

---

# Group Project 07 Final Report

---

*Authors:* Mosopefoluwa David Adejumo  
Ryan Gouldsmith  
Harry Flynn Buckley  
Zack Lott  
Mark Radcliffe Pitman  
Jack Alexander Reeve  
Mark Alexander Smith  
Martin Vasilev Zokov  
Maciej Wojciech Dobrzanski

*Config ref:* SE\_07\_FR\_01

*Date* February 17, 2014

*Version* 2.2

*Status* Draft

Department of Computer Science  
Aberystwyth University  
Aberystwyth  
Ceredigion  
SY23 3DB  
Copyright ©  
Aberystwyth University 2013

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Scope . . . . .	4
1.3	Objective . . . . .	4
<b>2</b>	<b>THE END-OF-PROJECT REPORT</b>	<b>5</b>
2.1	Management . . . . .	5
2.2	History . . . . .	7
2.3	Final State . . . . .	10
2.4	Member Evaluation . . . . .	12
2.4.1	Mosopefoluwa David Adejumo . . . . .	12
2.4.2	Ryan Gouldsmith . . . . .	12
2.4.3	Harry Flynn Buckley . . . . .	12
2.4.4	Martin Vasilev Zokov . . . . .	13
2.4.5	Mark Alexander Smith . . . . .	13
2.4.6	Jack Alexander Reeve . . . . .	13
2.4.7	Mark Radcliffe Pitman . . . . .	13
2.4.8	Zack Lott . . . . .	14
2.4.9	Maciej Wojciech Dobrzanski . . . . .	14
2.5	Team Evaluation . . . . .	15
<b>3</b>	<b>APPENDICES</b>	<b>16</b>
3.1	The Project Test Report . . . . .	16
3.2	The Project Maintenance Manual . . . . .	18
3.2.1	Android Maintenance . . . . .	18
3.2.1.1	Program Description . . . . .	18
3.2.1.2	Program Structure . . . . .	18
3.2.1.3	Significant Algorithms . . . . .	20
3.2.1.4	Files . . . . .	23
3.2.1.5	Interfaces and Significant Algorithms . . . . .	23
3.2.1.6	Change Hazards . . . . .	24
3.2.1.7	Possible Improvements . . . . .	24
3.2.1.8	Physical Limitations . . . . .	24
3.2.1.9	Rebuilding and Testing . . . . .	24
3.2.2	Web Application Maintenance . . . . .	26
3.2.2.1	Program Description . . . . .	26
3.2.2.2	Algorithms . . . . .	26
3.2.2.3	Main Data Areas . . . . .	26

3.2.2.4	Files . . . . .	27
3.2.2.5	Interfaces . . . . .	27
3.2.2.6	Suggestions For Improvements . . . . .	27
3.2.2.7	Things To Watch Out For When Making Changes .	27
3.2.2.8	Physical Limitations Of The Program . . . . .	27
3.2.2.9	Rebuilding And Testing . . . . .	28
3.2.3	Database Maintenance . . . . .	29
3.2.3.1	Program Description . . . . .	29
3.2.3.2	Program Structure . . . . .	29
3.2.3.3	Algorithms . . . . .	31
3.2.3.4	Main Data Areas . . . . .	31
3.2.3.5	Files . . . . .	31
3.2.3.6	Interfaces . . . . .	31
3.2.3.7	Suggestions For Improvement . . . . .	31
3.2.3.8	Things To Watch Out For . . . . .	31
3.2.3.9	Physical Limitation . . . . .	32
3.2.3.10	Rebuilding And Testing . . . . .	32
3.3	Personal Reflective Reports . . . . .	33
3.3.1	Martin Vasilev Zokov . . . . .	33
3.3.2	Ryan Gouldsmith . . . . .	33
3.3.3	Zack Lott . . . . .	35
3.3.4	Jack Alexander Reeve . . . . .	36
3.3.5	Mosopefoluwa David Adejumo . . . . .	37
3.3.6	Mark Radcliffe Pitman . . . . .	38
3.3.7	Mark Alexander Smith . . . . .	39
3.3.8	Harry Flynn Buckley . . . . .	40
3.3.9	Maciej Wojciech Dobrzanski . . . . .	41
3.4	Revised Project Plan . . . . .	43
3.5	Revised Design Document . . . . .	78
<b>4</b>	<b>REFERENCES</b>	<b>110</b>
<b>5</b>	<b>DOCUMENT HISTORY</b>	<b>111</b>

# **1 INTRODUCTION**

## **1.1 Purpose**

This document contains a report of all the events and status of the project and is a compilation of all the documents used

## **1.2 Scope**

This document should be read by all members of the group for approval. It contains an overall evaluation of every group member and the details of the project, it's final state with details of all the events of the project

## **1.3 Objective**

- Provide a historical account of events during the project
- Provide a report on how all the members performed and the team
- Provide a report from every member about how they feel they performed
- Provide a compilation of all documents used for development and planning
- Provide details on how to maintain the project for future use

## 2 THE END-OF-PROJECT REPORT

### 2.1 Management

The program satisfied most of the basic requirements. On the website, all the requirements were met and the user is able to view walks and their associated information including images, points of interest and the path of a walk. However on the Android, while the user is able to upload and record walks as well as edit the name of the walk and its descriptions, the Android application had a bug where it would crash after uploading a walk instead of returning to the home screen. This bug was identified before submission, however, a fix was found after the submission deadline, and thus not implemented in the final product. Users were also unable to edit or remove individual points of interest once created and thus to remove a point of interest, the walk had to be restarted.

There was also heavy feature creep much of which was removed from the final product. The feature creep resulted in time wasted on features that were cut. One main issue was displaying a map on the Android this was going to be used to display points of interest and allow editing of points of interest, but was cut due to difficulties in implementation. There was also an issue where the development on the website stagnated. This lack of progress led to the website being off schedule. Extra work was performed by the QA Manager to bring us back on schedule. A third issue was issues with decoding the JSON data on the server. To allow us to see what data was being sent, we created a log file and used it to assist in the coding of the decoder. Images were also converted to a string and decoded using base64.

Overall, the team performed well, however the lack of progress on the web team caused issues late in development and the tasks had to be reorganized. The Android programmers performed exceptionally well, remaining on schedule. However, the manager spent times focusing on several areas particularly during coding week and so several members would occasionally cease to do work, having not been assigned a task. There were issues with Github not tracking the number of commits made by members, despite showing these commits in the log.

Below is an example of a weekly report. This report is a summary of work done as the actual reports provided by the members were more detailed

Group Project 07 – Weekly Report

(Release) -Version 1.1

**1.2. 5 November 2013 – 11 November 2013**

Task name	User ID	Action	Time
Familiarisation	JAR39	HTML5 familiarisation	Tuesday, 2 hrs
	MRP2		Sunday, 1 hr
Prototyping	MAS69	Creation of database	Friday, 2hrs
		Storage of data in the database (incomplete)	Sunday, 2 hrs
	MVZ	Started Android prototyping	Saturday, 30 mins
	HFB1	Began android coding and prototyping	Thursday, 3 hrs 30 mins
	MDA	Assigned tasks for upcoming week	Wednesday, 40 mins
		Updated and assigned new tasks for the week	Monday, 20 mins
Test specification	MVZ	Added tests to test spec	Tuesday, 30 mins
	RYG1	Updated and edited test specification	Wednesday, 1 hr
	HFB1	Updated and edited test specification	Saturday, 30 mins
	MDA	Reviewed test specification and Class diagram from HFB1	Friday, 1 hr
		Reviewed test specification and requested clarification on tests	Sunday, 1 hr
Design specification	MWD5	Edited and discussed class diagram with HFB1	Thursday, 1 hr
		Design logo and application name	Sunday, 1 hr
		Edited and updated class diagram	Monday, 1 hr
	HFB1	Wrote notes on class diagram	Wednesday, 2 hrs
		Creation of new class diagram	
		Worked on class diagram	Thursday, 2 hrs
		Transferred UML diagrams	Friday, 1 hr
Project plan	MDA	Continued conversion of class diagram	Monday
		Edited and updated gantt chart	Tuesday, 20 mins
		Renamed images from project plan to meet standards	Saturday, 1 hr
Other tasks		Reviewed progress made	Monday, 30 mins
	JAR39	CSS creation	Thursday, 1 hr
	ZAL	Research into android and editing and updating the test specification	
	RYG1	Created minutes from meeting	Tuesday, 30 mins
			Thursday, 30 mins

Hours on Prototyping                      9 hours  
 Hours on Test specification                4 hours  
 Hours on Design Specification            8 hours  
 Hours on Project Plan                      1 hour 20 minutes  
 Hours on Familiarisation                  3 hours

**On Schedule**

Page 4 of 5

Aberystwyth University / Computer Science

Figure 1: Example of a weekly report.

## 2.2 History

At the start of the project, the group were all required to produce reports of all work performed during the week which was usually a summary of the blog. These reports were submitted to the project manager to ensure the project remained on schedule. Planning began immediately and the project manager made several revisions to the Gantt chart and one of the members requested a change of roles which was included in the Gantt chart. During the design phase we wanted the walks to record actual user movement and not a direct point to point depiction of the walk. This meant storing more coordinate information in the database. We used the term Point of Interest (POI) to refer to locations where the user added a description or images. These points of interest would be where markers are displayed on the website, with locations simply being coordinates. I.e. A Location is a set of coordinates while a point of interest is a location with more info such as a description and images.

The project plan was produced during a meeting in which all members suggested the initial design of the system and their understanding of the requirements. MDA typed the document and used images created by MWD5 and JAR39. Subsequent documents were produced as a contributed effort where different members worked on their specialized areas. I.e the web programmers worked on the website aspect of the design specification, while the android programmers worked on the Android aspect. The class diagram was produced by HFB1 and approved by MDA. There was also a lack of Android devices in the group particularly among the Android programmers, and thus three devices were purchased due to the limitations of the emulator particularly with GPS coordinates.

During the process of the project, meetings were held twice weekly, one on Tuesday's with the Project Supervisor and a second on Fridays to discuss work during the week and goals for the next Tuesday meeting. These meetings helped identify and eliminate feature creep in some areas, but also resulted in the addition of feature creep in other areas. These meetings also were used to discuss changes in design before coding. One such instance was the use of the OpenSpace API. It was during a meeting that one of the Android programmers MVZ notified the project manager of the inefficiencies of the API and this resulted in the switch from OpenSpace to Google Maps API.

During prototyping, several issues were encountered but allowed certain design decisions to be made early. Production of the website prototype had several issues. These included the lack of any dynamic code (PHP). However there was a map present in the website, and an idea of the behaviour of the website was

given through the prototype. The Android progressed further during prototyping in which core features including basic navigation through screens and uploading text data were operational. However, the prototype did contain features which were removed later during development. The web based prototype was hosted on MVZ's public html. It was not hosted on the web programmers filestore because of an assignment for a web module as we wanted to avoid the possibility of modifying the permissions of the web assignment files thereby causing a late submission.

Creation of the database also proved difficult, due to the decision to host it on the universities network. There were issues with creating the database but after enquiries by MAS69, the database was created but didn't have any data. Due to an ongoing assignment, several group members were unable to focus on the group project. The project manager, who didn't have any required assignments at the time, created a web based interface which was used to create and delete tables with ease. Over the Christmas holiday, RYG1 created a simple interface for manually adding information in the database as well as modifying a file which displayed all the information in the database. These scripts were later used extensively for testing. However by the start of the second semester, the database tables were not being joined correctly.

During the Christmas holiday, RYG1 worked extensively on the website producing the basic files for the website, however the server still had no means of putting data in the database tables. The Android development also proceeded at a fair pace during this time, with the walk recorder being implemented. However, image uploading was not functional at the time.

After completion of the exams, a meeting was held to discuss plans for Integration and Coding Week and all progress made over the Christmas holiday. Integration and coding week saw all group members working from 9am to about 5pm or 6pm. The first day of the week saw the implementation of uploading basic walk information and elimination of all feature creep that had arisen during the planning stage. There were difficulties in reading the JSON data on the server and this led to the creation of a log file which proved useful for further development. By Tuesday, the website still was not functional to an acceptable state however the web programmers had developed code which turned out to only be functional for hard coded data. In order to increase the website development, the project manager began working on the file\_saver.php file which handled the data being received from the Android device. MAS69 assisted MDA in finding a tutorial on how to extract variables from the JSON data.



RYG1 was put on full website development and the website was moved to MDA's public html as project leader, while MWD5 began producing JUnit tests while ZAL also produced PHPUnit tests. Initially these were to be used for the file\_saver.php file, but lack of knowledge led to the removal of several tests. The project manager worked extensively on the file\_saver.php file on Tuesday and Wednesday and was able to add information to the database, however joining the tables was solved by RYG1. On the Android a conflict arose where the Android programmers had mapped the same button to different functions. While this initially was not seen as a major issue, MVZ was instructed to create a screen to override the button conflict. This presented an issue where the buttons on the screen were not functioning. This was solved by the project manager who passed the variables as required and this fix in turn solved another similar bug that had appeared elsewhere. By Wednesday, RYG1 had to refactor all the website code to bring in correct functionality and thus the remaining members of the web team were moved into production of CSS files and testing the walks.

By Thursday, at the suggestion of the QA manager, the map being displayed on the Android was removed, due to group vote, although the project manager did not agree with such a move, it did help the project remain on schedule. By early morning, the website became fully operational, with the project manager displaying multiple images on the website when a point of interest is located. By this time, the Android satisfied most of the requirements, however during late testing a bug was discovered which caused the application to crash on upload. Friday morning was spent doing extensive bug fixing, testing and commenting code with Javadoc and PHP doc. The upload crash bug was partially solved, but the application began crashing after uploading. The project manager put a work deadline 15 minutes to the final deadline by which all work would stop. On submission, the group produced very little work during the next week, but the final documentation was completed shortly after a meeting on the week prior to the submission deadline.

A major issue was misunderstanding the definition of MIME type and all images were converted to a string to decode. The image decoding had problems as decoders were unable to decode the text data. HFB1 used a text editor to remove backslashes which solved the problem so code was added to file\_saver.php to remove the escape slashes added by JSON.

By the final week of documentation, development stagnated. MDA and RYG1 had to put in extra effort to see to the completion of the final document. However other members did contribute though not to the level of MDA and RYG1.

## 2.3 Final State

On the website, the project has been completed and the only known flaw was the map not displaying correctly on certain browsers. The website was tested on the following: Safari, Opera, Firefox, Internet Explorer, Google Chrome, Mobile Opera (Android), Android default browser and Mobile Safari (iPad). The following browsers had issues: Firefox: the map was displayed out of place on older versions, Android default browser: Did not display the map. All other features worked correctly, Internet Explorer: Some versions did not display the map, other versions displayed it out of place.

The final website can be found here <http://users.aber.ac.uk/mda/csgp07/wtc>  
all other server based files including database testing files can be found here <http://users.aber.ac.uk/mda/csgp07>  
(Login for database testing- Username: grp7admin Password: Wc7gp)

The database had constraints applied to its data as required and walks were stored correctly on the website. The Android device crashed when uploading a large number of images or after a completed upload. The crash was possibly caused by the recorder only stopping the recording on completion of the upload. However the device iterates through the array of locations during upload. If there is a GPS signal during upload, the crash occurs due to data being added to the array during iteration through said array. The application also crashed when attempting to upload without a connection to the internet, however the crash may have been caused by the aforementioned bug and we were unable to isolate the cause.

The user is also only able to edit the overall walk details: name, long and short descriptions. Individual points of interest cannot be edited once created. There is no user feedback on edit completion, however there is feedback when recording starts. The recording may also stop after 15 minutes of inactivity due to the Android device terminating the service. This is probably caused by the method used to implement the service. On successful uploads, the user is returned to the start screen however, due to the bug discovered late, the application may crash instead of returning the user to the start screen.

There may be several grammatical issues as of yet unidentified in the program. The GPS may also cut out leading to erratic lines when viewing a walk. This is a device and GPS issue and cannot be resolved by the application. Places, Locations and photo ID's are acquired by getting the ID of the last entry into the database, as a result, it may be possible to have details of a walk linked incorrectly if two or more walks are uploaded and processed within fractions of seconds from one

another. However, we could not test this out and cannot verify as to whether or not it can occur. Any other errors have not been identified by the project members.

There were several website files submitted which were not our own work. These files were used solely for early designing and prototyping. The files that were not created by us are:

- all font files
- all external javascript files
- all files in the shadowbox-3.0.3 folder
- all images in the Website/images folder
- lightbox.css

## 2.4 Member Evaluation

### 2.4.1 Mosopefoluwa David Adejumo

*Mosopefoluwa David Adejumo - MDA: Project Leader*

MDA requested the creation of weekly reports to monitor group data and handed out tasks as well as oversaw the development of the project. Several aspects of feature creep were not identified or accounted for early on in planning which affected late development. Also occasionally put a lot of focus into single parts of the project resulting in slowed development in others. Development was brought back on schedule via task reorganisation. Also worked on the file\_saver.php file as well as displaying multiple images on the browser and minor bug fixing on the Android. Would send out emails regularly to group members requesting tasks be done.

### 2.4.2 Ryan Gouldsmith

*Ryan Gouldsmith - RYG1: Quality Assurance Manager*

Oversaw the maintenance of standards in quality assurance. Assisted the project manager in overseeing the web programmers in development as well as performing extensive work on the website. Also worked on the Test Specification producing a large number of tests to be done. The QA performed well doing web development to keep the project on schedule and was easy to make contact with for the duration of the project. Also produced minutes for all the meetings when required. Reviewed the documents as required. Excellent performance throughout development.

### 2.4.3 Harry Flynn Buckley

*Harry Flynn Buckley - HFB1: Android Programmer*

Worked on Android coding. Developed the first prototype and worked on image integration of the Android application with both the camera and the library. Also integrated the uploading of the device and storing walks. Towards the end of coding week, he put a lot of effort into development and bug fixing in order to get the program to an acceptable state despite setbacks. Performed excellently throughout the project, and remained very active in keeping the project manager up to date with progress

#### 2.4.4 Martin Vasilev Zokov

*Martin Vasilev Zokov - MVZ: Android Programmer*

Worked on Android coding. Implemented the recording of walks as well as some GUI screens. Coded the algorithms for adding points of interest and recording the walk as well as GPS integration and recording checking. After completion of tasks, worked on bug fixing and Javadocs. Excellent performance throughout the project. Although there was an occasional loss of communication, work continued as expected keeping the project on schedule.

#### 2.4.5 Mark Alexander Smith

*Mark Alexander Smith - MAS69: Web Programmer*

Saw the creation of the database and provided assistance in decoding the JSON data. Also worked on the base template of CSS in the final product and performed extensive testing. One particular field test proved extremely useful in development. Occassinally slowed down development wise, but would quickly pick up, keeping the project on schedule. Great performance throughout the project and would perform tasks in a timely manner.

#### 2.4.6 Jack Alexander Reeve

*Jack Alexander Reeve - JAR39: Web Programmer*

Produced early prototypes of the website. Also produced the initial drawing of walks on maps between points of interest. Worked on CSS as well as website fixing and testing on different browsers. Also worked on the list walks.php file. Did lots of tidying up of the website, particularly towards the deadline. Excellent performance and produced most items on time. Difficult to reach between the start of the Christmas holidays and the semester starting week, but quickly made up for lost work.

#### 2.4.7 Mark Radcliffe Pitman

*Mark Radcliffe Pitman - MRP2: Web Programmer*

Initially slow performance, and occasionally difficult to contact, worked on CSS of the final site as well as creation of the terms of service page. Picked up performance after the first day of coding week. Would perform tasks as soon as requested. Worked with other web programmers to produce CSS for the list walks page and

also produced an early prototype for login page and worked on the website maps API. However this was not used as accounts were removed from the final project. Great performance during coding week. Also produced some PHP Unit tests when requested. Also worked on testing the website.

#### **2.4.8 Zack Lott**

*Zack Lott - ZAL: Web Tester*

Worked on the test specification and also produced a test report and worked on testing the website to ensure it met all the requirements. Also produced some PHP Unit tests which. Occasionally slowed down development during coding week. However this was due to lack of technical knowledge of PHP Unit testing. Picked up performance, once PHP Unit was learnt. Also did extensive field testing of the Android and testing of the website. Great performance particularly during coding week.

#### **2.4.9 Maciej Wojciech Dobrzanski**

*Maciej Wojciech Dobrzanski - MWD5: Android Tester*

Worked on the test specification. Was initially working solely on documents but requested to be moved to testing, to increase involvement in the group project. Was particularly eager to contribute and offered assistance where necessary or requested tasks to perform. Produced extensive JUnit tests and extensive Android testing and identification of bugs. Performed well and completed tasks in a timely manner throughout the project. Also produced several UML diagrams for the project.

## 2.5 Team Evaluation

As a team there were occasional issues mainly due to miscommunication or lack of communication. However, when all members were present and actively working together, development would progress rapidly as conflicts both program wise and personality wise were easily resolved. Consistent reports helped keep the project on schedule. However, communication could have been improved as there were issues that members had that the project leader would be unaware of, but would be notified eventually, usually during a meeting where the project supervisor was present.

The team members collaborated with each other well, preventing an overlap of tasks and allowing development to proceed smoothly as well as helping keep feature creep out of the final product. If things were to improve, it would be to improve communication as that was the main issue with the group. Contacting each other was rarely the issue, it was mainly due to misinterpreting statements and sentences, possibly due to cultural differences of the members.

Tasks could also be monitored and better allocated to a more precise level and to take advantage of the strengths and weaknesses of each of the group members, with members sorting out issues with the affected members as needed.

The project also had issues with understanding the requirements and identifying feature creep. There was a large amount of feature creep initially and if these had been identified and removed earlier on, there may have been more time for bug fixing, and the Android may have been able to fully implement the requirements. Another issue was understanding and interpreting the requirements. Better contact or communication with the client should have been taken. This would have helped keep things to the point as the final product would better match what the client wanted as the specification given was vague in certain areas. This led to development of features or planning of features which were not required and thus time and spent or wasted performing those tasks.

It is important for members to work closely together and maintain communication and collaboration with each other. A project must be managed to a precise level and all problems should be taken up and sorted out with the appropriate person, be they the client or another member of the team. Also, in order to ensure a project is completed on time, the manager should assign tasks based on the feedback of the team members and the technical knowledge of such tasks. With a team it is fairly easy to run into conflicts in development due to poor communication and these can in turn have a greatly negative impact on the final product.

### 3 APPENDICES

#### 3.1 The Project Test Report

Test ID	Pass / Fail	Fail description	CCF / issue #
SE-F-001	Pass		
SE-F-002	Pass		
SE-F-003	Pass		
SE-F-004	Pass		
SE-F-005	Pass		
SE-F-006	Pass	There is no output message, but the app returns to the main menu, which clearly indicates that the walk is saved	
SE-F-007	Pass		
SE-F-008	Pass		
SE-F-009	Pass		
SE-F-010	Pass		
SE-F-011	Pass		
SE-F-012	Pass		
SE-F-013	Pass		
SE-F-014	Pass		
SE-F-015	Pass		
SE-F-016	Pass		
SE-F-017	Pass		
SE-F-018	Pass		
SE-F-019	Pass		
SE-F-020	Pass		
SE-F-021	Pass		
SE-F-022	Fail	Cannot remove the POI	#38
SE-F-023	Fail	No POI counter	#39
SE-F-024	Fail	Cannot remove POI	#41
SE-F-025	Pass		
SE-F-026	Pass		
SE-F-027	Pass		
SE-F-028	Fail	The application screen does not show the time stamp, it is visible on the website though.	#42
SE-F-029	Pass		
SE-F-030	Pass		
SE-F-031	Pass	Cannot check the image format on Android. Images are saved with the JPEG extension	



Test ID	Pass / Fail	Fail description	CCF / issue #
SE-F-032	Pass		
SE-F-032	Pass		
SE-F-034	Pass		
SE-F-035	Fail	Cannot view POI co-ordinates on the map pop up, on the website. But it does use the co-ordinates to map the POI	#53
SE-F-036	Pass		
SE-F-037	Fail	No 404 Page	#54
SE-F-038	Pass		
SE-F-039	Pass	Pages did not validate to HTML5. Some external scripts do not validate. CSS does not validate due to browser specific code	
SE-F-040	Pass		
SE-F-041	Pass		
SE-F-042	Pass		
SE-F-043	Pass		
SE-F-044	Pass		
SE-F-045	Pass		
SE-F-046	Pass		
SE-F-047	Pass		
SE-F-048	Pass		
SE-F-049	Fail	Doesn't inform the user if the information is corrupt or not	#55
SE-F-050	Pass		
SE-F-051	Pass		
SE-F-052	Pass		
SE-F-053	Pass		
SE-F-054	Pass		
SE-F-055	Pass		
SE-F-056	Pass		
SE-F-057	Pass		
SE-F-058	Pass		

## 3.2 The Project Maintenance Manual

### 3.2.1 Android Maintenance

#### 3.2.1.1 Program Description

The function of this program is to allow the user to document walks. The path of the walk is tracked by the GPS of the device and by tracking the GPS of the device and providing the user with the means add places of interest.

#### 3.2.1.2 Program Structure

The program is organised in to Android Activities classes that control the application until a new Activity is started.

The initial activity that the user sees is the StartScreen, that presents the user with a button prompting the start of a new walk, this button will start a new activity that creates the walk.

WalkSetupScreen allows the user to enter basic walk data such as title and description. This input is used to create a new walk. The majority of functionality is dealt with by the WalkScreen, it is responsible for providing the user with options about the walk. From here places of interest can be created and added to the current walk.

**Walk module-** This module handles all data about a walk - the route that was taken during the walk, the points of interest that were added and the information for the walk itself.

**Classes:** WalkModel, LocationPoint, PointOfInterest

WalkModel stores the details of a walk. it uses three string variables for the title, short description and long description. It also keeps all the route data into a Vector of LocationPoint objects. The WalkModel object also calculates the distance traveled and the time spent on a walk.

LocationPoint objects are created when GPS data is received and the longitude and latitude are stored as well as a time stamp for the location.

PointOfInterest objects are inherited from the LocationPoint class and whenever a new POI is added. These objects have a Vector of ImageInformation objects that were added from either the camera or the gallery. PointOfInterest objects also have a title and a description. This information is added in the pop up dialog which is shown when the user adds a new PointOfInterest. The exact coordinates for the object are taken from the last known position, which is recorded in the

RouteRecorder class.

### **Recorder module**

This module handles the recording and storing of route data.

**Classes:** RouteRecorder, PositionListener

RouteRecorder creates the LocationManager (Android API) and gives it a PositionListener object to respond to received GPS data. The listener then calls the newLocation method from the RouteRecorder to add it to the Vector of Location-Points.

The RouteRecorder keeps track of the last known position which is used when adding new points of interest.

### **Image module**

This module is responsible for adding images to a walk. It is used to either get an image from the camera or select an image from the gallery on the device.

**Classes:** ImageHandler, ImageInformation

ImageHandler uses two result codes to choose whether to start an activity to pick an image from the gallery or take a new picture.

ImageInformation is used to store information for a single image. It is mainly used to encode the image information as a string in Base64 encoding and is then sent to the server.

### **File Transfer module**

This module is responsible for sending

**Classes:** FileTransferManager

FileTransferManager is responsible for packaging all the data for the current walk into a single JSON object that is then sent as an HTTP POST to a web server. It also has an inner class which is responsible for the upload. It starts in a new Thread in which the upload is handled.

### **Popup dialogues module**

This module is used to display popup dialogues for various.

**Classes:** DialogView, CancelWalkView, EditWalkView, GpsCheckDialog, PoiDialogView, WalkFinishedView

All of these classes are essentially the same. They use the abstract class DialogView

to create a default pop up window and show different layouts depending on the situation. Each of the popup classes uses a different layout that is defined in an XML file and displayed in the pop up. There is a listener which has a different implementation in each of the classes to respond to the button clicks in different ways.

CancelWalkView brings up a warning message that this action will delete all information about the walk.

EditWalkView gives options to set a different title, short and long descriptions for the current walk. A future improvement could be displaying the current title and descriptions for the walk.

GpsCheckDialog is displayed if the GPS on the device is not turned on and gives the user an option to go to the GPS settings and turn it on.

WalkFinishedView gives the option to stop the current walk, upload it and go back to the main screen of the application.

### 3.2.1.3 Significant Algorithms

The methods below open up new screens

```
public void finishWalk(View v){}
```

```
public void editWalkDialog(View v){}
```

```
public void cancelWalk(View v){}
```

The below methods are related to the **Walk Model**:

```
public WalkModel() {}
```

This is the default constructor for the WalkModel that creates the Vector for the route.

```
public String getTitle(){}
```

This gets the title of the walk

```
public void setTitle(String newTitle){}
```

This sets the title of the walk

```
public String getShortDescription(){}
```

This gets the short description of the walk

```
public void setShortDescription(String newShortDesc){}
```

This sets the short description of the walk

```
public String getLongDescription(){} 
```

This gets the long description of the walk

```
public void setLongDescription(String newLongDesc) {}
```

This sets the long description of the walk

```
public Vector <LocationPoint >getRoutePath(){} 
```

This gets the Vector of LocationPoint objects

```
public void addLocation(LocationPoint point){}
```

This adds a new location to the Vector

```
public double getDistance(){} 
```

This calculates the distance between the start and end of the walk

```
public double getTimeTaken(){} 
```

This calculates the total time spent on a walk.

The below methods are for **Location Point**:

```
public LocationPoint(double lat, double lng){}
```

A constructor for the LocationPoint. Automatically takes a time-stamp.

```
public long getTime() {}
```

Gets the time-stamp of the point.

```
public double getLongitude() {}
```

Gets the longitude of the point

```
public double getLatitude() {}
```

Gets the latitude of the point

```
public static double distBetween(LocationPoint point, LocationPoint point2) {}
```

Calculates the distance between two locations.

The below methods are for **Point Of Interest**

```
public PointOfInterest(LocationPoint point) {}
```

Constructor for the PointOfInterest. Uses the super-constructor from the LocationPoint class

```
public void addImage(ImageInformation newImage) {}
```

adds an image to the Vector of images

```
public Vector <ImageInformation >getImages() {}
```

Gets the Vector of images

```
public String getDescription() {}
```

Gets the description of the PointOfInterest

```
public void setDescription(String desc) {}
```

Sets the description of the PointOfInterest

```
public String getTitle() {}
```

Gets the title of the PointOfInterest

```
public void setTitle(String title) {}
```

sets the title of the PointOfInterest

The below methods are for **Route Recorder**

```
public RouteRecorder(WalkScreen screen, WalkModel walk, LocationManager manager){}
```

Constructor for the class. Creates a PositionListener and registers it with the passed LocationManager

```
private void setLastKnownPosition(LocationPoint newPoint){} {}
```

sets the last known position (the last received GPS location)

```
public LocationPoint getLastKnownPosition() {}
```

returns the last known position

```
public void newLocation(LocationPoint loc) {}
```

adds a new location to the current walk's Vector of LocationPoints

```
public void finishWalk() {}
```

Stops the GPS updates for the LocationManager and makes it null

Below is the main algorithms for the **Image Module**

```
public ImageHandler(WalkScreen owner) {}
```

Constructor for the ImageHandler object

```
public void getPhotoFromLibrary() {}
```

Starts an activity to get a photo from the gallery

```
public void getPhotoFromCamera() {}
```

Starts an activity to take a picture with the camera

```
private File createFile() {}
```

Creates the file that the camera will save the taken picture to. The file name is just the current system time in milliseconds.

```
public ImageInformation(String filename) {}
```

Constructor for the ImageInformation object

```
public String getImageAsString() {}
```

Gets the encoded image as a string on Base64 encoding

```
public FileTransferManager(WalkScreen screen, WalkModel w) {}
```

Constructor for the FileTransferManager

```
private static JSONObject packageData(WalkModel data) {}
```

Packages all the WalkModel as a JSON object

```
private static JSONArray packagepathData(Vector <LocationPoint >route) throws  
JSONException {}
```

Packages the whole route as a JSON object

```
private static boolean post(JSONObject packagedData, WalkScreen walk) {}
```

Sends the data as a POST message to the server

Any further Algorithms are mentioned in the Design specification.

#### **3.2.1.4 Files**

[1] Image for map background

#### **3.2.1.5 Interfaces and Significant Algorithms**

Route recording algorithm See Design Specification SE\_07\_DS, Section 4

### 3.2.1.6 Change Hazards

If changing JSON variables, ensure the PHP files `file_saver.php` has been updated with the new variables. Make sure the URL is correctly assigned. The recorder image must be present or the application will crash. For new Popup dialogues be sure to pass the layout to the superclass for inflation and get the inflated layout from the superclass (`super(context, myLayout); this.layout = this.getlayout();`) failure to do so may prevent buttons from working.

### 3.2.1.7 Possible Improvements

Fix crash on upload bug

Edit individual points of interest

An improvement for future versions could be a better algorithm for managing GPS usage, GPS frequency and an algorithm for selecting which of the received locations can be discarded based on the accuracy of the received coordinates and other factors like user standing in one spot for an extended period of time.

### 3.2.1.8 Physical Limitations

Number of images that can be uploaded is dependent on the amount of RAM on the device

Requires a device with inbuilt GPS capability. Cannot use cell network or WIFI for location finding

There is a known bug in the upload method which some times causes a crash when uploading. This is due to the fact that while uploading the GPS data is still being received and written to the Vector of LocationPoints. This can be fixed by having the route Vector copied into a local variable and the loop for the upload would iterate through the local variable instead of the one that is passed to the method.

Also a crash which was identified as being caused by a null pointer can be fixed by removing line 164 of WalkScreen as this function is also called on line 176 (`recorder.finishWalk();`).

### 3.2.1.9 Rebuilding and Testing

JUnit test provided

Built and tested on Android 4.2.1 and 4.2.2

Ensure required images are available



If the user wishes to edit this document then they will need to install a LaTeX editor, and compile and build the document based on the software's documentation on how to do so. If the user wishes to add a new test, because of a fault then all they need to add the tests at the end of the test document by entering a test number, requirement, Test Content, Input, Output and a pass Criteria. They can do this in LaTeX by adding the & symbol between each text data column - they can then finish it off by adding \\ to the end of the information - finally add a \hline and this will complete the table.

### **3.2.2 Web Application Maintenance**

#### **3.2.2.1 Program Description**

The walk\_details.php file is used to show a walk in details. It uses Google Maps API to show the path of the walk and markers to show a point of interest. This file also shows the images from the selected walk. The images can be viewed in a lightbox.

On the list of walks page, all the walks that have been uploaded from the Android application show up. This is done by connecting to the database and retrieving all the walk details and echoing them onto the page. The user can select walk he would like to view by clicking on said walk. This will redirect him to the walk\_details.php sending a GET request containing the walk ID which will show more detail about the walk. The list is responsive to the size of the browser window. The bigger the window, the more columns there are. This is done via CSS by making the information that is echoed float to the left.

#### **3.2.2.2 Algorithms**

In the walk\_details.php file, we use \$\_GET to get the correct 'walk' and 'walk\_id' from the MySQL database. This is needed so that we can use a query which pulls just the data we want. This is also needed to grab the photos which were taken at each specific POI on the walk. The images are pulled using a loop which prints an image via an absolute url and a php array.

We use JSON to encode each of the variables (e.g. \$lat, \$long, \$title) these can then be put into the corresponding JavaScript arrays. These can then be displayed on the Google Maps.

The list\_walks.php, first connects to MY SQL database while using database\_layer.php to sanitise the queries and then echoes out all the walks. This is done by \$\_GET which receives title, id, walk title and short description. Once that is all done, it closes the connection to the database.

#### **3.2.2.3 Main Data Areas**

The main data for the walk\_details page is held in PHP array variables when they are initially pulled from the database. This data then gets used to display map points on the Google Maps API.

This page only receives important information that is needed for this page. The way it has been laid out is with the title shows up more clearly than others cause that what the user is looking for.

#### **3.2.2.4 Files**

All CSS and external JavaScript files are required for some features.

#### **3.2.2.5 Interfaces**

This page will need permissions set to allow the page on the server.

The following files can be used to manually add data into the database and view all data in the database for testing: `list_table.php`, `field_form.php` and `add_fields.php`

##### List Tables

This file displays all data in the tables without filters and also displays all the tables created along with the table names and column names.

##### Field Forms

Field forms allows the developer to add information like name, description, etc to the database. This is where the information that is entered will be sent to the database. It can also be used for testing the database constraints.

##### Add Fields

The reason we need to add test data is to see if everything is working. The data will be sent to the database and see if it shows up on the website. We can see if the website also works without the android. Also to see if the website can retrieve everything and show them properly on the page. If that does not work then there is something wrong with the database or the website is not retrieving.

#### **3.2.2.6 Suggestions For Improvements**

Being able to edit descriptions and titles would have been a good feature to have. The structure of the program allows for expansion and extra features to be added. I feel that this would be easy to add to the existing program structure. It would also be a good idea to allow walks to be deleted. This could be done with a simple MySQL query.

Having images show up on the list would have made it more appealing so users can see already what kind of walk it would be.

#### **3.2.2.7 Things To Watch Out For When Making Changes**

Ensure the database connection information is correct and the database language is in MySQL. Make sure all external files such as the JavaScript and CSS are in the correct directories

#### **3.2.2.8 Physical Limitations Of The Program**

Certain features require JavaScript to be enabled. If the user has disabled or blocked JavaScript, some features will not be visible. The website can potentially display several images which may use a large amount of bandwidth or data.

**3.2.2.9 Rebuilding And Testing**

When rebuilding, make sure the CSS files are in the same place so the pages are still directed to the part. Just make sure if the connection to the database is fine. Just test this by PHPunit. It will show you if you have any errors. Also ensure the database has the required tables and their dependencies.

### 3.2.3 Database Maintenance

#### 3.2.3.1 Program Description

For the application to work successfully, and have a link between the Android application and the website then a Database is needed for communication. For our implementation, we used a MySQL database. From the requirements specification we have got 4 database relations. The database in this application is there to store the completed walks from the Android Application: The user will complete a walk and send the required information to the server, which will deode it and input the information into the correct fields in the table. .

#### 3.2.3.2 Program Structure

For our database we have 4 relations these are are:

##### List Of Walks:

This is where the user's overall walk gets uploaded to. For each walk in this table is a completely separate walk - no two walks in this table are identical as it will always have a different ID Key. In the list of Walks there is an if to identify the

List Of Walks	
•id	Integer
•Title	VarChar(100)
•shortDesc	VarChar(100)
•longDesc	varChar(1000)
•hours	Float
•distance	Float

Figure 2: List Of Walks Table

walk, this the primary key and all values are Unique. Next we have a title of the walk as a text field. Then we have a Short description of as a text field too, this has to be less than a 100 characters in length. Another field we have is the long description, which has to be to be less than 1000 characters. Finally, both hours and distance are of type Float.

##### Location:

This table is where the user's co-ordinates, as either "background" co-ordinates are stored, or co-ordinates for the Point of interest during the walk. This table

Location	
• <u>id</u>	int
•WalkID	Int
•latitude	float
•longitude	float
•Timestamp	float

Figure 3: Location Table

uses id as a unique primary key which is auto incremented and is of type integer. Another field in this table is the WalkId which is a foreign key references to Id in the list of walks - this is a foreign key as you will be able to keep track of which locations appear according to each walk. We then have latitude, longitude and a timestamp all as float representations. When the Android application records it will record background locations, so expect to have lots of co-ordinates in the table, that are not necessarily points of interest.

### Place Description

This table stores the description and name of any Location (Point of Interest) during the walk. Each Walk will be uniquely identified by the locationID This

Place Description	
• <u>id</u>	int
•locationId	int
•name	VarChar(100)
•description	VarChar(500)

Figure 4: Place Description Table

table refers to any point of interest during the walk. In this table you have a primary key called Id, which auto-increments every time a walk is added and is of type Integer. Then in the table there is the locationId, this is a foreign key reference to the key id, in the Location table - this again, is a way of keeping track of which point of interest is associated with a location and the co-ordinates for the point of interest. There is then a name and description which are both text fields in the database. These store the name of the point of interest and the description about the point of interest.

### Photo Usage:

This is the table which stores the photos for any given Point Of Interest. The table will allow multiple image for one location to be held as it stores the id for

Photo Usage	
• <u>id</u>	int
• placeId	int
• photoName	VarChar(100)

Figure 5: Photo Usage Table

the point of interest In this table, we have a primary key Id for the photo which is auto incremented and uniquely identifies each photo in the table. PlaceID is a foreign key reference to the id in the place description table - again this is used to keep track of which photo is in which point of Interest. Finally, photoname stores the name of the photo minus the .jpg extension.

### 3.2.3.3 Algorithms

All algorithms for inserting into the database is in file\_saver.php: This section is in the web maintenance section. All algorithms for interacting with the database is in the list\_walks.php file, and walk\_details.php: These section is in the web maintenance section.

### 3.2.3.4 Main Data Areas

There are no substantial data storage areas in the database as the database itself is the storage area. However when we get information from the database it returns it as an array. But this is covered in the Web Maintenance document.

### 3.2.3.5 Files

There only needs to be one file in which the database is set up. This file is the C\_tables.php; as without this file the database will not be set up correctly. The user will then need to edit the Java source code to point the JSON Encoded values to the correct hosted area for the file\_saver.php which will then add the information to the database.

### 3.2.3.6 Interfaces

The Database required MYSQL to be installed on the server. To view the tables then there are additional PHP files in the directory structure.

### 3.2.3.7 Suggestions For Improvement

An improvement which could be that the user would use software called PH-Pmyadmin. This would help greatly with having a visual representation of the database, as well as seeing the data in the database. This may help the user see it, rather than continuously using our PHP script.

### 3.2.3.8 Things To Watch Out For

On our scripts we have added the Foreign Key References at the bottom of create table script, rather than under the actual value - so if the user is to edit this

information and change any key dependencies then make sure that the last item is syntactically correct. Additionally, make sure that user chmod 700 the files - even though they're just SQL scripts inside, the file is .php - so you may not be able to run the script if the permissions are not set correctly.

### 3.2.3.9 Physical Limitation

We think a physical limitation of our database is such: When the user uploads a walk, it retrieves the last ID key from the database and then increments it by one. Therefore, we feel that if a user tries to upload two walks simultaneously then we do not know what will happen. Thus, this leads to the potential issue of locking; where two users try to access the database at the same time.

### 3.2.3.10 Rebuilding And Testing

To rebuild our database tables you simply have to firstly created a database to put the tables in to, from then you have to edit the mysqli\_connect parameters to your own host, your own username and password and your selected database. Then to create the tables you will simply have to run the c\_tables.php file; this will create the database for you and all the relations and dependencies.

To test the Database we have written a small PHP script/web page which allows you to view the tables and what information you have in them at any given time. There is also an additional file called d\_tables.php - this will allow you to drop all the data from the tables. **NOTE:** any tables which have been dropped will then be deleted, you will then need to call c\_tables.php again, to create the tables.

All tests to do with the database are in the Test Specification Document, under requirement: FR9 in the document. If the user wishes to edit this document then they will need to install a LaTeX editor, and compile and build the document based on that software's documentation on how to do so. If the user wishes to add a new test, because of a fault then all they need to add the tests at the end of the test document by entering a test number, requirement, Test Content, Input, Output and a pass Criteria. They can do this in LaTeX by adding the & symbol between each text data column - they can then finish it off by adding \\ to the end of the information - finally add a \hline and this will complete the table.



### **3.3 Personal Reflective Reports**

#### **3.3.1 Martin Vasilev Zokov**

The group project was an interesting experience from which I learned a lot. The task to make an Android application seemed a bit hard at first, but there are a lot of resources which help with understanding the problems we needed to deal with. I wanted to be one of the programmers on the projects so I was given tasks to implement some of the features in the Android application. During the process of developing the application, I learned a lot about the Android platform and how it works. Now I feel confident that I can use the knowledge gained to work on real projects for the Android platform.

Another great aspect of the group project is that I learned a lot about the way that commercial software is built. We were practically using the Waterfall model to implement our application and this gives insight into the way the industry works. I also improved my team working skills greatly, because the project concentrates heavily on collaboration between each member in the team. Not only about decisions that need to be made, but also when implementing certain aspects of the system, programmers should work with each other in order to create a working application.

The group leader (MDA) did a very good job with the task allocation and I feel that I was getting along with all of my team mates. We had no misunderstandings as a group and we worked pretty well in a team.

#### **3.3.2 Ryan Gouldsmith**

I feel that my contribution to this project has been extensive. First of deciding roles for the group was relatively straight forward, and although some people ended up being moved around to other tasks later on in the project to help with the lack of work being put forward, we all agreed that this was probably the best balance of skills in our group. I was delegated the role of Quality Assurance Manager, as well as helping with any issues to do with GitHub. Prior to the coding week, I was there to ensure that all the documents were adhered to by the standard specification that we received. As a group we all read the standard documents so we knew the outline for each of the documents that we were given. I thought we worked well as a team to help ensure that everything was to standard; overall, with the Quality Assurance I felt that we all worked together well and that standards were mostly adhered to.

However, around the Christmas period the Web Development team began to fall

behind with the work. Therefore, Dave asked if I could assist in pushing forward with the Web site because it was falling behind - and as I had prior knowledge of HTML and PHP I was happy to do so. However, this continued into Integration and Testing week in which I had to code majority of the website - as the Web team may have been struggling to complete the work. I felt that the web programmers had done prior PHP courses at University, which wasn't too dissimilar from our group project, and I feel they could have done a little bit more towards the implementation.

I found my team very easy to get a long with, and we didn't have any overall major disagreements. I thought we all worked well as a unit, and we managed to discuss the different alternatives that we all decided. I think that we all made an effort to be there for our group meetings, showing we were all engaged in the project. I felt the most difficult aspect of the project was Integration and Testing week. I thought that this was a very long week, and trying to juggle 3 different aspects was very challenging; I had be QA to make sure if we edited docs they were still in standard, code the functionality of the website and help with the decoding of the JSON file from the Android. However, it was enjoyable being able to work as a team and when I didn't have not the correct solution someone could input: this was present with Dave's help with the multiple images on the website.

From this project I feel that I have expanded additional web knowledge. Having only basic knowledge of PHP and JavaScript, it was good to be able to use this knowledge to produce an application which was flexible with the data that you gave it. I didn't however, have prior knowledge of how JSON worked, so having working decoding the JSON to put it into our Database was good and rewarding when we managed to be able to solve the issue, which stopped us for a couple of days. Additionally, the database design proved a bit of an issue as we were getting JOIN confused with foreign key, eventually we solved this and managed to sort out a database which is suitable to the design. However, I am happy I did manage to expand my knowledge on JavaScript and learning different programming techniques in the language: such as arrays and loops.

From the process I have learnt a great deal of useful information. I would happily work with my group again, but I feel that we could have potentially utilised the experience and confidence in programming a bit better. I would therefore have preferred to move to web development team from the start. However, I believe we have worked well as a team and I am happy with my contribution towards the final product.

### 3.3.3 Zack Lott

In CS22120 we was assigned a group project where we had to make a application and website. We had weekly meetings with Neil about the progress of the application and website. This allows us to review the changes we did to the assignment. Our meetings were on a Friday at 10am but we occasionally had extra meetings with and without Neil to figure out what needed to be done to push the assignment forward.

My role for this assignment was to be a tester, this meant I had to test the application and website, to check for bugs, errors and to make sure they met the requirements. I used the test specification that was created by the group with tests that were required, but with the web we used web validations and PHP unit testing.

From this assignment I have learnt new methods and ways of doing PHP from checking over the web code and creating PHP Unit testing, before this module I hadn't learn PHP unit testing so with the help of Dave(MDA), he helped me understand what it was doing and how to write/read it correctly. While doing the testing it made me understand the real purpose of testing, because checking the little things like checking for symbols in any text box can affect the whole application, this meant I spent a lot of my time while doing the works looking for ways that would break the application or make it crash, I managed to do this a few times but with that information our android workers put in a lot of effort and hard work to fix any problems I gave them.

The assignment did cause me a few problems but this was due to me not understanding many things due to me not studying 2nd part of Java in the first year, but Ryan(RYG1) and Dave(MDA) would quickly explain what it meant and how to solve the problem, this allows me to carry on with the work. If it was to hard then Dave would quickly reassign me something else so we could always keep the assignment going forward.

I believe the group worked well, everyone got a lot with each other, we all had a laugh and they motivated me to work harder. Dave(MDA) was a great leader, you could go to him with any problem and he would drop everything to help try and fix it for you. Ryan(RYG1) was also a great vice-leader as he was always there if Dave was busy and was ready to step in if Dave was unable to attend the meeting. These both helped me with the assignment if I needed anything and they always made sure I was either testing or finding out information for them.

### 3.3.4 Jack Alexander Reeve

My role in the team on this project was a web programmer / designer. This meant I was responsible for designing, building and populating the website. I mainly worked on this with Mark Pitman and Ryan.

Throughout this group project I feel as though I have made an acceptable contribution to the development of the system. In the design stages of the project I wrote up page descriptions for the website and made page designs in Adobe Photoshop. This work was used in the design specification document. I made a base template in HTML and CSS which was developed further into the finished design. In Integration and testing week I spent a lot of time coding parts of the website. I feel that my knowledge of databases is a little limited so this made it difficult to work on pulling the walk details out of the database and listing it on the website. I worked with Ryan on this.

I was able to pull images from the database but I wasn't sure how to pull only the images corresponding to the correct walk. Towards the start of the week I worked on plotting points and drawing lines on the map using the Google Maps API. To start with, we just plotted points using dummy long and lat values. Then I worked on getting the JavaScript code to read these values from PHP variables so that the points plotted on the map were correct. I also worked on displaying POI descriptions and titles in the pop outs from the location pins.

Communication in the team was fairly good. We used emails to organise things such as group meetings. This was a method that was reliable as everyone in the team checks their university account on a regular basis. We had meeting at least once a week but often more, at locations and times suitable for all. Sometimes we worked on aspects of the project at these meetings but most of the time we worked on sections at home and compiled them in the meetings.

Overall I feel we did well on the project as a group. We had a good spread of strengths which meant we were able to put in a good effort on all aspects of the project. We also had members with different skill levels which was good because it allowed other members of the team to learn.

I think if I was taking part in another project similar to this I would do more background research and learn more about the language I would be using. I had already worked with PHP, HTML5, JavaScript and CSS but not in as much detail as this project required. I should have researched specifically how to do what I needed to do well in advance of integration and testing week. Throughout integra-

tion and testing week I found myself having to research certain problems online.

I am fairly happy with the final system. We met the requirements and I feel that if we had more time we would have been able to implement some extra features (such as editing walk descriptions, deleting walks etc.) I also think we could have maybe improved the look and design of the site, although it is functional and is all that is needed for this purpose

### **3.3.5 Mosopefoluwa David Adejumo**

For the CS22120 Group Project, I had the role of group leader. I chose this because I had experience leading teams and had knowledge of both web and android development.

Initially, I found the role somewhat difficult, particularly in keeping all group members engaged and keeping the project on schedule. To assist me in monitoring the progress of all members, and ensure all members were contributing, I requested all members produce weekly reports which would be a summary of the work done in the last week and later on the reports would be a summary of the blog. This was useful in monitoring the amount of time each member spent on the project.

The reports were also useful for monitoring areas where members had problems. Weekly group meetings were arranged early and I made sure all members were kept up to date with proceedings of each meeting. Many issues were brought up during meetings by or with our project supervisor some of which would be solved during the Friday meeting.

I found the prototyping slow on the website end and decided to break work down into individual pages for the web prototyping. Despite initial setbacks, the members were generally responsive with instructions and I found, the carried out the tasks relatively quickly. The only issues I found serious was the lack of web development over the Christmas holiday and I lost contact with the web programmers and thus asked the quality assurance manager to begin development on the website. I ensured all members were able to contact me with ease particularly with issues.

I found pre-assigning tasks for coding week somewhat difficult due to my inability to determine how easily a task would be accomplished, however I did find it fairly simple to do overall planning and set milestones. The project supervisor suggested I use hour based deadlines/timeframes for project tasks. I found this difficult and the group members preferred our existing system whereby deadlines were set on

specific days. I.e. instead of spending a limited number of hours on a task, the task would have to be completed by a particular day. This system worked for the members. The project supervisor also suggested involving more group members, but the group members and I wanted to avoid this as it could compromise the system which we had and was working as required.

Overall I think the group performed well and constantly kept me up to date with all progress made. They would quickly work on a task that I instructed and lack of technical knowledge was the main barrier to accomplishing tasks. If they had issues they would keep me informed and also notify me about problems. I found it easy to explain certain areas to them, though there were issues of misinterpretation of instructions or statements. If they had issues it was easy to solve and while there were situations where several members would have problems, the members were very intuitive at solving them. This allowed me to assist multiple members in a short space of time and thus keep development on schedule.

I did find several members putting in more effort than others, though this was possibly due to the different areas they worked on and the fact that some members did not understand certain concepts which was brought to my attention and thus, I believe the members cannot be faulted as they would quickly pick up work where problems were encountered. I made several jokes to keep the mood light hearted and keep morale high, which had effects as there were times particularly during integration and testing week when members became overly burdened with work and thus I would instruct them to rest as required while I took up the task they had issues with in an attempt to solve them.

I learned a lot about the importance of feature creep identification during the project and the importance of communication between all the members and feel my team performed extraordinarily well.

### **3.3.6 Mark Radcliffe Pitman**

The group worked well together and there were little arguments, which was good. Everyone had their role and stuck to it.

My role was the web part. Mostly CSS but helped with the HTML and PHP too. I attended most meetings with the tutor and our general meetings. I missed 2 or 3 meetings for good reasons. We had meetings every Friday at 10am. Occasionally we had extra meetings if something needed to be sorted.

I learnt a lot for this project from working with a group and more knowledge

on HTML, PHP and CSS. I researched a lot on PHP and the assignment I did for web helped a lot. Also learnt a bit from the android side. It felt good once everything was done and everyone's work came together. When seeing the app uploading to the database and it shows on the website felt fulfilling.

The leader did a very good job keeping everyone busy and telling everyone what needs to be done and by when. Was very supportive and tried to help everyone with their jobs if they needed help.

We did have problems like the website wasn't receiving from the database and show on the map. Problems with the images to show up. The info box some-time didn't show what we wanted. The CSS could mess things up with the texts, for example, making the page smaller, the words would move around where they shouldn't be. This was all fixed after other people started helping each other cause they finished their part of the job. This showed good teamwork.

Personally I enjoyed working as a group. I felt more motivated cause I didn't want to disappoint the group. If something wasn't done then it just doesn't affect you but the whole group.

There were some stressful moments when things weren't working but that was expected. Nothing was really left to the last minute because the leader wanted things to be done some time before the deadline so it can be checked and changed. Glad everything went mostly well and it is finally done.

### **3.3.7 Mark Alexander Smith**

Our CS22110 module involved us being part of a nine-person team and creating an android application about walk tours. It would involve the application using GPS on a users phone, saving the details (co-ordinates, description and images) of that users walk; this would link to a database. This information could then be sent to a website which would display all of the users walks.

Our first meeting involved us deciding roles for our team. We decided that David should be our project leader; he was enthusiastic about taking up this role. Ryan took up the role as being QA manager. Martin and Harry both took the roles of being android developers. Myself, Jack and Mark P took the roles of website developers, myself creating the database. Zack and Maciej accepting the roles of being android and PHP testers.

Our group would meet on a regular basis, every Tuesday with our Tutorial Tutor,

Neil Taylor. We would then also meet together as a group once or twice a week in our own time, usually in the orchard or the old college in town. I think this gave us an edge over other groups as each week we would all attend and discuss any problems we had, and create new jobs before we visit Neil later in the week. Obviously not everyone could make every session due to other commitments, but communication between the group was good enough so that this happened as little as possible.

We were in a good position leading up to Integration and Testing week (ITW), however this week is where most of the work would get done. I started on working with Ryan on the filesaver.php file. This became an issue with the group and became a harder task to do than first project and therefore took a lot longer than first suggested. Later in the week I continued with the website by creating the base of the CSS and also helping out with some problems, and also taking out some testing of the android application.

Overall I believe that our group took the correct attitude towards such a big project and getting as much done as possible before ITW. Meeting in our own time took a big weight off our shoulders and lead us to getting a working application that would send information to the database and website. David lead us well in making sure deadlines were met and that we were up-to-date on the Gantt chart.

### **3.3.8 Harry Flynn Buckley**

During this task I occupied myself with the Android-server communication and general handling of the walk data on the device. I feel that I am largely responsible for the class design of the Android application, and although towards the end the project it had to stray a little from the initial design, due to unforeseen complications with the Android camera and gallery apps, I am still happy with majority of the design choices. As there were only two people programming on the android, I had quite a lot of input to all the aspects of the android, the pop up dialogs where the only area which I feel I only had little contribution.

The group worked quite effectively, discussing much of the design before making a decision, I do however think that class diagram was not looked at very thoroughly as I received little feedback on my designs. I think the UI design perhaps had too much focus in the early meetings because UI is something that is very accessible and everyone has an opinion on. When the implementation part of the project started I wrote the initial GUI based on what was decided by the group, this went on to become the GUI of the final submission. I then worked on writing the all the model classes (WalkModel, LocationPoint, PointOfInterest, ImageInformation).



A feature that was eventually cut was a local sqlite database that stored walks before they were uploaded, I managed to get the database working with much of the walk data but it was not complete. The next task that I was set on was the uploading of walk data to the server, this was initially done in name value pairs, before we decided on using JSON. To test the upload I had written a php file that receives and saves the send data and I think this file contributed to the final file\_saver.php. During ITW I finished off the JSON packaging, and did all the camera and gallery interaction. I had some trouble writing the camera and gallery, as it involved slightly redesigning the way images are stored and how places of interest are added to the walk model. The packaging of the walk data into a JSON format was also quite difficult.

Overall I have found the process quite fun and exciting, more so the closer we got to the code hand in date.

### **3.3.9 Maciej Wojciech Dobrzanski**

At the beginning of the project, we were assigned to certain roles. I was supposed to be a designer, tester and a person responsible for the documents. We also decided that we can all help each other with our tasks to make everything easier and more convenient. At the beginning I had quite a lot of work, which I feel I've done pretty well. I was delivering everything on time, was also trying to help others with their parts of the design, because a team of designers was supposed to work together to create a really specific design. To sum up, the first part of my work was done just as planned and I feel like I couldn't really do anything more than that. I certainly could work on a couple of things faster, to allow others to use my work, but no one really needed that.

I was also supposed to work on the Use Case and UML Diagrams. We had quite a few ideas of how they should look like, I was trying to gather and improve them. I've learned quite a lot about the UML Diagrams, which were not really my strong side. The use cases were designed quite well, we thought it might be a good idea to add a few features to the project to make it more appealing, so the changes were clearly visible on the diagrams. The coders helped me develop a good diagram. Although the use cases were not a big issue and they were ready really quickly, I was really worried about the UML diagram as it took me a really long time to come up with the idea of how it should look like. I knew that the coders were not able to start working because of me, and I was trying to figure it out as soon as possible, to give them more time on the project itself.

After the design specification was ready, I did not really have a lot of work to

do. I was trying to help the coders with the Android application, but the Android Java code was a little bit different than a normal Java code, so it was quite difficult for me. The group asked me to work on a Git Wiki. I managed to catch a few things of how to write it pretty quickly and spend about a week developing it. I think it wasn't really good as my overall knowledge of the project details was quite shallow, everyone had to add a few things from their own experience during development of the project. I was really struggling with it because I didn't really see a point in doing it. I didn't see how it could be useful.

When I came back from the Christmas break, I reviewed all of the documents we've prepared and I tried to catch up with the current state of the project to be able to spend my time well during the coding week. When I was really able to do my part(which was testing) I tried to help everyone else created JUnit tests, was helping the coders and doing a little bit of all-around help, asking other group members if they needed any help. I did all I could with the documentation when there was nothing else to do. The first plan was that I am supposed to write the final documentation, but when the coding week was coming to an end, we all knew that one person just cannot do that, it will have to be a group effort, as everyone did something else and no one really had a deep knowledge of every single part of the project.

To conclude, I've learned a lot during the group project. I was trying to do as much as possible to be an important group member, but sometimes I had an impression that everything is taking too much time and the group won't be happy with my work. I would like to think of myself as a valuable member of the group who was always able to have something to add and was always able to help others, but the truth is that sometimes my knowledge was just not enough and I was not satisfied with myself. On the other hand, I always managed to do anything that I was asked to do, even if that required more work and researching things that I've never seen before. I had a lot of fun and the whole project helped me develop not only as a person, but also a good future employee.

### **3.4 Revised Project Plan**

---

## Group Project 07 Project Plan

---

*Authors:* Mosopefoluwa David Adejumo  
Ryan Gouldsmith  
Harry Flynn Buckley  
Zack Lott  
Mark Radcliffe Pitman  
Jack Alexander Reeve  
Mark Alexander Smith  
Martin Vasilev Zokov  
Maciej Wojciech Dobrzanski

*Config ref:* SE\_07\_PM\_01

*Date* February 17, 2014

*Version* 2.4

*Status* Release

Department of Computer Science  
Aberystwyth University  
Aberystwyth  
Ceredigion  
SY23 3DB  
Copyright ©  
Aberystwyth University 2013

## Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Scope . . . . .	4
1.3	Objective . . . . .	4
<b>2</b>	<b>PROJECT OVERVIEW</b>	<b>5</b>
2.1	Platforms . . . . .	5
2.1.1	Android . . . . .	5
2.1.2	HTML 5 . . . . .	5
2.1.3	PHP . . . . .	5
2.1.4	MySQL . . . . .	5
2.1.5	Google Maps API . . . . .	5
2.2	Target Audience . . . . .	6
2.3	System Overview . . . . .	6
2.3.1	Android Application . . . . .	6
2.3.2	Online Offline . . . . .	6
2.3.3	Walk Screen . . . . .	7
2.3.4	Walk Recorder . . . . .	7
2.3.5	Database Protocol . . . . .	7
2.3.6	Server . . . . .	7
2.3.7	Request Handler . . . . .	7
2.3.8	Walks . . . . .	7
2.3.9	Physical Database . . . . .	7
2.3.10	Website . . . . .	7
2.3.11	Online Walk List . . . . .	7
2.3.12	Online Walk Viewer . . . . .	8
<b>3</b>	<b>USE CASE</b>	<b>9</b>
3.1	Android . . . . .	9
3.2	Website . . . . .	9
3.3	Interaction System . . . . .	10
3.4	Descriptions . . . . .	11
<b>4</b>	<b>ANDROID USER INTERFACE DESIGN</b>	<b>14</b>
4.1	Start Screen . . . . .	14
4.2	New Walk Screen . . . . .	15
4.3	Recording Screen . . . . .	16
4.4	New Point Of Interest . . . . .	17
4.5	Edit Walk Information Screen . . . . .	17

4.6	Walk Complete . . . . .	20
4.7	Cancel A Walk . . . . .	22
<b>5</b>	<b>WEBSITE USER INTERFACE DESIGN</b>	<b>23</b>
5.1	Home Page . . . . .	23
5.2	View Walks Page . . . . .	24
5.3	Walk Page . . . . .	25
5.4	Point Of Interest Selected Page . . . . .	26
5.5	Point Of Interest Image Page . . . . .	27
<b>6</b>	<b>NAVIGATION OVERVIEW</b>	<b>28</b>
6.1	Android . . . . .	28
6.2	Website . . . . .	29
<b>7</b>	<b>GANTT CHART</b>	<b>30</b>
<b>8</b>	<b>RISK ASSESSMENT</b>	<b>32</b>
<b>9</b>	<b>DOCUMENT HISTORY</b>	<b>34</b>

# 1 INTRODUCTION

## 1.1 Purpose

This document displays how the project will be completed and any risks involved. It outlines the requirements specified by the client as a series of documents.

## 1.2 Scope

This document should be read by all members of the group. It contains a list of tasks, the schedule and risks involved in the project. It details what the application and server will be required to do. It also gives an overview of the whole software - the Use Case diagrams, UML diagrams, the UI of the website and the navigation overview of Android application and the website. The Gantt chart gives an idea of our milestones and describes what tasks are assigned to every member of the group. The document doesn't give any specifics about the classes in the application, doesn't cover any information about the database connection and it doesn't provide information about the website. These will be covered in the design specification.

## 1.3 Objective

- List the platforms to be used for the project
- Provide a task schedule for the project
- Provide a description of how the application and website will be used.
- Provide a list of risks and how to reduce their effects
- Provide an idea of the UI for the Android application and the website
- Provide a description of how the application and website can be navigated

## **2 PROJECT OVERVIEW**

The proposed system is an application running on the Android operating system that will be used to record walks for a particular user. The application will allow the user to start a recording of a new walk and add points of interest to that walk and save the walk. The website will allow the user to view the walks they uploaded, with all the information associated with it, like points of interest with the photos and descriptions on the map, short and long description of the walk itself and the entire path the user recorded.

### **2.1 Platforms**

#### **2.1.1 Android**

As stated by the client, the operating system used will be Android. This will be developed for mobile devices. The operating system version will be 4.2. This is because we have a few devices running on that operating system, so it is just the most convenient one.

#### **2.1.2 HTML 5**

The website will be built using HTML 5 alongside CSS 2 and CSS 3. This will allow the latest version of HTML to be used for the website. We are going to use Google Maps, which requires the latest version.

#### **2.1.3 PHP**

PHP will be used to handle the communication between the mobile device and the server. It will be run server side and is understood to a working level by the web programmers.

#### **2.1.4 MySQL**

The database will be built using MySQL. It shall store information about each walk and the walk themselves. Information stored will include all points of interest added, their associated long and short descriptions and any pictures taken.

#### **2.1.5 Google Maps API**

This API gives us all the features we need. OpenSpace API gives some additional ones like offline use, but we decided that it is not really required in our project, so we decided that Google Maps is just easier to work on.



## 2.2 Target Audience

This application is aimed at Second Year Computer Science students. Precautions had to be taken while designing the user interface to prevent the user from having to navigate through too many screens.

## 2.3 System Overview

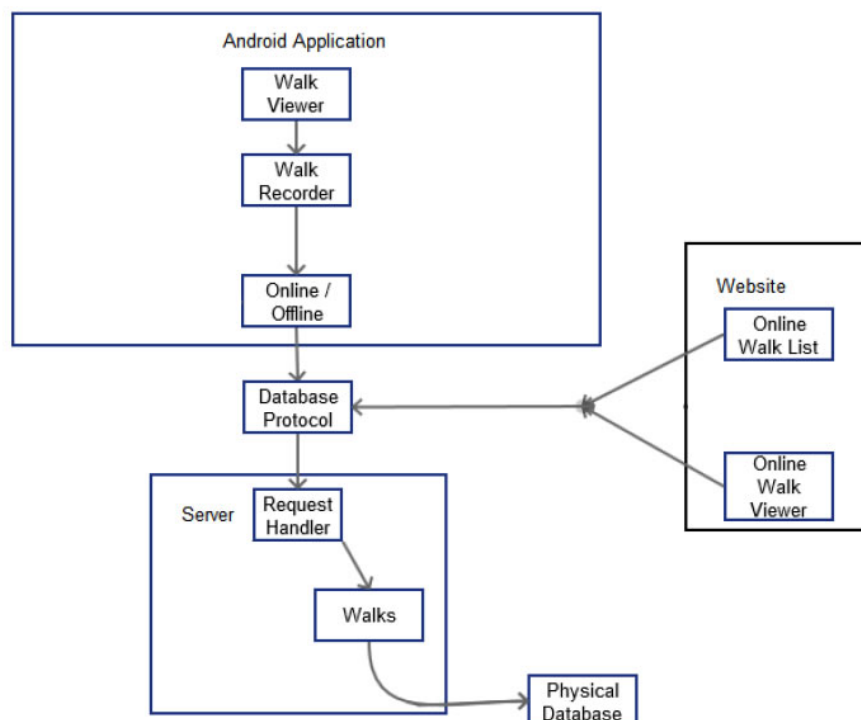


Figure 1: System Overview

### 2.3.1 Android Application

This is the application. All modules here are running on the mobile device

### 2.3.2 Online Offline

This module handles the location where data is stored. If the user is not connected to the internet they receive an error message saying that they won't be able to upload the walk.

### **2.3.3 Walk Screen**

This module handles the displaying options about the walk, like cancelling it, adding points of interest or uploading it.

### **2.3.4 Walk Recorder**

This module handles the storage of points of interest, the time taken for a walk and the walks location during recording.

### **2.3.5 Database Protocol**

This module handles the conversion of database request to their required language such as from POST to HTTP for the website.

### **2.3.6 Server**

This is the server that handles all requests between the database, website and android application

### **2.3.7 Request Handler**

This module deals with linking data between users

### **2.3.8 Walks**

This module handles the retrieval and presentation of the walks uploaded by the user.

### **2.3.9 Physical Database**

This is the machine where all request are handled

### **2.3.10 Website**

This module serves as the control for everything on the website

### **2.3.11 Online Walk List**

This module handles all lists being displayed to anyone on the website.

### **2.3.12 Online Walk Viewer**

This module handles the conversion of data into visual form for browser based viewing of walks.

### 3 USE CASE

#### 3.1 Android

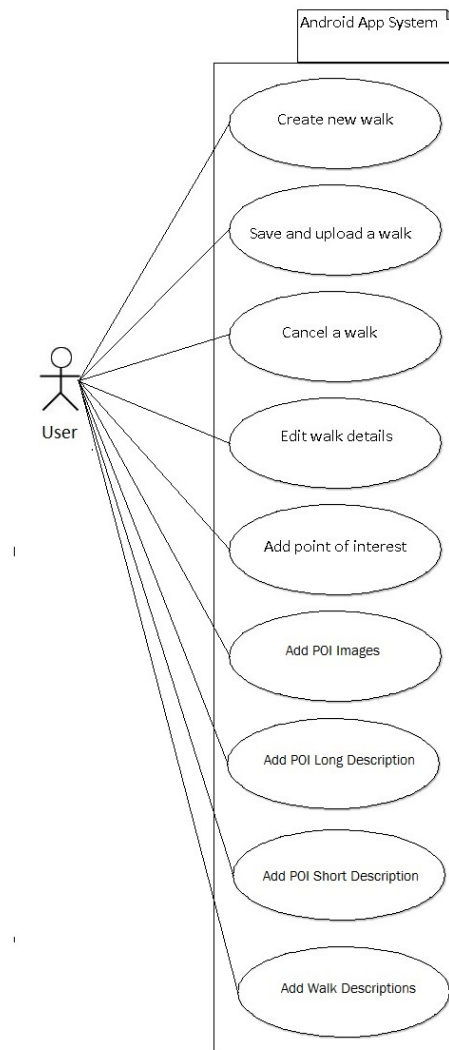


Figure 2: Android Use-Case diagram

#### 3.2 Website

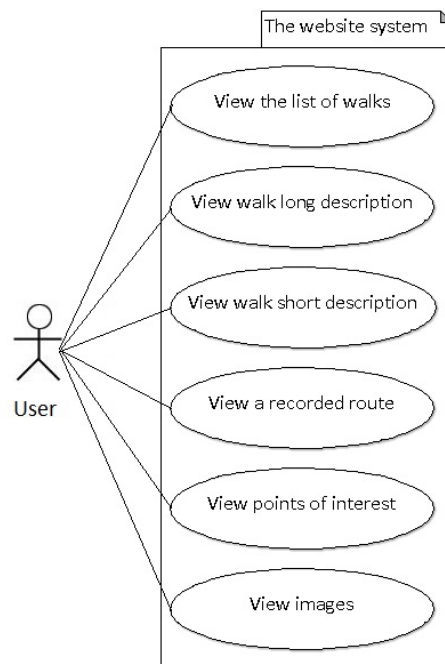
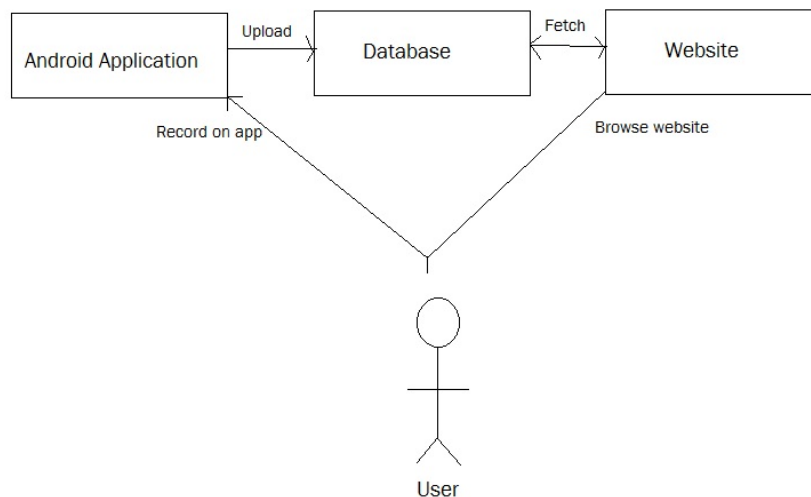


Figure 3: Website Use-Case Diagram

### 3.3 Interaction System



The diagram above represents the interaction of the whole system. The user interacts with both Android application and the website. The walks recorded by the user along with all the other information like description and the photos, are stored in the database and can be easily accessed later via the website. All uploads are stored in the database for the website to retrieve. The Android application uploads walks to the server, which deals with the information and inserts it appropriately in the database. The Android application does not have a direct link to the website. The website also pulls a list of walks from the database. The website will also list all the information associated to one singular walks, such as Long description, short description, any pictures. This will then be displayed appropriately on the website.

### 3.4 Descriptions

Diagram Name	Use case name	Description
Android	Create new walk	Allows the user to start recording a new walk
	Add a point of interest	The user must add points of interest on the walk. This includes a short description, an optional long description and optional images. A timestamp is automatically taken when a point of interest is saved
	Save and upload a walk	When the user has finished their walk they will click the finish the walk button, this will then upload the walk to the server where it will be processed.
	Cancel a walk	If the user wishes to end their walk then they can click the cancel button. This will cancel any recorded data associated with the current walk.
	Edit walk details	The user will be able to edit any information associated with the current walk. This could be a point of interest, or the title of the whole walk. The information about the walk, in which they're editing, has to be shown to the user.
	Add point of interest	The user can add points of interest. These are locations with descriptions with or without images
	Add POI Images	The user can take a picture of a location and add it to the walk. Alternatively, they can add a picture from their photo library. The user should be also able to upload multiple images to any given point of interest.
	Add POI Long Description	The user can add a detailed description of a point of interest
	Add POI Short Description	The user can add a brief description summarising a point of interest
	Add Walk Descriptions	The user can add description both long and short to the walk and can be edited later
Website	View the list of walks	This will show a list of the walks in the database that multiple user has uploaded to the server. They will be displayed in a list form on the website.
	View walk long description	Once a walk has been selected from the list of walks it will tell the user what the long description associated to that walk is. This will also appear, in the google maps popup.

	View walk short description	Once a walk has been selected from the list of walks it will tell the user what the short description associated to that walk is. This will also appear, in the google maps popup.
	View a recorded route	When the user selects the walk from the list of walks screen, then it should show the user the route they have walked. This will be shown on the map as a trail. Each of the points of interest along the walk will be located with a marker.
	View points of interest.	When the user views the walk and it has a series of points of interests, they will be shown on the map as a marker value. The user will then be able to click on the marker; there will then be a popup showing the walk title, description for that POI. Along images inside the popup, this allows the option for multiple images.
	View images.	When the user selects a walk they will be able to see all the images associated with the walk on the side. If they wish to see where the images comes from, the images will be associated with a given marker on the map.



## 4 ANDROID USER INTERFACE DESIGN

### 4.1 Start Screen

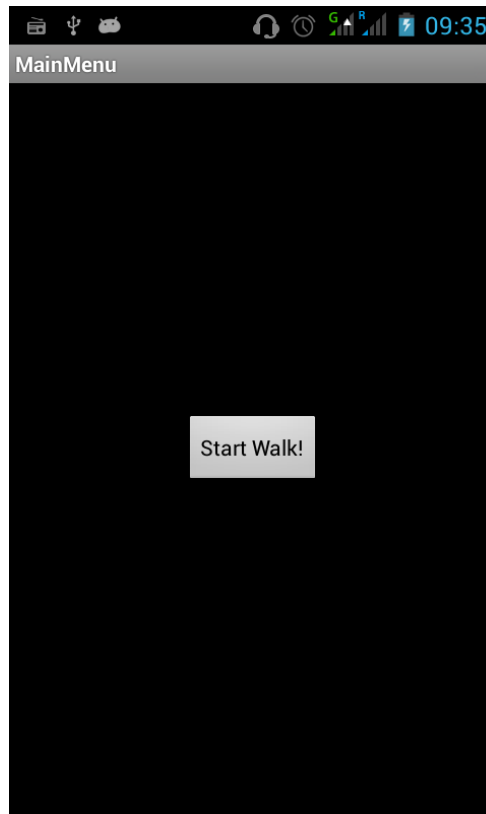


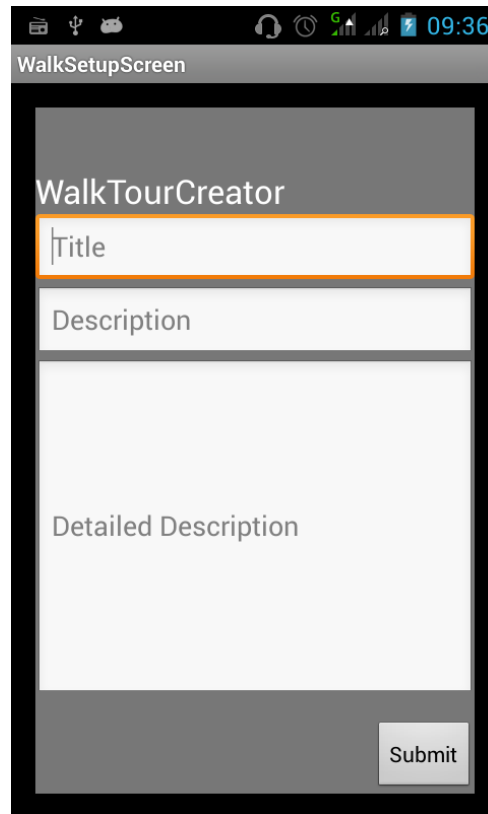
Figure 4: Start Screen

Used as a filler screen before the user starts a walk. This screen may be replaced with a tutorial or help screen on first launch in future.

#### NAVIGATION

Start → New Walk Screen (Fig. 4.2)

## 4.2 New Walk Screen



The screenshot shows a mobile application interface for creating a walk. The top status bar displays various icons and the time 09:36. Below this is a title bar labeled 'WalkSetupScreen'. The main content area has a header 'WalkTourCreator'. Underneath the header are three text input fields: 'Title' (which is highlighted with an orange border), 'Description', and 'Detailed Description'. At the bottom right of the screen is a 'Submit' button.

Figure 5:

This is the walk creation screen. It allows a short and long description to be added to a walk.

### NAVIGATION

Back → Main Menu (Fig. 4.1)

Start Walk → Recording Screen (Fig. 4.3)

### 4.3 Recording Screen

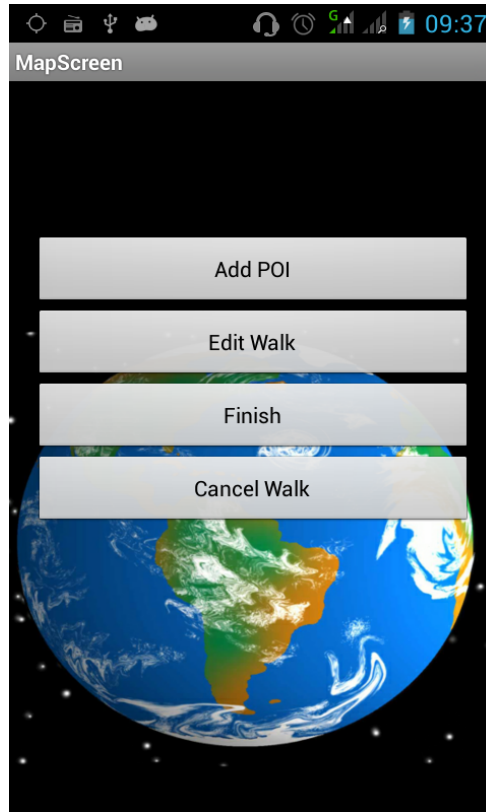


Figure 6:

This screen will give the user the option to edit the walk, add points of interest and finish or cancel walks. Recording will only begin when a GPS signal has been found. The user will receive a message to indicate recording has begun

#### NAVIGATION

- Add POI → New Point of Interest Screen (Fig. 4.4)
- Edit Walk → Edit Walk Information Screen (Fig 4.5)
- Finish → Walk Complete Screen(Fig. 4.6)
- Cancel Walk → Start screen without uploading walk (Fig 4.1)

## 4.4 New Point Of Interest

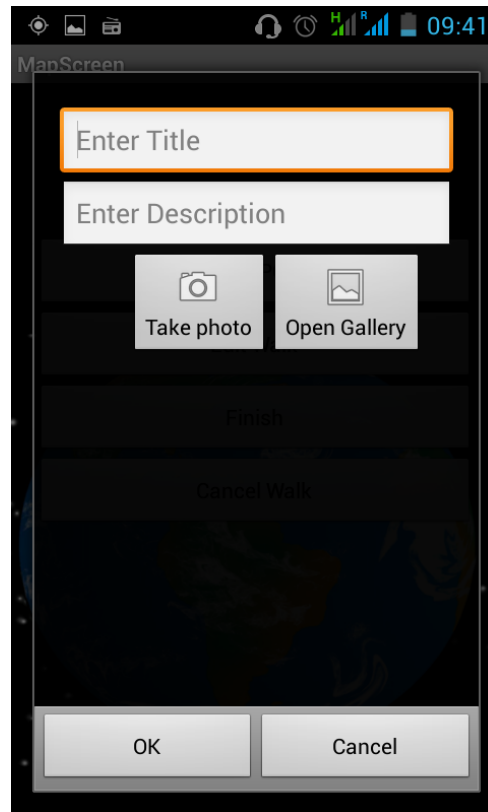


Figure 7: Add Point Of Interest

This screen is used to add a point of interest. It will appear over the recording screen. Adding images will open a dialogue asking whether to go to the photo library or the camera app, allowing images to be added. Images will appear between the short and long description and can be removed from here. Pressing save stores the point of interest but can be removed later.

### NAVIGATION

- Cancel → Recording screen without saving(Fig. 4.3)
- Add Image → Dialogue for Camera or Photo Library
- Save → Recording screen with save (Fig. 4.3)

## 4.5 Edit Walk Information Screen

This screen allows the user to edit the walk information

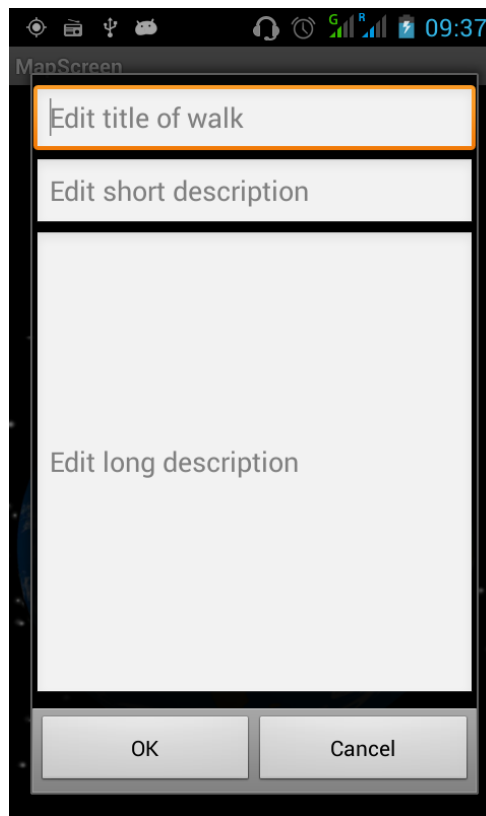


Figure 8: Edit Walk

OK → Recording Screen with new details(Fig 4.3)  
Cancel → Recording Screen without saving (Fig 4.3)

## 4.6 Walk Complete

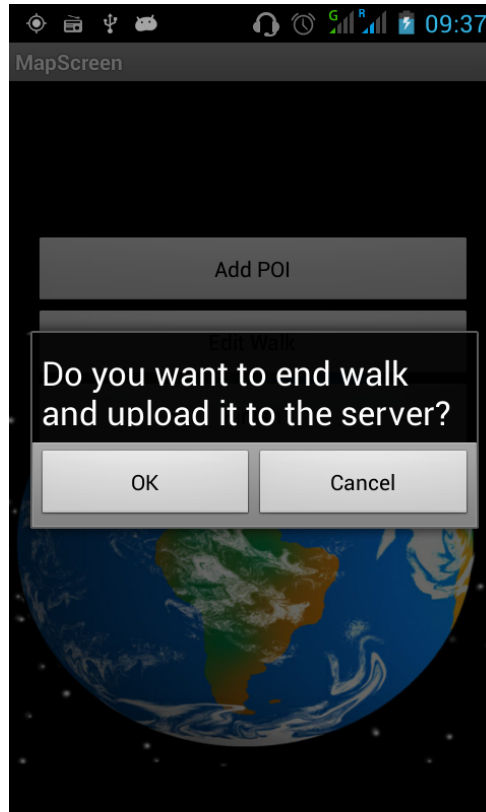


Figure 9: Upload Walk

This screen allows the user to save a walk. If upload is pressed, the walk is saved then uploaded to the server provided the user is signed in. This screen should be unavailable if there are no points of interest to prevent uploading or saving an empty walk. IN future, this screen could show the time taken to complete a walk, the name of the walk, the number of points of interest added and the location of the walk.

### NAVIGATION

Cancel → Recording Screen without uploading(Fig. 4.3)

OK → Recording screen during upload (Fig 4.3) Then Start Screen on upload complete(Fig. 4.1)

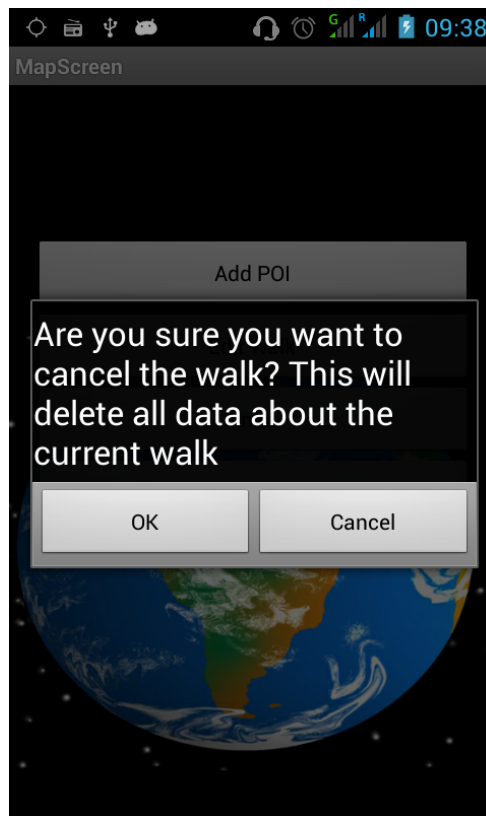


Figure 10: Cancel A Walk



## 4.7 Cancel A Walk

This screen allows the user to cancel a walk without saving. Selecting OK will return them to the start screen. Cancel will continue the walk **NAVIGATION**

Cancel → Recording screen (Fig 4.3)

OK → Start screen without uploading (Fig 4.1)

## 5 WEBSITE USER INTERFACE DESIGN

### 5.1 Home Page



Figure 11: Website Home Page

This is the homepage of the website. From here the user can find information about the application page and can view walks.

#### NAVIGATION

View Walks → View Walks Page (Fig. 5.2)

## 5.2 View Walks Page



Figure 12: View Walks Page

The user can view all uploaded walks via this screen. From here the user can see a small map overview of the walk and the short description of the points of interest.

### NAVIGATION

Click on Walk → Walk Page (Fig. 5.3)

Home → Home Page (Fig. 5.1)

### 5.3 Walk Page



Figure 13: Walk Page

This page displays a map overview of the walk, the average time taken to complete the walk and the long and short descriptions. The images from every point of interest are displayed at the bottom of the screen.

#### NAVIGATION

- Click on Image → Point of Interest Image Page (Fig. 5.4)
- Click Pin on Map → Point of Image Selected Page (Fig. 5.5)
- View Walks → View Walks Page (Fig. 5.2)
- Home → Home Page (Fig. 5.1)

## 5.4 Point Of Interest Selected Page



Figure 14: Point Of Interest Selected

Clicking on a pin on the map opens this page. The selected pin is also highlighted. The page displays the average time taken from the start of the walk to arrive at this point of interest. If there are any images taken from this point of interest, the user is can view them.

### NAVIGATION

Click on Map → Walk Page (Fig. 5.3)

Click Pin on Map → Point of Image Selected Page (Fig. 5.5)

View Walks → View Walks Page (Fig. 5.2)

Home → Home Page (Fig. 5.1)

## 5.5 Point Of Interest Image Page

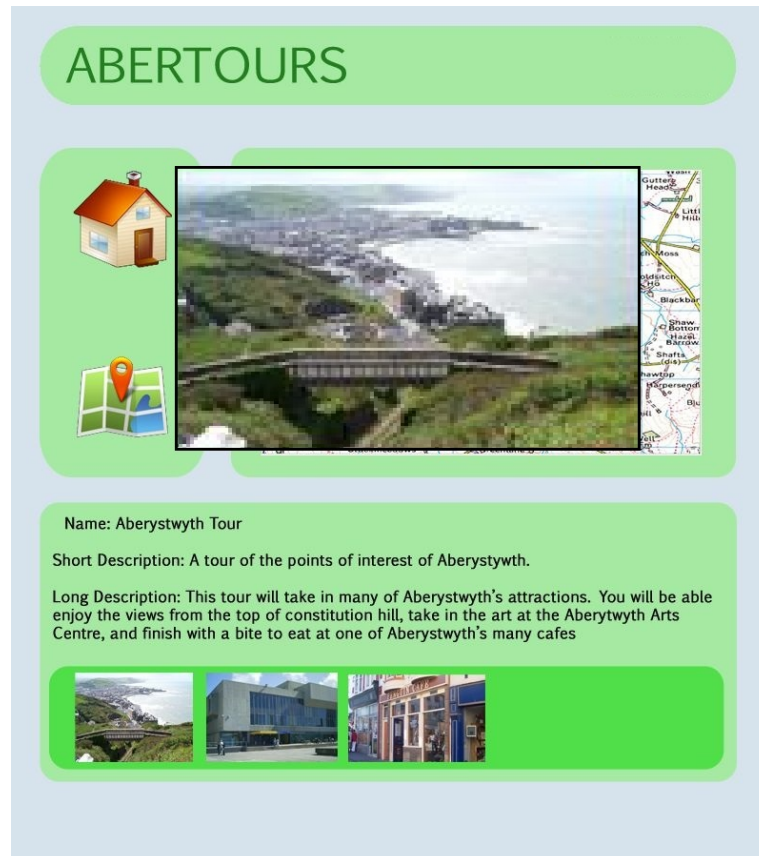


Figure 15: Clicking an Image associated with a walk

This simply enlarges the image clicked. Clicking outside the box minimizes the image back into the tray.

### NAVIGATION

Click Outside Image → Previous Page

## 6 NAVIGATION OVERVIEW

### 6.1 Android

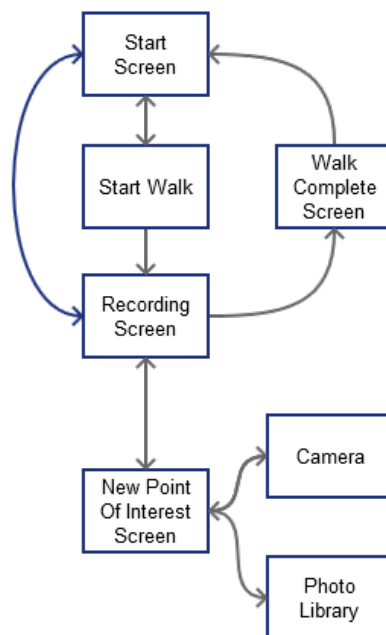


Figure 16: Android Navigation

Start screen is the entry point. All navigation is done via buttons and icons unless otherwise stated

## 6.2 Website

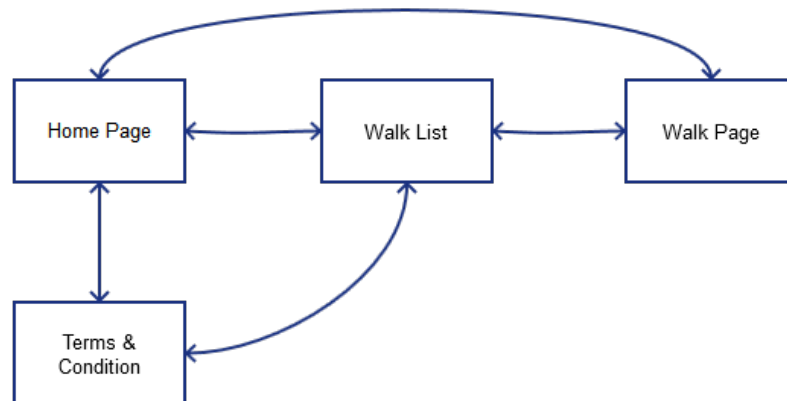


Figure 17: Website Navigation

Home page is the entry point. All pages link back to the home page



## 7 GANTT CHART

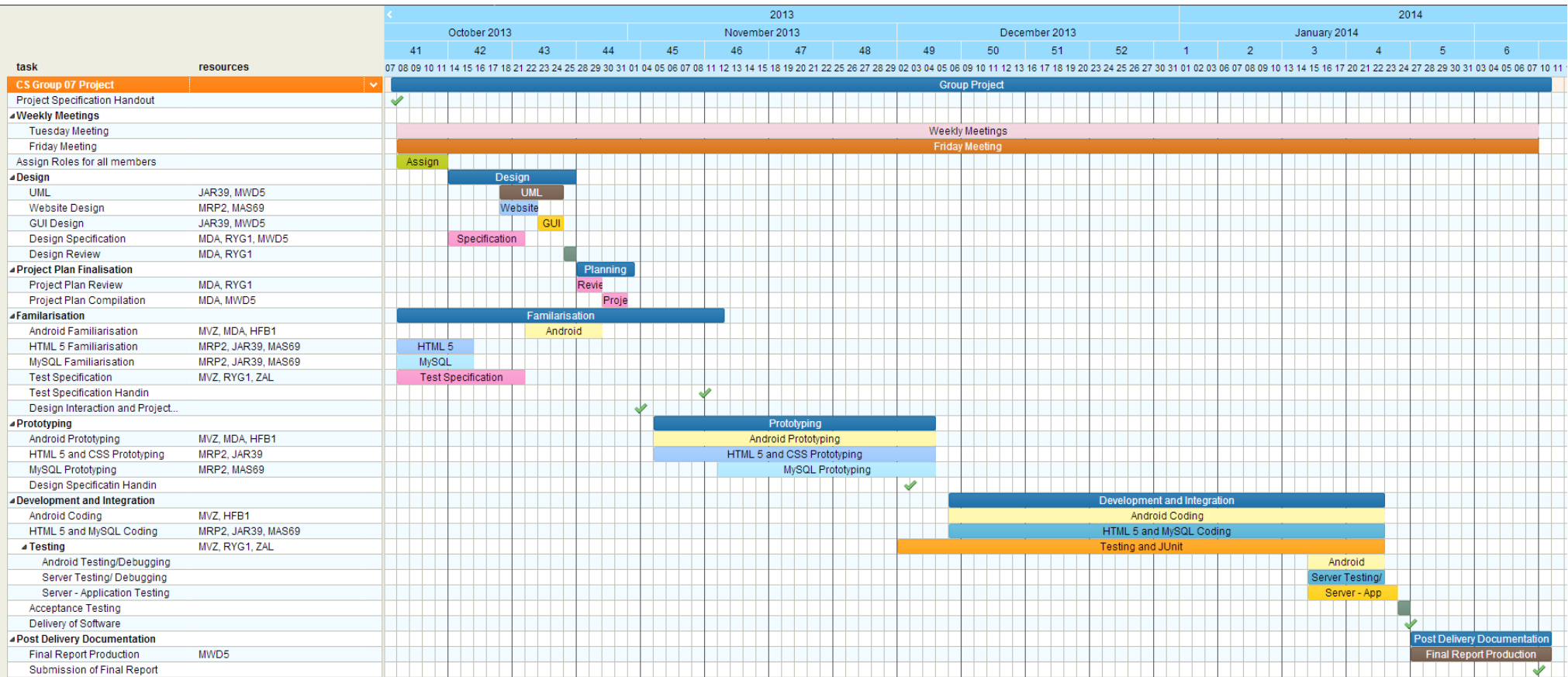


Figure 18: Gantt Chart

## 8 RISK ASSESSMENT

Event	Risk	Mitigation
Git Down-time	Low	All work should be backed up on multiple devices, preferably the University of Aberystwyth M: Drive and local backup locations. Work can continue on local branches
Absence of Team Leader	Low	The deputy team leader will take up responsibilities as required.
QA Manager Absence	Low	Team leader or deputy team leader will take up responsibilities as required.
Poor Quality Work	Low	All work must be verified and monitored by both the QA Manager and the Team Leader. Deadlines for tasks are given before official deadlines to provide a window in which work is brought up to standard.
Problems with Maps API	Medium	In the event of inability to use the OpenSpace API, Google Maps API will be used due to its wide use.
Absence of Team Member	Medium	In the absence of any member, work will proceed as normal. All members should notify the group leader if they will be absent at the next meeting. Any absent member should read the minutes of the last meeting and any other documents produced. Continued unauthorized absence will result in warnings then penalties.
Project Off Schedule	Medium	Members are required to stick to the schedule and provide weekly reports on all project related tasks throughout the week. In the event of failure to stick to the schedule, tasks must be revised to bring project back on schedule.
Server Down-time	Medium	Website and server development should be done locally and added to the university server regularly. In the event of downtime, work should proceed as normal locally. A local LAMP or similar server may be used for testing
Unrequired Features	Medium	Extra features should not be a priority and should not be added unless the final product meets the required specification. A copy of the final product must be used for adding any extra features.

Lack of knowledge of platforms	Medium	In the event of any team member being unable to do work due to not knowing how to perform a task on the platform, the team leader must be notified. Any members capable who know how to proceed will be assigned to performing that task. All members are required to gain as much knowledge about the API and languages during the familiarisation stage.
Member Unable to Continue Project	High	If for any reason a member is unable to continue the project, tasks will be reshuffled to accommodate the change. Multiple members are assigned similar tasks to help reduce the risk in such an event.
Loss of Data	High	Users are required to regularly backup data. If for any reason data is not backed up and is lost, the group leader must be notified immediately and more work must be done to bring the project back on schedule. Tasks may be reprioritised to ensure deadlines are met.
Change in Requirements	High	If requirements are changed by the client, a meeting will be called immediately to meet the new requirements. Regular communication between the client and the team leader is required.
Hardware Incompatibility	High	The application must be thoroughly tested on at least 2 android mobile devices. Tablet compatibility is not required. In the event of hardware incompatibility or related issues, extensive debugging and testing must be done and the team leader must be notified immediately.
Application Server Incompatibility	High	The application should send data in the format specified. The server must be able to parse the data accurately. In the event of incompatibility, android and server side debugging must be done to determine the cause of the incompatibility.

## 9 DOCUMENT HISTORY

Version	CFF No.	Date	Section Changed From Previous Version	Changed by
1.0	N/A	28/10/13	Original draft of document written by Mosopefoluwa David Adejumo	MDA
1.1	N/A	31/10/13	Added new screens. Updated project overview	MDA
1.2	N/A	31/10/13	Updated Android user interface	MDA
1.3	N/A	2/11/13	Updated Android user interface and description. Added Website User Interface Description Added Gantt chart. Added Navigation overview Updated risk assessment	MDA
1.4	N/A	2/11/13	Added use case and descriptions. Added system overview. Updated project overview	MDA
1.5	N/A	2/11/13	Updated system overview. Updated use case. Updated UI descriptions	MDA
1.6	N/A	3/11/13	Updated Fig. 5.3 and Fig 5.4 images. Added interaction system diagram and description. Moved risk assessment to item 8	MDA
1.7	N/A	4/11/13	Updated Interaction System and replaced image. Corrected config ref number	MDA
1.8	N/A	6/11/13	Updated Gantt chart.	MDA
1.9	N/A	13/02/14	Re-wrote the document in LaTeX.	RYG1
2.0	N/A	13/02/14	Added updated images to file	RYG1
2.1	N/A	15/02/14	Re-sized the Gantt Chart	RYG1
2.2	N/A	16/02/14	Edited Feature creep Images	RYG1
2.3	N/A	16/02/14	Updated navigation and interface description texts and use case descriptions. Added system interaction diagram	MDA
2.4	N/A	16/02/14	Added cancel walk descriptions and sorted formatting	MDA

### **3.5 Revised Design Document**

---

# Group Project 07 Design Specification

---

*Authors:* Mosopefoluwa David Adejumo  
Ryan Gouldsmith  
Harry Flynn Buckley  
Zack Lott  
Mark Radcliffe Pitman  
Jack Alexander Reeve  
Mark Alexander Smith  
Martin Vasilev Zokov  
Maciej Wojciech Dobrzanski

*Config ref:* SE\_07\_PM\_01

*Date* February 17, 2014

*Version* 2.9

*Status* Release

Department of Computer Science  
Aberystwyth University  
Aberystwyth  
Ceredigion  
SY23 3DB  
Copyright ©  
Aberystwyth University 2013

## Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Scope . . . . .	4
1.3	Objective . . . . .	4
<b>2</b>	<b>ARCHITECTURAL DESCRIPTION</b>	<b>5</b>
2.1	Programs In System . . . . .	5
2.1.1	The Android Application . . . . .	5
2.1.2	The Database Server . . . . .	5
2.1.3	Web Application . . . . .	6
2.2	Significant Classes . . . . .	6
2.2.1	Android Application . . . . .	6
2.2.1.1	WalkModel . . . . .	6
2.2.1.2	RouteRecorder . . . . .	6
2.2.1.3	FileTransferManager . . . . .	7
2.2.1.4	GeneralActivity . . . . .	7
2.2.2	Database Server . . . . .	7
2.2.3	Web Application . . . . .	7
2.2.3.1	Index . . . . .	7
2.2.3.2	Walk List . . . . .	7
2.2.3.3	Walk Details . . . . .	7
2.2.3.4	Google Maps Api . . . . .	8
2.2.3.5	File_Saver . . . . .	8
2.3	Table Mapping Requirements Onto Classes . . . . .	8
<b>3</b>	<b>DEPENDENCY DESCRIPTIONS</b>	<b>9</b>
3.1	Component Diagrams . . . . .	9
3.1.1	Android . . . . .	9
3.1.2	Website . . . . .	9
3.1.3	Database Server . . . . .	10
<b>4</b>	<b>INTERFACE DESCRIPTION</b>	<b>10</b>
4.1	Screens . . . . .	11
4.1.1	OptionsScreen . . . . .	11
4.1.2	MainMenuScreen . . . . .	11
4.1.3	MyWalkScreen . . . . .	12
4.1.4	WalkSetupScreen . . . . .	12
4.1.5	MapScreen . . . . .	12
4.1.6	GeneralActivity . . . . .	13



4.2	Views . . . . .	14
4.2.1	MapView . . . . .	14
4.2.2	WalkInfoView . . . . .	14
4.2.3	PopupView . . . . .	14
4.2.4	PoiInfoView . . . . .	14
4.2.5	PlacesVisitedView . . . . .	15
4.2.6	WalkFinishedView . . . . .	15
4.2.7	AddPoiView . . . . .	16
4.3	Models . . . . .	16
4.3.1	WalkModel . . . . .	16
4.3.2	LocationPoint . . . . .	18
4.3.3	PointOfInterest . . . . .	19
4.4	Controllers . . . . .	20
4.4.1	RouteRecorder . . . . .	20
4.4.2	FileTransferManager . . . . .	21
<b>5</b>	<b>DETAILED DESIGN</b>	<b>21</b>
5.1	UML Diagrams . . . . .	21
5.1.1	Android Sequence Diagram . . . . .	21
5.1.2	Sequence Diagram For Web . . . . .	23
5.1.3	Overall Interaction Sequence Diagram . . . . .	23
5.2	Class Diagram . . . . .	23
5.3	Significant Algorithms . . . . .	25
5.3.1	Android Algorithms . . . . .	25
5.3.1.1	RouteRecorder Algorithm . . . . .	25
5.3.2	PHP Algorithms . . . . .	25
5.3.2.1	Connect To The Database . . . . .	25
5.3.2.2	Append To The Server Database . . . . .	26
5.4	TODO CHECK THE ALGORITHMS . . . . .	27
5.5	Significant Data Structures . . . . .	27
5.5.1	WalkModel . . . . .	27
5.5.2	LocationPoint . . . . .	28
5.5.3	PointOfInterest . . . . .	28
<b>6</b>	<b>REFERENCES</b>	<b>29</b>
<b>7</b>	<b>DOCUMENT HISTORY</b>	<b>30</b>

# 1 INTRODUCTION

## 1.1 Purpose

The purpose of this document is to, specify the technical design of both the Android and web applications. It will go into detail regarding functions. This will allow us to more easily designate tasks to team members when it comes to coding week. It will also show how these functions interact with each other and how the website, server and Android app interact through the use of sequence diagrams. The document is structured in a way that makes it easy to refer to when the programmer needs clarification on how to build a certain function. The document will also show how the database will be structured and what the field names will be.

## 1.2 Scope

This document, will cover all aspects of the Android and web design and their implementation. It should be read by all members of the group and approved by the client. It will be used as a guide for the programmers to build from in coding week. The document will allow the team leader to assign a given function to a team member which they can then code.

## 1.3 Objective

The precise areas which this document will cover are:

- Provide a clear class diagram, covering all aspects of the Android app.
- Define, in detail, the interaction between all the programs in the system.
- Provide a structure for implementation of the applications.
- Outline the significant systems to be used in the applications.
- Provide descriptions of functions.

## 2 ARCHITECTURAL DESCRIPTION

### 2.1 Programs In System

The walk tour application consists of:

- The Android application.
- The Data server.
- The Website application.

#### 2.1.1 The Android Application

The Android application is used to create physical data representing a route allowing the users to record and upload a walk. It allows the user to add points of interest along a route and associate points of images. It displays a map screen and is used to record location data for a walk using GPS. It also gives the user options to add pictures to a walk.

Requirements Covered: (FR1, FR2, FR3, FR4, FR5, FR6, FR7,FR9, EIR1, PR1)

#### 2.1.2 The Database Server

Stores walk info in MySQL which it receives from the Android application as a MIME type. When the server application receives information for a walk it appends the location data to the database and stores all pictures on the server machine. The database server will also have a PHP file which handles the uploading of data from the Android device. The file that handles the upload can be accessed via the URL in a browser, but doing so will present an error message.

Requirements Covered: (DC3)

- List of Walks relation:
  - id
  - title
  - shortDesc
  - longDesc
  - hours
  - distance
- Location
  - id

- walkID
  - latitude
  - longitude
  - timestamp
- Place description
  - id
  - locationId
  - name
  - description
- Photo Usage
  - id
  - placeId
  - photoName

### **2.1.3 Web Application**

Allows the user to view walks in more detail. The website is also hosted on the data server and can be used for viewing information about walks including route taken, points of interest and pictures. This program overlaps with 1.1.2 (Database Server). It interacts with the database using PHP. Requirements covered: (FR8, FR9)

## **2.2 Significant Classes**

### **2.2.1 Android Application**

This section describes the most significant classes in the application. The complete set of classes can be seen in the class diagram Section 4.1.2. These classes will all be written in Java.

#### **2.2.1.1 WalkModel**

A WalkModel holds all the data concerning a single route, this includes a list of all location points that trace the path and a list of all the places of interest.

#### **2.2.1.2 RouteRecorder**

The RouteRecorder retrieves the current location from the system, and depending on factors such as speed and direction, the location information will be added to

the local WalkModel. This class will carry out some analysis of the path traveled so far to determine when to record points,i.e. if a recorded path seems to be traveling in a straight line then fewer point will be need added than if the path traces a circle.

#### **2.2.1.3 FileTransferManager**

A connection will be made with the server via the FileTransferManager. It is responsible uploading and downloading WalkModels, including all associated images, from the database server. This class only interacts with the WalkManager, so any objects wishing to upload or download content must connect through WalkManager, this is to add an extra layer of abstraction that simplifies the solution.

#### **2.2.1.4 GeneralActivity**

GeneralActivity is an abstract class that extends Activity. It defines the general layout of all the screens (MainMenuScreen, MapScreen, etc.). It provides subclasses with access to several static variable that describe the layout that allow changes such things as the background, and text colour. All the screens displayed to the user are subclasses of GeneralActivity.

### **2.2.2 Database Server**

The files here are used to control the interaction between the database and the other programs in the module. All these files will be written in PHP. Object Oriented Programming will not be implemented in this system.

### **2.2.3 Web Application**

The following are files in PHP that will be used to interact between the database and the website. These are also pages that will be visible and accessible by the user unless otherwise stated. Object Oriented Programming will not be implemented in this system.

#### **2.2.3.1 Index**

This file will serve as the homepage and holds links to view the list of walks and terms of service.

#### **2.2.3.2 Walk List**

This file will process information from our database and display it as a list of walks. The walks will be clickable in order to view them in more detail. Users will be able to select a walk via this file

#### **2.2.3.3 Walk Details**

This file will be used to give the user a more in depth look at a specific walk. This means they will be able to see a map view, images taken on the walk, and points

of interest.

#### 2.2.3.4 Google Maps Api

The Google Maps API will be used to portray a persons walk data into a visual map. The user will also be able to view points of interest on the map. This will serve as a separate file that will interact with Googles system.

#### 2.2.3.5 File\_Saver

The Apache HTTP Client will be used for by the android application to send data to our database server. This will mean our application will be able to 'POST' data to the server. This reduces load on the server compared to our previous idea of zipping and unzipping each set of files for a walk. The data will be sent as a JSON string. This file will decode the JSON string and add all the walk data to the appropriate tables where required

### 2.3 Table Mapping Requirements Onto Classes

This section gives an overview of what classes/files cover what requirements as specified by the client.

FR1	GeneralActivity, MapScreen, WalkSetupScreen, MyWalksScreen, MainMenuScreen, OptionsScreen, WalkInfoView
FR2	MapScreen, RouteRecorder, WalkModel
FR3	LocationPoint, PointOfInterest
FR4	LocationPoint, PointOfInterest
FR5	WalkInfoView
FR6	WalkManager, FileTransferManager
FR7	RouteRecorder

## 3 DEPENDENCY DESCRIPTIONS

### 3.1 Component Diagrams

#### 3.1.1 Android

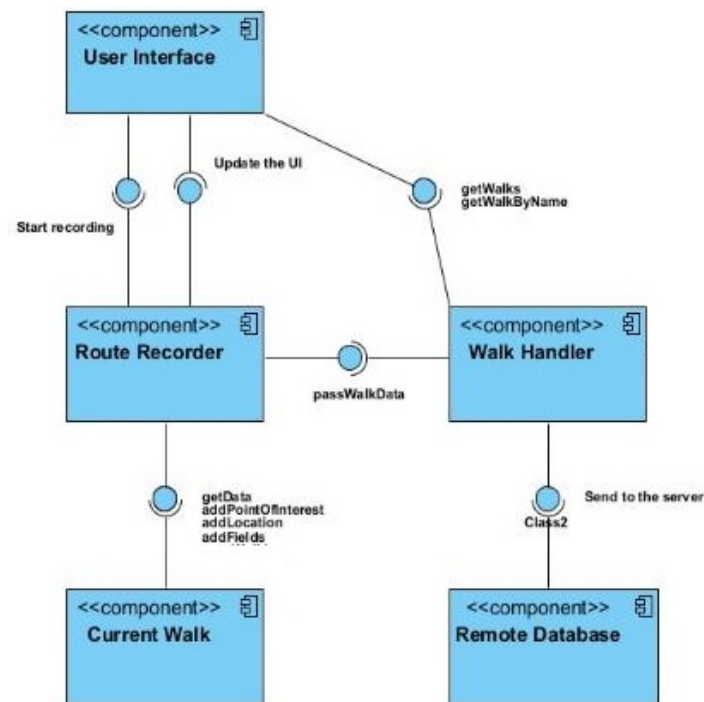


Figure 1: Android Dependency Diagram

#### 3.1.2 Website

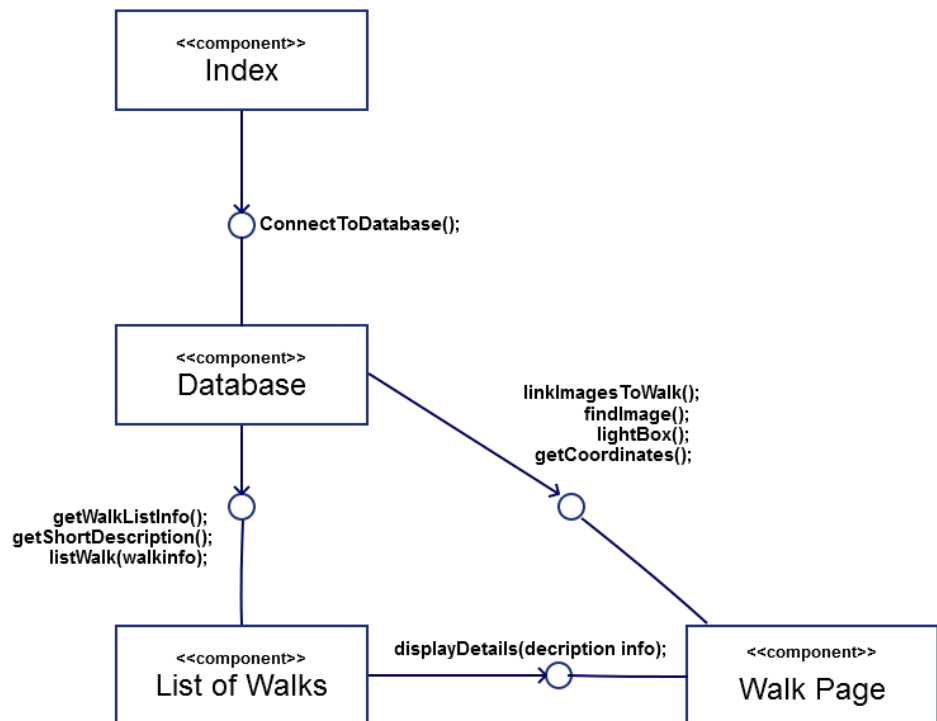


Figure 2: Web Component Diagram

### 3.1.3 Database Server

## 4 INTERFACE DESCRIPTION

This section contains an implementation of different classes and files in the program. However, there is a possibility that the implementation in the final product may differ from what is displayed here.



## 4.1 Screens

The following classes all extend Activity and are used to control the display.

### 4.1.1 OptionsScreen

```
public class OptionsScreen extends GeneralActivity {  
    /**  
     * This method changes the background color ,  
     * of all generalActivity subclasses  
     * @param v, is the object that called the method.  
     */  
    public void changeBackgroundColor(View v);  
}
```

### 4.1.2 MainMenuScreen

```
public class MainMenu extends GeneralActivity {  
  
    /**  
     * Starts a new MyWalkScreen activity ,  
     * and displays it to the user.  
     * @param v, is the object that called the method.  
     */  
    public void startMyWalksScreen(View v);  
  
    /**  
     * Starts a new WalkSetupScreen activity ,  
     * and displays it to the user.  
     * @param v, is the object that called the method.  
     */  
    public void startWalkSetupScreen(View v);  
  
    /**  
     * Starts a new OptionsScreen activity ,  
     * and displays it to the user.  
     * @param v, is the object that called the method.  
     */  
    public void startOptionsScreen(View v);  
  
    /**  
     * Starts a new LoginScreen activity ,
```

```
        * and displays it to the user.
        * @param v, is the object that called the method.
        */
    public void StartLoginScreen(View v);
}
```

#### 4.1.3 MyWalkScreen

```
public class MyWalksScreen extends GeneralActivity {

    /**
     * open a WalkInfoView popup on selected walk
     * @param v, is the object that called the method.
     */
    public void viewWalk(View v);
}
```

#### 4.1.4 WalkSetupScreen

```
public class WalkSetupScreen extends GeneralActivity {

    /**
     * Starts a new MapScreen activity
     * and displays it to the user.
     * The detail that the user has input,
     * are passed to the new activity.
     * @param v, is the object that called the method.
     */
    public void startWalk(View v);
}
```

#### 4.1.5 MapScreen

```
public class MapScreen extends GeneralActivity {

    /**
     * creates and displays a AddPoiView.
     * @param v, is the object that called the method.
     */
    public void addPOI(View v);

    /**
     * creates and displays a WalkFinishedView.

```

```
        * @param v, is the object that called the method.
        */
    public void finishWalk(View v);

    /**
     * creates and displays a PlacesVisitedView.
     * @param v, is the object that called the method.
     */
    public void showPlacesVisited(View v);
}
```

#### 4.1.6 GeneralActivity

```
public abstract class GeneralActivity extends Activity{

    /**
     * changes the background color of all
     * GeneralActivity
     * subclasses to the passed
     * value. The int c, represents a color.
     */
    public void setBackgroundColor(int c);

    /**
     * changes the foreground color of all
     * GeneralActivity subclasses to the passed
     * value. The int c, represents a color.
     */
    public void setForegroundColor(int c);

    /**
     * changes the text color of all GeneralActivity
     * subclasses to the passed value.
     * The int c, represents a color.
     */
    public void setTextColor(int c);
}
```

## 4.2 Views

### 4.2.1 MapView

```
public class MapView/* our class */ extends MapFragment/* from  
google API */{  
    /**  
        * sets the walks that is to be displayed  
        */  
    public void setWalk(WalkModel walk);  
  
    /**  
        * this method will cause the map 'window'  
        * to redraw the route on to itself.  
        */  
    public void updateWalk();  
}
```

### 4.2.2 WalkInfoView

```
public class WalkInfoView extends PopupView{  
  
    /**  
        * creates a WalkInfoView instance.  
        * The WalkModel that is passed to it  
        * is displayed in in the popup.  
        */  
    public class WalkInfoView(WalkModel walk);  
}
```

### 4.2.3 PopupView

```
public abstract class PopupView extends DialogFragment {  
    /**  
        * closes the popup view.  
        * @param v, is the object that called the method.  
        */  
    public void closePopup(View v);  
}
```

### 4.2.4 PoiInfoView

```
public class PoiInfoView extends PopupView{
```

```
/**
 * creates a PoiInfoView instance. The
 *   PointOfInterest
 * that is passed to it
 * is displayed in in the popup.
 */
public void PoiInfoView(PointOfInterest point);
}
```

#### 4.2.5 PlacesVisitedView

```
public class PlacesVisitedView extends PopupView{
    /**
     * creates a PlacesVisitedView instance.
     * All the PointOfInterest from the
     * passed walk are displayed in a table.
     */
    public void PlacesVisitedView(WalkModel walk);

    /**
     * opens a PoiInfoView.
     * @param v, is the object that called the method.
     */
    public void getPoiInfo(View v);
}
```

#### 4.2.6 WalkFinishedView

```
public class WalkFinishedView extends PopupView{
    /**
     * displays a screen displaying
     * a summary of the finished walk, and
     * shows various options to the user regarding the
     *   WalkModel.
     */
    public void WalkFinishedView(WalkModel walk);

    /**
     * open a PoiInfoView
     * @param v, is the object that called the method.
     */
}
```

```
        public void uploadWalk(View v);  
    }
```

#### 4.2.7 AddPoiView

```
public class AddPoiView extends PopupView{  
    /**  
     * displays an place description input popup,  
     * and gives it a link to the RouteRecorder  
     */  
    public void AddPoiView(RouteRecorder recorder);  
  
    /**  
     * creates a PointOfInterest out of the given  
     * data (from text fields) and add the point the  
     * the WalkModel  
     * @param v, is the object that called the method.  
     */  
    public void submit(View v);  
  
    /**  
     * uses ImageHandler to open the photoLibrary,  
     * the selected photo is then added to the  
     * PointOfInterest.  
     * @param v, is the object that called the method.  
     */  
    public void getPhotoFromLibrary(View v);  
  
    /**  
     * uses ImageHandler to open the camera app,  
     * the taken photo is then added to the  
     * PointOfInterest.  
     * @param v, is the object that called the method.  
     */  
    public void getPhotoFromCamera(View v);  
}
```

### 4.3 Models

#### 4.3.1 WalkModel

```
public class WalkModel {
```

```
/**
 * creates a WalkModel, with LocationPoints already
 * set, it is used
 * by the WalkManager when loading walk from
 * database.
 */
public WalkModel(String title , Vector<LocationPoint>
    path , String shortDesc , String longDesc);

/**
 * @return a vector of all the LocationPoint in the
 * walk.
 */
public Vector<LocationPoint> getRoutePath();

/**
 * @return the running total of km travelled.
 */
public double getDistance();

/**
 * @return the elapsed time since the walk was
 * started.
 */
public double getTimeTaken();

/**
 * @return the name of the walk.
 */
public String getTitle();

/**
 * @return a short description of the walk
 */
public String getShortDescription();

/**
 * set the short description of the walk.
 */
```

```
        public void setShortDescription(String newShortDesc
        );

        /**
         * @return a long description of the walk.
         */
        public String getLongDescription();

        /**
         * set the long description for the walk.
         */
        public void setLongDescription(String newLongDesc);

        /**
         * adds a LocationPoint to the walk.
         */
        public void addLocation(LocationPoint point);
    }
```

#### 4.3.2 LocationPoint

```
public class LocationPoint {

    /**
     * creates a new LocationPoint
     */
    public LocationPoint(double x,double y);

    /**
     * creates a LocationPoint,
     * used to recreate a point stored in the
     * database.
     */
    public LocationPoint(double x,double y,double time)
        ;

    /**
     * @return the time at which the point was recorded
     * .
     */
    public double getTime();
}
```



```
/**
 * @return the longitude, the east/west
 * distance from Greenwich.
 */
public double getLongitude();

/**
 * @return the latitude, the north/south distance
 * from the equator.
 */
public double getLatitude();

/**
 * @return the distance between itself and a passed
 * point.
 */
protected double distanceTo(LocationPoint point);
}
```

#### 4.3.3 PointOfInterest

```
public class PointOfInterest extends LocationPoint{

    /**
     * creates a PointOfInterest, at position x,y.
     * The time is set automatically
     */
    public PointOfInterest(double x,double y);

    /**
     * creates a PointOfInterest, at position x,y.
     * The time is also explicitly defined, this is
     * used when creating a PointOfInterest from a
     * database entry.
     */
    public PointOfInterest(double x,double y,double
        time);

    /**
```

```
        * @return all the images associated with this
          point.
      */
      public Vector<ImageInformation> getImages();

      /**
       * @return the description of this place.
      */
      public String getDescription();

      /**
       * sets the description of this point.
      */
      public void setDescription(String desc);
  }
```

## 4.4 Controllers

### 4.4.1 RouteRecorder

```
public class RouteRecorder extends Service implements
    LocationListener{

    /**
     * creates a RouteRecorder instance ,
     * with the MapView that will display the walk.
    */
    public RouteRecorder(MapView map);

    /**
     * starts the recording of location points
    */
    public void startRecording();

    /**
     * adds a PointOfInterest to the recorded path.
    */
    public void savePoi(PointOfInterest poi);

    /**
     * stops the recoding of locations.
    */
}
```

```
        public void finish();  
    }
```

#### 4.4.2 FileTransferManager

```
public class FileTransferManager{  
  
    /**  
     * @param walk  
     * makes a connection to data server and  
     * uploads all files belonging to the given  
     * file, the return values will be zero if  
     * the method succeeded without problems.  
     */  
    public int uploadWalk(WalkModel walk);  
}
```

## 5 DETAILED DESIGN

This section details the algorithms and interactions that will be implemented in the program. The algorithms used may differ from the final product.

### 5.1 UML Diagrams

#### 5.1.1 Android Sequence Diagram

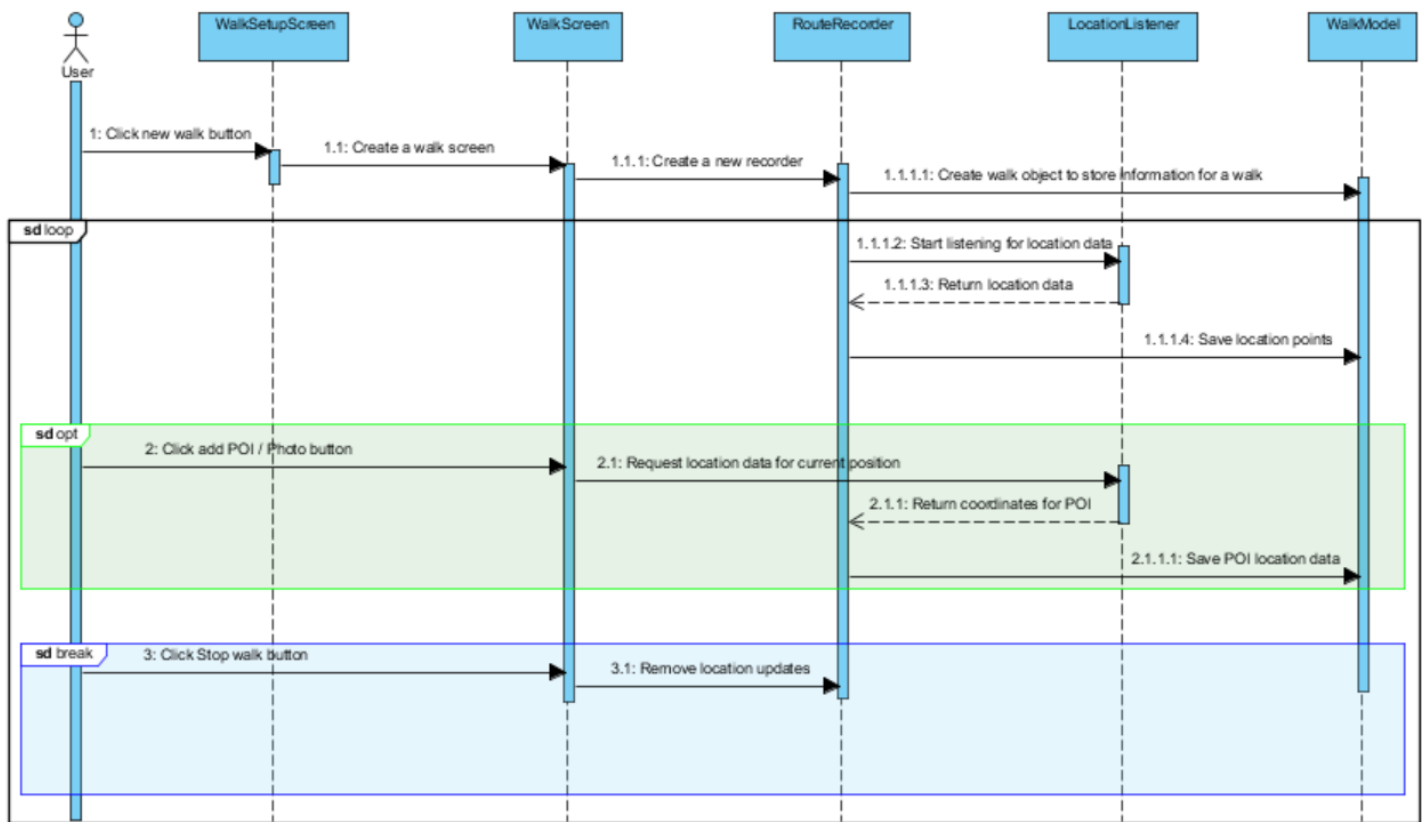


Figure 3: Android Sequence Diagram

The sequence diagram describes the recording of a walk and how the classes which are involved in the process interact. In action 1. the user is prompted for details in the WalkSetupScreen and after he/she presses the start walk button, a map screen is shown and a RouteRecorder and WalkModel objects are created. After that the application goes into a loop of actions from the RouteRecorder, Location-Listener and WalkModel classes. The recorder asks the listener for location data and when the data is returned, it is saved in the WalkModel's array of location points. Action 2 is optional for the user, because it is not mandatory to have a Point of interest or photos in every walk. If a user decides to click the Add POI button, the LocationListener gives the coordinates of the current location to the RouteRecorder and they are saved in the WalkModel object. Action 3 is the exit point of the loop for the current walk recording. It is done by clicking the stop button which brakes the loop and saves the last set of coordinates for the current walk. The LocationListener is deliberately not activated at all times while a walk is in progress in order to save battery life.

### 5.1.2 Sequence Diagram For Web

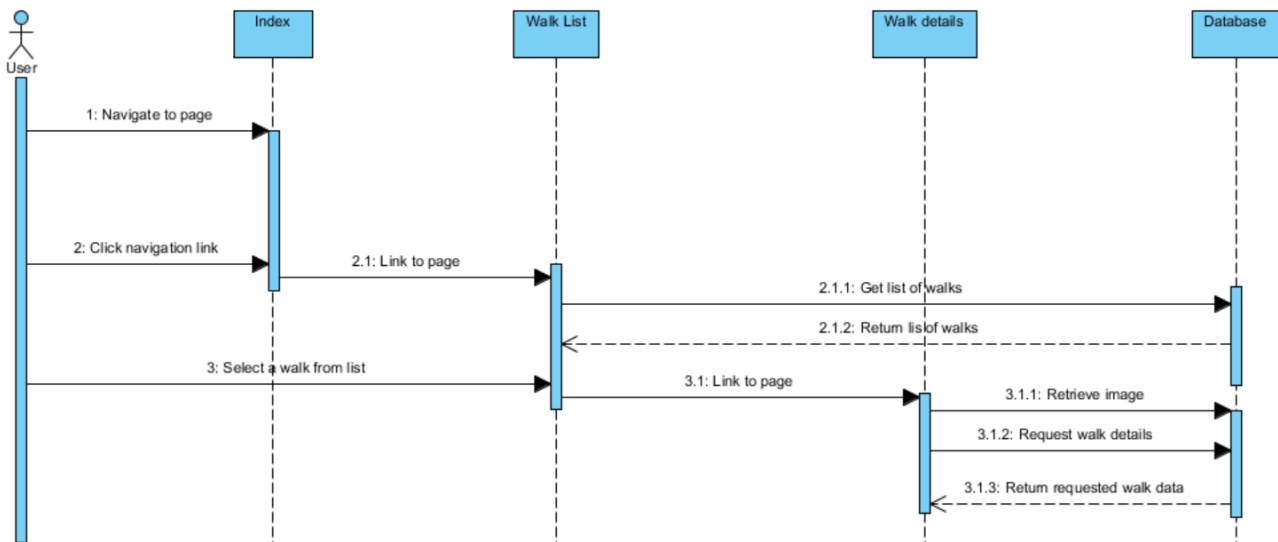
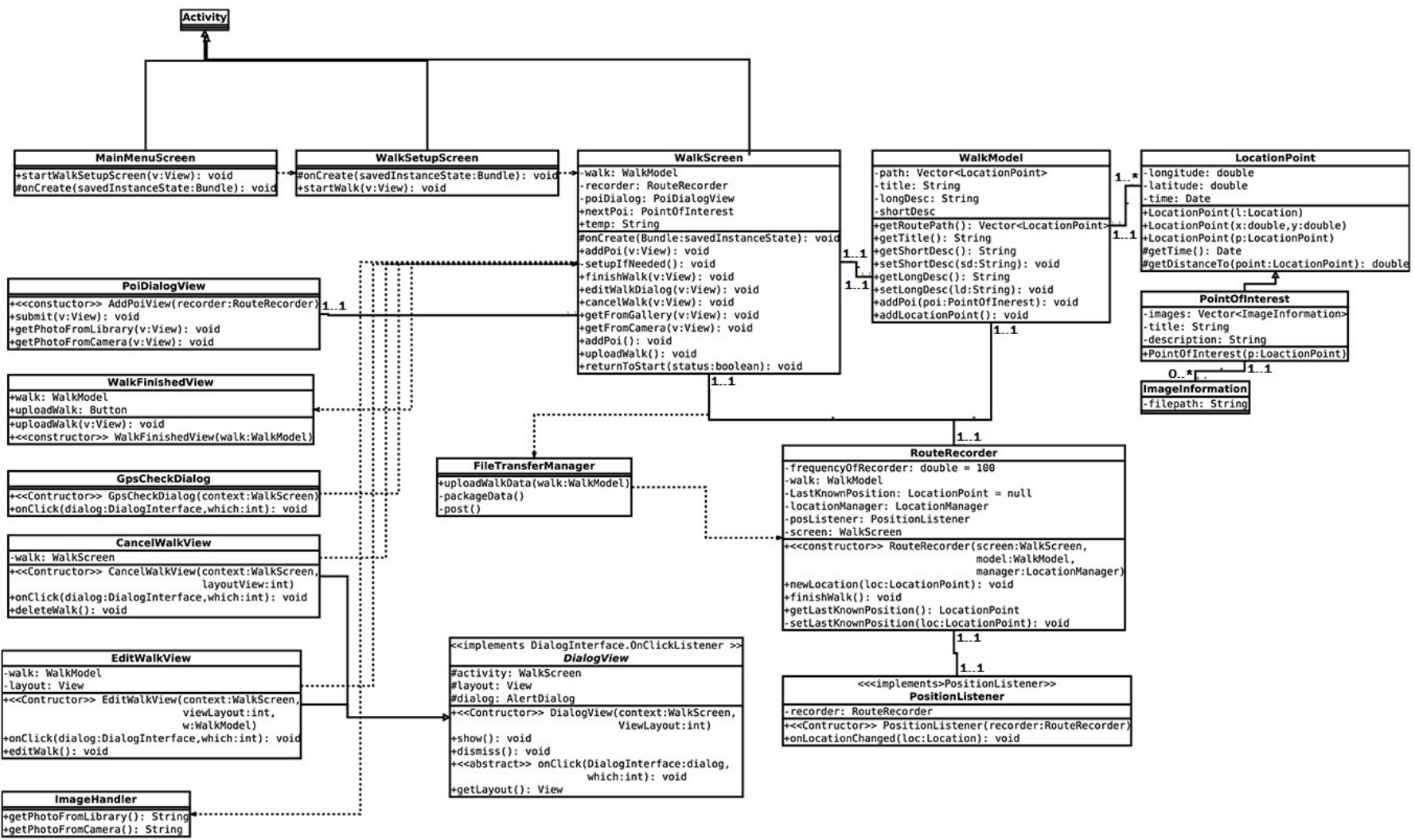


Figure 4: Web Sequence Diagram

### 5.1.3 Overall Interaction Sequence Diagram

## 5.2 Class Diagram



The classes ending in screen, are all Activities. They all, in some way, display a layout to the screen and respond to user input. Any response that requires further processing would be passed to another class and then handed back to be displayed, but it would be the screen class itself that initialised the action. There are several classes that have been suffixed with View, these classes all extend the android class View. They are all visible to the user and act much like screen classes except that they don't use the whole screen and do not change the displayed screen only create new Views. The classes WalkModel, PointOfInterest and Location can all be considered to be model classes. They are used to store the walks data in an organised fashion, and have no methods to do anything other than to set and get information. WalkManager, ImageHandler and FileTransferManager all perform some tasks that are not immediately apparent to the user. They are the utility classes that are used by others.

## 5.3 Significant Algorithms

### 5.3.1 Android Algorithms

#### 5.3.1.1 RouteRecorder Algorithm

```
while walk not finished do  
    get location  
    if distance between new location, old location > X then  
        add new location to walkModel  
    end if  
end while
```

### 5.3.2 PHP Algorithms

#### 5.3.2.1 Connect To The Database

```
/**  
 * This is the function to connect to the a database  
 */  
connectToDatabase();  
  
/**  
 * This code will connect to our own database with our  
   database name,  
 * username and password  
 */
```

```
$con=mysqli_connect("db.dcs.aber.ac.uk",  
    cs g p 0 7 _ 1 3 _ 1 4 , "csadmgp07", "c54admgp07");
```

```
/*  
 * If the php fails to connect to the database this will  
   appear  
 */  
//heck connection  
if(mysqli_connect_errno())  
{  
    echo "Failed to connect to MySQL: " .  
        mysqli_connect_error();  
}  
mysqli_close($con);
```

#### 5.3.2.2 Append To The Server Database

```
$sql = "INSERT INTO List_of_Walks(title , shortDesc , longDesc , hours , distance)  
VALUES(' $title ', ' $short_desc ', ' $long_desc ', ' $hours ', ' $distance ')" ;  
  
mysqli_query($walk_conn,$sql);  
  
$walkID = mysqli_insert_id($walk_conn);  
  
foreach($route as $loc){  
  
    $longitude = $loc['longitude'];  
    $latitude = $loc['latitude'];  
    $time = $loc['time'];  
    $sql = "INSERT INTO Location(walkID , latitude , longitude , timestamp)VALUES  
        ( ' $walkID ', ' $latitude ', ' $longitude ', ' $time ')" ;  
  
    mysqli_query($walk_conn,$sql);  
  
    $locID = mysqli_insert_id($walk_conn);  
    if(isset($loc['description'])) {  
        $description = $loc['description'];  
        $name = $loc['title'];  
        $sql = "INSERT INTO Place_description(description , locationId , name  
            ) values(' $description ', ' $locID ', ' $name ')" ;  
        mysqli_query($walk_conn,$sql);  
    }
```



```
$placeId = mysqli_insert_id($walk_conn);
    if (isset($loc['images']))\{
        $photoCount = 0;
        foreach($loc['images'] as $image)\{

            $image = implode($image);

            $image = base64_decode($image);

            $photoName = $walkID . "_" . $locID . "_" .
                $placeId . "_" . $photoCount;

            file_put_contents("images/" . $photoName . ".
                jpg", $image);

            $photoCount++;

            $sql = "INSERT INTO Photo(photoName,
                placeId) values ('$photoName', '$placeId')
                ";

            mysqli_query($walk_conn, $sql);

        }

    }
```

## 5.4 TODO CHECK THE ALGORITHMS

## 5.5 Significant Data Structures

### 5.5.1 WalkModel

This is the most significant data structure in the Android application. It contains the information for the route taken, all of the GPS coordinates that the user has walked through, Points of interest.

### **5.5.2 LocationPoint**

This class is responsible for storing a point on the map. It has variables for longitude, latitude and a timestamp. After a GPS reading is taken for the current physical location is taken, it is put in an object of this class and stored in the WalkModel.

### **5.5.3 PointOfInterest**

This data structure is used when adding a point of interest. It holds information for the description and title of a POI. The class extends the LocationPoint so a POI can have location coordinates and a time stamp. This data structure is used when adding a point of interest. It holds information for the description and title of a POI. The class extends the LocationPoint so a POI can have location coordinates and a time stamp.

## 6 REFERENCES

- [1] Software Engineering Group Projects *Requirements Specification*. C. J. Price and B.P.Tiddeman, 1.2 (Release), 7 November 2013
- [2] Software Engineering Group Projects. *Design Specification Standards*. C. J. Price and N. W. Hardy, SE.QA.05A, 1.6. Release.
- [3] Software Engineering Group Projects *Project Plan*. Mosopefoluwa David Adejumo -all names-, 1.8 (Release). 6th November 2013

## 7 DOCUMENT HISTORY

Version	CFF No.	Date	Section Changed From Previous Version	Changed by
1.0	N/A	28/11/13	Created original document	HFB1
1.1	N/A	01/12/13	Added sections created by other members. Updated config reference Updated layout.	MDA
1.2	N/A	01/12/13	Fixed some formatting issues, added information to what fields will be used.	RYG1
1.3	N/A	04/12/13	Added a new sequence diagram and a description for section 1.2	MVZ
1.4	N/A	05/12/13	Updated section 1.1. Added descriptions to all sections	MDA
1.5	N/A	05/12/13	Added a sequence diagram for the web.	MRP2
1.6	N/A	05/12/13	Updated class diagram, added FileTransferManager interface.	HFB1
1.7	N/A	06/12/13	Added Apache HTTP Client description.	JAR39
1.8	N/A	06/12/13	Added methods to the MapView interface.	HFB1
1.9	N/A	06/12/13	Changed sequence diagram for web and added overall interaction sequence diagram	MVZ
2.0	N/A	06/12/13	Updated the Introduction section.	JAR39, MRP2
2.1	N/A	06/12/13	Added web app diagram and Significant algorithms	ZAL
2.2	N/A	06/12/13	Updated author list. Updated formatting. Merged different versions of the document	MDA
2.3	N/A	06/12/13	Added image reference numbers. Updated images from MWD5 and MAS69. Added missing images. Added images and descriptions to section 3. Added references.	MDA
2.4	N/A	06/12/13	Formatting corrections. Changed version	MDA
2.5	N/A	12/02/14	Re-wrote in LaTeX, removing feature creep	RYG1

2.6	N/A	13/02/14	Minor error checks and removal	MDA
2.7	N/A	13/02/14	Added Images	RYG1
2.8	N/A	13/02/14	Added Sequence Diagrams	RYG1
2.9	N/A	13/02/14	Updated database file saver algorithm	MDA

## **4 REFERENCES**

1

Earth image from [http://www.abcteach.com/free/e/earth\\_3d\\_rgb.jpg](http://www.abcteach.com/free/e/earth_3d_rgb.jpg)

## 5 DOCUMENT HISTORY

Version	CFF No.	Date	Section Changed From Previous Version	Changed by
1.0	N/A	13/02/2014	Set the Original Document Layout	RYG1
1.1	N/A	13/02/2014	Added other team members reports.	RYG1
1.2	N/A	13/02/2014	Added new reports and made small edits.	MDA
1.3	N/A	13/02/2014	Added new reports and Android maintenance details.	MDA
1.4	N/A	13/02/2014	Moved the Maintenance to the correct section of the document. Added the Web Application maintenance report.	RYG1
1.5	N/A	13/02/2014	Text correction. Added new paragraphs.	MDA
1.6	N/A	13/02/2014	Added the Database Manual Section	RYG1
1.7	N/A	13/02/2014	Added more information to the Database manual	RYG1
1.8	N/A	13/02/2014	Added more information to the Android manual	RYG1
1.9	N/A	13/02/2014	Grammar, syntax and spelling corrections. Capitalized user names. Updated system interaction and added use case descriptions. Added maintenance docs for field form and add fields PHP files.	MDA
2.0	N/A	13/02/2014	Updated Version numbering.	RYG1
2.1	N/A	13/02/2014	Formatting issues fixed. Updated evaluation	MDA
2.2	N/A	17/02/2014	Added Test Log	RYG1
2.2	N/A	17/02/2014	Edited some typos	RYG1