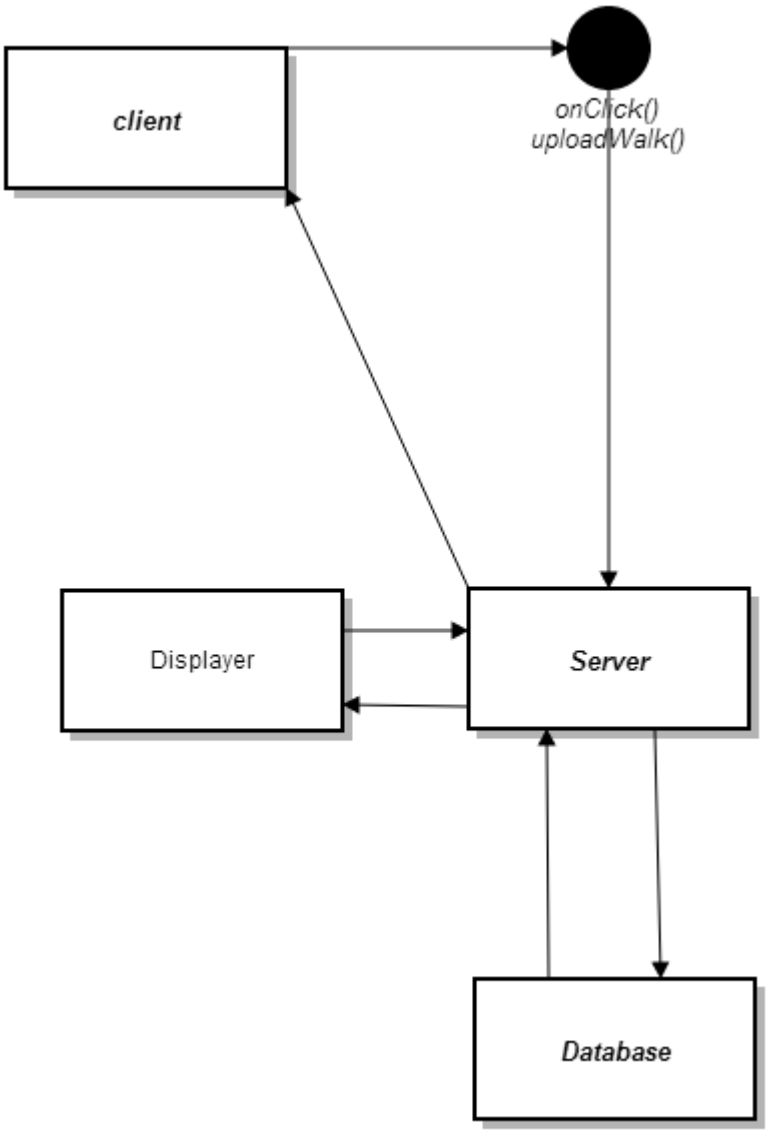
## 2.2 Mapping Function Requirements

| Functional Requirement: | Classes providing requirement: |
|---|---|
| **FR 1** | StartScreenActivity |
| **FR 2** | CreateWalkActivity |
| **FR 3** | WalkRecording, CreateNewPOIActivity, PointOfInterest |
| **FR 4** | WalkRecording, CreateNewPOIActivity, PointOfInterest |
| **FR 5** | WalkRecording, CancelUploadActivity |
| **FR 6** | WalkRecording, ConfirmUploadActivity, Understanding HTTP POST requests (Server) |
| **FR 7** | StartScreenActivity, CreateWalkActivity, CreateNewPOIActivity, CancelUploadActivity, ConfirmUploadActivity |
| **FR 8** | Storing information in the database, the website |
| **FR 9** | WalkRecording, Storing information in the database, the website |

# 3. DEPENDENCY DESCRIPTION

## 3.1 Component Diagram






How the classes map to the components
The component diagram shows the main system component which mainly relates to the Android application and how communication across the several modules and the server is accomplished. The below outlines the main ideas and principles behind this internal communication procedure:

**User Interface:**

The majority of the Android Classes will represent the user interface, as most of them extend Activity, thus providing a 'in focus' task on screen for the user to interact with. The only class which does not provide the user with some form of user interface directly is the PointOfInterest class. This is because it is the fundamental layout/data structure for every PointOfInterest (Location) on the walk, so the application user does not need to know how this information is stored or maintained.

The server also plays a role in the user interface as it provides the ability to store the walks in a database, which can then be accessed using MySQL queries. The walks can then be displayed on a webpage(s) in order to provide the user interface element of the web application.

**Displaying the map component:**

The map component is split between the two applications; both the Android client and the web application have functional requirements that specify a map must be displayed.

In regards to the Android application the main classes which feature in the displaying of the map and locations upon said map are as follows;

The WalkRecording class relates to the activity in which the actual map will be visible to the user. This will use the GoogleMaps for Android API's in order to achieve this. This is not the only class which will handle the displaying of the map for the Android client as it is a functional requirement that you can add locations to this map. This will be done using the interfaces provided in CreateNewPOIActivity class and the data stored in PointOfInterest class. The information will be relayed to the WalkRecording activity using the ability to add extra information to intents as well as using the provided startActivityForResult method, provided as part of the Android programming language specification.

The web application will display a map on the webpage and update the locations on this map based on the users selected walk. Once again the displaying of the map will be achieved using the GoogleMap's API and potentially JavaScript and PHP.

**Recording a walk (GPS)**

As an extension to the preceding paragraph which is related to the displaying of the GoogleMap inside the Android client, the requirement to record the walks GPS coordinates will also be handled using the WalkRecording class. This class will at some point use the LocationManager API's in order to access the network state of the device in order to achieve an accurate list of GPS coordinates along the walk. These coordinates can then be used to "trace" the route in which the walk has taken place.

**POI creator/ Creating new PointsOfInterest/ POI**

The ability to create new PointsOfInterest (locations) for the walk is handled using the interface provided in the CreateNewPOIActivity class. This class will provide a simple to use interface which will be used to enter information related to a new PointOfInterest. The information entered will be stored in respective variables/data structures in the POI class, and feedback to the WalkRecording activity which, as previously mentioned will handle where this POI will be displayed on the map object.

**Upload choice/ Upload to server**

The user must be prompted when upload to make sure that they do in fact want to upload to the server. The Android application will prompt the user using a User Interactive Dialog inside the ConfirmUploadActivity class. If the user does in fact mean to upload the walk to the server, then this will also be handled in the same class using some variation of a new thread or ASyncTask, in order to prevent possible complications associated with performing network connections on the main thread in Android development.

**Deleting a walk**

If the user does not want to upload the walk then the CancelUploadActivity will handle the cancellation of the walk upload. It states that if a walk does not want to be uploaded, then it must be deleted and a suitable prompt and warnings should be displayed to the user. If the walk is to be deleted then the CancelUploadActivity class will call the correct deletion methods in the WalkRecording class, this will effectively reset the application removing all points of interest and clearing the current map object of all information.

## 3.2 Inheritance relationships

There are two main inheritance relationships in the Android side of the walking tours application. These relationships are required in order to interact with the device and the provided API's as well as displaying information to the user in a valid format.

**Activity (Android provided superclass):**
The main inheritance relationship is the link between the Android Activity class and several classes in the application. The Activity superclass allows your designed screen to become a fully interactive screen when viewed on the android device, this is required to switch between views (e.g. map to upload screen). All of the classes that are listed against FR7, will extend Activity.

**FragmentActivity (Android provided superclass):**
FragmentActivity is the Class provided by Google Android API's. This Fragment Activity enables the application to display an instance of a GoogleMap object to the user. This is part of the functional requirements

as without this inheritance you cannot effectively deploy a solution using GoogleMaps effectively. This class is inherited by WalkRecording.

# 4. INTERFACE DESCRIPTION

## 4.1 Class Interface for CancelUploadActivity

```
//this class displays the cancel upload screen, ensuring the user wants to cancel the current walk

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

//outline for CancelUploadActivity class

public class CancelUploadActivity extends Activity {
// Activity is extended in order to implement and manage multiple screens with varying methods/activities.

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /* Creates an activity using the Activity super class
        *  The screen is then set to display the "cancel upload activity" interface.
        */
    }

    public void EvaluateUserAction() {
 //This method verifies that the user wants to cancel the upload.
    }

    public void returnToMapView() {
   //This takes the user back to the screen that has the map of the current walk displayed.
    }

    public void deleteWalk() {
  // This method removes all data for the walk that has been created so far.
    }
```

```
    public void showCancelMessageToUser() {
    //A message is displayed to the user telling them that the walk they created has been cancelled
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present based on the current interface.
        getMenuInflater().inflate(R.menu.cancel_upload_activty, menu);
        return true;
    }
}
```

**4.2 Class Interface for ConfirmUploadActivity**

```
//this class displays the confirm upload screen, ensuring the user wants to upload the current walk to the server.
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageButton;

public class ConfirmUploadActivity extends Activity {
// Activity is extended in order to implement and manage multiple screens with varying methods/activities.

    private Context context;
    private ImageButton helpButton;
    private Button confirmButton, cancelButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /* Creates an activity using the Activity super class
        *  The screen is then set to display the “confirm upload activity” interface.
        *  A help button will also be added using the addOnClickListeners() method so that the user can get
        *  help with anything they need in regards to the app.
        */

    }
```

```
    public void addOnClickListeners(){

        //adds click functionality to the help button which takes the user to the help screen (HelpScreen.class)



    }

    public void EvaluateUserAction() {
        //checks to see if the user really wants to proceed with the upload

    }

    public void uploadToServer() {
        //Sends all data regarding the walk that has been created to the server

    }

    public void showUploadMessageToUser() {
       /* A message is displayed to the user telling them that the walk they created has been successfully
       *  uploaded to the server.
       */



    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present based on the current interface.
        getMenuInflater().inflate(R.menu.confrim_upload, menu);
        return true;
    }

}
```

### 4.3 Class Interface for CreateNewPOIActivity

//this class displays the create new point of activity screen, allowing the user to create a new point of interest and set the name, description and image.

```
import android.media.Image;
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;

public class CreateNewPOIActivity extends Activity {
// Activity is extended in order to implement and manage multiple screens with varying methods/activities.

    private Context context;
    private ImageButton helpButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /* Creates an activity using the Activity super class
        *  The screen is then set to display the "create new poi activity" interface.
        *  A help button will also be added using the addOnClickListeners() method so that the user can get
        *  help with anything they need in regards to the app.
        */
    }


    public void addOnClickListeners() {


       //adds click functionality to the help button which takes the user to the help screen (HelpScreen.class)


    }



    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present based on the current interface.
        getMenuInflater().inflate(R.menu.confrim_upload, menu);
        return true;
```

```
    }

    public String getWPOINameText() {

 //the name of the point of interest is returned in the form of a String.
        return "DEFAULT VALUE";
    }

    public String getPOIDescText() {

  //the description is returned in the form of a String.
        return "DEFAULT VALUE";
    }

    public Image getPOIImageFromDevice() {

   //the image is returned in the form of an Image.
        return null;
    }

    public PointOfInterest addPointOfInterest(){

  //adds this PointOfInterest to the walk
    }

}
```

**4.4 Class Interface for CreateWalkActivity**

//this class displays the create walk screen, allowing the user to create a new walk and give it a title, short description and long description.

```
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageButton;

public class CreateWalkActivity extends Activity {
```

```
// Activity is extended in order to implement and manage multiple screens with varying methods/activities.

    private Context context;

    private Button recordButton;

    private ImageButton helpButton;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /* Creates an activity using the Activity super class
        *  The screen is then set to display the "create walk activity" interface.
        *  A help button will also be added using the addOnClickListeners() method so that the user can get
        *  help with anything they need in regards to the app.
        *  A record button will also be added using the addOnClickListeners() method that takes the user
        *  to the Walk Recording screen which monitors the walk as it progresses.
        */

    }

    public void addListenerOnButtons(){

        /*  adds click functionality to the help and record buttons which take the user to the help
        *  screen (HelpScreen class) and record screen (WalkingRecording class) respectively.
        */

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present based on the current interface.
        getMenuInflater().inflate(R.menu.create_walk, menu);
        return true;
    }



    public String getWalkTitleText() {

// The name of the walk is returned in the form of a String.
```

```
            return "DEFAULT VALUE";
        }



        public String getWalkShortDescText() {

       // the short description is returned in the form of a String.
            return "DEFAULT VALUE";
        }

        public String getWalkLongDescext() {

        // the long description is returned in the form of a String.
            return "DEFAULT VALUE";
        }
}
```

**4.5 Class Interface for EditWalksInfoActivity**

```
//this class displays the edit walk screen, allowing the user to modify the title, short description and long description for the current walk.
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class EditWalksInfoActivity extends Activity {
// Activity is extended in order to implement and manage multiple screens with varying methods/activities.

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            /* Creates an activity using the Activity super class
            *  The screen is then set to display the “edit walk activity” interface.
            */
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is present based on the current interface.
            getMenuInflater().inflate(R.menu.edit_walks_info, menu);
            return true;
        }
```

```
    public String getWalkTitleText() {
  // the walk title is returned in the form of a String.
        return "DEFAULT VALUE";
    }

    public String getWalkShortDescText() {
 //the short description of the walk is returned in the form of a String.
        return "DEFAULT VALUE";
    }

    public String getWalkLongDescext() {
   // the long description of the walk is returned in the form of a String.
        return "DEFAULT VALUE";
    }

}
```

**4.6 Class Interface for HelpScreen**

```
// this class displays the help screen, allowing the user to cycle through different tips.
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class HelpScreen extends Activity {
// Activity is extended in order to implement and manage multiple screens with varying methods/activities.

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /* Creates an activity using the Activity super class
        *  The screen is then set to display the “help screen” interface.
        */
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
         // Inflate the menu; this adds items to the action bar if it is present based on the current interface.
         getMenuInflater().inflate(R.menu.help_screen, menu);
         return true;
    }
```

```
    public void switchToNextHelpText() {
  //changes the help displayed on the screen to the next help tip/advice.

    }

    public void switchToPrevHelpText() {
   // changes the help displayed on the screen to the previous help tip/advice.

    }


    public void returnToApplication() {
        // the user is taken back to the page that they were last on before they accessed the help page.

    }

}
```

**4.7 Class Interface for PointOfInterest**

// this class allows for the creation of a 'PointOfInterest' object where each object has a name, description and values for both longitude and latitude. Some of these objects also have an image.

```
import android.media.Image;

public class PointOfInterest {


    private String name, description;
    private double longitude, latitude;
    private Image image;

    public PointOfInterest(String name, String description, double longitude, double latitude) {

   // assigns the values passed in to the method as parameters of an individual point of interest.
        this.name = name;
        this.description = description;
        this.longitude = longitude;
        this.latitude = latitude;
```

```
    }

    public void addImage(Image im){
   // sets the value of the image to the image passed in to the method.
        this.image = im;
    }


}
```

**4.8 Class Interface for StartScreen**

```
// this class displays the start screen, allowing the user to either create a walk or view the help screen
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.Toast;

public class StartScreen extends Activity {
// Activity is extended in order to implement and manage multiple screens with varying methods/activities.

    private Button createWalkButton;
    private ImageButton helpButton;
    private Context context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        /* Creates an activity using the Activity super class
        *  The screen is then set to display the "start screen" interface.
        *  A help button will also be added using the addOnClickListeners() method so that the user can get
        *  help with anything they need in regards to the app.
        *  A create walk button will also be added using the addOnClickListeners() method that takes the
        *  user to the CreateWalkActivity screen, allowing the user to create a new walk.
        */
```

```
    }

    public void addListenerOnButtons() {
        /*adds click functionality to the help and create walk buttons which take the user to the help
        *screen (HelpScreen class) and create walk screen (CreateWalkActivity class) respectively.
        */
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present based on the current interface.
        getMenuInflater().inflate(R.menu.start_screen, menu);
        return true;
    }

}
```

### 4.9 Class Interface for WalkRecording

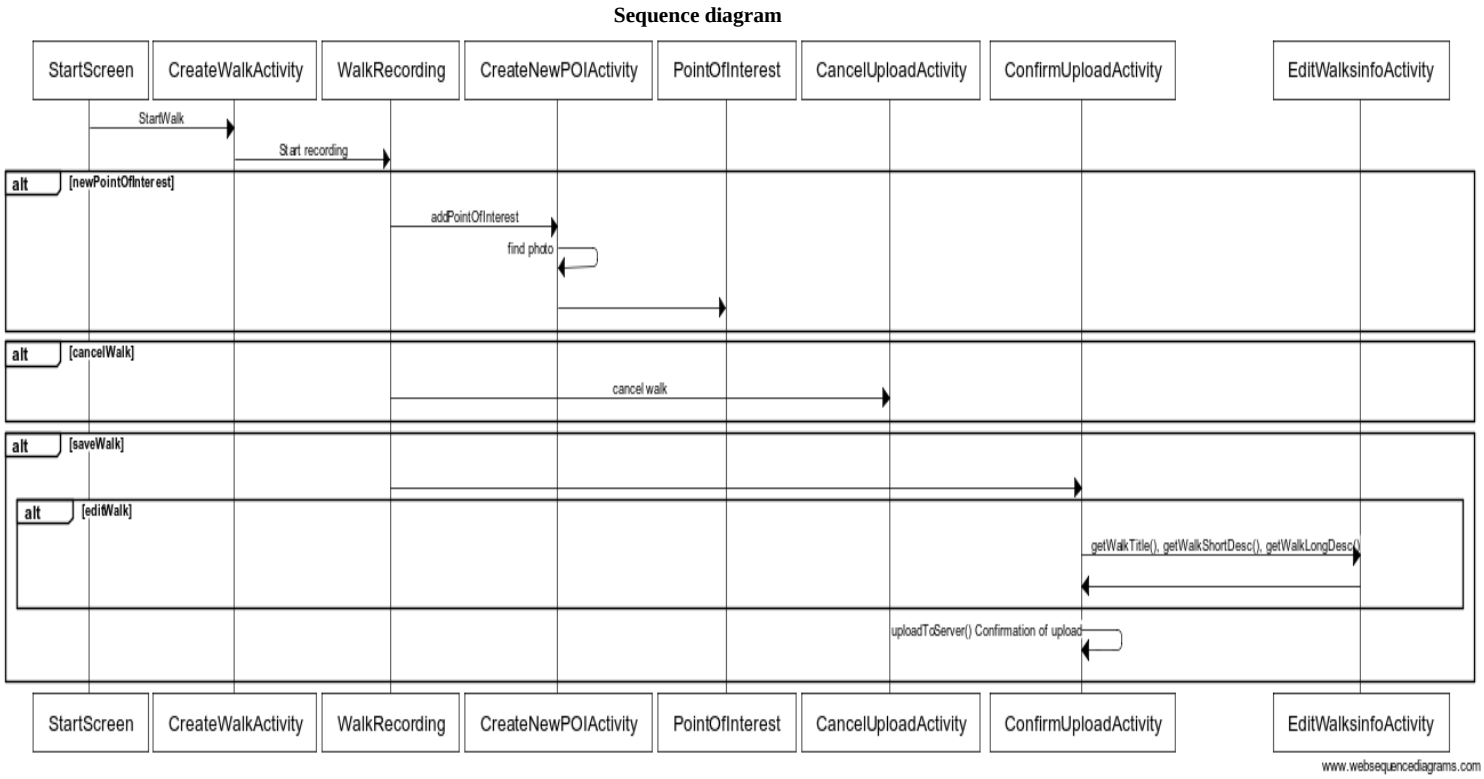
```
// this class displays the record walk screen and allows the user to add new points of interest, access the help
screen and view the current map.
import java.util.Vector;

import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.SupportMapFragment;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageButton;

public class WalkRecording extends FragmentActivity {
// FragmentActivity is extended in order to implement the map where the walk will be represented.
```

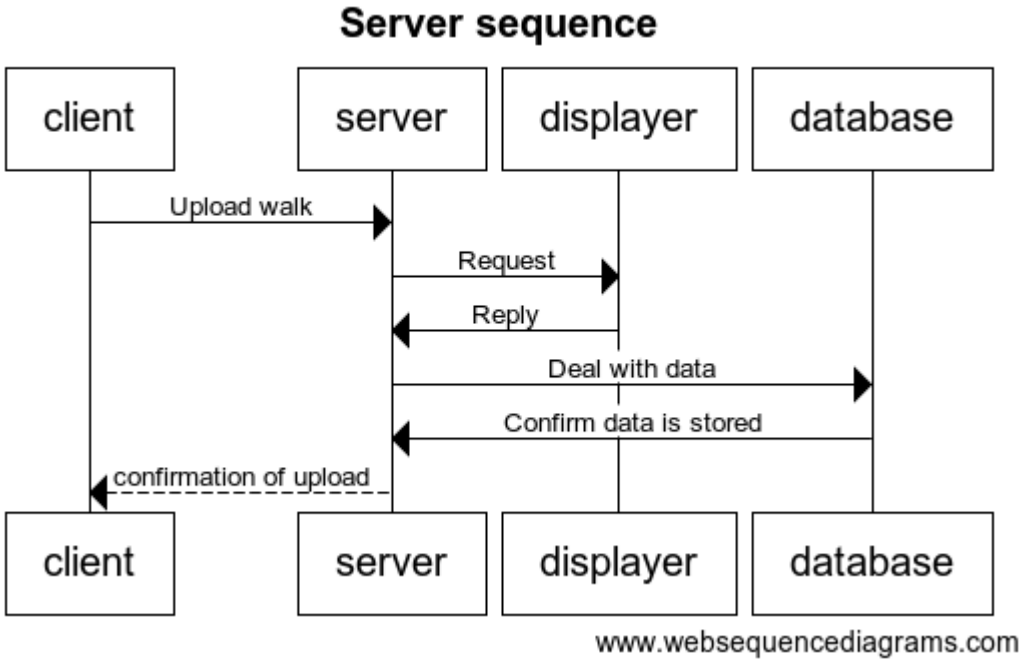
```
private GoogleMap map;
private Context context;
private Button newPOIButton, uploadButton;
private ImageButton helpButton;
private Vector<PointOfInterest> pois;

@Override
protected void onCreate(Bundle savedInstanceState) {
    /* Creates an activity using the Activity super class
    *  The screen is then set to display the "create walk activity" interface.
    *  The map is added to the screen using the FragmentActivity super class
    *  A vector of PointOfInterest is created in order to allow for numerous points of interest allowing
    *  them to be easily managed.
    *  A help button will also be added using the addOnClickListeners() method so that the user can get
    *  help with anything they need in regards to the app.
    *  A new point of interest button will also be added using the addOnClickListeners() method that
    *  takes the user to the CreateNewPOIActivity screen, allowing the user to create a new walk.
    * An upload button is added, allowing the user to finalise the walk and have it uploaded to
    * the server by going to the ConfirmUploadActivity screen.
    */

    }

@Override
public boolean onCreateOptionsMenu(Menu menu) {
     // Inflate the menu; this adds items to the action bar if it is present based on the current interface.
     getMenuInflater().inflate(R.menu.walk_recording, menu);
     return true;
}

public void addNewPointOfInterest() {
// adds a point of interest to the vector pois.

}

public void uploadToserver() {
// Sends all data regarding the walk that has been created to the server

}
```

```
        public void deleteWalk() {
     // removes all data that has been created so far and stops creating the walk



        }

        public double getLongitude() {
      // the longitude is returned to the app in the form of a double
              return 0;
        }

        public double getLatitude() {
    // the latitude is returned to the app in the form of a double
              return 0;

        }

}
```

# 5. DETAILED DESIGN

## 5.1 Sequence Diagram




## 5.2 Walking Tour Displayer

## 5.3 Overview

The Walking Tour Displayer is the second half of the product that we have been tasked to create. It is a web application that is used in conjunction with the Walking Tour Creator. The purpose of the Walking Tour Displayer is to show the user the routes they have saved via the Walking Tour Creator to a database.

## 5.4 Database

The Walking Tour Displayer requires a database, populated with routes created by the Walking Tour Creator. The Walking Tour Displayer will also accept uploads from the Walking Tour Creator and will store such uploads in the database accordingly.

### 5.4.1 Technologies

The database must be one utilising SQL, as specified in Functional Requirement 9. As such, the developers of the product have decided to use MySQL, or its close fork, MariaDB.

### 5.4.2 Structure

The structure of the database has been defined in the appendix to the Requirements Specification (SE.QA.RS).

This document uses SE.QA.RS version 1.4. Please note: more recent versions of the Requirements Specification may alter the requirements significantly, and have not been taken into account for this iteration of the document.

Design Constraint 3 makes it clear that the structure is mandated and cannot be modified from the specification.

The database has four tables: a “walks” table, a “locations” table, a “places” table, and a “photo” table.

#### 5.4.2.1 List of Walks

The first table stores basic information about each individual route. The word ‘route’ is used interchangeably with ‘walk’. The routes table, which shall be named **tbl_routes**, contains seven fields, thus:

| Field name | | Field type | (SQL type) | Description / notes |
|---|---|---|---|---|
| id | PK | Integer | int | Primary key. |
| Title | | Text | varchar(255) | Name of the walk |
| shortDesc | | Text | varchar(255) | Subtitle of walk |
| longDesc | | Text | varchar(1024) | Longer description |
| hours | | Float | float | Hours a walk should take |
| distance | | Float | float | Kilometres a walk takes |

#### 5.4.2.2 Location

The second table, **tbl_locations**, stores details of Points of Interest displayed on each walk.

| Field name | | Field type | (SQL type) | Description / notes |
|---|---|---|---|---|
| id | PK | Integer | int | Primary key. |
| walkId | FK | Integer | int | Foreign key from tbl_routes |
| latitude | | Float | float | A decimal representation of GPS coordinates, using WGS84. |
| longitude | | Float | float | |
| timestamp | | Float (?!) | float | Seconds elapsed since walk began |

#### 5.4.2.3 Place Description

The third table, **tbl_places**, is used to associate locations with routes. The Requirements Specification states that the ID is used to order items within a walk.

| Field name | | Field type | (SQL type) | Description / notes |
|---|---|---|---|---|
| id | PK | Integer | int | Primary key. |

| | | | | |
|---|---|---|---|---|
| locationId | FK | Integer | int | Foreign key from tbl_locations |
| description | | Text | varchar(255) | Description of location |

#### 5.4.2.4 Photo Usage

The final table, **tbl_images**, is used to associate pictures with places.

| Field name | | Field type | (SQL type) | Description / notes |
|---|---|---|---|---|
| id | PK | Integer | int | Primary key. |
| placeId | FK | Integer | int | Referenced from tbl_places |
| photoName | | Text | varchar(255) | JPEG filename (without extension) |

### 5.4.3 Web Application

The application shall be a PHP application running on an Apache HTTP server. Both PHP and Apache HTTP Server are tried, tested, and trusted technologies. To allow quick development, the webpages will use the Bootstrap front-end framework for a springboard for HTML and CSS templates.

#### 5.4.3.1 Requirements

The requirements for the Web Application, also known as the Walking Tour Displayer, can be found in the Requirements Specification (SE.QA.RS).

The relevant Functional Requirements are Functional Requirements 6, 8 and 9. For convenience, FR8 is replicated below:

> There will be a separate program (web application), the Walking Tour Displayer (WTD) that lets the user select a walk to display. When a walk is selected, the user will see each of the places included in the walk displayed at the correct coordinates on a map. The user will be able to choose a place, and see the correct details for that place within the selected walk, including textural [sic] descriptions and images.

The requirements for the Walking Tour Displayer seem to be as follows:

- There is a web application known as the Walking Tour Displayer
- It shall have a list of all routes generated by users, which will show the name and sub-title of each
- A user can select a route and view a second page with more detailed information to do with the route
- This Route View shall have a longer description of the route
- The Route View will have a map of the route, with all locations along the way shown
- A user can click a location and view images and descriptions associated with such a route

#### 5.4.3.2 File Structure

Although, theoretically, the entire web application could be written in one long index.php file, it is not good practice. The scripts will be split into a couple of directories as follows:

- The root directory will contain the entry-points for the server, for example, index.php and upload.php
- An includes directory will contain functions and code used in multiple scripts. These functions will be included through use of the require and include functions built into PHP.
- An uploads directory will contain any JPG files the Walking Tour Creator has uploaded to the server.
- A resources directory will contain static assets, such as CSS and JavaScript.
- A templates directory will contain templates used to define the layout of the rendered HTML pages.

#### 5.4.3.3 Viewing Walks

The initial homepage index.php will show a list of all walks uploaded to the server, displaying in a table the names and subtitles of each. Clicking on a name of a walk will take you to a second page (e.g. index.php?id=2) that will show a map of the route. The map shall show the points of interest specified in the walk, and interacting with a point of interest will display any photographs or descriptions associated with it.

#### 5.4.3.4 Uploading Walks

FR6, replicated below, defines the way that the server communicates with the client.

> The user should be able to save the walk, by sending it to a server where it is saved into a database. The message should be formatted as a Multipurpose Internet Mail Extensions (MIME) message and sent to the server via an HTTP POST to a predefined URL.

However, FR6 gives considerable leeway in the way that the data is encoded. It is proposed that the client will send a JSON ('JavaScript Object Notation') formatted request to the server (at upload.php), encoding the data that the server should save into the database. For example:

```
{
    "authorization": {
        "hash": "76EADA9B070BB27659359220A460C264C34745A9",
        "salt": "swordfish"
    },
    "walk": {
        "title": "Whitehall Wander",
        "shortDesc": "A short walk around Westminster",
        "longDesc": "A walk around London, viewing sights such as Downing
Street, Trafalgar Square, and Scotland Yard",
        "locations": [
            {
                "latitude": 51.503396,
                "longitude": 0.127640,
                "timestamp": 0,
                "descriptions": [
                    "10 Downing Street"
                ],
                "images": [
                    "no10door",
                    "primeminister"
                ]
            },
            {
                "latitude": 51.506758,
                "longitude": 0.128692,
                "timestamp": 20,
                "descriptions": [
                    "Admiralty Arch"
                ],
                "images": [
                ]
           },
           {
                "latitude": 51.49861,
                "longitude": 0.13305,
                "timestamp": 60,
                "descriptions": [
                    "New Scotland Yard",
                    "Metropolitan Police HQ"
                ],
                "images": [
                    "revolvingsign"
                ]
           }
        ]
    }
}
```

Images should be uploaded separately, with a request for each, by POSTing the file and its name to imageUpload.php.

#### 5.4.3.4.1 JSON Schema

| Name | Type | Description |
|---|---|---|
| **authorization** | Object, containing one **hash**, and one **salt** | A container to ensure that an upload definitely comes from the server |
| **hash** | String | SHA1 of the **salt** combined with a secret passphrase hardcoded into the client and server |
| **Salt** | String | Randomly generated to ensure the hash is different every time |
| **Walk** | Object, containing one of **title**, **shortDesc**, **longDesc**, and **locations** | A walk is one fully contained route. |
| **Title** | String | The title of the route, to be stored in the database |
| **shortDesc** | String | The short description of the route |
| **longDesc** | String | A long description of the route |
| **locations** | Array of **location** | |
| **location** | Object, containing a **latitude**, a **longitude**, a **timestamp**, a **descriptions**, and an **images** | One point of interest along the route |
| **latitude** | Float | A decimal representation of the latitude of the point of interest |
| **longitude** | Float | A decimal representation of the longitude of the point of interest |
| **timestamp** | Float | The time since the start of the walk that the point of interest is reached, in minutes |
| **descriptions** | Array of zero or more Strings | Descriptions of the point of interest |
| **images** | Array of zero or more Strings | The name of images associated with each point of interest, without file extension |

#### 5.4.3.4.2 Server Responses

The server should respond with a HTTP header with a status code as defined in RFC 2616. In the event of an error, a JSON formatted error message may be sent to the client, who should display it accordingly. Common HTTP status codes are specified below:

| | |
|---|---|
| 200 OK | The server received the information and all was well. |
| 201 Created | The server received the information and created a new walk. |
| 400 Bad Request | The JSON was not formatted correctly; the walk was not saved. |
| 401 Unauthorized | The authorization failed, and the hashes did not match; the walk was not saved. |
| 404 Not Found | The data was not sent to the correct URL; the walk was not saved. |
| 500 Internal Server Error | The server encountered a problem; the walk was not saved. |
| 503 Service Unavailable | The server is overloaded; try again later; the walk was not saved. |

#### 5.4.3.4.3 JSON

| | | |
|---|---|---|
| Test for config disallowing uploads | 400 | Uploading has been disabled by the system administrator. |
| Disallow GETs | 405 Method Not Allowed | Only POST is accepted. |
| Catch empty POST requests | 400 | No POST variables were sent. |
| POST needs a key to send its data within key-value pairs.<br>The key "data" has been chosen. | 400 | Data was not POSTed correctly. |
| Check if JSON decode failed; PHP does this by returning null on failure, and then has another function to see what the error was | 400 | Unable to parse JSON. |
| Verify authorization<br>/<br>authorization section exists | 400 | No hash present.<br>/<br>No salt present |
| SHA1 is 40 chars long when expressed in hex | 400 | Hash is wrong length. |
| Hashing is case sensitive; as PHP returns lowercase. we need to ensure the user's hash is also lowercase | 400 | Invalid credentials. |
| This will be shown when there are no credentials found. | 401 is not applicable as we are not using authentication headers | No credentials found. |
| Check there is actually data | 400 | No route found. |
| We are enforcing that no properties are optional this does not check whether the properties contain anything<br>And instead checks merely whether they exist. | 400 | Route has no title.<br>Route has no subtitle.<br>Route has no description.<br>Route has no locations. |
| Check that all mandatory properties exist | 400 | Location $index has no latitude.<br>Location $index has no longitude.<br>Location $index has no timestamp.<br>Timestamp $index is not numeric.<br>Location $index has no descriptions.<br>Location $index has no images.<br>Location $index has no name. |
| Sanity checks some latitude is drawn from the equator (0) to the poles at 90 and -90. | 400 | Latitude $index is out of bounds. |
| longitude is east-west, and is therefore 0 (Greenwich) to<br># +-180, which are equivalent. | 400 | Longitude $index is out of bounds. |
| If a timestamp is not numeric | 400 | Cannot parse timestamp $index. |
| If a location-description is not an array | 400 | Location $index has bad descriptions (not an array). |
| If a description is not in an array | 400 | One or more descriptions are not in an array. |

| | | |
|---|---|---|
| If the description array is “!=2” | 400 | One or more description arrays are malformed. |
| If a location contains bad images | 400 | Location $index has bad images (not an array). |
| If an image tied to a location is not in an array | 400 | Non-arrays are not in the image array. |
| If the image array is “!=2” | 400 | One or more image arrays are malformed. |
| simplistic way of checking if name has an extension | 400 | $image [0] does not appear to have a file extension. |
| If an image is equal to another image | 400 | File name collision; $image [0] already exists and we won't clobber. |
| If an image has the wrong encoding | 400 | $image [0] has bad base64 encoding; check and try again. |
| Server error | 500 | File put failed; server error", true. |
| Default server error | 500 | File upload failed; no further information available. |
| Saving to the database failed. This is a server error, not a client issue, so set 500. | 500 | Unable to save to database: $dbInput". |

# 6. BIBLIOGRAPHY

Help with globe logo
http://24p.com/wordpress/index.php?cat=5&paged=3 accessed 22/01/2014

Help with icon
http://www.iconarchive.com/show/plump-icons-by-zerode/Help-Support-icon.html accessed 22/01/2014

Help with Button null pointer:
http://stackoverflow.com/questions/20957891/android-why-is-this-button-satying-as-null-when-assigning-its-value-based-off-f accessed 24/01/2014

GPS tracker:
http://stackoverflow.com/a/15757944/2942536 accessed 24/01/2014

Photo help
http://developer.android.com/training/camera/photobasics.html accessed 06/01/2014

Timer help:
http://stackoverflow.com/questions/21316366/android-how-can-i-update-a-view-element-from-a-private-inner-class-which-is-an accessed 27/01/2014

Timer problem:
http://stackoverflow.com/questions/21319364/android-how-to-stop-my-timer-correctly-display-the-hhmmss-they-carry-on-counti accessed 27/01/2014

base64 sting bitmap
http://stackoverflow.com/q/14926005/2942536 accessed 29/01/2014
http://mobile.cs.fsu.edu/converting-images-to-json-objects/ accessed 29/01/2014

JSON error messages
http://github.com/cs22120/server/upload.php accessed 29/01/2014

Distances between points
http://forums.asp.net/t/1952508.aspx?Draw+lines+between+markers+on+Google+Map+ accessed 30/01/2014

Map workings and Google APIs
https://developers.google.com/maps/articles/phpsqlajax_v3 accessed 28/01/2014

QA Document SE.QA.03 - General Documentation Standards.

QA Document SE.QA.08 - Operating Procedures and Configuration Management Standards.

# DOCUMENT HISTORY

| Version | CCF No. | Date | Changes made to document | Changed by |
|---|---|---|---|---|
| 0.1 | N/A | 2014-12-03 | Collated document | bar5 |
| 0.2 | N/A | 2014-12-06 | Additional collating | bar5 |
| 0.3 | N/A | 2014-01-27 | | ays8 |
| 0.4 | N/A | 2014-01-29 | Formatting, spelling and grammar checks and adding additional content and bibliography | bar5 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# Software Development Group Project 02
# Test Procedure Standards


| | |
|---|---|
| Author: | Ben Rainbow;Dillon Cuffe;Jostein Kristiansen, Ashley Smith;James Woodside;Douglas Gardener;Luke Horwood;Chris Edwards |
| Config Ref: | SE_02_TE_01 |
| Date: | 2014-01-27 |
| Version: | 0.3 |
| Status: | Released v1.1 |



*Department of Computer Science*
*Aberystwyth University*
*Aberystwyth*
*Ceredigion*
*SY23 3DB*

# CONTENTS

# 1 INTRODUCTION

The contents on this document will show the understanding of the tests and the planning of these tests onto the Walking Tour App that is currently being developed.

## 1.1 Purpose of this Document

The below content of this document shows the tests that will be carried out onto the developing system, these tests are necessary to ensure that the walking tour app will have full functionality and follows all the protocols that will be required.
The tables within the document show the input of the test which is the procedure that we wish the app to process and show the correct response which is the expected output; furthermore the test will be passed or failed depending on the result of the output from the app.
In addition the purpose of this document is to show that the application works correctly so that it can be used without any malfunctioning problems or coding errors.

## 1.2 Scope

This document will look at the testing of the written software; it will do this through the means of system level tests. This document will identify areas within the system that can be tested, it will provide information that can be used to test that area and provide details of the result which we expect to receive from that test. This document will give the testing team a guideline on what tests need to be completed to ensure the functionality of the system.
This document will not include any tests at a unit level.

## 1.3 Objectives

The Main objectives of this document are:

1. To ensure that the system functions correctly.
2. All output information is correct in relation to input.
3. To provide testers with a basis to conduct their tests.
4. To provide a test for every system requirement.
5. To provide details of each test and its expected outcome.

# 2 TEST SPECIFICATION

| Test Ref. | Req being tested | Test Content | Input | Output | Pass criteria |
|---|---|---|---|---|---|
| SE-F-001 | FR1 | Check that Android application loads without complication | Using an Android based device (e.g. mobile phone or tablet) run the application. | The application should load successfully | The application loads correctly |
| SE-F-002 | FR1 | Check that the Start Screen is displayed correctly | Run the application on the android based device | Design screen should be displayed to the user as it was designed | The start screen is displayed, correctly, to the user |
| SE-F-003 | FR1 | Check that the user is prompted to input some basic details when a new walking tour is created. | Start a new walking tour, and look at the fields in the following window. | A new window should be displayed, along with some fields and descriptions for those fields. | The user is prompted for information, and the fields are displayed in an understandable manner. |
| SE-F-004 | FR1 | Check that the GPS recording begins recording at the right time. | The GPS recording begins after the user has entered the walks information (FR2 Test) and confirmed that the information is correct. | The users GPS should be tracked and the route should be displayed on screen to the user. | GPS recording starts after the user has confirmed the walks details; the GPS location is displayed and correctly updated to the screen. |
| SE-F-005 | FR1 | Check that the user is shown the option for cancelling the walk | Cancel the current walking tour. | Walking tour cancel screen is displayed | Walking tour cancel screen is displayed shortly after option is selected |
| SE-F-006 | FR1 | Check the user is shown The option for saving the current walking tour | Save the current walking tour | Save screen shows when the option is selected | Save screen is displayed shortly after the option is selected |
| SE-F-007 | FR1 | Check the user is shown the option for Adding a point of interest to the current walking tour | Add a new point of interest to the current walking tour | Screen for user to input new point of interest details is displayed | Point of interest screen is shown shortly after the option is selected |
| SE-F-008 | FR2 | Check that The user can specify the name, short description and the long description for the walk. | Enter “group02” for the walks name, “short description” for the short description and “This is a long description for the walk” for the long description. | The walk should have the following attributes:<br>Name;<br>- Title of walk<br>Short description;<br>- short description of walk<br>Long description;<br>- Long description of walk” | The walks attributes are correctly set. |
| SE-F-009 | FR2 | Check that the user | Enter “’{]%&£$/=” as | The application should not | Test passed if |

| | | | | | |
|---|---|---|---|---|---|
| | | cannot input other characters than letters, A-Z (upper/lower case), and numbers, 0-9, in the title of the walking tour. | the title of the walking tour | accept this as input, and will tell the user to input another(valid) title for the walking tour | the input is rejected and the user is told that they cannot input such characters. |
| SE-F-010 | FR2 | Check title field cannot be submitted as empty | Leave title field empty and try to create the new walking tour | Application should inform user of blank field and ask them to supply information | Test passed if input is rejected, and the user is told to fill in any empty fields |
| SE-F-011 | FR2 | Check short description field cannot be submitted as empty | Leave short description field empty and try to create the new walking tour | Application should inform user of blank field and ask them to supply information | Test passed if input is rejected, and the user is told to fill in any empty fields |
| SE-F-012 | FR2 | Check long description field cannot be submitted as empty | Leave long description field empty and try to create the new walking tour | Application should inform user of blank field and ask them to supply information | Test passed if input is rejected, and the user is told to fill in any empty fields |
| SE-F-013 | FR2 | Check the title field only accepts 1 word(<15 characters) | Input more than 15 characters | Input should not be accepted and the user should be informed that they have entered too many characters. They should be informed to remove characters | Test passed if user informed that they have entered too many characters and are required to remove some |
| SE-F-014 | FR2 | Check the short description field only accepts <=100 characters | Input more than 100 characters | Input should not be accepted and the user should be informed that they have entered too many characters. They should be informed to remove characters | Test passed if user informed that they have entered too many characters and are required to remove some |
| SE-F-015 | FR2 | Check the long description field on accepts <=1000 | Input more than 1000 characters | Input should not be accepted and the user should be informed that they have entered too many characters. They should be informed to remove characters | Test passed if user informed that they have entered too many characters and are required to remove some |
| SE-F-016 | FR3 | Check that the ability to add points of interest to the walk functions correctly | The point of interest button will be pressed during the recording phase of the walk (see FR4 tests for adding a photograph). | Screen displaying newly created point of interest should be shown | Point of interest is added to the current walk |
| SE-F-017 | FR3 | Check name is saved when point of interest is saved | User will input the name on the point of interest details screen | Name entered should be displayed correctly on the map screen of the point of interest | Name should be displayed on point of interest and should be correct to user's input |
| SE-F-018 | FR3 | Check description is saved when point of interest is saved | User will input a description on the point of interest details screen | Description should be displayed correctly in the details of the point of interest | Description should be displayed on point of interest details and should be |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | correct to user input |
| SE-F-019 | FR3 | Check time stamp is correct when point of interest is saved | Time will be saved when point of interest is saved | Timestamp of the point of interest should be the same as the time the point of interest was created | Time stamp should be correct to time of point of interest creation |
| SE-F-020 | FR3 | Check GPS coordinates are used correctly when saving a point of interest | GPS coordinates will be retrieved and assigned to this point of interest | Point of interest should display on the map, where the point of interest was saved | GPS coordinates should be accurate to where the point of interest was created |
| SE-F-021 | FR3 | Check that the input in the name, and description fields are not invalid (I.e. empty, containing invalid characters, or too large. | Input should not be empty, should only contain letters found in the English alphabet and numbers 0-9, and it should not exceed the upper character limit for that field | Input is only accepted if it is valid; input is rejected if it is invalid, and a message explaining the rejection is displayed. | The input to the field(s) is valid, and the user can confirm that this information is correct before saving this location. |
| SE-F-022 | FR3 | Check input of invalid data into name and description fields | Input can be empty or a series of symbols. E.g. "!£$%^" | Application should reject information as invalid and ask user for valid information | Test passed if application does not accept information as valid |
| SE-F-023 | FR4 | Check that the ability to add photographs from the camera functions correctly. | The user can create a point of interest with (an) attached photograph(s); this photograph can be taken with the device's camera then added to the point of interest as a jpg. | The camera application will open and the taken photo will be added to the point of interest. This photo will be displayed when viewing the point of interest. | The camera application opens successfully and the taken photograph is returned to the walking tour application and assigned to the correct point of interest as a jpg. |
| SE-F-024 | FR4 | Check that the ability to add photographs from the device's image library functions correctly. | The user can create a point of interest with an attached photograph; this photograph can be selected from the stored jpg images on the device, and then added to the point of interest. | The device's media library will open and the taken photo will be added to the point of interest. This image will be displayed when viewing the point of interest. | The device's media library application opens successfully and the selected jpg image is returned to the walking tour application and assigned to the correct point of interest. |
| SE-F-025 | FR4 | Check that the input in the name, and description fields are not invalid (I.e. empty, containing invalid characters, or too large. | Input should not be empty, should only contain letters found in the English alphabet and numbers 0-9, and it should not exceed the upper character limit for that specific field | Input is only accepted if it is valid; input is rejected if it is invalid, and a message explaining the rejection is displayed. | The input to the field(s) is valid, and the user can confirm that this information is correct before saving the jpg image. |
| SE-F-026 | FR4 | Check GPS coordinates | GPS coordinates will | GPS coordinates should | The jpg image is |

| | | | | | |
|---|---|---|---|---|---|
| | | are added to image | be retrieved and assigned to the image on saving | match the location of where the image was taken | attached to the specified location with the GPS location |
| SE-F-027 | FR4 | Check Timestamp is added to image | Time will be saved and applied to the image when saved | Time stamp should be the same as when the image was taken | Time stamp is applied to the jpg image and is accurate to time of photo being taken |
| SE-F-028 | FR5 | Check that the user is able to abandon the uploading of the current walk. | Before uploading, the walk's review is displayed to the user. The user can then decide if they want to upload it to the server, or cancel the upload. This test will be carried out under the presumption that the user clicks the cancel upload button. | The current walk should not be uploaded and the user should be prompted for confirmation that they want to cancel this upload and stop recording. | The upload should not take place so the current walk should not be stored in the database, or be parsed by the server. |
| SE-F-029 | FR5 | Check that the prompt informing the user that they can edit the walk instead of deleting it completely displays correctly and the response executes the correct action. | When the cancel upload button is pressed, the option of editing the existing walk should be displayed to the user. This will then continue the walk's recoding (as if upload wasn't pressed) or deleted the current walk. In this case the user will select to edit the walk. | The walk will continue recording, maintaining the current route and points of interest. | The walk is displayed correctly on the map maintaining all its information. |
| SE-F-030 | FR5 | Check that the prompt informing the user that they can edit the walk instead of deleting it completely displays correctly and the response executes the correct action. | When the cancel upload button is pressed the option of editing the existing walk should be displayed to the user, this will then continue the walks recoding (as if upload wasn't pressed) or deleted the current walk. In this case the user will select to cancel the walk. | The user will be told that the walk will now be deleted, the walk will then be deleted locally and should not be present in the database and not be parsed by the server. | The walk is correctly deleted from all aspects of the application. |
| SE-F-031 | FR5 | Check that deleting a route will also delete all the locations associated to the walk. | When deleting a route, any saved points of interests and images created for this walking tour will be removed locally and from the database (if they exist there). The user will be prompted for confirmation of this action. | The user is informed that deleting this walk will permanently remove all locations, information, and images associated with the walk, from their device; A confirmation of this action is needed to finish the deletion. | The user is correctly prompted about the application's actions and the correct deletion operations take place. |

| | | | | | |
|---|---|---|---|---|---|
| SE-F-032 | FR6 | Check that a walk can be correctly uploaded to the server. Check the target URL is correct and present. | The upload confirmation button will be pressed, sending the current walk's information to the server as a Multipurpose Internet Mail Extension (MIME) message via an HTTP POST to the predefined URL. | The user's recently uploaded walk will be sent to the server, containing information about: name, title, long and short descriptions (as per FR2). List of GPS coordinates for the walk, from start to end, with a timestamp for each location. List of locations (points of interest) with associated information (as per FR3). Photos with associated information (as per FR4). | The walk is uploaded, and it matches the format expected by the server. The walk is saved. |
| SE-F-033 | FR6 | Check that the user is shown a summary of the walk before confirming the upload to the destination sever. | When the submit button is pressed all the current information associated with the walk should be retrieved and displayed neatly to user (including points of interest), before the upload is initiated. | The walks information will be retrieved and displayed in a neat, readable manner to the user. The information displayed should match the information added to the walk during the recording session. | The information is displayed correctly in a neat readable manner and containing the correct data. |
| SE-F-034 | FR7 | Check that the walking tour application can be minimised and still maintain functionality while running in the background, and after being reopened. | Minimise the walking tour application, run a different application, and finally reopen the walking tour application again. | The application can correctly drop in and out of focus whilst maintaining integrity and functionality. All information about a walk that was present before the user switched application will still be available when the user comes back to the walking tour application, and the GPS tracking should have captured your movements while minimised. | The switching between applications goes smoothly. There is a clean transition, no data/information is lost during this transition phase, and your route has been recorded while minimised. |
| SE-F-035 | FR7 | Check that it is not possible to open two separate instances of the walking tour application. | Attempting to launch a new instance of the walking tour application while another instance of the application is already running in the background. | Instead of opening a new instance of the walking tour application, the device will reopen the minimised instance of the application running in the background. | Minimised instance of the walking tour application is reopened when the icon for the application is activated on the device. |
| SE-F-036 | FR8 | Check that the web application can display a list of all walks and also a specific walk's information. | The web application can retrieve all walks currently saved in the database and display these as a neat list, each element in the list being a link to a walk. The user will then click the walk they want to view. | When a link is clicked, the correct walk will be displayed on a map in a new webpage. This map will display the walk's information about the route, including its name, duration and its descriptions. The Map will also contain every point of interest and its correct GPS coordinates. These points | The website correctly displays all saved walks. Each link displays the correct walk's information on the map, in the correct format. Points of interest also |

| | | | | | |
|---|---|---|---|---|---|
| | | | | of interest, when selected, will display all their information (name, description, timestamp, and jpg image(s)). | exhibit the correct functionality, and display all of the information the user expects to see. |
| SE-F-037 | FR9 | Check that the server can understand the POST message and parse the information to the database. | The information about the current walk is sent to the server under the circumstances of FR6. The server will understand the POST request and correctly store the data in the specified SQL database ready for retrieval. All information about the walk should be present in the database. A message should be displayed to the user stating if the upload was successful. This walk should then be viewable in the web application (see FR8) | The server is contacted by the application and the POST message is sent in the correct format (FR6). The walk's information is then stored correctly in the database and is viewable in the database's tables.<br>A message will be displayed to the user to confirm a successful upload.<br><br>If the upload is unsuccessful an error message will be displayed (e.g. upload failed, check internet connection).<br>If the upload is successful, a walk, and all its information, should be viewable in the web application. | The user is informed about the successful completion of the upload.<br>If successful, the walk's data will be viewable as expected in the web application and database. |

# BIBLIOGRAPHY

1. **C.J. Price, B.P.Tiddeman.** *Walking Tour Creator Requirements Specification.* [PDF Document] Aberystwyth : Aberystwyth University, Aberystwyth University, 2013.

2. **C.J.Price, N.W.Hardy, B.P.Tiddeman.** *Design Specification Standards.* [PDF Document] Aberystwyth : Aberystwyth University, 2013.

## DOCUMENT HISTORY

| *Version* | *CCF No.* | *Date* | *Changes made to document* | *Changed by* |
|---|---|---|---|---|
| 0.1 | N/A | 2013-11-11 | Test Specification added to the document | jok13 |
| 0.2 | N/A | 2013-11-14 | Contents Page updates, Purpose of document updates | dic6 |
| 0.3 | N/A | 2014-01-27 | Changes based on feedback provided | ays8 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# TEST LOG FORM

| | | |
|---|---|---|
| Test Log No: 2 | Group: 02 | Testers(s):- jaw57, jok13 |
| Date: 30/01/14 | Tagged version ID:-1.4 | |

| Test ID | Pass / Fail | Fail description | CCF / issue # |
|---|---|---|---|
| SE-F-001 | Pass | N/A | |
| SE-F-002 | Pass | N/A | |
| SE-F-003 | Pass | N/A | |
| SE-F-004 | Pass | N/A | |
| SE-F-005 | Pass | N/A | |
| SE-F-006 | Pass | N/A | |
| SE-F-007 | Pass | N/A | |
| SE-F-008 | Pass | N/A | |
| SE-F-009 | Pass | N/A | |
| SE-F-010 | Pass | N/A | |
| SE-F-011 | Pass | N/A | |
| SE-F-012 | Pass | N/A | |
| SE-F-013 | Pass | N/A | |
| SE-F-014 | Pass | N/A | |
| SE-F-015 | Pass | N/A | |
| SE-F-016 | Pass | N/A | |
| SE-F-017 | Pass | N/A | |
| SE-F-018 | Pass | N/A | |
| SE-F-019 | Pass | N/A | |
| SE-F-020 | Pass | N/A | |
| SE-F-021 | Pass | N/A | |
| SE-F-022 | Pass | N/A | |
| SE-F-023 | Pass | N/A | |
| SE-F-024 | Pass | N/A | |
| SE-F-025 | N/A | Not a required test. Not tested | |
| SE-F-026 | Pass | N/A | |
| SE-F-027 | Pass | N/A | |
| SE-F-028 | Pass | N/A | |

| Test ID | Pass / Fail | Fail description | CCF / issue # |
|---|---|---|---|
| SE-F-029 | Pass | N/A | |
| SE-F-030 | Fail | Future maintenance task | 6 |
| SE-F-031 | Pass | N/A | |
| SE-F-032 | Pass | N/A | |
| SE-F-033 | N/A | Not a functional requirement. Not tested | |
| SE-F-034 | Pass | N/A | |
| SE-F-035 | Pass | N/A | |
| SE-F-036 | Pass | N/A | |
| SE-F-037 | Pass | N/A | |