CS211 Group 15
Final Report

Authors: lpd1; jcm14; jap38; jaj42; owd2; jam67; gad16
Date: 17-02-14
Version: 1.0
Config Ref: SE_15_FinalReport

Department Of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

Contents

1 INTRODUCTION

1.1 Purpose of this document

This document is our final report for the CS22120 Group Project, and is a report on our progress through the project and a compilation of all of our final documents.

1.2 Scope

This document will include a management summary, a historical account of the project, a description of the final state of the project and reviews of each team member. Also, its' appendices will contain the project test report, the maintenance manual and the revised project plan and design.

1.3 Objectives

- Provide a summary of what the project achieved and what difficulties stood in the way.
- Give an outline of the main events over the course of the project.
- Describe the final state of the project and any known problems.
- Give reviews of the performance of each team member.
- Include the project test report, maintenance manual, plan and design specification.

2 Management Summary

Our project successfully managed to create an application that fulfilled the specifications provided to the group, despite having problems with the formatting of the first group of design documents, a number of management issues and subsequent changes, as well as suffering from having a significantly smaller group than average. The fact that our team was 2 members fewer than the competing groups meant that all of us had to complete extra work outside of the brief of our individually assigned position (e.g. all of us had to complete a fair amount of code, everybody also had to complete parts of the final hand-in and testing documents). We also had to make some adjustments to our final application; due the increased time it took our group to complete the basic requirements outlined in the specification we were unable to complete some of the more optional features, such as the function to browse and edit previously saved tours from within the mobile application, and the ability for the user to view and delete photos within a local gallery that have been taken through the application.

Due to the fact that we knew that we would not be able to complete every requirement at the start of Integration and Testing Week we made a group decision, which was partly forced into action after having to drastically change our plans due to the loss of one of our current acting leader at the time Josh Moss, to focus on the more core aspects of the application and to ensure that it was of above optimal quality to attempt to make up for our missing functionality. By the end of Integration and Testing Week we had managed to create a very stable and fully working mobile application, as well as a complete website capable of satisfying all of the Core Requirements.

During the digital hand-in of the first group of Design and Analysis documents we were confronted with a number of formatting issues due to a lack of communication within the group, as well as a lack of standardisation in the programs that we individually used to produce the documents. These issues meant that the invigilators were unable to read the documents, specifically some of the diagrams supplied for the initial design of the User Interface, as well as a Gantt chart that was completely unusable.
The roles of each group member were:

- Lewis Dale: *Group Leader, Lead Web Developer*
- Jack Pendlebury: *UI Design and Implementation*
- Jafnee Jesmee: *Version Control Expert*
- Owen Day: *Testing Manager*
- Gavin Davies: *Lead Android Developer*
- Jamie McLoughlin: *UI Design and Implementation/Android Developer*

- Joshua Moss: *Android Devloper*

As a group, after we finally managed to sort out the roles each person was assigned and everyone was sure they were actually part of the group and participating in the project, we had good chemistry, maintained good communication and all contributed both during meeting and when it came to physical or digital hand-ins. Lewis was a strong leader, and gave us just the right amount of guidance in order to make sure that everything was completed to an acceptable standard, all while maintaining a positive working atmosphere.

3 Historical Account

This section covers the procedures the group has undertaken to go from scratch to the final product. Group 15 has fewer members than the average group attempting the same project, therefore each member will have to do more work than was initially thought. There are seven main sections which breakdown the project; these are:

The client did not turn up to the first group meeting nevertheless we quickly got down to work. The initial group consisted of 8 people with varying skillsets, each more suited to a specific job. Everyones abilities were analysed and were assigned roles accordingly, key roles include the team leader, lead android developer, QA manager, web developer. The group leader position was assigned to Josh Moss and deputy leader was assigned to Lewis Dale.

In the second group meeting we had found out that we would be one member short as he would be retaking the first year. This meant that we as a group were one member down the average size and members may be delegated more tasks that initially thought out. We established when the group would hold weekly group meetings aside from the one we already had with Nigel and had a room booked. Everyone checked their timetables to find a suitable time where everyone would be free too meet up.

The first document to be created was the project plan. The QA document for the project plan was used to allow the group to plan out tasks that had to be completed. The main sections required for the document included a system overview, project time scale, user interface design, GUI design and risk analysis. During a group meeting the tasks were handed out to group members and a deadline was set within the group, this was to allow sufficient time for all members to submit their section of the document to the QA manager for merging and for a review meeting to take place; giving us time to improve and correct mistakes in the document. This document was handed in late due to problems with communication between group members and a lack of knowledge of the procedures to submit the document.

The second document to be produced was the test specification. The documents sections included system tests for the android application and the web application. The documents main tasks were given to the testing manager to work on the android side of testing and to the group member assigned to the web application to work on testing for the website. This document was reviewed by the group during a web meeting and was submitted to the client on time.

The third document to be worked on was the design specification. This documents main sections included the significant classes in both programs, component diagram for both programs, interface description and a detailed design which contains the significant algorithms and data structures. During the group meeting we found out that we would be down another member, our QA manager. The role of QA manager had been reassigned and the group leader position has been passed to Lewis Dale. A review meeting was held with Nigel to help provide insight over how our document could be improved. The submission to the client was done on time.

Implementing the prototype was the next step of the process. The two applications to be worked on was the android application and the web application, these two were developed separately by different members. Lewis Dale worked on creating the website prototype and the rest of the group were given tasks to implement the Android application. The more experienced members who have worked on android before were given tasks deemed more difficult and more crucial for the application to work and other members were given the task of completing the GUI and the model classes. The prototype was then given to the client to be reviewed.

Significant work on the program was not done until after our end of semester examinations. The main work began during integration and testing week where group members were expected to work full-time on the project. The prototype was scraped and the group had decided to rewrite the code as the code found in the prototype was full of errors and was uncommented making it hard for other members to continue the work of Josh Moss who would not be present during coding week. Everyone was delegated work to do as we were down three members and were under a very heavy time constraint. We had problems trying to get everyone to work on the same project file as members were using different environments to work on the code, this caused us to work on sections of code and have the work sent to Lewis Dale who merged all of the code together in order to get a system that would run. When the system was in a state that could be run, the Android application was loaded up onto the members Android devices in order to test the application in the real world. Once the system had been tested Javadoc was generated and all of the source code was uploaded to the control section of the Git repository.

The group then arranged a meeting with the client in order to have the application acceptance tested. This was done by the client going through the android program and checking its functionality against a checklist that the program should be able to accomplish.

4 Final State

4.1 What We Completed

The software fulfilled most of the functional requirements bar a handful of exceptions. The initial screen displays a choice of creating a new walking tour; when clicked, the application accepts details of the walk, such as the short and long description. This is in keeping with requirements FR1 and FR2.

The user is able to add locations (defined in the design as Waypoints) to the walk. GPS coordinates are recorded and the user is prompted to add information about the location, which fulfils requirement FR3.

Requirement FR5 is fulfilled as the user is able to cancel a walk from the default walk creation screen.

The transfer of walk data to the server was sucessful, sending the data as a string encoded into JSON.

The web application displays walk routes on a Google Maps screen, clearly showing the route as a line on the map between the coordinates recorded by the application, which fulfils requirement FR8.

Requirement FR9 is fulfilled as the Android client receives a confirmation message from the web application upon successful transfer of the data. Otherwise, the process times out and the application attempts to re-establish the connection.

4.2 Known Problems

One problem that we were aware of was that the application would not send images to the server, only waypoint and walk information. This was due to time and manpower constraints.

Also, we are aware that occasionally the application will allow the user to create a new waypoint despite not having collected GPS results. If the user can not connect to GPS, the waypoint cannot be created. However, if the user has GPS but does not leave enough time for the application to calculate their location, they can submit a tour with coordinates at 0.0, which can lead to unexpected results when viewing.

Another problem with the project was that we did not use proper config references, meaning that many of them are missing appropriate references.

5 Team Member Reviews

In this section are the reviews of each group member, made by the group leader and agreed upon by the respective member.

5.1 jap38

Role: UI Designer

Jacks' performance throughout the group project was generally good. Whenever he was delegated a task he would complete it, usually to the deadlines that he was set. One one occasion the work was handed in late -but once he was prompted, Jack sent me the work, showing it was completed to a fairly high standard. During Integration and Testing Week, he was always prepared to do whatever task was handed to him - often going above what was needed. For example, when asked to create the user interface for the application, he not only created it, exactly as it was required, but also included extra sections of code to handle button presses and other events.

Time Spent On Project: 57 Hours
Agreement Reached

5.2 jaj42

Role: Version Control Expert

Jafnee took his role as the Version Control Expert seriously, and seemed to have spent quite some time becoming aquainted and capable at using Git. This proved to be extremely helpful during Integration and Testing Week, as we experienced some quite serious git errors, which he was then able to fix for us, and ensure that we could all work with the repository well. Aside from his role as Version Control Expert, Jafnee was also a capable worker, often providing excellent input into sections of our documentation, always on time. He was very reliable; I knew that if I asked Jafnee to do something, that it would get done, and generally to a good standard. On the occasions where he was late to meetings, he would generally have a valid excuse and would make the time up.

Time Spend On Project: 59 Hours
Agreement Reached

5.3 owd42

Role: Testing Manager

Owens' role as the Testing Manager involved him creating the test specification and performing the system tests, as well as having some input on the documentation. Whenever he was allocated a task, Owen completed it quickly, and to a good standard - there were rarely things to fault in his work, and when there was a fault it was minor and fixed within a short time. Particularly towards the end of the project, Owen proved himself to be extremely dedicated, and would appear to spend a lot of time and put a lot of effort into completing his tasks.

Time Spent On Project: 60.5 Hours
Agreement Reached

5.4 jam67

Role: UI Designer / Android Development

Jamie's role originally consisted of just performing Android user interface designs. However, with the withdrawal of several of our members (1/3 of the group), he found his role significantly expanded to mainstream Android development. He took these extra roles in his stride and certainly worked very hard to ensure that he performed them to the best of his ability. There were some areas that he struggled with when it came to Android, but he continued to work and work at it, until he was almost the whole way there, by which point i was able to step in and help him to solve the problems he was having.
Overall, however, I found him to be hardworking and willing to follow instructions in a timely and well-executed manner. There were no instances that work was handed in after a deadline, and when adjustments to the work that was handed in were necessary, they were completed and handed in quickly.

Time Spent On ProjecT: 74.5 Hours
Agreement Reached

5.5 gad16

Role: Lead Android Developer

Over the course of the project, Gavin proved himself to be one of our more able group members. In particular, he was quite a strong programmer, and I was able to trust him with large portions of the code over integration and testing week and be safe in the knowledge that the work that would be returned to me would be of a high quality. He was also fairly reliable in regards to deadlines and showing up where expected - work was always handed in on time and he would always turn up to meetings. Gavin was, however, quite quiet in meetings and would rarely speak unless prompted. But, overall, Gavin was a hardworking and reliable member of the team.

Time Spent On Project: 46 Hours
Agreement Reached

5.6 jcm14

Role: Android Developer

Initially, Josh appeared to be hardworking and reliable - he was originally the group leader and seemed well-suited for that role. However, once the first assignment hand-in came around things proved to be a little different. He assigned himself the task of writing most of the Project Plan, as well as compiling all of our work and handing it in, yet failed to complete these to a reasonable standard. The work he had done was incomplete and contained a lot of inconsistencies, and the document was badly formatted - to the degree where images were completely separate from their respective text. On top of all of that, he then failed to hand the document in on time, instead handing it in approximately 24 hours after the deadline. After this, he conceded that he did not feel the role as group leader suited him, as he had other commitments outside of university that took up a lot of his time, and instead passed the role to the deputy leader (Lewis Dale). After that, his work picked up considerably, and what he was producing was over a good quality. Approximately three weeks before Integration and Testing Week, however, he informed me that he would be unable to attend as he had somewhere else to be (something which I assumed had been cleared with the department). I agreed to him leaving for the week on the condition that he produced the equivalent to his side of work in Android development, meaning we would not feel the effects of being yet another team member fewer than before. He did not do this, the work that he left us was simply the user interface, when it was agreed he would also add some work to the model. This, combined with the fact that he had used an unsupported version of gradle (we were using $v1.7$ while he was using a much later version) meant that we had to spend a lot of time just getting the application that he had left us to run. After Integration and Testing Week, he then did not attend the final group meetings and did not partake in any work regarding the final document and maintenance manual.

Time Spent On Project: Timesheet not submitted
Agreement Not Reached: Josh did not attend the review meeting.

5.7 lpd1

Role: Team Leader/Web Developer

Written by Jafnee Jesmee Lewis was originally the lead programmer for the web application. Due to circumstances where the group was unable to commit to deadlines on time, Lewis stepped up to become the groups new leader and would maintain his position as the lead programmer for the web application. The change in the group was very apparent after his appointment as the new group leader as no submission deadlines were missed and quality of work from the group has increased. The key to Lewis being a good leader was his strong character and his persistent communication with group members to insure work was completed on time and was approachable enough that members would not hesitate to ask for help when required. His second role as the lead web application developer has also shone through as he was capable of designing and programming a fully functional web application by himself. Lewis is a very straightforward leader that would give constructive feedback and was capable of taking criticism from the group, making self improvements as a leader. The state that the project has reached would have to be credited to Lewis stepping up to the position of leader, driving the team forward; regardless of our position being one of the smallest groups to work on the project.

Time Spent On Project: 88 Hours
Agreement Reached

6 Appendices

6.1 Project Test Report

2.2.1 Project Test Report

Project Test Specification
Walking Tour Application System Tests

Test Ref	**Req being tested**	**Test Content**	**Input**	**Output**	**Pass Criteria**
ST1.1	FR1	Checks that software is able to open correctly and doesn’t crash	User selects the application from their phone	Software opens and shows main screen	Software opens correctly and shows the main screen
ST1.2	FR1	Checks user is able to create a new walk	User clicks ‘Start Walk’	New page should open prompting user to enter details about walk	Software creates a new walk and prompts user to enter details
ST1.3	FR1	Checks user is able to start walk when all information is entered	User enters valid information about the walk and selects start walk	Information is submitted and walk starts	Walk starts successfully
ST1.4	FR2	Checks for error message if name of walk is not added	User selects start walk without entering name of walk	An error message will appear telling user they have to add a name for the walk	Error message displays correctly
ST1.5	FR2	Checks for error message if short description is not added	User selects start walk without entering short description	An error message will appear telling user to add short description of walk	Error message displays correctly
ST1.6	FR2	Checks for error message if long description is not added	User selects start walk without entering long description	An error message will appear telling user to add a long description for the walk	Error message displays correctly
ST1.7	FR2	Checks to see if text stops getting typed when it reaches 99 characters for short description	String of text above 99 characters	Text appears as it is being typed until character limit is reached	Text stops being typed when character limit is reached

ST1.8	FR2	Checks to see if text stops getting typed when it reaches 1000 characters for long description	String of text above 1000 characters	Text appears as it is being typed until character limit is reached	Text stops being typed when character limit is reached
ST1.9	FR3	Checks user is able to add new waypoint	User selects 'Waypoint'	User is taken to Waypoint screen where user can enter details about walk	Waypoint screen opens
ST1.10	FR3	Checks user is able to add waypoint when all information is entered	User enters valid information and selects add waypoint	Message appears telling user that waypoint has been added	Message appears
ST1.11	FR3	Checks if error message appears if user doesn't enter description of location	User selects 'add waypoint' without entering description of location	An error message appears telling user to enter a description of the walk	Error message displays correctly
ST1.12	FR4	Checks user is able to choose how to add a photo at a waypoint	User selects 'add photo'	User taken to new page where they can choose to capture new image or load image from gallery	New page opens
ST1.13	FR4	Checks camera opens when capture new image is selected	User selects 'capture new image'	Camera opens and functions correctly	Camera opens from within the app
ST1.14	FR4	Checks user is able to capture a picture from within the app	Users captures new image	Camera open and users is able to capture an image	Camera opens correctly and image is captured correctly
ST1.15	FR4	Checks if user is able to open gallery	User selects 'load image from gallery'	Gallery opens and user is able to select picture from gallery	Gallery opens correctly
ST1.16	FR4	Checks if user is able to select photo from library	User selects image from gallery	User selects photo from library and image is added to database	Image can be selected from library and added to database

ST1.17	FR5	Checks user is able to cancel a walk	User selects ‘Cancel walk’	Warning message appears ensuring user wants to cancel walk	Warning message appears
ST1.18	FR5	Checks if warning message appears when user decides to cancel walk	User agrees to cancel walk	Walk stops recording and all data recorded is deleted	Walk is cancelled and all data deleted
ST1.19	FR5	Checks if warning message appears but user decides not to cancel walk	User decides not to cancel walk	Walk continues recording and all data remains the same	Walk continues recording without disruption
ST1.20	FR7	Check that current walk information gets saved when switching app	After partially creating a new walk, use the Android task manager to change to another application. Then close that application and re-open the WTC.	When re-opened, the partially-completed walk should open.	The walk opens with all of the correct information. If the walk does not load or it has not saved all the information properly.
ST1.21	FR6/FR9	Checks user is able to successfully submit a walk.	User presses the “save walk” button on the app interface.	User should receive a message from the server with either a confirmation of success or failure.	The user receives a success message from the server.
ST1.22		Checks for error message if phone is not connected to network	User opens walking tour application	Error message should appear stating that the phone is not connected to a network	Error message appears
ST1.23		Checks for error message if phone loses network coverage during walk	User starts new walk	Error message should appear stating that the phone has lost connection to the network	Error message appears

Walking Tour Web Interface System Tests

Test Ref	**Req Being Tested**	**Test Content**	**Input**	**Output**	**Pass Criteria**
ST2.1	FR8	Checks that the user is able to select and view a walk.	Select walk in list	The relevant walk information is displayed, showing the walk name, description and all of the walk waypoints are displayed on the map.	The walk that was clicked on is displayed, with all of the correct information. The test team should have submitted the walk in test ST1.22 to ensure that it was performed correctly.
ST2.2	FR8	Checks that a user can select a waypoint and view the information about that waypoint.	Select waypoint on map.	Information about that waypoint should be loaded onto the screen and made visible.	The correct information about the waypoint should be displayed. If no information is displayed, or it is the wrong information, then it is a failed test.
ST2.3	FR8	Check that waypoints are at the correct map coordinates.	Select a walk and view the map of waypoints.	Each waypoint should be pinned in the correct location on the map.	If the waypoints are at the wrong coordinates compared to where they were submitted from, then the test fails.
ST2.4	FR8	Check that images are displayed correctly.	Select a walk and then a waypoint, and view the images of that waypoint.	The images uploaded with the waypoint should be displayed correctly.	If the correct images are displayed, and display as they are defined in the design specification, then the test passes.

Test Log 1
Date: 30/01/2014
Tester(s): owd2; jam67; jap38

Test ID	**Pass/ Fail**	**Fail Description**
ST1.1	Pass	
ST1.2	Pass	
ST1.3	Pass	
ST1.4	Fail	No error message appears telling user to add a name for walk Walk started
ST1.5	Fail	No error message appears telling user to add a short description for walk Walk started
ST1.6	Fail	No error message appears telling user to add long description for walk Walk started
ST1.7	Fail	No limit to amount of text allowed to be entered
ST1.8	Fail	See explanation for test ST1.7
ST1.9	Pass	
ST1.10	Pass	
ST1.11	Fail	No error message appears telling user to enter description for waypoint Waypoint added
ST1.12	Fail	User is unable to choose option of how to upload photo
ST1.13	Pass	
ST1.14	Pass	
ST1.15	Fail	User isn’t given option to open gallery
ST1.16	Fail	See explanation for test ST1.15
ST1.17	Fail	User isn’t given option to cancel walk

ST1.18	Fail	See explanation for test ST1.17
ST1.19	Fail	See explanation for test ST1.17
ST1.20	Pass	
ST1.21	Pass	
ST1.22	Fail	No error message appears telling user phone is not connected to network
ST1.23	Fail	No error message appears telling user phone has lost connection to network
ST2.1	Pass	
ST2.2	Pass	
ST2.3	Pass	
ST2.4	Fail	Images are unable to be viewed on the website

Test Log 2
Date: 31/01/2014
Tester(s): owd2; jam67; jap38

Test ID	Pass/ Fail	Fail Description
ST1.1	Pass	
ST1.2	Pass	
ST1.3	Pass	
ST1.4	Pass	
ST1.5	Pass	
ST1.6	Pass	
ST1.7	Pass	
ST1.8	Pass	
ST1.9	Pass	
ST1.10	Pass	
ST1.11	Pass	
ST1.12	Fail	User is unable to choose option of how to upload photo
ST1.13	Pass	
ST1.14	Pass	
ST1.15	Fail	User isn’t given option to open gallery
ST1.16	Fail	See explanation for test ST1.15
ST1.17	Fail	User isn’t given option to cancel walk

ST1.18	Fail	See explanation for test ST1.17
ST1.19	Fail	See explanation for test ST1.17
ST1.20	Pass	
ST1.21	Pass	
ST1.22	Pass	
ST1.23	Pass	
ST2.1	Pass	
ST2.2	Pass	
ST2.3	Pass	
ST2.4	Fail	Images are unable to be viewed on the website

6.2 Maintenance Manual

CS211 Group 15
Maintenance Manual

Authors: lpd1; jcm14; jap38; jaj42; owd2; jam67; gad16
Date: 11-02-2014
Version: 0.1
Config Ref: SE_15_MaintenanceManual

Department Of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

Contents

1 INTRODUCTION

1.1 Purpose of this document

The purpose of this document is to provide maintenance guidelines and help for use when modifying the Pathfinder application developed by Group 15.

1.2 Scope

This document will include a short description of the program, some information about the structure of the program, pseudocode description of the major algorithms used, a description of the main data structures, information regarding interfaces needed to use the program. Also, it will have suggestions for potential improvements, some pointers of things to be careful of when making changes, physical limitations of the program, and information for rebuilding and testing the program.

1.3 Objectives

- Give a brief description of what the program does
- Describe the design of the program
- Describe the significant algorithms
- Specify the data structures where important information is stored
- Give information about any rules that must be observed to allow the program to run
- Describe possible improvements that were left out or incomplete.
- List any changes that may cause a knock-on effect to other parts of the program.
- Describe the physical limitations of the program.
- Describe how to test and rebuild the program.

2 Program Description

Pathfinder is an android and web based application created as a walking tour creator and displayer. Pathfinder uses GPS from the users phone to track the location of the user to create a tour. The application allows the user to create waypoints along the tour that can contain a description and a photo. These tours can then be saved to the server and displayed on the web application using Google maps to display the tour.

3 Program Structure

For information on the structure of the program, see the design specification, section 3 [1]

4 Algorithms

See the design specification, section 5.2 [1].

5 Main Data Areas

Information regarding data structures can be found in the design specification, section 5.3[1].

6 Interfaces

In this section I will describe the rules that must be observed for the application to run successfully.

6.1 GPS Access

One of the key features of the Android application is it's ability to collect data about the users' current location, and the primary way it achieves this is by using the mobile devices GPS capabilities to find the users' current coordinates. To allow the application to access the GPS, it must have the necessary permissions, which are requested with:

```
<manifest>
  <uses-permission android:name="android.permission.
    ACCESS_FINE_LOCATION" />
   ...
</manifest>
```

[2]
The coordinates can be retrieved using the *getLatitude()* and *getLongitude()* methods in the GPSLocation class [1]. These methods should be called periodically, so that the application always has the users' most recent location.

6.2 Internet Access

To allow the application to send information to the server, it is necessary that is is capable of connecting to the internet. To be able to acquire Internet access, the application must have the relevant permissions, requested using:

```
<manifest>
  <uses-permission android:name="android.permission.INTERNET"
    />
  <uses-permission android:name="android.permission.
    ACCESS_NETWORK_STATE" />
   ...
</manifest>
```

[3]
Then, to ensure that the device is connected to the internet, use the method *public boolean isOnline()* in *MainActivity.java* to check that the user does have network access whenever it is necessary for the device to access the network. If this returns false, then the user is not connected and an error message should be shown.

6.3 Database Access

The web application also must be able to access a MySQL database that matches the database schema defined in the requirements specification[4]. A SQL file can be found in the repository that will build the database upon import. As well as being of the correct format, the file *config.lib.php* in the Lib folder of the web application must also be updated with the new database information, in the following format:

```
//The database host server - often localhost
$host  = "localhost"
//The username used to access the database
$user = "lewisdal_groproj"
//The password used to access the database
$pass = "gr0uppr0j3ct"
//The name of the database
$dbname = "lewisdal_groupproject"
```

7 Suggestions For Improvements

This section describes the potential improvements that could be made to the application if more time had been available or in future updates, and will also talk about the issues of time and man power and how they affected the final program.

Firstly I will discuss the potential improvements that could be implemented further down the line, in an update to the application. One improvement would be the creation and implementation of sign up and login system, which would allow users to create a user ID that would be used to login and be traceable through the web application as well. This could lead to a number of improvements such as searching for a specific users tours, for example a museum creating a tour of the local landmarks and putting it up for other users to use or a walking club creating walks for its users to view and repeat.

Another improvement that could stem from the login system is the creation of an online forum and review system integrated with the web application and mobile application. This would allow users to find and review tours they have already walked and review them, leave a thumbs up/down or 1 to 5 star rating, as well as leave comments about the tours and how they could be improved e.g. taking a different or more scenic route. This system would be integrated with social networking sites such as facebook and twitter to allow users to share their tours and provide free advertising of the application to their friends and followers. The forum part of the improvement could be used to allow users to ask questions about the application creating a map of questions of answers that will help all users with any issues that occur with the applications and users to request for a specific tour to be created by a trusted source e.g. requesting a walk in a certain area to be created by a walking club that give creates good in depth tours.

One issue that occurred due to a lack of time and man power was that the photos taken on the application could not be sent or displayed on the server properly. This issue could be worked out fairly fast if more time or man power was available during the programming period of the applications.

Another improvement to the application would be the integration of an on screen map on the mobile phone; this would allow users to see their location on screen and make waypoints more accurately and to view their tour as they create it. This would also make the application look more professional. This could also lead to other improvements such as the creation of an on phone tour editor allowing users to edit saved tours on their phone and re-upload it to the server. It could also lead to the creation of a tour downloader, so that users could search for tours of an area download it from the server so it can be followed e.g. following a tour created of the local landmarks or a running route.

Another potential improvement would be the creation and implementation of a offline storage system that would save tours on local storage if no signal is available so it can be uploaded later when a signal is available as well as storing the waypoints and photos locally.

Many updates and improvements could be made to the application depending on the level of maintenance that is planned in the future and the amount of money that is invested on improving the application.

8 Things To Watch For When Making Changes

When making modifications to the code it is possible to influence other parts of the system. This could cause the program to break or act in an unintended manner. The main components that could be affected by change are the local database handler and the HTTP post sender.

When making changes to the data structure of a Route object or a Waypoint object changes to how the application writes data into the phones database, how it transmits the data to the web server and how the web server processes the data will be affected.

The phones database has been implemented with tables that have attributes assigned to them. The tables will need to have their attributes changed if additional data is to be recorded for each tour. This would mean the method used to construct the tables will have to be edited with the new required attributes. The methods in which data from an object is to be filled into the new table will also have to be changed for data to be stored as intended.

The HTTP post sender works by creating JSON objects and populating it with the data from the Walking Tour that will be sent. If additional/different data for the walking tour is added, the tuples to be encoded into the JSON data structure will have to be changed. The method used to build the POST message will have to be corrected to suit the new data structure.

The web server will also store walking tour data that have been submitted from the android application. The database will also be used to display walking tour content on the web page, therefore any problems in code that lead up to this will cause a problem in the website accomplishing its task. A change in the Waypoint or Route data structure would also mean the web applications code will have to change how it processes the POST message. The code to be changed are the functions that are used to populate the route and waypoint object. The attributes of the MySQL server will also have to change to reflect the new data structure.

This demonstrates that changes that happen lower down in the program will cause a major changes to how the system will act up until the point of displaying a walking tour. Care should be taken when making changes to the model side of the code, as the Model View Controller software pattern is used in this project. Modifications to the GUI (View) is less likely to create a knock-on effect which affects other parts of the program.

9 Physical Limitations Of The Program

Our program was specifically produced for the Android Operating System, and as such will not be able to be run on any other OS. This is the main physical limitation of the application, and can only be solved by porting the application to another OS, requiring large amounts of changes to the original code, as well as large amounts of time.

The application that we have created as part of out project has an almost negligible memory usage, and only requires the use of local storage for temporary storage of photos taken by users through the application, as are the tours created by the user. Files stored temporarily in the local storage of the mobile phone are sent to the server side of the program to be stored there.

The application requires the following hardware for guarantees successful running, local storage of photos and tours and communication between the server and the application:

- Android Operating System: Minimum version 2.3, built for version 4.4
- Internet connectivity
- Built-in camera
- Local storage. Minimum: $64MB$, recommended $100MB$

The variables containing the values for the GPS co-ordinates are the most vulnerable within the application to corruption, due to possible differences between the number of decimal places used in the application and on the server side of the program. Protection against this has been implemented within the application by setting the number of decimal places of the GPS variable to 3 when it is accessed from the Phone System.

10 Rebuilding And Testing

The purpose of this section is to explain to future maintainers what they need to do when rebuilding or maintaining the program, what tests should be run when maintaining the programme, and how to add new tests to the test specification.

Maintainers will be able to find all source files and documentation for the application on GitHub `https://github.com/eitharus/fuzzyNinja/tree/lewis-new/SRC/Controlled`. All files are stored in an easy to follow file structure. All code for the web interface is stored in the "Web" folder, all the code for the android application is stored in the "App" folder, and all testing documents are stored in the "Test_Docs" folder.

To rebuild the android application it is recommended to import the source files into an appropriate integrated development environment (IDE), such as Android Studio or Eclipse. The same should be done for the web interface, using an IDE such as Netbeans. All code is commented and contains JavaDoc which contains information on what all the classes aim to do and how all the methods work.

Once the rebuilding has been completed it is important that the code is tested to ensure the programme still works and no functionality has been lost. Included in the "Test_Docs" folder is a test specification which includes a number of repeatable tests which should be carried out every time the programmes source code is changed. The test specification specifies what input should be inputted into the programme and what the output should be for each test. If the output is not what is expected then test fails. If the output is what is expected then the test passes. If a test does fail it is important that you fix the source code in order for the programme to work effectively and functionality remains the same.

The test specification is written is a Microsoft Word document so adding a test to the specification is simple. To add a test you simply add a new row to the test table and fill in the details of the test such as test reference number, test content, input, output, and pass criteria.

11 References

References

[1] Group 15, *Design Specification*, 4.1 Interface Specification.

[2] Android, *Location Strategies*, `http://developer.android.com/guide/topics/location/strategies.html`

[3] Android, *Connecting To The Network*. `http://developer.android.com/training/basics/network-ops/connecting.html`.

[4] C.J. Price and B.P. Tiddeman, *Requirements Specification*. 29 Jan 2014, SE.QA.RS.

6.3 Personal Reflective Reports

6.3.1 lpd1

Initially, I was quite excited to begin the group project as I saw it as an opportunity to really test myself. In a way, that has proven true - it has not tested my abilities as a developer, but as a team member and leader.

At first, I chose to become deputy leader, and allowed Josh Moss to be the group leader, as I thought that it would allow more time for me to focus on the programming side of things and that he would be a better leader than I would - mostly due to his army background.

The first document seemed to go quite smoothly - we were already one member down as one had decided to retake the first year, and as such had slightly more work to do each. Josh elected to compile the document, but I was not given any work to do, but instead was to work on the testing documents with Owen. Things seemed to go well, until we received our feedback, and discovered that there was a large number of errors that had come from poor formatting and grammar when compiling the document.

After this, I was chosen to be the group leader, and I think that it was in that role that I began to work my best. I feel that as group leader, the group members each had a fairly equal allocation of work, and seem to think that I was a good leader. I took on a lot of responsibility myself - particularly during integration and testing week, when I intended to work on documentation after finishing the web application, but due to Josh's absence

6.3.2 owd2

Before starting the group project I was apprehensive as I knew that the project was going to be a lot of work and knew it was going to take up a lot of time with meetings and strict deadlines. During our first group meeting we quickly got down to work even though the client did not turn up to the initial meeting. We quickly discussed the strengths and weaknesses of each team member and discussed who would be best suited to each role. It was decided that I would be test manager as I felt that testing was one of my main strengths. It was also thought that Josh Moss would be a good team leader due to his experience in the army.

After the first meeting we discovered that one of the team members, Himalya, decided he was going to retake first year, so we was one person down before we started any work on designing the application. We decided that it would be sensible to have more than one meeting a week, so we decided we would have an extra meeting on a Wednesday afternoon to discuss process on the project and review any documents that were about to be handed in to the client for review.

The first document that had to be produced was a project plan. The plan included details such as a system overview, project time scale, user interface design, GUI design, and a risk analysis. The majority of this document was completed by Jack and Jamie, who were in charge of the user interfaces and GUI for the application. The document was then put together my Maria who was the QA manager at that point. Unfortunately, due to confusion of hand-in procedures, the document was submitted late as it was thought that the document only needed to be uploaded onto GitHub.
The next document that needed to be produced was the test specification. The test specification included system tests for the application and the web interface. This document was completed by Lewis and I. I wrote system test for the Android application and Lewis wrote tests for the web interface. I completed the test specification and uploaded the file to the groups Facebook group ready for review in the next meeting. Unfortunately, due to other commitments outside of university I was unable able to make the review meetings or the meeting where we handed over the test specification to the client.

After the test specification had been submitted we had a meeting to discuss the next document that had to be produced, the design document. During this meeting we discovered that Maria had decided to leave university to pursue other interests, so I took over the role as QA manager as I felt as test manager the majority of my work had been completed until the application had been designed and implemented. So, at this point I was testing manager and QA manager. It was also discussed that Josh should step down as team leader and allow Lewis to step up as team leader. This was due to poor leadership, poor attendance at meetings and failure to complete tasks such as uploading to files to GitHub.

My main task for ITW was to write interfaces for JUnit tests and ensuring that the test specification was up to date. I also wrote some of the code for data to be written to the local database on the Android device using SQLite. Once the project had been completed the completed project was uploaded to my Android device for me to complete system tests. On my first test run it was discovered that the Android application was talking effectively with the web interface, it was creating a walk but none of the data was being added to the web interface. This error was diagnosed and fixed by Lewis. During my second test run the application work effectively and was able to send data to the web interface to be viewed on the website. While completing the system tests it was clear that some functionality had not been added yet such as form validation or error messages. These functions were added and the next test run I completed had form validation and error messages appearing when needed.

I felt I worked well during the group project. Every task I was given I tried to complete to the best of my ability. I felt the team worked well together and everyone got on with everyother in the group. There were times of stress, especially during ITW when we had to rewrite the Android application on such a short deadline. Our team was able to continue working effectively when team members dropped out and roles were quickly reassigned to ensure minimum disruption. Most deadlines were met on time and if they were not met there was a reason given. During the start of the project I felt my attendance in meetings could have been better, but as the project progressed I attended all relevant meetings to ensure the work the group and I produced was to a good standard.

6.3.3 jaj42

The first thing that i was worried about before starting the group project was what i would be able to bring to the team. I feel that i did not have a clear strength in any sections of the project such as, programming, design, testing or documentation; but instead I was evenly skilled at every aspect. During the first meeting we identified our strengths and weaknesses in order to determine what role well take on for the group project.

The role that I would take on is the groups version control expert. The version control system thats required for the project is Git and GitHub is where the repository will be hosted. Up to that point Ive never used Git before but there was a workshop to be ran to help groups setup their repositories and learn how to use git. I did a lot of my own research into how Git should be used and thought of ways members who were not/less experienced with Git could make use of it easily. Ive recommended the use of a Git client, SourceTree (`http://www.sourcetreeapp.com`), Ive built up my understanding of the tool and thought it will be easier for me to help fix issues that arise by going through the steps to fix it using the GUI. I wanted to hold off creating the repository until after the Git workshop to ensure the repository created fulfills all requirements the assignment needed. Our current group leader at the time, Josh had created the repository before the workshop. I felt my role as version control expert was undermined as I wasnt given the opportunity to complete my own task. My other task was the continuous maintenance of the repository to ensure it follows the standard set by the QA documents.

Due to the reduction in group members we were given multiple roles to do such as various sections of documentation. Some of the sections were not optimal to give to members who were not as involved in that section, sections I was not familiar with I felt were completed to the best of my abilities.

I felt it was a good experience to be able to know what entails working in as a team and has taught me valuable skills that will definitely benefit me in the future. My own skills as a computer scientist has benefited greatly as i am a lot more confident in my abilities and have managed to improve and learn multiple aspects. During the phase of the project I felt that my time management should off definitely been better, such as preparing myself with adequate time to arrive to meetings on time and to complete tasks. My team work skills have improved over the last phases of the assignment as I felt I was able to more effectively communicate with group members, although I should of been less reluctant to seek help from group members as this would of saved a lot of time. I find our current group leader Lewis helped push everyone to reach the stage where we are now and without his strong leadership the project would have fallen further behind. My blog can be found at `http://groupproject.jafnee.com`.

6.3.4 jap38

We began our project with one team member less than all other groups as one of our potential members left before the official formation of the project group. In our first meeting we allocated jobs based upon our description of our own individual strengths and weaknesses. I was given the job of User Interface Design, along with Jamie McLoughlin. Joshua Moss was the only volunteer to be the Team Leader, so we gave him the position.

The first group of documents that we had to complete as a group was the Analysis and Design document, where we had to complete a basic design of the Mobile Application, along with a decomposition and analysis of the Project Specifications given to us for the project. Our group immediately had problems with communication, specifically problems with job allocation.

Another major problem we had during the production of the Design documentation was a major formatting error, caused by a lack of standardisation in the programs we used to author the documents. Jamie and I completed the initial designs for the User Interface and handed them in on time, along with almost all the other documents required for the Design.

After we handed in the Design Documentation Maria left the team to persue a year out. This left us 2 people short compared to the majority of other groups in the project. Implementation and Testing week was hard. Due to the fact we were now 3 people short on our team, after Josh decided not to be there, meant that we all had to begin work on the application early and end late. I was working, with Jamie, on the User Interface for the mobile application as well as beginning initial debugging on pieces of back-end code that other members of the team had completed. This enabled us to progress through the coding at a fast pace, although it would have been much faster with a full strength team.

The evidence that the group worked well together, and worked hard, is shown by the fact that we managed to hand in an project that fulfilled the basic requirements, even with a significantly understrength team. We managed to fulfil the basic requirements set, but didn't manage to complete any of the extra requirements, such as the ability to browse previously taken photos through the application.

Overall I feel that I performed well during the group project. I was punctual to arrive to most of the meetings, and throughout Implementation and Testing Week I regularly arrived early and completed all of the work that was allotted to me throughout.

6.3.5 gad16

My first task for the project was to draw up a use case diagram of the primary components of the android application. This took me longer than expected as I had to consult some online information on use case modelling.

When the use case was finished, I was instructed to write a class description for the Android app. For this task I had to expand on the use case diagram I drafted to the group on 05/11/13. This was a fairly simple task, though it was fairly time-consuming as I had to keep consulting the QA documentation to ensure my work was in keeping with the project standards.

After completing the initial use case diagram, I was tasked with drafting a UML class diagram for the back-end classes. Though I had to brush up on some UML syntax, this diagram wasnt too difficult to complete since I had already designed the main classes in the previous task.

The component diagram involved modelling the interactions between each component of the application. After brushing up on my UML knowledge, I drew up this diagram which includes the front-end GUI components designed by Jack and Jamie. Because this was mostly combining my designs with theirs, this task did not take very much time.

My final role in the design phase involved drafting some initial interface code that would become the foundation for the final application. I finished this fairly quickly and easily due to my experience with Java programming and the presence of the previous documents I had written. While this document only required stub methods, the information these methods contained helped eliminate a lot of the early work in the implementation phase, which helped the group move ahead into testing the application.

Before implementation and testing week began, I wrote some code for HTTPPost-Sender in an editor. Though I could not test any of this code yet as the other components of the application were missing, I still decided to write it so I could debug it right away when implementation and testing week began.

Most of implementation and testing week has been covered in my reflective journal. At the start of the week, I brought my code for HTTPPostSender and began working on the Route and Waypoint classes. Initial progress was slow since I wasnt used to the Mac OS user interface and problems the whole group was having with Git and the Android SDK didnt help much. However, progress dramatically began to pick up towards the middle of the week as I got used to the environment. I spent much of this time debugging and refactoring my code, as well as integrating it into the front- end components. By Thursday we had our first successful build of the application and I carried out some system testing.

6.3.6 jam67

he project went well from the start with all but one group member attending the first meeting, (Himalya who had decided to retake the first year) and we got straight to work discussing the project and our strengths and weaknesses. We all agreed on suitable roles to take on and decided that Josh would be group leader with Lewis as deputy as Josh had experience with the army. I was given the role working on the User Interface with Jack.

We organised meetings for Wednesday afternoon as we were all free and able to attend. We began work on the project plan the first document that needed handing in. I worked on the project plan designing the basic user interface and working on the original plan of how to navigate the application. This document was however handed in late due to some confusion with the QA manager Maria who also dropped out soon after leaving us two members down.

The next document that needed handing in was the test specification this was mainly completed by Lewis and Owen and was reviewed by the group. This was handed in late also. It was discussed that Josh would step down as group leader due to a lack of attendance, communication and mistakes with handing in work and using github. Lewis was made project leader and did an excellent job of distributing tasks and making sure everything went smoothly. With a new project leader the group banded together to create the design document and this was handed in and submitted on time. After the design document we began work on the prototype application and web side of the application. It was decided that Lewis was the best person to do the web side as he has experience creating web sites. I worked on getting the user interface prototype working. We decided to wait until after the exams to begin work again as we needed to do revision.
After the exams we started work on the final application in coding week. The group worked well together, however Josh told us he would not be around during coding week and we have not heard from him since, leaving us with only 6 members to do the actual coding and creation of the final application and final report. He did leave some work that he had done the prototype but it was poorly written, confusing and had no comments so we had to start again, it caused us more problems than it saved.

The group worked well together with each other and everyone left put the effort in to get the project completed on time. I was working on the user interface mainly but also worked on the database and the taking and displaying of the photo on the phone. Overall the project went well and all remaining group members worked well together and we all got along well. I believe I contributed to the group attending whenever meetings were called and doing sections of work on time and to the best of my ability. I believe that the project was a success even though not all the requirements were met such as displaying the photos online, as with our setbacks to the number of group members we had we still managed to create a working application with most of the requirements met. I found the project to be both difficult in parts and rewarding as I had not done any android work before this project and was not used to heavy programming being a business information technology student which has less programming based than other computer science courses. This meant I had to spend time learning and researching for some aspects of the project and sometimes needed help from the group, this was always given. I enjoyed the project and working in a group to meet the deadline, I enjoyed creating the user interface particularly, creating a sleek useable design.

6.4 Revised Project Plan and Design

6.4.1 Project Plan

CS221 Group 15
Project Plan

Authors: lpd1; jap38; jaj42; owd2; jam67; gad16; jcm14; mab68
Version: 1.0
Config Ref: SE_15_ProjectPlan
Date: 07-11-13
Revision Date: 17-02-14

Department Of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

Contents

1 INTRODUCTION

1.1 Purpose of this document

The purpose of this document is to demonstrate our plan for development of the Walking Tour Creator project.

1.2 Scope

This document will include an overview of the proposed system, including:

- Platform choice
- A description on target users, including their expected skill level
- Use case diagrams and descriptions
- Core user interface features and a planned layout
- A Gantt chart to show the expected project timescale
- Project risk analysis, where we predict possible issues and state how we intend to prevent or prepare for these.

1.3 Objectives

- Provide a brief overview of the proposed system, including what technologies we expect to encounter.
- Show the major components of the proposed system.
- Show how elements will interact with each other.
- Display a basic model for the user interface.
- Identify possible issues with the development of the system and explain how these issues will be resolved.

2 System Overview

The Proposed system is:

- An Android Application to record walking tours around places of interest and to include images and a textual description of each place
- A web interface to display the walking tour and the related information to the user

2.1 Platforms and High-Level Architecture

The following is a list of prospective technologies we expect to use throughout the project.

2.1.1 Android Platform

The client wishes to commission an application solely for the Android mobile platform. The minimum Android version we will be targeting is version 2.3, but our application will be written for version 4.2.

2.1.2 Android API and Java Language

We will use the Android API (Application Programming Interface) and the Android SDK (Software Development Kit) version 17, as well as using the Java programming language to develop the application.

2.1.3 PHP

We will be using PHP as the primary programming language backing our web interface. We will use to communicate between the server and the database, as well as handle HTTP POST requests. PHP is a widely-used programming language available on most server configurations.

2.1.4 MySQL

As it is one of the most widely-used database platforms, we have elected to use it in lieu of other database management systems such as PostgreSQL.

2.1.5 Google Maps API

The Google maps API will be used to display the tour on the web interface. We chose to use the Google Maps API due to its' ease-of-use and the high-quality of the documentation available.

2.1.6 Internet Connectivity

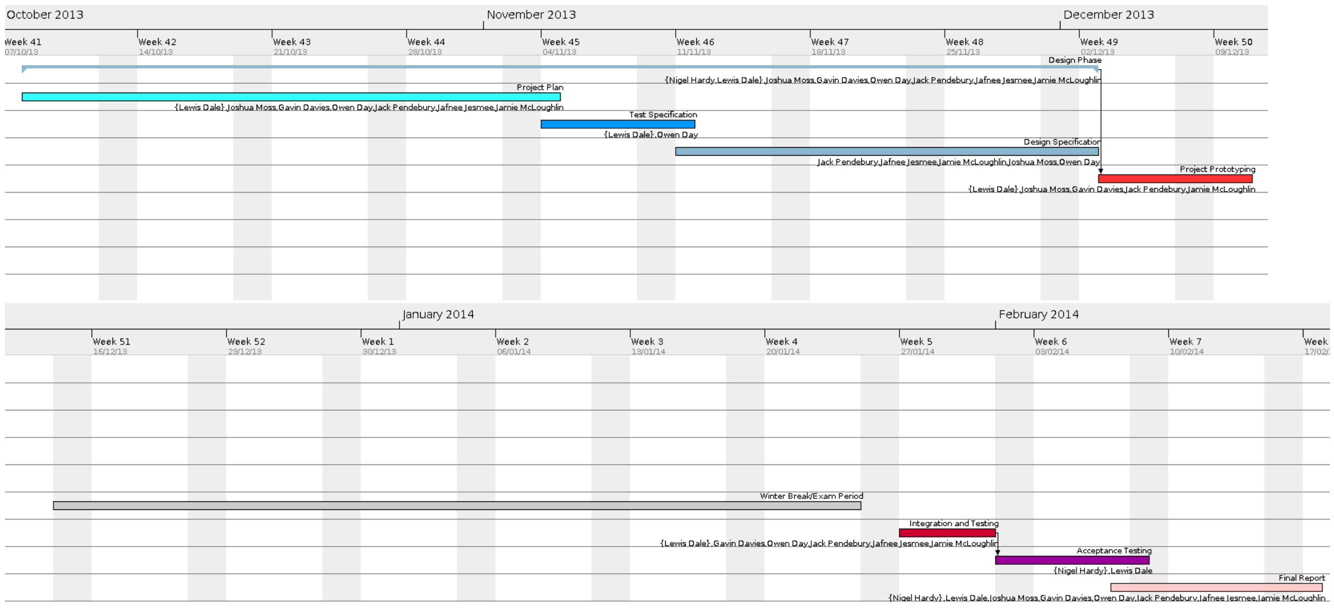
- Wifi
  - We are required to specify Wifi access for internet connectivity, to prevent users from incurring massive data charges [1].

- 3G/Mobile Data
  - We are also to required that using cellular internet is needed, again to prevent the user incurring large data costs [1].

2.2 Target Users

As per the requirements specification the ”software will be used by second year Computer Science students” [2] so our target users are generally between the ages of 19 to 25. However, if the software is built to the requirement specifications and the software is intuitive to use it will not be difficult for members of many other age groups to use.

3 Gantt Chart

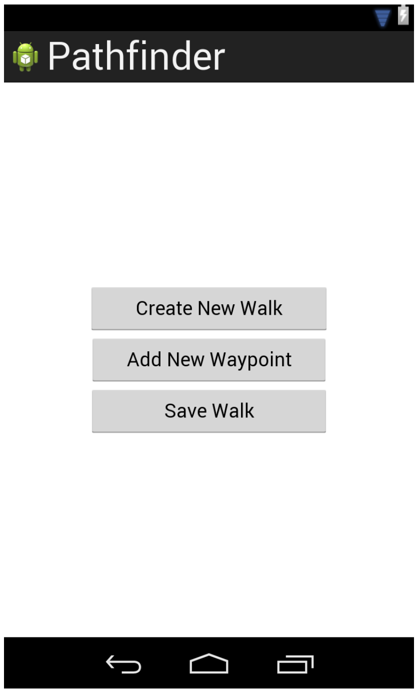


Roles

- Nigel Hardy *Project Manager*
- Lewis Dale *Group Leader/Web Developer*
- Gavin Davies *Lead Android Developer*
- Joshua Moss *Android Developer*
- Jack Pendlebury *UI Designer*
- Jamie McLoughlin *UI Designer/Web Developer*
- Owen Day *Testing Team*
- Jafnee Jesmee *Version Control Expert*

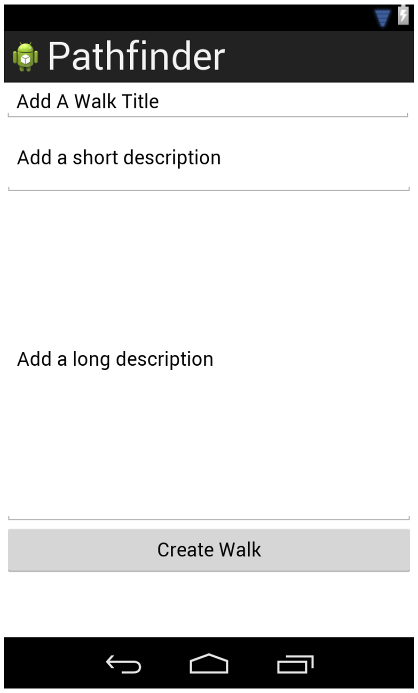
4 User Interface Design

4.1 Main Screen



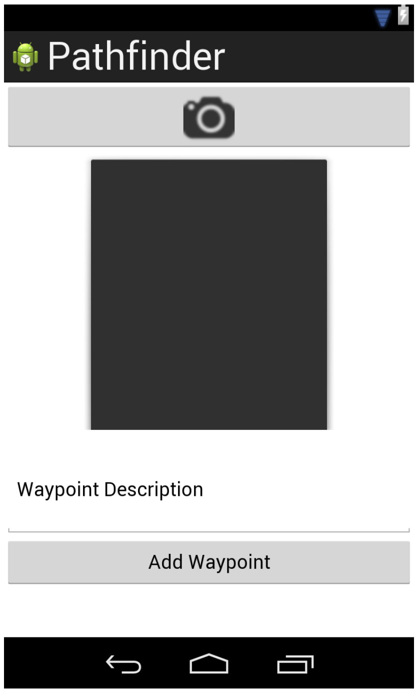
This user interface design shows the first screen that users will see when the application is started on the device. The buttons shown will be the primary points of navigation within the application. The button labelled ”Create New Walk” will take the user to the screen which will allow them to create a new walk. The ”Add New Waypoint” button will, providing there is a tour currently running, take them to a screen allowing them to create a new waypoint in that tour. Finally, the ”Save Walk” button will prompt the current tour to be sent to the web server, allowing it to be displayed on the website.

4.2 Create Tour Screen



This screen is shown to the user when they press the ”Create New Walk” button on the main screen. It prompts the user to enter a walk title, with a max length of 30 characters, a short description with a max length of 100 characters and a long description with a max length of 1000 characters. If the user has not filled out any of these fields when the ”Create Walk” button is pressed then an error will show asking the user to fill the fields. If the fields have all been filled, then an entry for the tour will be created in the database and the applications’ current running tour set to the newly-created one.

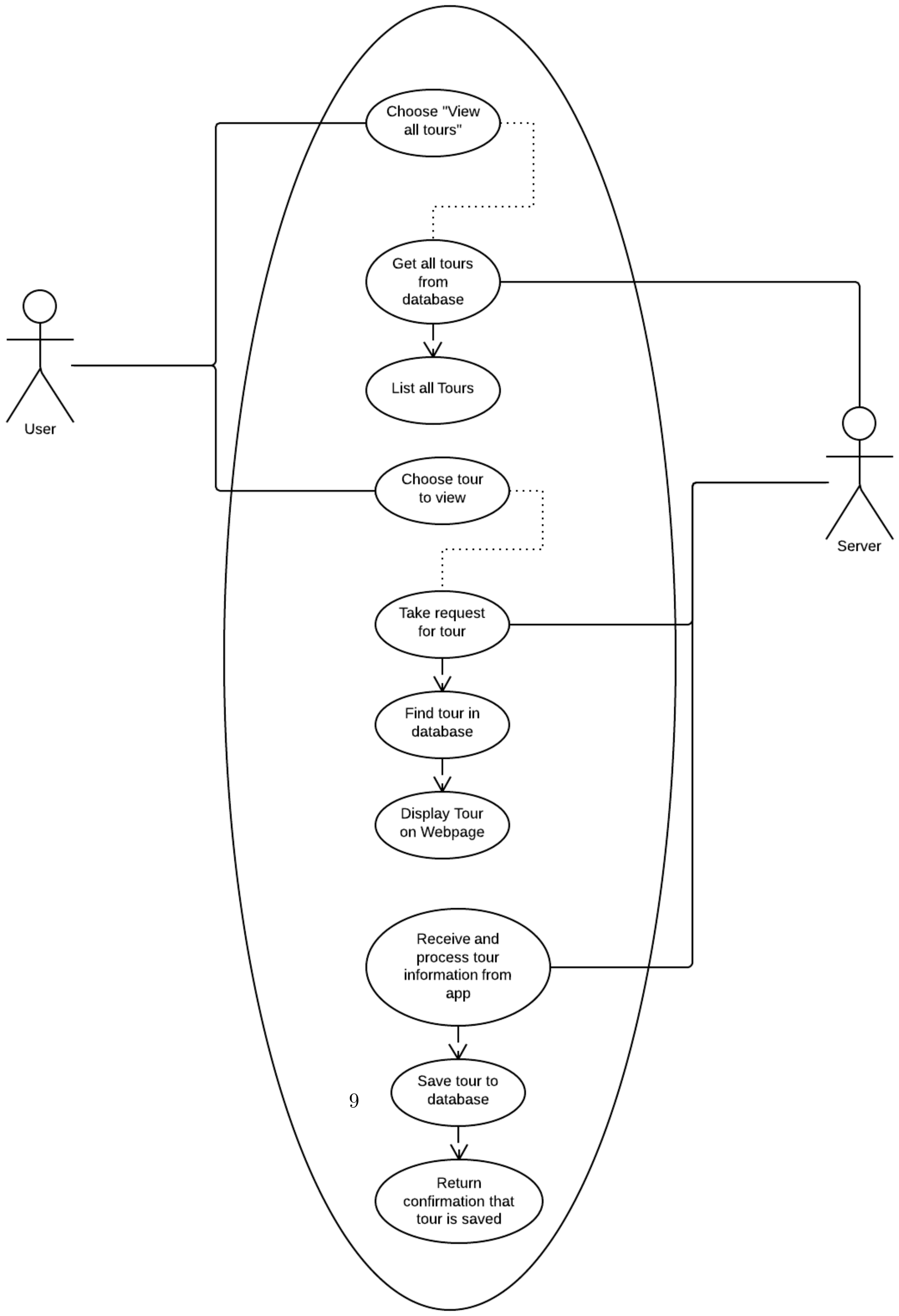
4.3 Add Waypoint Screen



This user interface will be shown when the user selects the ”Add New Waypoint” button on the main screen. If the camera icon is pressed, then the default Android camera application will start, allowing the user to take as many pictures of the current waypoint as they want. The latest picture will be displayed in the Image View widget, the black box just below the camera icon. The user will also be able to add a short description of the current waypoint (100 characters maximum). When the user presses ”Add Waypoint” the waypoint is stored in the local database in the relevant tables.

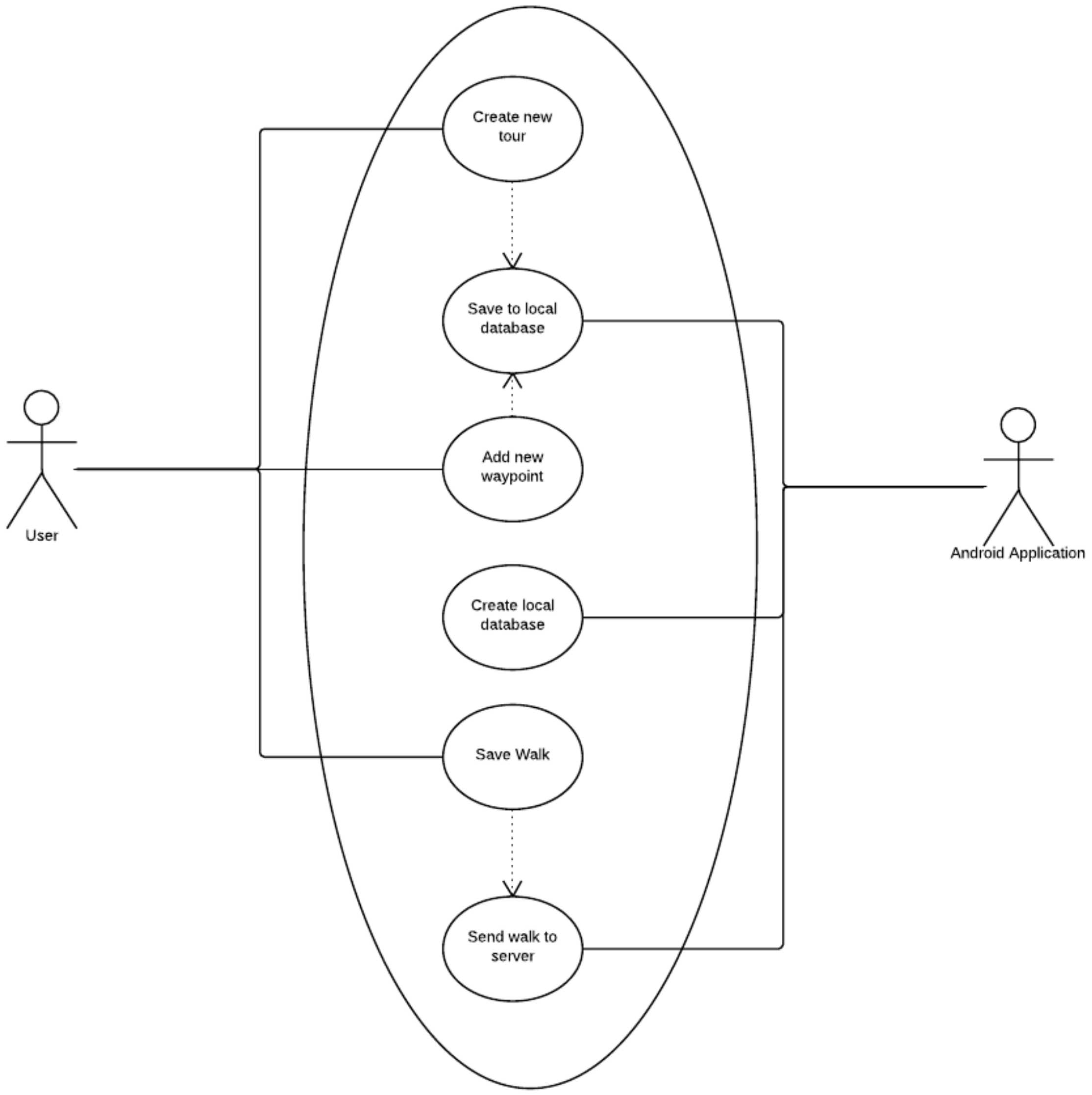
5 Use Cases

5.1 Server



Use Case	**Description**
Choose ”View All Tours”	The user will select an option on the web interface allowing them to view all of the tours that have been submitted. They can then select one of those tours to view it.
Get all tours from database	When the user selects ”view all tours” the web server will submit a query to the database that will return all of the submitted tours.
Choose tour to view	From the ”List all tours” page, the user selects a tour that they wish to view, which will then display all of the waypoints from the tour on a map, as well as descriptions and associated images from each one.
Take request for tour	The web server receives a tour ID as a *GET* variable, and queries the database to find if a tour with that ID exists, and if so displays it on a webpage.
Receive and process tour information from app	The server will receive some data via POST from the app, which will contain the details of a new tour, and all of the waypoints associated with that tour and their images. Then, all of this information will be saved to a new entry in the database

5.2 Android Application



Use Case	**Description**
Create New Tour	This use-case describes the user selecting the "Create New Walk" button. When the user pushes this button, A screen will show allowing the user to create a new tour, filling the fields out and pressing the button to create the walk.
Save To Local Database	The application will save the new information that has been provided to it into a local SQLite database. If the user has submitted a waypoint, it will be saved based on the current walk ID. If the user has submitted a new tour, and ID will be generated for it.
Add New Waypoint	When the user chooses the option to add a new waypoint, the application will show them the graphical interface that will allow them to add images, a title, and a description, and then save a new waypoint on the tour. When the user chooses to save the waypoint, it is then saved to the local database.
Create Local Database	When the application attempts to save to the local database, if that database does not already exist then the database class will contain code that will create a new, empty database and format it according to the database specification.[2]
Save Walk	If the user chooses to save a walk, then the application will store the walks information to the local database.
Send Walk To Server	After the user has saved the walk, it will then be formatted into a pre-defined data structure and sent to the server, where it will be parsed and stored on the external database.

6 Risk Analysis

Risk Event	Risk Level	Measures	Risk Chance
Missing Group Members	Low	If a member is missing from a meeting work should proceed as normal. The missing team member should read the minutes to get informed. Team members that do not show up to meetings and do not perform their jobs will receive a warning.	Low
Communication between application and server	Medium/High	Problems may occur with application-server communication. This must be tested before project submission	Low
Missing features	Low/Medium	Some features may take more time to implement than expected or a group member may not be able to complete their work. If a group member foresees such an issue arriving, they should inform the rest of the group and/or request help from the project manager or leader.	Low
SQL Injection	Medium	There is a risk of SQL injection as the application uses database communications from user input, so all input must be sanitised before it is queried on the database.	Low
Server downtime	Medium	There is a risk of unexpected downtime with the web server, which could interrupt application-server communications and web interface accessibility. Backups and stress-testing for the server must be performed to ensure that it is capable of supporting the application.	Low
Running out of time	Medium	In-group deadlines before the actual deadlines to make sure everything will be completed on time.	High
Poor interface quality	Low/Medium	Information should be displayed clearly and the user should be able to navigate easily.	Low/Medium
Git Desynchronisation	Low/Medium	All the group members should make sure that they correctly upload all of their work in order to avoid desynchronisation.	Low
Late submission of the document	Low	In-group deadlines before submission deadlines to ensure that all documentation is appropriate and of a good quality. If any problems occur and a group member cannot finish their work by the deadline, they should inform the group manager or leader and seek advice where necessary.	Low
Low documentation quality	Low	All documentation should be reviewed by the group and in the event that the submission is of a low quality, be returned to the group member for improvements.	Low

7 References

References

[1] Android, *Connecting To The Network*. `http://developer.android.com/training/basics/network-ops/connecting.html`.

[2] C.J. Price and B.P. Tiddeman, *Requirements Specification*. 29 Jan 2014, SE.QA.RS.

8 DOCUMENT HISTORY

Ref	**Date**	**Status**	**Description**
0.1	31-10-13	Draft	Initial document creation.
1.0	07-11-13	Release	First release.
2.0	17-02-14	Release	Rewritten and re-released version complete.

6.4.2 Design Document

CS211 Group 15

Design Specification

Authors: lpd1; jcm14; jap38; jaj42; owd2; jam67; gad16
Date: 02-12-2013
Version: 1.0
Config Ref: SE_15_DesignSpec

Department Of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB

Contents

1 INTRODUCTION

1.1 Purpose of this document

The purpose of this document is to show the design specification for the Walking Tour Creator.

1.2 Scope

This document will include a decomposition description which will include programs in the system, significant classes in each program, modules shared between programmes and mapping from requirements to classes; a dependency description which will include component diagrams for all programs and inheritance relationships; an interface description; and a detailed design of the Walking Tour Creator which will include sequence diagrams, significant algorithms and significant data structures.

1.3 Objectives

- Describe each of the programs and the relationship between programs
- Provide a short description of the purpose of each class
- Describe the relationships and dependencies between modules
- Provide component diagrams for each program, showing the method links between modules
- Produce an interface description which includes:
  - The name and type of the class or interface
  - Classes or interfaces which it extends
  - Public methods implemented by the class or interface, including parameter names and types for each method
- Provide sequence diagrams which shows how the classes work together for the major operations of the program
- Provide a textual description of difficult parts of the system that need to be implemented (algorithms)
- Outline significant data structures using class diagrams to show entity relationships between classes, along with object diagrams which show how static relationships in class diagrams work types for each method

2 DECOMPOSITION DESCRIPTION

2.1 Programs In System

We can split the overall system down into two programs, which allow the system to be used, and the database to be accessed, both on the internet and on any Android based mobile system.

- The mobile application written in Java for Android
- Web-based application to allow a user to view submitted tours

While these two programs will be handled separately from each other, they will share certain properties, such as similar data structures to handle the same type of data, and both will use the same database to store and retrieve data
The Server is PHP based, and handles data to and from both of the programs and the SQL Database.
We will be using a PHP server to process incoming and outgoing database requests from both the web application and the mobile application.

2.2 Significant Classes In Each Program

2.2.1 Significant classes in Android Application

Qualities: The Application will make use of:

- Android.App.Activity;
- Android.Location;
- Android.Net

These are the core API section that are used by the app.

Model

Public Class Model
This Object is responsible for binding all other elements together, and being a base level interface for any Activity to access Instances of Objects or Constant Values.

Public Class GPSLocation
This Object is responsible for managing the GPS Location of the mobile device, and return this information to Public Class WayPoint.

Public Class Route
This Object is responsible for holding and managing all WayPoint objects. Waypoints will be stored in a Queue Structure, from Java.Util. There will be a method to receive a specific, or all Waypoints, there will also be a method to remove and add waypoints. Any other methods will be supplementary (non-core), or inherited.

Public Class WayPoint
This Object is responsible for storing the information about each waypoint. This information includes, location from Android.Location; location timestamp from Android.Location short description in String format; long description in String format; image list using Android.Widget.ImageView; an optional sound recording using Android.Media.MediaRecorder.

Public Abstract Class AbsHTTPPostBuilder
This Object is responsible for building a String to use in the HTTP Post Request. It does not upload it, however it manages the build, and stores it during this phase. It will use Java.Lang.StringBuilder to construct this. The Post will consist of all elements from WayPoint, and a waypoint id from Route

Public Class HTTPPostSender
This Object is responsible for sending the HTTP Post. It will also manage monitoring of network availability, which controls the ability to upload information. It will run its own thread, and send when it is possible.

View

Public Class ActivityLogin
This Activity is responsible for the display of the Login screen and sending information to the Model via its Controller. It will link to the Register and Main Menu Activitys.

Public Class ActivityRegister
This Activity is responsible for the display of the Register screen and sending information to the Model via its Controller. It will link to the Login Activity.

Public Class ActivityMenu
This Activity is responsible for the display of the Main Menu screen and sending information to the Model via its Controller. It will link to the Login, Route and Settings Activities.

Public Class ActivtyRoute
This Activity is responsible for the display of the Route screen and sending information to the Model via its Controller. It will link to the WayPoint, Settings, and Menu Activitys.

Public Class ActivityWayPoint
This Activity is responsible for the display of the WayPoint screen and sending information to the Model via its Controller. It will link to the Route, Settings, and Menu Activities.

Public Class ActivitySettings
This Activity is responsible for the display of the Settings screen and sending information to the Model via its Controller. It will link to the Route, WayPoint, Menu, Login Activitys.

2.3 Modules Shared Between Programs

2.3.1 Database Elements

The Database will be using MySQL engine. All parties will interact through this Database. The Mobile Application, will have an inhouse local database, to store waypoint, and walk information, while they cannot be uploaded. The local database connection will be declared in Public Class Model.

2.3.2 HTTPPost Object

This is a formatted String Object, that will be recognised by the Web and Mobile Applications. Its design will be used in HTTPPostBuilder, HTTPPostSender, and HTTPPostReciever.

2.4 Mapping From Requirements To Classes

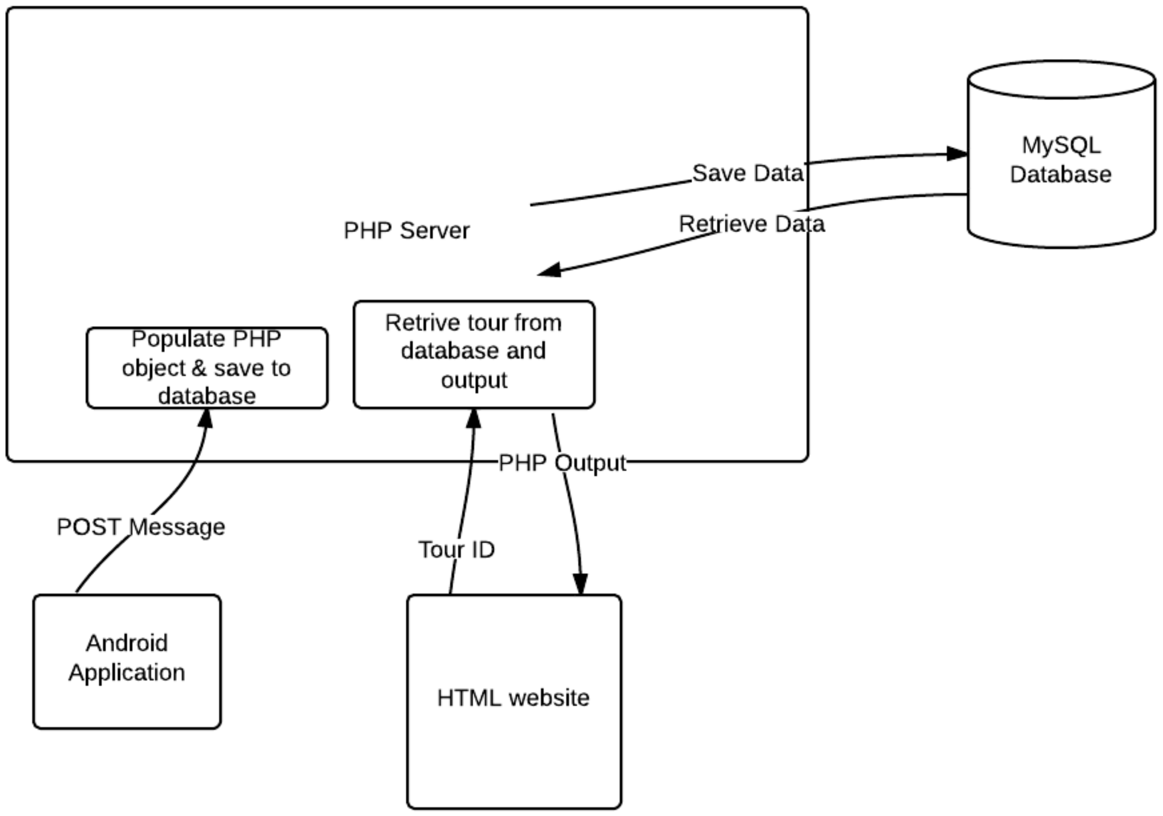
The Classes that actively translate to fulfilling the Functional RequFuzzyN-inja,irements are mainly located within the Android application, although the WTD will be required to complete a few of the requirements not completed by the Android application.

FR1	This will be fulfilled by the Android application. This is the most basic functional requirement, and will most likely be partly completed by [Main Class] as opposed to any specific class. Creating a new walk will be processed by the [Walk Creater] and will have the information sent to the server for later recall by either the App or the Website.
FR2	The Title and the descriptions of the walk will be stored as properties of the Walk, to be displayed upon access by either the Application or the Website. This will be done by displaying the appropriate properties when the file is accessed through the [Opening Class]. The Application will use the [Walk Creator] to save the properties when the walk is initially created by the user.
FR3	Adding the Process will be done through very similar methods as used through saving the Title and Descriptions, except that the Waypoint will be saved as an object with the GPS Co-Ordinates, Name, Description and Time Stamp saved as properties, and will be saved itself to the walk through the [Walk Creator]
FR4	This functional requirement will be completed using the same class as FR3, as the photo will also be saved as an object with different properties. The properties of the image will be the image file path, GPS Co-Ordinates, Name, Description and Time Stamp.
FR5	Within the Walk Creator their will be a button to Delete the currently in progress walk, which will link to [Deletion Class?] which will permanently delete that walk, along with all of it's properties and objects associated uniquely with that walk. Due to the fact that this could be disastrous for the user their will be a confirmation message displayed as part of the running of this class.
FR6	This functional requirement will be completed by the [Server Upload Class]. The information will be uploaded to the server using PHP, specifically using a HTTP POST method to upload the required information to a predefined URL, which will then be translated into the correct formatting by the server, and then uploaded into the database.
FR7	Should the user switch away from the application it will use a predefined Android local storage method to keep the user's data until such point that the user either closes the program or switches back to it.
FR8	This will be completed by the WTD. It will use the information loaded onto the server (see FR6 above) as well as a Google Maps API in order to correctly display the information required by the user on the map, correct to the corresponding GPS Co-Ordinates. This will be done by the client side web application's main displaying class, [Will the Web Site even have classes?].

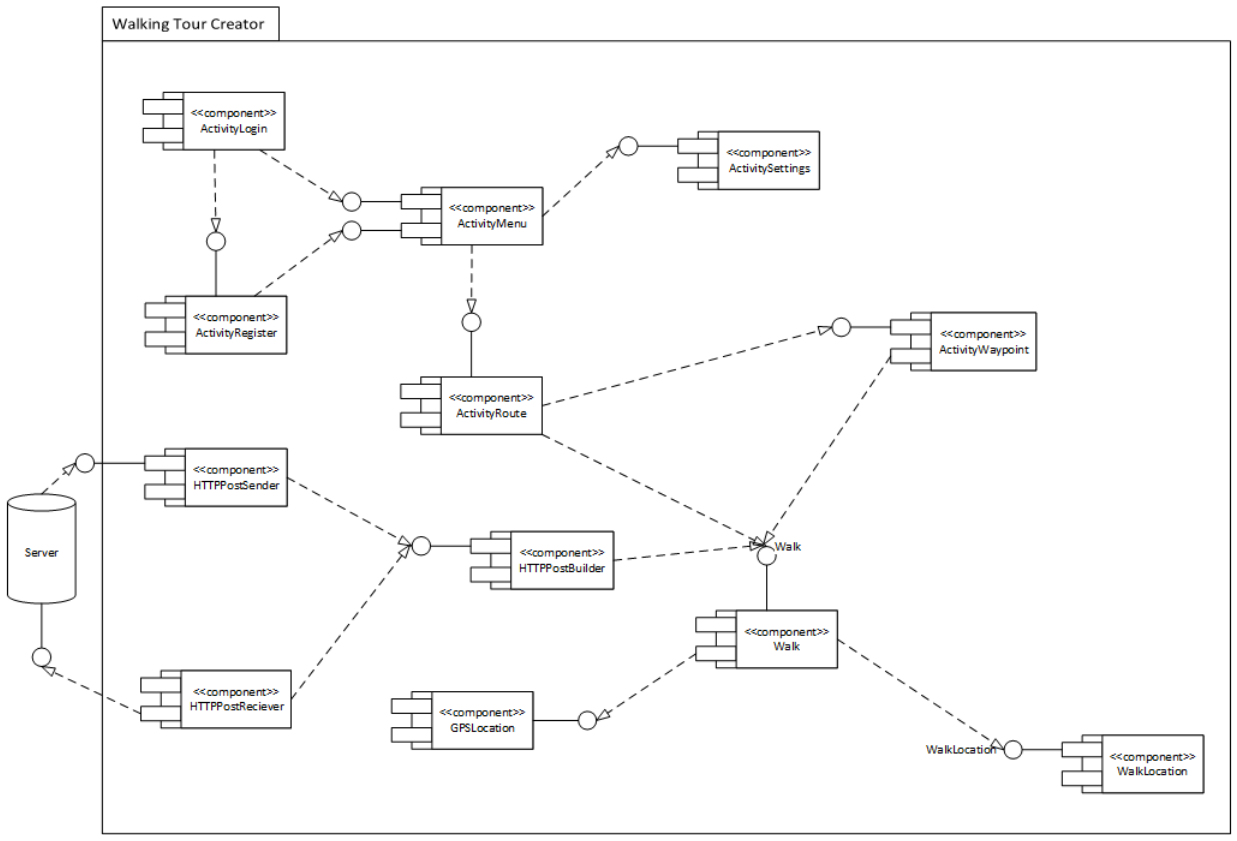
3 DEPENDENCY DESCRIPTION

3.1 Component Diagrams

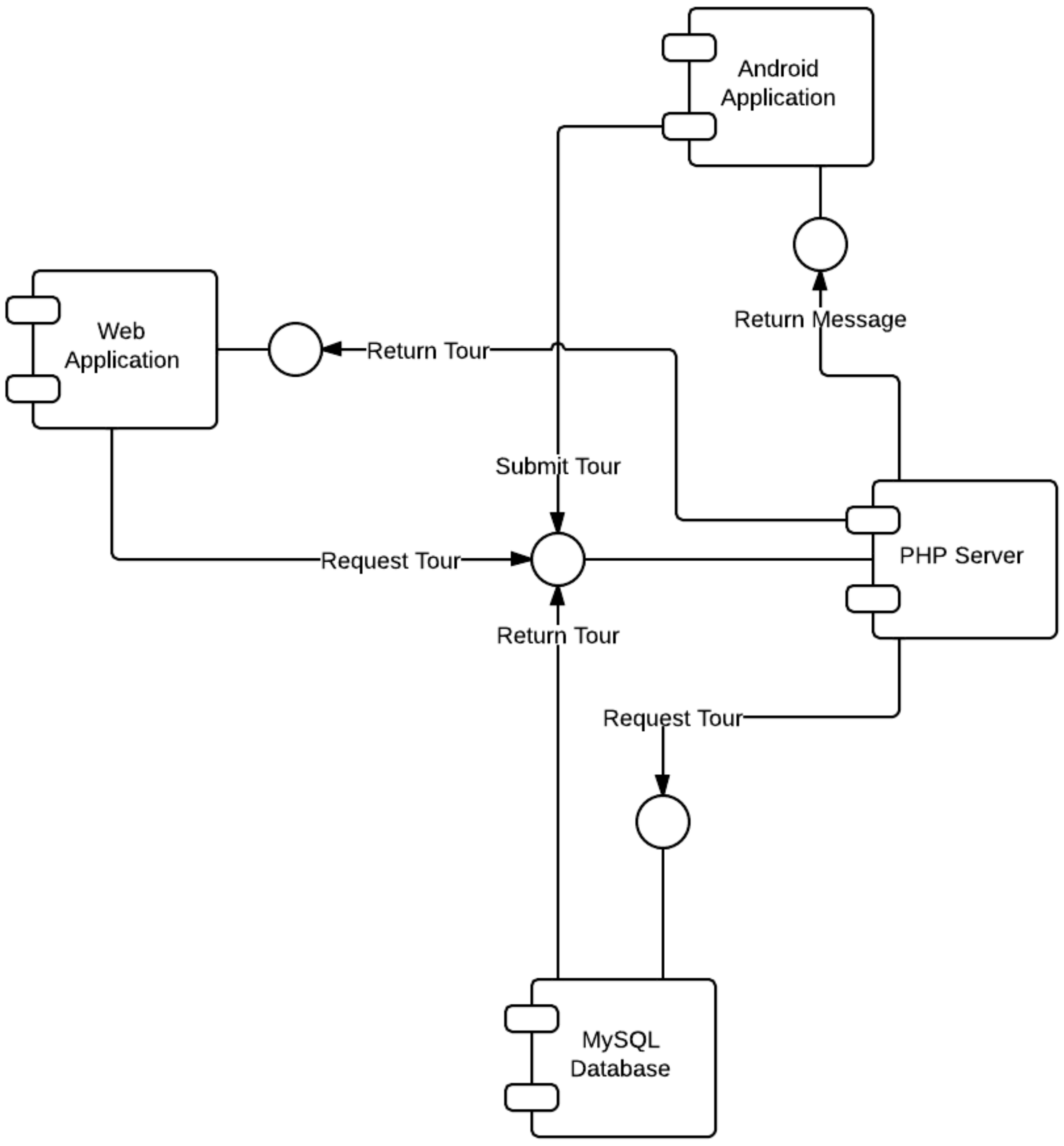
3.1.1 Deployment Diagram



3.1.2 Component diagram for Walking Tour Creator


Walking Tour Creator
<<component>> ActivityLogin
<<component>> ActivitySettings
<<component>> ActivityMenu
<<component>> ActivityRegister
<<component>> ActivityWaypoint
<<component>> ActivityRoute
<<component>> HTTPPostSender
Server
<<component>> HTTPPostBuilder
Walk
<<component>> Walk
<<component>> HTTPPostReciever
<<component>> GPSLocation
WalkLocation
<<component>> WalkLocation

3.1.3 Component diagram for Walking Tour Displayer



4 INTERFACE DESCRIPTION

4.1 Class 1 Interface Specification

HTTPPostBuilder:

```java
package cs221.group15.pathfinder;

import java.util.StringBuilder;

/** This class is responsible for building a string to use in an
    HTTP post request
* It is only responsible for building and storing the request
    string.
*/
public class HTTPPostBuilder {
  private StringBuilder postString; // The string used in an HTTP
    post request

  /** Builds an HTTP post request based on the attributes of a Walk
      objects
  * @param postWalk the Walk object used to build the HTTP post
    request
  */
  public void buildString(Walk postWalk);
}
```

GPSLocation:

```
package uk.ac.aber.cs22120.fuzzyNinja.pathFinder;

public class GPSLocation {
  private double latitude; // Represents a latitude, given in
    degrees
  private double longitude; // Represents a longitude, given in
    degrees

  /** Sets the latitude
  * @param lat the latitude in degrees
  */
  public void setLatitude(double lat);

  /** Gets the latitude
  * @return Returns the latitude in degrees
  */
  public double getLatitude();

  /** Sets the longitude
  * @param lng the longitude in degrees
  */
  public void setLongitude(double lng);

  /** Gets the longitude
  * @return Returns the longitude in degrees
  */
  public double getLongitude();
}
```

HTTPPostSender:

```
package uk.ac.aber.cs22120.fuzzyNinja.pathFinder;

import java.lang.Thread;
import android.net.ConnectivityManager;
/**
* This class is responsible for sending the HTTP Post.
* It will also manage monitoring of network availability,
* which controls the ability to upload information.
* It will run its own thread, and send when it is possible.
*/
public class HTTPPostSender implements Runnable{
  Thread networkThread;
  boolean isAvailable;
  ConnectivityManager cm;
  /** Sends an HTTP post request to the web server
  * @param request the string obtained from HTTPPostBuilder
  */
  public void send(String request);
}
```

Walk

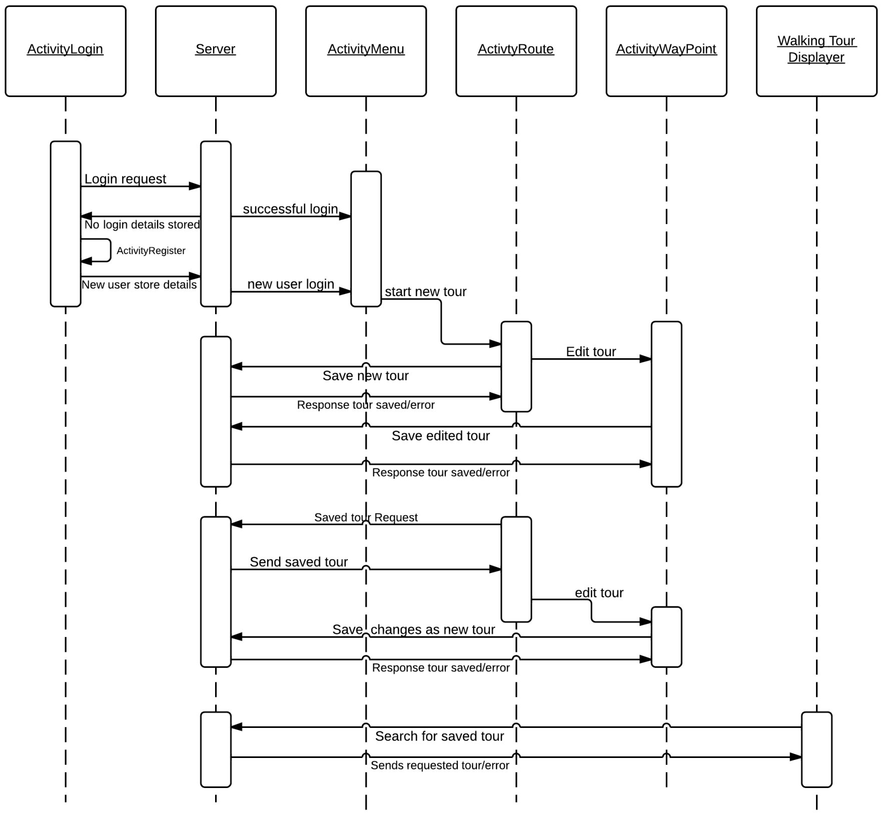
```
package uk.ac.aber.cs22120.fuzzyNinja.pathFinder;

import uk.ac.aber.cs22120.fuzzyNinja.pathFinder.WalkLocation;
import java.util.ArrayList;

/**
* This class contains information associated with a walking tour.
*
*/
public class Walk {
  private String title; // The title of the walking tour
  private String shortDescription; // A short description of the
    walking tour
  private String longDescription; // A long description of the
    walking tour
  private ArrayList<WalkLocation> waypoints; // A list of the
    waypoints along the walking tour

  /** Sets the title of a given walking tour
  * @param title the title of the walking tour
  */
  public void setTitle(String title);

  /** Gets the title of the walking tour
  * @return Returns the title of the walking tour
  */
  public String getTitle();

  /** Sets the short description of the walking tour
  * @param description the short description for the walking tour
  */
  public void setShort(String description);

  /** Gets the short description of the walking tour
  * @return Returns the short description for the walking tour
  */
  public String getShort();

  /** Sets the long description of the walking tour
  * @param description the long description for this walking tour
  */
  public void setLong(String description);

  /** Gets the long description of the walking tour
  * @return Returns the long description for the walking tour
  */
  public String getLong();

  /** Adds a location to the end of the list
  * @param location the location added to the list
  */
  public void addLocation(WalkLocation location);

  /** Gets a location from the queue at the specified index
  * @param index the index from which to get the location
  */
  public WalkLocation getLocation(int index);

  /** Deletes a the location at the specified index
  * @param index at which to delete location
  */
  public void deleteLocation(int index);

  /**Default constructor*/
  public Walk() {}
}
```

WalkLocation:

```java
package uk.ac.aber.cs22120.fuzzyNinja.pathFinder;

import uk.ac.aber.cs22120.fuzzyNinja.pathFinder.GPSLocation;
import java.util.ArrayList;
import java.util.Date;

/** Specifies a waypoint in the walking tour
*
*/
public class WalkLocation {
  private GPSLocation coordinates; // GPS coordinates for the
    waypoint
  private String name; // Name given to the waypoint
  private String description; // Description given to the waypoint
  private ArrayList<Bitmap> photos; // List of photos associated
    with the waypoint
  private Date timestamp; // The date and time the waypoint was
    recorded

  /** Default constructor for this class
  * GPS coordinates and timestamp should be assigned in this method
  */
  public WalkLocation();

  /** Adds a Bitmap object to the list of photos
  * @param photo photo to be added
  */
  public void addToPhotos(Bitmap photo);

  /** Gets a Bitmap object from list of photos
  * @param index index of the required Bitmap object
  * @return Returns the Bitmap object at the specified index
  */
  public Bitmap getFromPhotos(int index);

  /** Deletes an element at the specified index
  * @param index index of the Bitmap to be deleted
  */
  public void removeFromPhotos(int index);
}
```

5 DETAILED DESIGN

5.1 Sequence Diagrams



Above is the sequence diagram showing how the mobile application works and the messages passed through the mobile application, the walking tour displayer (website) and the server. The six objects show the mobile application screens (ActivityLogin , ActivityMenu , ActivtyRoute and the ActivityWayPoint), the server and the walking tour displayer to view the tours. The arrows between them show the messages sent between the objects. This chart is useful for showing the processes of the system, how the server interacts with both the website and mobile application and the ordered way the classes interact with each other.

5.2 Significant Algorithms

In this section we have designed pseudo code outlining each of the major algorithms in our system.

5.2.1 Algorithm 1 - Creating A New Walk

Classes Used:
GPSLocation
Walk
WalkLocation

```
while App is open do
  if User presses "Create new walk screen" then
    if User has connection to internet and User has GPS signal then
      Show Walking Tour Creation screen
    else
      Show error message informing the user that they must be connected
      to create a walking tour
    end if
  end if
end while
```

This algorithm allows the user to create a new walk on the Walking Tour Creator application. It checks whether the user is connected to the internet and has a GPS signal before allowing them to begin the process.

5.2.2 Algorithm 2 - Adding Waypoints To The Walk

Classes Used:
GPSLocation
Walk
WalkLocation

```
while App is open do
  if User is in walking tour creation mode and User has internet and GPS
  connection and User presses "Add Waypoint" then
    Take current GPS Location
    Take details of waypoint
    Save Waypoint information to local database
  end if
end while
```

This algorithm will allow the user to add a new waypoint to the walk. When the user presses the "add waypoint button" it will prompt for details of the waypoint - such as images, description, title etc. and save them to the local database.

5.2.3 Algorithm 3 - Sending The Walk To The Server

Classes Used:
HTTPPostSender

```
if User presses submit button and User has internet and GPS connection then
    Get tour information from local Android database
    Create instance of post structure as defined in Section 5.3.$x$
    for all Waypoints in Database do
        Add waypoint information to POST
    end for
    Send POST to server
    Wait for server response
    if No response message received or Message shows that an error has occurred then
        Show error message to user
    else
        Show success message to user
    end if
end if
```

This algorithm details the process of sending a walk to the server, using a HTTP POST message containing the information for the Walk and each WalkLocation.

5.2.4 Algorithm 4 - Processing The Walk On The Server

```
Receive POST from mobile application
Check structure of POST
if POST is structured correctly then
    Create database entry for tour in tours table
    return Tour ID
    if Tour ID returned then
        for all Waypoints in tour do
            Create database entry for waypoint in tours table, referencing tour ID
        end for
        Return success message to mobile application.
    else
        Return error message to mobile application.
    end if
else
    Return error message to mobile application.
end if
```

This is the algorithm to process the walk, submitted via POST, on the server. It checks that the POST is structured correctly, and then iterates through each WalkLocation and adds it to the database.

5.3 Significant Data Structures

The data structures that will mainly be used on the Android application are ArrayLists. The reason ArrayList has been chosen over an Array is because, in Java an Array is a fixed length data structure whereas an ArrayList is a variable length Collection class. This means the ArrayList will re-size itself when it reaches its capacity. This is required because the size of the arrays we will be using will not be known until after elements have been entered into it.

5.3.1 WalkLocation

An object of the WalkLocation class will hold variables which contain data about the current location the user is at, such as:

- Coordinates
  - The latitude and longitude value of the location.
- Name
  - The name given to the location by the user.
- Description
  - A description of the location given by the user.
- Photos
  - An optional image of the location selected by the user .
- Timestamp
  - The date of the current moment when the user creates the current WalkLocation object.

5.3.2 Walk

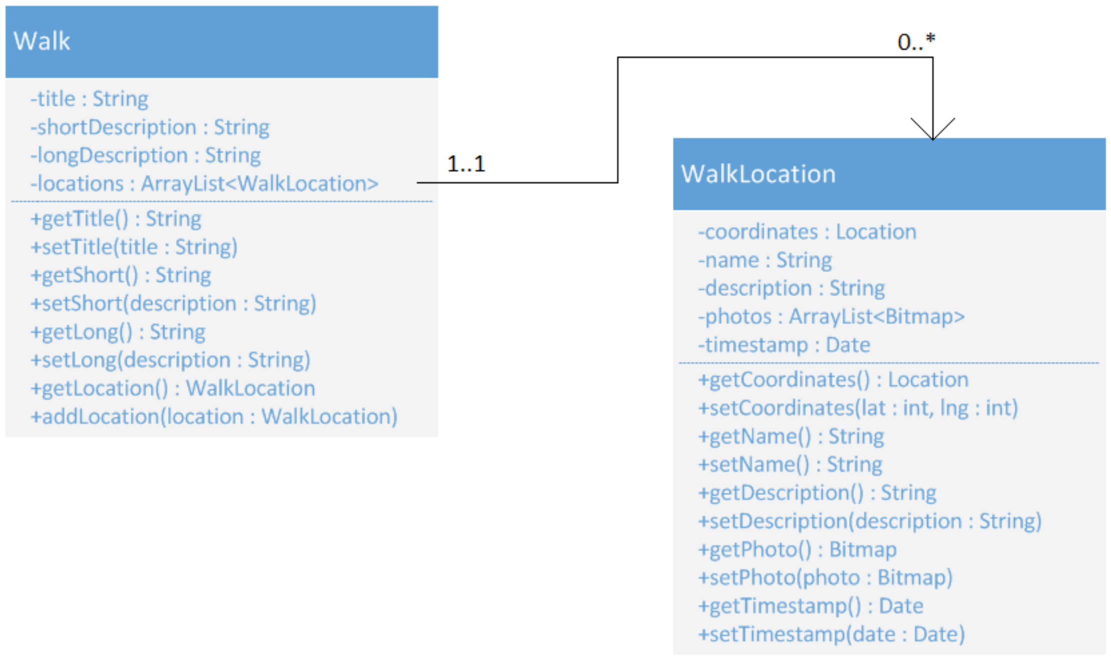
An object of the Walk class will hold variables which contain data about the walk tour the user created such as:

- Title
  - The name given to the walk tour.
- Short Description
  - A short summary of the walk tour.
- Long Description
  - A full description of the walk tour.
- Locations
  - An ArrayList of WalkLocation objects.

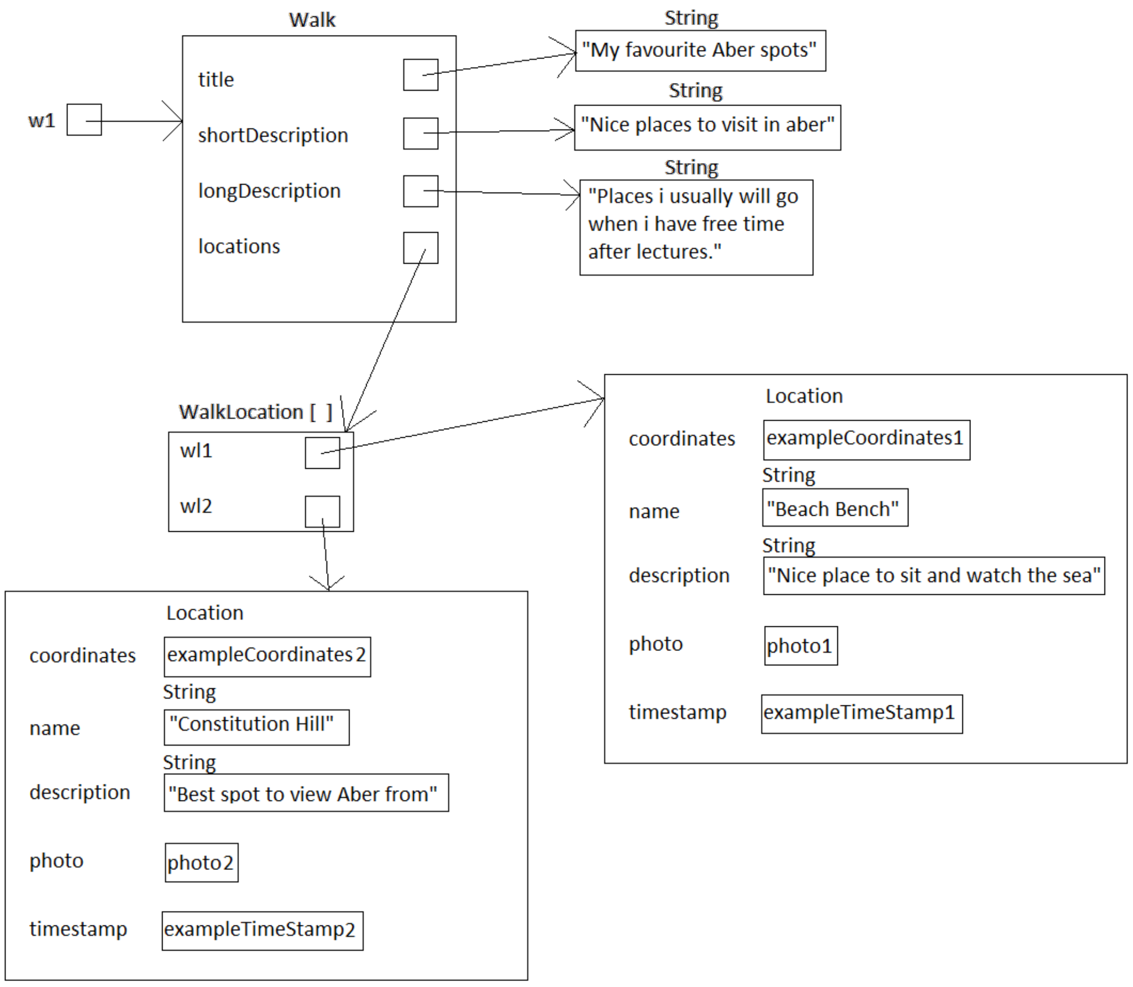
5.3.3 POST Stucture

The application will need a method of sending a walk tour to the server, when the user decides to upload their walk tour. The POST message will be sent from the android application to the PHP server; the server will then populate the PHP object and save to the MYSQL database.

Class Diagram:


Walk
-title : String
-shortDescription : String
-longDescription : String
-locations : ArrayList<WalkLocation>
+getTitle() : String
+setTitle(title : String)
+getShort() : String
+setShort(description : String)
+getLong() : String
+setLong(description : String)
+getLocation() : WalkLocation
+addLocation(location : WalkLocation)
1..1
0..*
WalkLocation
-coordinates : Location
-name : String
-description : String
-photos : ArrayList<Bitmap>
-timestamp : Date
+getCoordinates() : Location
+setCoordinates(lat : int, lng : int)
+getName() : String
+setName() : String
+getDescription() : String
+setDescription(description : String)
+getPhoto() : Bitmap
+setPhoto(photo : Bitmap)
+getTimestamp() : Date
+setTimestamp(date : Date)

Object Diagram:



6 DOCUMENT HISTORY

Ref	Date	Status	Description
0.1	03-12-13	Draft	Initial document creation.
1.0	04-12-13	Draft	First draft complete.
2.0	06-12-13	Release	Reviewed and modified version complete

7 References

8 DOCUMENT HISTORY

Ref	Date	Status	Description
0.1	03-12-13	Draft	Initial document creation.
1.0	04-12-13	Draft	First draft complete.
2.0	06-12-13	Release	Reviewed and modified version complete