Software Engineering Group Project
Design Specification

Author: Group 14
Config. Ref.: SE-14-DESIGN
Date: 2013-12-05
Version: 2.0
Status: Final

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

CONTENTS

1 INTRODUCTION

1.1 Purpose of this Document

The purpose of this document is to describe the design of the Walking Tour Creator and the Walking Tour Displayer applications. It should be read in the context of the Group Project, taking into account the details of the group project assignment and the group project quality assurance (QA) plan. [2]

1.2 Scope

The design specification gives a detailed explanation of how the individual components of the Walking Tour Creator and the Walking Tour Displayer applications should be implemented.

This document should be read by the Android application development team and the website development team.

1.3 Objectives

The main objective of this document are:

- To describe the main components of the Walking Tour Creator and the Walking Tour Displayer
- To show the dependancies between the components and the communication between the application and web server
- To provide interface details for each of the main classes in the Walking Tour Creator

2 DECOMPOSITION DESCRIPTION

2.1 Programs in system

The system is composed of two programs:

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

2.1.1 Walking Tour Creator

The Walking Tour Creator is an application to allow the user can create walks on a mobile device for them to be uploaded to the server. It implements the requirements (FR1), (FR2), (FR3), (FR4), (FR5), (FR6), (FR7), (FR9), (PR1) and must conform with the requirements (EIR1), (PR2), (DC1), (DC2).[1]

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

2.1.2 Walking Tour Displayer

The Walking Tour Displayer is website for viewing walks created by users via the Walking Tour Creator. It implements the requirements (FR8), (FR9), (PR1) and must conform to the requirements (EIR1), (PR2), (DC1), (DC2), (DC3).[1]

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

2.2 Significant classes in each program

2.2.1 Significant classes in Walking Tour Creator

MainAppActivity. This is the main class of the application.

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

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

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

*JsonPackager.*Changes the data taken from the app into Json to transfer to the website.

2.2.2 Significant functions in Walking Tour Displayer

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

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

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

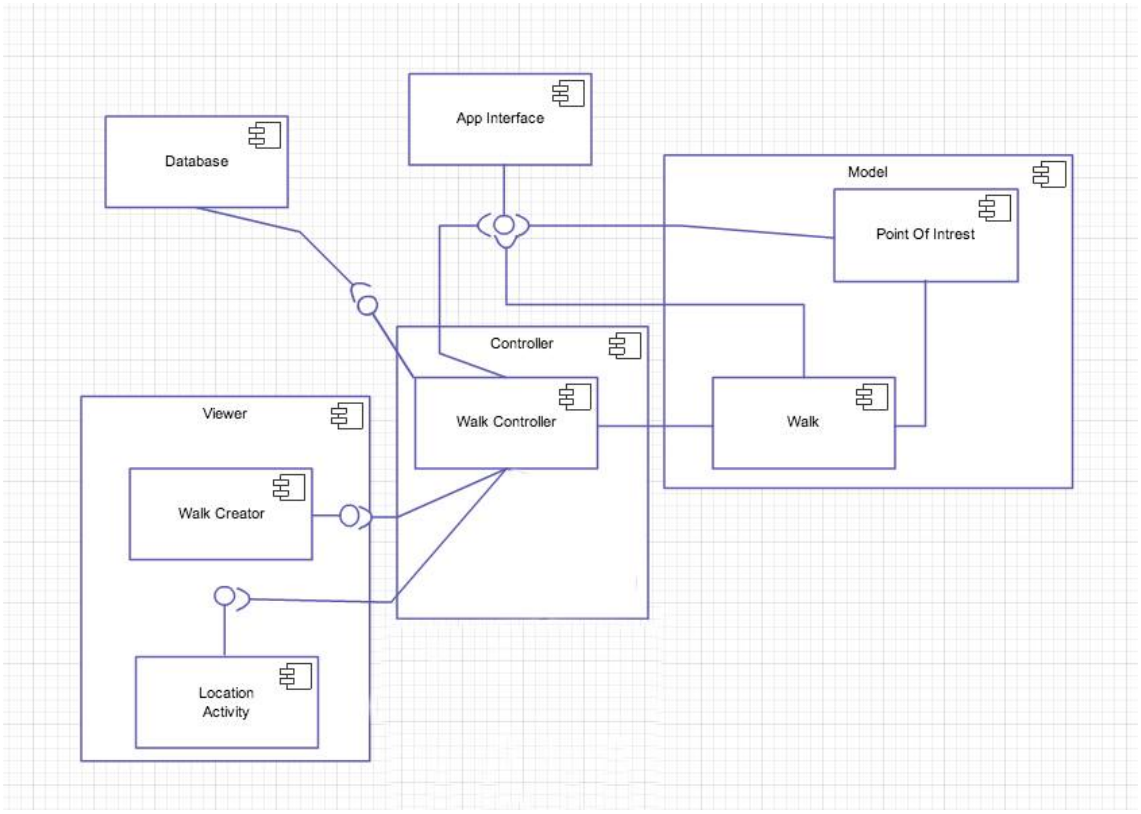
2.3 Mapping requirements onto classes

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

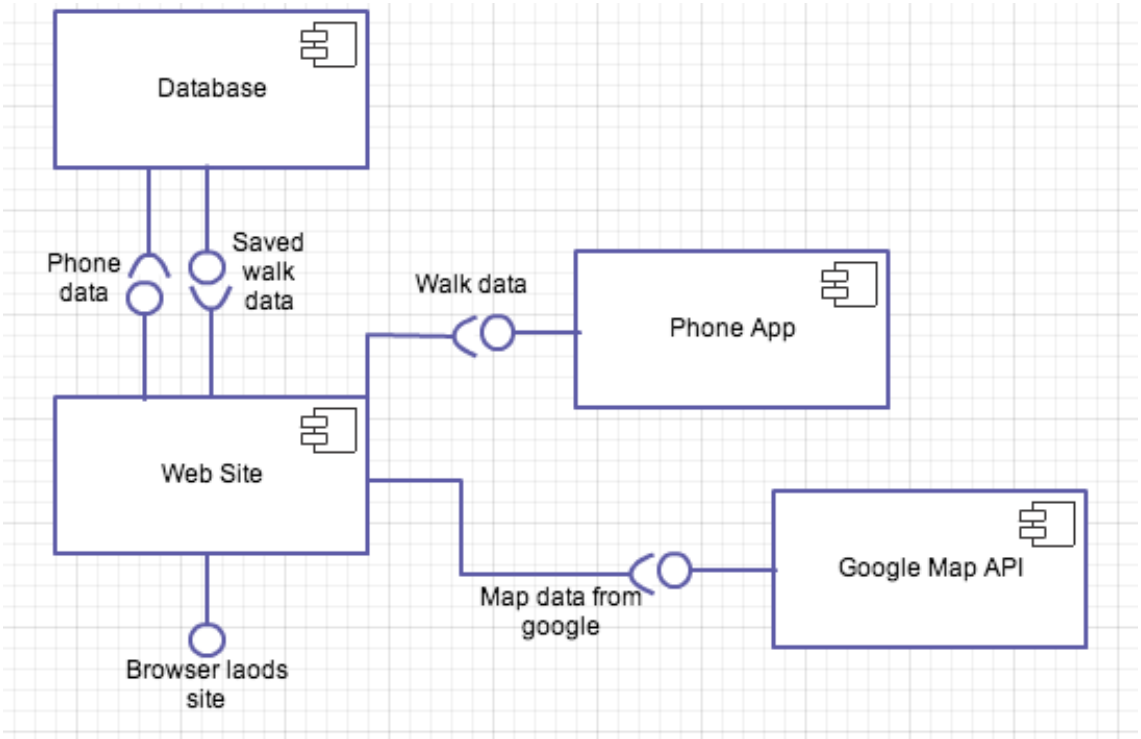
3 DEPENDENCY DESCRIPTION

3.1 Component Diagrams

3.1.1 Component Diagram for Walking Tour Creator



3.1.2 Component Diagram for Walking Tour Displayer



3.2 Inheritance Relationships

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

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

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

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

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

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

4 INTERFACE DESCRIPTION

4.1 MainAppActivity interface specification

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

4.2 WalkDetailsActivity interface specification

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


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

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

```
public void setName(String name);

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

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

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

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

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

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

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

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

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

}
}
```

4.3 Json interface specification

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

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

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

}
```

4.4 Walk interface specification

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


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

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

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

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

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

```
public void setShortDescription(String desc);

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

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

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

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

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

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

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

```
    public void addLocation(Location location);

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

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

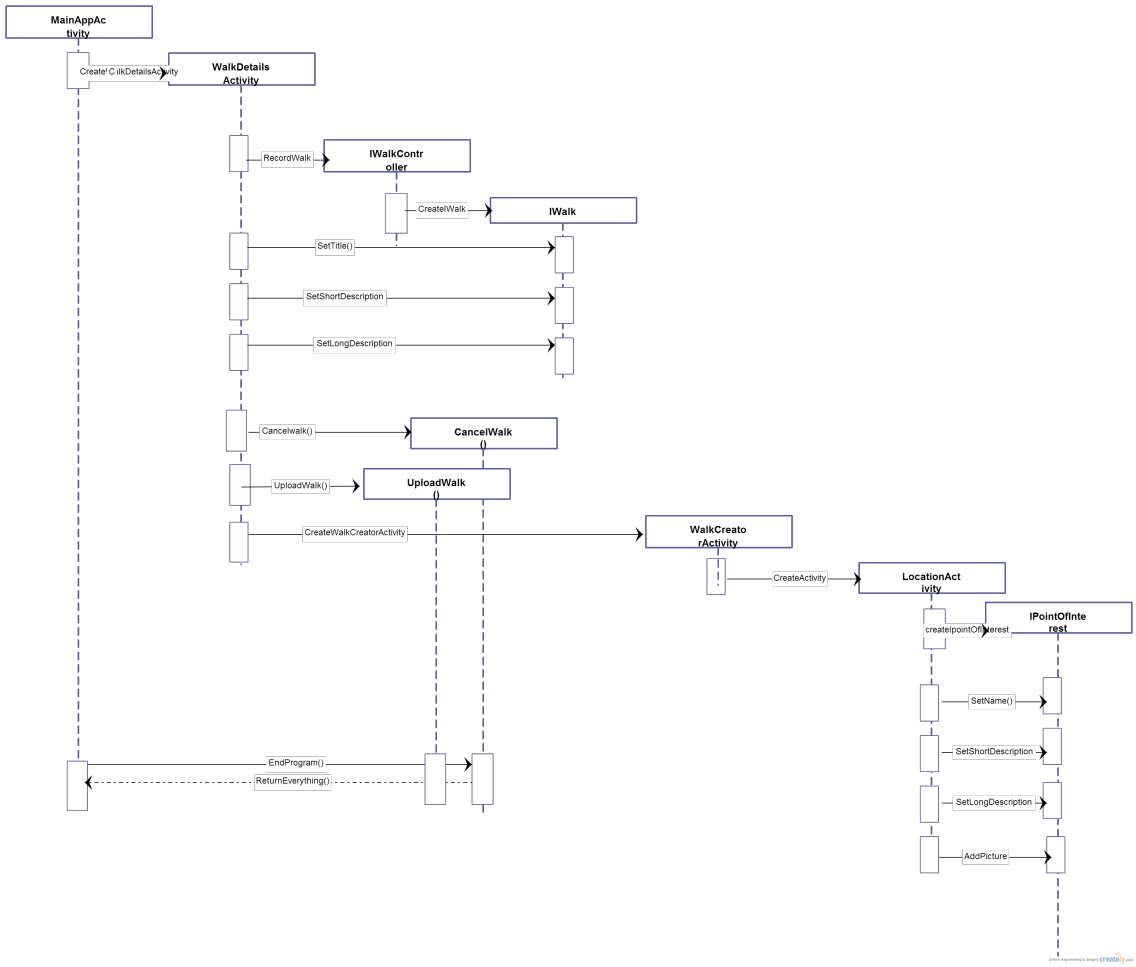
4.5 PointOfInterest interface specification

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

import android.os.Parcelable;

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

   /**
    * This method is used to set the IPointOfInterest's name
    * @param name String representing the new name of the IPointOfInterest
    */
   public void setName(String name);

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

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

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

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

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

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

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

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

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

}
```

5 DETAILED DESIGN

5.1 Sequence diagrams



5.2 Significant algorithms

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

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

5.3 Significant data structures

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

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

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

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

REFERENCES

[1] *Software Engineering Group Projects* Requirements Specification.
C. J. Price and B.P.Tiddeman, SE.QA.RS. 1.4 Release.

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

DOCUMENT HISTORY

Version	CCF No.	Date	Changes made to Document	Changed by
0.1	N/A	2013-12-04	Initial creation	tht5, jam66, meo9
0.2	N/A	2013-12-04	Moved document to project template	jmt14
1.0	N/A	2013-12-04	Added remaining content	jmt14, tht5, jam66
1.1	N/A	2014-01-29	Updating and making changes to content	meo9, jam66
1.2	N/A	2014-02-13	Updating interface specifications and formatting.	meo9
2.0	N/A	2014-02-13	Final touch-ups.	meo9