Group Project 05

Design Specification

Authors: bmo; sr11; hac22; wia2; Department of Computer Science

wjl3; njv1 Aberystwyth University

Config Ref: SE\_05\_DS\_01 Aberystwyth, Ceredigion

Date: 14/11/13 SY23 3DB

Version: 1.0 Copyright © Group 05

Status: Draft Aberystwyth University 2013

Contents

[1. Introduction 3](#_Toc373244409)

[1.1. Purpose of this document 3](#_Toc373244410)

[1.2. Scope 3](#_Toc373244411)

[1.3. Objectives 3](#_Toc373244412)

[2. Architectural Description 4](#_Toc373244413)

[2.1. Android program 4](#_Toc373244414)

[2.2. Web program 5](#_Toc373244415)

[2.3. Table mapping requirements 6](#_Toc373244416)

[3. Dependency Description 6](#_Toc373244417)

[3.1. General architecture 6](#_Toc373244418)

[4. Interface Description 7](#_Toc373244419)

[5. Detail Design 8](#_Toc373244420)

[5.1. Sequence Diagrams 8](#_Toc373244421)

[5.2. Algorithm Description 8](#_Toc373244422)

[5.3. Data Structures 8](#_Toc373244423)

[5.4. ER 8](#_Toc373244424)

[6. References 9](#_Toc373244425)

[7. Document History 9](#_Toc373244426)

# Introduction

## Purpose of this document

The purpose of this document is to describe the outline design for the Walking Tour application. Taking into account the details of the group project assignment and group project quality assurance.

## Scope

This document includes detailed description of: classes used in the Android application; methods used in each class; sequence diagrams; interaction between the application and the database; database design and data handling.

## Objectives

The objective of this document is:

* To describe the main components of the Walking Tour Application
* To describe the main components of the Web site
* To depict the dependencies between the components

# Architectural Description

The Walking Tour System consists of two main components:

* The Android application;
* The Website;

## Android program

The android application will allow the user to create their walking tour and add photos and more detailed information about places of interest they come across during the walk. The walk can then be uploaded to the online database.

**Significant classes and methods:**

* **Tour class:**

1. Description:

The main storage class of the application. Tour 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;

1. Methods:
   * **Constructors:**
     + Default:   
       The default constructor will generate a blank tour object, with blank values (will be used for testing);
     + With name:  
       This is the most basic constructor that the user will have access to, all tours will need a name to progress, the values for long description and short   
       description will be left blank;
     + With name and short description:  
       Similar to the previous constructors, just without a blank short description;
     + With name, short description and long description:   
       Full constructor, consisting of all text fields filled for the walk;
   * **Add/Remove Location:**   
     Takes a Location and adds/removes it into the linked list of locations.

* **Location class:**

1. Description:

A key location within the tour. It consists of:

* + a longitude and latitude for the location;
  + a name of the location where the user is;
  + a short description of the location limited to 140 characters;
  + a time stamp that can be used to calculate a total time for the walk;
  + an array of photos taken along the walk with a maximum of 5;

1. Methods
   * **Constructors:** 
     + Default

Constructs a blank location with default values for the longitude and latitude, and a photograph(again for testing).

* + - With longitude and latitude

Constructs a location with longitude and latitude taken from the phones current location, will prompt the user if he/she wants to attach a photograph.

* + **Add/Remove Photo:**

Adds/removes a file path of a photograph to the array.

* **Communication class:**

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

* **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.

## Web program

The Website allows the user to view walks currently stored in the database using the mapping API.

## Table mapping requirements



How does the overall system fit together? What are the pieces?

Significant classes description for each program.

# Dependency Description

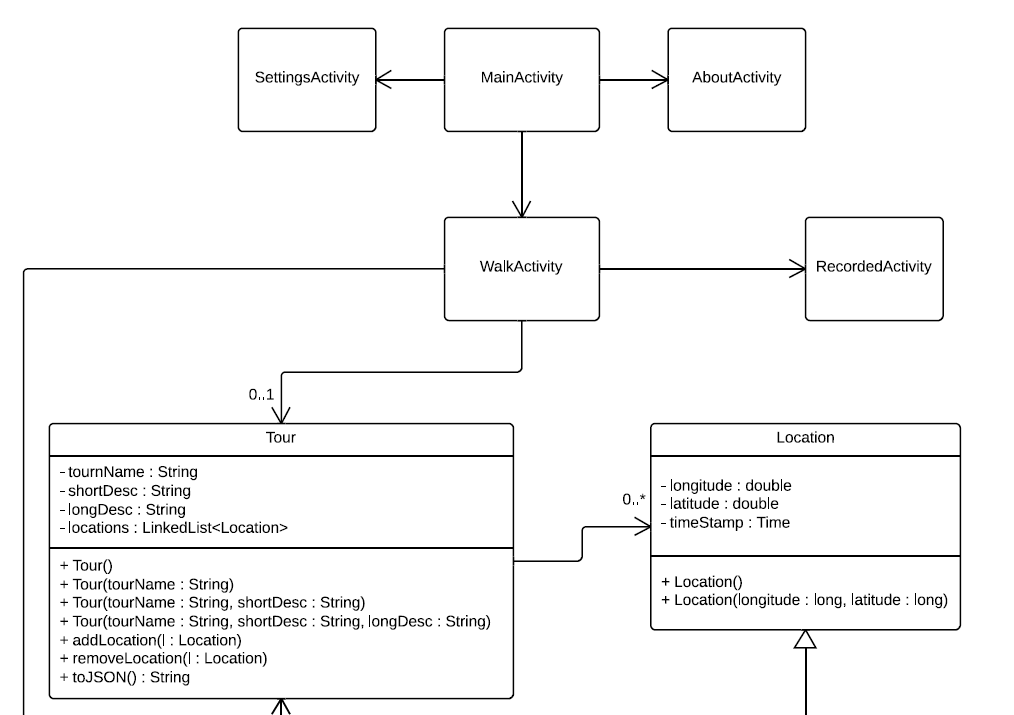
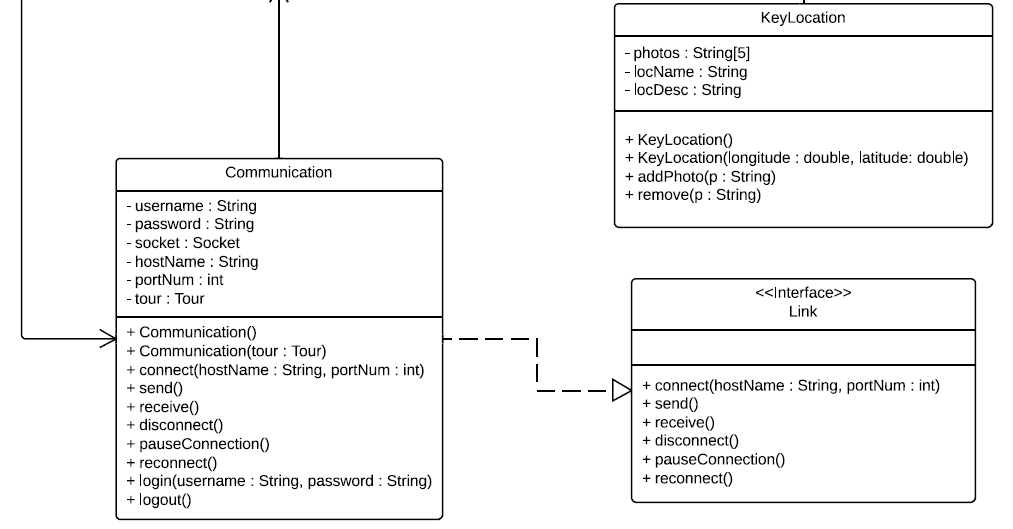
## Component Diagrams



## General architecture

* Android:

MAGIC/discussion tomorrow!



* Web:

## **POST Request:**

All the data will be packed into a single string, which will be transmitted from the android phone in a HTTP POST request. The value will be paired using the key “message” and will be accessed in PHP using the $\_POST[‘message’] handle.

The request will be made to the file upload.php which will be 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.

## **MIME message:**

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—

## **JSON code design:**

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.

Table 6 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 Table 2) |
| locations | A set of locations of interest along the tour. | A collection of objects representing locations of interest. (See Table 3) |
| 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 |

Table 7 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 |

Table 8 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 format:**

{

"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

}

The following tables represent the structure of the database relations:

Table : 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 : 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 : 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 : 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) | integer |

Table : 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 |

How do the pieces interact with each other?

# Interface Description

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();

}

What is the way each piece allows others to talk to it?

Should provide everything designers, programmers and testers need to know to use the facilities provided by a module.

Interface specification for each class in the system should include:   
The name of the class also modifiers such as public or abstract.   
Classes which it extends (and why).

Public methods implemented by the class.

Properly formatted Java code with Java doc comments

Method bodies empty

# Detail Design

Significant algorithms / Significant data structures

From requirements, identify and write out main operations of system, including start-up/closedown operations (UML has USE CASES for this)

From operation descriptions, use sequence diagrams to show how each main operation would work by calling a sequence of different classes/ methods.

## Sequence Diagrams

[MUST CHANGE!!!]



## Algorithm Description

* Web side:

## **Data Delivery Algorithm**

1. Android app produces JSON text representation of tour objective data structure
2. Android app packages JSON text and images (as attachments) in a MIME message format
3. Android app uploads MIME message to a PHP server page over the internet using the HTTP POST method
4. The PHP file unpacks the image attachments and saves them to a temporary directory with a unique name (ie tmp/05012013my\_tour). It records success/ failure to the server log and if successful, will progress to step 5.
5. The PHP file extracts the JSON code and produces an SQL query to insert the data into the database. It records success/ failure to the server log and if successful, will progress to step 6.
6. 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)

* Android:

## Data Structures

## ER



Especially where classes represent real world objects, there is often a complex data structure in the program made by objects pointing to each other

Readers of the design needs to get a feel for that structure, and nothing we have documented so far gives that feel   
Class interaction diagrams and object interaction diagrams in UML can do this, but often, just drawing the relationships between a set of objects is enough.

# References

# 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 |
|  |  |  |  |  |