WalkAbout 1.0

Phase III Final Proposal

Ryszard Kilarski (U81-39-8560)

Table of Contents

[1. Response to Phase II Comments 2](#_Toc354664854)

[Final Requirements 2](#_Toc354664855)

[WalkAbout Application Definitions 2](#_Toc354664856)

[Definite Requirements 2](#_Toc354664857)

[Nice To Do Requirements 2](#_Toc354664858)

[Screen Shots 4](#_Toc354664859)

[Feature Cross-Reference 8](#_Toc354664860)

[Design 9](#_Toc354664861)

[Screen Transition Diagram 9](#_Toc354664862)

[Object Relationship Diagram 10](#_Toc354664863)

[User Interface Object Relationship Diagram (High-Level) 11](#_Toc354664864)

[User Interface Object Relationship Diagram (Detail) 12](#_Toc354664865)

[Database Tables 12](#_Toc354664866)

[Partial Implementation of Risks 13](#_Toc354664867)

[Risks Associated With Requirements 13](#_Toc354664868)

[Riskiest Elements 14](#_Toc354664869)

[Schedule 16](#_Toc354664870)

[Bibliography 17](#_Toc354664871)

# 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.

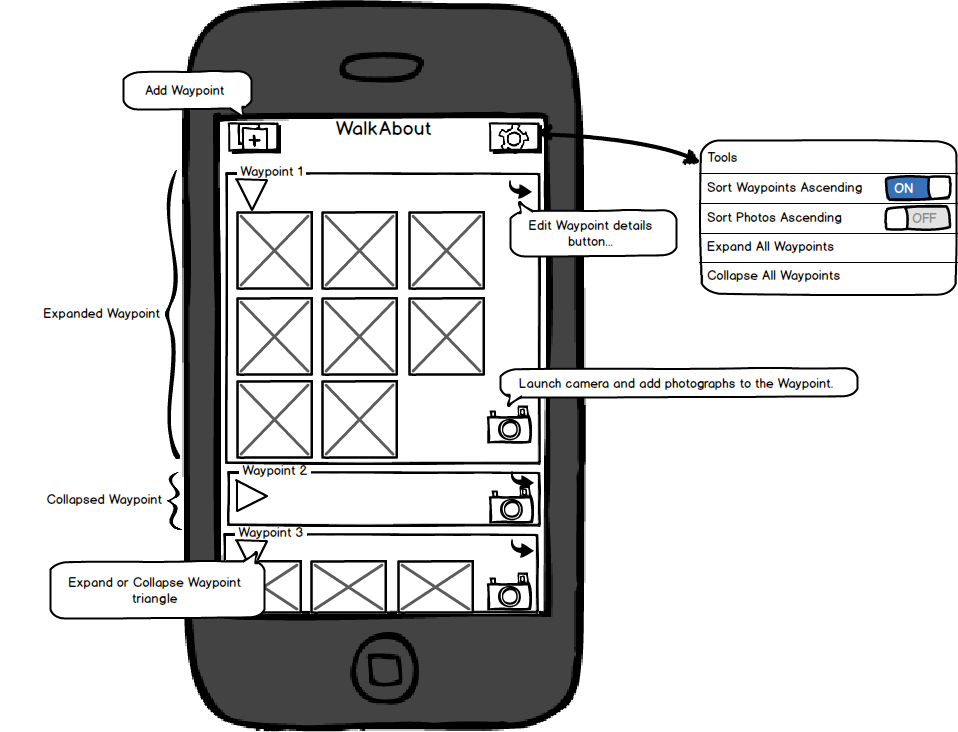


Figure The Waypoint List

#### 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.

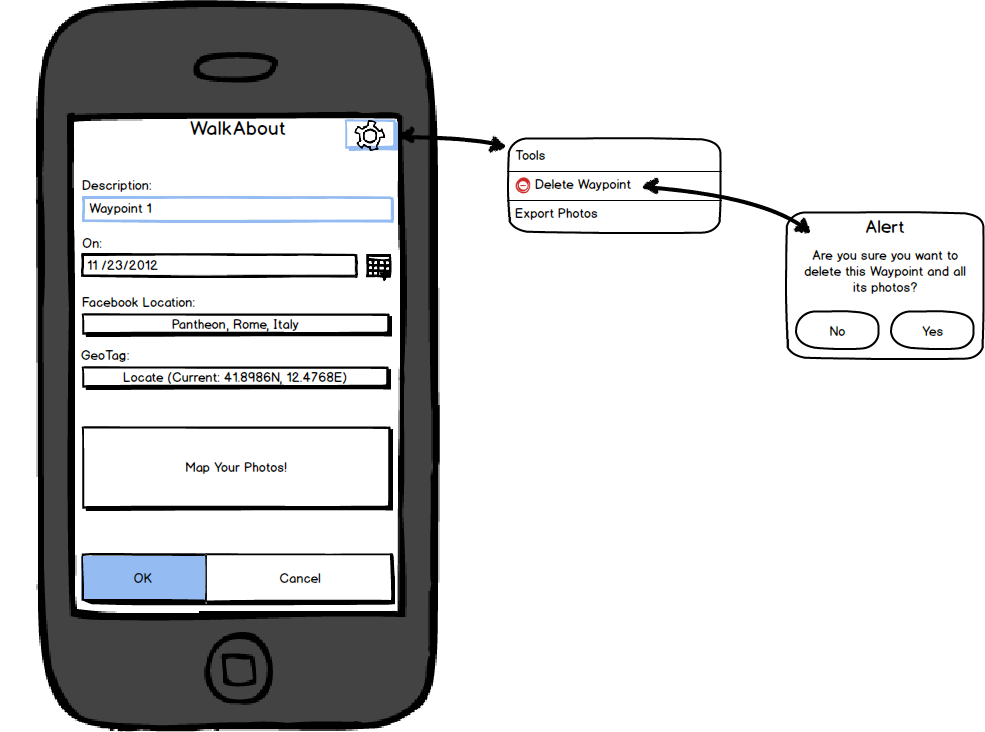


Figure The Waypoint Add/Edit Form

#### 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.

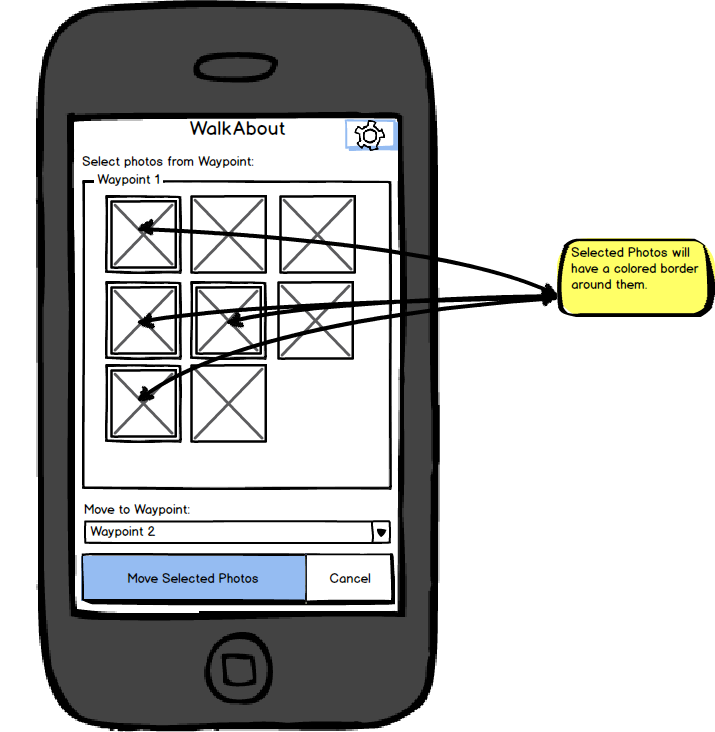
Note: If the Waypoint contains photos that are not geo-tagged, then the user will get a message saying that not all images were map-able.



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.

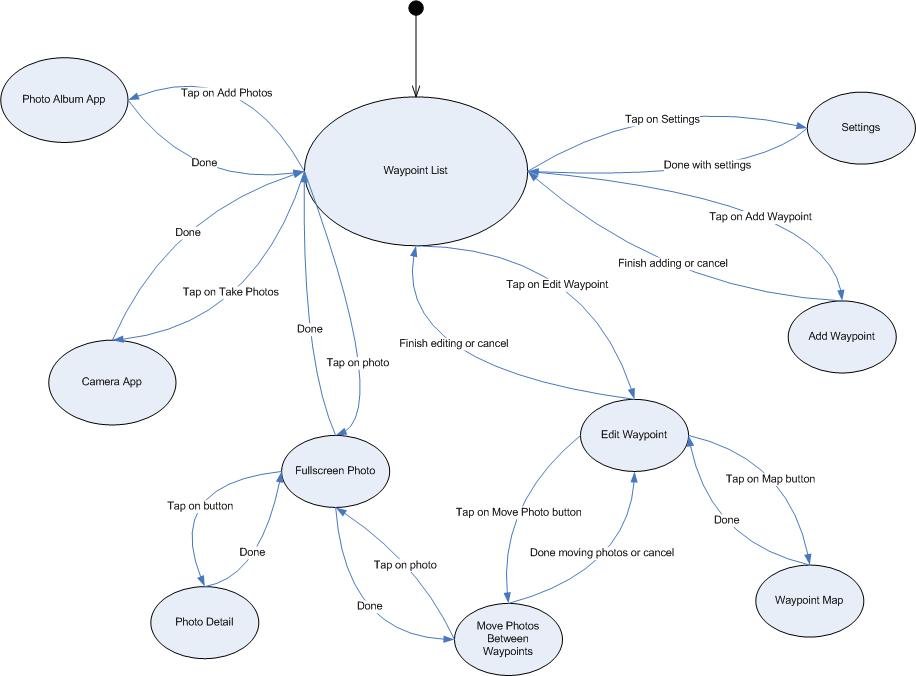


Figure Screen Transition Diagram, including Beginning State

### 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.)

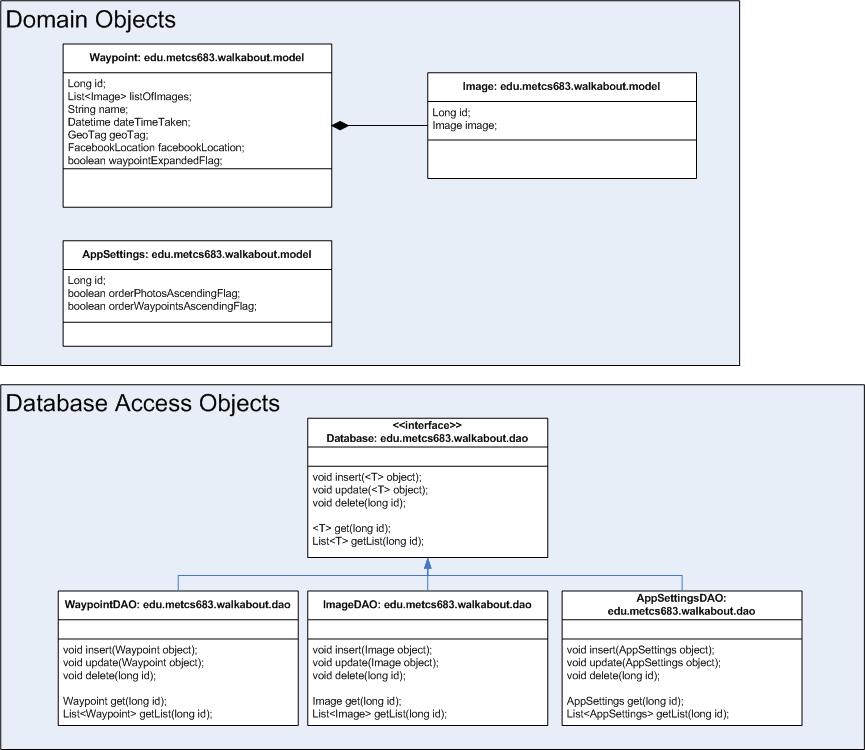


Figure Object Relationship Diagram, Domain and Database Objects

#### 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.

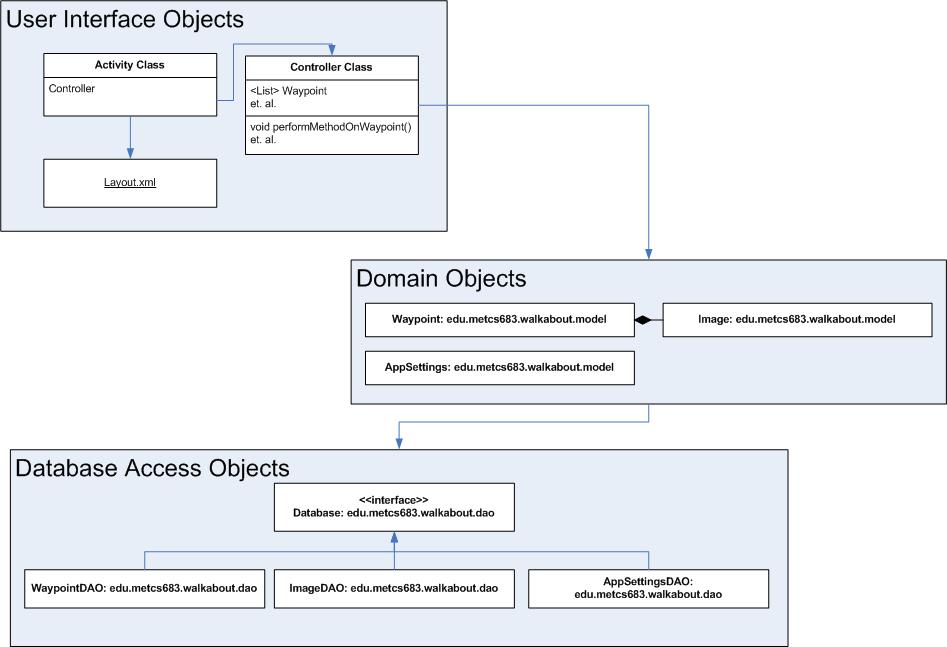


Figure Object Relationship Diagram, User Interface, High Level

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.)

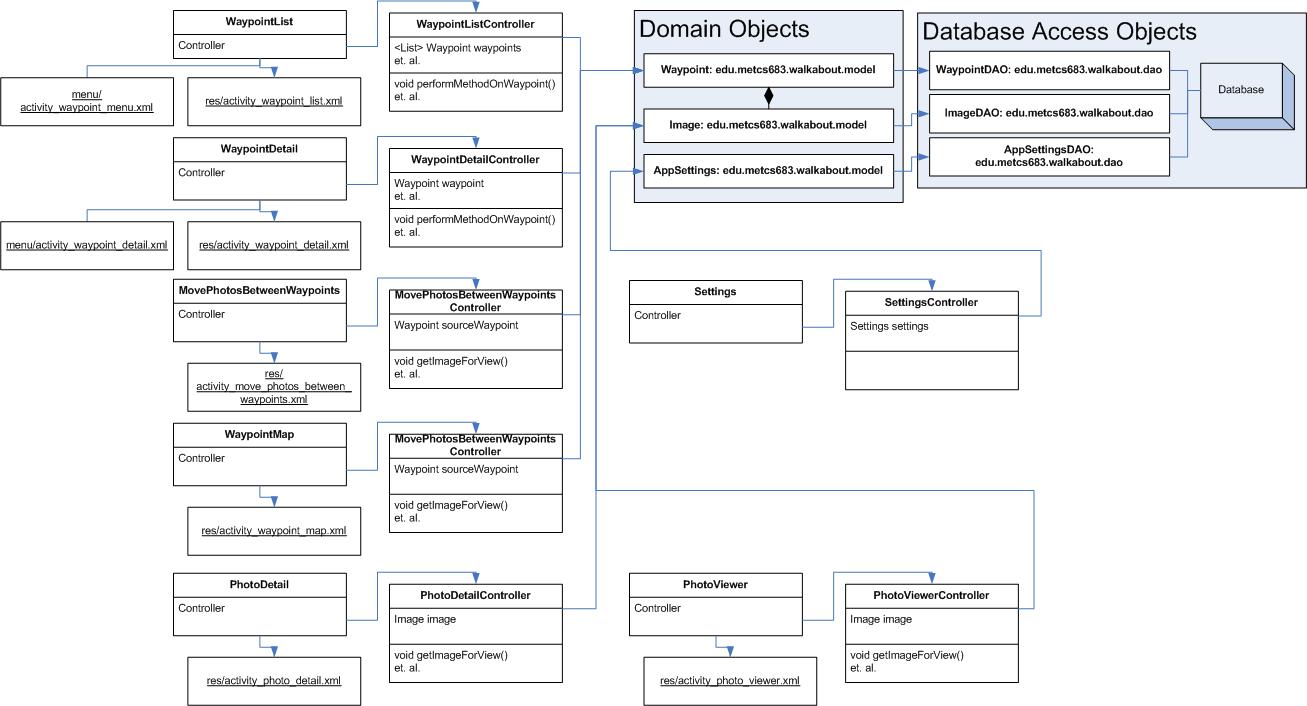


Figure Object Relationship Diagram, User Interface, Detail

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

Phase III consists of a report, as well as a recorded demonstration, both described below. Please use the numbering and headings shown, and be sure to observe the evaluation criteria (including the last one, on the use of references). Excluding appendices, and Sections 2, 8, and 9, the report should not exceed 5 pages.

Summarize the outcome of your project in a paragraph or two. Avoid duplication, though, especially with Section 3.

# Report on Requirements

Describe the extent to which you accomplished each definite requirement Dx, as well as any other fulfilled requirements. For each, include 1-4 sentences and screenshot, as appropriate. Note that there may be significant credit for an explanation of why a particular goal turned out not to be realizable. Your effectiveness depends on how much you show you learned.

# Report on Design

Describe the design that you finally used. Indicate how, where, and why it differed from your planned design. Describe clearly the design's advantages and its shortcomings.

5. Describe what you found more difficult than expected to implement and explain this.

6. Describe what you found as simpler than expected to implement, and explain.

7. With justification, describe what would be involved for this app to expand into something real.

8. Demonstration

Prepare, record, and separately attach a demonstration. This should last no more than 5 minutes.

9. SOURCE

Separately attach source and input in zipped form.

Jing download for free. It allows you to upload your video to their server and then provides the URL to access it. It's very easy to use.

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