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

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

The launch activity of the app, from here you can go to settings, start a walk and go to app details.

2.2.2. Tour class

Tour is the main storage class of the application and will hold a linked list of locations for the walking tour. It will contain variables as follows:

- a string for the location of the tour e.g. “Aberystwyth”;
- a string for the long description of the tour that will have no more than 1000 characters;
- a string for the short description/summary of the tour that is limited to 100 characters;

2.2.3. Location class

The location class represents the locations added automatically along the walk and consists of:

- a longitude and latitude for the location;
- a name of the location where the user is;
- a time stamp that can be used to calculate a total time for the walk;

The Location class is extended by the KeyLocation class, which represents the points of interest, the user can add and contains also:

- a short description of the location limited to 140 characters;
- an array of photos taken along the walk with a maximum of 5;

2.2.4. Communication class

This is the class that handles the communication to the server. It will implement the Link interface.

2.2.5. Walk activity class

This is the class that links the model and the user interface together. From here the user will add locations to the tour with descriptions and a set of up to 5 photos. The user will be able to set the sample frequency (how often a coordinate should be placed automatically on the map).

2.2.6. Link interface

This is the interface for communication between the app and the server. It contains the methods for connecting and disconnecting to the server and sending data.

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:

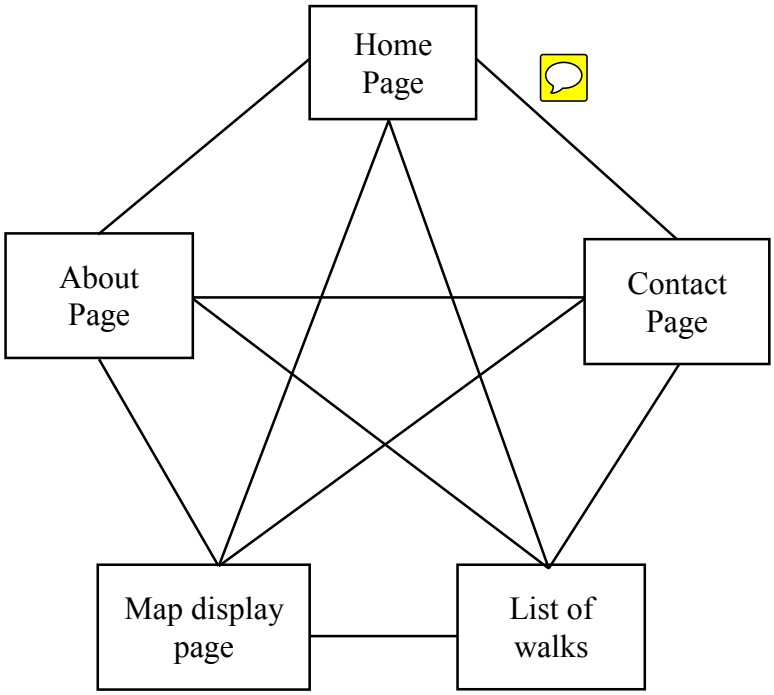
Table Name	Table Description
ListOfWalks	A list of walks/ tours, which the program will display.
Location	A list of geographical locations referencing a record in the tour table, describing the route of the tour as a sequence of locations.
PlaceDescription	A list of points of interest along tours referencing 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 block then a three smaller ones underneath side by side which will hold the most used routes. The page will include the database connection script that will be in the inc folder.

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 along with the time it takes to complete the walk. 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. There will also be a content block with information on the API and how to use the facility.

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.

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.

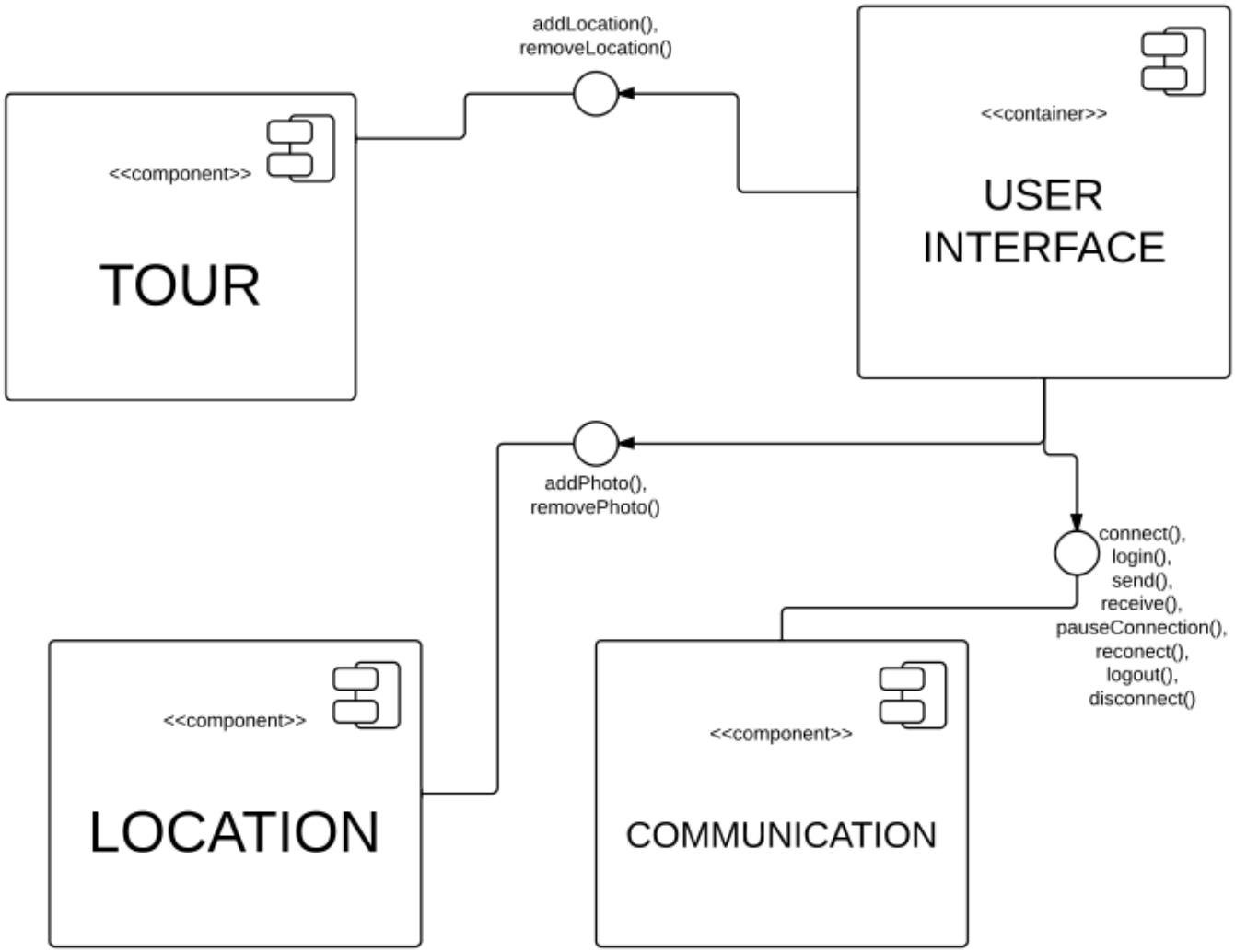
2.4. Table mapping requirements

Requirement	Modules providing requirement
FR1	MainActivityClass
FR2	TourClass, ListOfWalksTable
FR3	TourClass, WalkActivityClass, LocationsTable
FR4	KeyLocationClass, LocationClass, PhotoUsageTable
FR5	WalkActivityClass
FR6	CommunicationClass, LinkInterface, Upload.php
FR7	WalkActivityClass
FR8	Index.php, Map.php, List.php
FR9	Upload.php

3. Dependency Description

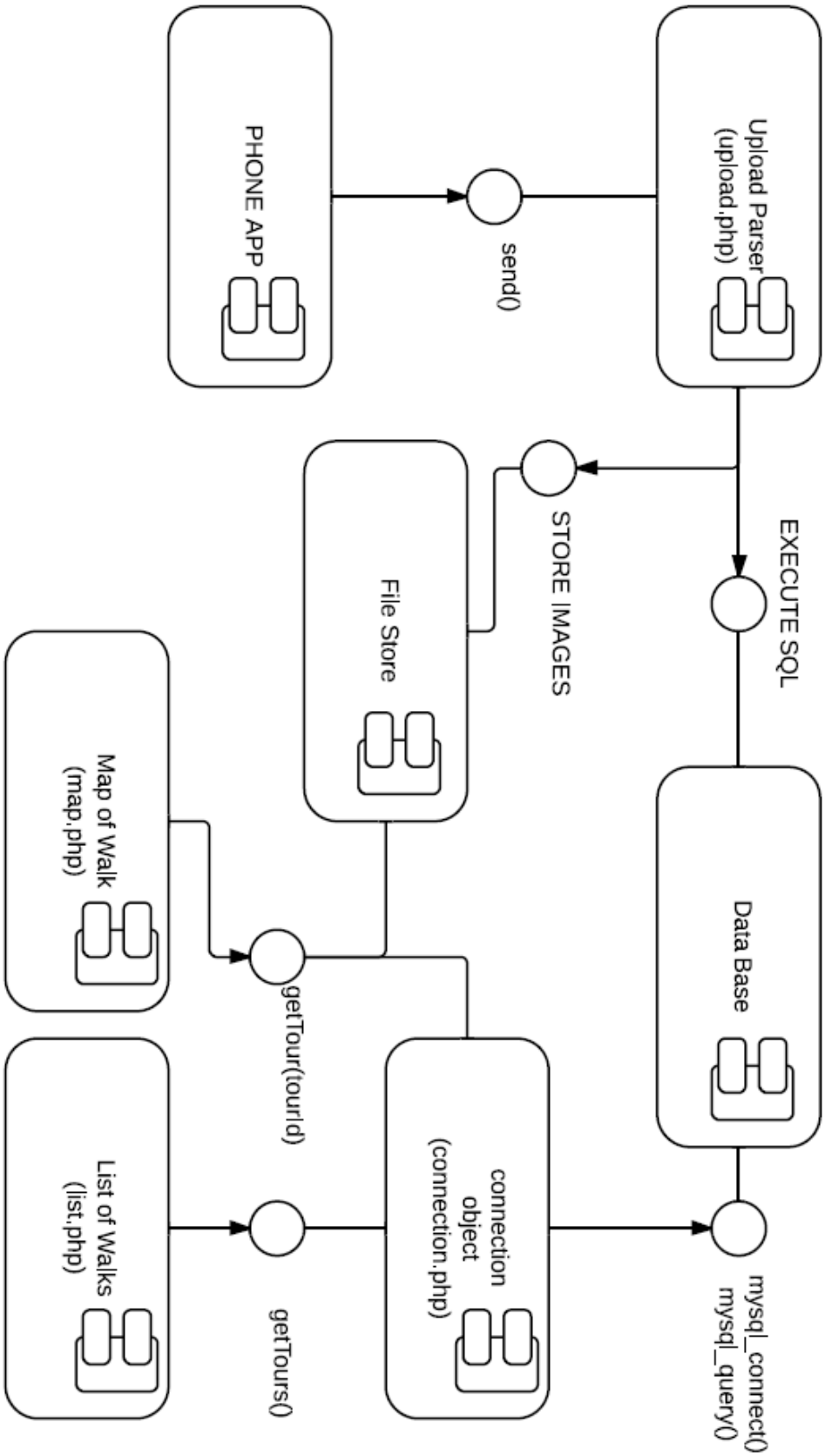
3.1. UML Component Diagram for the Android application

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.

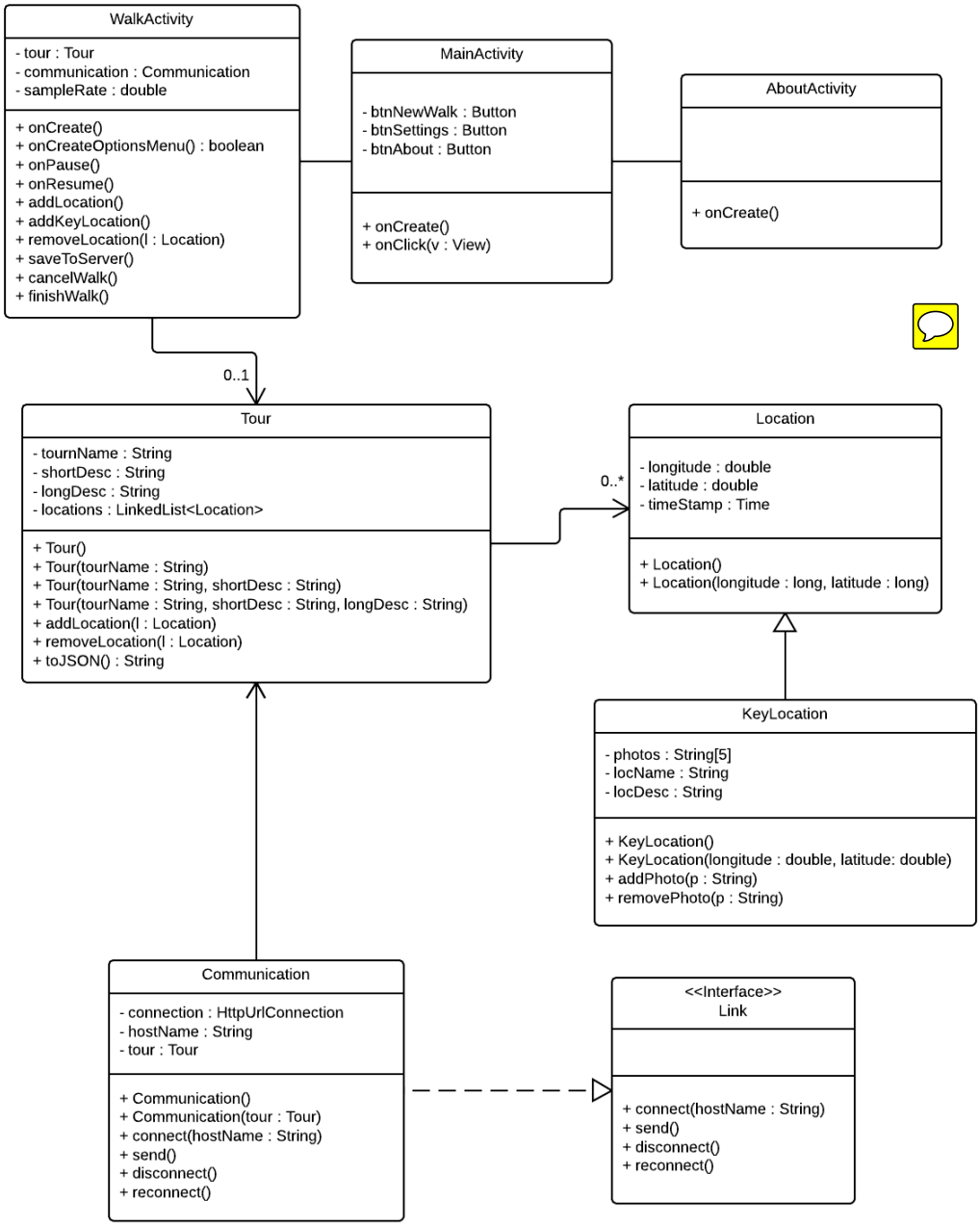


4. Interface description

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

```
package com.wtc.grp5;

public class MainActivity extends Activity implements OnClickListener{

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

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


}
```

4.3. Tour class

```
package com.wtc.grp5.model;

public class Tour {

        /**
        * Constructs a blank tour.
        */
        public Tour(){
        }

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

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

/**
* Removes a location from the tour.
*
* @param location the location being removed.
*/
public void removeLocation(Location location){
}

/**
* 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<Location> locations){
        }

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

}
```

4.4. Location class

```
package com.wtc.grp5.model;

public class Location {

        /**
         * Constructs a blank location.
         */
        public Location(){
        }

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

        /**
         * @return the time stamp for when this location was recorded
         */
        public String getTimeStamp(){
        }

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

4.5. Key location class

```
package com.wtc.grp5.model;

public class KeyLocation extends Location {


        /**
         * Constructs a blank key location.
         */
        public KeyLocation(){
        }

        /**
         * 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 KeyLocation(long longitude, long 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){
        }
}
```

4.6. Communication class

```
package com.wtc.grp5.model;

public class Communication implements Link {


        /**
         * Constructs a blank Communication object.
         */
        public Communication(){
        }

        /**
         * Constructs a Communication object with a specified Tour object.
         *
         * @param tour the object to sent to the server.
         */
        public Communication(Tour tour){
        }

        @Override
        public void connect(String hostName){
        }

        @Override
        public void send(){
        }

        @Override
        public void disconnect(){
        }

        @Override
        public void reconnect(){
        }

}
```

4.7. Link interface

```
public interface Link{

       /**
       * Connects to a server with a given IP address and port number.
       *
       *
       *@param ipAddr The IP address of the server you wish to connect to.
       *@param portNum The port number of the application that you are using.
       */
       public void connect(String hostName);

       /**
       * Sends data to the server.
       *
       *
       */
       public void send();

       /**
       * Receives data from the server.
       *
       */
       public void disconnect();
```

```
        /**
         * Re-opens a paused connection the server.
         *
         */
        public void reconnect();

}
```

4.8. Walk activity class

```
package com.wtc.grp5;

public class WalkActivity extends Activity {

        @Override
        protected void onCreate(Bundle savedInstanceState) {
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
        }

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


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

        /**
        * Removes a location from the tour.
        *
        * @param location the location being removed from the tour.
        */
        public void removeLocation(Location location){
        }

        /**
        * Saves the tour to the server.
        */
        public 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(Tour tour){
        }

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

        /**
        * @param communication the new value for this.communication.
        */
        public void setCommunication(Communication communication){
        }

        /**
        * @return the communication.
        */
        public Communication getCommunication(){
        }
}
```

4.9. About Activity

```
public class AboutActivity extends Activity {

        @Override
        protected void onCreate(Bundle savedInstanceState) {
        }
}
```

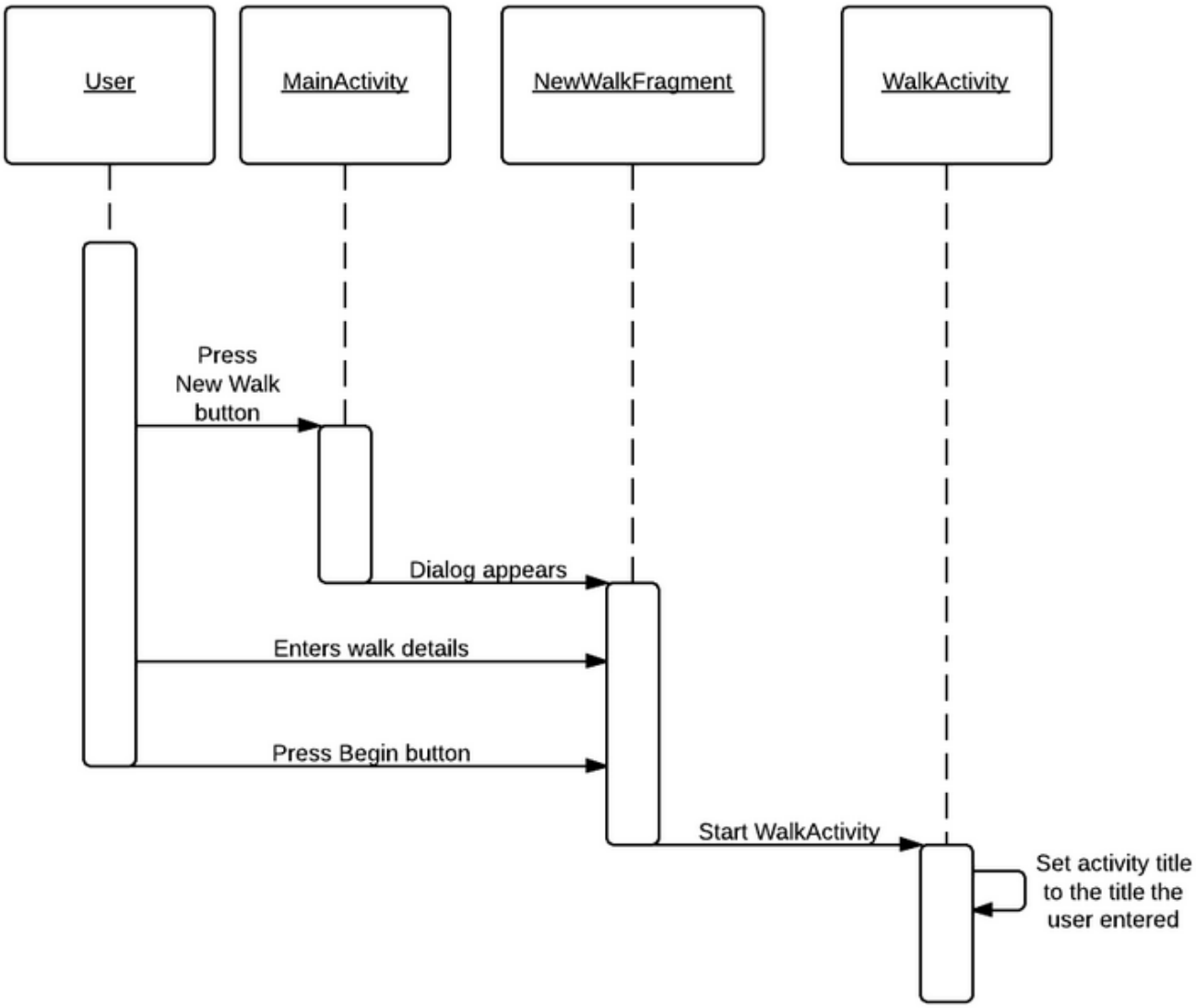
5. Detail Design

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.

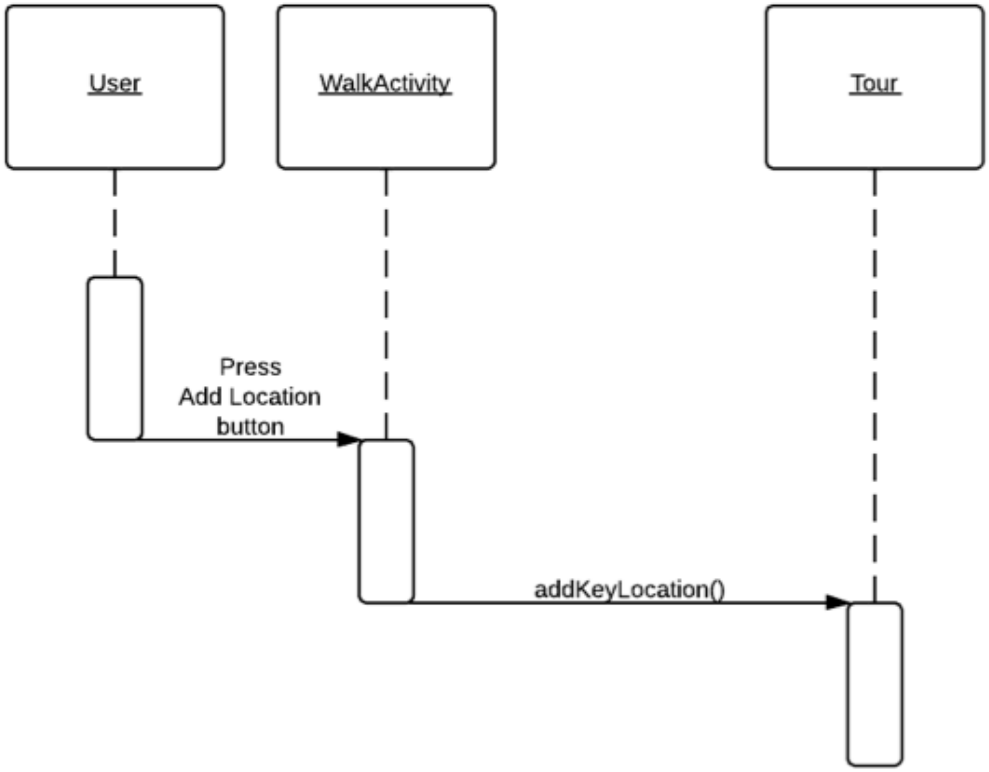
5.1. Sequence Diagrams

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

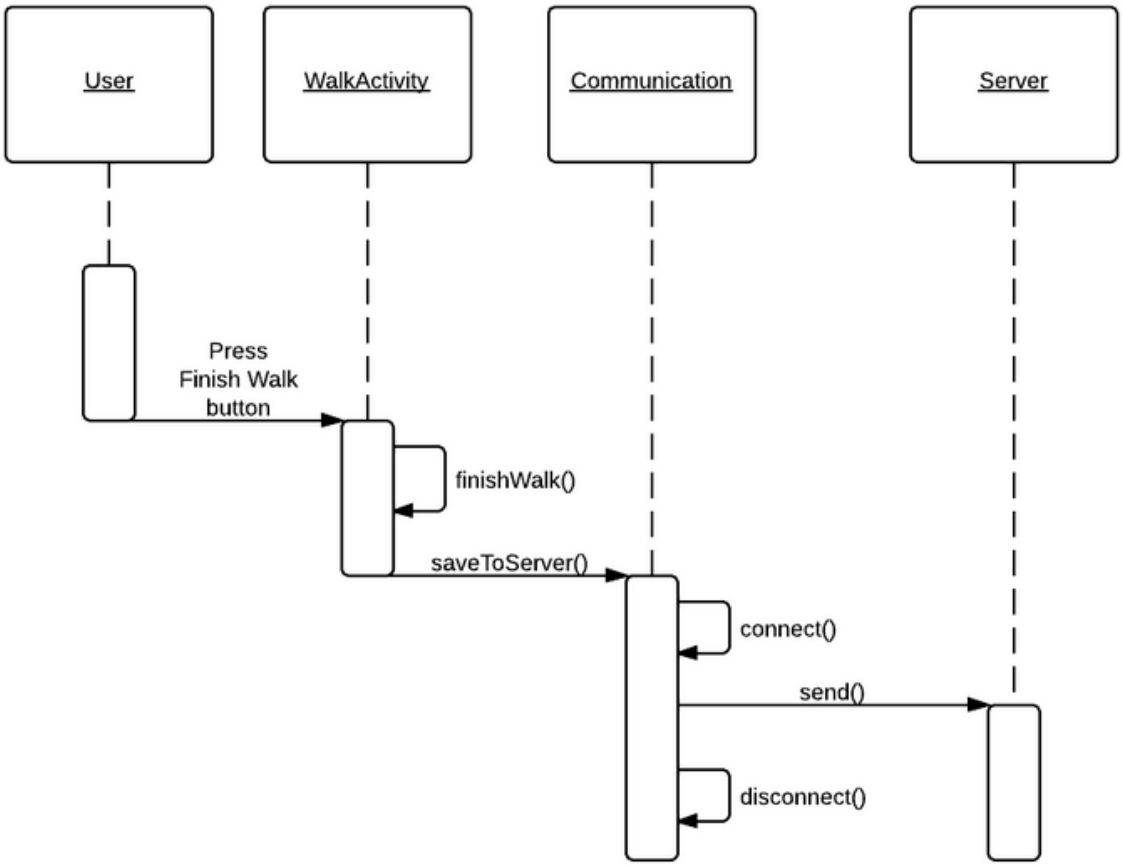
5.1.1. Tour creation sequence diagram



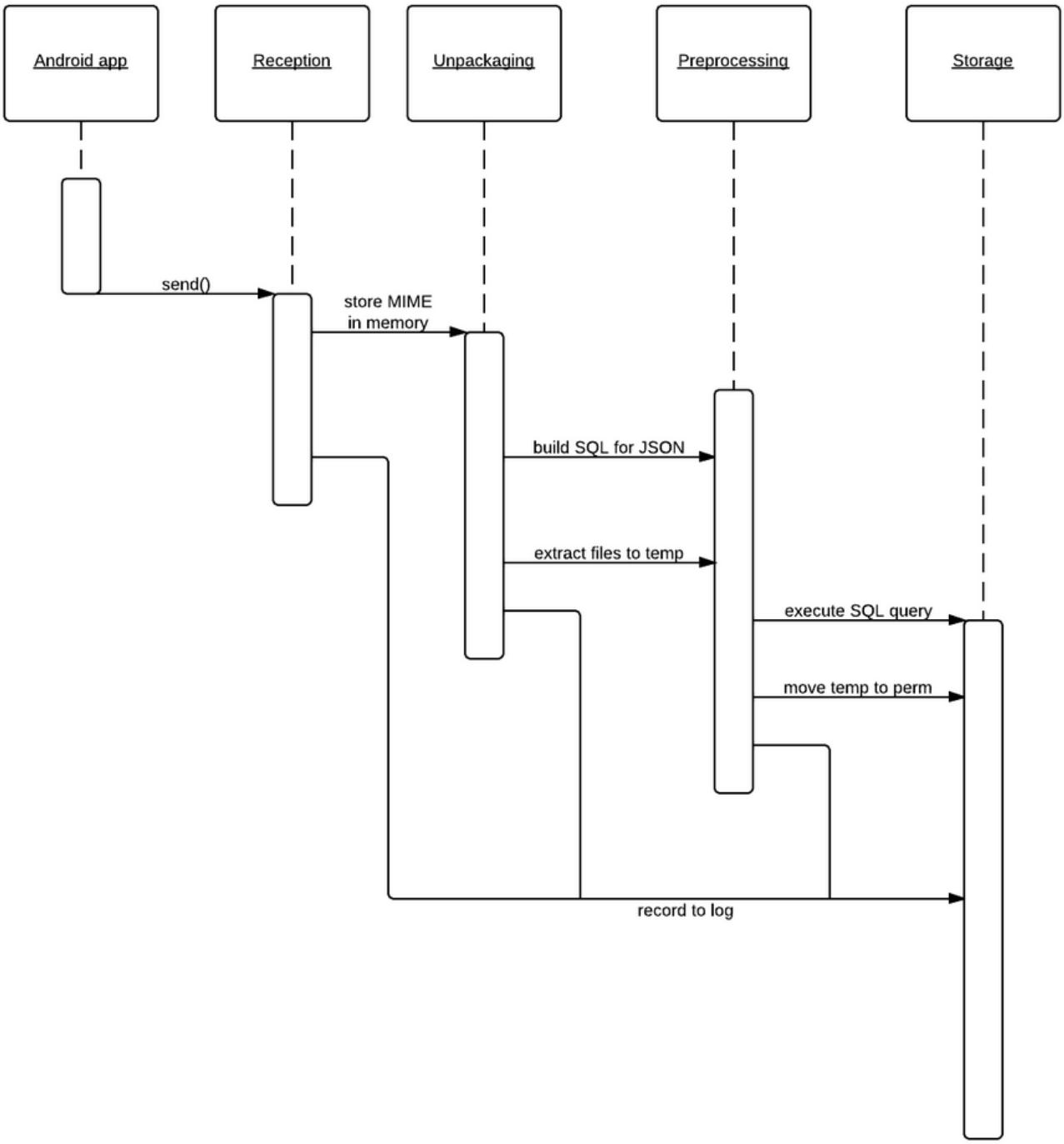
5.1.2. Add key location sequence diagram



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

1. 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;
5. 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

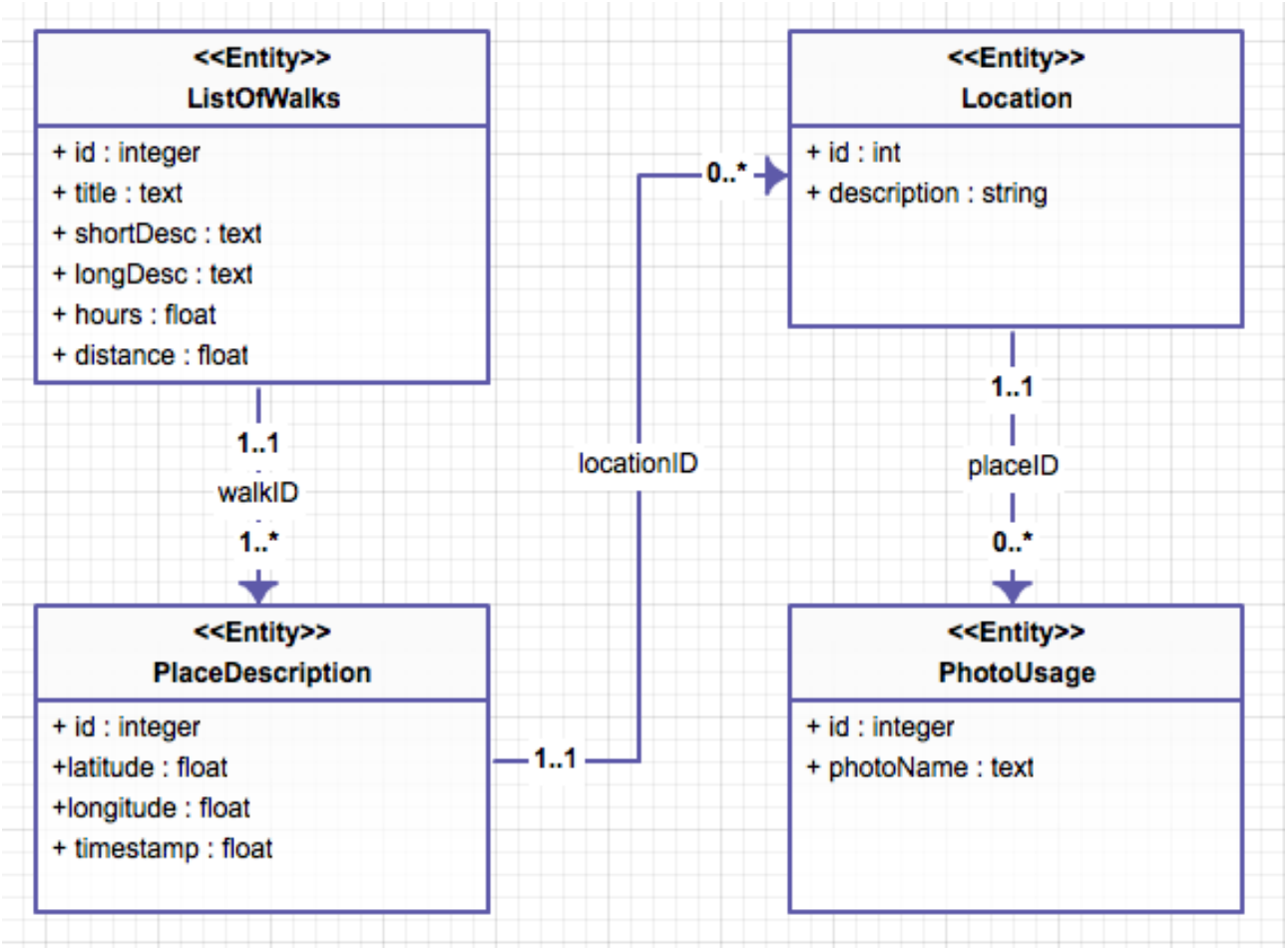
Place description table

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 beginning 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.

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
}
```

6. 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.

7. 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