WalkAbout 1.0

Phase III Final Proposal

Ryszard Kilarski (U81-39-8560)

Table of Contents

1. Response to Phase II Comments 3

Final Requirements 3

WalkAbout Application Definitions 3

Definite Requirements 3

Nice To Do Requirements 3

Screen Shots 5

Feature Cross-Reference 9

Design 10

Screen Transition Diagram 10

Object Relationship Diagram 11

User Interface Object Relationship Diagram (High-Level) 12

User Interface Object Relationship Diagram (Detail) 13

Database Tables 13

Partial Implementation of Risks 14

Risks Associated With Requirements 14

Riskiest Elements 15

Schedule 17

Bibliography 18

2. Summary 19

3. Report on Requirements 19

D1. (Requirement Met) The App shall allow you to add a new Waypoint 19

D2. (Requirement Met) The App shall allow you to edit an existing Waypoint 19

D3. (Requirement Met) The App shall allow you to delete an existing Waypoint. 20

D4. (Requirement Met) The App shall allow you to scroll and view Waypoints. 20

D5. (Requirement Met) The App shall allow you to take photos with the device camera and add them to the Waypoint. 20

D6. (Requirement Met) The App shall allow you to tap to view a photo full screen. 20

D7. (Requirement Met) The App shall allow you to delete photos from a Waypoint. 20

D8. (Requirement Not Met) The App shall allow you to produce a map from all the photos in a Waypoint. 20

N1. (Requirement Not Met) The App shall allow you to “check in” a Waypoint using Facebook. 20

N2. (Requirement Not Met) The App shall allow you to export all photos in a Waypoint to a Facebook album. 21

N3. (Requirement Met) The App shall allow you to export all photos in a Waypoint to the device album. 21

N4. (Requirement Met) The App shall allow you to move photos between Waypoints. 21

N5. (Requirement Met) The App shall provide the ability to change the Waypoint Order By setting. 21

N6. (Requirement Met) The App shall provide the ability to change the Photo Order By setting. 21

N7. (Requirement Met) The App shall allow you to expand/collapse individual Waypoints in the list, only displaying the name/date. 21

N8. (Requirement Not Met) The App shall allow you to change the order of photos in the Waypoint. 21

4. Report on Design 21

Design That Was Used 21

Differences From Plan 22

Advantages 22

Shortcomings 22

5. More Difficult Than Expected 23

Camera 23

Maps Integration 23

6. Simpler Than Expected 23

User Interface 23

GPS Locations 23

7. Expanding App Into Something Real 23

8. Appendix 24

# Response to Phase II Comments

Separately attach your phase 2, including my comments. Respond to each of our comments with a comment of your own or an appendage to my comment.

## Final Requirements

This section contains several items: The definite and nice-to-do requirements, and screenshots of all the major functionality.

### WalkAbout Application Definitions

The WalkAbout application allows a user to take and group together photographs and perform some cool activities in order to enjoy vacation pictures and other photo-taking opportunities. Here are some application-specific definitions used in WalkAbout.

#### Waypoint

A collection of photographs defined by the user. A Waypoint can be anything the user desires; in effect, it functions like a folder. However, the Waypoint has one additional important property: it can be associated with a geographical location (for example: the Parthenon in Italy, Times Square in New York, or the Freedom Trail in Boston). Photos added to the Waypoint will then be grouped together by this gross geo-tag.

Each photo, by its nature of being taken by a GPS-enabled device, also has its own geo-tag. The application will use all of this GPS data in order to provide a richer photographic experience.

### Definite Requirements

Here is a list of definite requirements that should form the minimum functionality for WalkAbout.

1. The App shall allow you to add a new Waypoint
   1. The Waypoint will be GPS geo-tagged
   2. The Waypoint will have time, date, and a name.
2. The App shall allow you to edit an existing Waypoint
   1. Rename a Waypoint
   2. Re-geo-tag a Waypoint
   3. Change date/time
3. The App shall allow you to delete an existing Waypoint.
4. The App shall allow you to scroll and view Waypoints.
5. The App shall allow you to take photos with the device camera and add them to the Waypoint.
6. The App shall allow you to tap to view a photo full screen.
7. The App shall allow you to delete photos from a Waypoint.
8. The App shall allow you to produce a map from all the photos in a Waypoint.

### Nice To Do Requirements

Here is a list of nice-to-do requirements that will add delight to the user when using WalkAbout.

1. The App shall allow you to “check in” a Waypoint using Facebook.
2. The App shall allow you to export all photos in a Waypoint to a Facebook album.
3. The App shall allow you to export all photos in a Waypoint to the device album.
4. The App shall allow you to move photos between Waypoints.
5. The App shall provide the ability to change the Waypoint Order By setting.
6. The App shall provide the ability to change the Photo Order By setting.
7. The App shall allow you to expand/collapse individual Waypoints in the list, only displaying the name/datetime.
8. The App shall allow you to change the order of photos in the Waypoint.

### Screen Shots

This subsection contains the screenshots of all the major functionality.

#### Waypoint List

The Waypoint List is the main user interface of the application. It contains all the (collapsible) Waypoints along with all the photographs in each Waypoint. It also provides the user with the ability to add/edit Waypoints and to add photographs (either by camera or from the photo app on the device) to the Waypoints. Finally, it has a settings/tools menu that allows the user to change the order and expand/collapse waypoints.

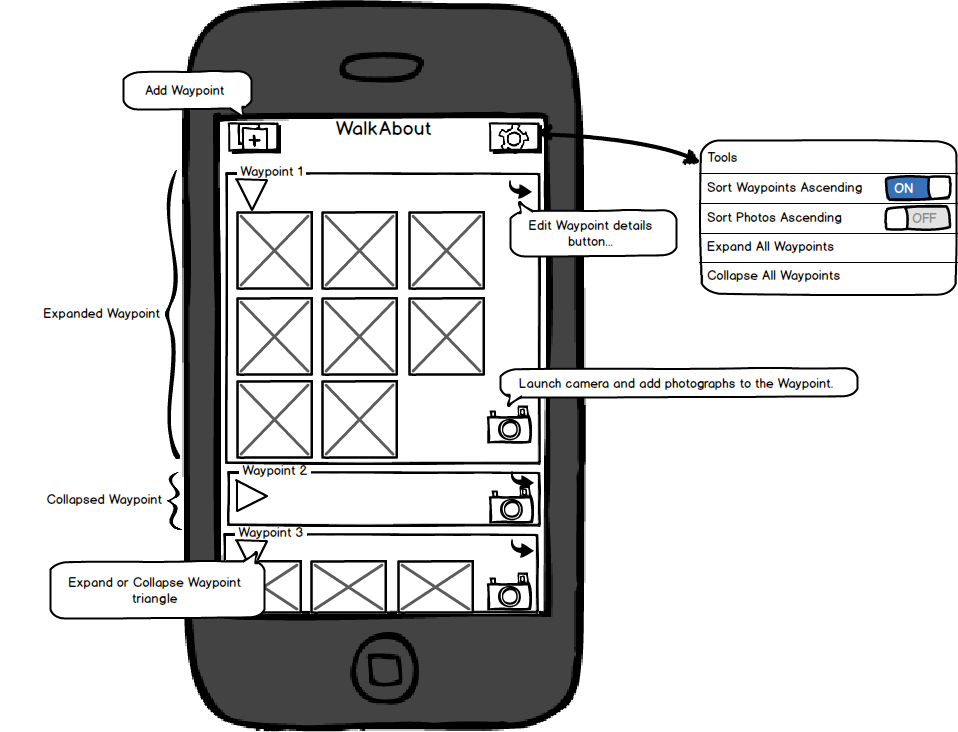


Figure The Waypoint List

#### Waypoint Detail Form (Add/Edit Form)

The add/edit form is the interface a user sees when editing or adding a Waypoint. Here, a user can name/rename/delete the Waypoint, change the date/time, update its Facebook location and its GPS geo-tag. From here, the user can also create the Google map that shows all the photos in the Waypoint.

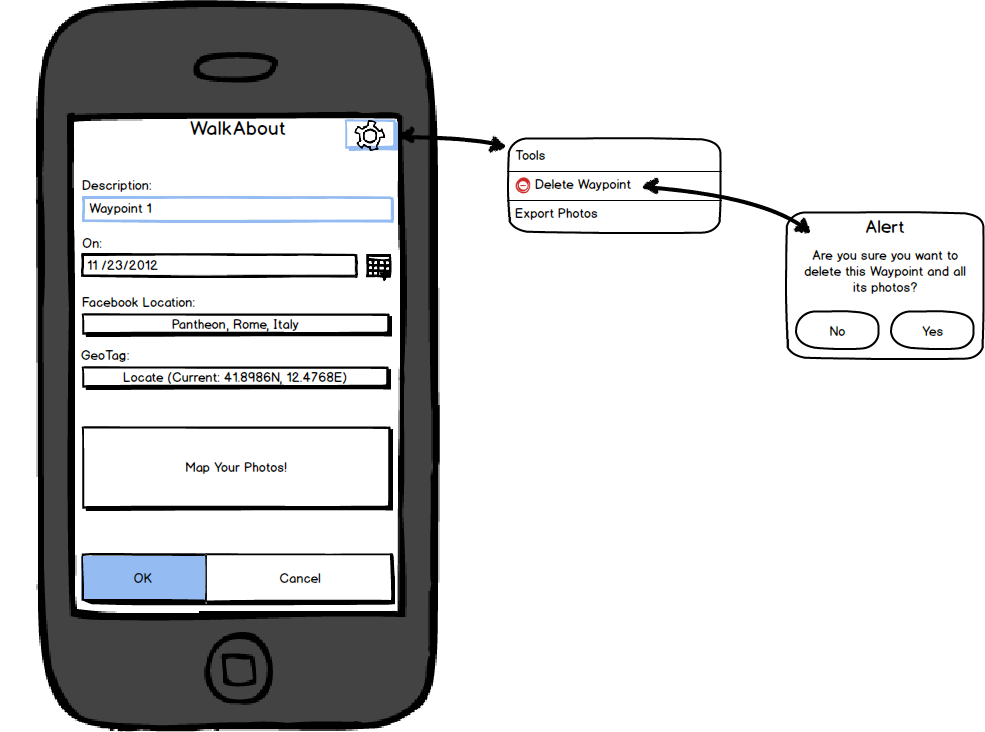


Figure The Waypoint Add/Edit Form

#### Waypoint Map Form

The Waypoint Map Form shows a map of the Waypoint area and a point of interest for each (geo-tagged) photo in the Waypoint.

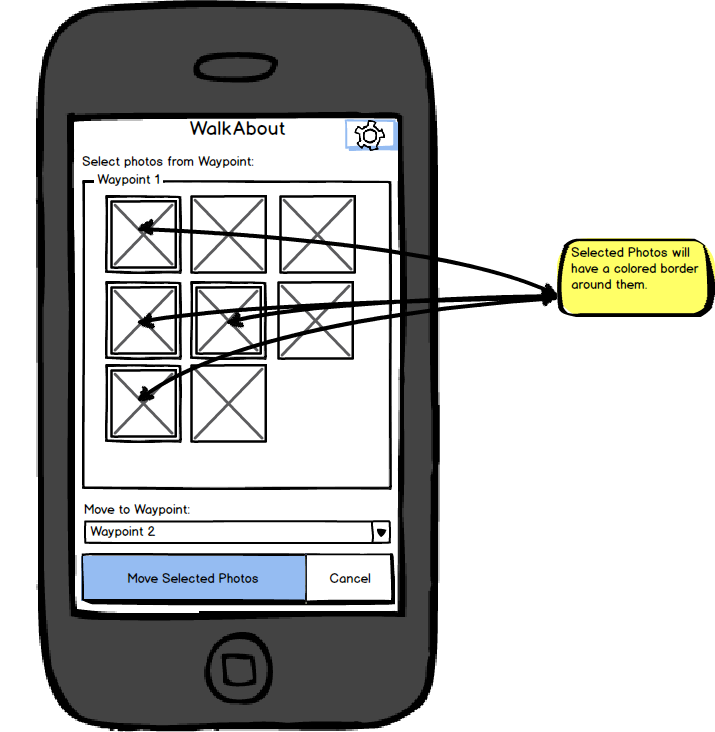
Note: If the Waypoint contains photos that are not geo-tagged, then the user will get a message saying that not all images were map-able.



Figure Waypoint Map Form

#### Waypoint Photo Move Form

The Waypoint Photo Move form shows all the photos in a particular Waypoint and allows you to select them. It then allows the user to select a target Waypoint and move those photos to that Waypoint.



## Feature Cross-Reference

Cross reference table that maps the requirements to Android features.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Android Feature |  | Requirements | | | | | | | | | | | | | | | |
|  | Definite Requirements | | | | | | | | Nice-to-have Requirements | | | | | | | |
|  | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 |
| Local Database Access | X | X | X |  | X | X | X | X | X | X | X | X | X | X | X | X |
| Camera |  |  |  |  | X |  |  |  |  |  |  |  |  |  |  |  |
| Local filesystem |  |  |  |  |  |  |  |  |  |  | X |  |  |  |  |  |
| GPS/Geo-tag | X | X |  |  |  |  |  | X |  |  |  |  |  |  |  |  |
| Cellular Connectivity |  |  |  |  |  |  |  | X | X |  |  |  |  |  |  |  |
| Accelerometer / Orientation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Touch/UI\* |  |  |  | X |  |  |  |  |  |  |  |  | X | X | X |  |
| Photo App Integration |  |  |  |  |  | X |  |  |  |  | X |  |  |  |  |  |
| Google Maps Integration |  |  |  |  |  |  |  | X |  |  |  |  |  |  |  |  |
| Facebook Integration |  |  |  |  |  |  |  |  | X | X |  |  |  |  |  |  |

\*Note: Touch/UI items have an X for those areas that cannot be tied to other Android features and are a purely user interface implementation. All the other requirements are trivially linked to Touch/UI as well.

## Design

This section explains the technical design of the application with a screen diagram, several class diagrams, and a section on database tables.

### Screen Transition Diagram

This diagram shows the flow of screens from the central Waypoint List.

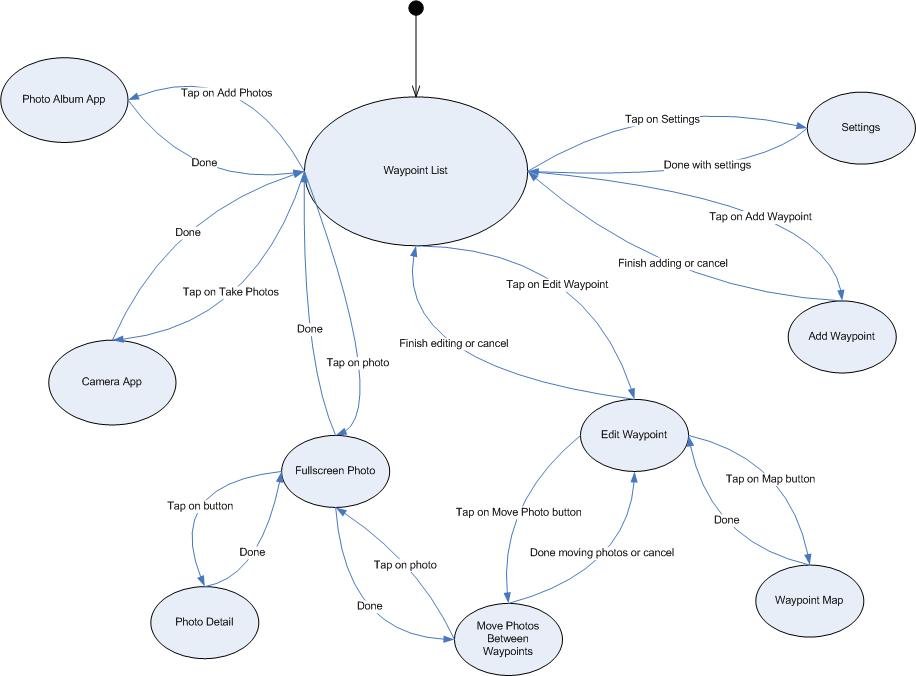


Figure Screen Transition Diagram, including Beginning State

### Object Relationship Diagram

This diagram shows the major domain and database objects. (All properties can be assumed to have their appropriate getter and setter methods.)

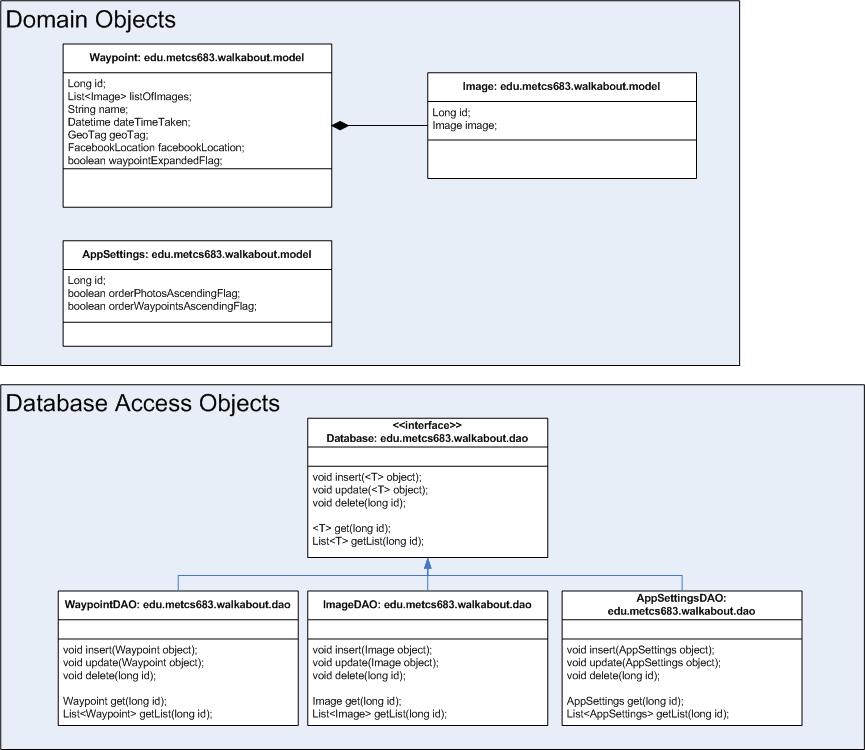


Figure Object Relationship Diagram, Domain and Database Objects

#### Package: edu.metcs683.walkabout.model

##### Object: Waypoint

The Waypoint is the main domain object, containing a list of images as well as “miscellaneous” data about the Waypoint. This includes

* The Waypoint’s unique ID in the database,
* The Waypoint’s name,
* The date/time it was created,
* Its geo-tag,
* Its Facebook location,
* Whether it is visually expanded or contracted by the user.

##### Image

The image object contains the photograph itself and a unique ID.

##### AppSettings

The AppSettings object contains the system settings for the application.

#### Package: edu.metcs683.walkabout.dao

##### Database

The Database interface contains the methods needed by all the DAO objects to load, save, and get a list of those objects.

##### WaypointDAO

This object contains the methods to load, save, and get a list of Waypoints.

##### ImageDAO

This object contains the methods to load, save, and get a list of images.

##### AppSettingsDAO

This object contains the methods to load, save, and get the Application settings object.

### User Interface Object Relationship Diagram (High-Level)

This diagram shows the high-level concept of how UI objects interface with the domain and DAO objects.

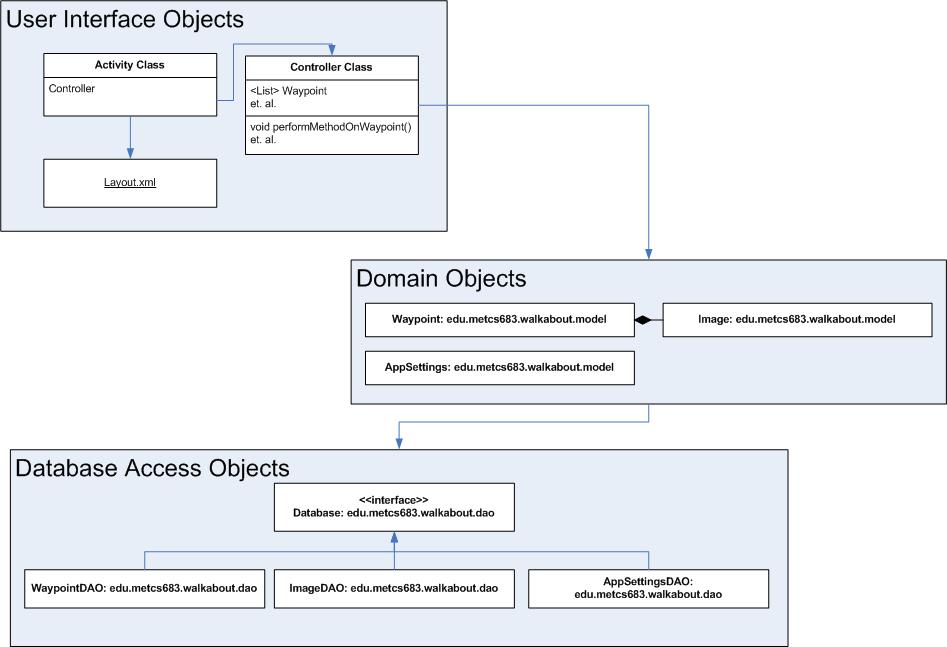


Figure Object Relationship Diagram, User Interface, High Level

The general rules on how the User Interface objects relate are as follows:

* Each Activity object references a corresponding XML file containing its layout.
  + Note that Settings is the only activity that does not reference an XML layout. This is because it will implement the SharedPreferences API.
  + Each Activity has methods (and private classes) to directly enable user interface functionality.
* Each Activity also has a corresponding controller.
  + The controller maintains the appropriate domain objects and exposes public methods that are used by the Activity to maintain the domain objects.
  + The controllers do not perform any UI work, nor do they know anything about the database/DAO. They simply maintain the domain objects and provide methods for the UI to do its business.
* The domain objects have private access to the database via the DAO objects. The rest of the application (the UI and controller) does not know about the database objects.

### User Interface Object Relationship Diagram (Detail)

This diagram shows the major UI objects and how they interface with the domain and DAO objects. (All properties can be assumed to have their appropriate getter and setter methods.)

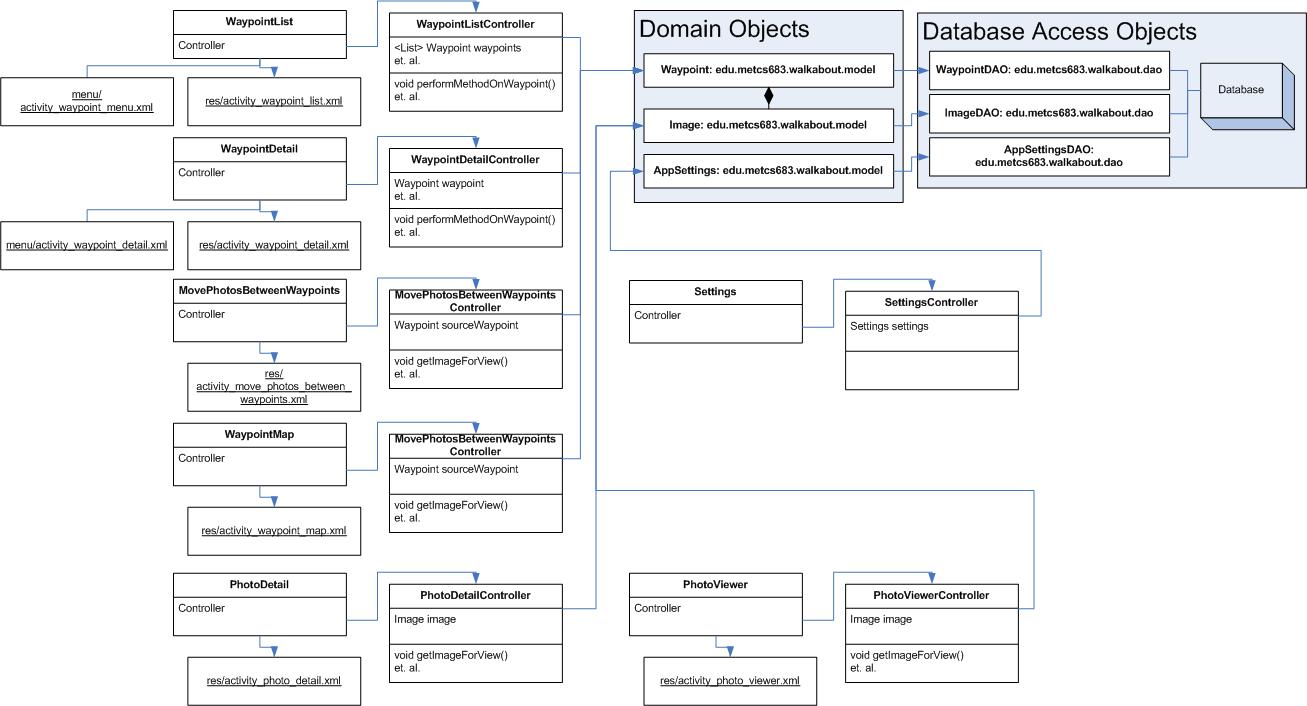


Figure Object Relationship Diagram, User Interface, Detail

This diagram shows the full relationship structure between the Activity, the layout, the menus, the controller, and the domain and DAO.

### Database Tables

This section lists the Sql tables needed for WalkAbout.

Note: Some data types are not yet known, they will be resolved later.

#### Table: WAYPOINT

Description: Contains each waypoint.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Indexed | Description |
| ID | Long | Y | Primary Key |
| DESCRIPTION | String |  | The Waypoint description. |
| DATE | Date |  | The date/time the user assigned to the Waypoint. |
| GEOTAG | String(?) |  | The GPS geo-tag of this waypoint. |
| FACEBOOK\_LOCATION | String(?) |  | The Facebook location assigned by the user to this Waypoint. |
| EXPANDED\_FLAG | Boolean |  | Is this Waypoint displayed as expanded or contracted? |

#### Table: IMAGE

Description: Contains each image.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Indexed | Description |
| ID | Long | Y | Primary Key |
| IMAGE\_DATA | Binary |  | The image |

#### Table: WAYPOINT\_IMAGE\_XREF

Description: Contains the join table for waypoints and images. This table maintains the assignments to which waypoint a particular image belongs.

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | Datatype | Indexed | Description |
| ID | Long | Y | Primary Key |
| IMAGE\_ID | Long | Y | Foreign Key into IMAGE |
| WAYPOINT\_ID | Long | Y | Foreign Key into WAYPOINT |

#### Settings

The Settings for Waypoint will be stored via the built-in SharedPreferences object for Android as described in (Android in Action (Third Edition), pp. 131-136) and not in a Sql table.

The data stored for Settings will include:

* In what order the Waypoints are sorted (ascending or descending),
* In what order the photos in each Waypoint are sorted (ascending or descending).

## Partial Implementation of Risks

### Risks Associated With Requirements

Definite Requirements Risks (with higher risks first)

* D8: Integration with Google Maps
* D5: Integration with camera

Nice To Do Requirements Risks (with higher risks first)

* N1: Facebook “check-in” integration
* N2: Facebook album integration

### Riskiest Elements

#### Risk 1: Time and Knowledge

##### Description

The highest risk in this project is in getting everything done and at the same time learning all I need to in order to do so. So there will be times where I’m reading chapters from the book (probably not in the syllabus sequence) and immediately implementing them in the code, in order to implement the requirements in the proper order.

##### Risk Retirement Plan

The best I can do is in the next section: create a schedule by week on what is needed to be done. Following the schedule, the time to implement the requirements can be managed so that the highest priority items can be delivered in a working product, even if many of the “nice to haves” fall off the list.

#### Risk 2: Google Maps Integration

##### Description

I was unsure of the Maps integration and if I could get the software to do what I wanted it to do. That is, put a marker down on a map for each photo in a Waypoint. In this way, I can create a map for a Waypoint and show the user all the places they took a photo.

##### Risk Retirement Plan

This document (Google Maps Android API v2 Markers) describes how to create multiple markers on a map. I can even put images in the map itself!

And here (Google Maps Android v1 API MapView Tutorial) is a full tutorial on the MapView object. Except for the geo location used, it will be almost exactly what I can use for this project.

#### Risk 3: Photo App Integration

##### Description

The App requires calling the photo application and allowing the user to take one or more photos which will then be imported into the Waypoint.

##### Risk Retirement Plan

I reviewed (Android in Action (Third Edition), pp. 267-272) and found almost exactly the code I need in order to get images into the application.

#### Risk 4: Facebook Integration

##### Description

Similarly, I was unsure how to integrate Facebook functionality into the app, in order to associate a Waypoint with a Facebook Check-in location and in order to export photos to a Facebook album.

##### Risk Retirement Plan

Researching these nice-to-have items, I am become convinced that I will not have time to implement them. I will keep it on the plan as a hopeful goal, but will not count it as a feature unless I have time. So the retirement of this risk is to remove it from the list of “to-dos”.

## Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| Week | Phase | General Description | Milestone at end of week |
| Feb 25 | Design | Phase II Proposal completed. | Design is done.  **Delivery of Phase II Documents on Friday, March 1** |
| March 4 | Coding | Domain Objects and DAO infrastructure   * Domain objects |  |
| March 11 | Coding | Domain Objects and DAO infrastructure   * DAO objects and integration with domain objects | Infrastructure ready to fulfill all definite requirements. |
| March 18 | Coding | User interface and navigation.   * Forms and Intents | All Intents created, major navigation works. |
| March 25 | Coding | User interface and must-have requirements.   * Camera and database integration * Add functionality | Ability to take photos, add to Waypoint and save to database. |
| April 1 | Coding | User interface and must-have requirements.   * GPS and Google integration * Edit functionality | Ability to geo-tag and edit Waypoint details.  **Definite requirements met.** |
| April 8 | Coding | User interface and nice-to-have requirements.   * Export functionality * Move functionality * Facebook integration(?) | Ability to export and move photos.  Ability to integrate with Facebook. |
| April 15 | Coding | User interface and nice-to -have requirements.   * Settings and sorting functionality | **Nice-to-do requirements met or retired.** |
| April 22 | QA | Bug fixing (and any spillover from nice-to-have requirements that were almost finished) |  |
| April 29 | Panic | Additional bug fixing | ZERO bugs! |
| May 2 | Spit & Polish | Final rush! | **Delivery on Thursday, May 2, 6pm.** |

# Bibliography

Ableson, W. F. (2012). *Android in Action (Third Edition).* Shelter Island: Manning Publications Co.

*Google Maps Android API v2 Markers*. (n.d.). Retrieved from https://developers.google.com/maps/documentation/android/marker

*Google Maps Android v1 API MapView Tutorial*. (n.d.). Retrieved from https://developers.google.com/maps/documentation/android/v1/hello-mapview

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **D** | **C-** | **C+** | **B-** | **B+** | **A** |  |
| **1. Clarity** | Disorganized or hard-to-understand | | Satisfactory but some parts of the submission are disorganized or hard to understand | Generally organized and clear | Very clear, organized and persuasive presentation of ideas and designs | Exceptionally clear, organized and persuasive presentation of ideas and designs | A |
| **2. Technical Soundness** | Little understanding of, or insight into material technically | | Some understanding of material technically | Overall understanding of much material technically | Very good overall understanding of technical material, with some real depth | Excellent, deep understanding of technical material and its inter-relationships | A |
| **3. Thoroughness & Coverage** | Hardly covers any of the major relevant issues | | Covers some of the major relevant issues | Reasonable coverage of the major relevant areas | Thorough coverage of almost all of the major relevant issues | Exceptionally thorough coverage of all major relevant issues | B+ |
| **4. Relevance** | Mostly unfocused | Focus is off topic or on insubstantial or secondary issues | Only some of the content is meaningful and on topic | Most or all of the content is reasonably meaningful and on-topic | All of the content is reasonably meaningful and on-topic | All of the content is entirely relevant and meaningful | A- |

# Summary

The WalkAbout application turned out very well. I was able to implement all definite requirements (except one, which is explained below) and was able to additionally implement most of the nice-to-have requirements and “delighter” features.

This application was a good introduction to Android programming because it allowed me to exercise some of the Android-specific features (GPS, maps, camera, database, filesystem, reuse existing Android functionality) without any extraneous dependencies (like writing web services or dealing with hand-built graphics). Also, I am an iPhone user, so this app showed me some interesting differences between iOS and Android.

# Report on Requirements

Here is presented the feature-cross reference table in order to mark which requirements were met and which were not.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Android Feature |  | Requirements | | | | | | | | | | | | | | | |
|  | Definite Requirements | | | | | | | | Nice-to-have Requirements | | | | | | | |
|  | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 |
| Local Database Access | X | X | X |  | X | X | X | X | X | X | X | X | X | X | X | X |
| Camera |  |  |  |  | X |  |  |  |  |  |  |  |  |  |  |  |
| Local filesystem |  |  |  |  |  |  |  |  |  |  | X |  |  |  |  |  |
| GPS/Geo-tag | X | X |  |  |  |  |  | X |  |  |  |  |  |  |  |  |
| Cellular Connectivity |  |  |  |  |  |  |  | X | X |  |  |  |  |  |  |  |
| Accelerometer / Orientation |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Touch/UI\* |  |  |  | X |  |  |  |  |  |  |  |  | X | X | X |  |
| Photo App Integration |  |  |  |  |  | X |  |  |  |  | X |  |  |  |  |  |
| Google Maps Integration |  |  |  |  |  |  |  | X |  |  |  |  |  |  |  |  |
| Facebook Integration |  |  |  |  |  |  |  |  | X | X |  |  |  |  |  |  |
| Requirement Met? | Y | Y | Y | Y | Y | Y | Y | N | N | N | Y | Y | Y | Y | Y | N |

## D1. (Requirement Met) The App shall allow you to add a new Waypoint

The easiest feature to implement, this allows the user to enter a Waypoint, GPS tag it to the current location, and save the Waypoint to the database. See **Appendix, Figure 2** to see the add functionality. This functionality is accessed from the main application bar “Add” icon shown in **Figure 1**.

## D2. (Requirement Met) The App shall allow you to edit an existing Waypoint

Reusing the functionality from D1, the ability to load and edit a Waypoint (including re-GPS tagging) was a simple extension of that requirement. See **Appendix, Figure 10** to see the edit functionality. This functionality is accessed from the Waypoint-specific menu shown in **Figure 14**.

## D3. (Requirement Met) The App shall allow you to delete an existing Waypoint.

Deleting a Waypoint from the database is just as important as adding one, completing the Add/Update/Delete lifecycle of a Waypoint. See **Appendix, Figure 13** for the confirmation screen for deleting a Waypoint. This functionality is accessed from the Waypoint-specific menu shown in **Figure 14**.

## D4. (Requirement Met) The App shall allow you to scroll and view Waypoints.

The list of Waypoints must be a scrollable area, allowing the user to see all Waypoints. See **Appendix, Figure 8** to see the scrollable list.

## D5. (Requirement Met) The App shall allow you to take photos with the device camera and add them to the Waypoint.

Each Waypoint has an associated Camera icon in order to take and add photographs to that waypoint. This was my first frustrating experience with Android: the camera app allows you to *either* return a very low-quality thumbnail image and its associated URI, *or* save a full-quality large image and not return the URI, but not both. I chose the latter, but I had to implement an Observer pattern to get the needed URI for that image. Additionally, while the requirement is met, you can see that the camera application returns back to WalkAbout after each photo rather than staying in the camera. To implement the camera properly (stay in the camera until the user is done taking multiple photos) would have required significantly more code, but I considered this a nice-to-do for the purposes of this project.

See **Appendix, Figure 8** to see all the Waypoint-specific camera buttons to take photos for each Waypoint.

## D6. (Requirement Met) The App shall allow you to tap to view a photo full screen.

Because of the ability to call existing Android applications, I was able to call the system photo viewer (with all its functionality like the zoom buttons) rather than have to write my own. See **Appendix, Figure 17** to see the standard Android image viewer.

## D7. (Requirement Met) The App shall allow you to delete photos from a Waypoint.

This feature was implemented in two ways. 1) The Waypoint menu has a “Delete Photos” item that allows the user to delete multiple photos from the system(as seen in **Appendix Figure 19**), or 2) a long-click on a photo allows you to delete that photo (**Appendix, Figure 13)**.

## D8. (Requirement Not Met) The App shall allow you to produce a map from all the photos in a Waypoint.

This requirement was not met, but not for lack of trying. Map integration was listed as a risk and it delivered. The main issue is that the v2.0 Maps does not work in the Android Emulator and I do not have any other Android device. Web searches are unclear at this point, but the only conclusive evidence that I found showed it does not work.

In the end, if you can’t see it, you cannot debug it or make it fully work, which made this requirement impossible to implement for this project.

## N1. (Requirement Not Met) The App shall allow you to “check in” a Waypoint using Facebook.

This nice-to-have was not implemented due to time constraints.

## N2. (Requirement Not Met) The App shall allow you to export all photos in a Waypoint to a Facebook album.

This nice-to-have was not implemented due to time constraints.

## N3. (Requirement Met) The App shall allow you to export all photos in a Waypoint to the device album.

This requirement was met in a roundabout way. It turns out that the photos that are taken are saved already in the device filesystem in the correct area to be accessible by the device album software. So I met this requirement by accident. See **Appendix, Figure 18** to see the Waypoint album in the standard Gallery Android application.

## N4. (Requirement Met) The App shall allow you to move photos between Waypoints.

While moving photos between waypoints is a nice-to-have, it rounds out the WalkAbout functionality because it allows for fixing accidental photos taken in the wrong Waypoint. While the user interface is more complicated (it needs a selectable list of photos and a target list of waypoints), the database activity is quite simple, simply updating the Waypoint that each photo is assigned (this is because of the design of the photo database). See **Appendix, Figure 11** and **Figure 12** to see the screen allowing you to move photos between Waypoints.

## N5. (Requirement Met) The App shall provide the ability to change the Waypoint Order By setting.

This functionality allows the user to change the order of Waypoints as displayed on the screen, and it persists in the application settings. So the application remembers this setting between executions.. See **Appendix, Figure 15**.

## N6. (Requirement Met) The App shall provide the ability to change the Photo Order By setting.

This functionality allows the user to change the order of photos as displayed on the screen, and it persists in the application settings. So the application remembers this setting between executions. See **Appendix, Figure 15**.

## N7. (Requirement Met) The App shall allow you to expand/collapse individual Waypoints in the list, only displaying the name/date.

The main display with expand/collapse functionality was where I spent a large amount of time. Each Waypoint can be collapsed to just its title/date bar, or expanded to see all photos directly in the main list. The choice of expanded/contracted is also persisted into the database so the application remembers the configuration of what the user wants to see. See **Appendix, Figure 16** for an example of collapsed and expanded Waypoints.

## N8. (Requirement Not Met) The App shall allow you to change the order of photos in the Waypoint.

This nice-to-have was not implemented due to time constraints.

# Report on Design

## Design That Was Used

The proposed technical design was what was mostly used. There were a few tweaks and additions for technical use and to better implement the peculiarities of the Android way of doing things.

### Differences From Plan

#### Database

The database differences were minimal. The most significant change was to store the image URI and to store the image itself on the filesystem, rather than store the image in the database. This was for both performance (serializing images to the database would be more expensive) and to learn about filesystem functionality.

Along with that change, the WAYPOINT\_IMAGE\_XREF table was eliminated and instead a WAYPOINT\_ID column was created in the IMAGE table.

The other minor change was to save the GPS location as two discrete columns (latitude and longitude) rather than a single GeoPoint column.

#### Data Access Object (DAO) Layer

The DAO layer was implemented as designed, though the Database parent interface was turned into a class in order to take on the database creation/upgrade responsibility for all the child classes.

#### User Interface (UI) Layer

The UI layer was mostly implemented as designed, though the Screen Transition Diagram became simpler because I incorporated more functionality into the Waypoint-specific menu. The code follows the XML File/Controller/Activity pattern for each form. The PhotoDetail and PhotoViewer items were not needed, I instead rely on the Android system photo viewer.

##### WaypointView

The WaypointView class is an exception. I needed to encapsulate the functionality of the waypoint title, date, photo grid, camera button, and waypoint-specific menu all in a reusable view that would then be added in the Waypoint list user interface for each Waypoint. So the Waypoint list is a collection of these WaypointView objects in a scrollable view.

## Advantages

Using the filesystem and URIs for images allows several performance gains.

1. The system can load a thumbnail-sized image into memory, rather than the full image from the database.
2. The database read/writes are of simple text rather than BLOBs.

Additionally, when we collapse a Waypoint in the UI, we release the memory held by the grid of that Waypoint. When we expand, we re-read the thumbnails back into memory.

## Shortcomings

I chose the simpler of the paths on several items.

* The camera handler should be a more significant class that allows the user to take multiple photos at once.
* The GPS functionality could be more robust and save all the information (time, altitude, etc) that it reports rather than just the latitude and longitude, and in a more friendly manner than displaying just the latitude/longitude (for instance, it could instead display a human-understandable address or Facebook Checkin location).

# More Difficult Than Expected

## Camera

As stated earlier, the out-of-the-box camera functionality is frustrating to work with. Either you have to “code around” the shortcomings (by implementing an Observer pattern so that the container can get the image URI that the camera uses to save the image), or write a camera-handling class from the ground-up.

## Maps Integration

The Maps Integration was my biggest headache. In the end, I could not get it to work because I found that the v2.0 Maps functionality does not work in the Android Emulator, the Google documentation is very confusing and haphazard, and I did not have a true Android device on which to run in order to perform proper development for this feature.

# Simpler Than Expected

## User Interface

The XML-based layouts proved to be significantly easier to use than expected. I really liked them and the ability to not even see in the code anything that was pure UI-related.

## GPS Locations

Implementing the GPS location software was quite easy, even though it does not function as expected in the Android Emulator

# Expanding App Into Something Real

There are several items to be done before this app can be considered a commercial app.

* Facebook integration—Any current photo app should have the ability to integrate with Facebook.
* Filesystem work—Photos should be saved under Waypoint-specific subdirectories, so that they look good when accessed via the Android Gallery application. Currently, all WalkAbout photos appear under the same “WalkAbout” album. This will also involve updating the WalkAbout database if a user deletes a photo using the Gallery, to keep them in synch.
* Maps Integration—The killer feature of WalkAbout is the ability of seeing each of your photos on a map, where the photograph was taken. This was unfortunately not finished for this project.

<http://www.techsmith.com/jing.html>

# Appendix

Figure Walkabout List, note the Add and Settings icons in the main action bar and the Waypoint-specific camera button and waypoint menu buttons on each waypoint. In this example, only one photo is in each Waypoint.

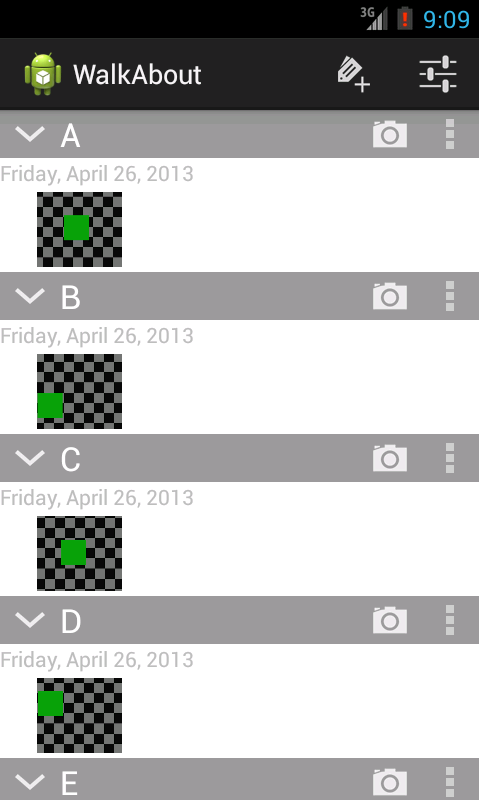


Figure Waypoint Add Screen

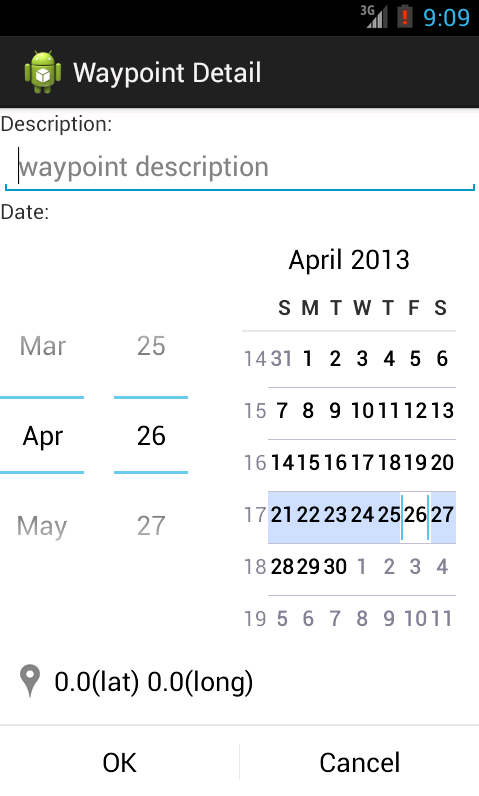


Figure Waypoint Edit Screen

## 

Figure Waypoint Move Screen

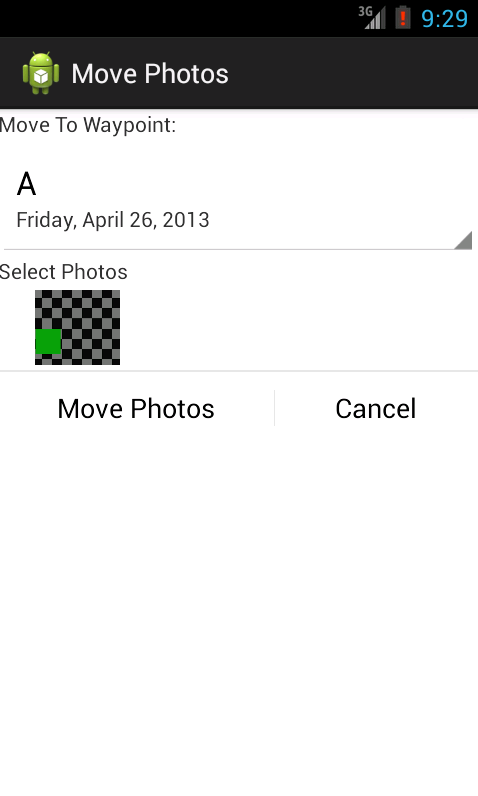


Figure Move Photos Between Waypoints List

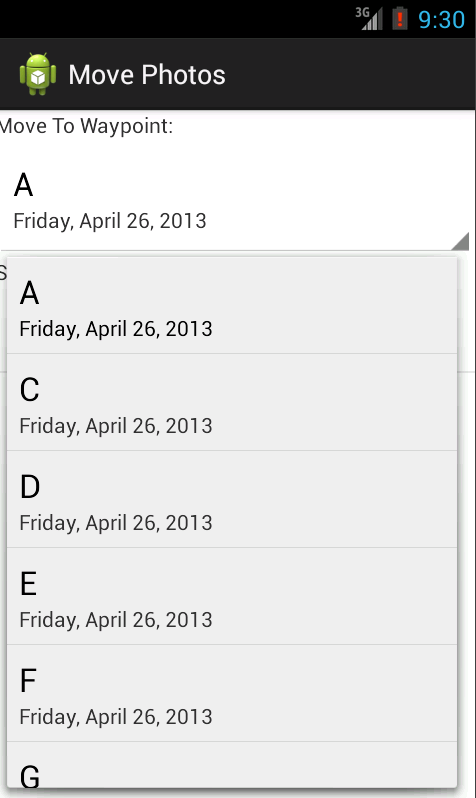


Figure Delete Waypoint Confirmation Screen

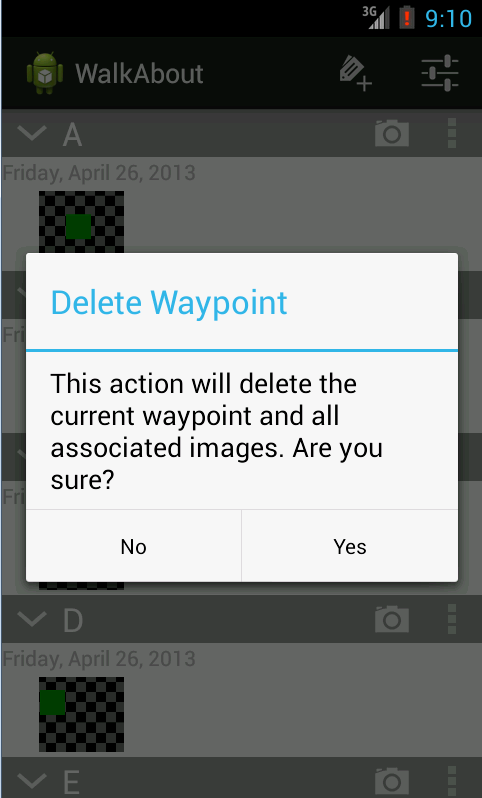


Figure Waypoint-Specific Menu

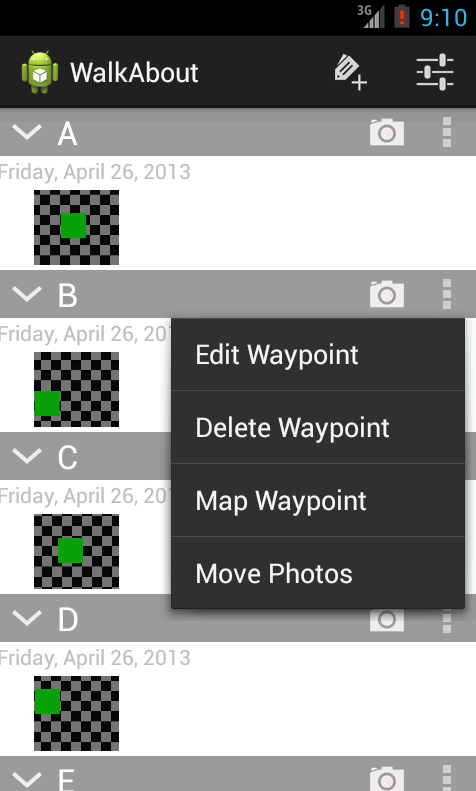


Figure System Options for Waypoint and Photo Order

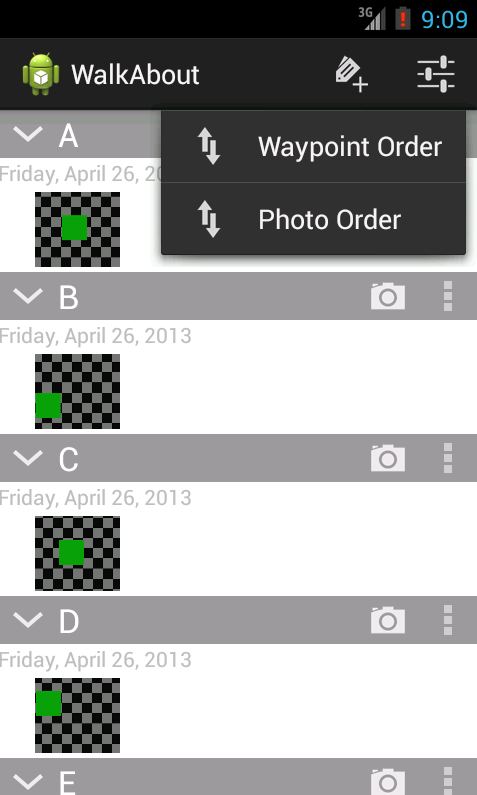


Figure Examples of Expanded and Collapsed Waypoints

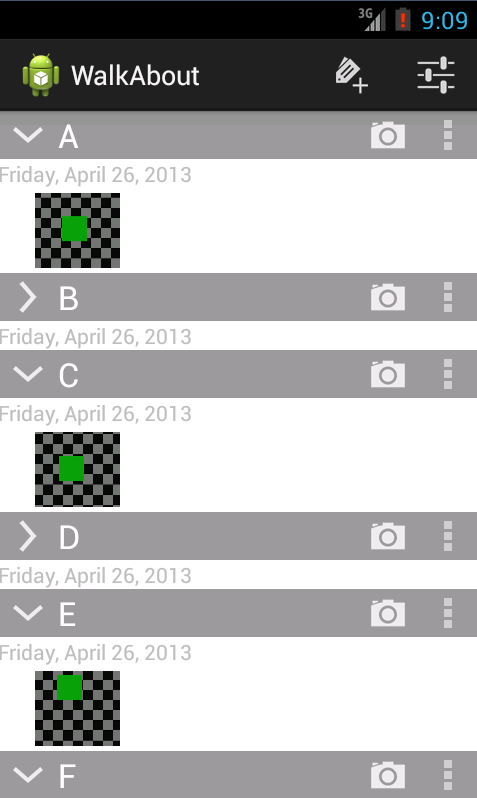


Figure Full Screen Photo View Functionality

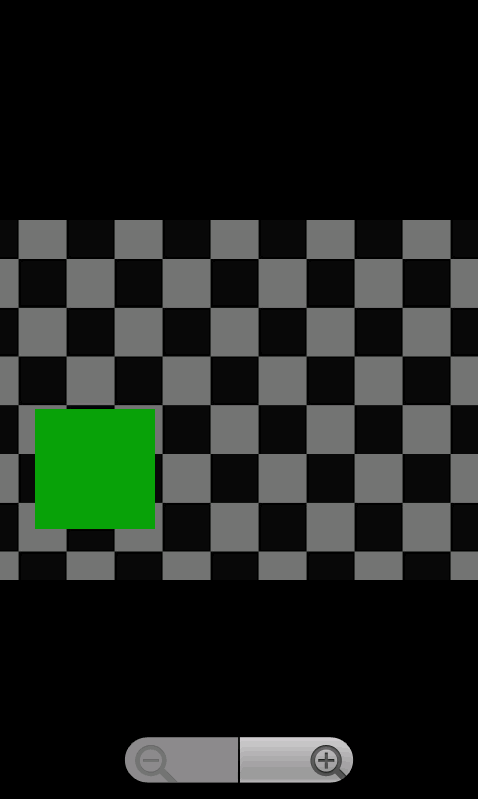


Figure Android Gallery with WalkAbout album

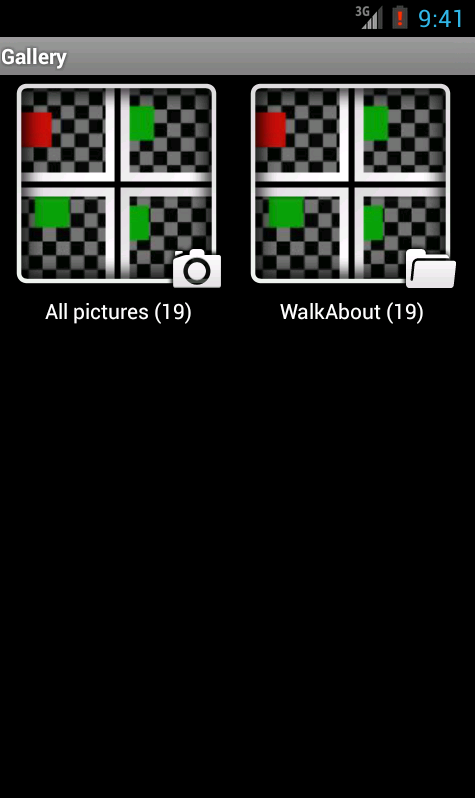


Figure: Multi-photo delete Functionality