Project Group 05
Final report

Authors: bmo; sr11; hac22; wia2; wjl3; njv1
Config Ref: SE_05_TS_01
Date: 11/11/13
Version: 2.0
Status: Release

Department of Computer Science
Aberystwyth University
Aberystwyth, Ceredigion
SY23 3DB
Copyright © Group 05
Aberystwyth University 2013

Table of Contents

1. Introduction

1.1. Purpose of this document

The purpose of this document is to present the final state of the group project and its manual, detail the process of delivering the system and describe all members' contribution and performance.

1.2. Scope

This document contains the maintenance manual for the system, all the revised documents delivered during development including project plan, test specification, design specification as well as a testing section where the tests performed on the system and their results are detailed. The group members' self-reflective reports and critical evaluation are also present. This document should be read by anyone wishing to use the Walking tour system or to modify it.

1.3. Objectives

The aim of this document is to present the maintenance manual of the walking tour system and a relevant FAQ section that can be used to aid anyone wishing to use or modify it. It presents the details of how the project was developed and in what state it is in currently.

2. Management summary

The project as it was submitted, is in an operational state, but with some limitations and some of the functional requirements not completely implemented.

On the android side of the system, the application works well and is able to fulfill most of the requirements. A walk can be created and it can have a name, a description and a long description. The requirement, to have a limitation on the character number, is in place and working. When the user decides to create a tour, the GPS tracking is activated and works. An option we have added, is that the user can define how often their position is being recorded, depending on the speed they are going at. Locations are saved along the route successfully. When the user decides, along the walk, so called key locations can be added. These are special locations that can have a name and description as well as a timestamp, which is to be used further on to determine the distance and length of the walk. After they are added, they appear as pins on the map. The user can add multiple photos (up to 5) per location after tapping the location and selecting the option to add a new photo. A limitation here is that the user can only add photos from the camera app, but not from the gallery, meaning the user will have to take the pictures they want associated with a location. The user can remove photos added to a location by selecting it and then pressing the delete photos button. If the user so chooses, they can remove a key location form the map similarly to the photos, by selecting a location and tapping the remove location button.

In the settings menu the option to cancel the walk is present. That ends the current walk and discards any data on the walk without uploading it to the server. When the user decides to finish their walk, they can do so by selecting the finish walk option and wait for the walk to be uploaded. After they are prompted with success, they can exit the app or create another tour. The upload is achieved with the data from the tour formatted as a JSON string inside a multipart MIME message sent as a HTTP POST request.

On the server side, the database intercepts this message decodes it and stores the data into the appropriate relations and records. The website, having a connection to the database then presents the user with the option to view the walk. The website consists of 4 main pages. The home screen allows for quick access to the most recent walks. Using AJAX the user can simply click on a tour name and it will be displayed instantly. Should the user decide to look at all walks, they can navigate to the list of tours page, and make their selection again. That will take them to a separate dedicated map page, where the map is larger and the focus is only on that one tour. Tours appear on the map as red lines, which connect the key and non-key locations. The key locations are pins, when clicked on display the location's name, description and photos if any are present.

The documentation submitted alongside the project has been revised and is of good quality.

The Project plan has been updated after the initial changes to design were made to more clearly reflect them (removed some features, not specified in the requirements). The test specification has been updated with more precise system tests and more room for fault description. The initially included unit tests have been removed, but still implemented in the actual code. The design specification has been improved and updated with the latest information on the project.

3. Historical account

The Project Management Plan was the first main event over the lifetime of the project. As a team we met up in various locations to discuss the main sections that needed to be completed for the deadline. In order to produce the first plan for the Project Plan, we firstly decided on the roles of each team member considering personal experiences within the fields. The very first hurdle that we tackled was the proposed system designs, figuring out the platforms and high level architecture that we were proposing to use and also the Use case diagrams for the android side and the server. These were mainly the tasks of the web and android teams. As well as the general design ideas, the both teams also had to create wireframes of the proposed GUI giving a reason why they chose these screens/pages. After each member of the team created written documents and image files of the design, the Management team then created the first draft of the document. During the same day, the web team had to change the use case diagram because of a few spelling mistakes with the actions.

The second main event was the Test Specification. All of the individuals in the group worked together for this document, to produce multiple tables and detailed descriptions of the tables. The android team worked along with the testing team in order to create module tests and system tests for the android device. And the web team also worked alongside the testing team to create similar documents for the server side system tests. After all of the final documents were completed, the management team put the Test Specification PDF together ready for submission. After creating the first draft of the document the web and testing teams decided to change the server side test table because we had missed out a lot of important content.

After creating the first two documents the team sat down and decided on how we were going to create the Design Specification. This became the busiest part of the project due to the detail that was needed from each team member. Knowing this, we all split into two groups; web and android team. The first main task within the design specification was the classes and components of the system; we planned to show structure and descriptions of them. One member of the android team designed all of the diagrams such as the component, sequence and UML class, whilst the other filled out the class or interface code with JavaDoc and comments. The web team also had to create a component and sequence diagram for the server side design. The second main task for the Design Specification was the Algorithms and data structures that we were proposing to use within the server and android sides. The two main team members completed this part of the task whilst the rest of the team worked on the existing document feedback. As well as completing the Algorithms and data structures the web and android teams put together sample data for the MIME strings and JSON code and the management team added them into the document as appendices.

The team came together after the Design Specification was completed to work on the submitted documents. In order to efficiently get each document updated the management team handed out roles for each member, from looking through the files comparing them to the feedback to readjusting work to fit in with the new, improved design. The main changes made to the Project Management Plan were the user account details that we had included in the first design. To change this we had to remove the Log in action from the android use-case along with the create account included. We also had to remove a lot of actions within the web side use case diagram, because we had included user preferences within the site too. Not only did the use cases change, the QA manager had to change the wireframes for both of the programs because they had originally included the user preferences.

The next document that the team had to work on was the Test Specification, alongside the prototype demo. The testing team had to work together with the android team for this event in the project, in order to create the test log forms and improve the testing tables.

Whilst getting ready for the demonstration of the prototype, the team had a full week in the Orchard on campus. In order to successfully complete this main event in the project, the team worked together to make sure that we had a prototype to demo. Throughout the week, the team split up into the assigned groups to work on specified tasks that the Project Leader had given. The android team worked on the general code, making sure that it tied in with the functional requirements. The web team also did the same, as well as making sure that the two systems worked together in unison. As we neared the end of the week, the testing team managed to apply the system tests to the application and the website, deciding whether they had passed or failed.

After receiving the feedback to the Design Specification, this became the main focus for the android and management teams. The teams had to work together to change the decomposition description due to the layout of the application changing, from previously stated designs. Many of the class and interface names had changed as well as the UML diagram structure. To complete the task we assigned the main programmer from the android team to go through the code and change what we discussed. Not only did the layout of the application change from original design implications, the web layout also had to change. Whilst the two teams worked on this document, the rest of the group were concentrating on specified sections of the Final Report given to them by the Project Leader. This included personal reports as well as group tasks.

Throughout the whole project, there has been up to eighty hours on average, from each team member. These hours include meetings as well as personal work on the project. In order to complete the milestones that were set these were the kinds of hours that were deemed needed from each member.

4. Final state of the project

The project functions that were defined in the specification were all met with the slight exception of a lack of ability to send photographs to the server that we had built. Due to being unable to send any pictures taken during the walk to the server a section of the functional requirements defined in the project specification were unable to be demonstrated on our elected demo day. The app that we built successfully managed to meet the following functional requirements fully and with no missing features; FR1, our app successfully boots, GPS recording starts and the user is asked if they wish to start a new walk. FR2 our app allowed the user to input a title a long description and a short description for the each walk that they created using our product. FR3 Every 5 seconds the app would automatically generate a basic location that consisted of a time stamp and a pair of GPS co-ordinates. On top of these locations the user could add their own locations with custom names and descriptions, although in the allocation of these values into internal variables and later JSON sent to the server there was a miss-match allowing for the name and description of key locations to get miss-matched with a key-location title being a description, and the title set to null. FR4 while not completely meeting this FR our app managed to utilise the camera using intents and save an image in a custom directory, although these images could only be found after a re-boot of our test device; however Images were not correctly displayed in app. FR5 The user can easily and simply cancel the walk with a single press of a button that prompts a conformation. FR6 The date we sent to the server consisted of a JSON string and, via the use of MIME, the attached images. However the app had issues with sending images to the server, although all locations were sent and displayed correctly, bar the above mentioned miss-match. FR7 The app that we developed was successful in its ability to switch to another app via multitasking, without the loss of user data.

Unfortunately due to time constraints a lot of the testing that we had outlined for the app became un reasonable to attempt. Also because of a lack of accuracy due to GPS signal not being strong enough and our sample rate varying, occasionally the tour's route can appear to zig-zag across roads, and on a rare occasion the tour suggest the user walk through a building

5. Performance of each team member

5.1. William Lea

William Lea took the initiative and started researching the android development side right away. He attended the android development seminar and showed his ability quickly. He also supplied the group with the testing device and devoted a lot of time keeping the rest of the group updated on the current state of the application. William attended all but a few of the meetings with our project manager and had legitimate reasons for missing the ones that he did. He attended all the group meetings dedicated to the android development and all other briefings we had across the first semester. William's contribution towards the project's completion was invaluable.

When addressing his teammates William was always respectful and helpful. Any questions and suggestions he got from the rest of the team was discussed and always considered during development. He proved to be able to work alone when needed and as part of a team. Most of the development was done with him and Harvey Clark working together on the android side. The two of them worked quickly and efficiently, proving they found a common language.

5.2. Harvey Clark

Harvey Clark was passionate about android development and along with William Lea, he went to the induction seminar, did research and worked towards software completion. Harvey proved to be very reliable as he was able to follow all of the project guidelines the project manager, leader and QA gave towards the android application. He showed his ability to deliver high quality level code: well documented, well structured, reliable and robust. Harvey attended most of the meetings with the project manager and had legitimate excuses for missing the ones he did, he attended all the group meetings where the android app was in focus and all other group briefings throughout the first semester. Harvey's contribution towards the project's completion was invaluable.

When teamwork was required, Harvey proved to be a team player, he was honest and to the point with everyone and was ready to help anyone that needed him to. When working with William or alone, he worked quickly and efficiently and produced the results he was expected to.

5.3. William Arslett

William Arslett was experienced in database design and administration, hence why he was in charge of that part of the project. He proved to be responsible as he volunteered to go to the GitHub seminar and was ready to explain to the whole group how to use it properly, as well as he volunteered to keep the meeting minutes. He demonstrated great teamwork and ability to communicate efficiently with the whole group and mainly with his partner Stephen with whom he worked very closely.

As his job was to deliver the database he had to actively communicate with the android team and the website team to ensure a proper connection between the two with the database and he managed to do that providing great results. He delivered code on time and with good functionality and was able to establish the link between both sides of the system. He attended all the group meetings where the main topic was website and database development and most meetings with the project manager. William's contribution towards the project's completion was invaluable.

5.4. Stephen Paul Clasby

Stephen Clasby was given the task to develop the website and connect to the database. He had previous experience and showed passion for working with web-based applications. He worked closely with William Arslett and proved to be a good communicator. During the project's development he was active and produced good results in a timely manner. He had to develop the website so that it would be usable on any device and produce the same desired effect. Implementation of the mapping API was also his responsibility.

Stephen attended most meetings with the project manager and group meetings dedicated to the website development. He was mostly absent from the group briefings, but was available online to communicate from a distance.

5.5. Neil Vicker

Neil Vicker was given the task to develop and implement the whitebox and blackbox testing of the system. He worked closely with Benjamin O'Donovan, the group's Quality Assurance manager to develop the tests and then to later run them on the system. He was to also research and produce a report about the multiple mapping APIs that we were considering to use. He presented his findings in front of the group and the group decided to choose the Leaflet mapping API. Neil was quick and efficient and completed all of his assigned tasks on time. He proved to be a good communicator by working with both the android and web side of the project to develop the appropriate system level tests.

Neil attended most meetings with the project manager and most meetings the group help do discuss the project in general.

5.6. Benjamin O'Donovan

Benjamin was assigned as the group's Quality Assurance manager and deputy leader. He showed passion and determination while working on the project. As QA he controlled the process of development making sure all the requirements are met and that the quality of code that was delivered was at the level expected. He proved to be a great communicator as his job involved constant contact with all group members. Benjamin also assisted Neil with the system testing. He developed tests alongside him, making sure they answer the standards. He performed and documented the testing done on the application and website.

When working with the rest of the group he addressed everyone with respect and was ready to help out anyone that asked for help. Benjamin attended most of the meetings the group held with the project manager, as well as most of the meetings the android and web teams held on their own. He attended all project briefings.

5.7. Sevastiyan Raychev

Sevastiyan was assigned to the task of project leader. His tasks included organizing meetings for the group, discussing any issues the group faced during development and answering any questions that the group had regarding the project. As group leader he had to keep in constant contact with all members of the group and to make sure every one of them had a task to work on and that that task was to be completed in a timely manner and that it would be what was expected. He was to develop a Gantt chart detailing the milestones the group was facing, who was to tackle them and how much time they had to finish. He also dealt with the majority of the documentation with help from the group QA.

Sevastiyan took into account everyone's opinion and made an effort to hear everyone out and implement their ideas into the project. Sevastiyan attended all the meetings with the project manager and most of the meetings for the android and web sides and all of the project briefings.

6. Critical evaluation of the team and the project

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

The team performed well as a whole. The biggest issue in the beginning was communication. That led to some uncertainties and miscommunication, but soon after we started working, we managed to solve the issue effectively. The biggest issue we faced really was the fact that out android developers were not able to work on campus, which meant that in order to compile and put the app on the device, someone would have to go down to town and back, which didn't waste as much time as we thought it would, because we just worked form a distance, but it could have been avoided. The project faced the issue of requirement creep. In the beginning we thought we would have to add a user system and we planned with that in mind, but later we decided to drop the feature for now as it would have slowed us down. The task assignment could have been different, to include some members that were left with less work at some points while others were juggling multiple tasks, but sometimes just the skills and knowledge of the person decided what tasks they would have to take on. To improve this situation the group took on pair programming to make sure that there was not only one person that knew everything about that one task. There is still room to improve on this though.

6.2. How could the project have been improved?

The project could have been improved by making sure all the features were delivered on time and by extension that all the functionality was in place at the deadline. From the management side, one thing that could be improved would be the milestones set. We had set milestones but they were large and not as clear as possible. If they were split into smaller increments it would have made development a little faster.

6.3. What were the most important lessons learned about software projects and about working in teams?

The most important things learned were that communication is key in any group task, but also that there need to be measures taken so that in the situation where communication is impossible or bad, the results of that do not lead to catastrophic consequences.

The group learned what the consequences of requirements creep are and how important it is to deal with them as soon as possible.

The importance of time management was very important. When a deadline was close the group was ready to present when the time of the members was used appropriately and tasks were spread out throughout and were clear and exact.

7. Appendices

A. Project Test Report

Group 05
Project Test Report

Authors: bmo; sr11; njv1
Config Ref: SE_05_TS_01
Date: 14/11/13
Version: 1.0
Status: Release

Aberystwyth University
Aberystwyth, Ceredigion
SY23 3DB
Copyright © Group 05
Aberystwyth University 2013

Test Log No: 001	Group: 05	Testers(s): bmo, njv1
Date: 28/01/2014	Tagged version ID: 0.1	

Test ID	Pass / Fail	Fail description	CCF / issue #
SE-F-001	Pass		
SE-F-001.1	Pass		
SE-F-002	Pass		
SE-F-003	Pass		
SE-F-004	Pass		
SE-F-005	Pass		
SE-F-006	Pass		
SE-F-007	Fail	The camera works and takes an image but it doesn't add to the location pop-up.	#3
SE-F-008	Fail	The camera works and takes an image but it doesn't add to the location pop-up.	#3
SE-F-009	Fail	It is in the Model, but not in the App	#6
SE-F-010	Pass		
SE-F-011	Pass		
SE-F-012	Pass		
SE-F-013	Pass		
SE-F-014	Pass		
SE-F-015	Pass		
SE-F-016	Pass		
SE-F-017	Pass		
SE-F-018	Fail	Multiple thumbnails are showing with invalid images inside	#7
SE-F-019	Pass		
SE-F-020	Pass		
SE-F-021	Pass		
SE-F-022	Fail	The image files are being saved as Null	#5

Test Log No: 002	Group: 05	Testers(s): bmo, srr11
Date: 28/01/2014	Tagged version ID: 0.2	

Test ID	Pass / Fail	Fail description	CCF / issue #
SE-F-001	Pass		
SE-F-001.1	Pass		
SE-F-002	Pass		
SE-F-003	Pass		
SE-F-004	Pass		
SE-F-005	Pass		
SE-F-006	Fail	Not completed yet	#4
SE-F-007	Fail	Not completed yet	#4
SE-F-008	Fail	Not completed yet	#4
SE-F-009	Fail	Not completed yet	#4
SE-F-010	Pass		
SE-F-011	Pass		
SE-F-012	Pass		
SE-F-013	Pass		
SE-F-014	Pass		
SE-F-015	Pass		
SE-F-016	Pass		
SE-F-017	Pass		
SE-F-018	Fail	Not completed yet	#4
SE-F-019	Pass		
SE-F-020	Pass		
SE-F-021	Pass		
SE-F-022	Fail	The image files are being saved as Null	#5

Test Log No: 003	Group: 05	Testers(s): bmo, srr11
Date: 13/02/2014	Tagged version ID: 0.3	

Test ID	Pass / Fail	Fail description	CCF / issue #
SE-F-001	Pass		
SE-F-001.1	Pass		
SE-F-002	Pass		
SE-F-003	Pass		
SE-F-004	Pass		
SE-F-005	Pass		
SE-F-006	Pass		
SE-F-007	Fail	Cannot access the gallery	#8
SE-F-008	Pass		
SE-F-009	Fail	Cannot edit the information	#9
SE-F-010	Pass		
SE-F-011	Pass		
SE-F-012	Pass		
SE-F-013	Pass		
SE-F-014	Pass		
SE-F-015	Pass		
SE-F-016	Pass		
SE-F-017	Pass		
SE-F-018	Pass		
SE-F-019	Pass		
SE-F-020	Pass		
SE-F-021	Pass		
SE-F-022	Pass		

B. Project Maintenance manual

WTC app maintenance guide

Program Description

The Walking Tour Creator app, WTC for short, is used to create walking tours of towns, cities or smaller areas of interest. A user will enter the details of the area they are making a tour for. These details include a single word name for the area they are touring, a short description for the area they are touring and a long description for the area they are touring. Whilst making the tour, the user will be able to add key points of interest to the tour. As they do this, they will be prompted for the name of this point of interest and a description for the point of interest. The user is able to add photos to the points of interest and also remove those photos. The user is also able to remove key points of interest they have added. The user is able to cancel a walking tour they are in the process of making, this will erase all the locations they have added to the tour. Finally the user is able to save their walking tour to an online database and view the tour they made.

Program Structure

The app is separated into three activities:

- MainActivity - this is the launcher activity for the app, when the app first starts this is the screen the user is presented with.
- WalkActivity - this is the main part of the app, this where the user builds their tour. From here they can:
    - Add / Remove points of interest.
    - Add / Remove photos to / from a point of interest.
    - Cancel their walk.
    - Save their walk for later viewing.
    - They are also able to adjust the sample rate for non-key locations to save on how much data they transmit to the server.
- AboutActivity - this is just a credits screen, it just says who is on the team, the app name and version number.

There are also some supporting UI classes for dialogs.

- NewWalkFragment - sets the details of the walk and starts WalkActivity with that information.
- EndWalkFragment - ends the walk if the user selects ‘OK’.
- FinishWalkFragment - does the same as EndWalkFragment but saves the walk to the server before quitting.
- NoNetworkFragment - alerts the user that there is no network available.
- LocationDetailsFragment - gets the details for a key location from the user.

Main Data Areas

The main data area of the app is under the package **com.wtc.grp5.model.** This package contains the classes which contain the data for a tour.

WTCTour - This class represents the walking tour the user will create.

Field type	**Field name**	**Description**
String	tourName	The name of this tour.
String	shortDesc	The short description of this.
String	longDesc	The long description of this tour.
LinkedList<WTCLocation>	locations	The list of locations in the tour (includes key locations by inheritance).

WTCLocation - This class represents non-key locations in the tour (used for tracing lines between points on the website map)

Field type	**Field name**	**Description**
double	longitude	The longitude of this location.
double	latitude	The latitude of the location.
Calendar	oldTime	The original time stamp in a date/time format.
long	timeStamp	The timestamp for this location as minutes since the start of the walk.

WTCKeyLocation - This class represents key points of interest in the tour.

Field type	Field name	Description
List<String>	photos	The list of file paths for photos the user adds to this key location.
String	locName	The name of this key location.
String	locDesc	The description of this key location.

Algorithms

Saving to the server - main thread

Check network connectivity

If connected

Set up async task to save to server with the server URL and Tour data.

Execute async task.

Else tell the user they have no network connectivity.

Saving to the server - async task

Adjust the timestamps of the locations in the tour minutes since the tour start.

Set up the HTTP Client.

Set up the POST message with the URL from WalKActivity.

Set up the Multi-part MIME.

Add Name/Value pair for the JSON data of the tour.

For each key location **in** the tour

For each photo file path **in** key location photo list.

Add photo file path to Name/Value pair list.

For each Name/Value pair

Add Name/Value pair to the Multi-part MIME.

Send the POST data to the server.

Adding key locations to the tour

User presses the 'Add Key Location' button.

User is prompted for the name and description of the key location they're making.

User enters the details in presses 'OK'.

A map marker is added to map at the user's current location.

The marker title is given the name of the location the user entered.

The marker description is given the description of the location user entered.

The user's current longitude and latitude are put in a key location object and added to the tour list.

Removing key locations from the tour

User selects the marker for the key location.

User presses the 'Remove Key Location' button.

For each location **in** the tour

If location's longitude and latitude match marker's longitude and latitude.

Remove location from tour.

Adding a photo to a location

User selects the marker for the location they want to add a photo to.

User presses 'Add Photo' button.

Image file is created.

Camera app is started.

User takes a photo.

Image saved to the image file.

For each Location **in** Tour

If currently selected marker's longitude and latitude match Location's longitude and latitude

add photo file path to Location.

Files

The app uses a serialised data file, which is made via the TourSave class, to store the tour when the user navigates away from WalkActivity or even the entire app. The app also produces JPEG files for the pictures the user takes. The facilities for saving the pictures are provided by the Android OS, however, the naming for the picture files is handled by the app and the names of the pictures is based on the date and time it was taken to ensure name uniqueness.

Physical Limitations

The physical limitations of this program are the need for network communication (mobile data Wi-Fi) and GPS signal. The GPS signal, obviously is for recording the user's location into the tour and the network is for saving the tour to the server. Without either of these prerequisites the app can't function properly.

Suggestions for Improvements

The first improvement to make would be to add the ability to save a walk temporarily when the user selects 'Finish Walk' and there is no network. Another improvement that is needed is the ability for the user to be able to see the photos they have taken for the locations in the walk. The user should also be able to add a photo to locations from their device's photo gallery.

Rebuilding and Testing

To build/rebuild and test this app you will need:

- The Eclipse IDE with the ADT plugin.
- A copy of the Google Play Services Library project to put in the same workspace as the app project.
- The Google Play Services SDK.
- An Android device with at least Android 4 installed on it.

For actual system testing you need to install the app on your Android device. To do this just plug it into the computer you developing on and press "Build/Run" in Eclipse. You may be prompted for a device to build the app to. Just select your phone and press "OK". The app should start up on its own.

WTC web side maintenance guide

Program Description

The purpose of the server side web application is to facilitate the reception of data from the phone, manage the database and the way the data is manipulated and to render the data stored in the database in a stylish, user-friendly presentation.

Reception of Data

The reception of data is facilitated using a PHP upload script. The data is sent from the phone using an HTTP POST request. The PHP file reads the POST request and renders it in the database using the model view controller.

We were unable to get the phone to upload images successfully, however, the upload script should, in theory upload images if they are included in the post request. We were unable to get the thumb nailing working so the upload script will only upload the full resolution image without any processing.

Manipulation of Data

The data is manipulated using a model-view-controller of the database. The model-view controller serves three purposes. Firstly, it implements methods for reading an objective representation of the data sent to the database by the phone and inserting it into the database. Secondly, it implements methods for outputting data on the database as JSON so that the mapping AJAX script can read it. Finally it includes methods for getting the data from the database in an objective format.

The presentation of Data

The front end of the website is built in HTML/XHTML and CSS. We use PHP to populate the pages with data from the model-view-controller.

The home page displays tours in a map, which are loaded dynamically using AJAX. The re is also a page, which lists the tours with their short description and links to a map page, which will display a given tour on a map with place markers and thumbnails in addition to the title, and long description.

The map facilitates the images and thumbnails, despite the fact that the phone does not upload images and the upload script will not process thumbnails. If an image is stored in the correct location, the map will display the full resolution image scaled down to a thumbnail which links to itself in full location.

Program Structure

Web site

The website is comprised of four PHP web pages, index.php, about.php, map.php and list.php. These pages all include the file src/template.php, which contains the common design template for each page.

The style of the website is configured by the CSS script src/style.css. Each page contains links to the home page, the about page and the list of tours page. The list of tours page contains a list of links to map.php with a given parameter identifying the tour.

Model View Controller

The model view controller is made up of two files, src/connection.php and src/tour.php. The connection file contains the dbConnection class, which defines methods for interacting with the database. Common tasks are represented as methods, which execute SQL queries.

The tour file contains the class Tour which defines a data structure for the tour and methods for interacting with that data. It also contains the classes Location and Place for defining the data structure of those tour attributes.

The Tour object also includes a function which puts all of the data into an associative array and then outputs it as a JSON string.

Leaflet

The map is facilitated by the leaflet API that renders data loaded by an AJAX script. We implement leaflet using the file leafletembed.js which contains two functions. Firstly, initmap() which initializes the map and sets default attributes. Secondly, loadTour(id) which loads a tour with a given id using an AJAX request and renders it on the initialized map.

The AJAX request queries the file plots.php which loads returns a JSON representation of the locations of the tour with the given ID.

Uploading

Upload.php and the model view controller facilitate the uploading process. Upload.php reads the JSON string and turns it into an object. That object is then parsed to the model view controller to be rendered in the database using the method insertTour($data).

insertTour contains SQL queries for inserting the data into the database. It also contains code which should loop through all of the photographs (which the phone does not upload) and upload them one by one to the relative directory.

Data Design

Table 1: Database Tables

Table Name	Table Description
tours	A list of walks/ tours, which the program will display.
locations	A list of geographical locations referencing a record in the tour table, describing the route of the tour as a sequence of locations.
pointsOfInt	A list of points of interest along tours referencing a location.

photos	A list of photographs referencing a point of interest

Table 2: List of Walks Table

Field Name	Field Description	Field Data Format
id	Primary Key (auto increment)	integer
title	Title of the tour	text
shortDesc	A short description of the tour (<100 characters)	text
longDesc	A detailed description of the tour. (<1000)	text
hours	The number of hours the walk will take	float
distance	The total distance of the tour in kilometers	float

Table 3: Location Table

Field Name	Field Description	Field Data Format
id	Primary Key (auto increment)	integer
walkID	Foreign key, referencing the id field of the tour that the location is associated with	integer
latitude	The latitude map reference for the location	float
longitude	A detailed description of the tour.	text
timestamp	The time in hours from the beginning of the tour	float

Table 4: Place description table

Field Name	Field Description	Field Data Format
id	Primary Key (auto increment)	integer
locationID	Foreign key, referencing the location that the point of interest is referencing	integer
description	The description of this point of interest. (<500 characters)	text

Table 5: Photo Usage Table

Field Name	Field Description	Field Data Format
id	Primary Key (auto increment)	integer
placeID	Foreign key, referencing the point of interest that the image is attached to	integer
photoName	The name of the jpg file for the photo (without ".jpg" suffix)	text

Suggestions for future improvements

The image upload from the phone needs to be fixed so that we can use images and ideally there should be some server side image processing to create thumbnails so that we do not waste bandwidth by displaying large resolution images.

I would strongly suggest implementing a user system so that tours can be traced back to a user who is accountable for what has been uploaded. The system has the potential for social network integration and integration with other services.

Although the interface is sufficient, it could receive some aesthetic upgrades with more consideration to HCI than in the first build.

C. Personal Reflective Reports

a. William Lea

Overall I would say that our group progressed and worked well together, in spite of some communication breakdowns. We were able to meet the deadlines for the deliverables, even if sometimes we did other tasks better than others. For example, the first prototype was severely lacking and we weren't able to demonstrate any real capabilities of our system.

At the end of the project around coding week there were multiple slip ups with the Android app. They weren't really massive issues but they were exacerbated by the project setup for the app. The app was only runnable on my home computer. Not thinking ahead, I elected to add a more involved UI using google maps. Since the university machines didn't have the Google Play Services SDK, the app couldn't be compiled and run/tested at the university with the rest of my team.

At the start of the project during the design phase we started off quite well with organising the meetings and planning things. However things slowed down a fair amount, there were was a week or two where nothing seemed to have progressed but we eventually got the ball rolling again.

The Android team (Harvey and myself) had regular Google Hangouts to work on the design of the app. We worked on the Class diagrams, Sequence diagrams, etc. We ended up redesigning it several times due to few issues with Android.

By the time we were meant to start prototyping things were moving slowly. I felt this was due to other assignments had to do around this time. Though I am unsure of what the web team were doing as far as prototyping goes, I spent around an hour every other night building the prototype.

When it came to finally showing the prototype, I felt the prototype was lacking. As far test info goes, the only thing that worked for testing was the server communication. The other buttons just output a message to the user about the action they performed. For example, if they pressed the add location button it would output "Location added".

What I would do better if we did another group project.

As far as doing another group project goes, the first thing I would do is bear in mind my development environment. When building the app, we had the idea to put a map on the app so the user can get a preview of what their walk would look like. This meant we needed addition APIs which the university did not have! So, because of this I had to do the Android stuff on my home computer, which would have been fine if it was a laptop. So next time I think it would be better to first check what I'm working with first before picking features. Another thing that the group as a whole could do better is communication. We had several communication breakdowns that lead work not being done. However, as far as meetings go we did alright, we would often meet on Wednesdays to discuss what had been achieved that before we went into our meetings with our project manager. In addition to these meetings, our teams had extra meetings to discuss how they were approaching their tasks. I.e. the web team discussed the database, mapping API and the frontend of the site. The Android team would, for example, discuss things like how to represent the tour in Java, how to collect GPS locations and the UI of the app.

b. Harvey Clark

Our aim was to build an app and website that worked together to create and display user created tours of local areas. We each chose our roles based on our skills I elected to be on the android side of the development due to my limited knowledge website development, and my enjoyment for programming. The project was very enjoyable and I learnt a lot about software development when working with a customer and a team, skills that I was lacking before starting the project. I had some over ambitious features that were not part of the specification that I wanted to incorporate into the app at the beginning of our project but quickly realised this wasn't realistic. I was asked to be on the back end of the android app and to write the model I elected to write the code that generated the JSON for the server. I was also the person in the group that helped solve issues on the git; assisting the members of the team that didn't understand git, understand it. As a group we synergised well each of our strengths making up for others weaknesses, for example Will Lea, another Android developer, has strength in structuring his code, whereas my skill set is more geared to bug fixes and finding solutions.

During integration and testing week I had some issues with the plumbing in my house which restricted me to working from home while repairs were carried out, unfortunately this did delay my getting the JSON delivery to the specified standard, but I quickly worked around it and had it corrected and ready the next morning, I also assisted in finding solutions to the bugs we were encountering quickly and effectively.

Unfortunately our app shipped with a slight bug whereas the app wasn't sending the images recorded to the server, and some of the fields were miss-matched, this was caused to the time constraints as a result of my unforeseeable situation that forced me to work remotely. However these are easily fixed and we have a version of the app where the photographs work.

c. William Arslett

I have enjoyed working on this project and I am satisfied with my own personal contribution. My role in the team was to take the lead in the server side aspects including the transmission of data, the management of the database and the rendering of the data in the front end.

This project has given me the opportunity to exercise and demonstrate my technical skills as a web developer, such as database design, object oriented PHP programming and AJAX programming, but also my people skills, team working skills and communication skills.

We invested many hours in the design stage of the web side of the project. In many respects this was extremely useful as it gave Stephen Clasby and myself a definitive direction when implementing the project however I recognize that there were areas that could have been designed in more detail before implementation.

I believe that Stephen and I worked well together as a team, particularly during the implementation stage where we worked very closely. I think it was useful that we were able to identify our strengths and weaknesses early on in the project and we were able to distribute work between us appropriately.

I would have liked to start testing earlier during coding week and I think it could be said that we slowed down towards the second half of the week. If we were to do coding week again I think that we would have had to be stricter with scheduling our milestones.

If I were to do the project again I would provide more technical detail in the design and I would have prototyped our designs more before implementation as this would have really helped us at implementation stage.

d. Stephen Paul Clasby

The project to design and create a Tour creation app and website to complement; I enjoyed the project. The roles were chosen based on the strengths and weaknesses of each person, I chose to be on the web side of the project as I was more suited to that task rather than for the android. I created the design and layout based around the group's feedback after each modification; I used PHP to generate the webpages based on what was in that pages PHP file after it generated the template. I did encounter a few troubles during this phase of development with the Leaflet API, which required the layout to be modified. I did have some problems while I was refactoring "connection.php" which I had to rewrite some of the methods, so they didn't depend too much on what was above. The group as a whole worked well together, there was a bit of a delay between the two teams in the group which frustrated me, but was only a minor setback, and I used the time to tidy up the website and the code, while we waited for the android app to catch up. The code in the template and the src files could be tidier, commented and overall improved but due to the time constraint I had to fit everything into the time scale required.

The front page needed to have a map with a list that was generated from the database and displayed neatly; I created a few more areas on the webpage and added another CSS entry for the front page map to display correctly. The PHP code needed loop through the database and displayed the entries as hyperlinks which linked the ID into the map thus loading all the relevant information from the database. This code I feel could be better, but there wasn't enough time, so I created a simple script to do what was required as fast as possible.

Living twenty miles away from the university hindered me somewhat as I couldn't attend all the group meetings due to bus time constraints; so there was some delay on what was discussed and what tasks were handed to me. The group's chemistry as a whole was well; there wasn't any clashes of personality between members of the group or myself, the general attitude of the group was well and focused on the task at hand.

e. Neil Vicker

My main role within the group project was testing but as this stage of the project isn't until late in the projects timeline, I also helped out with quality assurance and assisting the web team. In my role as tester I first had to develop the tests we would use later when we had a prototype to test on. I mainly focused on writing the server side tests because I had been working in close correlation with the web team meaning I had a greater understanding of what needed to be tested, I then reviewed and altered the android and server side system tests later on. As my main role was testing I also assisted in producing the test specification. After we had a solid prototype to test on I could start testing the app, to do this I made some test walks around the university campus and recorded my findings in test log forms with a pass or fail output. With the any tests that failed I made an issue on Git which included why the test failed/what needed to be changed and I also tagged the group members that should resolve the issues raised. All of the issues raised were dealt with quickly, I feel that this is a great and efficient way to raise an issues when testing as it means that issues can be resolved by anyone who isn't present when I am doing the testing and I am also notified when the issue is closed.

During the design phase of the project we had to decide on a mapping API we wanted to use on the website to view the user created torus, we knew that most groups were just going to use the Google maps API but we decided to do some more research and find out what other API's were out there. We managed to narrow the choices down to; Google, Bing, Polymaps and Leaflet. I then produced a document which compared the features of the API's as well as the browser support the provided. I came to the conclusion that Leaflet was the best choice as it had good browser support as well as great CSS3 implementation as opposed to Polymaps which did have some interesting features such as vector tiles using SVG file format but this meant that it didn't support anything below IE9. Once we agreed that Leaflet was going to be the mapping API we were going to use I produced some diagrams which showed what the leaflet interface was going to look like so the group could see what needed to be implemented.

I also helped out with some of the quality assurance, going back and looking at the comments left by Bernie on the design specification, test specification and the project plan and then in turn altering the relevant sections within the documents. Some of the sections included diagrams which I didn't have access to, so I notified the group members which it concerned to change them based on the information given.

I feel that the group worked well to produce a quality final product, there were some delays during the work and integration week with the android app, but they were resolved in time. In my opinion the correct group leader was chosen from the group, he delegated roles well and kept the group organised as well as making sure that the group met regularly to keep the project on track. My contribution to the project was more a supportive role as I wasn't part of the android or web team so during the parts of the project when I wasn't too busy working on something I was helping produce documents and diagrams for those who were busy elsewhere. Overall the group worked well as a unit with little to no personality conflicts.

f. Benjamin O'Donovan

At first as QA manager, I felt that I had a tough role in the group because of the amount of QA documents I had to read over. But after the project really got up to steam, I realised that other members had been given tougher roles. My duty was to make sure that every piece of work that any team member was doing was in line with the requirements in the specifications. I was given eleven documents to read and familiarise myself with, in case any team member had a question that involved the specification.

Alongside being the QA manager, I also had a part in the testing team. I got along with my other team mate and I felt that we managed to complete all of the milestones set under the testing specification. We managed to create system level tests that the android and web teams could read and create systems that coincided with.

I came across a few problems during this project, including writing the functional requirements table. Although I found a couple of tasks challenging, I managed to create work good enough to add into the four main documents the team had to submit. I feel as though I have done a good enough job, considering it is my first QA role.

Even though I have done testing in personal assignments, I felt as if this was a different and more challenging because of the pressure of having other team members relying on me. I enjoyed the testing more than being the QA manager, because it felt like I was more involved with the 'nitty gritty' part of the project. Originally I had a couple of issues with the testing, but with help from the web and android team I managed to create tests that coincided with the work that each team was doing, as well as having help from the other testing team member.

Although I live fifteen minutes from the University, I didn't manage to make all of the tutorials set by the University and the group meetings. I was ill for a few, but generally missed more due to oversleeping or poor time management. However, I felt as if I contributed enough toward the project to make up for those absences.

The group took a few weeks to gel completely. It was fairly difficult at the start to put across ideas and actions, but after a while the Group leader and whoever else had something to say put their point across. As a group we managed to get the work done, even if it took another member of the group to have a word with a particular group member. This was good though, because I feel that without authority over members then a few of the tasks may not have been finished up to a considerable standard.

There were several cases where the Group Leader would assign a date and time to meet and only a few members would turn up, including me. I felt that this hindered us a few times due to deadline dates and heaping more pressure upon other members in the group. However, thinking about the workload spread across each person, I feel that it was even and fair.

Overall we had no major problems with single members or teams. We got along well as a team, and gelled reasonably quickly.

g. Sevastiyan Raychev

In the beginning we were discussing roles in the team. When talking about the different members' abilities, I realized that the team was well balanced and I wanted to help as much as I can. I had experience in project management and I thought I could do ok as project leader. When I saw the amount of work that needed to be done I was ready to do it, so when project leadership came up, I said I would like to take that work on.

The first task we were faced with was the project plan. To start work on that I handed out work to everyone and they worked alongside the QA to produce results. Here was the point I realized that I had let the communication between the team fall through. There was miscommunication and some chaos for about a week. Our project manager felt like there wasn't enough work being done and I was concerned by this. As it turned out, the work was being done, but no one was saying they had done it. I decided that the best thing to do is just re-establish the connection between everyone. After that same week's project manager meeting I talked to everyone and in order to avoid further confusion I asked the members of the web team and the android team to meet separately once a week, but I also organized for the whole group to have briefings twice a week as well. We ended up having enough meetings so that work was done in the groups incrementally, but also when the whole group met, we could discuss the linking between the two parts of the system.

When I was creating the Gantt chart I wanted to make sure that there was no one without any work at any point, but that couldn't always happen, so I attempted to think ahead and assign non-emergent tasks to people who had already done their current work.

Along development an issue was Git. While we were supposed to upload all work there, some of the group had not gotten into a habit of doing that, so I asked William Arslett, who was our group's representative at the Git introduction lecture to explain how the process works, on home PCs and on the university machines for those who were unsure. Soon after everyone was uploading to the repository everything they did.

When I arranged meetings most of the time all members would come, but sometimes it seemed as if the rest of the group had to spend extra time explaining to those absent, what decisions were made and what tasks were handed out to them. I confronted them about this and I was met with understanding, but it took a few more warnings for them to realize what was at stake.

Whatever the cause of their absence though, when there was a task to be performed, everyone gave it their best and the results were good. Whenever there was someone who had not done their work, I took the approach of talking with them and working alongside them or putting someone else with them to work together. I noticed that that approach got the most work done, when two or three people were working on one task at the same time.

Because of this, I realized that the Project plan I had developed myself was not as good as it could be and could be improved, that's why I asked Benjamin to work alongside me on the documentation, making sure everything was in order. This produced better results.

Overall the project went ok, but what I could have done better overall would be time management. I trusted my teammates with their work in their own time and regretted it not once. I should have been stricter with deadlines in general. If I could change something, that would be the fact that I relied on some members' ability to track their own work too much.

D. Revised project plan

Project Group 05
Project Management Plan

Authors: bmo; sr11; hac22; wia2; wjl3; njv1
Config Ref: SE_05_TS_01
Date: 11/11/13
Version: 2.0
Status: Release

Department of Computer Science
Aberystwyth University
Aberystwyth, Ceredigion
SY23 3DB
Copyright © Group 05
Aberystwyth University 2013

Contents

1. Introduction

1.1. Purpose of this document

The purpose of this document is to show how the project group has decided to carry out the client's requirement specification for a walking tour application as a set of objectives and milestones.

1.2. Scope

This document contains the details of the group project but does not go into detail about design, testing or maintenance. It contains the group's choice of platforms and high level architecture as well as justification for the choices made. The Use-Case diagram is included showing how different users will interact with the different parts of the system as well as screenshots and descriptions of the GUI. The proposed Gantt chart is part of this document, detailing what the group will be working on in the process of development and in what timeframes. [1] This document was created after familiarization with the Project Management Standards. [2]

1.3. Objectives

The objectives of this document are as follows:

1.3.1. To describe the overview of the proposed system;

1.3.2. To describe how the main components of the system will interact with each other;

1.3.3. To present the base user interface and describe how the user will interact with it;

1.3.4. To provide a list of the project milestones;

1.3.5. To provide a list of all tasks that need to be completed on the project and their anticipated timeframe in the form of a Gantt chart;

1.3.6. To list possible issues the team might encounter during development in the form of risk analysis.

2. Overview

The proposed system is a walking tour android application that allows people to "record" walks they make through GPS and add information and pictures at points they find something interesting. The walks will be available to view on a website for everyone.

2.1. Platforms and high level architecture

2.1.1. Android

The platform has been specified by the client in the project guideline.

2.1.2. IDE

We are using the Eclipse IDE with the ADT plugin because that is the IDE the team is most familiar with. We took into account Android Studio but reached the conclusion that Eclipse was more stable provided a better user interface.

2.1.3. Android Mapping API

We are using Google Maps for the Android mapping API because it comes as part of the Android SKD and gives the user a full screen map to view, which the programmer can overlay with their own controls. This extends the FR8 requirement in the requirement specification document [1] to allow for the user to view the map on their android device.

2.1.4. PHP

The research showed that PHP is capable of processing easily JSON files, which we will be using, also it is available on most servers and is currently taught in one of the second year modules.

2.1.5. JSON

JSON would be the best data set to use for sending the information about the recorded walk to the server. This is largely due to it being significantly lighter in weight than XML and how easy it is to process in PHP.

2.1.6. Web side mapping API

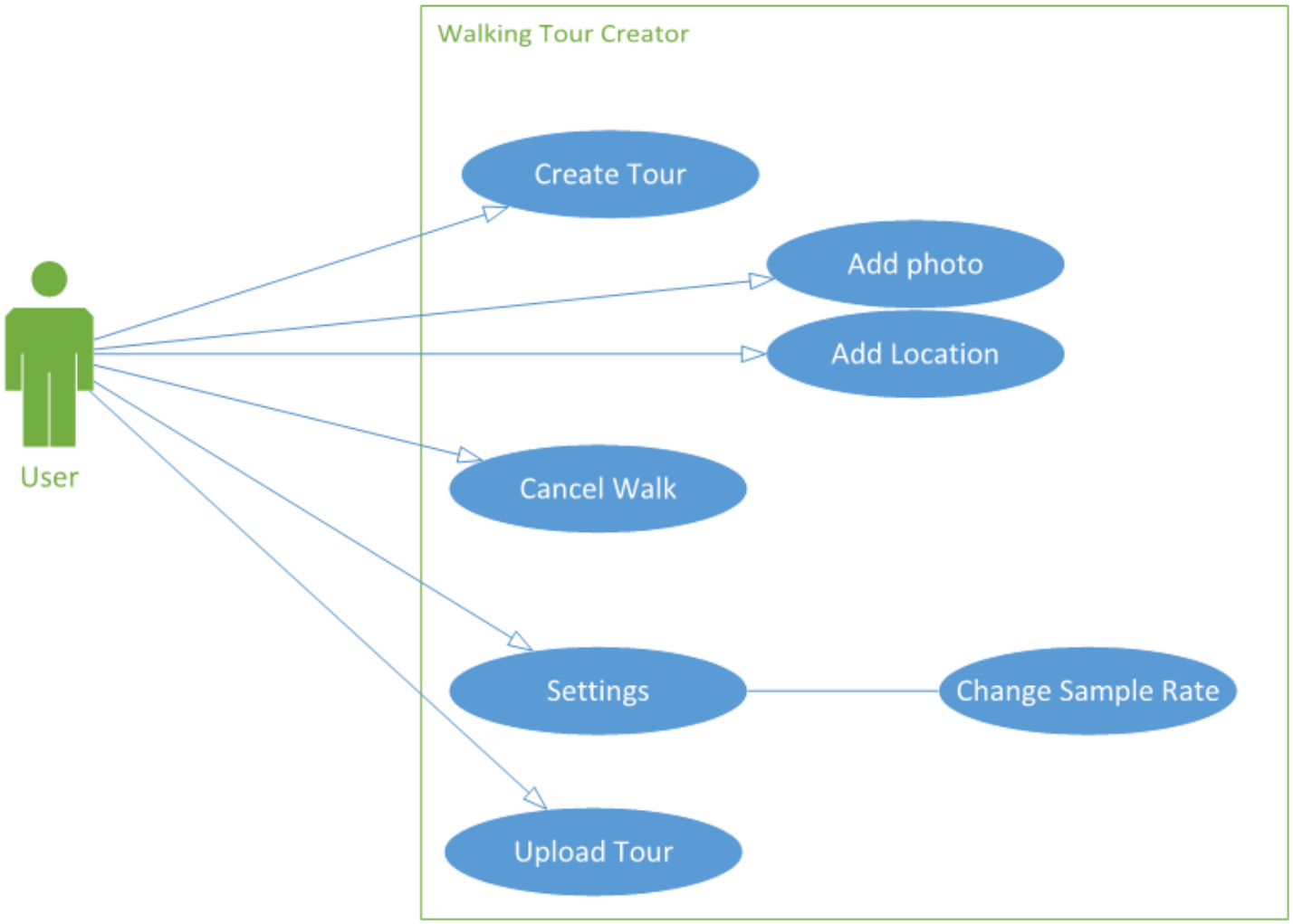
Because of its wide variety of browser support, features and simplicity, the Leaflet API stood out as exactly what the project group would need to use.

2.2. Target user base

The client has suggested second year computer science students as target base, but the app is not targeted specifically at that user group. Its purpose can be different for different users thus making it usable in a variety of ways and the actual system has to be easy to use by most age/background groups

3. Use case

3.1. Android Use Case



3.2. Descriptions of the Android Use Case

3.2.1. **Create Tour:**

This will allow the user to create a new walking tour, regardless of logging in to our servers or not the user will be asked to select a title for their tour and a short description (as minimal) before starting the tour, during the tour they can attach photos of local scenery and the like with a short description of the photo.

3.2.1.1. Add Photo:

This will use the built-in camera app on the android device to take a photo for the user to add to their walk.

3.2.1.2. Add Location:

This will get called periodically to allow for an accurate walking tour to be created.

3.2.1.3. End Walk:

When the user presses to end the tour, they will be given a summary of the tour, and be asked to fill in the missing long description (if they did not do so before creating the tour), they will then be prompted to see if they want to have the walk saved locally or uploaded straight away.

3.2.2. **Settings**:

This is where the user will give their preferences for different in app options, such as the upload option, if a user is concerned about their data limits they can choose to only upload over Wi-Fi.

3.2.2.1. Upload Tour:

If the user has chosen to save their tours locally on the device, this option will allow the user to select which tours they wish to upload to the website.

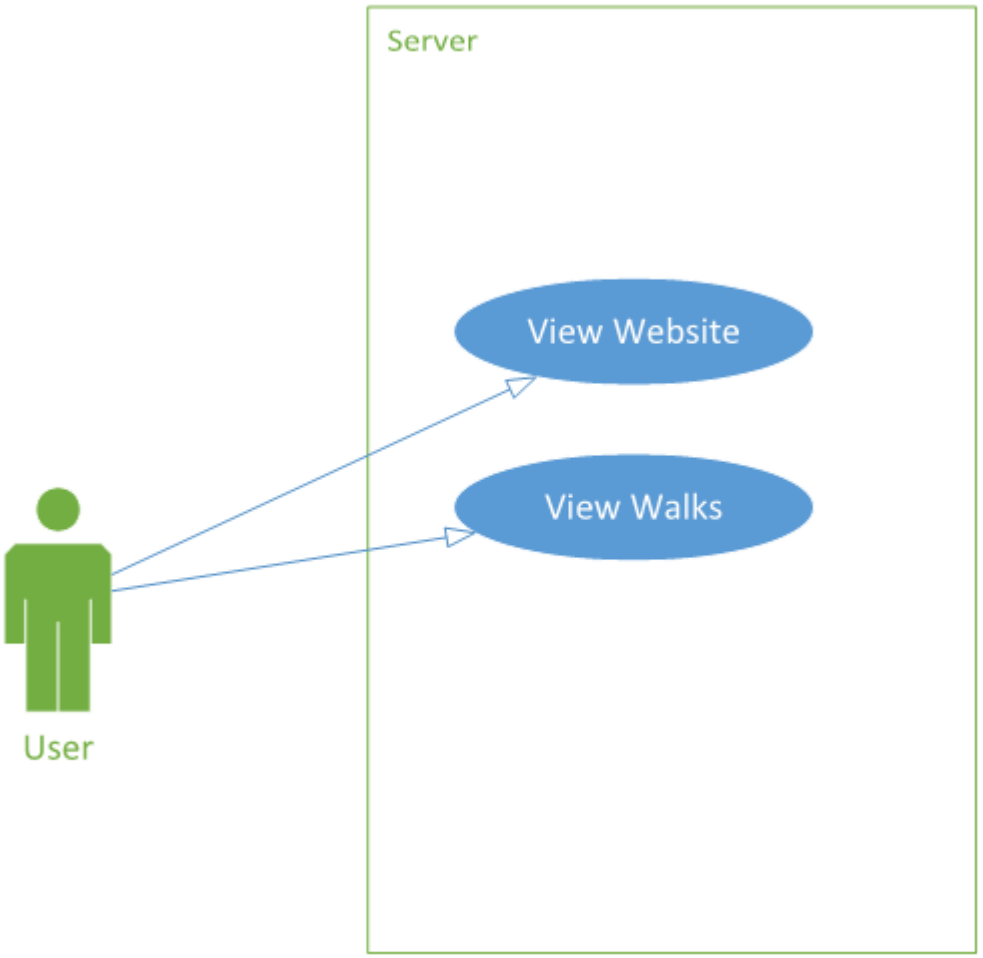
3.2.2.2. Cancel Tour:

This will allow the user the ability to cancel a tour at any time if they desire.

3.2.2.3. Change sample rate:

The sample rates that the user can change to are two, four and six seconds.

3.3. Web Side Use Case



3.4. Description of the Web Side Use Case

3.4.1. View Website

Displays our walking tour viewer homepage.

3.4.2. Database Dependent Use Cases

3.4.2.1. Manage Walks

Allows the user the ability to edit/delete previous walks on their profile.

3.4.2.2. View Tour:

This will display a map for the user to view with "pins" in it that have pictures attached along with the notes associated with the walk.

3.4.2.3. Search Walks:

Searches based on keyword/location and delivers the top “x” amount of walks.

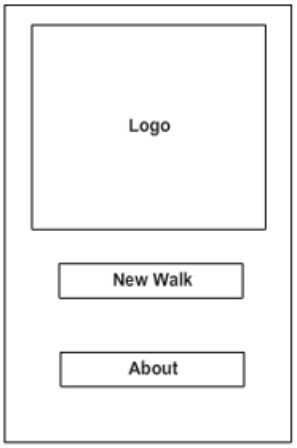
3.4.2.4. Receive Tour:

Takes a MIME file sent from the app, decodes it and stores the information in a SQL database.

4. Android User Interface Design

The following wireframes are initial concept for the design and will be modified accordingly with time.

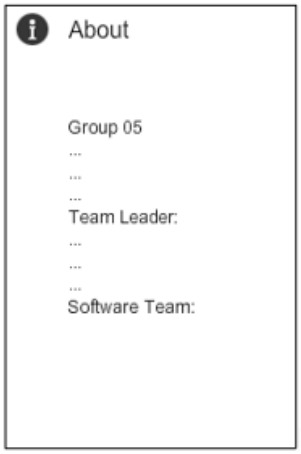
4.1. Main Screen



On the main screen the user can see whether he is logged in or not. The start button will take the user to the preparation screen for a recording where they can add descriptions and a title for the walk.

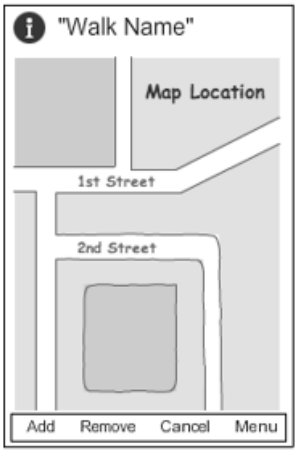
The settings and Info buttons take the user to the specified sub-screen.

4.2. About screen



This screen displays information on the app such as version, development team and so on. The user will be able to get contact details for the group from there.

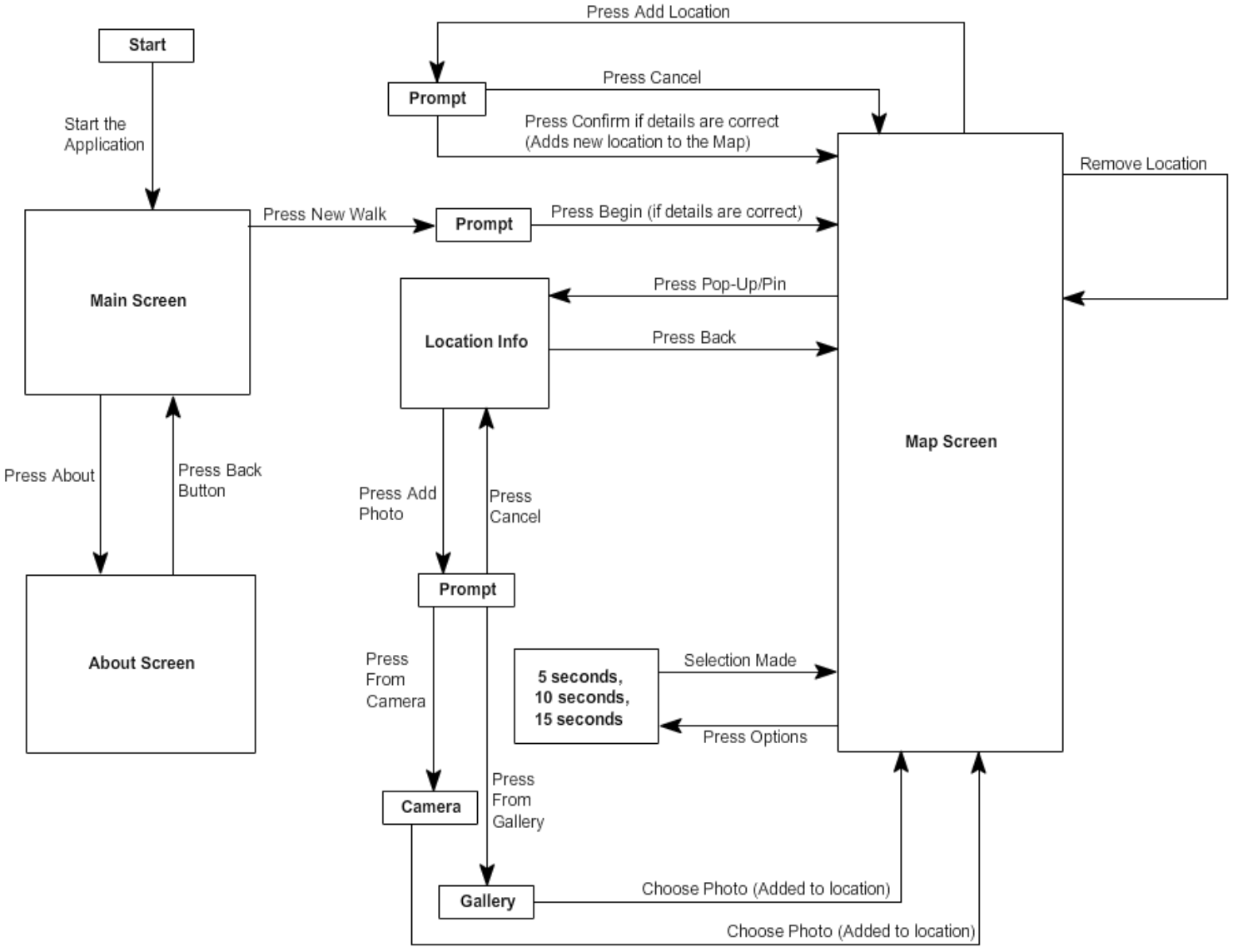
4.3. Map screen



On the map screen the user will see his position on the map and the path that he has already walked on. They can use the add photo button to select/take a photo and add a description to it. The cancel walk will exit the walk without saving it.

4.4. User Navigation

Below is a flow Diagram that describes the relationships between the screens in the Android App. The arrows show the direction of the link and what the user needs to do in order to follow that arrow.



5. Website User Interface Design

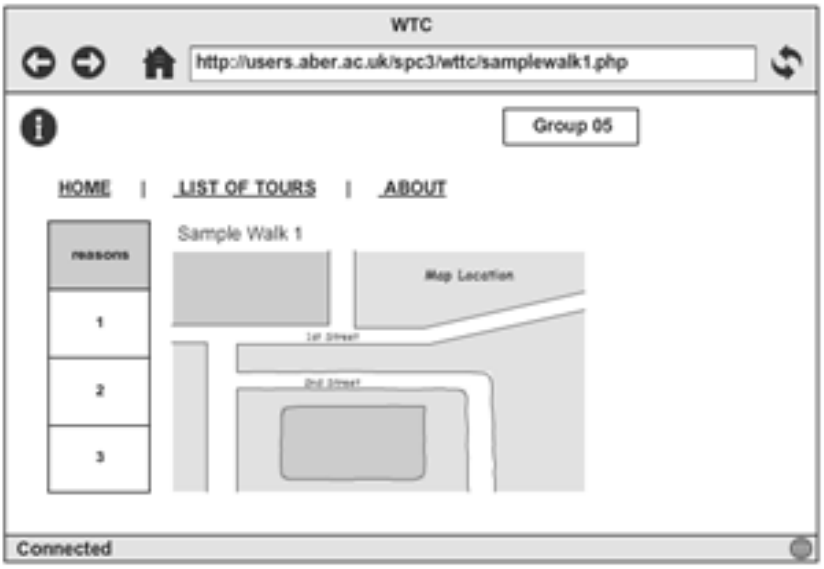
5.1. Home Page

The home page contains the list of recent walks and the user can choose a walk instantly to show up on the map.

5.2. List of Walks page

The list of walks page displays all the walks with all the descriptions. When a walk is selected, the user is taken to the map page to display it.

5.3. Map page

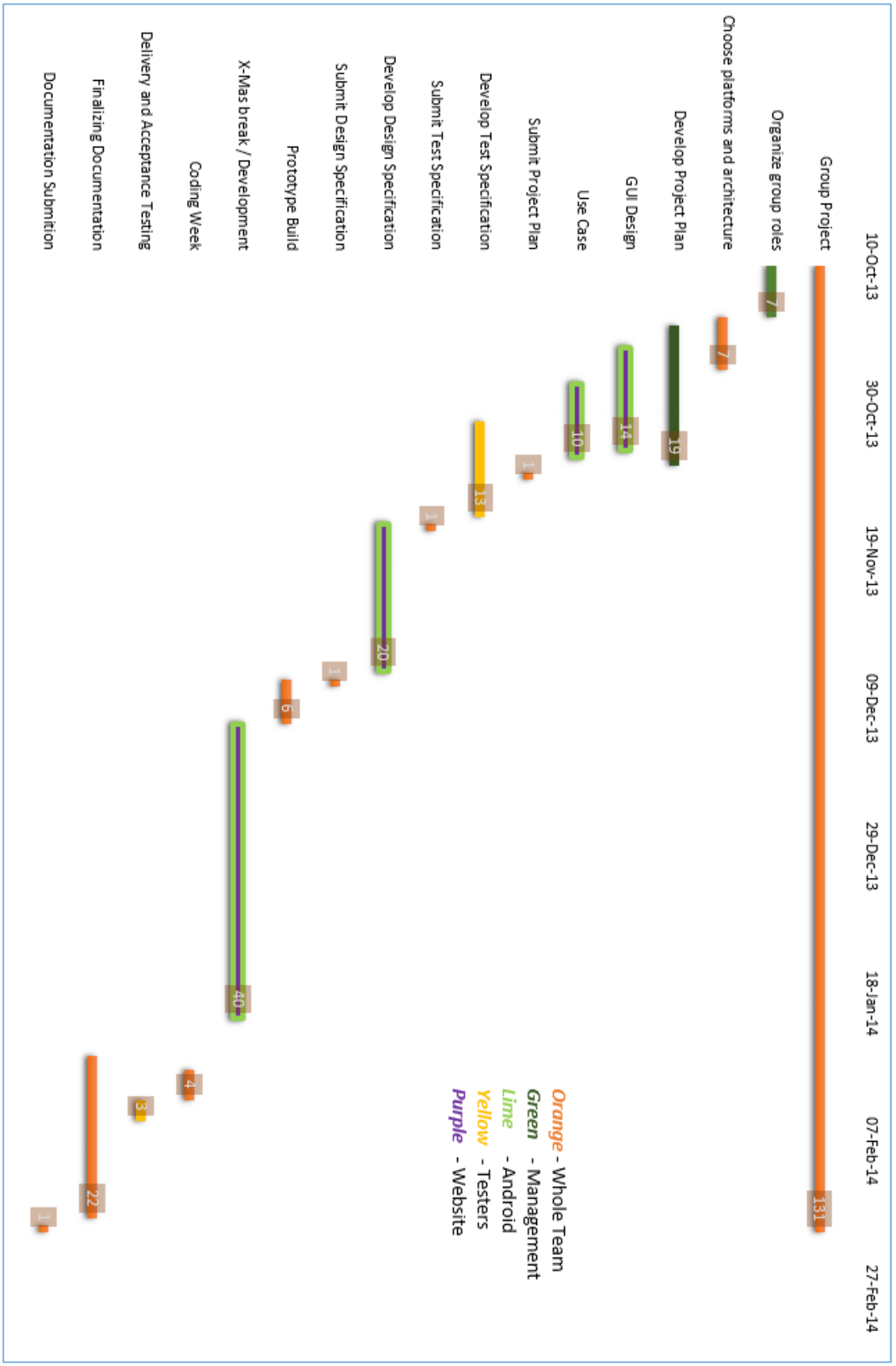


The map page is dedicated to one tour and displays it completely with pins as the key locations. The pins can be clicked to display the name, description and photos.

5.4. About page

The about page contains the group's details and contact information.

6. Gant Chart


10-Oct-13
30-Oct-13
19-Nov-13
09-Dec-13
29-Dec-13
18-Jan-14
07-Feb-14
27-Feb-14
Group Project
Organize group roles
Choose platforms and architecture
Develop Project Plan
GUI Design
Use Case
Submit Project Plan
Develop Test Specification
Submit Test Specification
Develop Design Specification
Submit Design Specification
Prototype Build
X-Mas break / Development
Coding Week
Delivery and Acceptance Testing
Finalizing Documentation
Documentation Submition
Orange - Whole Team
Green - Management
Lime - Android
Yellow - Testers
Purple - Website

7. Risk Assessment

Risk Event	*Risk Level*	*Mitigation*
Absence of team member	*2*	All meetings are announced with enough time before them for everyone to make room in their schedules. If someone cannot make it, they need to appologise and look through the minutes of the meeting.
Loss of contact	*4*	Members need to be in constant contact throughout the project. To avoid such a scenraio, all members have exchanged all possible contact information.
University filestore/GitHub Downtime	*3*	The University filestore is reliable and so is GitHub, but in case of such an event relevant information will be sent via the group e-mail/social media. Everyone in the group will cease work and wait for further instructions on what exactly everyone is expected to work so that there is no duplicating or loss of information.
File corruption	*4*	The files are going to be backed up on the university filestore and on GitHub so as long as team members are concious about their duties to back up everything, the risk is not too high.
Illness or other unexpected circumstance	*5*	In the event of serious illness or other event preventing a team member to contribute, they should inform everyone else as soon as possible so that appropriate action can be taken.
Documentation errors	*5*	Every document, after it has been drafted, has to be presented to the whole team. After everyone has agreed on it's contents and the QA manager has accerted that it fits within standards, it is finished and submitted.
Not fuffiling the project timetable	*5*	Every task to be started as soon as possible allowing enough time for revision and redaction. Programming and web teams have to keep close contact with the project leader and QA manager to make sure they are working as efficiently as possible.
Parts of the impelementation missing or incomplete	*4*	The web and android teams have to document every part of the system they build and refer to the project leader and QA manager if any problems arise. The project lead and QA manager have to keep track of general progress and make sure all requirements are fufilled.

Risk Level	*2*	*3*	*4*	*5*	*6*
Consequences	*Low*	*Low/Med*	*Med*	*Med/High*	*High*

8. References

[1] QA Document SE.QA.RS – Requirement Specification.
[2] QA Document SE.QA.02 – Project Management Standards.

9. Document History

Version	CCF No.	Date	Changes Made to the Document	Changed by:
1.0	N/A	06/11/2013	N/A – First release of project plan	srr11
1.1	N/A	06/11/2013	Changed the Web Side Use Case	bmo
1.2	N/A	03/12/2013	2.1.3 - Referenced requirement specification document. 2.1.5 - Added where JSON is being used. 4.2 - Removed the log in and delete walk feature. 4.4 - Removed subsection entirely.	njv1
2.0	#1	28/01/2013	Updated information about the project and methodologies used in development, updated wireframes for the android app and the website, updated the UML Use-Case diagrams for both platforms. Updated Gantt chart.	srr11

E. Revised test specification

Group Project 05
Test Specification

Authors: bmo; sr11; hac22; wia2; wjl3; njv1
Config Ref: SE_05_TS_01
Date: 14/11/13
Version: 2.0
Status: Release

Department of Computer Science
Aberystwyth University
Aberystwyth, Ceredigion
SY23 3DB
Copyright © Group 05
Aberystwyth University 2013

Contents

1. Introduction

1.1. Purpose of this document

The purpose of this document is to display all of the system testing that the team will use to make sure the walking tour application meets the standards of the requirement specification that the client gave.

1.2. Scope

This document includes all of the system tests that the team will implement on the android and server side of the application.

This Document should be read after familiarisation with the Project Management Plan [1]. The specifications listed in Functional Requirements [2] and the Test Procedure Standards [3].

1.3. Objectives

The objective of this document is to show what tests will be used to make sure the application, the website and database will meet the requirement specification given by the client. All tests will have description and implementation information. The client should make sure that the tests provided here test the application fully and if the application passes all the tests displayed here, the client is to accept the project as complete.

2. Android side testing

2.1. System testing table

Test Ref	Req being Tested	Test Content	Unexpeted Input	Expected Input	Output	Pass Criteria
SE_05_TEST_01	FR1	When the user creates a "new walk" check that they are prompted to add a name, a short and long description.	N/A	N/A	Prompt appears after pressing "new walk" button	Prompt shows and asks for name, short and long description
SE_05_TEST_01.1	FR2	When information is added to the prompt, check that the short description has a maximum of 100 characters and the long description has a maximum of 1000 characters	exceed maximum description lengths	location description length within maximum	values are stored and added to the new walk information	descriptions are within the maximum character boundaries
SE_05_TEST_02	FR1	When the new Tour is created, check that the new Location shows on the map	No GPS signal or Mobile data	Adding a Walk name, short and long description	GPS coordinates at the user's current position.	Coordinates are correct, and a pin shows up showing the walk name.
SE_05_TEST_03	FR2	When creating a new Walk, check that the title of the tour displays on the action bar.	Blank title	Name of the walk	Value is stored and shown in the action bar.	Correct title showing in the action bar when the user presses 'Begin'
SE_05_TEST_04	FR3	When a Walk is being created, check that the Location is added to map and saved to memory	No GPS signal or Mobile data	Name of location & location coordinates.	Location is now added to the map.	Correct location coordinates and name are stored.
SE_05_TEST_05	FR3	When adding a Location, check if the user can add a name along with a short description.	blank text fields	Name and Location description	Name and Location description is added to the object.	Correct name and description within a pin to that location.
SE_05_TEST_06	FR3 / FR4	When adding a Location, check if a time stamp is added to locations and infact the correct time.	No GPS signal or Mobile data	New location added.	Timestamp is added to location with correct time.	Correct time is added to the correct location.
SE_05_TEST_07	FR4	Check if the user can add a photo for a location from the gallery.	doesn't allow you to access the gallery	Photo from Gallery.	Photo added to the location on the map.	Correct photo from gallery added.
SE_05_TEST_08	FR4	Check if the user can add multiple photos to a single Location	doesn't allow you to add any more photos	Photo from Gallery.	Photo added to the location on the map along with previously added photos	Correct photo from gallery added.
SE_05_TEST_09	FR5	When a walk has already been created with locations added, check if the the user can edit the information.	user deletes all of the information from the text fields	Changed walk name, long or short description. Or everything.	Updated changed information	name has changed and showing in the action bar. Short and Long description changed also.
SE_05_TEST_10	FR3 / FR4	Check if coordinates are added to the map during the walk.	No GPS signal or Mobile data	No input (Automatic)	Coordinates of location shown on map.	Correct coordinates stored on map.
SE_05_TEST_11	FR5	When wanting to remove a walk, check if the walk data is deleted.	N/A	Cancel command.	Return to home screen.	Tour data is deleted correctly and the home screen is displayed.
SE_05_TEST_12	FR6	Check if the data sent to the server is saved as a MIME message.	Not saved correctly as a MIME message	Tour object.	MIME string.	MIME string is correct and not null.
SE_05_TEST_13	FR7	Check that WTC stores tour data when user switches android application.	WTC doesn't store the information when switching app	Application switch.	n/a	Application is idle and stores data.
SE_05_TEST_14	FR7	WTC reloads data when user switches back to the application.	No information to reload	Resume application.	n/a	Reloads correct data.

3. Server side testing

3.1. System testing table

Test Ref	Req being tested	Test content	Unexpected Input	Expected Input	Output	Pass Criteria
SE_05_TEST_15	FR8	Test that the sample route has correct GPS locations.	N/A	Co-ordinates of sample route.	Vector polygon plotted on the map.	The plotted line is the same as the co-ordinates of the sample route.
SE_05_TEST_16	FR8	Test that the points of interest are recorded along the sample tour.	N/A	Set of points of interest.	The points of interest plotted on the map.	The co-ordinates of the points of interest are plotted to the correct locations on the map.
SE_05_TEST_17	FR8	Test that the pop-up shows up in the correct location and it holds the correct information.	N/A	Click on a point of interest.	A CSS popup with correct information.	A css popup appears by the POI on the map. The title and short description are the same as the correct route.
SE_05_TEST_18	FR8	Test image thumbnails in pop-ups and check that they are the correct images.	N/A	Click on point of interest on the sample tour.	Strip of thumbnails.	The thumbnails represent all of the images associated with the point of interest.
SE_05_TEST_19	FR6	Check that the phone can send a HTTP post to the server.	Empty POST request.	A user sending a request to the server, via the phone.	Site log file.	The log file is updated with the transaction associated with the user.
SE_05_TEST_20	FR6	Check that the data is formatted as a valid MIME message.	Invalid MIME message	A String field in the post request.	The attachments and the JSON data from the MIME messages.	The PHP program is able to decode the MIME message and extract the data and the attachments.
SE_05_TEST_21	FR6	Check that the data is formatted as valid JSON.	Invalid JSON	String of JSON data.	Records of the tour and its relations in the database.	The record and its relations in the database contain the correct data in the correct fields.
SE_05_TEST_22	FR6	Check that the image files have been saved.	Image in the wrong format or missing an expected image	The image files associated with the sample route.	The file system on the server.	The images have been saved to the correct directory within the file system on the server.

4. References

[1] Software Engineering Group 05. Project Plan. S. Raychev, B. O'Donovan, H. Clark, W. Arslett, W. Lea, N. Vicker and S. Clasby. 1.2 Release.

[2] Software Engineering Group 05. Requirements Specification. C. J. Price and B. P. Tiddeman. SE.QA.RS. . 1.4 Release.

[3] Software Engineering Group 05. Test Procedure Standards. C. J. Price, N. W. Hardy and B. P. Tiddeman. SE.QA.06. . 1.7 Release.

5. Document change history

Version	CCF No.	Date	Changes Made to the Document	Changed by
1.0	N/A	14/11/2013	N/A – First release of test specification.	srr11
1.1	N/A	15/11/2013	Corrected server side test table	wia2
1.2	N/A	24/01/2014	Fixed problems with document and added test log forms	bmo, njv1
2.0	#1	28/01/2014	Removed all issues pointed out by Berinie.	bmo

F. Revised design specification

Group Project 05
Design Specification

Authors: bmo; srr11; hac22; wjl3; wia2; njv1; spc3
Config Ref: SE_05_DS_01
Date: 06/12/13
Version: 1.2
Status: Release

Department of Computer Science
Aberystwyth University
Aberystwyth, Ceredigion, SY23 3DB
Copyright © Group 05
Aberystwyth University 2013

Table of Contents

1. Introduction

1.1. Purpose of this document

The purpose of this document is to describe the outline design for the Walking Tour Creator system, consisting of an Android application, database and website taking into account the details of the group project requirements and group project quality assurance.

1.2. Scope

This document includes detailed description of:

- The classes used in the Android application;
- The methods used in each class;
- Sequence diagrams;
- Interaction between the application and the database;
- Database design and data handling;
- Website design.

This document should be read after familiarisation with the Project Management Plan [1] and Test Specification [2]. The specifications listed in General Document Standards [3], Design Specification Standards [4] and Java Coding Standards [5] have been followed in the process of constructing this Design Specification.

1.3. Objectives

The objective of this document is:

- To describe the main components of the Walking Tour Application;
- To describe the main components of the database and website;
- To depict the dependencies between the components.

2. Decomposition Description

2.1. Programs in system

The Walking Tour System consists of two main components:

- The Android application;
- The Website and database;

The Android application will allow the user to create a walking tour. This tour will consist of automatically added locations through GPS along the walk as well as separate points of interest the user has added on their own along with a description and possibly photos. When the user is done, the walk can be uploaded to the online database in the form of a MIME message containing the information on the walk formatted as a JSON file. This message is sent through HTTP POST, intercepted by the PHP on the site and gets stored in the database. The Website allows the user to view walks currently stored in the database by querying it and displaying the walks using the Mapping API.

2.2. Significant classes in the Android program

2.2.1. Main Activity

This is the launcher activity for the app, when the app first starts this is the screen the user is presented with.

2.2.2. Walk Activity

This is the main part of the app, this where the user builds their tour. From here they can:

- Add/remove points of interest.
- Add/remove photos to/from a point of interest.
- Cancel their walk.
- Save their walk for later viewing.
- They are also able to adjust the sample rate for non-key locations to save on how much data they transmit to the server.

2.2.3. About Activity

This is just a credits screen. It mentions who is on the team, the app name and version number. There are also some supporting UI classes for dialogs.

2.2.4. Tour Save

This class is used to save the current tour the user is creating to a temporary file so they don't lose their tour when they leave the app.

2.2.5. WTC Tour

This class represents the tour the user is making. This class contains the name, short description, long description and list of locations in the tour.

2.2.6. WTC Location

This class represents non-essential locations in the tour and is only used for tracking purposes.

2.2.7. WTC KeyLocation

This class represents the key locations in the tour. This class contains a name for the location, a description have the location and list of photos.

2.3. Web program components

2.3.1. Database

The Server side of the system consists of a database, holding the information on the tours and a website that displays the tours on an interactive map to the user.

All the data is packed into a MIME message containing all the information about the tour formatted as a JSON file. Appendix A defines the format of the MIME message; Appendix B defines the format of the JSON file. The message is transmitted from the android device in a HTTP POST request. The value will be paired using the key "message". The data is accessed in PHP using the $_POST ['message'] handle. The request is made to the file upload.php which is stored in the root of the website. All requests will be recorded in by upload.php in a file called log.txt in the root of the site as per the testing strategy**.**

The data set will need to contain a title for the tour, a short description of the tour and a long description, a collection of GPS coordinates for the route, a collection of locations associated with GPS coordinates on the route including images and description, the total time of the route and the total distance of the route.

The Database will have the following structure:

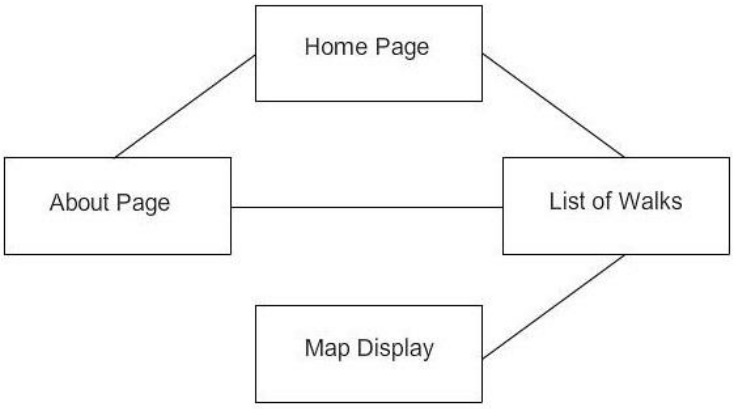
Table Name	Table Description
ListOfWalks	A list of walks/ tours, which the program will display.
Location	A list of geographical locations referencing a record in the tour table, describing the route of the tour as a sequence of locations.
PlaceDescription	A list of points of interest along tours referencing a location.
PhotoUsage	A list of photographs referencing a point of interest

2.3.2. Website

The Website consists of 4 main pages:

Page Name	Page Description
Home page	The index page containing general information.
List of walks page	Displays a list of walks available to view from the database.
Contacts page	Information on how to contact the group.
About page	Other info for the site such as version and links.

The website will contain the following pages:



On the Home page there will be a little bit of information about the WTC project and a few of the most common routes. The layout of the home page has a content which will hold the most used routes. There will be a map on that screen to display one of the selected walks immediately.

The list of walks page contains a list of all walks in the database that the user can select to view on the map.

The map page will have the mapping API and will take the information on locations for a walk from the database and display them. Locations added by the user will be indicated by pins, which on click will display additional information the user has given and a list of photos that can be viewed.

The Contact page is so people can get in touch and contact us via a form which will be written in either JavaScript or PHP which will allow people to send us feedback and suggestions on how to improve on the system.

The About page will contain general information for the site such as creators, copyright and other legal matters and site version. The names of the developers and a contact e-mail address will be present so we can be contacted with recommendations etc.

All of the pages are accessible via a Navigational Bar under the header with buttons to each of the pages on the website, allowing for a dynamic experience and feel to the site as the buttons change upon behaviour of highlighted, clicked and also passive. The map will only be displayed when a walk is selected to be displayed.

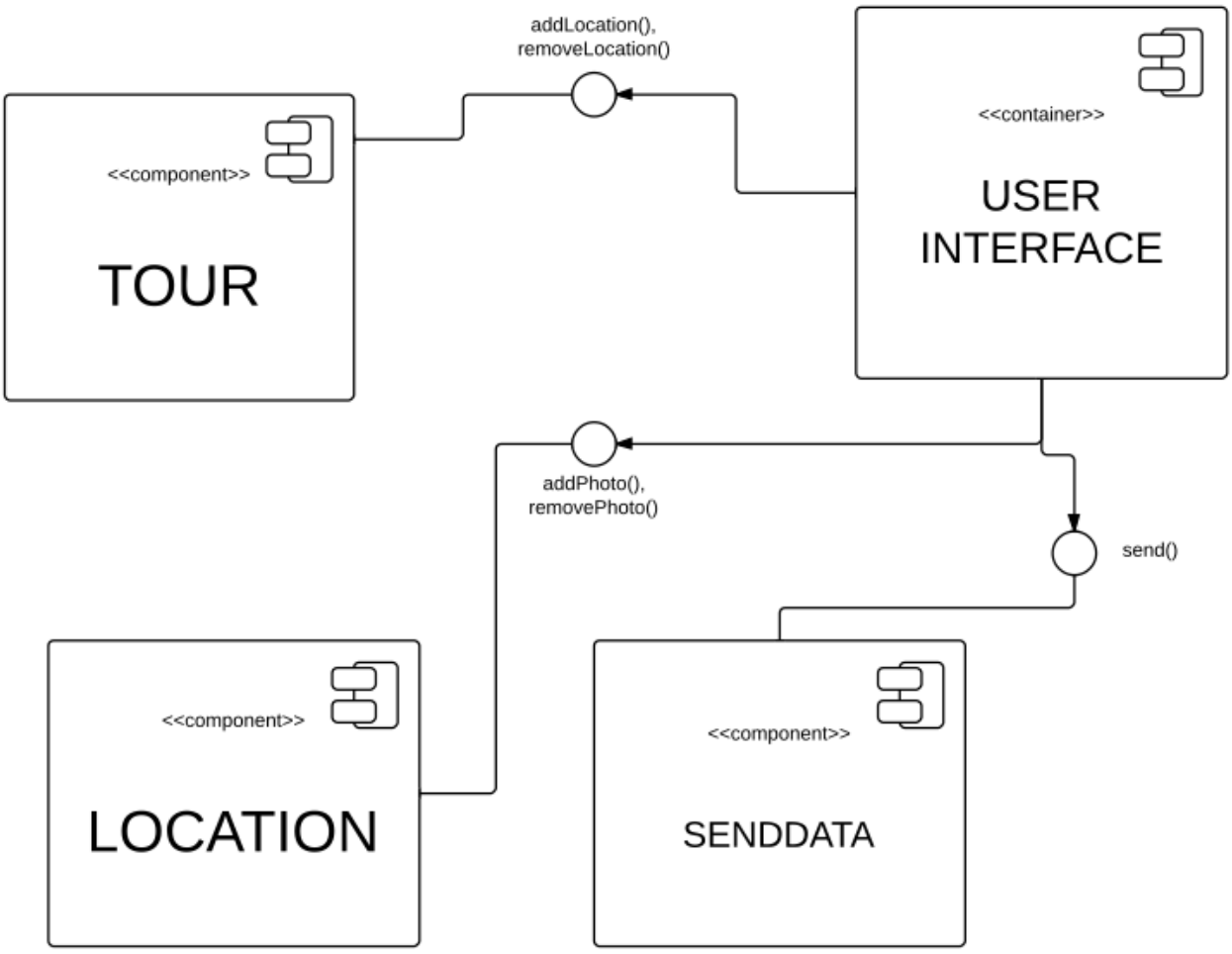
2.4. Table mapping requirements

Requirement	Modules providing requirement
FR1	MainActivity class
FR2	NewWalkFragment class, WTCTour class, ListOfWalksTable
FR3	WalkActivity class, WTCLocation class, LocationsTable
FR4	WalkActivity class, WTCKeyLocation class, PhotoUsageTable
FR5	EndWalkFragment class, FinishWalkFragment class, TourSave class, DialogCallbacks interface
FR6	NoNetworkFragment class, SendData class, Upload.php
FR7	WalkActivity class, TourData class
FR8	Index.php, Map.php, List.php
FR9	Upload.php

3. Dependency Description

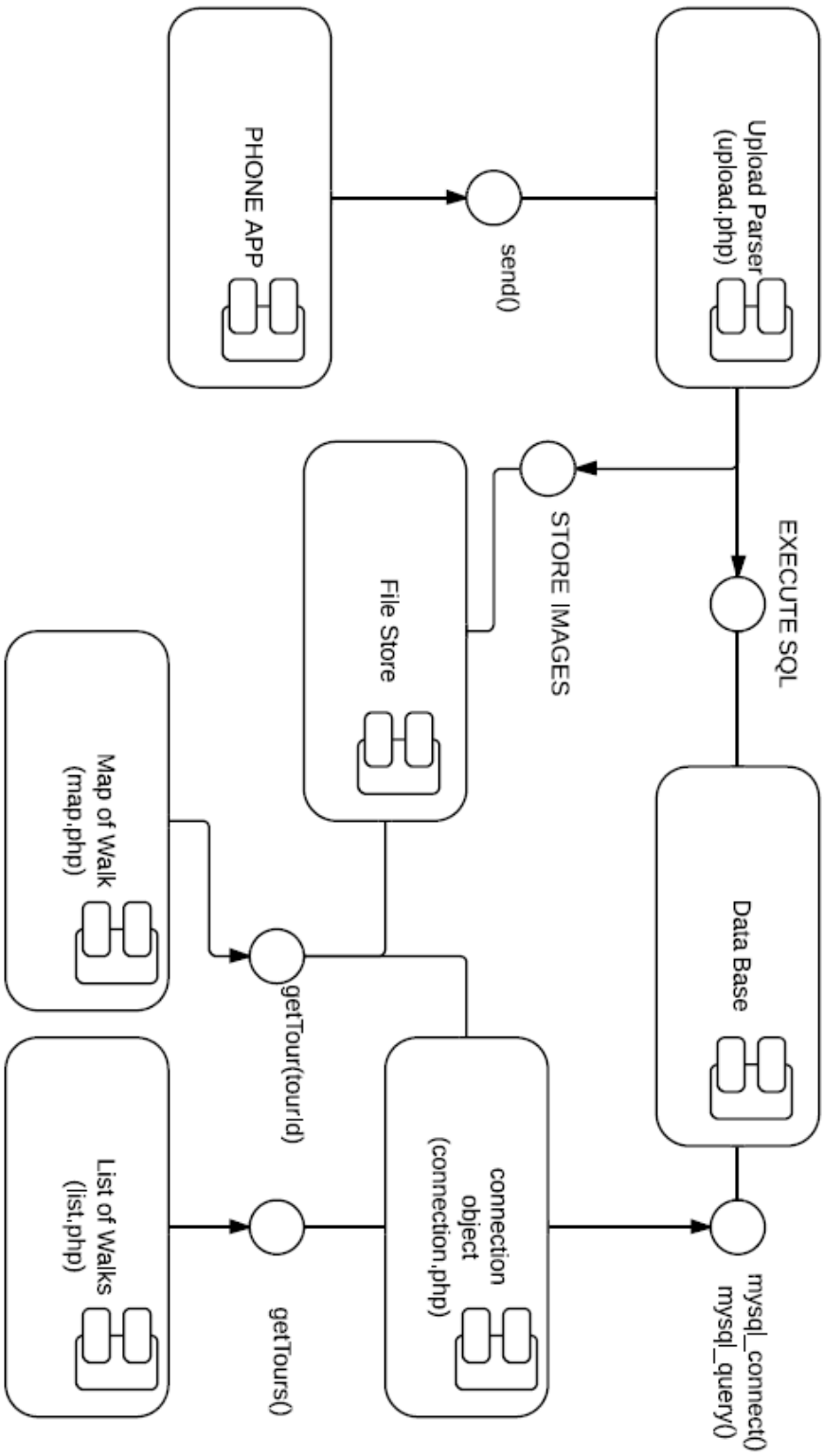
3.1. UML Component Diagram for the Android application

The diagram depicts the different components, the Android application consists of and how they interact with each other. The User Interface component can interact with the Tour, Location and Communication components through the respective methods depicted in the diagram.



3.2. UML Component Diagram for the Database

The diagram depicts the different components, the Server side application consists of and how they interact with each other. After the data is received from the phone application it is processed by the Upload Parsed component and sent to the Database component. From there depending on what the user wants to see, either the List of walks or the Map component is called.

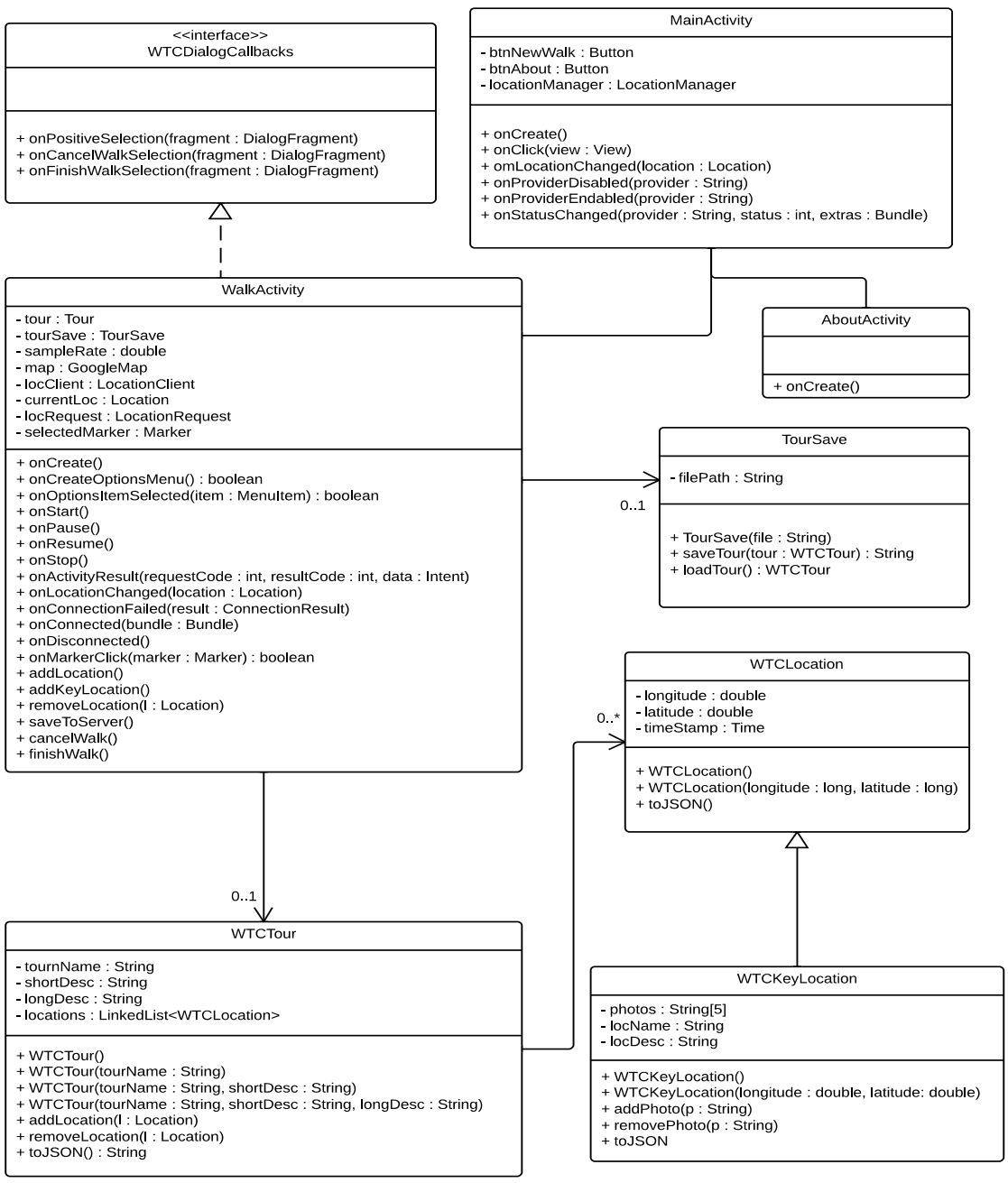


4. Interface description

This section contains all of the classes in the Android application and their class skeletons, depicting the methods that are to be implemented in them along with the appropriate JavaDoc commenting for clarification.

4.1. Android Class Diagram

The UML class diagram depicts all the significant classes the application consists of. The default Android classes such as Activity are extended by some the program uses like MainActivity etc. but are omitted from the class diagram.



4.2. Main Activity class

```
/*
 * @(#) MainActivity.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class MainActivity extends Activity implements OnClickListener, LocationListener {

        /**
        * Creates this activity and sets its layout. It also initializes the buttons and
sets
        * their listeners.
        */
        @Override
        protected void onCreate(Bundle savedInstanceState) {
        }

        @Override
        protected void onStop(){
        }

        /**
        * Dictates what what happens when the user clicks a button.
        */
        @Override
        public void onClick(View view) {
        }

        @Override
        public void onLocationChanged(Location location) {
        }

        @Override
        public void onProviderDisabled(String provider) {
        }

        @Override
        public void onProviderEnabled(String provider) {
        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
        }
}
```

4.3. About Activity class

```
/*
 * @(#) AboutActivity.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class AboutActivity extends Activity {

        /**
        * Creates this activity and its layout.
        */
        @Override
        protected void onCreate(Bundle savedInstanceState) {
}
```

4.4. Walk Activity class

```
/*
 * @(#) WalkActivity.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class WalkActivity extends Activity implements ConnectionCallbacks,
OnConnectionFailedListener,
        LocationListener, OnMarkerClickListener, WTCDialogCallbacks{

        @Override
        protected void onCreate(Bundle savedInstanceState) {
        }

        @Override
        protected void onStart() {
        }

        @Override
        protected void onPause() {
        }

        @Override
        protected void onResume() {
        }

        @Override
        protected void onStop() {
        }

        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        }

        @Override
```

```
public void onLocationChanged(Location location) {
}

@Override
public void onConnectionFailed(ConnectionResult result) {
}

@Override
public void onConnected(Bundle bundle) {
}

@Override
public void onDisconnected() {
}

/**
 * Sets the selected marker.
 *
 * Called when the user clicks on a map marker.
 */
@Override
public boolean onMarkerClick(Marker marker) {
}

/**
 * Creates the action bar where all the buttons are.
 */
@Override
public boolean onCreateOptionsMenu(Menu menu) {
}
/**
 * Dictates what happens when the user clicks a button on the options menu.
 */
@Override
public boolean onOptionsItemSelected(MenuItem item) {
}
/**
 * Used to add a new key location to the tour.
 */
@Override
public void onPositiveSelection(DialogFragment fragment) {
}
/**
 * Used to cancel the tour if the user selects 'OK' on the dialog.'
 */
@Override
public void onCancelWalkSelection(DialogFragment fragment) {
}

@Override
public void onFinishWalkSelection(DialogFragment fragment) {
}
```

```
    /**
    * Adds a location to the tour.
    */
    public void addLocation() {
    }




    /**
    * Adds a key location to the walk.
    */
    public void addKeyLocation(WTCKeyLocation keyLoc) {
    }

    /**
    * Removes a location from the tour.
    *
    * @param marker the marker being removed from the map; also used to identify which
location to      remove.
    */
    public void removeLocation(Marker marker) {
    }

    public void addPhoto() {
    }

    /*
     * The following method was taken from the Android Developers Documents
     *
     * URL: http://developer.android.com/training/camera/photobasics.html
     */
    private File createImageFile() throws IOException {
    }

    public void removePhoto() {
    }

    /**
    * Saves the tour to the server.
    */
    private void saveToServer() {
    }

    /**
    * Stops the recording of the and deletes the data.
    */
    public void cancelWalk() {
    }

    /**
    * Adds a stopping location to the tour and stops the recording.
    */
```

```
        public void finishWalk() {
        }

        /**
        * @param tour the new value for this.tour.
        */
        public void setTour(WTCTour tour) {
        }




        /**
        * @return the tour.
        */
        public WTCTour getTour() {
        }

        /**
         * Sets a new value for this.sampleRate.
         *
         * @param sampleRate the new value for this.sampleRate.
         */
        public void setSameRate(long sampleRate) {
        }
        /**
         * @return the sample rate.
         */
        public long getSampleRate() {
        }
}
```

4.5. WTC DialogCallbacks interface

```
/*
 * @(#) WTCDialogCallbacks.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public interface WTCDialogCallbacks {

        /**
         * This method is used when the user clicks the positive button on a dialog.
         *
         * @param fragment the DialogFragment that made this callback.
         */
        public void onPositiveSelection(DialogFragment fragment);

        /**
         * This method is to be called when the user clicks the positive button
         * on the CancelWalkFragment.
         *
         * @param fragment the DialogFragment that made this callback.
```

```
         */
        public void onCancelWalkSelection(DialogFragment fragment);

        /**
         * This method is to be called when the user clicks the positvite button
         * on the FinishWalkFragment.
         *
         * @param fragment the DialogFragment that made this callback.
         */
        public void onFinishWalkSelection(DialogFragment fragment);
}
```

4.6. TourSave class

```
/*
 * @(#) TourSave.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class TourSave {
        /**
         * Constructs a TourSave object with a file path to where the tour data will be
saved.
         *
         * @param filePath the file path for where the tour will saved on the device.
         */
        public TourSave(String filePath){
        }

        /**
         * Serializes the tour data out to file.
         *
         * @param tour the tour being saved.
         * @return a message to say whether the save was successful or not
         */
        public String saveTour(WTCTour tour) {
        }

        /**
         * Deserializes a tour data file.
         *
         * @return the tour loaded in from file
         */
        public WTCTour loadTour(){
        }
}
```

4.7. WTCTour class

```
/*
 *  @(#) WTCTour.java Version 1.0
 *
 *  Copyright(c) Group 5 @Aberystwyth University
 *  All rights reserved
 */
public class WTCTour implements Serializable{
        /**
        * Constructs a blank tour.
        */
        public WTCTour(){
        }

        /**
        * Constructs a new tour with a specified name.
        *
        * @param tourName the name of the tour.
        */
        public WTCTour(String tourName){
        }

        /**
        * Constructs a new tour with a specified name and short description.
        *
        * @param tourName the name of the tour.
        * @param shortDesc a short description for the tour.
        */
        public WTCTour(String tourName, String shortDesc){
        }

        /**
        * Constructs a new tour with all necessary details.
        *
        * @param tourName the name of the tour.
        * @param shortDesc a short description for the tour.
        * @param longDesc a long description for the tour.
        */
        public WTCTour(String tourName, String shortDesc, String longDesc){
        }

        /**
        * Adds a location to the tour.
        *
        * @param location the location being added to the tour.
        */
        public void addLocation(WTCLocation location){
        }

        /**
        * Removes a location from the tour.
        *
        * @param index the index of the location being removed.
```

```
*/
public void removeLocation(int index){
}

/**
* Takes the data stored this object and converts it to a JSON String.
*
* @return the JSON String.
*/
public String toJSON(){
}

/**
* Sets a new value for this.tourName.
*
* @param tourName the new value for this.tourName.
*/
public void setTourName(String tourName){
}

/**
* @return the name of the tour.
*/
public String getTourName(){
}



/**
* Sets a new value for this.shortDesc.
*
* @param shortDesc the new value for this.shortDesc.
*/
public void setShortDesc(String shortDesc){
}

/**
* @return the short description for the tour.
*/
public String getShortDesc(){
}

/**
* Sets a new value for this.longDesc.
*
* @param longDesc the new value for this.longDesc.
*/
public void setLongDesc(String longDesc){
}

/**
* @return the long description for the tour.
*/
```

```
        public String getLongDesc(){
        }

        /**
        * Sets the list of locations in the tour.
        *
        * @param locations the new value for this.locations.
        */
        public void setLocations(LinkedList<WTCLocation> locations){
        }

        /**
        * @return the list of locations in the tour.
        */
        public LinkedList<WTCLocation> getLocations(){
        }

        /**
         * A shortcut method to get the size of the list of locations
         *
         * @return the size of the list of locations in the tour
         */
        public int getLocationsSize(){
        }
        /**
         * Adjusts the timestamps for each location in the tour to be
         * minutes from the start of the tour.
         */
        public void fixTime(){
        }


        /**
         * Calculate a rough estimate for the distance of the tour
         */
         private double calcDist(){
        }

        private static double GreatCircle(WTCLocation loc1, WTCLocation loc2){
        }
}
```

4.8. WTCLocation class

```
/*
 * @(#)WTCLocation.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class WTCLocation implements Serializable{

        /**
         * Constructs a blank location.
```

```
     */
    public WTCLocation(){
    }

    /**
     * Constructs a location object a specified longitude and latitude.
     *
     * @param longitude the longitude of the location.
     * @param latitude the latitude of the location.
     */
    public WTCLocation(double longitude, double latitude){
    }

    /**
     * Sets a new value for this.longitude.
     *
     * @param longitude the new value for this.longitude.
     */
    public void setLongitude(double longitude){
    }

    /**
     * @return the longitude of the location.
     */
    public double getLongitude(){
    }

    /**
     * Sets a new value for this.latitude.
     *
     * @param latitude the new value for this.latitude.
     */
    public void setLatitude(double latitude){
    }

    /**
     * @return the latitude for the location.
     */
    public double getLatitude(){
    }

    /**
     * Sets a new value fort this.timeStamp.
     *
     * The time stamp is set completely by the method and therefore takes no
parameters.
     */
    public void setTimeStamp(long newTime){
    }

    /**
     * @return the time stamp for when this location was recorded
     */
```

```
        public String getTimeStamp(){
        }

        /**
         * @return the original timestamp for the location
         */
        public Calendar getOldTime(){
        }

        /**
        * Converts the data stored in this object to a JSON string.
        *
        * @return the JSON String.
        */
        public String toJSON(){
        }
}
```

4.9 WTCKeyLocation class

```
/*
 * @(#) WTCKeyLocation.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class WTCKeyLocation extends WTCLocation {
        /**
         * Constructs a blank key location.
         */
        public WTCKeyLocation(){
        }
        /**
         * Constructs a key location with a given longitude and latitude.
         *
         * @param longitude the longitude of the location.
         * @param latitude the latitude of the location.
         */
        public WTCKeyLocation(double longitude, double latitude){
        }

        /**
         * Adds the file path of a photo to this key location.
         *
         * @param path the file path.
         */
        public void addPhoto(String path){
        }

        /**
         * Removes the file path of a photo from this hey location.
         *
         * @param path the file path.
         */
```

```
        public void removePhoto(String path){
        }

        /**
         * @return the list of photo paths in this location
         */
        public List<String> getPhotos(){
        }

        /**
         * @return the name of this location
         */
        public String getLocName() {
        }

        /**
         * Sets the name of this location
         *
         * @param locName the name of the location being set
         */
        public void setLocName(String locName) {
        }

        /**
         * @return the description of this location
         */
        public String getLocDesc() {
        }

        /**
         * Sets the description of this location
         *
         * @param locDesc the description of the location being set
         */
        public void setLocDesc(String locDesc) {
        }

        @Override
        public String toJSON(){
        }
}
```

4.10 SendData class

```
/*
 * @(#) SendData.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class SendData extends AsyncTask<String, Void, Void> {

        /**
         * Constructs a SendData object.
```

```
         *
         * @param activity a link back to the activity that made this object.
         * @param tour the tour being saved to the server
         */
        public SendData(Activity activity, WTCTour tour){
        }

        /**
         * Sends the tour data to the specified server.
         * <p>The tour data and the accompanying photos are placed into a
         * Multi-Part MIME and sent to the server using HTTP POST.</p>
         *
         * @param params the list of URLs to send the tour to.
         */
        @Override
        protected Void doInBackground(String... params) {
        }

        /**
         * Notifies the user that the tour was saved and takes them back to the home
screen.
         */
        @Override
        protected void onPostExecute(Void result) {
        }
}
```

4.11 NewWalkFragment class

```
/*
 * @(#) NewWalkFragment.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class NewWalkFragment extends DialogFragment implements OnClickListener{

        /**
        * Creates the dialog box that collects the details for the walk.
        */
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
        }



        /**
        * Dictates what happens when the user clicks the positive or negative button on
the
        * dialog.
        */
        @Override
        public void onClick(DialogInterface dialog, int which) {
        }
```

```
}
```

4.12 LocationsDetailsFragment class

```
/*
 * @(#) LocationDetailsFragment.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class LocationDetailsFragment extends DialogFragment implements OnClickListener{

        /**
         * Links this dialog back to the activity that created it.
         */
        @Override
        public void onAttach(Activity activity){
        }

        /**
         * Creates the dialog and sets its layout.
         */
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
        }

        /**
         * Dictates what happens when you click a button on the dialog.
         */
        @Override
        public void onClick(DialogInterface dialog, int which) {
        }

        /**
         * @return the text field for the location name.
         */
        public EditText getTfLocName() {
        }

        /**
         * @return the text field for the location description.
         */
        public EditText getTfLocDesc() {
        }
}
```

4.13 NoNetworkFragment class

```
/*
 * @(#) NoNetworkFragment.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class NoNetworkFragment extends DialogFragment implements OnClickListener {
        /**
         * Create this dialog and sets its layout.
         */
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
        }

        /**
         * Does nothing in this instance, just needs to be here.
         */
        @Override
        public void onClick(DialogInterface dialog, int which) {
        }
}
```

4.14 FinishWalkFragment class

```
/*
 * @(#) FinishWalkFragment.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class FinishWalkFragment extends EndWalkFragment {
        /**
         * Creates the dialog and sets its layout.
         */
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
        }

        /**
         * Dictates what happens when you click a button on the dialog.
         */
        @Override
        public void onClick(DialogInterface dialog, int which) {
        }
}
```

4.15 EndWalkFragment class

```
/*
 * @(#) EndWalkFragment.java Version 1.0
 * Copyright(c) Group 5 @Aberystwyth University Computer Science Dept: Yr 2 (2014)
 * All Rights Reserved
 */
public class EndWalkFragment extends DialogFragment implements OnClickListener{

        /**
         * Link the this dialog back to the activity creating it.
         */
        @Override
        public void onAttach(Activity activity){
        }

        /**
         * Creates the dialog and sets its layout.
         */
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
        }

        /**
         * Dictates what happens when you click one the dialog buttons.
         */
        @Override
        public void onClick(DialogInterface dialog, int which) {
        }
}
```

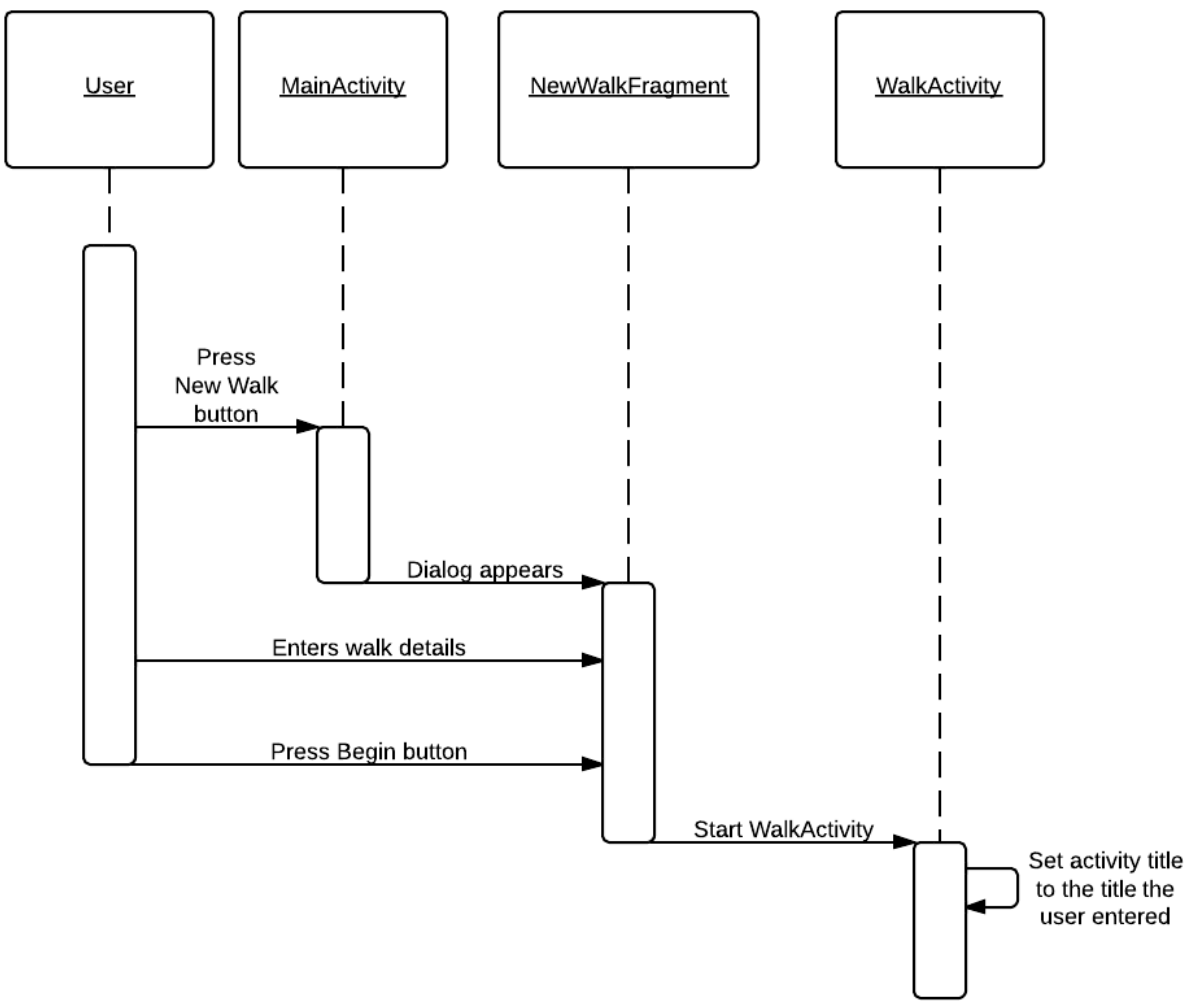
5. Detail Design

This section contains information about the more difficult and not self-evident parts of the system design. It includes the sequence diagrams for the different parts of operation to describe in more detail the order in which events in the program are executed. Further explanation on the algorithms used by both the Android and the Server side of the system are explained to clarify how the system's main components interact.

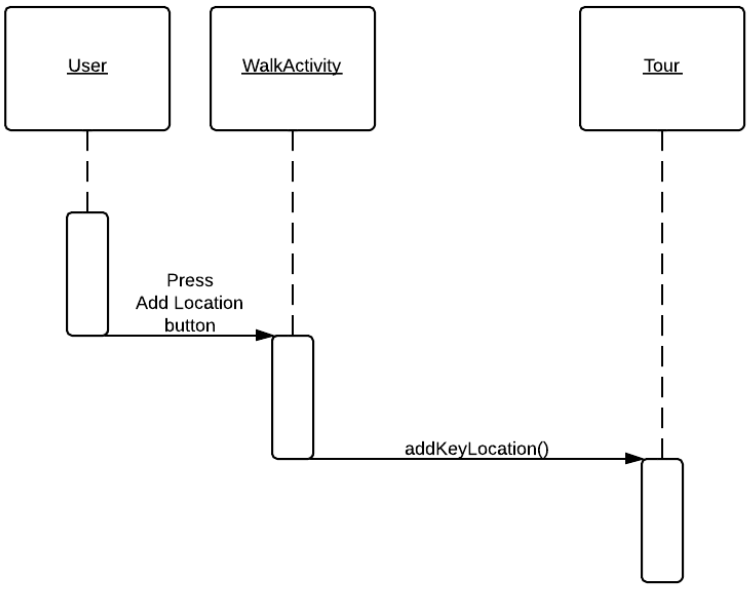
5.1. Sequence Diagrams

This section contains the sequence diagrams for the whole system. They have been split into different segments representing different aspects of system operation.

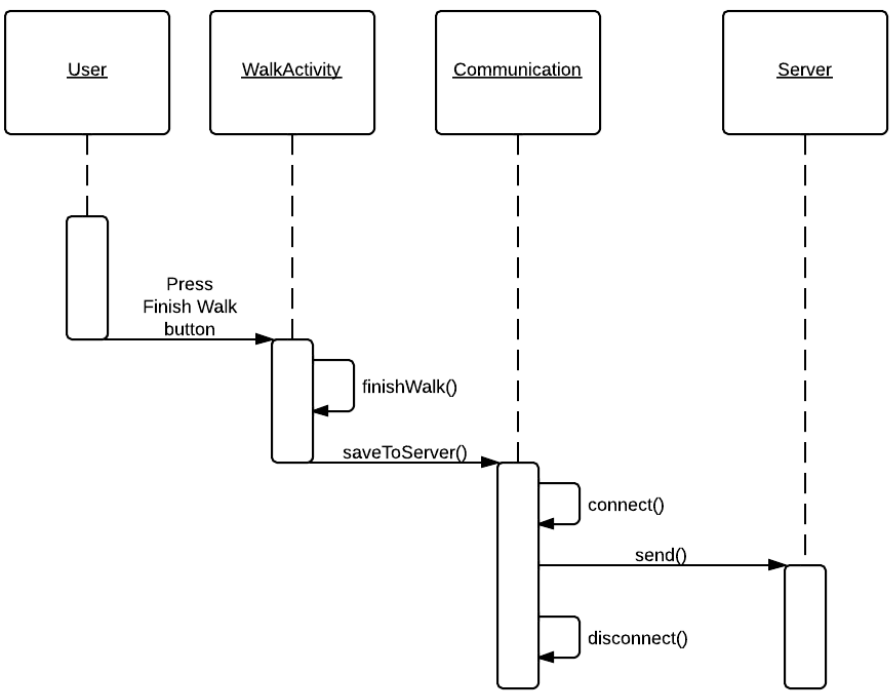
5.1.1. Tour creation sequence diagram



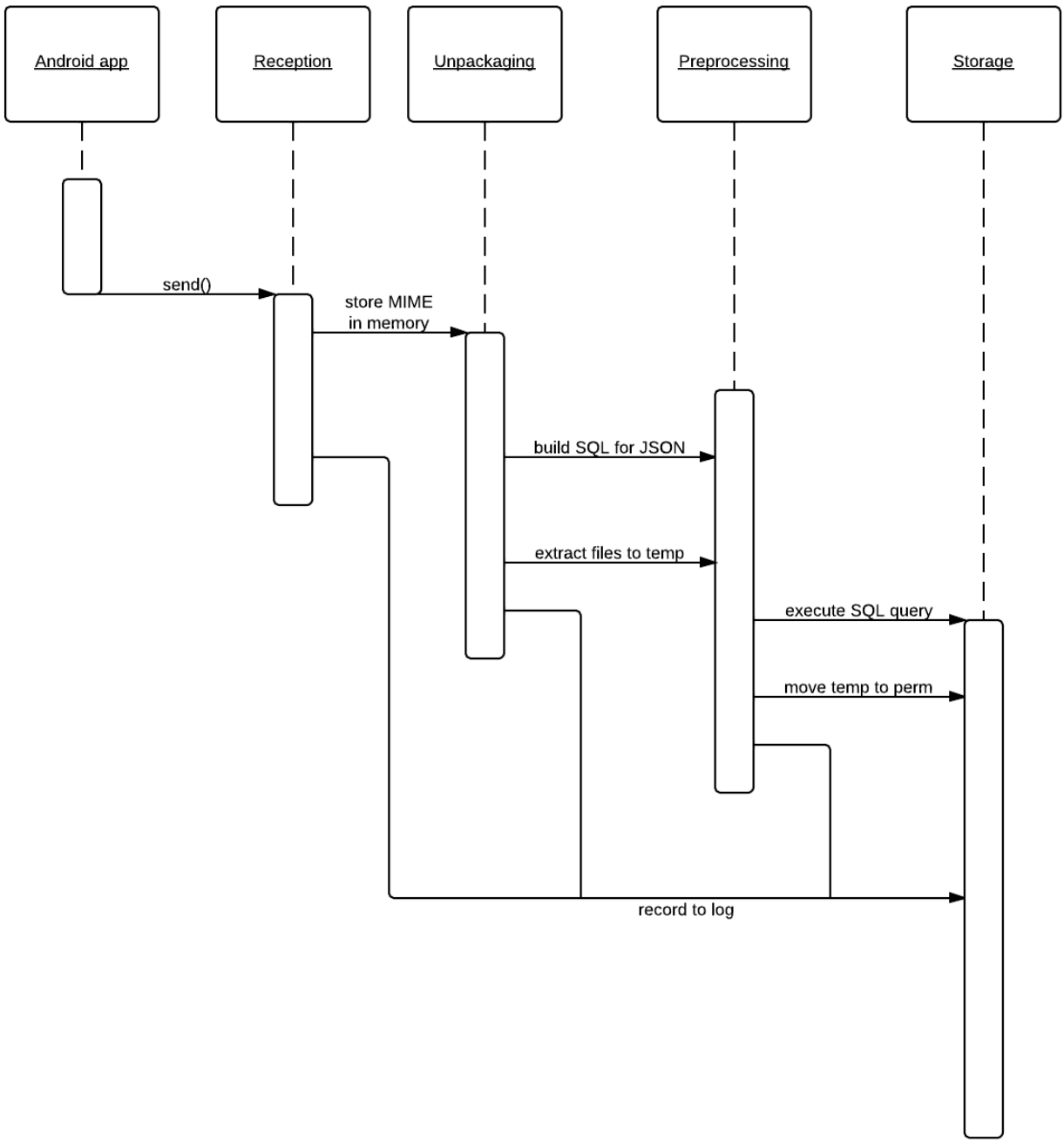
5.1.2. Add key location sequence diagram



5.1.3. Data transfer sequence diagram



5.1.4. Server side sequence diagram



5.2. Algorithm Description

5.2.1. Server Side Data Delivery Algorithm

- Android app uploads MIME message to a PHP server page over the internet using the HTTP POST method;
- 2. The PHP file unpacks the image attachments and saves them to a temporary directory with a unique name (e.g. tmp/05012013my_tour). It records success/ failure to the server log and if successful, will progress;
- 3. The PHP file extracts the data from the JSON file, checks whether the data is valid and continues;
- 4. It then produces an SQL query to insert the data into the database. It records success/ failure to the server log and if successful, will progress;
- The files are moved from the tmp directory to a permanent directory with a unique name. (The primary key of the tour in the database).

5.2.2. Android side algorithms

- **Generating JSON:**

All relevant classes will have a toJSON() method that constructs a JSON object based on the contents of the class (Appendix B). This includes any objects constructed by the class itself. Any photographs that are going to use have their file path added as a component of the generated JSON objects.

- **Communicating with the server:**

The created JSON objects are sent as part of a MIME message via HTTP Post.
In Android the sending of information via HTTP Post is rather simple, we create an object known as a HTTP Post object, with a URL attached to it, we then add all associated information necessary and send it to the URL previously attached.

- **Keeping track of locations:**

To keep track of the route the user is taking we are going to generate a location every few meters determined by a persistent algorithm checking the difference in the longitude/latitude and calculating if the difference equates to or is greater than the prescribed difference for adding a location. If the user adds a location within 5 minutes of the app generating one the app generated one will be removed to reduce on data transfer for the user.

- **Location management:**

As stated above the previous location will be checked on the creation of a key location by the user if the user-generated location's timestamp more than 5 minutes older than the previous location, just add the location to the end of the linked list; however if the previous non-user created location is less than 5 minutes old, replace it with the new user-generated location.

- **Photo management:**

The android operating system allows us to simply store our own images that the user will create in app in our own file storage system under the images folder on the device. This will allow us to easily attach the images for the relevant key locations while being able to just reference a file path in the JSON string.

5.3. Data Structures

5.3.1. Android data structures

The data structure for the application is essentially one class containing a linked list of objects of another class. More specifically it has a 'Tour' class which holds the linked list of 'Location' objects. The Tour class contains fields for the tour's name, short description and long description; along with a field which is the linked list. The Location class holds fields for the latitude and longitude of the location; it also has a timestamp for when that location was recorded. There is also a 'KeyLocation' class which contains further information. More specifically it contains an array of file paths to the photos the user takes for that location, a name for that location and a short description for that location.

5.3.2. Server side data structures

The tables in the database that will hold all the data for the walk:

- *List of Walks Table:*

Field Name	Field Description	Field Data Format
id	Primary Key (auto increment)	integer
title	Title of the tour	text
shortDesc	A short description of the tour (<100 characters)	text
longDesc	A detailed description of the tour. (<1000 characters)	text
hours	The number of hours the walk will take	float
distance	The total distance of the tour in kilometres	float

- *Location Table :*

Field Name	Field Description	Field Data Format
id	Primary Key (auto increment)	integer
walkID	Foreign key, referencing the id field of the tour that the location is associated with	integer
latitude	The latitude map reference for the location	float
longitude	A detailed description of the tour.	text
timestamp	The time in hours from the beginning of the tour	float

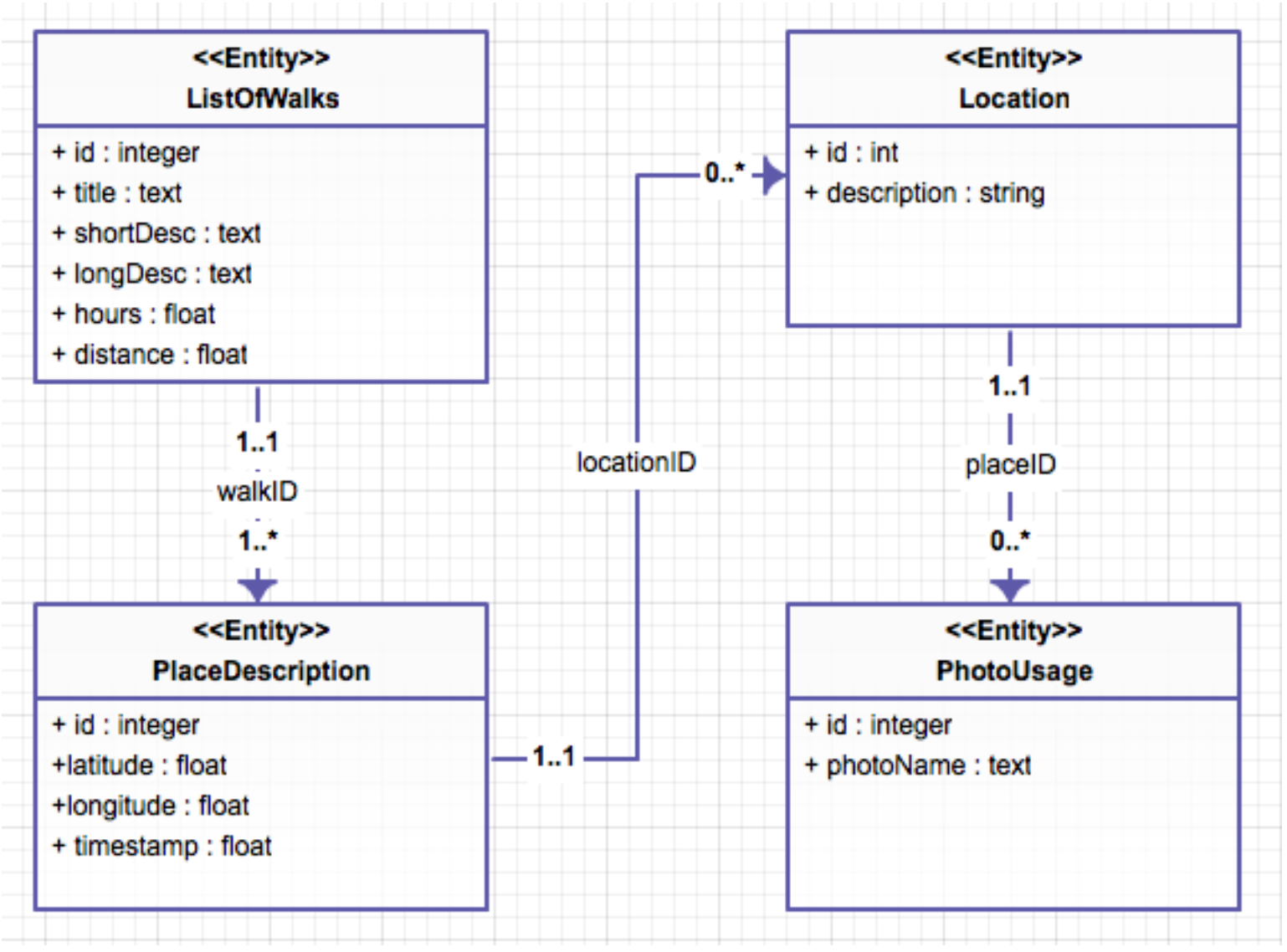
- *Place description table:*
-

Field Name	Field Description	Field Data Format
id	Primary Key (auto increment)	integer
locationID	Foreign key, referencing the location that the point of interest is referencing	integer
description	The description of this point of interest (<500 characters)	text

- *Photo Usage Table:*

Field Name	Field Description	Field Data Format
id	Primary Key (auto increment)	integer
placeID	Foreign key, referencing the point of interest that the image is attached to	integer
photoName	The name of the jpg file for the photo (without ".jpg" suffix)	text

5.4. Entity Relationship Diagram



6. APPENDIX A

The mime message will contain a “From” field which will store the user’s name and email (From: User’s Name <user@usershost.com>) and the name of the tour in the “Subject” field (Subject: My Tour). Writing the tour name to the subject field will allow the server to record the process in the log, even if there is an error with the JSON code. It will include a MIME version declaration of version 1.0 (MIME-Version: 1.0) and a multipart content type declaration (Content-Type: multipart/mixed; boundary=”part”). The JSON code will be stored in the only text type part. All of the images will be stored as attachments in jpeg format.

Sample MIME message:

From: John Doe <example@example.com>
Subject: TOUR NAME
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="part"
--part
Content-Type: text/plain

JSON CODE GOES HERE

--part
Content-Type: image/jpeg;
Content-Disposition: attachment;
filename="file1.jpg"
jgfc,jbjytf,nmvk-0987y6trfgi9876trdfvbhjytrfdc
--part—

7. APPENDIX B

- *Fields for root of JSON data set:*

Variable Name	Description	Format
title	The title of the walk	A string of <30 characters
shortDesc	A short description of the tour to be displayed in lists of tours on the website.	A string of <100 characters
longDesc	A long description of the tour to be displayed alongside the map on the website.	A string of <1000 characters
route	A sequence of GPS locations that describe the route of the tour	A collection of objects representing GPS coordinates. (See List of walks Table)
locations	A set of locations of interest along the tour.	A collection of objects representing locations of interest. (See Location Table)
time	The number of seconds that elapsed during the recording of the tour. (Not including when paused)	Integer
distance	The distance of the route of the tour in meters.	Integer

- *Fields for location/ route objects:*

Variable Name	Description	Format
id	A unique ID indicating the index of the location in the sequence	Integer
longitude	The longitude of the current GPS location on the route	Integer
latitude	The latitude of the current GPS location on the route	Integer
time	The number of seconds that elapsed from the be-ginning of the tour to this recorded location. (Not including when paused)	Integer

- *Fields for POI object:*

Variable Name	Description	Format
coord	The ID of the GPS coordinate object that the location is attached to	integer
description	A short description of the current location.	A string of <500 characters
media	A set of URLs pointing to the images to be associated with the location	A string collection of variable length.

8. APPENDIX C

Sample JSON file:


```
{
	"Title": "My Walk",
	"shortDesc": "A walk from grans house to my house",
	"longDesc": "This is a walk that I take from my house to my nans. I hope you enjoy it…",
	"route": [
{
	"id": 0,
	"longitude": 345674,
	"latitude": 583848,
	"time": 0
},
{
	"id": 1,
	"longitude": 345684,
	"latitude": 583848,
	"time": 5
},
```


//LOTS MORE HERE…


```
],
	"pointOfInts": [
{
	"coord": 5,
	"description": "This is where I live",
	"media": [
	"file1.jpg",
	"file2.jpg"
]
},
{
	"coord": 17,
	"description": "This is about half way",
	"media": []
},
{
	"coord": 25,
	"description": "This is where my gran lives",
	"media": [
	"file3.jpg"
]
}
],
	"time": 45676,
	"distance": 23454
}
```

9. References

[1] Software Engineering Group 05. Project Plan. S. Raychev, B. O'Donovan, H. Clark, W. Arslett, W. Lea, N. Vicker and S. Clasby. 1.2 Release.

[2] Software Engineering Group 05. Test Specification. S. Raychev, B. O'Donovan, H. Clark, W. Arslett, W. Lea, N. Vicker and S. Clasby. 1.1 Release.

[3] Software Engineering Group Projects. General Document Standards. C. J. Price, N. W. Hardy and B. P. Tiddeman. SE.QA.03. . 1.6 Release.

[4] Software Engineering Group Projects. Design Specification Standards. . C. J. Price, N. W. Hardy and B. P. Tiddeman. SE.QA.05A. . 1.7 Release.

[5] Software Engineering Group Projects. Java Coding Standards. . C. J. Price, A. McManus. SE.QA.09. . 1.7 Release.

10. Document History

Version	CFF No.	Date	Changes made to the document	Changed by
1.0	N/A	02.12.13	N/A – First release of the Design Specification	srr11
1.1	N/A	05.12.13	Updated Diagrams & Tables; Added algorithms	srr11
1.2	N/A	06.12.13	Finalised the design specification	srr11
2.0	#1	13.02.14	Updated with Bernie's remarks.	bmo

8. References

9. Document history

Version	CFF No.	Date	Changes made to the document	Changed by
1.0	N/A	17/02/2014	N/A – First release of the Final report	bmo, srr11