Software Engineering Group Project
Design Specification

Author:	Group 14
Config. Ref.:	SE-14-DESIGN
Date:	2013-12-05
Version:	1.0
Status:	Released

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 comform 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 a photo. 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.

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 area.

LocationLoggerService. This is a service that runs in the background recording the user's walk.

2.2.2 Significant functions in Walking Tour Displayer

intialize. This JavaScript function creates an embeddded 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.

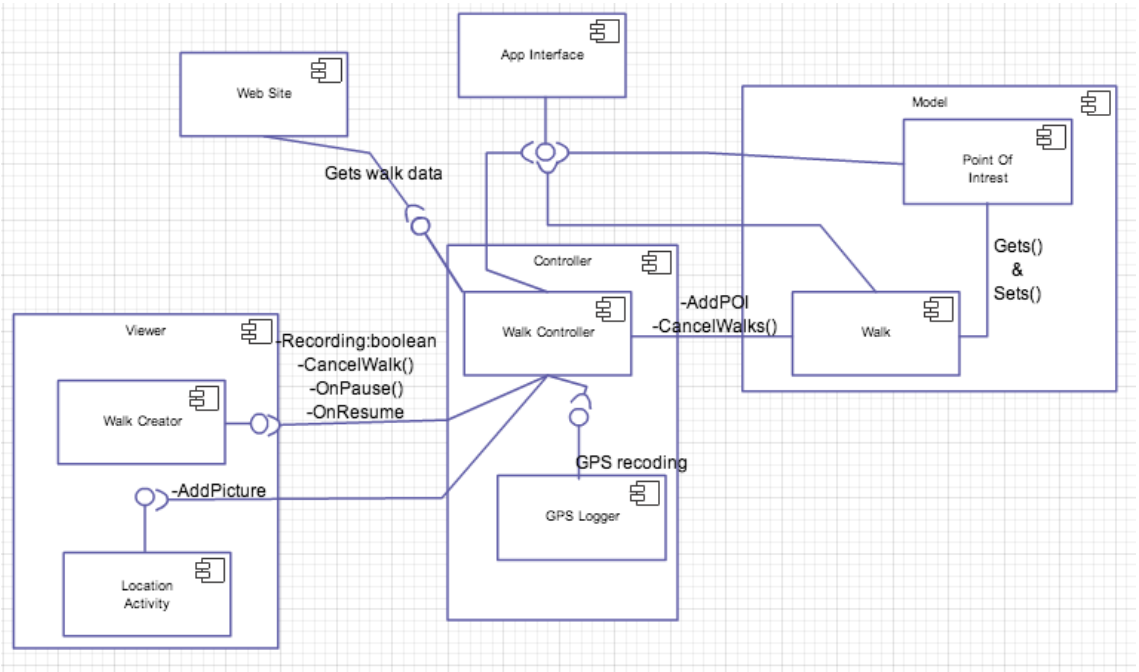
2.3 Mapping requirements onto classes

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

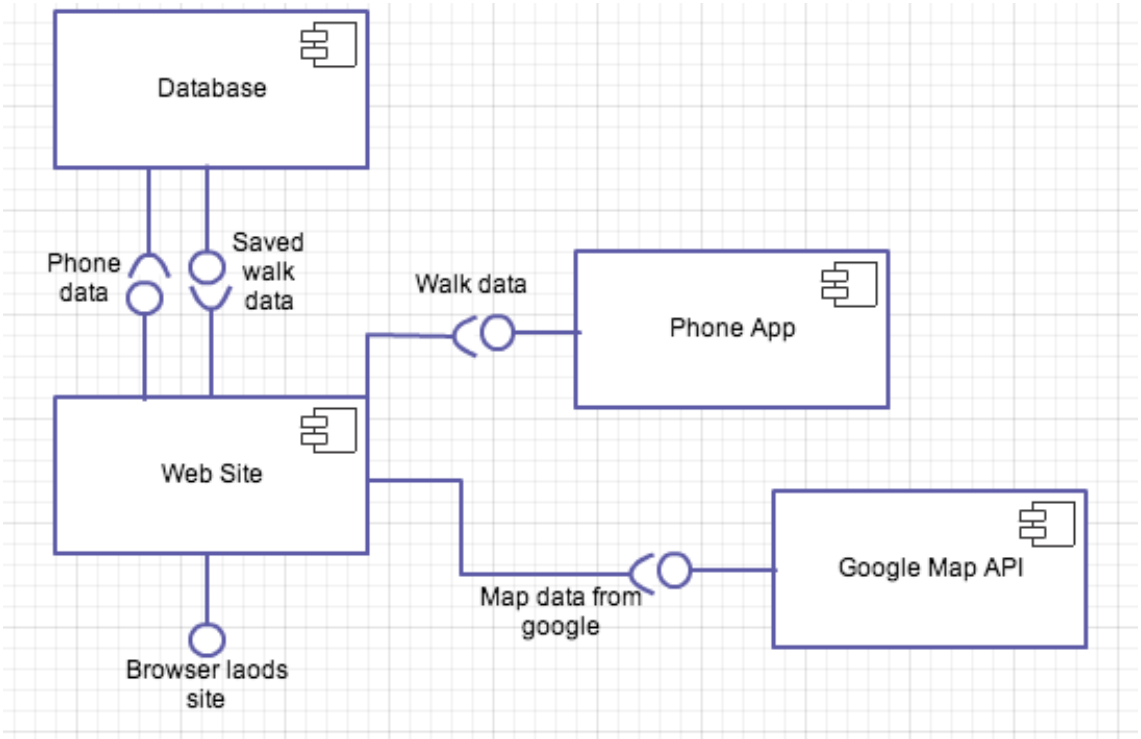
3 DEPENDENCY DESCRIPTION

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.

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

- Walk controller on GPS logger, 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

```
/**
 * This activity requires the user to enter a name, short description, and long
 * description of the walk. When they press the "Create route" button this
 * launches the "WalkCreatorActivity" giving it the details to create the walk
 * from.
 *
 * @author Group 14
 */
public class WalkDetailsActivity {
}
```

4.3 WalkController interface specification

```
/**
 * This interface is used to control the IWalk object.
 * When instantiated, this will create an IWalk and an AGPSLocationLogger.
 * It allows the viewer to add a point of interest, finish, or cancel the walk.
 *
 * @author Group 14
 */
public interface IWalkController {
   /**
    * This method takes an IPointOfInterest as an argument and adds it to the
    * IWalk.
    */
   public void addPointOfInterest(IPointOfInterest poi);

   /**
    * This method will stop the AGPSLocationLogger, remove the current IWalk,
    * and return to the MainAppActivity.
    */
   public void cancelWalk();
}
```

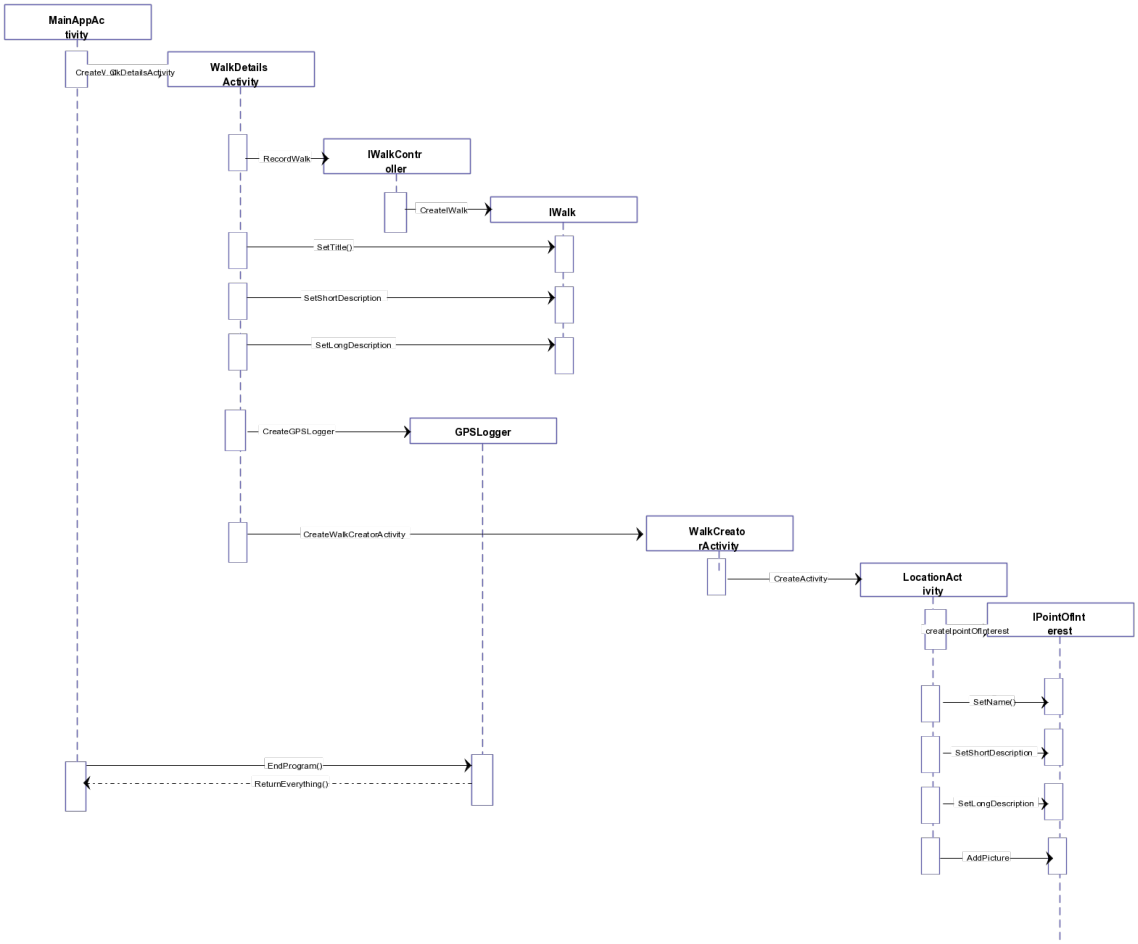
4.4 Walk interface specification

```
/**
 * This interface is what actually stores the data of the walk.
 * This contains a list of Location objects for the walk, a list of
 * IPointOfInterest objects, a title, a short description, and a long
 * description
 *
 * @author Group 14
 */
public interface IWalk {
   /**
    * This method takes an IPointOfInterest object and adds it to the list of
    * points of interest.
    */
   public void addPointOfInterest(IPointOfInterest point);

   /**
    * This method adds a LinkedList of Location objects to the walk.
    * This is to allow the controller to put a large number of Location objects
    * into the walk once the walk is finished and the logger as returned all of
    * the points of data.
    */
   public void addLocations(LinkedList locations);

   /**
    * This sets the walk's user-viewable title.
    */
   public void setTitle(String title);

   /**
    * This sets the walks short (<100 char) description.
    */
   public void setShortDesctiption(String Sdesc);

   /**
    * This sets the walk's short (<1000 char) description.
    */
   public void setLongDescription(String Ldesc);

   /**
    * This returns an array of all of the IPointOfInterest objects.
    * This is to be used when finishing and uploading the walk.
    */
```

```
    public IPointOfInterest[] getPointsOfInterest();

    /**
     * This returns an array of all of the walk's Location objects.
     * This is used when finishing and uploading the walk.
     */
    public android.location.Location[] getLocations();

    /**
     * This returns the String containing the walk's title.
     */
    public String getTitle();

    /**
     * This returns the string containing the walk's short (<100 characters)
     * description.
     */
    public String getShortDescription();

    /**
     * This returns the string containing the walk's long (<1000 characters)
     * description.
     */
    public String getLongDescription();
}
```

5 DETAILED DESIGN

5.1 Sequence diagrams



5.2 Significant algorithms

5.2.1 LocationLoggerService algorithm

This encompasses several aglroithms. First it decides wich method it picks by looking at averages over several points and looking at the standard deviation to see if any anomolies might indicate a change in direction. This is separated into several different algorithms:

Straight line
No change of direction across several points (likely 4 or more, but this could be changed in the algorithms implementation as a variable) will cause the points in the middle to be removed. This will look at the standard deviation of points and general direciton of points to ensure it's not slowly going round a corner.

Corner
It will be possible to test for corners above 60 easily by testing for a straight line followed by another straight line in a completely different direction. This will probably rely on the previous algorithm to thin out some points before this can take effect.

Slight turn
Detecting a slight turn with a low angle of difference will be difficult. It will be quite similar to detecting a corner but will likely require more points of data to detect that the change of direction is not just an anomaly in a straight line. We can make a vector out of the direction of each 3 points in succession with a direction and detect when several lines in one direction are suddenly followed by several lines in a slightly different direction.

Rounded turn
Detecting a rounded turn will be the most challenging algorithm. This will be used for large, sweeping corners and will most likely require either a large loop or some recursion. Due to the constant change of direction across the turn this will not be picked up by any of the other algorithms. This itself will ignore the corner and leave the points in place, allowing the corner's GPS points to be preserved.

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 AGPSLocationLogger 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