

Software Engineering Group Project

Final Report

Author: Group 14
Config. Ref.: SE-14-FINAL
Date: 2014-02-16
Version: 2.0
Status: Final

Department of Computer Science,
Aberystwyth University,
Aberystwyth,
Ceredigion, SY23 3DB,
U.K.

©Aberystwyth University 2014

CONTENTS

1	INTRODUCTION	5
1.1	Purpose of this Document	5
1.2	Scope	5
1.3	Objectives	5
2	MANAGEMENT SUMMARY	6
3	HISTORICAL ACCOUNT OF THE PROJECT	7
4	FINAL STATE OF THE PROJECT	9
4.1	Android App	9
4.1.1	Features	9
4.1.2	User Interface	9
4.1.3	GPS	9
4.1.4	Points Of Interest	9
4.1.5	Server Interaction	9
4.2	Website	10
4.2.1	Features	10
4.2.2	Walk List	10
4.2.3	Walk Viewer	10
4.3	Documentation	10
4.3.1	Test report	10
5	TEAM MEMBER PERFORMANCE	11
5.1	Josh Tumath (jmt14) - Project Leader	11
5.2	Theo Taylor (tht5) - Deputy Project Leader	11
5.3	Lars Lunde (lah25) - QA Manager	12
5.4	Jake Maguire (jam64) - Deputy QA Manager	12
5.5	James Berry (jab73)	13
5.6	Rob Bolton (rab26)	13
5.7	James Mellors (jam66)	14
5.8	Michael Oddie (meo9)	14
5.9	Dan Wakefield (daw46)	15
6	CRITICAL EVALUATION	16
7	APPENDIX A: PROJECT TEST REPORT	18
7.1	TEST TABLE	18
8	APPENDIX B: MAINTENANCE MANUAL	19
8.1	Walking Tour Creator	19
8.1.1	Program Description	19
8.1.2	Program Structure	19

8.1.3	Algorithms	19
8.1.4	Main data areas	19
8.1.5	Files	19
8.1.6	Interfaces	19
8.1.7	Suggestions for improvements	20
8.1.8	Things to watch for when making changes	20
8.1.9	Physical limitations of the program	20
8.1.10	Rebuilding and Testing	21
8.2	Walking Tour Displayer	22
8.2.1	Program description	22
8.2.2	Program structure	22
8.2.3	Algorithms	22
8.2.4	Main data areas	22
8.2.5	Files	22
8.2.6	Interfaces	22
8.2.7	Suggestions for improvements	23
8.2.8	Things to watch for when making changes	23
8.2.9	Physical limitations	23
8.2.10	Rebuilding and Testing	24
9	APPENDIX C: REVISED PROJECT PLAN	25
10	OVERVIEW OF PROPOSED SYSTEM	25
10.1	Project Overview	25
10.1.1	Android	25
10.1.2	PHP	25
10.1.3	JAVA	25
10.1.4	MySQL	25
10.2	High Level Architecture	26
10.2.1	Map API	26
10.2.2	App UI	26
10.2.3	Internet Usage	26
10.2.4	Target Users	26
11	USE-CASES	27
11.1	App	27
11.1.1	Create Walk (FR1)	28
11.1.2	Upload Walk (FR6)	28
11.1.3	Switch to/from app	28
11.2	Server	28
11.2.1	Store walk	29
11.2.2	Delete walk	29
11.2.3	Browse walks	29

12 USER INTERFACE DESIGN	29
12.1 Android Application UI Design	29
12.1.1 Main Menu (FR1)	30
12.1.2 Create New Route (FR2)	31
12.1.3 Create Route (FR1, FR5, FR6)	32
12.1.4 Adding Location (FR3, FR4, FR9)	33
12.2 Web Application Design	34
12.2.1 HomePage	34
13 GANTT CHART	37
14 RISK ANALYSIS	39
14.1 Gold plating	39
14.2 Illnesses	39
14.3 Time management	39
14.4 Compromising on design	39
14.5 Backups	39
14.6 Released project has low quality	40
14.7 Conflicting Ideas	40
15 APPENDIX D: REVISED DESIGN SPECIFICATION	41
15.1 DECOMPOSITION DESCRIPTION	41
15.2 Programs in system	41
15.2.1 Walking Tour Creator	41
15.2.2 Walking Tour Displayer	41
15.3 Significant classes in each program	42
15.3.1 Significant classes in Walking Tour Creator	42
15.3.2 Significant functions in Walking Tour Displayer	42
15.4 Mapping requirements onto classes	42
15.4.1 DEPENDENCY DESCRIPTION	43
15.4.2 Component Diagrams	43
15.4.3 Component Diagram for Walking Tour Creator	43
15.4.4 Component Diagram for Walking Tour Displayer	44
15.4.5 Inheritance Relationships	45
15.5 INTERFACE DESCRIPTION	46
15.5.1 MainAppActivity interface specification	46
15.5.2 WalkDetailsActivity interface specification	46
15.5.3 Walk interface specification	49
15.6 DETAILED DESIGN	52
15.6.1 Sequence diagrams	52
15.6.2 Significant algorithms	53
15.6.3 Significant data structures	53
REFERENCES	54

DOCUMENT HISTORY

54

1 INTRODUCTION

1.1 Purpose of this Document

The purpose of this document is to give a view of the final state of the project. It will cover all aspects such as changes to the project plan that were made while coding and Test tables to ensure the app worked as expected and group reflection.

1.2 Scope

Everyone who wants a clear understanding of how the team functioned during this project should read this document. Individual members of the team looking for places that they could improve their teamwork skills may find suggestions in here as well.

1.3 Objectives

1. Show changes made in the project over its lifecycle.
2. Show how the project was collaborated on with reasons for the choices made

2 MANAGEMENT SUMMARY

The group achieved a working version of the Walking Tour Creator (WTC) that is capable of providing the user the means to record a walk and add a description about the walk, add points of interests at any point of the walk, also to be able to take pictures and have them saved and upload it to a database for viewing, in other words a fully functional personal Walking Tour Creator.

The group was able to verify that documents were in good state after the drafts had been reviewed. The majority of our documents got a grade of B and only one with a grade of C for the draft. The document with the C grade was in a bad state so what the group did was go and fix the issues with it to get the grade up. This doesn't mean that we left the other documents, they all had some minor issues so we delegated the fixes throughout the team to get them into a good state.

The group had minor issues with collaboration and version control. This was mainly due to group members struggling to use Git at the beginning of the project, so it was difficult to get all of the documents together. The way the group overcame these problems is that Dan Wakefield wrote a tutorial and put it on the repository. We also switched to using the free online L^AT_EX editor writeL^AT_EX for producing documentation, as it allowed for easy collaboration (but consequently discouraged frequent commits to the repository).

When developing the Android app, there was a major bug where if the user changed the orientation of their phone during an operation, the app would lose the asynchronous task, causing the task to attempt to open popups on an Activity that no longer existed. To overcome this, the Android sub-team passed the AsyncTask to a newly opened Activity each time this happened and updated the AsyncTask for the new Activity.

When developing the Web site, one of the problems that the Web sub-team had was deciding how to store in a database whether a coordinate was a point-of-interest or just an ordinary coordinate along the route. They overcame this problem by meeting together to work on it. They decided to add another column in the database which was a simple boolean value to represent whether a coordinate was a point-of-interest.

On a whole, the entire team performed with a hard-working attitude towards the goals of the project. While there were some issues with some group members not pulling their weight initially, by the end of the project each member displayed a passion and enthusiasm towards their work. Communication between the group was good particularly during integration and testing week where everyone showed a team spirit.

3 HISTORICAL ACCOUNT OF THE PROJECT

At the beginning of the group work, the group decided on major roles. These included Project Leader (Josh Tumath), Deputy Project Leader (Theo Taylor), QA Manager (Lars Lunde), Deputy QA Manager (Jake Maguire), Chief Architect (Rob Bolton) and Version Control Manager (Daniel Wakefield). Later during the prototyping phase of the project, the group was split into two development teams: the Android team and the Web team. These sub-teams were led by Rob Bolton and Jake Maguire, respectively.

The group met during Fridays for general meetings with the Project Manager. Additionally, there would be mid-week meetings, typically on Wednesdays, where the group would normally discuss the progress of work to be completed by the following Friday meeting. Facebook was the preferred platform for communication.

The first task the group had was to write the Project Plan document. This involved outlining what is to be produced and the architecture of both the app and website. This gave the group a first attempt at working together to produce something. Each section was given to different group member to work on. If a section required more work compared to others, more group members were assigned to it.

Following this, the group was tasked with two documents to produce simultaneously. These were the Test Specification and the Design Specification. During this time, many members of the group had other assignments. Fortunately, the test Specification did not require too much time to write. Therefore, only Dan Wakefield was tasked to write it. The Deputy Project Leader led the creation of the Design specification with James Mellors and Michel Oddie. They delegated sections of the Design specification to each other. James Mellors was assigned to do the dependency description, Theo Taylor was assigned to do the introduction, decomposition description and the interface description, and Michel Oddie was assigned to do the detailed design.

Before the submission of each of these documents, a formal review meeting would be held by the Project Leader and QA Manager. Following this, issues would be created on GitHub listing the necessary changes.

During the creation of the Design Specification, the members of the group who were not working on it were simultaneously producing the prototypes for Android application, the front-end of the Web site and the database. This ensured every member of the group had work while also allowing for efficiency in the rate at which work was completed. By Christmas, both the Android sub-team and Web site sub-team were able to give a good demonstration of the front-ends of their software to the Project Manager.

Following the Christmas break, integration and testing week itself was a very busy week because of the great amount of work that needed to be completed in time for

the Acceptance Test with the Project Manager at Friday afternoon. The way the group tackled it was to begin with a meeting together to talk about how far along we are and what tasks need to be done. The group made a plan of what needed to be done for each day. After the meeting, the group looked for a suitable workspace to allow them to work together.

Once the group was set up, they separated into the two sub-teams; as was the case during the prototyping stage. The sub-team leaders then delegated all the tasks that needed to be completed for the day. This process of meetings and task delegation was repeated each day of integration and testing week and the group found it to be a very efficient way of handling the work load.

Towards the end of the week, the app and Web site were coming together. This meant that some group members had no tasks assigned to them and therefore worked on revisions of previously submitted documentation to be efficient with their time.

During the Acceptance Test, the Project Manager was pleased that the majority of the produced software was fully functional.

4 FINAL STATE OF THE PROJECT

4.1 Android App

4.1.1 Features

The features of the app were mostly correct. The only feature that was missing was the ability to edit previous Points Of Interest. Most features were implemented correctly, except that the app is meant to tell you if you try to take a Point Of Interest without any GPS signal, and prevent a Point Of Interest from being created. Instead, our app uses the user's last known location for the new Point Of Interest. GPS, the UI, and server interaction all work correctly.

4.1.2 User Interface

The UI works correctly. The user cannot enter any invalid data and are told what is wrong when they attempt to do so. The user cannot navigate to displays which would not make sense e.g. returning to the "Create a new walk" Activity whilst a current walk is being created. The UI can properly handle the user pressing the home button or rotating the screen on certain windows. These are any which require input before closing, and cannot be dismissed. Any windows which can be dismissed will not return upon the user opening the app again.

4.1.3 GPS

The GPS collects data correctly and filters any points which are not accurate enough. The only issue with the GPS is that it will automatically use the last known position if no new data is available, after warning the user once that they need GPS data.

4.1.4 Points Of Interest

The only issue with the Points Of Interest is that they were meant to have "one or more" images, but we allowed the user to have no images as well. The user can take as many pictures as they want, and when adding pictures to a Point Of Interest, they can look through thumbnails of pictures they've already taken. Tapping on a picture deletes it. This was our interpretation of editing a Point Of Interest.

4.1.5 Server Interaction

Aside from a memory issue, the server interaction works well. It attempts to upload the image to the server and if anything other than an http "200 OK" is received

then the user is told that their upload failed. The user can attempt to upload again if they want. This means that if anything goes wrong with sending the image then the user will be told - this includes website issues and connection issues.

4.2 Website

4.2.1 Features

All features work correctly. The only potential issue would be if two separately uploaded images had the same hash. If this were to occur then the old image would be overridden with the new one. Parsing of the uploaded JSON and saving this data in the database works correctly.

4.2.2 Walk List

The page which displays the list of all recorded walks works correctly, and they are shown in a large grid with the walk's name and a picture from it. If a walk has no image then the app/webiste logo is displayed instead.

4.2.3 Walk Viewer

The walk viewer works correctly. On the left of the page it gives the user the walk's name, short & long descriptions, the total distance covered, how long the walk took. On the right the walk is displayed using Google Maps. Walks are correctly displayed in this area, with each GPS point and Point Of Interest being linked.

4.3 Documentation

With the exception of the Test Report, all of the documentation is up to standard.

4.3.1 Test report

There were no references to any issues on github in the test report.

5 TEAM MEMBER PERFORMANCE

5.1 Josh Tumath (jmt14) - Project Leader

Josh was given the role of Project Leader, where he led and directed the team in preparation for the final implementation of the software. During integration and testing week, he was also a Web Developer in the Web site sub-team, due to his previous experience in Web programming and design.

It was remarked by other project members that Josh was very well-organised throughout the duration of the project, planning the team's actions well. He was generally very helpful towards others; being easy to work with and not forceful in regards to pushing for completion of work.

However, Josh could be very unclear about deadlines, failing to set targets for when particular sections of work should be completed. Additionally, he did not delegate work evenly, leaving some members to be completing more work than others.

The above report was agreed to by all members of the group.

5.2 Theo Taylor (tth5) - Deputy Project Leader

Theo was given the role of Deputy Project Leader, where he led group meetings in times when the Project Leader was unavailable. He also led ad-hoc sub-teams that worked on a specific task for the project, such as the Design Specification. He was also an Android Developer in the Android sub-team.

During integration and testing week, Theo worked on certain features in the Android app, such as disabling the back button and creating Bundles for communication between Activities. He struggled in this area due to a lack of experience with Android development.

However, overall Theo could be quite unreliable at times. Additionally, when leading members of the team, he failed to push for high quality of work - being too lenient towards slacking and meeting deadlines. For example, the Design Specification required many last-minute changes before it was handed in.

The above report was agreed to by the group member.

5.3 Lars Lunde (lah25) - QA Manager

Lars was given the role of QA Manager, where he took minutes in meetings, held formal reviews and aided the group in meeting the QA standards.

As Lars's native language is not English, he was concerned this may affect some of the duties of his role. However, this was not an issue. For the most part, he followed through with his role effectively, and was willing to put in the hours for the sake of the group. During integration and testing week, he was very focused and worked closely with Rob on the development of the Android app, while always ensuring he had work to do.

The main issue with Lars was that he could be very slow at publishing meeting minutes - usually taking more than 24 hours after the meeting. Additionally, as the weeks progressed, his minutes became less detailed, which affected some members understanding of their tasks following the meetings, or reminding themselves of the items discussed during the meetings. Furthermore, he did not enforce the Java coding standards during integration and testing week.

The above report was agreed to by the group member.

5.4 Jake Maguire (jam64) - Deputy QA Manager

Jake was given the role of Deputy QA Manager, where he mainly took minutes in meetings when Lars was no able to, and prepared formal reviews when the QA Manager was too busy. During the prototyping and implementation stages of the project, he was also tasked as Lead Developer of the Web site sub-team, due to his previous experience in server-side programming. Additionally, he allowed us to use his Web server for hosting the Web site.

From the beginning of the project, Jake had been very helpful and involved. When taking the minutes for meetings, he would usually be very prompt in uploading them. He also worked hard on developing the user interface design of both the Android app and the Web site. Jake had very good attendance of meetings and in the computer rooms during implementation and testing week.

Jake had proven himself to be very efficient in leading the Web site sub-team, as the majority of their prototyping and final implementation has been completed by Christmas. However, it is possible that he took on more work himself and not enough for group member James Berry.

The above report was agreed to by the group member.

5.5 James Berry (jab73)

James was given the role of Web Developer in the Web site sub-team, where he aided in the early prototyping and implementation of both the client-side and server-side systems. This suited James's skillset, having studied Web development in previous university modules.

James was reliable in both attendance and completion of work. At the beginning of the project, he wrote the risk analysis and a schema for the Tour database, which were done to a high standard. Working with Jake Maguire, they worked efficiently to complete a prototype of the Web site before Christmas.

During integration and testing week, James put a lot of work into researching how to solve certain problems, such as decoding base64 encoded images on the server-side. Additionally, he aided with the Google Maps APIs to ensure the front-end was feature-complete. Finally, James was very helpful in completing the PHPdoc.

The above report was agreed to by the group member.

5.6 Rob Bolton (rab26)

Rob was given the role of Chief Architect, where he was expected to handle the overall design of the system from initial planning to implementation. In early meetings during the project, he expressed an interest and ability in planning class structures and imagining relationships between parts of the system. During the prototyping and implementation stages of the project, he was also tasked as Lead Developer of the Android sub-team. This required him to gain a detailed understanding in the Android SDK - building on his experience in using the Java programming language.

Rob displayed a great passion for the work of the project and a strong desire for work to be completed to a high standard. He explained that he has enjoyed the experience; particularly during integration and testing week, when he stated that he thrived from the pressures of working towards a deadline and being heavily relied upon by other members of the team. He was also very helpful and patient with other Android developers in the team when allocating and expecting work from them.

However, he was perhaps too willing to take on too much work for himself. Much of the work on prototyping for the Android app was completed by him - as was much of the coding for the completed app. One night, he and Dan Wakefield stayed awake for over 24 hours to work on the final features for the app.

The above report was agreed to by the group member.

5.7 James Mellors (jam66)

James was given the role of Android Developer in the Android sub-team, where he helped with smaller changes in the development of the Android app. However, James's more significant contributions were towards documentation efforts, where he worked on various sections of the Project Plan and Design Specification. In addition, he designed the logo of the software.

James very quickly proved himself to be reliable and prompt when the project began. Work he was given - such as gathering information on the Google Maps APIs or writing about the high-level architecture of the Android app - was completed promptly. He was always seeking or work to do.

During the former half of integration and testing week, James did some peer programming on adding certain features to the Android app, such as checking if a device supports location services. In the latter half, he worked on the UI and updating the design specification.

The above report was agreed to by the group member.

5.8 Michael Oddie (meo9)

Oddie was given the role of Android Developer in the Android sub-team, where he aided in the implementation of the Android app. However, he was initially tasked to create the initial wireframe design for the Web site and certain sections of the Design Specification. During integration and testing week, he worked mainly on testing and documentation.

Initially, Oddie's meeting attendance and contribution effort could be quite poor. Part of this was due to society events clashing with the group's unofficial weekly meetings. Other times, however, work was simply not being completed or completed late - namely specific sections of the Project Plan he was tasked on completing. This improved over time, and, by the latter half of Semester 1, he was contributing much more.

Oddie performed greatly during integration and testing week, when he was very enthusiastic about working on the Android app and testing it very extensively. This showed he thrives much better in a constant working environment. He was a great value to the team, working with other members of the group to help them and ensure that they were producing a good quality of work.

The above report was agreed to by the group member.

5.9 Dan Wakefield (daw46)

Dan was given the role of Version Control Manager, where he acted as a consultant for the group when using the project repository on GitHub. This required him to attend a lecture on the subject and maintain the health of the tree during development. He also aided in the development of both the Android app and the Web site.

Though Dan lacked access to an internet connection off-campus during Semester 1, he still put in a great deal of work and contribution towards the project. Over the course of the project, Dans experience with Git improved greatly, and his help was invaluable by integration and testing week.

He worked very hard on the project. Most notably, he devoted a lot of time to helping Jake Maguire and James Berry to fix a critical bug in the communication between the Android app and Web server. At one point, he stayed awake overnight with Rob Bolton trying to refactor the software when the completion of some features was behind schedule.

The above report was agreed to by the group member.

6 CRITICAL EVALUATION

In terms of final product, the whole team performed efficiently and competently. The project was finished on time with nearly all the functionality that was specified at the start. Our initial designs were thorough to the point where the implementation of the project did not encounter errors or big changes from what we decided early on.

Each member played their part in the project with everyone coming together at the end to get everything done. The Project Leader made sure that progress was being made in the appropriate places, that all team members had a job to do and that deadlines were set to ensure the project would be finished on time. Everyone pulled their weight and was committed to getting their part done for the rest of the team.

There were a few issues with delegation with those in various leadership roles preferring to tackle most things on their own rather than delegating tasks to other member on their team. With more organised delegating the project may have been completed quicker and a rush at the end to get everything completed would have been avoided.

Communication could have been better in terms of responding promptly to questions and tasks on online platforms such as GitHub and Facebook. It could be hard getting a response from all members quickly when a question was asked or tasks assigned on GitHub would be left open even if completed.

Improvements could have been made during our project lifecycle. Git was an under-utilised tool by some members, especially by the Web team. Due to a lack of knowledge, complications with merges and branches were apparent during implementation. This led to Git being side-lined and meant features such as rolling back to previous versions and allowing multiple members to work on the same file could not be used effectively.

Our project also had some missing functionality. The ability to edit locations on a walk was not present, which was reflected in our acceptance testing. This was more a oversight from the team rather than a technical issue, but meant that our Android app did not do everything that was asked by the Requirements Specification.

The team learned about the importance of communication and how critical it is to keep in contact to track progress and know what to do. The minutes had to be taken with detail so that members who were unable to attend the meeting still knew what task they would have to complete that week. Both the Web sub-team and Android sub-team leaders had to communicate to make sure that things were set up at their own ends to allow tests to take place between the two platforms, to check they were interacting with each other correctly. The Project Leader had to communicate tasks to each member giving them specific targets of deadlines and the content of their work. Feedback from members would also have to be given to the Project Leader in

terms of whether or not these targets were attainable or if they had disagreements with what had been set.

Another lesson learned was the equal importance of all stages of the project life cycle. Our emphasis on a proficient design meant that our implementation ran into little problem and allowed us to finish on time. The analysis we carried out early on meant that most functionality was in place as we knew what was required from our end programs, making our project a viable solution to the specification given to us at the start.

7 APPENDIX A: PROJECT TEST REPORT

7.1 TEST TABLE

Test Ref	Result
SE-F-001	PASS
SE-F-002	PASS
SE-F-003	PASS
SE-F-004	PASS
SE-F-005	PASS
SE-F-006	PASS
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-W-001	PASS
SE-W-002	PASS
SE-W-003	PASS
SE-W-004	PASS

8 APPENDIX B: MAINTENANCE MANUAL

8.1 Walking Tour Creator

8.1.1 Program Description

This application allows a user to record a walk, tracking their GPS coordinates and allowing them to add named points of interest with pictures. It can upload this walk to a server once it is finished.

8.1.2 Program Structure

The component diagram can be found in *Design Specification 3.1.1* The sequence diagram can be found in *Design Specification 5.1* A list and definition of the classes & interfaces and their methods used can be found in the Design Specification:

- **MainAppActivity** 4.1

8.1.3 Algorithms

The significant algorithms can be found in *Design Specification 5.2*

8.1.4 Main data areas

The main data areas can be found in *Design Specification 5.3*

8.1.5 Files

Our application does not use any specific files. The only files it relies on being there are the photos taken for the Points Of Interest. The path returned by the camera intent after taking the photo is used for these. If the application cannot find the file, then the photo will not be added to the JSON before sending it to the server.

8.1.6 Interfaces

The only requirements for the application to successfully function are that the user has a camera, internet connection, and GPS running. If the user does not have a camera then they cannot install the application. If the user does not have GPS running then the application will not let them create a new walk. Turning this off whilst the application is running will prevent the application from gaining any

more GPS data, and it will continue to use the last known location for new Points Of Interest. If the user does not have an internet connection then saving and uploading the walk to the server will fail.

8.1.7 Suggestions for improvements

Several improvements could be made to the application. At the moment there is no way to edit points of interest after they have been created. In order to fix this, I would recommend having a new activity using a list view and an adapter, which takes the Walk in its bundle to see which points of interest we have.

When the methods converting the walk to JSON were created they were made as a quick solution to upload the walk. As a result of this they attempt to compile the whole walk in one go. This means that the entire JSON object of the walk is in memory. Two solutions which would help are downscaling the image before converting it to base 64, and finishing the code which converts and sends the JSON one image at a time in a stream. Realistically, both of these should be completed for good practice, as we do not need high resolution images on the website, and storing the whole JSON object in memory can cause phones with small amounts of memory to crash.

8.1.8 Things to watch for when making changes

There are a couple of places in which you need to take care when changing. Anything dealing with an AsyncTask needs to ensure that if it displays any dialogs that the context it originally had is still available for use. An example of doing this correctly is where we call "setDialogsAndNotify" on our walkUploader in order to give it the new dialogs for the current Activity. Another area in which it is important to take care when changing is the JsonPackager. Any changes to the format of the JSON in here need to be reflected in the website's code which deals with the received JSON.

8.1.9 Physical limitations of the program

There are a few physical limitations of the program. In order to be able to install it, the user's device needs to have a camera. If the user does not have GPS when attempting to create a new walk then they will not be able to create one. If the user does not have enough memory to store all of the images they took, then the application will crash upon compiling the JSON representation of the walk. Another physical limitation is that the application will not accept any GPS data which is too inaccurate. This uses the Location.getAccuracy() method and filters out any results which are greater than 15 meters. This method returns *accuracy* in meters, which is a circle of radius *accuracy* which has a 68% probability that the device

was within the bounds of this circle. Due to this, if the device never receives any Location objects which are at least within 15 meters accurate then no GPS data will be recorded. The last limitation is that the user needs an internet connection to upload the walk. The application will keep the walk until the user successfully uploads it or cancels it.

8.1.10 Rebuilding and Testing

All of the documentation is either in LaTeX or Javadoc. Any pdftoLaTeX compiler should work for rebuilding any of the LaTeX documentation. The Javadoc can be rebuilt using Eclipse by selecting *File > Export... > Java > Javadoc* and following through with the wizard. Another alternative is to use the "javadoc" command.

8.2 Walking Tour Displayer

8.2.1 Program description

This Web site receives tours from the Walking Tour Creator and stores the data within a MySQL database. Web site visitors can view a list of the tours and view each of them on a detailed map that displays the route with points-of-interest along it.

8.2.2 Program structure

A list of significant functions can be found at *Design Specification 2.2.2* and a component diagram can be found at *Design Specification 3.1.2*

8.2.3 Algorithms

The significant algorithms can be found in *Design Specification 5.2*

8.2.4 Main data areas

The main data areas can be found in *JSON QA 2.1, 2.2*

8.2.5 Files

There are two main files that the walking tour displayer relies on. The first being the css file for the website, which is stored in the directory /projects/wtc/css and is responsible for the layout of the website. The second directory being /projects/wtc/images which contains all of the images for the webpage. This directory also includes another subdirectory called /images/walkimages, which is where all of the images sent from the app are stored.

8.2.6 Interfaces

The walking tour displayer has been tested on many different browsers, including mobile devices. On mobile devices the user is unable to scroll on a marker if there are many different pictures. The only browser that the website has not been tested on is IE6 so the user may encounter problems on that particular browser.

8.2.7 Suggestions for improvements

You could entertain the idea of adding a search bar to the website. This would be particularly useful when the website is populated with a large amount of walks. You could add a search bar where the user can search for a particular name of a walk that they wish to view. An SQL statement would then be used to search through the database and return the results of the search. Ajax could be used to create a more dynamic webpage, where the results for the search would get returned while a user is still typing in their search. Other search features that could be implemented would be to search walks by the distance or by the time they take to complete.

Another feature that could be added is to print out the directions for the walk next to the map. I imagine that there is probably a google api which can be used to do this automatically, which would save quite a bit of time. This api would check all of the coordinates and guide the user through their walk, giving them directions for the turnings that they need to take.

A pagination should also be considered where large amounts of data are being used so that the user would not have to endlessly scroll through all of the walks, but rather go through pages with a certain amount of walks per page. This could be implemented using a php script which takes in all of the walks from the database and then divides them all by a certain number, say 9, and splits all of the walks into several pages of 9 walks.

A final feature that should be added to the website is to have a clear start and a finish of the walk. At the moment the map only displays all of the coordinates and locations of a walk, but it is not clear where the walks start and end. The only clue is that the centre of the map is set to the first coordinate. However this can cause confusion, particularly if the walk is a giant loop which starts and finishes in the same place. To overcome this a marker, similar to those used for a location but different in some way, should be used to show where the first coordinate is and the last coordinate is.

8.2.8 Things to watch for when making changes

It should be noted that if you wish to change any of the SQL querying, we have used a PDO abstraction layer to implement these, rather than using the MySQLi APIs.

8.2.9 Physical limitations

As our Website has been written on a personal server, it should be noted that if it is intended for this website to handle a greater amount of traffic then it would be necessary to move to a new host server, as the current server only allows for

a certain amount. There is also a limit to the disk space available, so if there is expected to be a large amount of data stored on the server, particularly with image data, then extra disk space will most likely be needed to stop this from becoming a problem.

8.2.10 Rebuilding and Testing

As this has been written on a personal server, the files cannot be easily viewed. However they are all available at "www.jakemaguire.co.uk/projects/wtc".

In terms of testing the website, the best thing to do would be to run the app and put some dummy data in, including some locations with pictures to see that they write correctly to the database and then check the website homepage to see that it then displayed correctly.

9 APPENDIX C: REVISED PROJECT PLAN

10 OVERVIEW OF PROPOSED SYSTEM

10.1 Project Overview

Our app that we are going to produce will be android based. Our app is going to be a walking tour app that will help people make their own walks and to be able to edit and save them to view again a later day.

Platforms to be used and Architecture for the app: gskip

10.1.1 Android

The assignment stated that we have to make an app for android phones so that the platform we will be using, as well as using android platform version 2.2

10.1.2 PHP

We will be using PHP for the website and for the phone to communicate to the server (database). I think we will find it easier to use than javascript and it will be a good opportunity to get better at writing in that language. We plan to host the Website on our group member Jake Maguire's personal VPS and the web address for the project will be www.jakemaguire.co.uk/projects/wtc

10.1.3 JAVA

Java is what we will be using to program the app. We just need to decide what we will use to write it in, there is a choice between eclipse with the android plug in or android studio. Our preference will be to use android studio

10.1.4 MySQL

MySQL is what we will be using to make the database to be able to store the walks that have been created.

10.2 High Level Architecture

10.2.1 Map API

The map will be a big part of our app because we will need to display the routes and be able to let people see where there going and make their own route. We will be using the map api on our web server to display a selected route from the database.

10.2.2 App UI

The Menu / UI, first we will have the main page where you will be able to click on the button create new route to be able to start mapping the path while you are walking, right underneath this button their will be 2 more preferences and exit. All together there will be up to 4 menus to this app: main page, naming of the walk, route map editor and the adding location.

10.2.3 Internet Usage

The app will only need to use the wifi or 3G/4G to be able to improve the tracking capability of the app and to be able to upload the finished walks onto the website.

10.2.4 Target Users

Our target users are people that enjoy going on walks and would like to start marking out their walks and to be able to save them so they can look back at them and do that walk a later day.

11 USE-CASES

The use cases for our application will be split into two cases. These are the "App" and the "Server". The App use case describes all interaction with the application itself and, as such, only contains interaction with the mobile user. The server is not listed as interacting with the app here as interaction between the app and the server is functionally one-way (the app pushes walks to the server, but cannot retrieve or view walks).

The Server use case describes all interaction with the server, which includes the web user, app, and administrative user.

11.1 App

The app's use cases include:

11.1.1 Create Walk (FR1)

Here the user starts to create a walk. Once this interaction is started (by pressing the button in the app's main menu), and a valid walk name is provided, the user's location is tracked and periodically added to the on-going walk's list of GPS points. As a part of this case the user may choose to:

- Add location: This will allow the user to add the current location to the walk as a point of interest. (FR3)
 - Take picture: The user can take a picture to add to this location. (FR4)
 - Set details: The user can set the location's details (name, description).
- Set details: The user can set the walk's details (title, short description(100 characters), long description(1000 characters). (FR2)
- Cancel walk: The user may choose at any point to cancel their walk. This will stop recording their position and drop all current data of the on-going walk. (FR5)

11.1.2 Upload Walk (FR6)

This is used at the end of the "create walk" case in order to upload the walk to the server.

11.1.3 Switch to/from app

Whenever the user switches from the WTC app to another app, it will store the current walk's data. When they switch back it will retrieve the previously stored data. (FR7)

11.2 Server

The server's use cases include several actors.

The "App" actor is an instance of the WTC app on a client's phone and can upload walks.

The "User" is a client accessing the WTC website in a web browser. They may view a list of walks, select one, and view its details.

The "Administrative User" is a user with the power to delete walks as well as view

them.

The server's use cases include:

11.2.1 Store walk

A walk recieved from a client using the WTC app can be stored in the database of walks. . (FR9)

11.2.2 Delete walk

This use case allows for administrative users to delete any walks which they feel do not belong in the database. This would include any corrupted/broken walks and walks which contain obscene or offensive content.

11.2.3 Browse walks

This use case provides html pages to users which contains a list of all the walks in the database. The cases in this case are:

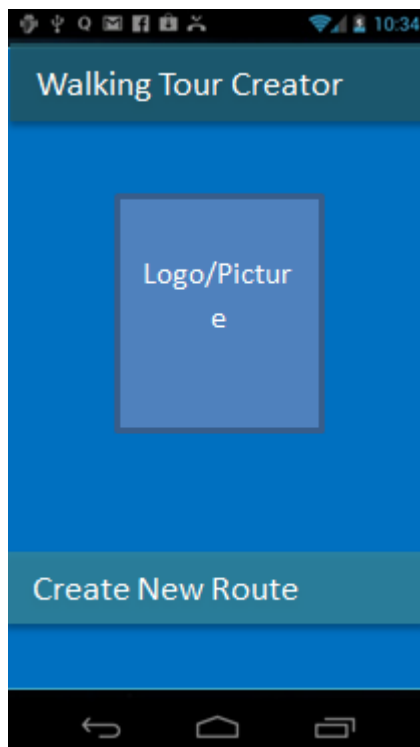
- List walks: This provides a list of all the stored walks.
- View walk: This will allow the user to view a specific walk, including its name, short and long descriptions, and its points of interest. (FR8)
 - View location: This will allow a user to look at a specific point of interest. This includes its name, description, location, and any picture it has.

12 USER INTERFACE DESIGN

12.1 Android Application UI Design

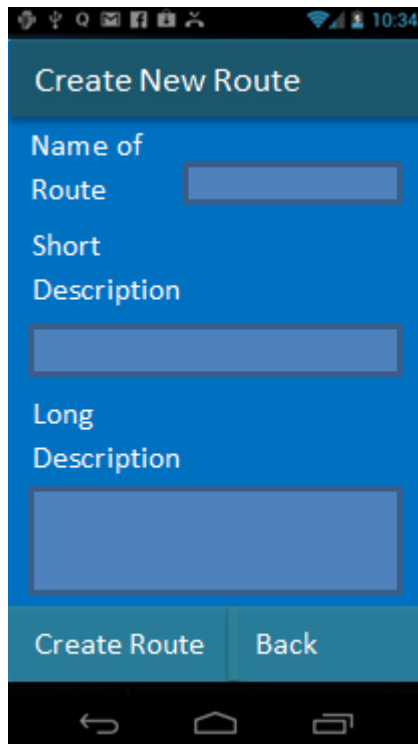
This section of the documentation will give an idea for how the UI of both the Android application and the web application will look like and also the flow of the program from screen to screen. These designs are not final due to technical restraints we may face later on in development but they will give a feel for the sort of layout and feel we are working towards.

12.1.1 Main Menu (FR1)



The first screen that the user is presented with is the main menu. This screen has one main option for the user to select. The user can create a new route which allows them to progress into the application and start recording their routes. There will also be some sort of logo or picture above these options.

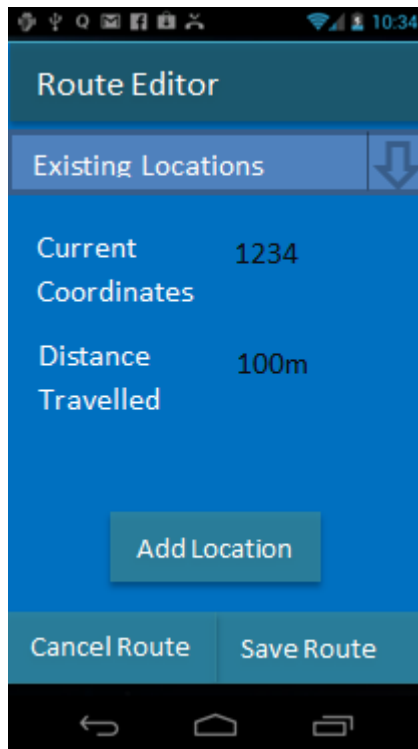
12.1.2 Create New Route (FR2)



On the Create New Route screen the user is able to set up a new route for them to record. They will be presented with a field to enter the name of the route, a short description of the route and a longer description. The short description will be restricted to up to 100 characters while the long description is up to 1000 characters. The user will be required to fill in all of the fields, as per the functional requirements.

There will be two buttons at the bottom which allow the route to be created or to go back to the previous page.

12.1.3 Create Route (FR1, FR5, FR6)

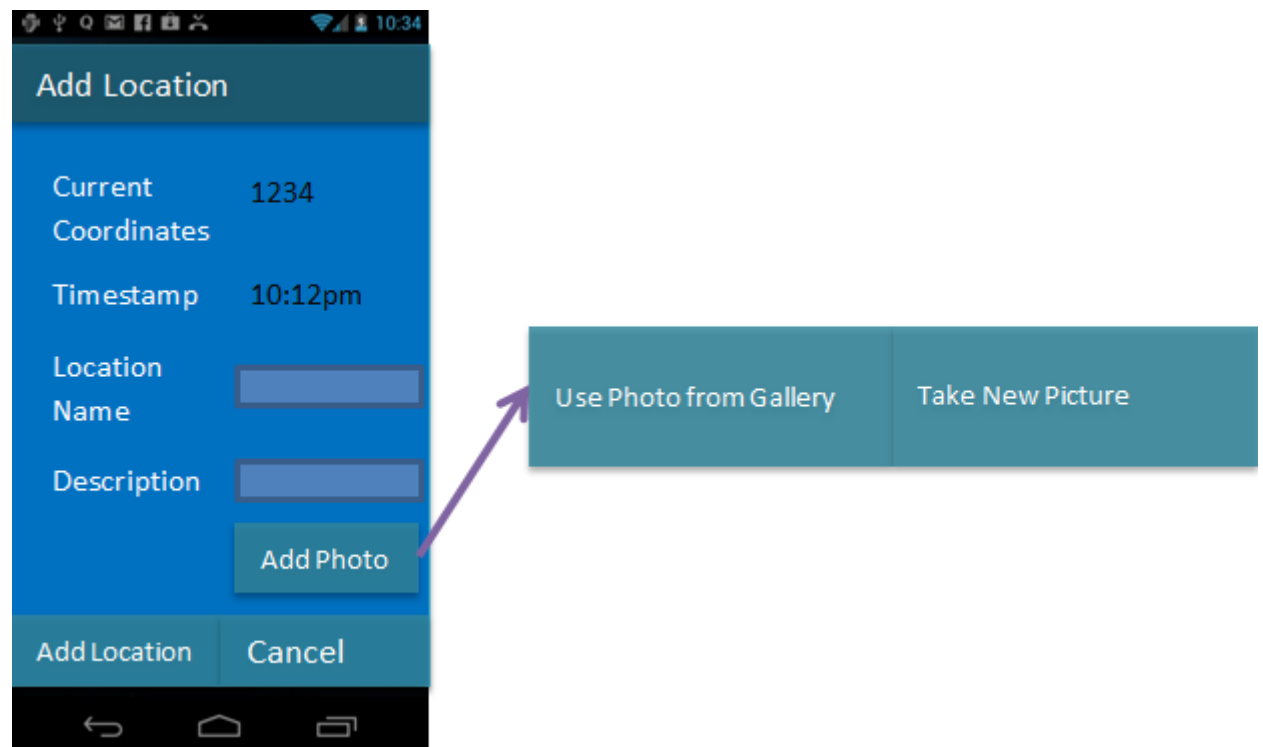


When the route has been created the user will be presented with their current coordinates which will update in real time as the user moves around, their current distance travelled from the starting location, along with options to add a location and save the current route. The existing locations drop down will allow the editing

of locations that have already been added, letting the user change the name, coordinates and description of a previous location. Adding a location will save the current

position and move to the next screen where they can specify information about that location. Once the user has finished recording and has added their locations, the user can press the save location button at the bottom of the screen to send this data to the server where it can be read by the web application. They can also cancel the walk if they do not require it after recording.

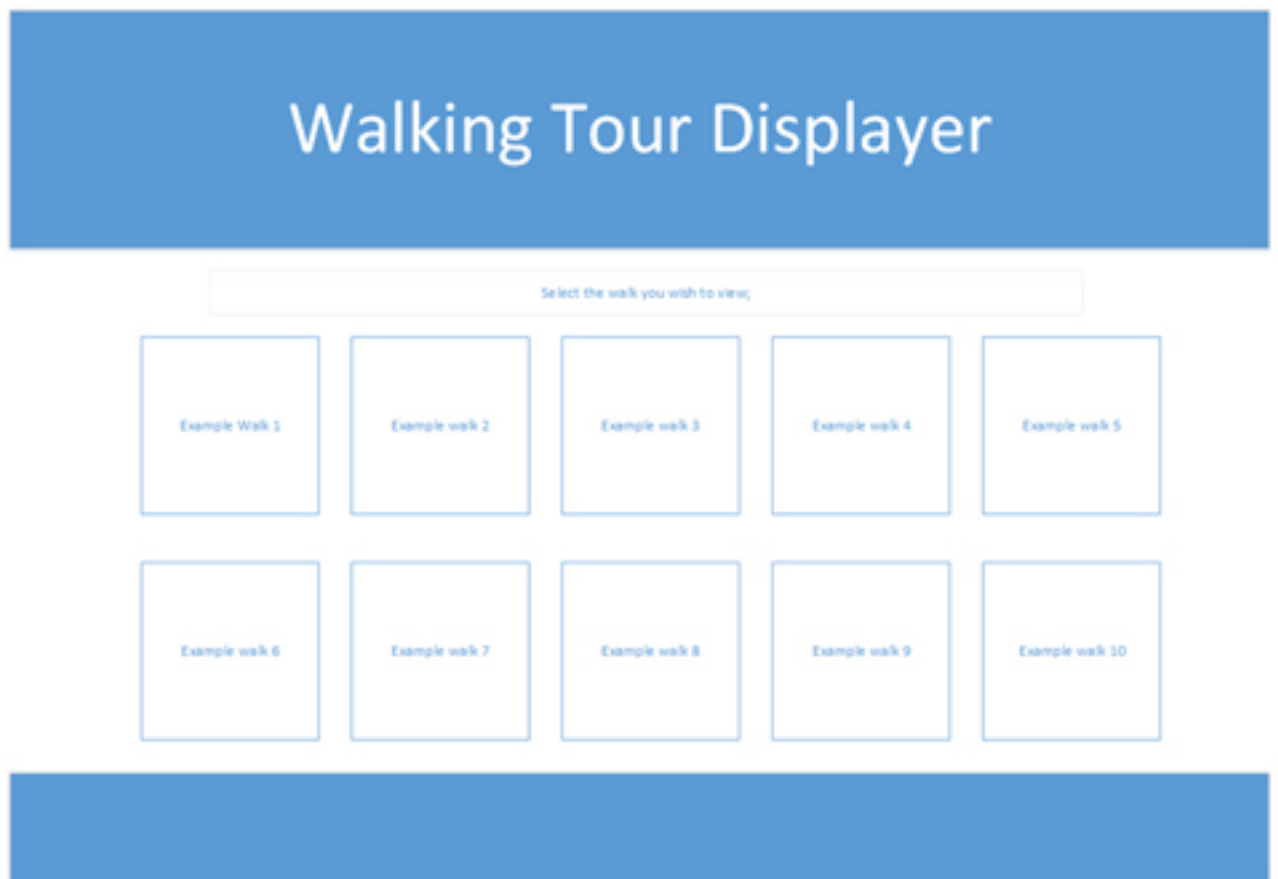
12.1.4 Adding Location (FR3, FR4, FR9)



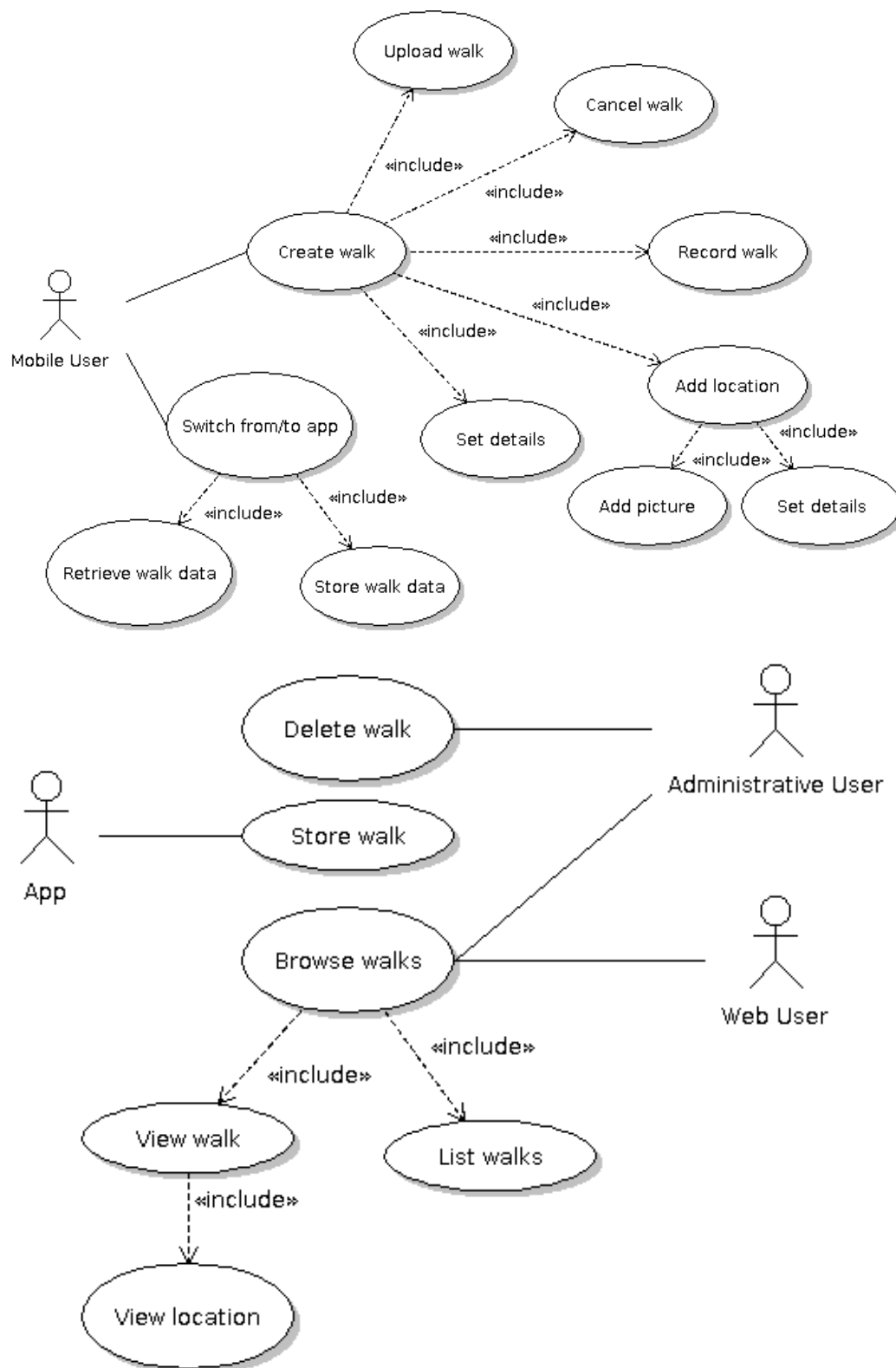
When adding the location the user is required to enter some information along with some already given to you. The user's current coordinates are displayed along with the current time the location was added. The user then enters the name of the location and a short description. There is also an option to add a photo where a photo can be taken using the camera option or use a photo already in the gallery.

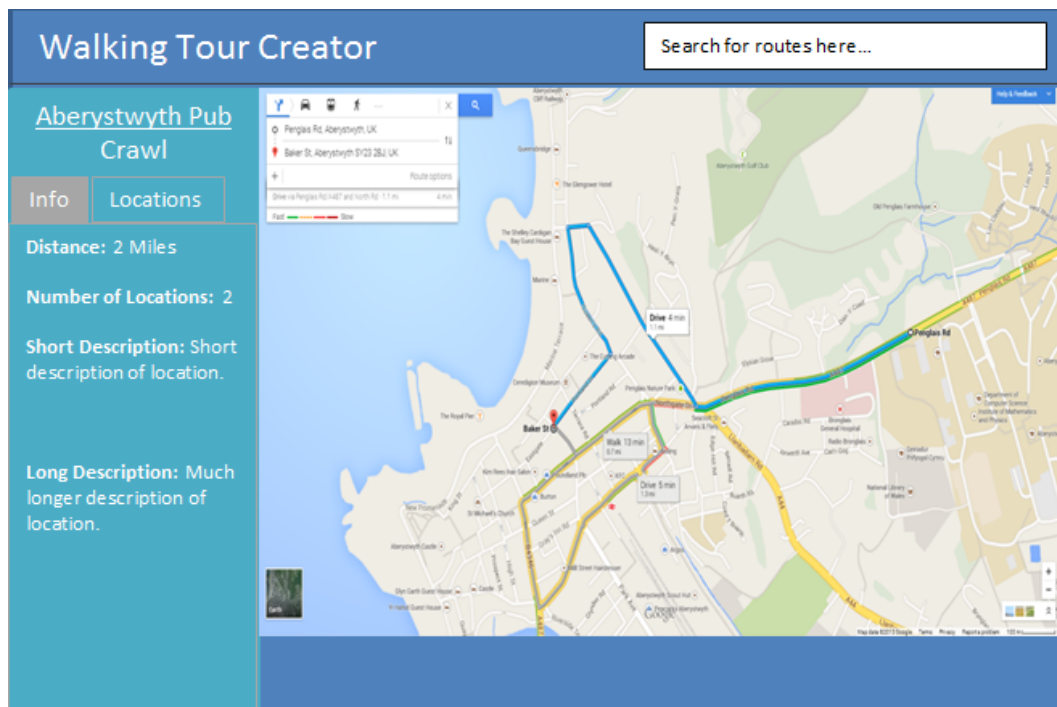
12.2 Web Application Design

12.2.1 HomePage

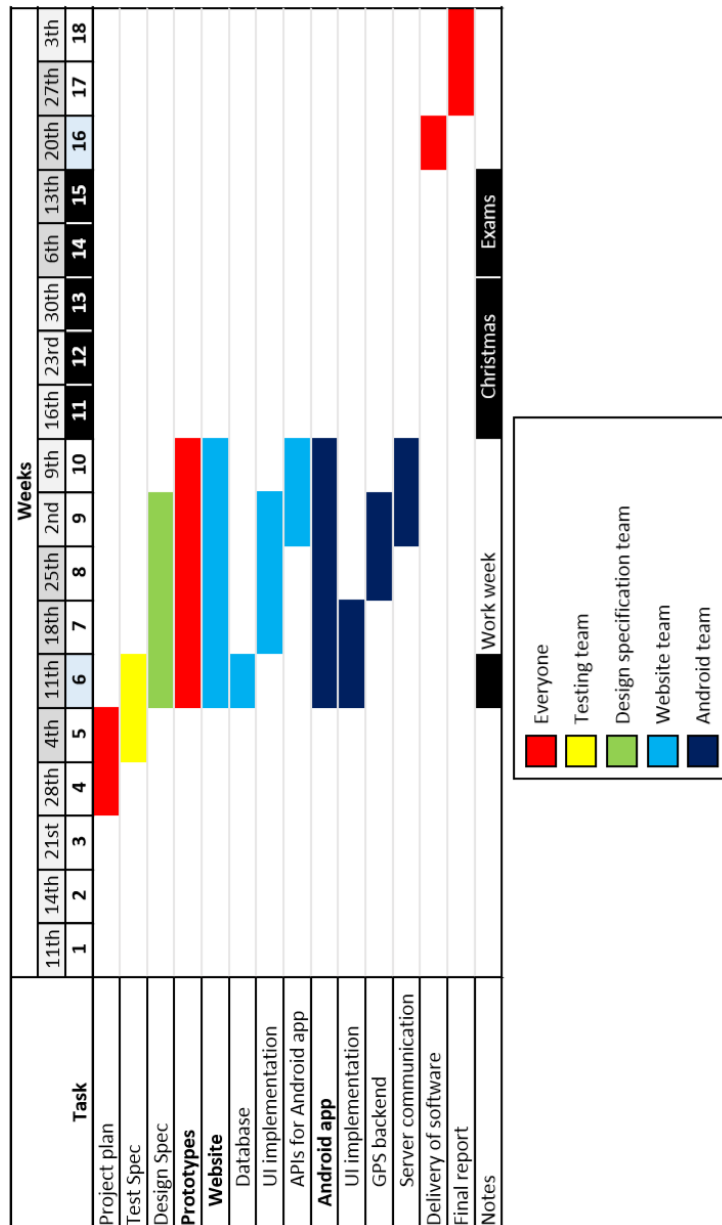


There will be images on the home page of the walks. The user can then select a walk from the homepage. The user can then select this route which will then populate the sidebar and Google Map. The sidebar will allow the user to toggle between information about the walk and information about the locations, as there is not enough space to contain both.





13 GANTT CHART



14 RISK ANALYSIS

14.1 Gold plating

As programmers we sometimes like to show off our skills by adding unnecessary features to our program. The schema clearly states that there are no extra marks for extra features and so doing this would just be a waste of programming hours. This can be avoided by having a clear plan of what is needed for the program before the actual programming starts.

14.2 Illnesses

This is an unavoidable risk. It is inevitable that members of the group will fall ill at some point during the year. To stop this becoming a problem, any members that fall ill will need to let the group know as soon as possible, so that other members can help out with any work that the ill member had been assigned. It is then the responsibility of the ill member of the group to then put in some extra work time when he recovers from illness. Also more than one programmer will work on and understand each part of the code.

14.3 Time management

One of the biggest risks for the group is that we will run out of time and that the project will be unfinished in the due date, or that all members of the group will have to start putting in crazy hours, in a desperate attempt to get the program done. Again this can be avoided by having a clear plan, where each specific part of the website and program have a clear completion deadline.

14.4 Compromising on design

In order to get stuck into the harder tasks of a program earlier, programmers will compromise on design on the easier parts of the program. This is also a waste of programming hours as design is arguably the most important part of a program.

14.5 Backups

The worst situation that could happen is that all the data that the group has been working on gets lost. This is avoided by having multiple backups of our program, and not just on the same hard drive. These backups need to be made in a separate physical location, and also need to be made every time work has been done on the

project. As well as several physical copies, all the work will be constantly updated and stored on github.

14.6 Released project has low quality

There is always the risk that the project, when completed is of low quality. This can be avoided by assuring that everyone is happy with suggested Interface design and everyone is clear on what needs to be done. We will also need to ensure a disciplined development is used.

14.7 Conflicting Ideas

As programmers, we are all likely to have our own ideas on how we think the app should be developed. To avoid any conflict, any issues that an individual may have with the design of the app, will have to bring them forward sooner rather than later. This will overall make for a happy group who will be able to work efficiently as a team.

15 APPENDIX D: REVISED DESIGN SPECIFICATION

15.1 DECOMPOSITION DESCRIPTION

15.2 Programs in system

The system is composed of two programs:

- The Walking Tour Creator Android application
- The Walking Tour Displayer website

15.2.1 Walking Tour Creator

The Walking Tour Creator is an application to allow the user can create walks on a mobile device for them to be uploaded to the server. It implements the requirements (FR1), (FR2), (FR3), (FR4), (FR5), (FR6), (FR7), (FR9), (PR1) and must conform with the requirements (EIR1), (PR2), (DC1), (DC2).[?]

This application will use a GUI as a means for the user to create a walk. The user will be able to walk around an area and the GPS component of their device will capture the route that they take. Along the route, they can mark points of interest, which contains a description and optionally a photo. Once the user has completed the walk, they can choose to either discard their creation or upload it to the Walking Tour Displayer.

15.2.2 Walking Tour Displayer

The Walking Tour Displayer is website for viewing walks created by users via the Walking Tour Creator. It implements the requirements (FR8), (FR9), (PR1) and must conform to the requirements (EIR1), (PR2), (DC1), (DC2), (DC3).[?]

The walks are stored in a MySQL database with the attributes specified in (DC3). The database will be queried by the website and allow the user to select a walk. The selected walk would then be displayed on a map, and the user can select a points of interest along the route of the walk.

15.3 Significant classes in each program

15.3.1 Significant classes in Walking Tour Creator

MainAppActivity. This is the main class of the application.

WalkCreatorActivity. This is an activity panel holding the UI for recording the walk. It also has a Location Listener running in the background recording the users walk

Walk. This contains a list of the coordinates of the route that the user is creating, and the metadata of the walk.

PointOfInterest. This contains a point of interest in the route, which is a coordinate with a description and optionally a photo of what is in that.

15.3.2 Significant functions in Walking Tour Displayer

intialize. This JavaScript function creates an embedded Google Map on the page.

addMarker. This JavaScript function takes coordinates as parameters and sets a marker on the map using them.

addPath. This JavaScript function takes two coordinates as parameters and it sets a path on the map using them.

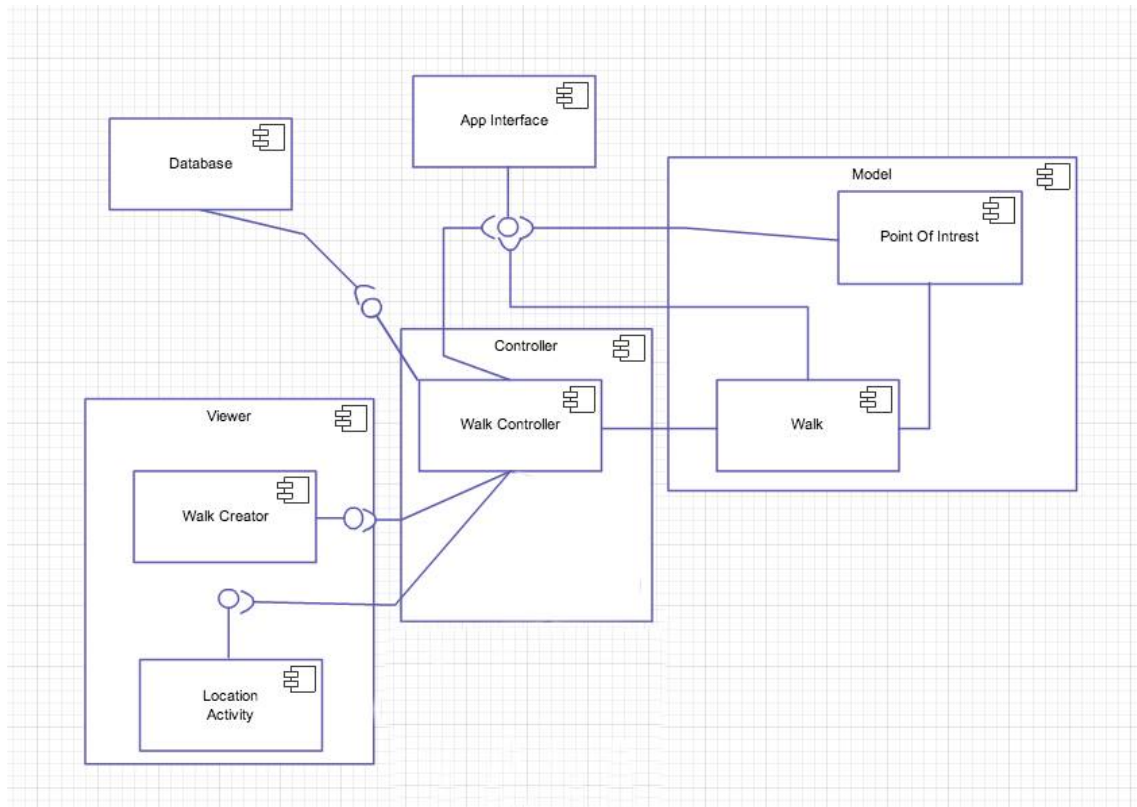
15.4 Mapping requirements onto classes

<i>Requirement</i>	<i>Classes providing requirement</i>
FR1	MainAppActivity
FR2	WalkCreatorActivity, Walk
FR3	WalkCreatorActivity, Walk, PointOfInterest
FR4	WalkCreatorActivity
FR5	WalkCreatorActivity
FR6	WalkCreatorActivity
FR7	MainAppActivity
FR8	N/A
FR9	WalkCreatorActivity

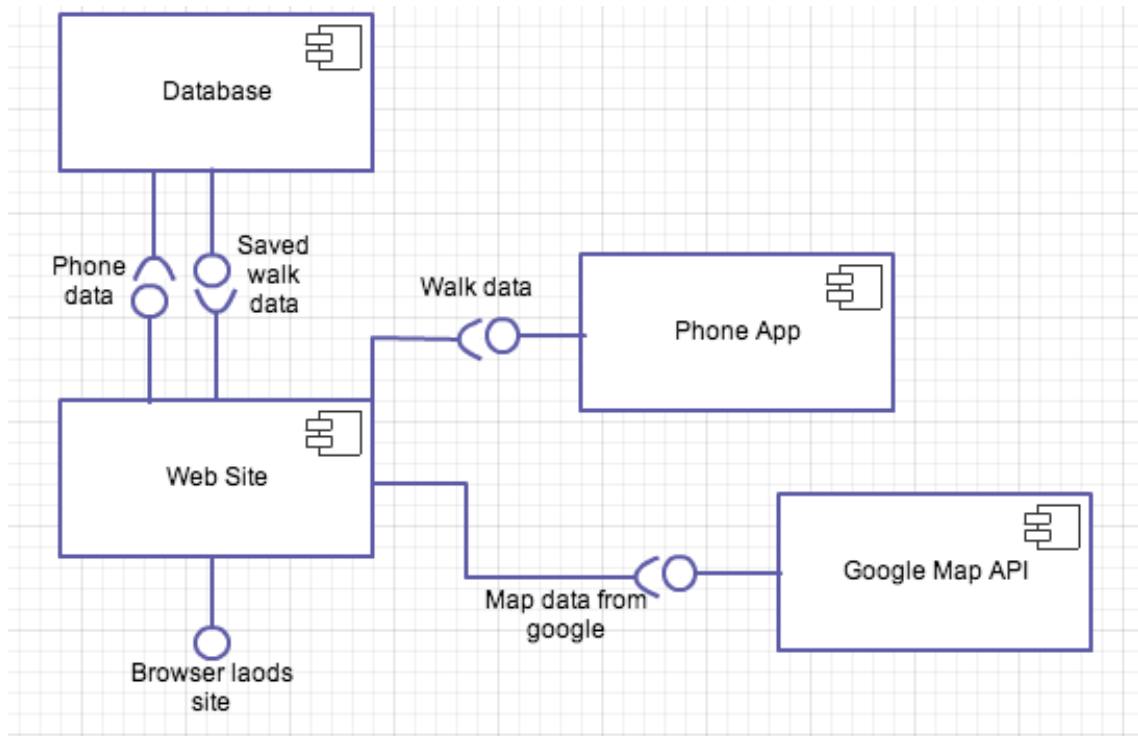
15.4.1 DEPENDENCY DESCRIPTION

15.4.2 Component Diagrams

15.4.3 Component Diagram for Walking Tour Creator



15.4.4 Component Diagram for Walking Tour Displayer



15.4.5 Inheritance Relationships

The class with the most dependency is the WalkController, which has GPS recording, walk creator and location activity. Just about all of the classes need WalkController and this is where the app will send the data from the walk made from the app to the website. Walk has a dependency with PointOfInterest because it calls the gets and sets from that class. WalkCreatorActivity does GPS recording passes to controller then to walk. WalkCreatorActivity has the most dependencies.

For basic understanding of the dependencies of the Walking Tour Creator:

- Walk controller on walk creator and Location activity.
- Walk on point of interest and walk controller.
- Web site on walk controller.

The website has dependencies on the database to be able to get data of the walks and to be able display them on the map and also show the description of the walk on the side panel next to the map. Finally, the phone has a dependency on the website because the data from the walk needs to go through the website to be able to send the data to the database.

For basic understanding of the dependencies of the Walking Tour Creator:

- Website on database and Google Maps API.
- Phone app on website.

15.5 INTERFACE DESCRIPTION

15.5.1 MainActivity interface specification

```
/**
 * This activity is shown when the user opens the app.
 * It has a single button which allows the user to "Create a walk".
 * This button launches the "WalkDetailsActivity" Activity.
 *
 * @author Group 14
 */
public class WalkDetailsActivity {
    /**
     * This is the method which launches the "WalkCreatorActivity" Activity.
     * This is called when the user presses the "Create route" button.
     */
    public void createWalk();
}
```

15.5.2 WalkDetailsActivity interface specification

```
/**
/*
 * @(#) IPointOfInterest.java 1.0 2014-01-31
 *
 *
 */

/**
 * This interface is used represent a point of interest.
 * This will store a String name, String description,
 * Location object, and optional String representing
 * a picture associated with the point
 * @author Group14
 *
 */
public interface IPointOfInterest extends Parcelable {

    /**
     * This method is used to set the IPointOfInterest's name
     * @param name String representing the new name of the IPointOfInterest
     */
}
```

```
public void setName(String name);

/**
 * This method is used to get the IPointOfInterest's name
 * @return String representing the name of the IPointOfInterest
 */
public String getName();

/**
 * This method is used to set the IPointOfInterest's description
 * @param desc String representing the new description of the IPointOfInterest
 */
void setDescription(String desc);

/**
 * This method is used to get the IPointOfInterest's description
 * @return String representing the IPointOfInterest's description
 */
public String getDescription();

/**
 * This method is used to set the picture of the IPointOfInterest.
 * This must be a valid path to a picture in the filesystem.
 * @param picture String representing the location of the picture in the filesystem
 */
public void addPicture(String picture);

/**
 * This method is used to get the picture of the IPointOfInterest
 * @return A string representing the location on the filesystem of the picture
 */
public String getPicture();

/**
 * This method is used to get the IPointOfInterest's latitude
 * @return A double representing the IPointOfInterest's latitude
 */
public double getLatitude();

/**
 * This method is used to get the IPointOfInterest's longitude
 * @return A double representing the IPointOfInterest's longitude
 */
public double getLongitude();
```



```
/**
 * This method is used to get the IPointOfInterest's timestamp.
 * @return long timestamp in the unix epoch format.
 */
public long getTime();

/**
 * This method is used to get the IPointOfInterest's Location object
 * @return A Location object, representing the GPS location of the IPointOfInterest
 */
public android.location.Location getLocation();

}
}
/*
 * @(#) IJsonPackager.java 1.0 2014-01-31
 *
 * Copyright (c) 2014 Aberystwyth University.
 * All rights reserved.
 *
 */
package uk.ac.aber.group14.model;

/**
 * This interface is used to convert an IWalk to a JSON String
 * @author Group14
 *
 */
public interface IJsonPackager {

    /**
     * This takes an IWalk and converts it into a JSON String
     * @param w The walk which will be converted
     * @return A String containing the walk represented as a JSON object
     */
    public String JSONify(IWalk w);

}
```

15.5.3 Walk interface specification

```
/**
 *
 * @(#) IWalk.java 1.0 2014-01-31
 *
 */
package uk.ac.aber.group14.model;

/**
 * This interface is used to represent a walk
 * This stores a collection of Location objects,
 * a collection of IPointOfInterest objects,
 * a String name, String short description, and String
 * long description.
 * @author Group14
 *
 */
public interface IWalk extends Parcelable{

    /**
     * This method is used to add a single IPointOfInterest to the IWalk
     * @param point The point to add to the IWalk
     */
    public void addPointOfInterest(IPointOfInterest point);

    /**
     * This method is used to add a LinkedList of Location objects to the IWalk
     * @param locations The Locations to add to the IWalk
     */
    public void addLocations(java.util.LinkedList<android.location.Location> locations);

    /**
     * This method is used to set the name of the IWalk
     * @param name String representing the name of the walk
     */
    public void setName(String name);

    /**
     * This method is used to set the short description of the IWalk
     * @param desc String representing the short description of the IWalk
     */
}
```

```
public void setShortDescription(String desc);

/**
 * This method is used to set the long description of the IWalk
 * @param desc String representing the long description of the IWalk
 */
public void setLongDescription(String desc);

/**
 * This method is used to return the points of interest
 * as an array of PointOfInterest
 * @return An array of type PointOfInterest representing all the points of interest i
 */
public IPointOfInterest[] getPointsOfInterest();

/**
 * This method is used to return the GPS locations as an array of
 * type Location
 * @return An array of type Location representing all of the Locations in the IWalk
 */
public android.location.Location[] getLocations();

/**
 * This method is used to get the name of the IWalk
 * @return A String representing the name of the IWalk
 */
public String getName();

/**
 * This method is used to get the short description of the IWalk
 * @return A String representing the short description of the IWalk
 */
public String getShortDescription();

/**
 * This method is used to get the long description of the IWalk
 * @return A String representing the long description of the IWalk
 */
public String getLongDescription();

/**
 * This method is used to add a single Location to the IWalk
 * @param location The Location to add to the walk
 */
```

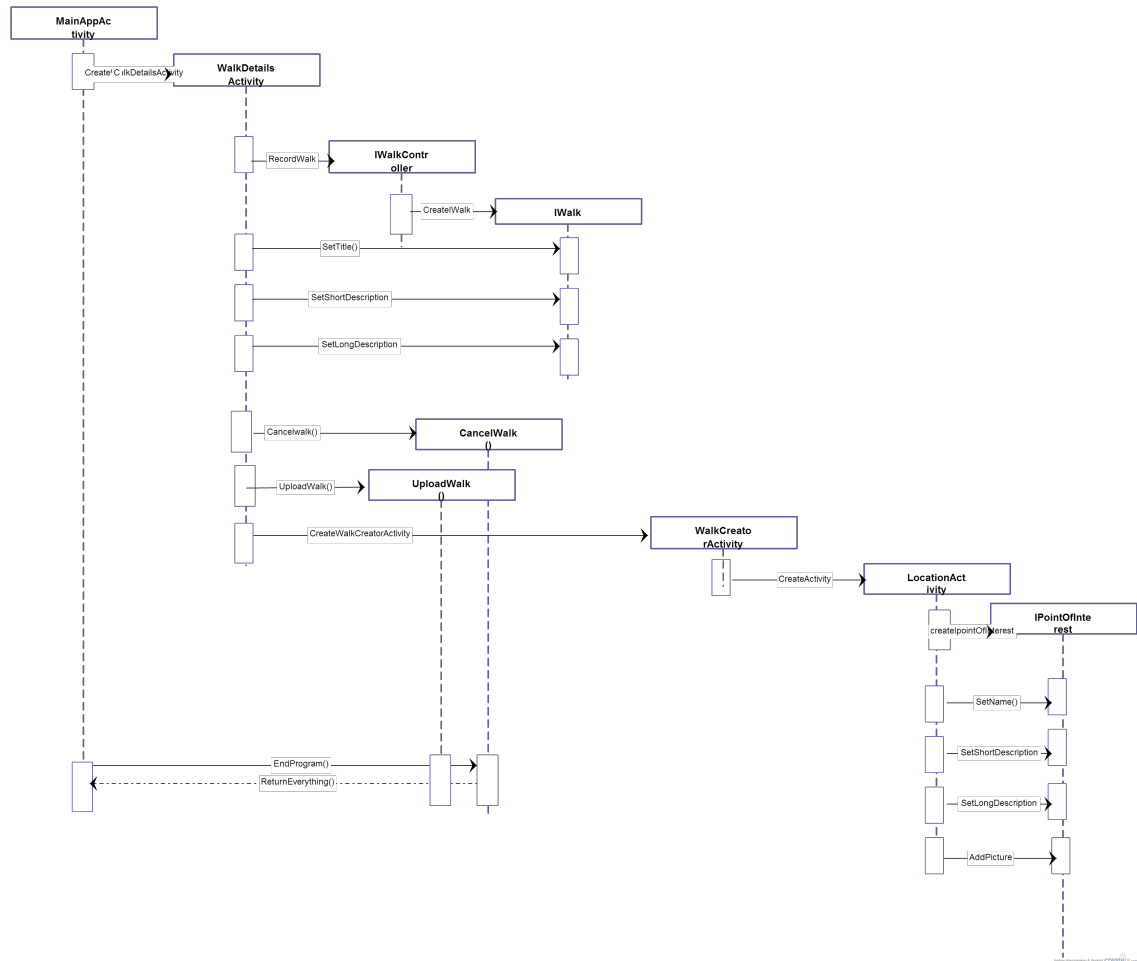
```
public void addLocation(Location location);

/**
 * This method is used to get the number of Location objects
 * in the IWalk
 * @return An int representing the number of Location objects in the IWalk
 */
public int getNumberLocations();

/**
 * This method is used to get the number of IPointOfInterest objects
 * in the IWalk
 * @return An int representing the number of IPointOfInterest objects in the IWalk
 */
public int getNumberPOI();
}
```

15.6 DETAILED DESIGN

15.6.1 Sequence diagrams



15.6.2 Significant algorithms

SetMarkers SetMarkers basically takes points from the database and plots them using the LineBetweenPoints function and the google API. It takes in the latitude and longitude from a 2 dimensional array and plots the points on the map

LineBetweenPoints LineBetweenPoints uses the google API to go from of marker to the next and plots a route in between them. Before we discovered the API, we thought to write our own version in java. see the JSON document for the data transmission algorithms.

15.6.3 Significant data structures

The data structures we will be using are the IPointOfInterest interface and the IWalk interface.

The IWalk interface will be implemented by a class and will hold all of the points of interest. It is the current plan to use a LinkedList due to it being dynamic in size. If the team member implementing the interface has a compelling enough reason to use an alternative means of storing them then they can do. Due to it being an interface this will be up to whoever implements it - the public methods will remain the same.

The IPointOfInterest will be used to store the data about a point of interest. For storing the pictures it is again likely that a LinkedList will be used.

The Location objects will be stored in a LinkedList in the walk whilst they are being recorded. This will allow us to quickly and easily add new Location objects to the end of it. Unlike an array, we will not need to repeatedly recreate the list with a larger amount of memory each time a new location is added.

REFERENCES

- [1] *Software Engineering Group Projects* General Documentation Standards. C. J. Price, N. W. Hardy, SE.QA.03. 1.5 Release.
- [2] *Software Engineering Group Projects* Design Specification. Group 14, SE.14.DESIGN. 1.2 Release.

DOCUMENT HISTORY

Version	CCF No.	Date	Changes made to Document	Changed by
0.1	N/A	2014-02-09	Initial creation with performance reports	jmt14
0.2	N/A	2014-02-14	Management summary draft	tht5
0.3	N/A	2014-02-14	Critical evaluation	jam64
0.4	N/A	2014-02-14	Final state of the project	rab26
0.5	N/A	2014-02-14	Historical account of the project	jam66
0.6	N/A	2014-02-14	Maintenance manual initial draft	rab26, jab73
0.7	N/A	2014-02-14	Minor changes to language to meet QA	jmt14