

# Design and Development Document

---

## Trail Seeker

**By Peter Mitchell**

**November 2012**

## Table of Contents

1 Introduction .....	3
2 Overview .....	4
3 Storyboard .....	5
4 Application Walkthrough .....	6
4.1 Tab Interface Basics .....	6
4.2 Create Trail Interface .....	7
4.3 Summary Interface.....	9
4.4 Trail Viewing Interface .....	11
5 Software Design .....	12
5.1 Class Diagram .....	12
5.2 Decisions Made .....	13
5.2.1 Interface Design .....	13
5.2.2 Relative Point Graphing .....	13
6 Development Issues and Resolutions .....	14
6.1 GPS Positioning and GoogleMaps.....	14
6.2 Database Management Solution .....	15
6.3 Network Solution .....	15
6.3 Summary of Known Issues .....	16
6.3.1 Critical Issues.....	16
6.3.2 Other issues .....	16
7 Testing Checklist .....	17
8 Future Work .....	22
9 Conclusions .....	22
10 References .....	22

## Table of Figures

Figure 1. Storyboard Activity Diagram .....	5
Figure 2. Application Icon. ....	6
Figure 3. My Maps tab showing one trail. ....	6
Figure 4. Empty Create Trail Interface. ....	7
Figure 5. Populated Create Trail Interface.....	7
Figure 6. Point Editing Interface. ....	8
Figure 7. My Map list showing new element.....	8
Figure 8. Summary Interface.....	9
Figure 9. Summary Interface Uploading. ....	9
Figure 10. Find More with item. ....	10
Figure 11. Summary Interface via Find More. ....	10
Figure 12. Downloaded Tab with Item.....	10
Figure 13. Summary Interface via Downloaded.....	10
Figure 14. Start point not reached.....	11
Figure 15. Hint at point. ....	11
Figure 16. A pair of points shown. ....	11
Figure 17. Class Diagram.....	12

## 1 Introduction

The purpose of this project was to develop an application that utilises multiple sensors with the GPS sensor as the primary focus. This application has used only the GPS and touch sensors; however, the camera sensor could be incorporated in a couple of possible ways as will be described in Future Work (section 8). The GPS sensor is used in this application to determine a series of waypoints that can be modified to have associated description attributes and then stored in a local database. These way points can be played back as a path that can be followed by using the description from each point to tell the user how to reach the next point. The touch sensor is used primarily in the native GUI controls interface of the android system. The touch sensor is also used to allow access of previous point's information that is shown on the path map. This system is extended community development by allowing users to upload their paths for anyone to download and participate in.

This document serves the function of both a design document identifying the core functionality and the development process and outcomes that were achieved. The early sections on the design identify the Overview (section 2) with the features of the application, and a Storyboard (section 3) in the form of an activity diagram. The Application Walkthrough (section 4) covers a simple walkthrough demonstrating the core features of the application taken from the final version. A more detailed walkthrough guide for testing can be found at the Testing Checklist (section 7). The Software Design (section 5) covers the identification of class structuring (section 5.1) and the decisions made (section 5.2) that influenced the development. The Development Issues and Resolutions (section 6) covers the issues encountered during development and how they were overcome and worked around. The Testing Checklist (section 7) covers a walkthrough that can be used to pass the application with most of the features tested for reliability to a basic level. Future Work (section 8) and Conclusions (section 9) tie up the experiences and future directions. Finally the References (section 10) indicate core influences for the base code that were incorporated in development.

## 2 Overview

This application provides a basic implementation of a waypoint creation and following system. The title assigned to this project was “Trail Seeker”. This title was indicative of seeking trails to follow where the user may not know where they will end up. It is the responsibility of the creators of the trails to make sure their trails give a method of reaching the next point. Trails that have been created can be uploaded to an online database to be retrieved by any other user of the software. At some later stage a more complex management and user control of these uploaded elements could be implemented. At this time the data is dumped onto the server and once there the uploaded can’t modify that version of the content. Core features include:

- listing of trails that have been created by the user, downloaded, or are online.
- creation of trails that have properties for:
  - title: the title of the trail that is shown in the lists.
  - author: the author of the trail.
  - creation date: the date the trail was created on (automatically assigned).
  - description: a description telling the user what to expect in the trail; these are details the user should know before reaching the start point.
  - detection range: to indicate distance that must be achieved from target nodes during the use of trails to constitute reaching each waypoint.
  - list of nodes with properties for:
    - node id: to identify the order of nodes.
    - latitude and longitude: to indicate the location associated.
    - description: to give a hint on how to reach the next point or to congratulate the user at the end point.
- viewing of a summary by selection from the list that allows for:
  - viewing of title, author, creation date, and description.
  - deletion of trail from personal maps or downloaded lists.
  - uploading of trails from personal maps.
  - viewing of a GoogleMaps view showing the marked start point.
  - entry into beginning a trail for the personal maps or downloaded lists.
- trail tracking system with features:
  - indicate distance from starting point until reached.
  - show points on automatically resizing map as each point is reached.
  - show description when points are reached indicating a hint on reaching the next point.
  - allow touch based selection of previously visited points to show the description associated.
- basic server code to make incoming connections for activities:
  - downloading of list of trails.
  - downloading of complete data on a specific trail.
  - uploading of a complete set of data for a trail.

### 3 Storyboard

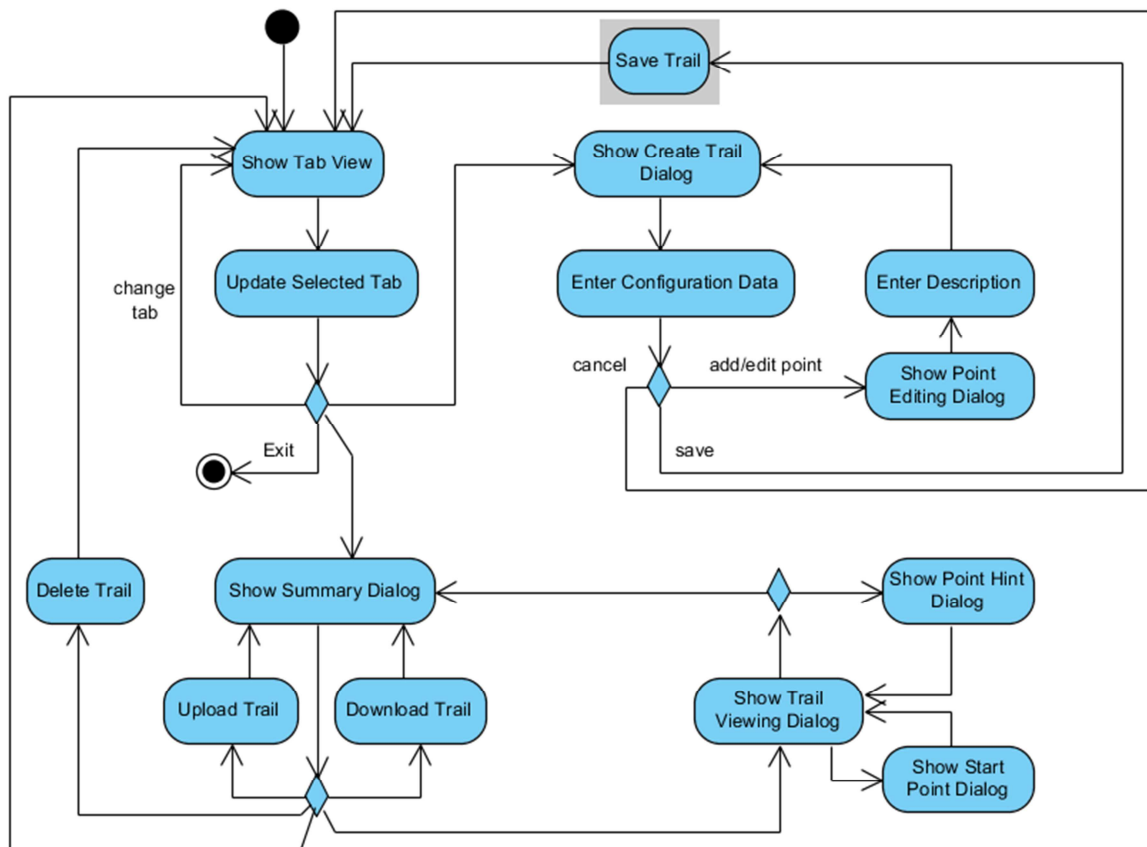


Figure 1. Storyboard Activity Diagram

The storyboard flow seen in Figure 1 shows the general flow with core functionality shown. The tab view provides an entry point to the application giving the option of selecting to list user created trails, downloaded trails, or the online list of trails. When a tab is selected the list is updated from the central database, and in the case of online list a new list is downloaded. The create trail dialog allows creation by entry of configuration data and points. The summary dialog is shown when a list item is selected from the main view. From the summary dialog depending on the content, upload, download, and delete can be used. Additionally this interface provides the method to open the trail viewing dialog where trails are “played through”. Point hint dialogs are shown at the correct intervals based on the processing in the trail viewing dialog.

## 4 Application Walkthrough

This section will walk through showing examples of content from the application and a basic description of the process that has to be used to achieve the displayed results. When entering the application the icon that was created to provide this entry point is shown in Figure 2. The icon was designed to show a trail as a yellowish path with green region either side.



Figure 2. Application Icon.

### 4.1 Tab Interface Basics

After entering the application the first screen that is shown is seen in **Error! Reference source not found..** The list view shows the list of trails based on the current tab. In the figure below the “My Maps” tab is selected. The elements shown in this list are those that the user of the device has created personally. The “Downloaded” tab shows the items that have been downloaded from within the “Find More” tab. The “Find More” tab shows the list of trails that are downloaded from the online server. These two tabs will be discussed further in section 4.3. The pictures are taken from a camera because for some reason the app no longer allows screen captures to be performed. Reason is unknown.

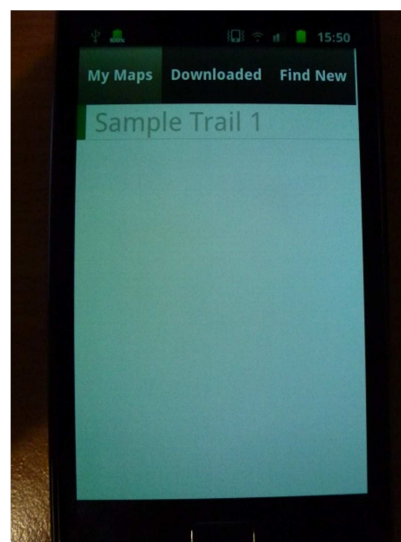


Figure 3. My Maps tab showing one trail.

From this tab interface there are a number of methods of performing operations apart from switching tabs. The most basic method of interaction is by selecting an item with a press and release. This operation will change the view to the summary view that will be covered later. The main method of interaction with the tab interface view is the pressing of the menu button on the device. Pressing this button will make the options “Create New Trail” and “Cancel” appear. Pressing

“Cancel” will make the menu disappear. Pressing “Create New Trail” will result in opening the create trail interface seen in Figure 4 below.

## 4.2 Create Trail Interface

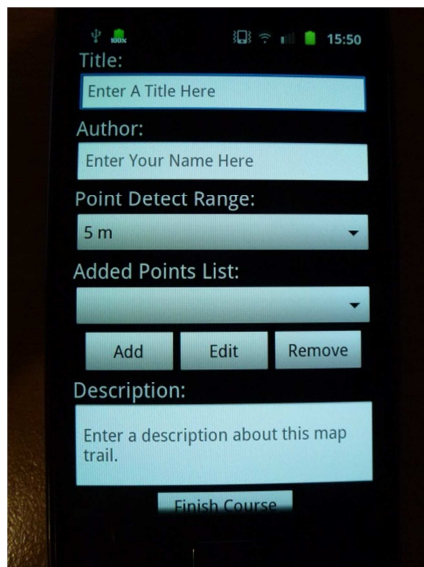


Figure 4. Empty Create Trail Interface.

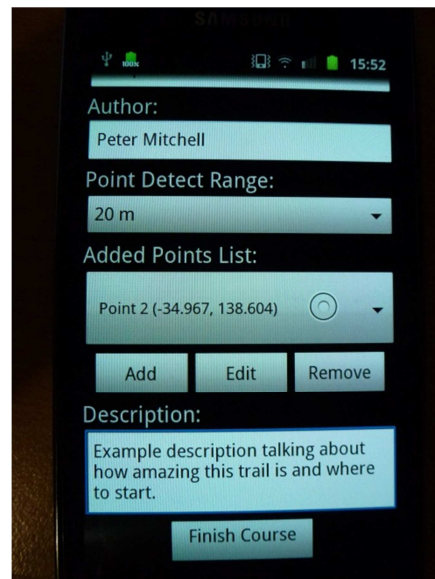


Figure 5. Populated Create Trail Interface.

Figure 4 shows the empty interface as would initially be presented to the user. The text boxes contain hints to provide context for the user to understand what should be entered into each box. After entering details for everything the dialog may appear as seen in Figure 5. When the “Finish Course” button is pressed the application validates each of the input elements. Each of the 3 text input fields are checked for at least one character present. Then the list of points is checked to ensure at least two points are included. This was made the minimum requirement so there would always be a start point and an end point at the very least. Cancellation of the creation can be performed by pressing the back button and confirming to not want to continue. In the cases where errors are detected when pressing the buttons in this interface a Toast message is shown to indicate to the user what the error is. Errors are presented one at a time rather than all at once to ensure each point is completed successfully. The remove and edit buttons apply to the currently selected point and automatically update the list as required. They will flash an error message if there are now elements. The add button will create a new point and open the interface to edit that point. This is seen in Figure 6.



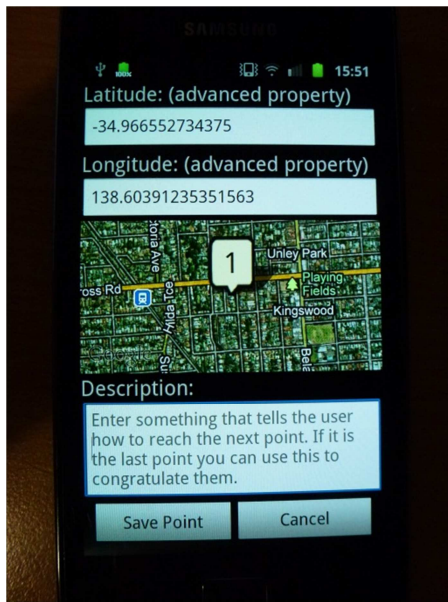


Figure 6. Point Editing Interface.

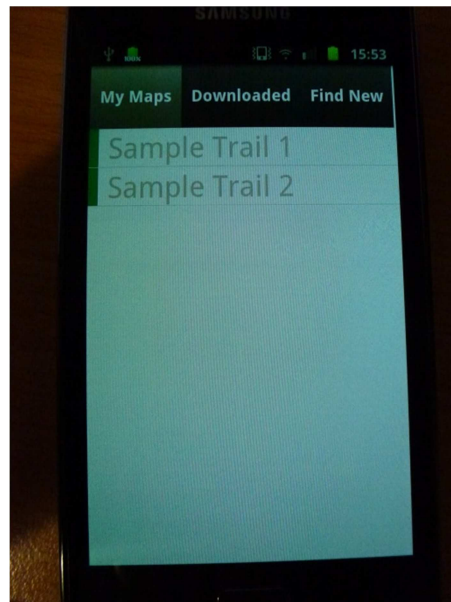


Figure 7. My Map list showing new element.

The point editing interface seen in Figure 6 indicates to the editor the current location they are editing for (this may be a new entry), and provides input capability only for entering into the description input. The latitude and longitude have been disabled in this version of the application. They could be enabled perhaps using an application options system in future developments. The map view shows a preview of where the location is with the id number shown in the custom generated overlay. There is a large issue here as the location manager did not appear to be reporting locations correctly. This is to confirm the point is correct. After completing the creation stages from using the dialog in Figure 4 the application will return to the tabbed list. Figure 7 shows the example inserted correctly into the list.

### 4.3 Summary Interface

The summary interface provides features that target individual trails that have been opened by selection from the lists in the tab interface. Figure 8 shows an example of the summary that appears based on the trail that was created in the earlier stages of this walkthrough. Depending on which list the trail has been opened from the summary interface shows different views or provides different options. As this particular example was just created it is only listed in the “My Maps” list. The options provided to items opened from this list include: delete, upload, and begin trail. To show the delete and upload options the menu button must be pressed. The map view shows the start point for the application. Pressing upload will begin uploading the required data to a server. As seen in Figure 9 a message is displayed showing the status of the uploading. The message “Uploading...” is shown initially to indicate the upload has begun. This message changes to either a success or failure message upon the related event occurring. This network interaction is performed asynchronously so the UI thread is not influenced.

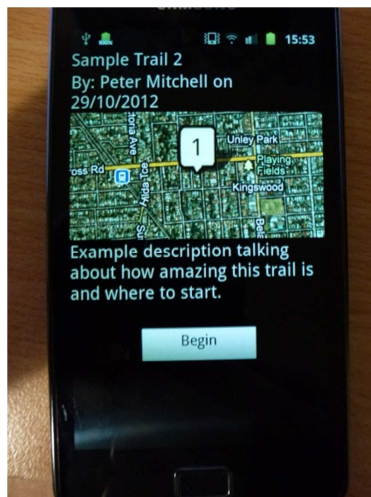


Figure 8. Summary Interface.

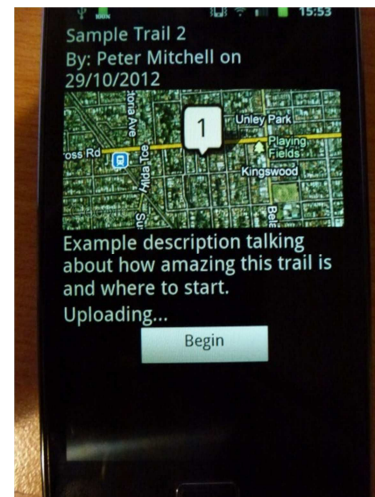


Figure 9. Summary Interface Uploading.

After pressing upload returning to the tabs interface and selecting the “Find More” tab as seen in Figure 10 the uploaded item was successful (in the testing the same item was uploaded twice, that is why two appear). Selecting this item shows the summary interface with modifications specific to online elements. This is seen in Figure 11. The differences in this interface compared to the other summary interface include the lack of an additional options menu meaning no delete or upload commands. Also instead of a begin button this is replaced with a download button. Pressing the download button provides a status in the same way as was seen in Figure 9. This status starts as “Downloading...” and then changes to either a successful or failed message. Additionally the button that said “Download” will change its text to say “Begin” and change in functionality. This allows immediate starting of use with the downloaded trail. If the trail had previously been downloaded this button change will also occur.

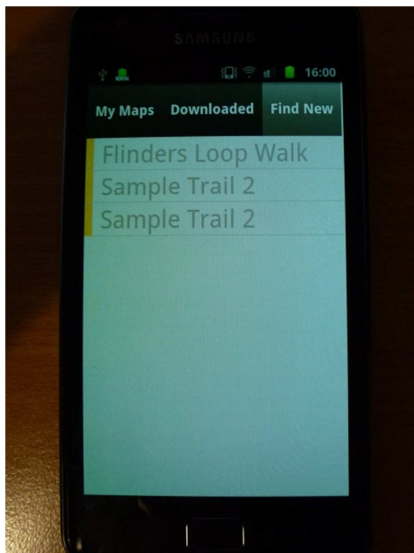


Figure 10. Find More with item.

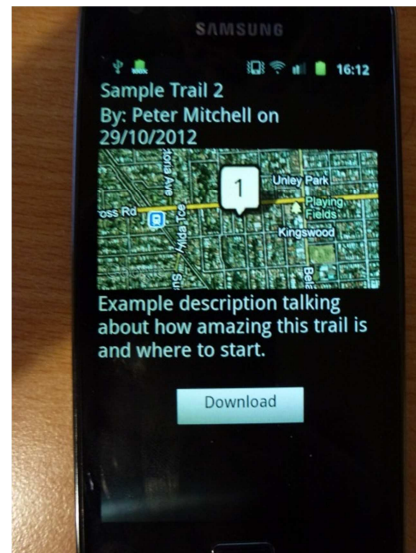


Figure 11. Summary Interface via Find More.

After completing this download returning to the tab list and selecting the “Downloaded” tab will reveal the downloaded item. This is seen in Figure 10. Selecting the item reveals Figure 11. This is exactly the same as the interface appeared in Figure 8. There is one change though. The delete option is available from the menu; however, the upload option is not available for the obvious reason that if an item has been downloaded then it must have already been online. The final item that hasn’t been discussed is the “Begin” button. This is covered in further detail in the next section (section 4.4).

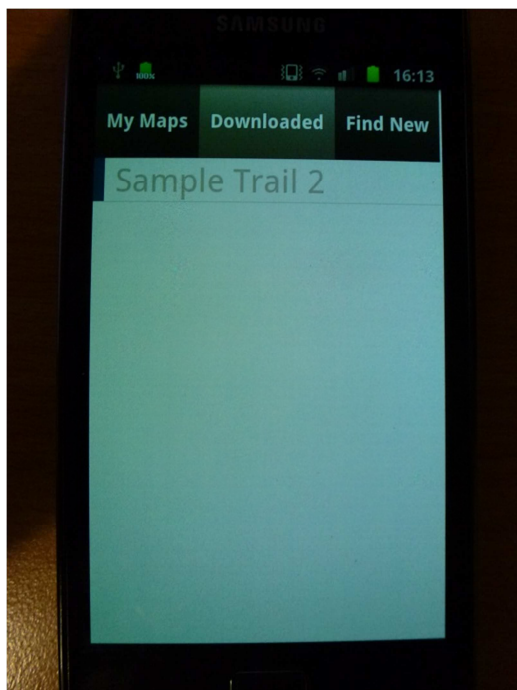


Figure 12. Downloaded Tab with Item.

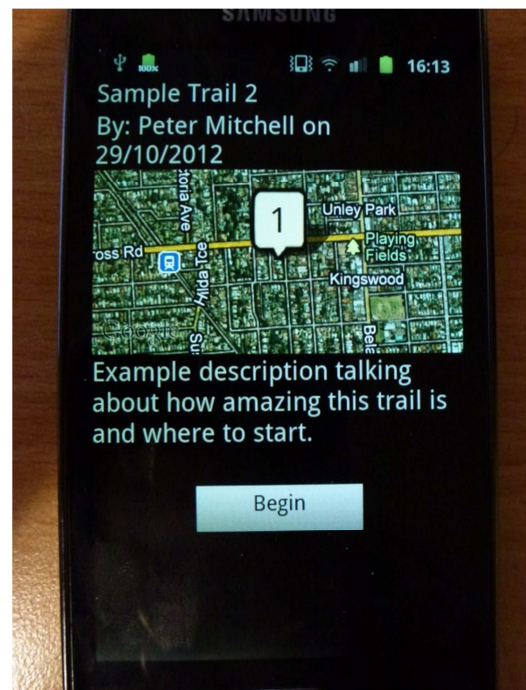


Figure 13. Summary Interface via Downloaded.

## 4.4 Trail Viewing Interface

The trail viewing interface allows users to play through a defined trail. Their first goal is to reach the start point. After completing this task they must then follow the instructions at each waypoint to work out how to get to the next one. Eventually they will be told once they have reached the final node that they have completed the trail. As each node is discovered the displayed map changes to expand and include the new nodes. For this reason the map may be regularly changing; this makes it apparent to the user the consequence of their actions. To open the trail interface the “Begin” button in either the My Maps or Downloaded list items must be used.

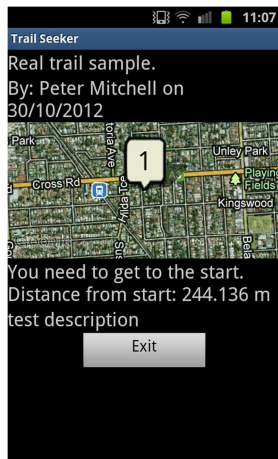


Figure 14. Start point not reached.

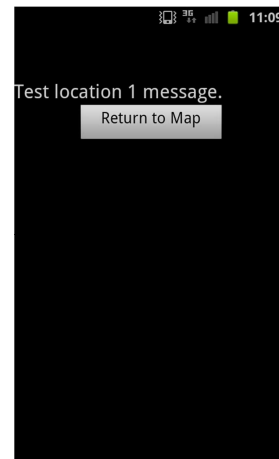


Figure 15. Hint at point.

Figure 14 shows the view that is displayed while trying to reach the start point. The distance from the start point is displayed and updated to indicate how long it will be till they are at the point. When the start point is reached Figure 15 shows the hint interface showing a description of how to get to the second point. After exiting this interface control returns to the trail view as seen below in **Error! Reference source not found..** (After also triggering a second point, because they were triggered at the same time due to the location issues.



Figure 16. A pair of points shown.

As the collected GPS points were not correct and useful there is not much point showing any further points marked on the map. Essentially though, after each point is reached the hint dialog in Figure 15 is displayed with eventually a completion message shown.

## 5 Software Design

### 5.1 Class Diagram

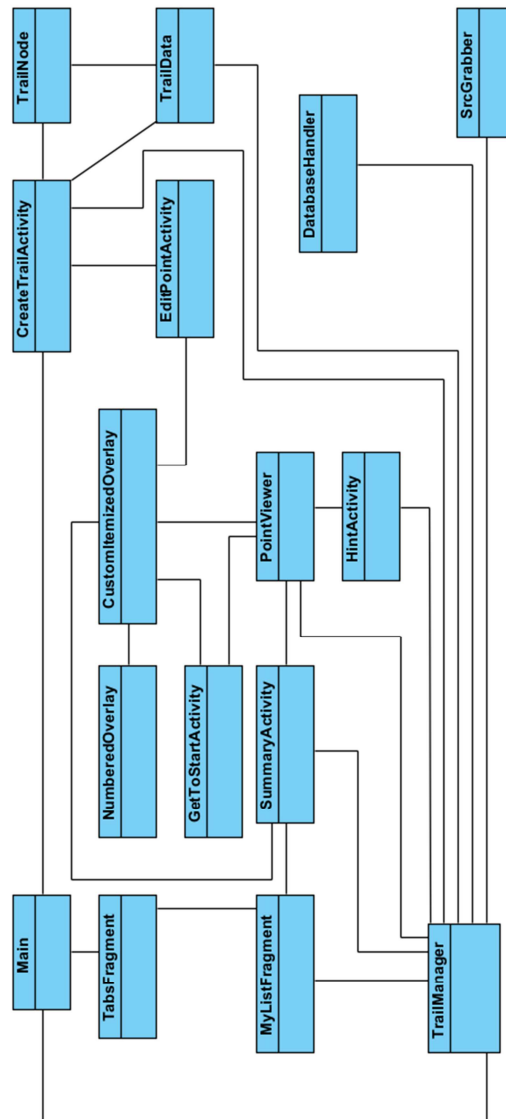


Figure 17. Class Diagram

Figure 17 shows the class diagram for this application. It can be seen from the diagram that the TrailManager is responsible for providing the data throughout the application. This class provides a global instance of itself to allow simplified database access. The other major point to note is the pair of NumberedOverlay and CustomItemizedOverlay classes. These provide the custom generated number overlays for use in the maps.



## 5.2 Decisions Made

### 5.2.1 Interface Design

The interface is definitely not perfect. There are likely simpler more aesthetic designs that allow for the type of interaction displayed in this application. In the development of this application the focus was primarily on the functionality rather than the perfection of visual design. The interface design principles have been used throughout the application to provide a constant experience. The obvious different exception is the trail point view. To summarise the major points that were kept the same where possible throughout the application are the following points:

- Basic input verification ensuring simply that there was input there that could be expanded to provide a more rigorous set of tests on data to test for validity.
- Confirmation using an alert dialog in all situations where data would be lost or an action performed that could not be undone easily. For example: deletion, uploading, cancelling and creation.
- Hiding of less used options in the option menu to reduce the likelihood of them being pressed accidentally by the novice user. The point about confirmation is used as a secondary insurance.

### 5.2.2 Relative Point Graphing

There were a number of possible ways of representing the points for the waypoints to the user. One certainly was to just use the GoogleMaps API to mark the locations. That was less desirable as it did not provide something as interesting to attempt to write code for initially. There were issues with integrating the GoogleMaps API so that was originally going to be left out. Instead an automatically rescaling map system was designed. The code built in XNA can be found in addition to the code translated into the Android project within the submission for this assignment. The XNA build shows a dummy rectangle to represent the draw space. The system works based on a comparison between the constant size of the draw space and an ever changing size in metres. The rectangle surrounding the units in metres contains all the points and expands when a new point is added that is outside the bounds. Points are determined by using the position of the previous point and applying a direction and distance. This will work well enough for trails smaller than a few thousand metres probably. Trails that have lots of points close together will suffer though. One possible fix for this issue would be to add a zoom function.

This feature was scrapped in a later stage of the project when additional time was granted to work on it. I discovered what the issue was with getting the GoogleMaps API integration to work and then incorporated that instead. The reason for this was the custom generated interface that contained just numbers was quite boring. The GoogleMaps graphics provide a further richness and continuity to the application. At this point the GoogleMaps elements were incorporated into the editing of waypoints as a preview, and the summary screens to show the start point. This was simplified from the original design that included a button that had to be pressed with the intention of opening the map to show the start location. This automatic integration provided a needed simplification.

## 6 Development Issues and Resolutions

This section gives an overview of the primary issues that were encountered during development and how they were dealt with.

### 6.1 GPS Positioning and GoogleMaps

During development I was sceptical about the correctness of the positional data that my device was receiving. Toward the end I believe the issues that were showing were due to being near buildings or other objects that reduced the quality of the signal. This sometimes causes the position reported to be way off from the actual position. Partly for this reason the GoogleMaps was used to provide a redundancy. It is unlikely any normal person would be able to read GPS coordinates and know exactly whether they are correct or not. Therefore the inclusion of the map inside of the point editing dialog gives the user the satisfaction of knowing that they are storing the correct location. The code does not work regardless though.

To access the location information the following code provided the starting point:

```
locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);  
Criteria criteria = new Criteria();  
provider = locationManager.getBestProvider(criteria, true);
```

This code gets the location server and detects the current best provider that is available. Then in the onResume event the location manager updating was requested:

```
locationManager.requestLocationUpdates(provider, 400, 1, this);
```

And in the onPause it was stopped:

```
locationManager.removeUpdates(this);
```

To handle events throughout the application the method below provides the updates that allowed comparison between points and the storing of the latitude and longitude for later use:

```
public void onLocationChanged(Location location) {  
  
}
```

GoogleMaps was initially difficult to integrate because both the generation of a key to match the application was an initial challenge (although this was accomplished without too much trouble), and the other thing was that the API was not registering. It turned out that the version of the Android API that had been included was not the API that included GoogleMaps. Changing the package fixed this issue. That then meant I could play with the GoogleMaps API a lot more. The main thing that my application required was numbered points. It was easy to work out how to add overly elements with custom icons. I needed to show the element with a custom rendered icon though. The generateMarker() method of the NumberedOverlay class is what handles this process. This was largely found based on other methods. A number of looked at to get the overall class to work properly. The largest issue once written was that the markers were not rendering. I worked out the reason for this was the drawing space was not being registered correctly. Eventually the solution to

this was to include the following line in the constructor of NumberedOverlay and this immediately fixed the issue.

```
this.mMarker.setBounds(-mMarker.getIntrinsicWidth()/2, -mMarker.getIntrinsicHeight(),  
mMarker.getIntrinsicWidth() /2, 0);
```

## 6.2 Database Management Solution

The TrailManager object is responsible for the overall management of the databases. The TrailManager keeps track of the summary data for all “My Maps”, “Downloaded”, and “Online” lists. A string list of the names is also stored to be used for updating the tab lasts. The object is also responsible for management of the local SQLite database and the communication for uploading and downloading of content to and from the server. The database for rapid simplicity was dumbed down to content that could be simply serialised in to let the client code do most of the work.

The client side SQLite database contains four columns in a single table. They are:

- type: defines whether the element is locally created or was downloaded. This is used to modify the context of ids.
- id: a unique identifier that becomes unique when paired with the type attribute. Locally created ids are incrementally larger, while downloaded ones use the ones that are stored on the internet based web server. This allows for future developments that could point back to the existing trail for updates or other purposes.
- summary: the summary contains the basic data about the trail that is used for showing the summary interface. This includes everything except the node list (start point is included in summary). (stored as an XML element)
- fulldata: this is an XML based data element that contains the full data inclusive of the summary and also the full list of nodes.

The server side database contains the id, summary, and fulldata fields. The id field is automatically assigned when the elements arrive to be uploaded and the summary and full data fields are stored along in the database as well.

## 6.3 Network Solution

The very first thing that I developed as a mini-test application during the topic was one that connected to a page on my web server and retrieved data. To accomplish this, the SrcGrabber.java file allows two modes of grabbing. One method uses a GET type approach to page retrieval and the other uses a POST type method with an additional package of data. The GET method was used for the downloading of the list of trails. The HTTP request is sent to the server and the PHP page just grabs a list of all the content. This is packaged in a simple XML structure and then parsed on the other side. All of the XML parsing has been broken down into sub components to allow for fluid dissemination of the data. The same approach is used when retrieving the full data of a single element. The POST method was only required therefore when an upload needed to occur. The XML copy of the summary element and of the full data structure is placed into the header and sent along with the request. In all cases there is an ERROR response replied with in case of a server side error and all other errors are caught on the client side when they occur. The code for the server is included in the Server folder with the submission.



## 6.3 Summary of Known Issues

### 6.3.1 Critical Issues

- Rapid changing of device orientation causes TabsFragment to crash.
- Random data loss while screen not visible in CreateTrailActivity.
  - Difficult to replicate, but happens occasionally when the screen has turned itself off and the screen is resumed.
- Location tracking does not appear to correspond to the correct locations. For example the GPS marker shows a few streets away or even a number of km away.
- The “full data” retrieval does not work straight away. For some reason there is a delay in storing the data into the SQLite database. This means the “Begin” and “Upload” functions that need to retrieve data proactively from the database can’t until the database has saved itself. The reason for this is unknown as the database is closed after saving data each time. To get around this issue currently the best way it to close the application and reopen it.

### 6.3.2 Other issues

- Icon does not appear correctly for application.
- Workaround on client side to handle additional data appended to source code received from free server. The free web host that is in use attaches hit logging code to the end of any page requests. This crashes the XML parser so a filter was put in place inside of SrcGrabber to automatically remove these. Although this should not be required if a proper host were used.
- Additional overhead of data due to deliberate simplification. Duplicate information of all the summary data is sent to the server so that the server code was minimal for generating retrieval listings.
- The full list of trails is sent from the server to client all as one block. Should be buffered instead of downloading full list every time.
- Server could be exploited as data is not checked thus causing potential crashes for all clients who use the application and download the list. (Vulnerable to SQL injection).
- Interface needs to be made more user friendly for optimal use in some way.

## 7 Testing Checklist

TestID	Test Description	PASS/ FAIL	Notes
1	View application listing on device. <ul style="list-style-type: none"> <li>See correct app name</li> <li>See correct icon</li> </ul>		
2	Open application. <ul style="list-style-type: none"> <li>Application successfully opens.</li> <li>Three tabs visible at top.</li> </ul>	PASS	
3	Open menu by pressing menu button. <ul style="list-style-type: none"> <li>Options "Create New Trail" and "Cancel" displayed.</li> <li>Cancel closes the menu.</li> </ul>	PASS	
4	Open menu and press "Create New Trail". <ul style="list-style-type: none"> <li>New Trail Dialog with empty fields displayed.</li> <li>No points in the list.</li> <li>Hints visible in text fields.</li> </ul>	PASS	
4a	Create Trail Dialog: <ul style="list-style-type: none"> <li>Point Detect Range contains "5 m", "10 m", "20 m".</li> <li>Pressing "Edit" shows Toast with "Error! Please add point first."</li> <li>Pressing "Remove" shows Toast with "Error! Please add point first."</li> <li>Pressing "Finish Course" shows Toast "Please enter a title!"</li> </ul>	PASS	
4b	Create Trail Dialog: <ul style="list-style-type: none"> <li>Enter Title</li> <li>Pressing "Finish Course" shows Toast "Please enter an author!"</li> </ul>	PASS	
4c	Create Trail Dialog: <ul style="list-style-type: none"> <li>Enter Author</li> <li>Pressing "Finish Course" shows Toast "Please enter a description!"</li> </ul>	PASS	
4d	Create Trail Dialog: <ul style="list-style-type: none"> <li>Enter Description</li> <li>Pressing "Finish Course" shows Toast "You need at least two waypoints!"</li> </ul>	PASS	
4e	Create Trail Dialog: <ul style="list-style-type: none"> <li>Press "Add"</li> <li>Point creation dialog appears.</li> <li>Latitude and Longitude populated with current coordinate.</li> <li>Map visible showing the correct location.</li> <li>Hint displayed in description box.</li> </ul>	FAIL	Does not always correspond to correct location.
4f	Create Trail Dialog: (Point creation dialog) <ul style="list-style-type: none"> <li>Press physical back button.</li> <li>Confirm dialog displayed.</li> </ul>	PASS	

	<ul style="list-style-type: none"> <li>• Press “Cancel”.</li> <li>• No points listed in “Added Points List” still.</li> </ul>		
4g	Create Trail Dialog: (Point creation dialog) <ul style="list-style-type: none"> <li>• Press “Add” button.</li> <li>• Confirm dialog displayed.</li> <li>• Press “Cancel” button.</li> <li>• No points listed in “Added Points List” still.</li> </ul>	PASS	
4h	Create Trail Dialog: (Point creation dialog) <ul style="list-style-type: none"> <li>• Press “Add” button.</li> <li>• Pressing “Save Point” shows Toast “Please enter a description.”</li> <li>• Enter a description.</li> <li>• Pressing “Save Point” returns to Create Trail Dialog and a point is listed in the “Added Points List”.</li> <li>• Point in list shows as “Point 1” followed by longitude and latitude with 3 decimal places.</li> </ul>	PASS	
4i	Create Trail Dialog: <ul style="list-style-type: none"> <li>• Press “Remove”.</li> <li>• Confirm dialog is displayed.</li> <li>• Pressing “No” closes dialog without changing list.</li> <li>• Press “Remove”.</li> <li>• Confirm dialog is displayed.</li> <li>• Pressing “Yes” closes the dialog and removes the point from the list.</li> </ul>	PASS	
4j	Create Trail Dialog: <ul style="list-style-type: none"> <li>• Add three points.</li> <li>• After each point is added the last point is the one selected as the option in the list.</li> <li>• The points are listed in the correct order 1, 2, and 3</li> <li>• Select point 2 from the list.</li> <li>• Press “Edit” and the point creation dialog is shown.</li> <li>• Description, latitude, and longitude all contain the correct information.</li> <li>• Change the description.</li> <li>• Press “Cancel” dialog is closed and returns to create trail dialog.</li> <li>• Press “Edit” to edit point 2 again.</li> <li>• Description is showing the original text and not the modified text before the cancel step.</li> <li>• Edit the description again and press “Save Point”.</li> <li>• After returning to the Create Trail Dialog</li> </ul>	FAIL	Points do not always correspond correctly.

	press "Edit". The description should show the new text.		
4k	Create Trail Dialog: <ul style="list-style-type: none"> <li>Select point 2 from the list.</li> <li>Press "Remove" and confirm.</li> <li>The point should not be visible in the list and the points remaining should be listed as 1 and 2.</li> </ul>	PASS	
4l	Create Trail Dialog: <ul style="list-style-type: none"> <li>Press "Finish Course".</li> <li>Dialog exits and the My Maps list shows the newly created Trail's title.</li> </ul>	PASS	
5	Press and hold on the trail title in the list. <ul style="list-style-type: none"> <li>A context menu should appear with "Open" and "Cancel" options.</li> <li>Cancel should close the context menu.</li> <li>"Open" should perform the same action as described in test 6.</li> </ul>	FAIL	DEPRECATED FEATURE: Removed because of complexity with collecting item id without adding any benefits over the press in test 6.
6	Press and release on the trail title in the list. <ul style="list-style-type: none"> <li>Summary Dialog should appear.</li> <li>Summary Dialog contains title, author, creation date, a map with the start point and description. Button "Begin" is visible.</li> <li>Press physical back button and this should return to the list.</li> </ul>	PASS	
7	Summary Dialog: <ul style="list-style-type: none"> <li>Open Summary from method in test 5 or 6.</li> <li>Press physical menu button.</li> <li>"Delete", "Upload" and "Cancel" options are displayed.</li> <li>Press "Cancel" and the menu should disappear.</li> <li>Open the menu and press "Delete". The dialog should close and the list updated to not show the item.</li> </ul>	PASS	
8	Summary Dialog: <ul style="list-style-type: none"> <li>Create another trail.</li> <li>Open summary dialog for the trail.</li> <li>Open the menu and press "Upload".</li> <li>A message showing "Uploading..." should appear.</li> <li>Once uploading is complete "Upload Complete!" should be visible confirming upload success.</li> </ul>	FAIL	Sometimes fails due to the object not having been stored in the database yet.
9	Online List: <ul style="list-style-type: none"> <li>Return to the main view.</li> <li>Select the "Downloaded" tab.</li> <li>There should be no elements.</li> <li>Select the "Find New" Tab.</li> </ul>	PASS	

	<ul style="list-style-type: none"> <li>• A Toast should appear saying “Downloading List...”.</li> <li>• Once complete a Toast should appear saying “List Downloaded” and the uploaded trail should be visible.</li> </ul>		
10	<p>Summary Dialog from Find More:</p> <ul style="list-style-type: none"> <li>• Select the item that was uploaded.</li> <li>• The summary dialog should be visible.</li> <li>• Title, author, creation date, a map view with the start location, and description visible.</li> <li>• “Download” button visible.</li> <li>• No options menu should appear when pressing the physical menu button.</li> <li>• Pressing “Download” should show a message saying “Downloading...”.</li> <li>• Once downloaded the “Download Complete!” message should be visible.</li> <li>• Download button should read as “Begin”.</li> </ul>	PASS	
11	<p>Summary Dialog from Downloaded:</p> <ul style="list-style-type: none"> <li>• Return to the main tabbed list display.</li> <li>• Select the “Downloaded” tab.</li> <li>• The item that was downloaded should be visible.</li> <li>• Use the method in test 5 or 6 to open the summary dialog for the item.</li> <li>• The title, author, creation date, and description should all be seen correctly.</li> <li>• “Begin” button should be visible.</li> <li>• Pressing the physical menu button should show options for “Delete” and “Cancel”.</li> <li>• Pressing “Cancel” should close the menu.</li> <li>• Press “Delete” and the dialog should be exited back to the tabbed list view. The item should no longer be listed.</li> </ul>	PASS	
12	<p>Preparation for remaining test:</p> <ul style="list-style-type: none"> <li>• Re-download the task.</li> <li>• Return to the “Find More” tab and select the item that was downloaded again.</li> <li>• The “Begin” button and a message indicating it has already been downloaded should be visible.</li> </ul>	PASS	
13	<p>Trail Completion:</p> <ul style="list-style-type: none"> <li>• Standing outside of start point.</li> <li>• Distance to point should be shown and decrease as approaching point.</li> <li>• Move into start point.</li> <li>• Hint dialog should appear containing the first description.</li> </ul>	FAIL	Points do not correctly correspond, but the start position dialog does appear and matches when reaching the relative location.
14	Trail Completion:	PASS	

	<ul style="list-style-type: none"><li>• Pressing back should return back to the trail view.</li><li>• The single number should be visible.</li><li>• Walk to additional waypoints. At each point the hint should correctly display and the map should update.</li></ul>		
15	Trail Completion: <ul style="list-style-type: none"><li>• Pressing on waypoints in the map opens the corresponding hint.</li></ul>	PASS	
16	Trail Completion: <ul style="list-style-type: none"><li>• Reaching the end point shows the end success message.</li></ul>	PASS	

## 8 Future Work

Features that could be added to extend or improve this application include:

- Improving the user interface to provide a faster method for agile trail creation.
- More varied options for allowing users to create trails by including other media such as images or puzzles even that must be viewed or completed at way points.
- Improved network and database design that isn't quite so sloppy that minimises the data storage.
- Further inclusion of GoogleMaps support to provide additional features such as creating trails without needing to move to the locations physically.

These are just a few of the many changes that could be applied to extend this application to become a far more useful tool.

## 9 Conclusions

This project took the most time in the areas of setting up the data infrastructure and the development of the user interface. The GPS component of input was merely a few lines of code to handle the actual input although this code does not appear to work for some reason. There was a lot of processing and passing of data between objects though after the initial logging of geographical points. With more time it would have certainly been interesting to attempt to include some of the additional features listed in Future Work. On the whole it was an enjoyable and interesting project that has provided insight into a wider base of Android development technologies.

## 10 References

Using locations:

[http://www.vogella.com/articles/AndroidLocationAPI/article.html#locationapi\\_overview](http://www.vogella.com/articles/AndroidLocationAPI/article.html#locationapi_overview)

GoogleMaps Sources:

<https://developers.google.com/maps/documentation/android/hello-mapview>

<http://mobiforge.com/developing/story/using-google-maps-android>

<http://tech.truliablog.com/2012/02/23/custom-map-markers-for-android-google-maps/>