Group 10 Design Specification

December 3, 2013

SE.10.D3

Version: 1.0
Status: Release

Contributor Name	**Role**
Daniel Clark	Project Lead
Mark Lewis	QA Manager
Charles Newey	Deputy Project Lead & Android Developer
Martin Ferris	Android Developer
Ashley Iles	Android Developer
Kenny Packer	Android Developer
Stephen McFarlane	Deputy QA & Web Developer
Kieran Palmer	Web Developer

Department of Computer Science,
Llandinam Building,
Aberystwyth University,
Aberystwyth,
Ceredigion,
SY23 3DB

Contents

1 INTRODUCTION

1.1 Purpose of This Document

The purpose of this document is to show that we have met the outlined objectives specified by the client.

1.2 Scope

Scope

1.3 Objectives

- Objective 1
- Objective 2

2 DECOMPOSITION DESCRIPTION

2.1 Programs in System

The System that will be created will contain three systems; the Android app, a server and a web page.

The Android side of the system will be used as the platform for the walking tour creator app. Within this app users can create a new tour, they will then be able to add locations to the tour. These locations will contain information such as the tours title, description, the way points of the location and possibly an image. Once the tour is saved and finished it can then be sent via Wi-Fi or internet connection on the device to the server. The data will be sent to the server as a Multimedia Internet Message Extension in a http POST.

The server will be used to store all data about tours created on the Android app. On the server a database will be created and within the database a table for storing the tour's details. Once data is received a new record will be created and the data stored for easy re-use on the web page. To access this data the web page will use a php script with mySQL commands.

The final program in the system is the web page, which will be used for displaying tours. On the website there will be a list of saved tours from which the user can click to view them. The tour will then be displayed on a map which will incorporate the use of the Google Maps API. By clicking on a way point the user will be able get more information depending on what was saved by the creator using the Android app.

3 SIGNIFICANT ALGORITHMS

3.1 Upload Data

1. Bundle data
2. Cache Bundle
3. Send Bundle To Server
4. Display Progress bar in android drop down menu
5. Wait for confirmation...
6. If confirmation timeout reached... resend bundle.

3.2 Bundle Data

1. Base 64 encode all images
2. Collate data for each waypoint into JSON
3. Collate every waypoint route metadata into JSON

3.3 Validate Inputs

1. Check if empty
2. Check if contains "unsanitised" characters
3. Check image sizes

3.4 Change Screen

1. Validate Inputs
2. Display next screen

4 MAPPING REQUIREMENTS TO CLASSES

4.1 FR1; Startup of software on Android device

Android class Home and Android class AbstractActivity.

4.2 FR2; Providing info about the whole walking tour

Android class AbstractActivity, Android class DataEntryActivity and Android class AbstractActivity.

4.3 FR3; Adding locations to the walking tour

Android class CreateWaypoint, Android class DataEntryActivity and Android class AbstractActivity.

4.4 FR4; Adding photos to the walks

Android class CreateWaypoint, Android class DataEntryActivity and Android class AbstractActivity.

4.5 FR5; Cancelling walks

Android class CreateWaypoint and Android class AbstractActivity.

4.6 FR6; Sending the walk to the server

Android class FinishRoute and Android class AbstractActivity.

4.7 FR7; Switching from the WTC

Android class AbstractActivity.

4.8 FR8; Trying out a created walking tour

Web Page, Map and Database Interaction.

4.9 FR9; Saving data on server

Android class FinishRoute, Android class AbstractActivity, Web Page, Map and Database Interaction.

5 SIGNIFICANT CLASSES - ANDROID

5.1 StumblrData

This will be an abstract class which will contain variables and validation methods that other classes will use, such as short description and title.

5.2 Waypoint

This class will handle all information and methods regarding locations on the route. A method will be used to create a waypoint containing a title, short description and an image. Coordinates will be stored in a linked list of type coordinate.

5.3 Route

This class will contain all the methods to handle all parts of creating a route, such as adding waypoints to a linked list of type Waypoint.

6 SIGNIFICANT CLASSES - WEB

6.1 Page

The page class is the class that will make use of the map and the database interaction classes. This class will be written in HTML and CSS, this class will display the map and web interface.

6.2 Map

The Map class will be the interface to the Google Maps API; which will be in JavaScript. This class will be used in conjunction with the database interaction to load information into the map in the web interface, along with other details which will be entered into the information box on each location. This class will be implemented using JavaScript and PHP.

6.3 Database Interaction

The database interaction class is where the database will be loaded; to be used to show locations on the map, as well as showing different tours in a table next to it and pointers to each Waypoint in a tour in another table. this class will implemented with PHP and SQL.

7 INTERFACE DESCRIPTIONS - ANDROID

7.1 Android Activities

7.1.1 AbstractActivity {abstract}

Listing 1: AbstractActivity {Abstract}

```
1 package uk.ac.aber.cs.group10.stumblr;

3 import android.support.v7.app.ActionBarActivity;
  import android.support.v4.app.Fragment;
5 import android.os.Bundle;
  import android.view.LayoutInflater;
7 import android.view.Menu;
  import android.view.MenuItem;
9 import android.view.View;
```

```
import android.view.ViewGroup;

public abstract class AbstractActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        stumblrOnCreate(savedInstanceState);
    }

    public abstract void stumblrOnCreate(Bundle savedInstanceState);

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is
           present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    /**
     * A placeholder fragment containing a simple view.
     */
    public static class PlaceholderFragment extends Fragment {

        public PlaceholderFragment() {
        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup
           container,
                Bundle savedInstanceState) {
            View rootView = inflater.inflate(R.layout.fragment_abstract,
               container, false);
            return rootView;
        }
    }

}
```

7.1.2 DataEntryActivity {abstract}

Listing 2: DataEntryActivity {Abstract}

```
package uk.ac.aber.cs.group10.stumblr;

import android.support.v7.app.ActionBarActivity;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;

public abstract class DataEntryActivity extends ActionBarActivity {
    /**
     * Load the activity on creation (using a bundle if one is present)
     * @param savedInstanceState The bundle containing the saved instance
        state.
     */
    public abstract void stumblrOnCreate(Bundle savedInstanceState);
}
```

7.1.3 Home

Listing 3: Home

```
package uk.ac.aber.cs.group10.stumblr;

import android.os.Bundle;

import uk.ac.aber.cs.group10.stumblr.data.Route;

public abstract class Home extends AbstractActivity {
    /**
     * Loads the activity on creation (using a bundle if one is present)
     * @param savedInstanceState The bundle containing the saved instance
        state.
     */
    public void stumblrOnCreate(Bundle savedInstanceState) {
        // Called by super().onCreate
        setContentView(R.layout.activity_abstract);

        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.container, new PlaceholderFragment())
                    .commit();
        }
    }

    /**
     * Begin the Route entry activity
     */
    public abstract void startCreateRouteIntent(Route r);
}
```

7.1.4 CreateRoute

Listing 4: CreateRoute

```
1  package uk.ac.aber.cs.group10.stumblr;
2
3  import android.os.Bundle;
4
5  import uk.ac.aber.cs.group10.stumblr.data.Route;
6
7  public abstract class CreateRoute extends DataEntryActivity {
8      /**
9       * Loads the activity on creation (using a bundle if one is present)
10      * @param savedInstanceState The bundle containing the saved instance
            state.
11      */
12     @Override
13     public void stumblrOnCreate(Bundle savedInstanceState) {
14         // Called by super().onCreate
15         setContentView(R.layout.activity_abstract);
16
17         if (savedInstanceState == null) {
18             getSupportFragmentManager().beginTransaction()
19                     .add(R.id.container, new AbstractActivity.
                           PlaceholderFragment())
20                     .commit();
21         }
22     }
23
24
25     /**
26      * Create a new Route object (using the information entered by the
            user)
27      */
28     public abstract void createNewRoute();
29
30     /**
31      * Start the WaypointList activity (list the current Route).
32      */
33     public abstract void startWaypointListIntent(Route r);
34 }
```

7.1.5 CreateWaypoint

Listing 5: CreateWaypoint

```
package uk.ac.aber.cs.group10.stumblr;

import android.os.Bundle;

import uk.ac.aber.cs.group10.stumblr.data.Route;
import uk.ac.aber.cs.group10.stumblr.data.Waypoint;

public abstract class CreateWaypoint extends DataEntry {
    /**
     * Loads the activity on creation (using a bundle if one is present)
     * @param savedInstanceState The bundle containing the saved instance
        state.
     */
    public void stumblrOnCreate(Bundle savedInstanceState) {
        // Called by super().onCreate
        setContentView(R.layout.activity_abstract);

        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.container, new AbstractActivity.
                       PlaceholderFragment())
                    .commit();
        }
    }

    /**
     * Create a new Waypoint based on user input.
     */
    public abstract void createWaypoint();

    /**
     * Obtain a photo from user and add it to current Waypoint.
     */
    public abstract void getImage(Waypoint w);

    /**
     * Set a timestamp on the current Waypoint.
     */
    public abstract void getTimestamp();

    /**
     * Obtain coordinates from Android system and add to current Waypoint.
     */
}
```

7.1.6 WaypointList

Listing 6: WaypointList

```
package uk.ac.aber.cs.group10.stumblr;

import android.os.Bundle;

import uk.ac.aber.cs.group10.stumblr.data.Route;

public abstract class WaypointList extends AbstractActivity {
    /**
     * Loads the activity on creation (using a bundle if one is present)
     * @param savedInstanceState The bundle containing the saved instance
        state.
     */
    public void stumblrOnCreate(Bundle savedInstanceState) {
        // Called by super().onCreate
        setContentView(R.layout.activity_abstract);

        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.container, new AbstractActivity.
                       PlaceholderFragment())
                    .commit();
        }
    }

    /**
     * Renders Waypoint list on screen.
     * @param r The Route object containing Waypoints to render.
     */
    public abstract void drawWaypointList(Route r);
}
```

7.1.7 FinishRoute

Listing 7: FinishRoute

```
package uk.ac.aber.cs.group10.stumblr;

import android.os.Bundle;

import uk.ac.aber.cs.group10.stumblr.data.Route;

public abstract class FinishRoute extends AbstractActivity {
    /**
     * Loads the activity on creation (using a bundle if one is present)
     * @param savedInstanceState The bundle containing the saved instance
        state.
     */
    public void stumblrOnCreate(Bundle savedInstanceState) {
        // Called by super().onCreate
        setContentView(R.layout.activity_abstract);

        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                    .add(R.id.container, new PlaceholderFragment())
                    .commit();
        }
    }

    /**
     * Will package and upload the current Route object.
     */
    public abstract void uploadRoute(Route r);
}
```

7.2 Data Classes

7.2.1 StumblrData {abstract}

Listing 8: StumblrData

```
1  package uk.ac.aber.cs.group10.stumblr.data;
2
3  /**
4   * Abstract class containing basic common structure for all Stumblr data
       formats.
5   */
6  public abstract class StumblrData {
7      /**
8       * The title of the given piece of data. This can be extended to
           Waypoints, Routes
9       * and any other piece of relevant data inside the structure of
           Stumblr.
10      */
11     private String title;
12
13     /**
14      * The short description pertaining to the given piece of data. This
           will also be
15      * extended to other component classes of StumblrData
16      */
17     private String shortDesc;
18
19     /**
20      * @param title The title to set.
21      * @param shortDesc The short description to set.
22      */
23     public StumblrData(String title, String shortDesc) {
24         this.title = title;
25         this.shortDesc = shortDesc;
26     }
27
28     /**
29      * Checks the StumblrData item for validity. Returns a boolean. (true
           = valid)
30      * @return Whether the data is valid or not. (true = valid)
31      *
32      * MUST be implemented in any subclasses.
33      */
34     public abstract boolean isValidData();
35
36     /**
37      * Returns the title.
38      * @return this.title
39      */
40     public String getTitle() {
41         return this.title;
42     }
43
44     /**
```

```
45          * Sets the current title.
            * @param title The title to set.
47          */
        public void setTitle(String title) {
49          this.title = title;
        }
51
        /**
53          * Returns the short description
            * @return this.shortDesc
55          */
        public String getShortDesc() {
57          return this.getShortDesc();
        }
59
        /**
61          * Sets the current short description.
            * @param shortDesc The short description to set.
63          */
        public void getShortDesc(String shortDesc) {
65          this.shortDesc = shortDesc;
        }
67  }
```

7.2.2 Route

Listing 9: Route

```
1  package uk.ac.aber.cs.group10.stumblr.data;
2
3  import java.net.URL;
   import java.util.LinkedList;
5
   /**
7   * Created by charles on 29/11/13.
    */
9  public class Route extends StumblrData {
       /**
11      * A slightly longer description of the contents of the route. Set by
            the user when
        * they create a Route.
13      */
       private String longDesc;
15
       /**
17      * A LinkedList of Waypoint objects that the Route comprises of.
        */
19     private LinkedList<Waypoint> route;
21     /**
        * Checks if the data in the Route is valid or not, and returns a
            boolean.
23      * @return If the data is valid or not. (true = valid)
        */
25     public boolean isValidData() {
           return false;
27     }
29     /**
        * Adds a Waypoint to the tail of the Route LinkedList
31      * @param w The waypoint to add
        */
33     public void addWaypoint(Waypoint w) {
           this.route.addLast(w);
35     }
37     /**
        * Returns the last Waypoint to the LinkedList.
39      * @return The last Waypoint in the Route.
        */
41     public Waypoint getWaypoint() {
           return this.route.getLast();
43     }
45     public String getLongDesc() {
           return this.longDesc;
47     }
49     /**
```

```
     * Constructor for Route.
51   * @param title The title of the Route object.
     * @param shortDesc A short description of the Route.
53   * @param longDesc A longer description of the Route.
     */
55  public Route(String title, String shortDesc, String longDesc) {
        super(title, shortDesc);
57
    }
59
    /**
61   * To be implemented. Will return a URL containing filesystem location
         of bundled Route file
     * (ready for upload to server)
63   * @return The URL of the bundle
     */
65  public URL bundle() {
        return null;
67  }
}
```

7.2.3 Waypoint

Listing 10: Waypoint

```
package uk.ac.aber.cs.group10.stumblr.data;

import android.graphics.Bitmap;
import android.location.Location;

import java.util.Date;
import java.util.LinkedList;

/**
 * Created by charles on 29/11/13.
 */
public class Waypoint extends StumblrData {
    /* title and shortDesc are declared in StumblrData and are accessed
       through get/set methods */

    /**
     * Image contained within Waypoint.
     */
    private Bitmap image;

    /**
     * Time at which Waypoint was created.
     */
    private Date timestamp;

    /**
     * List of coordinates (from the last location to the current location
        )
     */
    private LinkedList<Location> coordList;

    private int index;

    /**
     * Constructor for a Waypoint object.
     * @param title Title of the waypoint.
     * @param shortDesc A short description.
     */
    public Waypoint(String title, String shortDesc, int index) {
        /* Calls superclass constructor */
        super(title, shortDesc);

        /* Sets index variable (used for locating position in array */
        this.index = index;

        /* Uses Android system to get time. */
        this.timestamp = new Date();
        /* Initialise LinkedList */
        this.coordList = new LinkedList<Location>();
    }
```

```
50      /**
        * To be implemented.
52      * @return Validity of data (true = valid)
        */
54      public boolean isValidData() {
            return false;
56      }

58      /**
        * Add a coordinate to the list.
60      * @param c The location to be added to the tail of the LinkedList.
        */
62      public void addCoordinate(Location c) {
            this.coordList.addLast(c);
64      }

66      /**
        * Returns coordinate instance at specified index.
68      * @param index Specified index of coordinate.
        * @return The specified coordinate.
70      */
        public Location getCoordinate(int index) {
72          return this.coordList.get(index);
        }
74
        /**
76      * Sets the current image.
        * @param image The image to add to the Waypoint.
78      */
        public void setImage(Bitmap image) {
80          this.image = image;
        }
82
        /**
84      * Returns current image.
        * @return The current image that the Waypoint has,
86      */
        public Bitmap getImage() {
88          return this.image;
        }
90
        /**
92      * Returns current timestamp.
        * @return The current timestamp.
94      */
        public Date getTimestamp() {
96          return this.timestamp;
        }
98

100 }
```

8 Website Interface Specification

The web interface will use all of these classes to create a page that will load in a database of routes that the user will be able to view a route map from. The user will also be able to choose from a range of tours provided right next to the map.

8.1 Page

The page class is the web page that will hold the map and database interaction. This class will be the way the interface is shown by using css. This class will hold the map class and the database interaction class. The page class will be done in HTML and it will validate to XHTML 1.0 Strict.

8.2 The Map

The map class is the class which holds the Google maps API which will be in JavaScript. This class holds functions that are provided from the google maps API website that initialise the maps and the pointers on the map. There also other function such as myclick() which records when the user click on the map. Also there is a createMarker() function which will create an info box for each point provided this is were we will provide the information for each place by using the database interaction here to create all the pointers. >Here are the functions and javacript I will using from the Google Maps API website that will be working with the database interaction.

Listing 11: Google Maps API Javascript Example

```
<script type="text/javascript">
  var side_bar_html = "";
  var gmarkers = [];
  var map = null;
//initializes the Google Map.
function initialize() {
// create the map
var myOptions = {
zoom: 15,
//sets the map to Aberytwyth
center: new google.maps.LatLng(52.4140,-4.0810),
mapTypeControl: true,
mapTypeControlOptions: {style: google.maps.MapTypeControlStyle.
   DROPDOWN_MENU},
navigationControl: true,
mapTypeId: google.maps.MapTypeId.ROADMAP
}
map = new google.maps.Map(document.getElementById("map_canvas"),
                          myOptions);

google.maps.event.addListener(map, 'click', function() {
    infowindow.close();
    });

create pointer
//cretates pointers this will be used with the database interaction class
   to load
//in each place in the tour and its information
var point = new google.maps.LatLng(,);
var marker = createMarker(point,"NAME","DESCRIPTION")

document.getElementById("side_bar").innerHTML = side_bar_html;
}
//creates the map and set the size of the map.
var infowindow = new google.maps.InfoWindow(
{
  size: new google.maps.Size(200,75)
});


function myclick(i) {
google.maps.event.trigger(gmarkers[i], "click");
}


function createMarker(latlng, name, html) {
  var contentString = html;
  var marker = new google.maps.Marker({
      position: latlng,
      map: map,
      zIndex: Math.round(latlng.lat()*-100000)<<5
      });
```

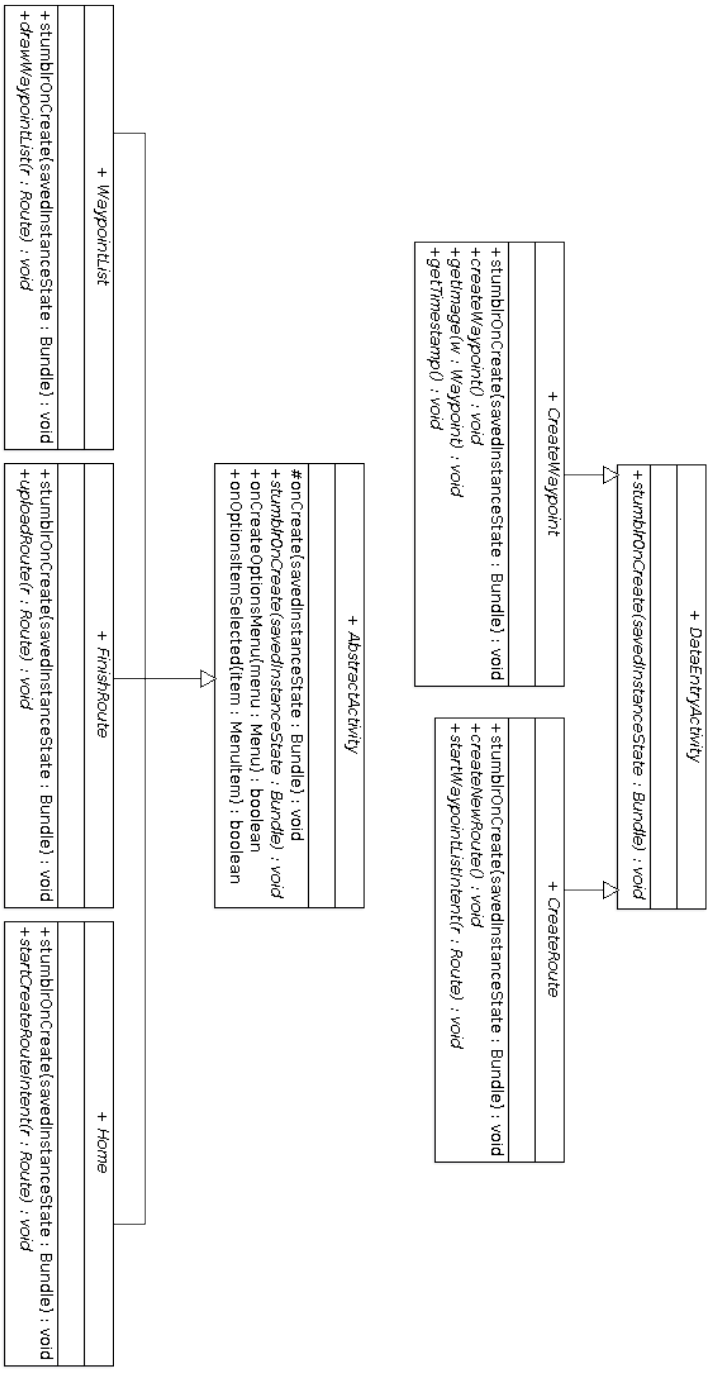
```
52    google.maps.event.addListener(marker, 'click', function() {
          infowindow.setContent(contentString);
54        infowindow.open(map,marker);
          });
56
    gmarkers.push(marker);
58    side_bar_html += '<a href="javascript:myclick(' + (gmarkers.length-1) +
        ')">' + name + '<\/a><br>';
}
60  </script>
```

8.3 Database Interaction

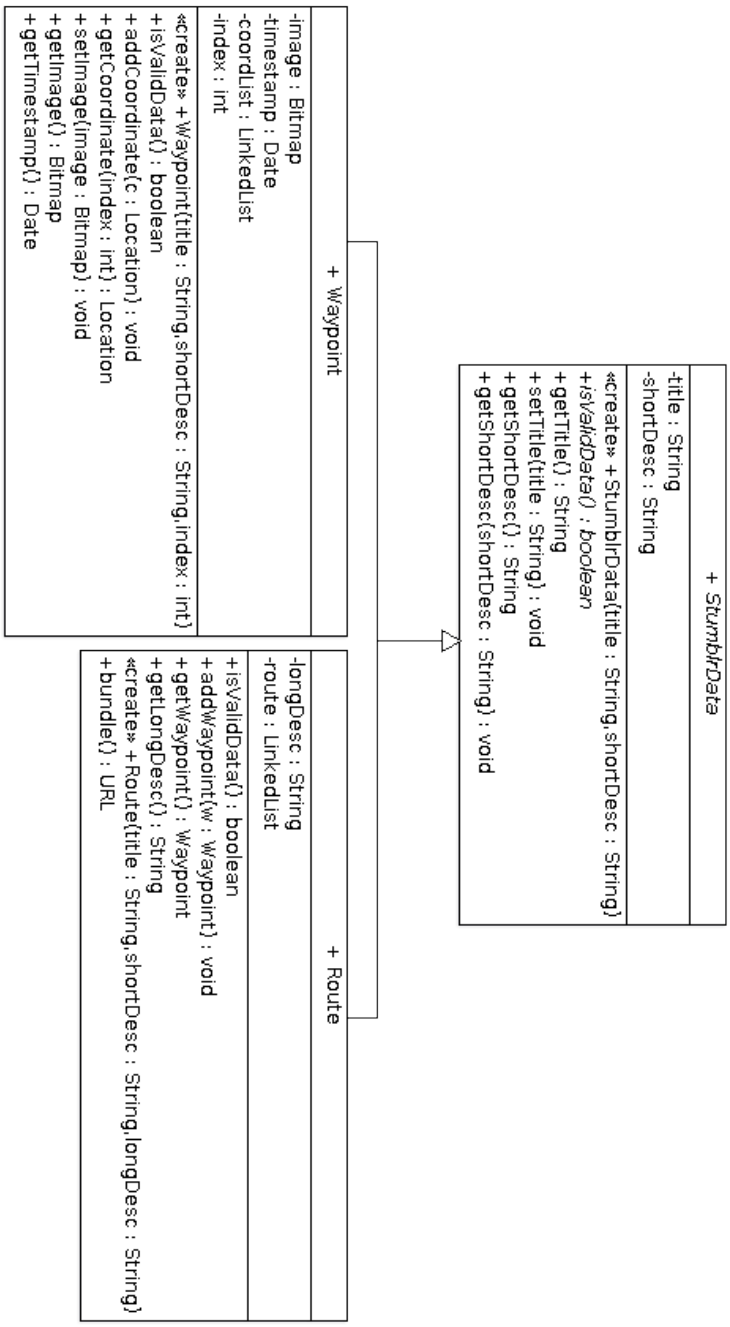
The database interaction class is where the web page will connect to the databases then It will use jquery to get all the points on the tour that the user choose. It will present the tours as links to the left of the map which when clicked will connect to that database and load in all the points to the map and load pointer to the right of the map. The map will have information boxes for each point on the map which have had information put in them by using Jquery as the information for each point is save alongside the coordinates. A thumbnail of an image of the location will also be in these information boxes and when clicked on a bigger image will show at the centre of the page. >PHP code will be used to read in the databases of tours which will contain each tours the code will create links on the left of the map to bring up a new tour and load in the poiner.The PHP code used will mostly be in the create pointer part of the map class where it will create pointer for each entry in the database which will then create a link on the right of the map to link to each pointer on the map.

9 CLASS DIAGRAMS

9.1 Android Activity Classes



9.2 Android Data Classes



10 Android Design Justifications

Use of JSON (spoke to customer... etc)

11 Android Design Justifications

Use of JSON (spoke to customer... etc) Google maps

12 REFERENCES

[1] Hurnherr, T. LaTeXListings Package Info.
http://texblog.org/2008/04/02/include-source-code-in-latex-with-listings/.
Accessed: Dec 03, 2013.

[2] Pavlic, T. Programming Assignment template for LaTeX.
http://www.latextemplates.com/template/programming-coding-assignment.
Accessed: Oct 23, 2013.

[3] Various. LaTeXListings Java Highlighting.
http://tex.stackexchange.com/questions/115467/listings-highlight-java-annotations.
Accessed: Dec 03, 2013.

13 VERSION HISTORY

Author	Date	Version	Change made
CCN	03/12/2013	1.0	Initial build of document