Copyright

by

Anh Luong

2014

HIDDEN TEXT: Optional—If you do not include a copyright page, delete entire page and the following page break.

HIDDEN TEXT: NOTE: this page in hard copy with all original signatures must be submitted with the dissertation to the Graduate School; this is required whether the document is in electronic format or on paper. Whereas, the page included in the electronic document will be unsigned unless it is scanned in.

The Report Committee for Anh Luong

Certifies that this is the approved version of the following report:

ClosetStylist: An Android app to manage closet and programmatically pick out the outfit

**APPROVED BY**

**SUPERVISING COMMITTEE:**

|  |
| --- |
| **Supervisor:** |
| Adnan Aziz |
| Christine Julien |

HIDDEN TEXT: The top line is for the Supervisor’s signature. There should be as many lines as there are members on the committee. Lines must be solid, not dotted. To delete signature lines, select the line you want to delete, go to the Table menu, select Table Properties, click on the Table tab, and click on the Borders and Shading button, then remove the bottom border of the table. Use the professor's name without titles or degrees

ClosetStylist: An Android app to manage closet and programmatically pick out the outfit

by

Anh Luong, B.S.E.E.

HIDDEN TEXT: Given first name, and previous academic degrees (B.A. or higher) B.A., B.S., etc. Your official name is the name which appears on your UT transcript.

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

HIDDEN TEXT: The degree sought must be worded in the form given in the Graduate Catalog, such as Doctor of Philosophy, Doctor of Musical Arts, Doctor of Education.

The University of Texas at Austin

May 2014

Dedication

To my parents and my wife.

Acknowledgements

First of all, I would like to thank all of my professors and teachers who have given me invaluable knowledge I have today. Especially, I would like to express my gratitude to my supervisor, Professor Adnan Aziz, who has offered guidance on the iTrak project, and to my reader Thangavel Subbu. I would also like to thank the friends I traveled with, who gave me the inspiration for iTrak. Most importantly, I want to thank my parents and my sister for all the hard work and sacrifices they have made to unconditionally support me with my studies.

ClosetStylist: An Android app to manage closet and programmatically pickout the outfit

Anh Luong, M.S.E.

The University of Texas at Austin, 2013

Supervisor: Adnan Aziz

iTrak is a combined mobile and web application that takes advantage of the GPS to allow travelers to share their experience while travelling. The application gathers GPS data and broadcasts it via a web interface or social networks such as Facebook to update user’s status during a trip. iTrak is also equipped with other features such as writing notes or recording video journals to offer a rich experience and provide an interactive diary, along with a real-time tracking ability, for travelers.

## 

## Chapter 1 Introduction

### Motivation

Travelers, especially backpackers, often stay quiet among the users of mobile Internet services and social networks. Instead, teenagers or college students are often mentioned as users of these services. However, different researches and surveys have proved that travelers are among the early users of mobile Internet services. For example, surveys conducted by Tjøstheim, Lous, Nordlund, and Fuglerud between 2000 and 2003 showed that travelers with e-commerce experience would be among the early adopters of mobile Internet services [TLNF2003]. The 2002 survey showed that among the travelers who participated in their surveys, 100% (458) of them were Internet users and 73% (334) of them had e-commerce experience, compared to 58% (3094) and 29% (1544) respectively in a national survey with random participants in 2001. These results showed that travelers are above the national average of Internet users and made them potentially early adopters of mobile Internet services.

Further studies conducted by Jeff Axup and Stephen Viller indicated that in the backpacker culture, there was a high demand for mobile devices and services to help travelers share their experience or socialize and connect with each other [AV]. With the recent blooming of the mobile technology and social networks, it is feasible to fulfill such demand. Nowadays, a user can use a single smart phone to do different things while travelling, such as note taking or updating their trip status on social networks. However, the user likely has to use separate apps for different tasks. As more applications and social networks are being created, it would be tiring to maintain all the apps and update status in every network. It would be more convenient if there was a single application that allowed travelers to achieve all those tasks.

### Vision

iTrak addresses the above problem by providing a centralized service where travelers can use a single mobile app to record their exciting travelling moments and stay connected with people across all networks. The target audiences of iTrak are backpackers, or travelers who possess the backpacker culture. The application itself will focus on the core functionality of tracking and creating travel notes, while at the same time providing a seamless portal for users to connect with people in their existing social networks. It will integrate with the GPS to provide live status update and trip tracking, which a user can share with friends and family. iTrak will also focus on providing functionalities that are often needed by travelers such as taking photos or recording voice and video journals.

iTrak will consist of a mobile app, a web app and a cloud service. The mobile app is mainly for obtaining GPS data, creating notes and saving them to an online repository. The web app will act as a personal blogging page that presents the travel experience collected from the mobile app. This interface will provide a user with a private blogging space to share travel experience, while still staying connected with friends and family in the user’s social networks. To aid the interaction between the mobile app and web app, a collection of cloud services will be developed to handle data transferring, GPS data processing, and media managing.

### Scope

This report will focus on three main aspects: the design of iTrak, the iTrak prototype, and the testing methodologies.

First, the report will visit the user interface and architecture designs of iTrak. It will go through the technologies used in iTrak to acquire the GPS data, how to store the data and how the data will be used for the live tracking feature. The report will also explain how iTrak will be integrated with existing social networks.

Then the report will discuss the iTrak prototype, which has been developed in parallel with the writing of this report. While the user interface and architecture designs cover a larger and more complete vision of iTrak, the prototype focuses on demonstrating only core functionalities of iTrak, including acquiring and using GPS data, how the data can associate with travel diaries and how the Facebook social network will take part in the application as a whole.

Along with the iTrak architecture and the prototype, this report will visit several testing methodologies and metrics that can be used to ensure the data integrity and the performance of the application.

### Report organization

In the next section, I discuss provide the user stories, user interface, and mockups to present the functionalities and the workflow of the ClosetStylist. Section 3 describes the technology stack applied to this app and the architecture of this app. Section 4 provide the result, testing methodologies used for quality control of the app. Section 5 concludes the paper, provides related and future work.

## Chapter 2 User Interface Design

### 2.1 Overview

In this section, we describe several user stories to demonstrate what ClosetStylist app can be used for and some high level use cases that serve as user guide of the app. These are just some examples and does not dictate or limit users on how to user the app.

A usage flow will also be presented by a set of mockups to visualize the user interface.

### 2.2 User Stories

The below stories highlights some aspects that ClosetStylist has achieved:

* Users can easily go through and look at every item in their own clothing inventory.
* Users can find out how many dirty items are there to schedule to do laundry.
* Users can go through their worn history and look for what apparels they wore on any date in the past.
* Users can choose any outfits that the app has recommended based on the weather at the current location.

### 2.3 Use Cases

The use cases presented in this report followed the Agile modeling approach [Amb] to effectively model the high level structure and the workflow of the application. These use cases focus on describing how social networks should interact with iTrak activities. Each use case consists of an activity diagram, which is based on the Grid Flow Chart model developed by Les Matthies [Bry], and a textual description that describes the use case steps in more details. This approach worked well with the iTrak prototype because it allowed the general model of the application to be rapidly constructed and visually presented, while at the same time offered enough information about the main activities and features that would need to be developed.

In all of the below cases, there are always two actors – a user and the ClosetStylist Android app.

#### 2.3.1 Register



Figure 2.1: Register diagram

The precondition: ClosetStylist has already been installed on the device under test.

The purpose: show how the user can register an account to use the app the very first time and how to populate all the required fields.

The steps: the app has its own simple authenticating method to validate, independent of any social networks, so that the users can still use the app if they choose to not use any social network feature. When this is the very first time the app is launched, users click on the “Don’t have account – register here” and fill in the required fields, one of which is the postal code. The interesting point about this field is this is the default location that the app will use if it fails to find the current location later. If users don’t know their current zipcode, they can click on “Get Location” and the app will find the zip code based on the current location. Once the users have filled in all the required fields, they can click on the “Register” button to log in to the app. The user’s information is also persisted to the app database so that user can login the next time.

#### 2.3.2 Login



Figure 2.2: Login diagram

The precondition: Users have already registered.

The purpose: Users will need to enter username and password to login after the their registration

The steps: launch the app and enter their credentials. The app will navigate to the main screen where users can find information about the current location, date, weather, and they can to proceed to any of the four main pages: Outfit of the day, My Closet, My Laundry Bag, My Outfit History.

#### 2.3.3 Add new item



Figure 2.3: Add new item diagram

The precondition: users are logged in.

The purpose: Users need to populate their closets with their clothes so that the app can programmatically pick out the outfit for any day.

The steps: From the main screen, users click on “My Closet”. In the bottom of the My Closet page, there is “ADD ITEM” button. After filling in the required fields, users can take picture of the clothes by clicking on the camera icon. Once saved, users can crop the newly taken picture to get rid of the unnecessary parts. Users can choose either Reset all the fields or Save it. Clicking on “Save” button will save the detail of this item while clicking on “Reset” button will reset all the fields to the default values.

#### 2.3.4 View or edit an item



Figure 2.4: View or edit item diagram

The precondition: the item has been added to the closet.

The purpose: View the detail of an item and update the information if needed.

The steps: From the main screen, users click on “My Closet”. The wardrobe is categorized as “Outer”, “Top”, and “Bottom”. Users choose the tab that the wardrobe belongs to and click on the items they want to see. They can choose to change any of the fields, and can even mark an item is dirty to be sent to laundry.

#### 2.3.5 Pick an outfit



Figure 2.5: Pick an outfit diagram

The precondition: “My Closet” has been populated with some items in both Top and Bottom.

The purpose: The app programmatically suggests a list of outfits that best fit the user based on the weather and the occasion.

The steps: From the main screen, users click on “Outfit of the Day”. The app will display a list of suggested outfits based on the current weather and the Occasion set to “Casual”. There are 5 options for Occasion – “Formal”, “Semi\_Formal”, “Casual”, “Day\_Out”, “Night\_Out” and users can choose the appropriate Occasion. There are arrows to switch to another Top or Bottom. There are double-arrows to let the user traverse through the list of suggested outfits. Once the users decide to choose a particular outfit, they can click on the “WEAR” button at the bottom and they will be navigated to the “My Outfit History” page.

#### 2.3.6 View outfit history



Figure 2.6: View outfit history diagram

The precondition: users have already chosen to wear some outfits.

The purpose: Display the outfits that users have already worn on any particular day.

The steps: From the main screen, users click on “My Outfit History”. The page displays the outfits that users have worn on a particular day, starting with today. If users have worn several outfits on the same day, all of them will be listed in chronological order, starting with the one worn earliest in the day. User can click on any of them and they will be navigated to the “Outfit Preview” to see a more detail picture of the outfit.

#### 2.3.7 Laundry bag



Figure 2.7: Laundry bag diagram

The precondition: Users have already chosen to wear some outfits.

The purpose: Display the dirty items so that the users can schedule to wash them.

The steps: From the main screen, users click on “My Laundry Bag”. The page will display all the dirty items as a list. Users can click on any of them to view more detail.

### 2.4 Mockups

A set of mockups was designed to provide a vision of how iTrak would look like on the mobile interface and web interface. All mockups were created using the Balsamiq [Bal] software, which offers a powerful and user friendly interface to conveniently create mockups. A storyboard of these mockups were also put together to describe the workflow of the iTrak application. This storyboard is presented in Figure 2.5. More details of each mockup in the storyboard are available in the following sections.

#### 2.4.1 Mobile App Mockups

|  |  |
| --- | --- |
| C:\Users\code_warrior\Desktop\mockup.png |  |

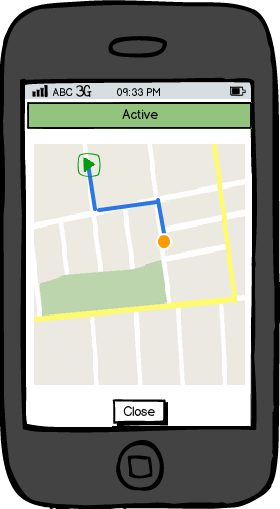
Figure 2.6: User login and registration mockups.

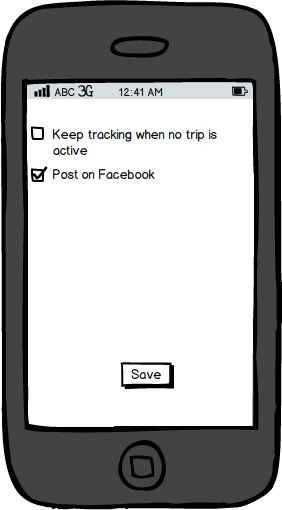
In Figure 2.6, two mockups for the user login and registration pages are presented. The Login page will be displayed if the mobile app does not find a login session saved previously when user first launches iTrak. Otherwise, iTrak will display the main page as seen in Figure 2.7.

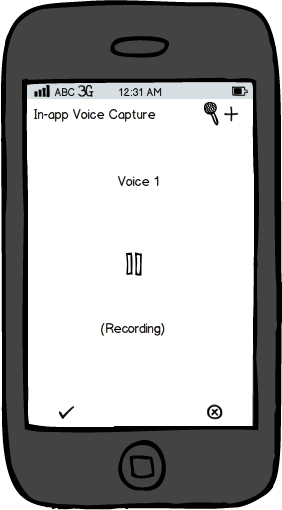
In Figure 2.7, Mockup 2.7a shows the main page of iTrak mobile app. The status bar at the top of page shows whether a trip is currently active or not. The main page also contains icons for the main functionalities which are described with more details in Table 2.1.

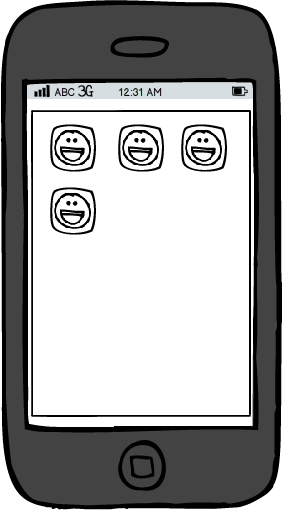
Mockup 2.7b shows the live map view when user presses the Live map icon from the main page. This live view page allows user to review the itinerary made within the day. When there is not an active trip, the live view will simply show the last location recorded by iTrak.

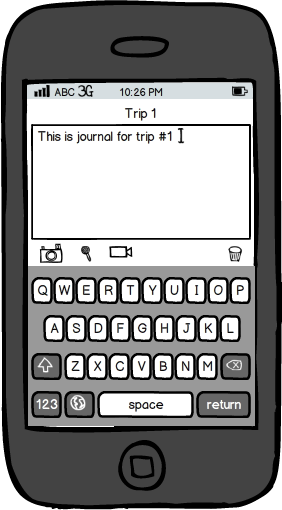


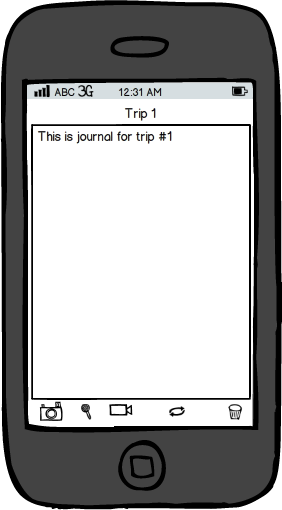


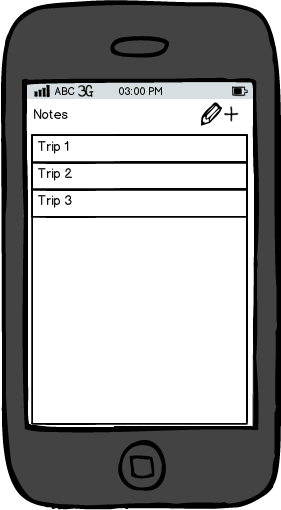


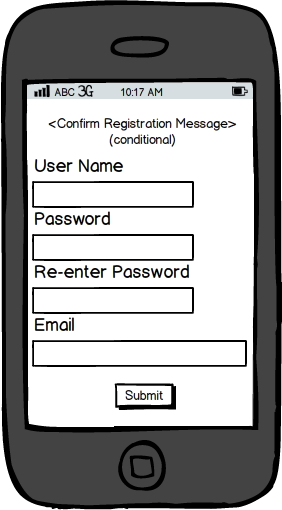


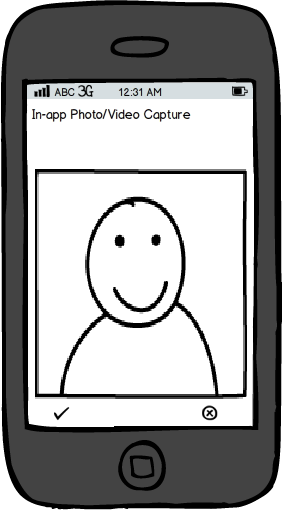


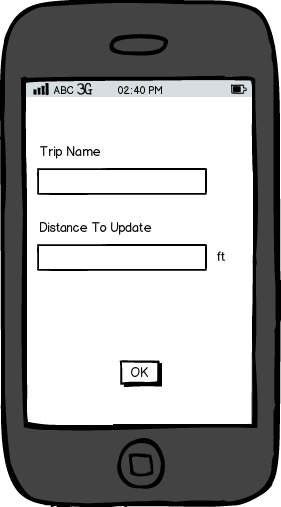


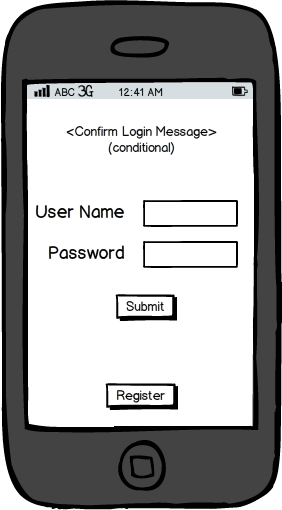












1a

1b.1

2b.3b

2a.1

2a.2

2a.3a

1b.2

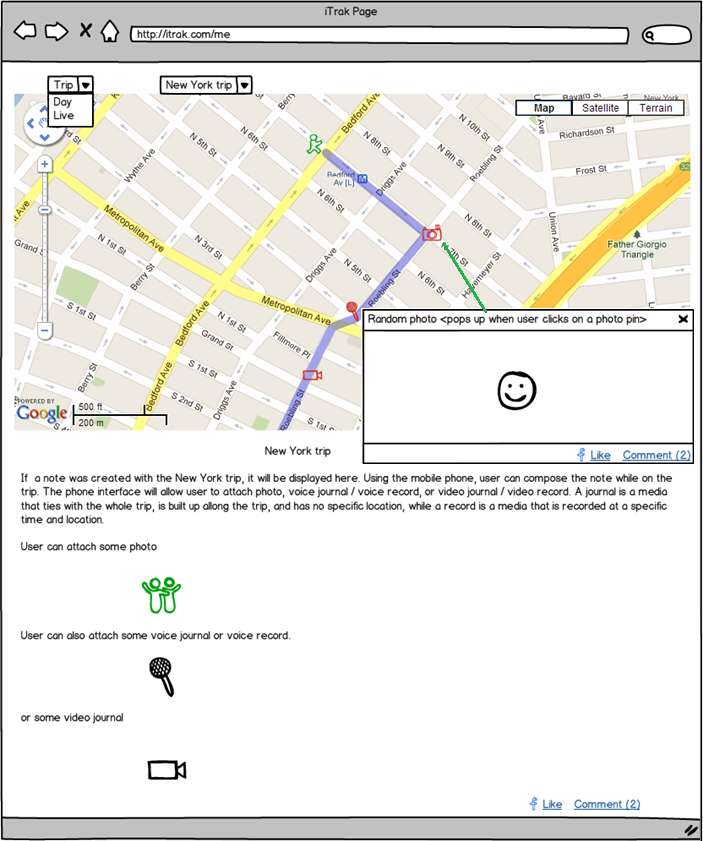
2b

2c

2d

2f





Notify

Notify

2e

Figure 2.5: Mockups storyboard.

|  |  |
| --- | --- |
| Mockup 2.7a | Mockup 2.7b |

Figure 2.7: Main page and live map view mockups.

|  |  |
| --- | --- |
| New trip | Takes to the configuration page for creating a new trip. See Mockup 2.8a in Figure 2.8. |
| Notes | Takes to the notes manager page for creating new notes or editing existing notes. See Mockup 2.8b in Figure 2.8. |
| Voice journal | Takes to the voice journals manager page. See Mockup 2.10a in Figure 2.10. |
| Video journal | Takes to the video journals manager page. See Mockup 2.10b in Figure 2.10. |
| Photo capture | In-app functionality to take picture. See Mockup 2.10b in Figure 2.10. |
| Live map | Takes to the live map view page. See Mockup 2.7b in Figure 2.7. |
| Settings | Takes to the settings page. See Mockup 2.10c in Figure 2.10. |
| Help | Takes to an external web site that has detail information about iTrak. |

Table 2.1: Functionality descriptions of mobile app icons on the main page.

|  |  |
| --- | --- |
| Note entries  Minimum distance to notify new GPS location  Name of the new trip  Mockup 2.8a | Mockup 2.8b |

Figure 2.8: Create trip page and note manager page mockups.

In Figure 2.8, Mockup 2.8a shows how user will create a new trip. User needs to enter the trip name and the minimum distance that triggers a new update for current location. The Distance to Update value helps optimize the performance of the app and the battery life of the smart phone. For example, if the user is travelling on feet, the travel area may be narrow and the distance to update would need to be small to make the map more accurate. If the user is travelling on a car or a train, it can be a long journey that covers a large area and the distance to update can be relaxed to a larger amount to avoid frequent location updates and to save battery life, while still maintaining a reasonably accurate map. More understanding of how the distance to update can impact the performance of a mobile app involving GPS location tracking can be read at [Wae].

Mockup 2.8b shows how the notes manager will be implemented. It consists of a list of created notes and a button to allow creating new notes. User can edit an existing note or start a new one. If a trip is currently active, a newly created note will automatically associate with the trip. If an existing note already associates with the currently active trip, the system will redirect the user to this note when the user attempts to create a new note.

|  |  |  |
| --- | --- | --- |
| Mockup 2.9a | C:\Users\code_warrior\Desktop\mockup.pngMockup 2.9b | Mockup 2.9c |

Sync button to update note

Buttons to attach media to note

Figure 2.9: Note edit mockups.

Figure 2.9 shows the mockups of the note edit page. Mockup 2.9a shows the note edit page in review mode where the content of the note stays in the center and takes most of the space of the page. At the bottom left corner of the page, there are three buttons that allow user to attach a media such as a photo, a voice journal, or a video journal to the note. The next button allows user to synchronize the content of the note to the server. The bottom right corner of the page has the trash bin button that allows the user to delete the note. Mockup 2.9b simply shows the note edit page in edit mode with the QWERTY keyboard.

In Mockup 2.9c, when the user selects one of the media buttons, a list of existing items of the selected media type will pop up and allow the user to attach an item to the note. The added media items will be uploaded to the cloud server if they are not yet available on the server by the time the mobile app updates the note content to the iTrak server (see Chapter 3).

|  |  |  |
| --- | --- | --- |
| Mockup 2.10a | Mockup 2.10b | C:\Users\code_warrior\Desktop\mockup.png    Mockup 2.10c |

Cancel

Accept

Figure 2.10: In-app media utilities and app settings mockups.

Mockups 2.10a and 2.10b show the in-app utilities for recording sound and taking photos or videos. At the bottom of each mockup, there are two buttons that allow user to either accept the captured media or cancel it.

Mockup 2.10c shows the settings page of the iTrak mobile app, where user can configure the app to allow continuous tracking even when no trip is active. User can also select which social network to post activity notifications on when a journal is created or when a trip has started.

As mentioned in the scope of this report, the prototype of iTrak will support only Facebook as the proof of concept. Figure 2.11 shows how the Facebook page of the iTrak user may look like when a new trip is created. In this screenshot, my Facebook test page received a notification from the iTrak prototype with the description “new trip created ok” and the logo image of iTrak. Clicking on the logo will take visitors of the Facebook page to the iTrak website as showed in Figure 2.14, where visitor can further monitor the trip via the map view.

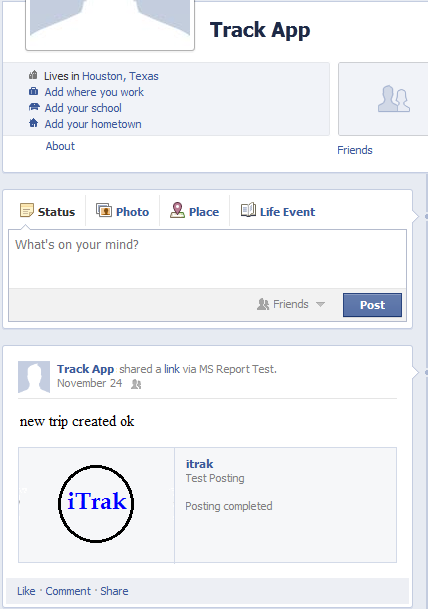


Figure 2.11: Facebook page when received status update from iTrak.

#### 2.4.2 Web App Mockups

The web app consists of a single web page that allows user to share his/her journeys with friends and family. Through this website, people can keep track of the status of the user’s current trip or review the past trips. Figure 2.12 shows the layout of the web app. The top of the website is the control bar which has a collection of the dropdown controls. The dropdown on the left allows a web browser to select the view mode of the page. There are three different view modes:

* Live mode: allows viewers to see status of the user’s current journey in real time (Figure 2.13).
* Trip mode: allows viewers to review the past trips of the user by trip names (Figure 2.12).
* Day mode: allows viewers to review the past trips of the user by date (Figure 2.14).

Below the control bar is the map view, which highlights the itinerary of a trip based on the GPS data collected. It also pins the media taken during the trip. Clicking on a pin will pop up a review window for the associated media. With this map view interface, audiences of the website can conveniently and interactively review the user’s trips and journals.

Below the map view is the note associates with the selected trip. The media that has been added to the note using the mobile app is also shown on the website. For a standalone note, which does not associate with a trip, the control bar and the map view will be hidden. Friends and family of the user access standalone notes via the direct links populated to the social network status updates. At the end of the page is the social network bar that allows viewers to further share a note or a trip of the user on their networks. In Figure 2.12, the Facebook Like and Comment buttons are used as examples for interactions with social network.

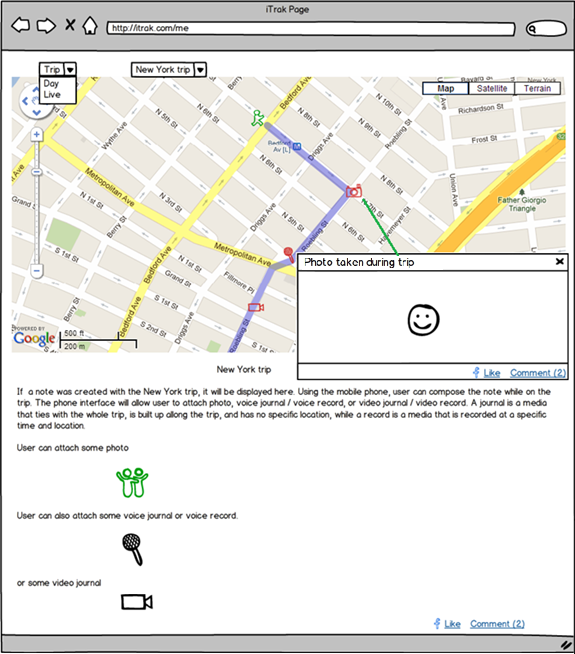


Figure 2.12: Web app in Trip mode.

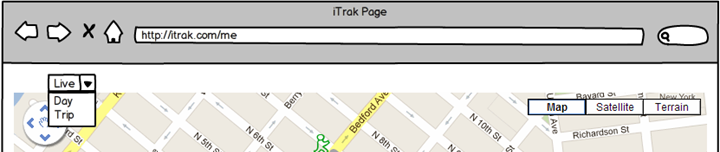


Figure 2.13: Control bar of the web app in Live mode.

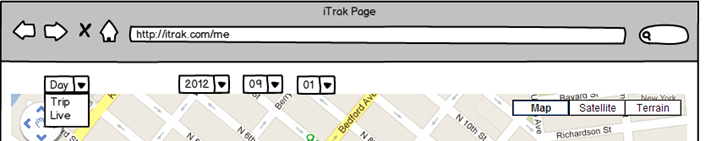


Figure 2.14: Control bar of the web app in Day mode.

## Chapter 3 Implementation

### 3.1 Technology stack

ClosetStylist is an Android app and Java is used as the main programming language. The table below shows a summary of my development environment.

|  |  |
| --- | --- |
| Language | Java, JavaEE, Javascript, KML 2.2 |
| IDE | Eclipse Juno 4.2, Kepler 4.3 |
| Additional code editor | GVIM, Chrome |
| SDKs | JDK 1.6, Android 2.3.3, Facebook 3.0, JSON, Gson 2.0 |
| Test equipment | Samsung S3 |
| System | Ubuntu 12.04 and Ubuntu 12.10. |

Table 3.1: Development environment.

The app was first development on a laptop with Intel i7-3720 2.6GHz, 16GB RAM, Windows 64-bit to run on the Android simulator. Since I tried to run the app on Samsung S3, the development was moved to a laptop with Intel Dual-Core, 4GB RAM, Ubuntu 12.10 32-bit, and a desktop with AMD Quad-Core, 16GB RAM, Ubuntu 12.04 64-bit. Initially, the development was carried on the main system of my laptop which has an i7-26020M 2.7GHz processor, Windows 7 64bit, and 8GB of RAM.

Several technologies have been applied to make this app work. Below are the main ones.

#### 3.1.1 Location Service

In this app, GeoNames database is used as a service to convert between postal code and the geographic coordinates (the longitude and latitude) of a location. Geonames has been chosen over Yahoo service (Yahoo BOSS PlaceFinder) because it is free and Yahoo service uses proprietary WOEID (Where On Earth Identification number). The later is subject to change from Yahoo and will cause problem if we want to switch to another service later.

In addition, Geonames provide many services in many formats such as XML, JSON, etc. In this app, the service converting from geographic coordinates to postal code (and then city and country) and vice versa is used the most. The URL that the app will get the JSON information about this information is listed below: <http://api.geonames.org/postalCodeSearchJSON?postalcode=78758&maxRows=10&username=demo> , <http://api.geonames.org/findNearbyPostalCodes?lat=30.4883997&lng=-97.7175117&username=demo> .

#### 3.1.2 Weather Service

We have looked into several weather services including Yahoo Weather, World Weather Online, Open Weather Map. [SWA] provides the sample code to retrieve weather information from Open Weather Map and it really helped us to decide to use Open Weather Map is the choice of the weather service.

The weather information consists of geographic coordinates, temperatures, humidity, pressure, wind and much more information.

Figure 3.1: Interactions with Facebook

#### 3.1.3 Clothes Matching Service

This smart service provides suggestions on which wardrobe users should put on based on the available items, the current weather information, and the occasion of the event, and user’s gender.

#### 3.1.3.1 High-level design

There are five steps to create the list of suggested outfits. Each outfit consists of a top, bottom, and an optional outer if the temperature is low. We grade each outfit based on the criteria described above to create a list of outfit descending in score.



Laundry Filter: eliminate dirty items out of the process.

Temperature Filter: eliminate items that do not cover the range of today’s maximum temperature and minimum temperature of the current location.

Occasion Matching: each item is given a score for the chosen occasion. For example, a short will be graded low for the “Formal” occasion but it will get a high point in the “Day\_Out” occasion.

Pair Matching: each combination of a top item with a bottom item (and optionally an outer) will be scored based on its category. This point is added together with the point of each item in the previous step to the outfit.

Color Matching: the color combination of the top, the bottom, and the outer of the outfit will be given a score. This is added to the points from the previous steps to produce the final score. The result is a list of outfits in the order of descending points.

#### 3.1.3.2 Low-level design: need more detail on the different lists created in each stage

##### 3.1.3.2.1 Laundry Filter

This step is pretty simple, given that each item has a “dirty” attribute to specify an item is clean or dirty. The implementation can simply query the database of clothes in the closet and return a list of clean items.

##### 3.1.3.2.2 Temperature Filter

Each item is assigned a range of temperature in which it can be worn. Among different attributes of an item, material and style are the two that we think the temperature can be based on. After consideration, style was chosen over material because it is more related to temperature. The rationale of this choice is that if an item of a specific material does not keep the keep warm enough, then another item of the same material can still be worn outside. Regarding style, once a particular style is chosen, it is more difficult to match with another style to keep warm.

Below is the look-up table we use to define the range of each style per gender.

|  |  |  |
| --- | --- | --- |
| **Style - Men** | Temp Min | Temp Max |
| Pants | -999 | 999 |
| Jeans | -999 | 999 |
| Shorts | -999 | 999 |
| Dress\_Shirt | -999 | 999 |
| Casual\_Button\_Down\_Shirt | -999 | 999 |
| Polo | -999 | 999 |
| T\_Shirt\_Short\_Sleeve | -999 | 999 |
| T\_Shirt\_Long\_Sleeve | -999 | 999 |
| Sweater\_And\_Sweatshirt | -999 | 999 |
| Coat\_And\_Jacket\_Light | -999 | 75 |
| Coat\_And\_Jacket\_Heavy | -999 | 40 |
| **Style - Women** | Temp Min | Temp Max |
| Pants | -999 | 999 |
| Jeans | -999 | 999 |
| Legging\_Skinny | -999 | 999 |
| Shorts | -999 | 999 |
| Skirts | -999 | 999 |
| Collared\_And\_Button\_Down | -999 | 999 |
| Blouse\_Short\_Sleeve | -999 | 999 |
| Blouse\_Long\_Sleeve | -999 | 999 |
| Blouse\_Sleeveless | -999 | 999 |
| T\_Shirt\_Long\_Sleeve | -999 | 999 |
| T\_Shirt\_Short\_Sleeve | -999 | 999 |
| Tank\_Camisoles | -999 | 999 |
| Party\_Top | -999 | 999 |
| Tunic | -999 | 999 |
| Pull\_Over | -999 | 999 |
| Cardigan | -999 | 999 |
| Sweater\_And\_Sweatshirt | -999 | 999 |
| Vest | -999 | 999 |
| Coat\_And\_Jacket\_Light | -999 | 75 |
| Coat\_And\_Jacket\_Heavy | -999 | 40 |

##### 3.1.3.2.3 Occasion Matching

Each item is given different point based on gender, category, style, and occasion. In this step, the score has higher weight than pair and color matching steps because occasion matching more important to the final outfit in our opinions.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Point | | | | |
| **Category** | **Style** | **Formal** | **Semi\_Formal** | **Casual** | **Day\_Out** | **Night\_Out** |
| Bottom | Pants | 40 | 40 | 5 | 0 | 20 |
| Bottom | Jeans | 5 | 20 | 40 | 20 | 40 |
| Bottom | Shorts | 0 | 0 | 20 | 40 | 0 |
| Top | Dress\_Shirt | 40 | 40 | 5 | 0 | 5 |
| Top | Casual\_Button\_Down\_Shirt | 5 | 20 | 20 | 20 | 40 |
| Top | Polo | 5 | 20 | 40 | 5 | 20 |
| Top | T\_Shirt\_Short\_Sleeve | 0 | 0 | 20 | 40 | 5 |
| Top | T\_Shirt\_Long\_Sleeve | 0 | 5 | 20 | 5 | 20 |
| Top | Sweater\_And\_Sweatshirt | 0 | 5 | 20 | 20 | 5 |
| Top | Coat\_And\_Jacket\_Light | 40 | 40 | 5 | 0 | 20 |
| Top | Coat\_And\_Jacket\_Heavy | 5 | 5 | 5 | 5 | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Point | | | | |
| **Category** | **Style** | **Formal** | **Semi\_Formal** | **Casual** | **Day\_Out** | **Night\_Out** |
| Bottom | Pants | 40 | 40 | 5 | 0 | 20 |
| Bottom | Jeans | 5 | 20 | 40 | 20 | 20 |
| Bottom | Legging\_Skinny | 0 | 5 | 20 | 20 | 40 |
| Bottom | Shorts | 0 | 0 | 5 | 40 | 5 |
| Bottom | Skirts | 5 | 20 | 20 | 20 | 20 |
| Top | Collared\_And\_Button\_Down | 40 | 20 | 0 | 0 | 5 |
| Top | Blouse\_Short\_Sleeve | 20 | 40 | 20 | 20 | 20 |
| Top | Blouse\_Long\_Sleeve | 40 | 40 | 5 | 5 | 20 |
| Top | Blouse\_Sleeveless | 5 | 20 | 20 | 40 | 20 |
| Top | T\_Shirt\_Long\_Sleeve | 5 | 20 | 40 | 5 | 5 |
| Top | T\_Shirt\_Short\_Sleeve | 0 | 5 | 40 | 40 | 5 |
| Top | Tank\_Camisoles | 0 | 0 | 5 | 40 | 20 |
| Top | Party\_Top | 0 | 0 | 5 | 20 | 40 |
| Top | Tunic | 0 | 5 | 20 | 20 | 20 |
| Top | Pull\_Over | 0 | 0 | 20 | 20 | 0 |
| Top | Sweater\_And\_Sweatshirt | 20 | 20 | 5 | 5 | 20 |
| Top | Coat\_And\_Jacket\_Light | 40 | 20 | 0 | 0 | 20 |
| Top | Cardigan | 40 | 40 | 20 | 5 | 20 |
| Top | Vest | 40 | 20 | 5 | 0 | 20 |
| Top | Coat\_And\_Jacket\_Heavy | 5 | 5 | 5 | 5 | 20 |

##### 3.1.3.2.4 Pair matching

Similar to previous steps, we create a big look-up table for each combination of a top item and a bottom item, and optionally an outer based on the weather, gender. One thing to notice here is the point is smaller than the occasion matching step because this step is not as important as that one.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bottom** | **Top** | **Mild Weather Outer** | **Cold Weather Outer** |  | **Bottom** | **Top** | **Point** | **Outer** |
| Pants | Dress\_Shirt | Sweater\_And\_Sweatshirt | Coat\_And\_Jacket\_Heavy |  | Pants | Dress\_Shirt | 20 | No |
| Jeans | Casual\_Button\_Down\_Shirt | Coat\_And\_Jacket\_Light | (temp range <40) |  | Pants | Casual\_Button\_Down\_Shirt | 5 | No |
| Shorts | Polo | (temp range <70) |  |  | Pants | Polo | 20 | No |
|  | T\_Shirt\_Short\_Sleeve |  |  |  | Jeans | Casual\_Button\_Down\_Shirt | 5 | No |
|  | T\_Shirt\_Long\_Sleeve |  |  |  | Jeans | Polo | 16 | No |
|  |  |  |  |  | Jeans | T\_Shirt\_Short\_Sleeve | 20 | No |
|  |  |  |  |  | Jeans | T\_Shirt\_Long\_Sleeve | 20 | No |
|  |  |  |  |  | Shorts | Casual\_Button\_Down\_Shirt | 6 | No |
|  |  |  |  |  | Shorts | Polo | 6 | No |
|  |  |  |  |  | Shorts | T\_Shirt\_Short\_Sleeve | 20 | No |
|  |  |  |  |  | Shorts | T\_Shirt\_Long\_Sleeve | 5 | No |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bottom** | **Top** | **Outer** | **Cold Weather Outer** |  | **Bottom** | **Top** | **Point** | **Outer** |
| Pants | Collared\_And\_Button\_Down | Cardigan | Coat\_And\_Jacket\_Heavy |  | Pants | Collared\_And\_Button\_Down | 20 | No |
| Jeans | Blouse\_Short\_Sleeve | Sweater\_And\_Sweatshirt | (temp range <40) |  | Pants | Blouse\_Short\_Sleeve | 16 | No |
| Legging\_Skinny | Blouse\_Long\_Sleeve | Vest |  |  | Pants | Blouse\_Long\_Sleeve | 16 | No |
| Shorts | Blouse\_Sleeveless | Coat\_And\_Jacket\_Light |  |  | Pants | Blouse\_Sleeveless | 16 | Yes |
| Skirts | T\_Shirt\_Long\_Sleeve | (temp range <70) |  |  | Pants | Tank\_Camisoles | 10 | Yes |
|  | T\_Shirt\_Short\_Sleeve |  |  |  | Pants | Party\_Top | 10 | No |
|  | Tank\_Camisoles |  |  |  | Jeans | Blouse\_Short\_Sleeve | 10 | Yes |
|  | Party\_Top |  |  |  | Jeans | Blouse\_Long\_Sleeve | 10 | No |
|  | Tunic |  |  |  | Jeans | Blouse\_Sleeveless | 10 | Yes |
|  | Pull\_Over |  |  |  | Jeans | T\_Shirt\_Long\_Sleeve | 20 | No |
|  |  |  |  |  | Jeans | T\_Shirt\_Short\_Sleeve | 20 | No |
|  |  |  |  |  | Jeans | Tank\_Camisoles | 16 | Yes |
|  |  |  |  |  | Jeans | Party\_Top | 16 | No |
|  |  |  |  |  | Jeans | Pull\_Over | 16 | No |
|  |  |  |  |  | Legging\_Skinny | Blouse\_Short\_Sleeve | 10 | No |
|  |  |  |  |  | Legging\_Skinny | Blouse\_Long\_Sleeve | 10 | No |
|  |  |  |  |  | Legging\_Skinny | Blouse\_Sleeveless | 10 | Yes |
|  |  |  |  |  | Legging\_Skinny | T\_Shirt\_Long\_Sleeve | 6 | No |
|  |  |  |  |  | Legging\_Skinny | T\_Shirt\_Short\_Sleeve | 6 | Yes |
|  |  |  |  |  | Legging\_Skinny | Tank\_Camisoles | 6 | Yes |
|  |  |  |  |  | Legging\_Skinny | Party\_Top | 20 | No |
|  |  |  |  |  | Legging\_Skinny | Tunic | 20 | No |
|  |  |  |  |  | Legging\_Skinny | Pull\_Over | 6 | No |
|  |  |  |  |  | Legging\_Skinny | Collared\_And\_Button\_Down | 6 | No |
|  |  |  |  |  | Shorts | Blouse\_Short\_Sleeve | 10 | No |
|  |  |  |  |  | Shorts | Blouse\_Long\_Sleeve | 10 | No |
|  |  |  |  |  | Shorts | Blouse\_Sleeveless | 10 | Yes |
|  |  |  |  |  | Shorts | T\_Shirt\_Long\_Sleeve | 16 | No |
|  |  |  |  |  | Shorts | T\_Shirt\_Short\_Sleeve | 20 | Yes |
|  |  |  |  |  | Shorts | Tank\_Camisoles | 20 | Yes |
|  |  |  |  |  | Shorts | Party\_Top | 6 | No |
|  |  |  |  |  | Shorts | Pull\_Over | 10 | No |
|  |  |  |  |  | Skirts | Collared\_And\_Button\_Down | 10 | No |
|  |  |  |  |  | Skirts | Blouse\_Short\_Sleeve | 20 | No |
|  |  |  |  |  | Skirts | Blouse\_Long\_Sleeve | 16 | No |
|  |  |  |  |  | Skirts | Blouse\_Sleeveless | 20 | Yes |
|  |  |  |  |  | Skirts | T\_Shirt\_Long\_Sleeve | 16 | No |
|  |  |  |  |  | Skirts | T\_Shirt\_Short\_Sleeve | 16 | Yes |
|  |  |  |  |  | Skirts | Tank\_Camisoles | 10 | Yes |
|  |  |  |  |  | Skirts | Party\_Top | 20 | No |
|  |  |  |  |  | Skirts | Pull\_Over | 6 | No |

##### 3.1.3.2.5 Color matching

There are too many colors and it is impossible to list every single color existing. We decide to divide color into twelve basic colors: beige, black, blue, brown, gray, green, orange, pink, red, violet, white, and yellow. Another one is also added for the multicolor or pattern item. Hence, we have a total of 13 colors to deal with. These 13 colors can further be divided into two groups – “Color” and “Neutral”. Some colors can be matched together, while others cannot. Also, “Neutral” colors can be easily matched with other while the “Color” colors are more restricted. These relations are expressed through the points given to each combination of these colors as shown below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Color** | **Group** |  | **Bottom** | **Top** | Point | **Top-Bottom** | **Outer** |
| Blue | Color |  | Beige | Black | 2 | Beige-Black | Color |
| Green | Color |  | Beige | Blue | 4 | Beige-Blue | Any |
| Violet | Color |  | Beige | Brown | 22 | Beige-Brown | Color |
| Red | Color |  | Beige | Gray | 24 | Beige-Gray | Color |
| Yellow | Color |  | Beige | Green | 20 | Beige-Green | Any |
| Orange | Color |  | Beige | Multicolor\_Pattern | 6 | Beige-Multicolor\_Pattern | Neutral |
| Pink | Color |  | Beige | Orange | 10 | Beige-Orange | Any |
| Multicolor\_Pattern | Color |  | Beige | Pink | 8 | Beige-Pink | Any |
| Gray | Neutral |  | Beige | Red | 16 | Beige-Red | Any |
| White | Neutral |  | Beige | Violet | 18 | Beige-Violet | Any |
| Black | Neutral |  | Beige | White | 12 | Beige-White | Color |
| Brown | Neutral |  | Beige | Yellow | 14 | Beige-Yellow | Any |
| Beige | Neutral |  | Black | Beige | 4 | Black-Beige | Color |
|  |  |  | Black | Blue | 6 | Black-Blue | Any |
|  |  |  | Black | Brown | 24 | Black-Brown | Color |
|  |  |  | Black | Gray | 20 | Black-Gray | Color |
|  |  |  | Black | Green | 18 | Black-Green | Any |
|  |  |  | Black | Multicolor\_Pattern | 8 | Black-Multicolor\_Pattern | Neutral |
|  |  |  | Black | Orange | 10 | Black-Orange | Any |
|  |  |  | Black | Pink | 12 | Black-Pink | Any |
|  |  |  | Black | Red | 22 | Black-Red | Any |
|  |  |  | Black | Violet | 14 | Black-Violet | Any |
|  |  |  | Black | White | 2 | Black-White | Color |
|  |  |  | Black | Yellow | 16 | Black-Yellow | Any |
|  |  |  | Blue | Beige | 6 | Blue-Beige | Any |
|  |  |  | Blue | Black | 8 | Blue-Black | Any |
|  |  |  | Blue | Brown | 10 | Blue-Brown | Any |
|  |  |  | Blue | Gray | 12 | Blue-Gray | Any |
|  |  |  | Blue | Orange | 2 | Blue-Orange | Neutral |
|  |  |  | Blue | White | 4 | Blue-White | Any |
|  |  |  | Brown | Beige | 2 | Brown-Beige | Color |
|  |  |  | Brown | Black | 16 | Brown-Black | Color |
|  |  |  | Brown | Blue | 10 | Brown-Blue | Any |
|  |  |  | Brown | Gray | 12 | Brown-Gray | Color |
|  |  |  | Brown | Green | 8 | Brown-Green | Any |
|  |  |  | Brown | Multicolor\_Pattern | 8 | Brown-Multicolor\_Pattern | Neutral |
|  |  |  | Brown | Orange | 10 | Brown-Orange | Any |
|  |  |  | Brown | Pink | 12 | Brown-Pink | Any |
|  |  |  | Brown | Red | 18 | Brown-Red | Any |
|  |  |  | Brown | Violet | 14 | Brown-Violet | Any |
|  |  |  | Brown | White | 4 | Brown-White | Color |
|  |  |  | Brown | Yellow | 6 | Brown-Yellow | Any |
|  |  |  | Gray | Beige | 24 | Gray-Beige | Color |
|  |  |  | Gray | Black | 20 | Gray-Black | Color |
|  |  |  | Gray | Blue | 10 | Gray-Blue | Any |
|  |  |  | Gray | Brown | 18 | Gray-Brown | Color |
|  |  |  | Gray | Green | 12 | Gray-Green | Any |
|  |  |  | Gray | Multicolor\_Pattern | 8 | Gray-Multicolor\_Pattern | Neutral |
|  |  |  | Gray | Orange | 2 | Gray-Orange | Any |
|  |  |  | Gray | Pink | 6 | Gray-Pink | Any |
|  |  |  | Gray | Red | 4 | Gray-Red | Any |
|  |  |  | Gray | Violet | 14 | Gray-Violet | Any |
|  |  |  | Gray | White | 22 | Gray-White | Color |
|  |  |  | Gray | Yellow | 16 | Gray-Yellow | Any |
|  |  |  | Green | Beige | 14 | Green-Beige | Any |
|  |  |  | Green | Black | 12 | Green-Black | Any |
|  |  |  | Green | Brown | 10 | Green-Brown | Any |
|  |  |  | Green | Gray | 8 | Green-Gray | Any |
|  |  |  | Green | White | 6 | Green-White | Any |
|  |  |  | Green | Yellow | 4 | Green-Yellow | Neutral |
|  |  |  | Green | Blue | 2 | Green-Blue | Neutral |
|  |  |  | Multicolor\_Pattern | Beige | 2 | Multicolor\_Pattern-Beige | Neutral |
|  |  |  | Multicolor\_Pattern | Black | 4 | Multicolor\_Pattern-Black | Neutral |
|  |  |  | Multicolor\_Pattern | Brown | 10 | Multicolor\_Pattern-Brown | Neutral |
|  |  |  | Multicolor\_Pattern | Gray | 8 | Multicolor\_Pattern-Gray | Neutral |
|  |  |  | Multicolor\_Pattern | White | 6 | Multicolor\_Pattern-White | Neutral |
|  |  |  | Orange | Beige | 10 | Orange-Beige | Any |
|  |  |  | Orange | Black | 4 | Orange-Black | Any |
|  |  |  | Orange | Blue | 2 | Orange-Blue | Neutral |
|  |  |  | Orange | Brown | 6 | Orange-Brown | Any |
|  |  |  | Orange | Gray | 12 | Orange-Gray | Any |
|  |  |  | Orange | White | 8 | Orange-White | Any |
|  |  |  | Pink | Beige | 6 | Pink-Beige | Any |
|  |  |  | Pink | Black | 2 | Pink-Black | Any |
|  |  |  | Pink | Brown | 4 | Pink-Brown | Any |
|  |  |  | Pink | Gray | 10 | Pink-Gray | Any |
|  |  |  | Pink | White | 8 | Pink-White | Any |
|  |  |  | Red | Beige | 6 | Red-Beige | Any |
|  |  |  | Red | Black | 10 | Red-Black | Any |
|  |  |  | Red | Brown | 8 | Red-Brown | Any |
|  |  |  | Red | Gray | 4 | Red-Gray | Any |
|  |  |  | Red | White | 2 | Red-White | Any |
|  |  |  | Violet | Beige | 8 | Violet-Beige | Any |
|  |  |  | Violet | Black | 10 | Violet-Black | Any |
|  |  |  | Violet | Brown | 12 | Violet-Brown | Any |
|  |  |  | Violet | Gray | 14 | Violet-Gray | Any |
|  |  |  | Violet | White | 4 | Violet-White | Any |
|  |  |  | Violet | Yellow | 2 | Violet-Yellow | Neutral |
|  |  |  | Violet | Pink | 6 | Violet-Pink | Neutral |
|  |  |  | White | Beige | 10 | White-Beige | Color |
|  |  |  | White | Black | 2 | White-Black | Color |
|  |  |  | White | Blue | 12 | White-Blue | Any |
|  |  |  | White | Brown | 8 | White-Brown | Color |
|  |  |  | White | Gray | 24 | White-Gray | Color |
|  |  |  | White | Green | 20 | White-Green | Any |
|  |  |  | White | Multicolor\_Pattern | 4 | White-Multicolor\_Pattern | Neutral |
|  |  |  | White | Orange | 18 | White-Orange | Any |
|  |  |  | White | Pink | 16 | White-Pink | Any |
|  |  |  | White | Red | 6 | White-Red | Any |
|  |  |  | White | Violet | 14 | White-Violet | Any |
|  |  |  | White | Yellow | 22 | White-Yellow | Any |
|  |  |  | Yellow | Beige | 8 | Yellow-Beige | Any |
|  |  |  | Yellow | Black | 4 | Yellow-Black | Any |
|  |  |  | Yellow | Brown | 6 | Yellow-Brown | Any |
|  |  |  | Yellow | Gray | 10 | Yellow-Gray | Any |
|  |  |  | Yellow | Green | 12 | Yellow-Green | Neutral |
|  |  |  | Yellow | Violet | 2 | Yellow-Violet | Neutral |
|  |  |  | Yellow | White | 14 | Yellow-White | Any |

##### 3.1.3.2.6 Pre-load to the app

All of these look-up tables were created in Excel. To preload these tables in the app, we first export these tables in CSV format and save them in res/raw folder of the project. Several methods were implemented to read these tables and create SQLite tables to be preloaded with the app at built time.

The goal of doing this is future change to the point for certain combinations will be integrated into our code without any code modifications. The steps include exporting CSV files, updating the files in the res/raw folder, rebuilding the project.

### 3.2 Architecture

ClosetStylist design is composed of three main layers: presentation layer, application layer, and data layer. The layer design is mainly for code reusability and portability. In addition, multiple design patterns have been applied to provide flexibility to switch between different services.

Figure 3.3 shows the top-level architectural design of ClosetStylist.

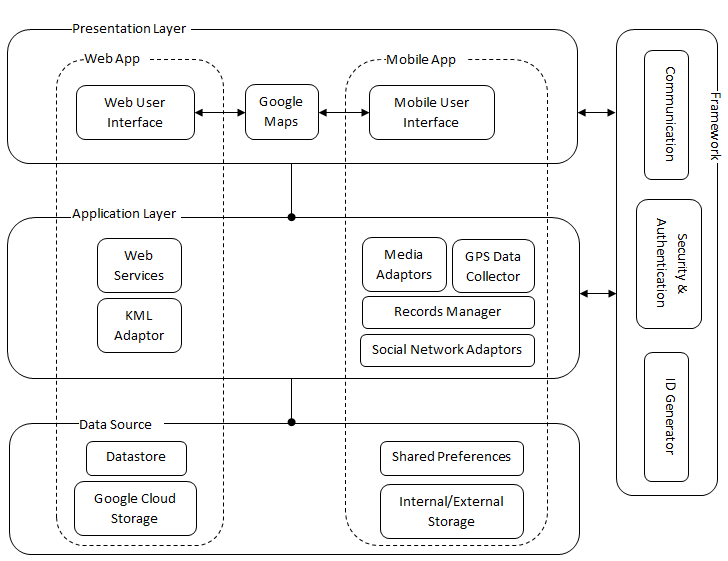


Figure 3.3: iTrak top-level architecture.

#### 3.2.1 Core

The core code includes common classes, common interfaces, helper classes, enum classes that are used to glue different layers together. The purpose of this layer is connecting different layers through common interfaces and features so that upper layers can work seamlessly with the information of the lower layers.

#### 3.2.2 Presentation layer

This layer contains UI and UX modules of the app. The purpose of this layer is to implement the behavior logic and provide the good user experience of the app flow as well as look and feel of the app. The design of this layer strictly follows that of Android design. We first draft our own design, and then we consulted with professionals UI/UX designer and UI developer.

#### 3.2.3 Application layer

The application layer consists of all the services employed in this app, including weather service, location service, and clothes matching service. The purpose of this layer is to implement application logic and provide all the functionalities of the app including but not limiting to organizing user’s closet, programmatically suggest outfits, keep track of outfit history, and managing laundry bag.

#### 3.2.4 Data layer

This layer includes two main components: the storage to store pictures and database to store smaller information about the user’s profile, clothing items, outfit history, and look-up tables for matching service. The purpose of this layer is to provide data management for the app. SD card was chosen over Cloud storage as for picture because we want to keep the picture-retrieving latency low which will help the responsiveness of the app. For small information, we use Android built-in database SQLite to manage. One thing to note is for each item, only the path to the SD card is saved in the SQLite of the item table and this is how we link the information in the SQLite database and the pictures in SD card.

An overview of the data structures used with the above data storages can be seen in Figure 3.8.

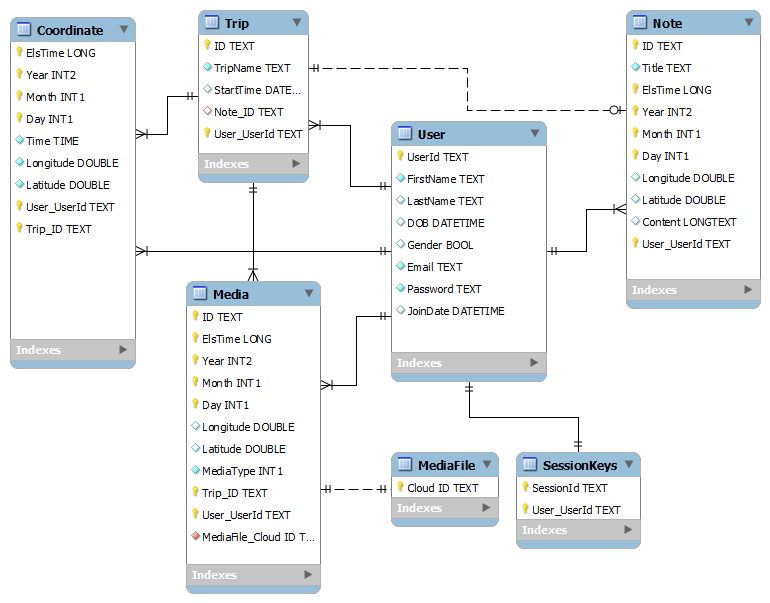


Figure 3.8: Database structures.

#### 3.2.5 Design patterns

##### 3.2.5.1 Factory Method Pattern

This pattern is applied to create different concrete storage types. SD card is chosen at the moment is the main storage for pictures, but the design is opened to use another type of storage with much of the code intact because the creator class is written without knowledge of the actual products that will be created. In other words, the implementation of the product is decoupled with its use. In addition, new storage it will not affect Creator class.

Below is the design



The abstract Creator class is StorageFactory and the concrete Creator classes are SDCardStorageFactory and GoogleAppEngineStorageFactory. The Product is StorageInterface and the concrete Product classes implementing this interface are SDCardStorage and GoogleAppEngineStorage.

##### 3.2.5.2 Abstract Factory Pattern

Clothes Matching service comprises of five steps. While step one and two can be filtered by querying the database, steps three to five require more complicated implementation based on gender (limited to male and female for now). Abstract Factory pattern was applied to provide an interface to create a family of matching steps: occasion matching, pair matching, color matching. Writing code that uses this interface helps us decouple our code from actual factory that creates these concrete matching steps (i.e., object classes). This also lets us expand to a variety of genders if we need to in the future. Once the users have registered their genders, we can substitute assign the correct matching steps.

In our design, the AbstractFactory interface is the ClothesMatchingComponentFactory, and the ConcreteFactory classes are ClothesMatchingComponentFactoryMale, and ClothesMatchingComponentFactoryFemale. There are several AbstractProduct classes OccasionMatching, PairMatching, ColorMatching and the corresponding concrete Product classes are OccasionMatchingMale, OccasionMatchingFemale, PairMatchingMale, PairMatchingFemale, ColorMatchingDefault.

Figure … shows the diagram of AbstractFactory classes and ConcreteFactory classes.



Figure … shows the diagram of AbstractProduct classes and ConcreteProduct classes of OccasionMatching, PairMatching, and ColorMatching.







##### 3.2.5.2 Template Pattern

This pattern is applied to the ClothesMatching class to encapsulate the five-step algorithm described above. ClothesMatchingMale and ClothesMatchingFemale are the two subclasses of ClothesMatching and we can modify the implementation steps if we need to tailor our need for each gender. This provides a framework to plug in new gender in the future. Besides the algorithm lives in one place (ClothesMatching class) and it is easy for code change later. ClothesMatching focuses on the algorithm and let subclasses such as ClothesMatchingMale and ClothesMatchingFemale redefine certain steps of that algorithm without changing the algorithm’s five-step structure.

##### 3.2.5.2 Other Design Patterns

Other design patterns applied in this app including strategy pattern, command pattern, etc. Some of these are inherited from Android architecture (e.g., AsyncTask for Command Pattern), while some others are used to implement features in this app.

#### 3.2.6 Mock-ups

### 3.3 Class diagrams

As mentioned in Section 3.2, iTrak employs the concept of separation of concerns. Therefore, separate modules for specific needs were created to maximize flexibility and extensibility. In the mobile class diagram (see Figure 3.9), the Activity classes are responsible for the presentation of the application, the Adapter and Listener classes handle internal data and events, and the Comms class allows communication between the mobile app and the web services.

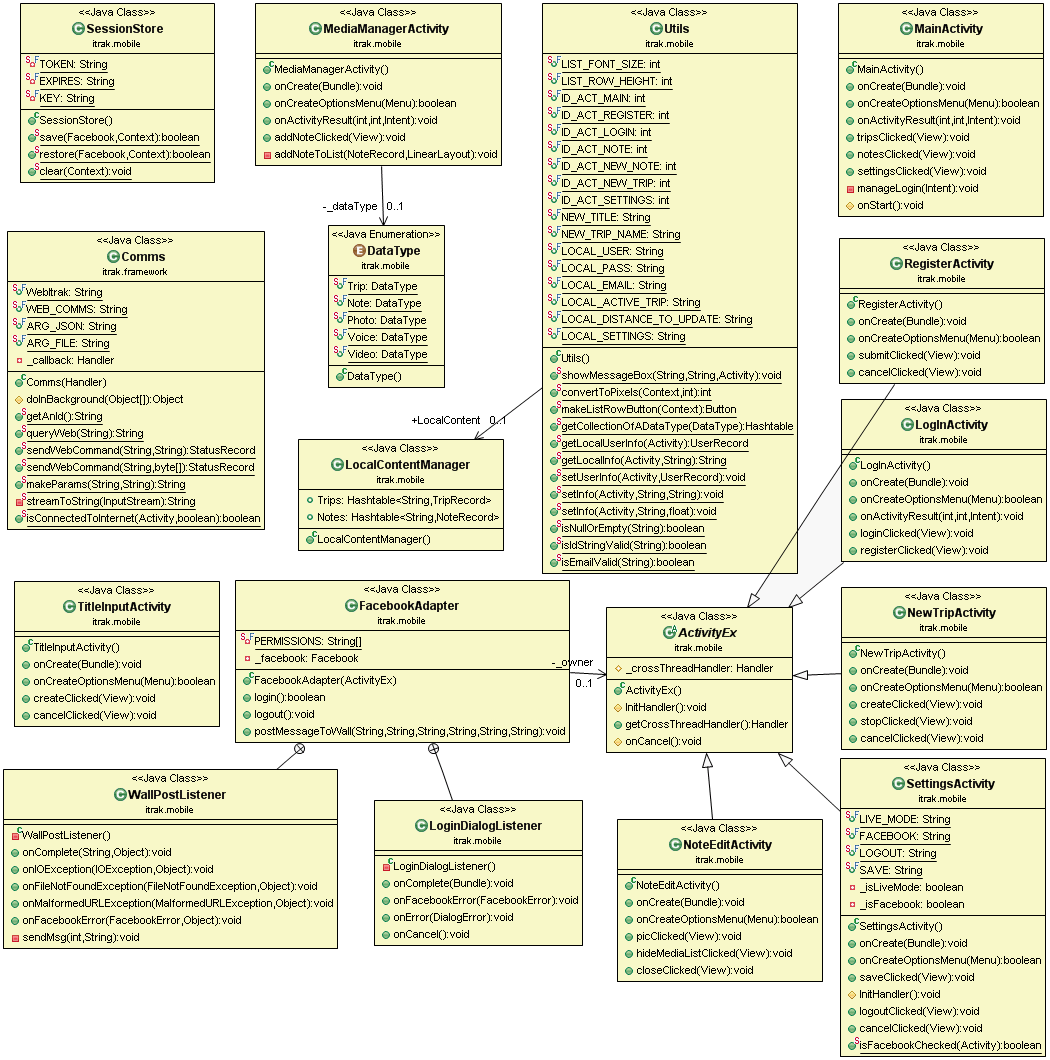


Figure 3.9: Mobile class diagram.

On the other hand, Figure 3.10 shows the web class diagram where the record classes hold persistent data. The IdGenerator class generates unique ID’s for both mobile app and web app. The Comms class filters requests that are sent to the web services and forwards valid requests to the Data Adapter. Finally, the DataAdapter class does the actual processing of these requests.

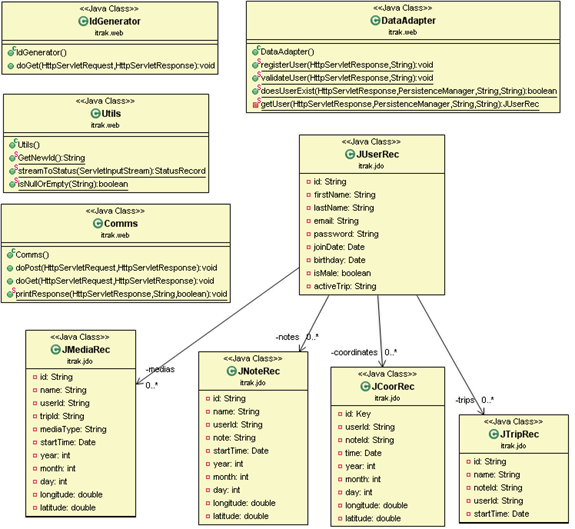


Figure 3.10: Web class diagram.

## Chapter 4 Results

iTrak involves a wide range of technologies and services, from GPS data acquisition to integration with social networks. Therefore, it would be vital to assure that the application delivers accurate data, operates correctly, and has reasonable performance. The following sections describe the main tests conducted for iTrak.

### 4.1 Outfit Of The Day Result

This is one of the most important features of this app, if not the most important one. Therefore, many different trials were run to change the knobs of the five-step clothes matching service. We tried to make the temperature filter dependent on both style and material, but material turned out not to be a good factor (as explained above, because we could layer it up). When we found the suggested list does not change too much while switching among Occasion options, we tried to increase the scale of Occasion matching and it was helpful to make the algorithm match what we expect.

There are certain limitations to the algorithm:

If the closet does not have many items, the service will offer very similar outfits.

If there are many items with the same style, the algorithm does not work quite well????

At the moment, there is not much difference between Formal and Semi\_Formal, Day\_Out and Night\_Out.

When the weather is cold, the outer item still stays the same because of the current implementation of choosing the outer first.

deployed to make the GPS data stays at the heart of the tracking feature. It is important to assure the GPS locations acquired by the smart phone are accurate. There are two methods to measure the accuracy of the acquired data for a route:

* By visually comparing the supposed route with the actual route showed on the iTrak map view.
* By computationally compare the binary data acquired with the supposed data. This method requires preparing a set of data that reflects the expected route. This activity can be achieved by entering a route on Google Maps and then using the GMapToGPX converter [Gcon] to convert the map information to GPX data, which can later be used in the computational comparison.

The GPS data acquisition testing could also be done with different settings of the smartphone. Nowadays, many smartphones are equipped with assisted GPS (AGPS), which is the GPS service using both GPS receiver and the cellular signal to improve the performance for acquiring GPS data [Agps]. However, during a travel trip, user may get to a place where cellular signal is not available. Therefore, iTrak should be tested with both situations when AGPS service is enabled and when it is not.

The iTrak prototype was tested with both visual and computational comparisons using non-assisted GPS receiver on the Nexus S phone.



Figure 4.1: Trip overlay tested in Houston area.

Figure 4.1 showed the results of a test conducted by driving a car around a neighborhood in the Houston area at an average speed of 20miles/hr. The blue lines show the trip recorded by the GPS receiver on the Nexus S with the distance-to-update set to 15ft. The red lines show the actual trip. The results show that while the recorded coordinates were somewhat accurate, there was a significant delay on the reception of the coordinates that made the number of received coordinates much fewer than expected, which resulted in an inaccurate overlay of the trip on the Google Map. In this test, the path highlighted in red covers a distance of about 550ft. An approximation of 36 points should be recorded given the 15ft distance-to-update option. However, only 5 coordinates were actually recorded. There were several factors that might affect this delay such as operating in the non-assisted GPS mode, geographical nature of the location, or the quality of the GPS receiver itself. Nevertheless, more testing would be needed to find the exact problem and to improve the tracking feature.

### 4.2 Picture Display

iTrak mobile client supports uploading media such as photos or videos attached in a note to the cloud server. Therefore, the upload process should be tested for efficiency and completeness. The media files are uploaded when a note is synchronized by using the Sync button (Figure 2.8). The following situations have been tested using Wi-Fi connections:

* Upload a large media file: individual photo and video files with average size varying between 1MB and 3MB were uploaded to the cloud server. The average upload speed showed good performance at an average of 113KB/s.
* Upload multiple media files together: this test was conducted to test the completeness and fault tolerance of the media uploading feature, based on the following criteria. First, all media files attached in a note must be completely uploaded before the note content is synchronized. Second, if iTrak failed to upload a media file, it should cancel the upload process, save the note content locally, and allow user to synchronize again later. Third, if a media file had already been uploaded earlier, iTrak should not upload the file again. Finally, iTrak should maintain a waiting dialog during the upload process.

For all criteria, iTrak has successfully passed the tests.

### 4.3 Weather service and location service latency

ds

### 4.4 UT with robotium

4.3 Weather service and location service latency

The interaction of iTrak with Facebook was tested and confirmed working with both the mobile app and web app. Figure 4.2 shows the Facebook status that was updated after a trip was created using the mobile app, while Figure 4.4 shows the web app with the content of the note that associates with this trip. On the web app, a note with some text, a photo, and a video were presented. At the bottom of the web page is the Facebook area where visitors can like the note, send the note to friends, or comment on the note. Although the web app shows the note with rich content of the media, the actual content of the note on the mobile app was minimal as showed in Screenshot 4.3b in Figure 4.3.

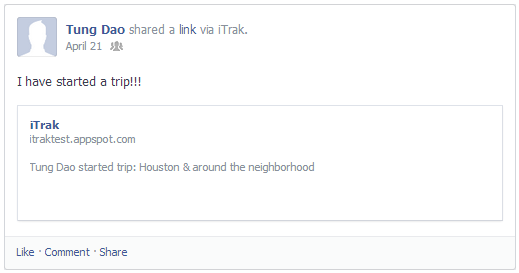


Figure 4.2: Facebook status update screenshot after a trip was created.

|  |  |
| --- | --- |
| Screenshot 4.3a | Screenshot 4.3b |

Figure 4.3: Mobile app screenshots.

Screenshot 4.3a shows the main page of the mobile app with all the buttons that allow performing different tasks as designed in Mockup 2.7a and Table 2.1.



Figure 4.4: Web app screenshot with one Facebook Like and one comment.

### 4.4 Costs and level of effort

The development has been done on free trial services with Assembla [Asbla] and Google. Therefore the development cost was minimal. Table 4.1 shows the cost of equipment and services purchased for the iTrak prototype.

|  |  |
| --- | --- |
| Item | Costs |
| Nexus S phone | $110 |
| MyWi hotspot application  (used to allow hotspot connection from the Nexus S to my iPhone for uploading GPS data during testing) | $25 |

Table 4.1: iTrak prototype development costs.

The iTrak project has consumed a significant amount of time during the past ten months. The hours spent are approximated as follows:

* 40 hours for architecture design.
* 40 hours for GUI design.
* 260 hours for researching, coding, and testing.
* 80 hours for writing the report.

4040 lines of code were written for the mobile client and 1588 lines of code were added for the web services. Figures 4.5 and 4.6 show the code metrics of the mobile client and the web services for further details. These metrics were calculated using the CodePro Analytix plugin [Cpro]. A red item shows an area that exceeds the metric threshold and may need improvement. For example, if average cyclomatic complexity is red, there may be too many nested IF statements or nested FOR loop statements in the code.

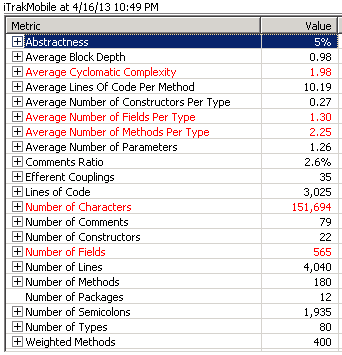


Figure 4.5: Metrics for mobile client.

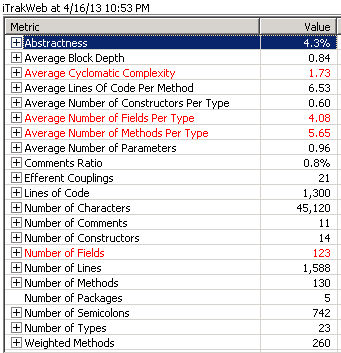


Figure 4.6: Metrics for web services.

### 4.4 Lessons learned

My experience with Eclipse and the Android development as well as the Google App Engine development involved a number of challenges and difficulties that consumed a significant amount of time to resolve. It would be helpful to share some of the lessons learned with developers who would be developing similar apps. The following are my top things to do and not to do.

Do:

* Think about designing a base class for Android Activity that handles multi-threading processes that involve GUI interactions. This will tremendously reduce effort of handling and debugging multi-threading code later.
* Consider code isolation for Javascript used with the web app. This will help avoid conflict with third party Javascript components, which is difficult to identify and debug. Based on the design of the application, an approach for code isolation could be applied. For example, for third party component such as VideoJS [Vjs], the component could be placed in an iframe tag to avoid conflicts with other components. In another example, global variables could be moved to inside an object to avoid conflicts with third party components that might already define global variables with the same names.
* Consider building a separate component that holds static resources and code that can be referenced by both Android project and AppEngine project. This will reduce effort of maintaining similar code in both mobile and web clients.

Don’t:

* Maintain the datastore indexes locally. Let the App Engine handle it automatically when the Persistent Manager interface is called. Editing the datastore indexes locally may make it unstable.
* Mistakenly assume the order of longitude and latitude on the Google Maps website is the same with what should be in the KML format. It depends on where latitude and longitude need to be set. If longitude and latitude are set using Javascript on the web app, the order needs to be latitude-longitude. However, in the KML file, the order must be in the longitude-latitude order. Bugs caused by wrong order of longitude and latitude in the KML file are extremely hard to identify. I spent almost two months to realize that the real-time data update using Google Maps did not work was because of the wrong order of longitude and latitude.

## Chapter 5 Conclusion

### 5.1 Summary

The prototype has been implemented closely to the architecture requirements. Separate modules for specific concerns were created to maximize flexibility and extensibility. The prototype delivered the features as defined in the scope of the report. It also demonstrated the technology and techniques used in the original design. The following captures the key items achieved from the prototype:

* Demonstrated the main workflow to show how the GPS location data and travel journal can associate with each other, and how social network can interact with iTrak to enhance the travel experience.
* Implemented the communication interface using GSON/JSON and proved that this interface significantly reduced effort in handling data exchange between mobile client and web services.
* Experimented with various technologies, including Android, Google App Engine and Google Maps, and showed how they worked together.

Although the prototype has incorporated a variety of technology and integrated a number of exciting features, there is still room to improve. While the UI of the mobile app is well organized, the UI of the web app is still very basic and needs more styling. Existing features such as recording video or recording sound can also be enhanced. More details about related improvements are discussed in Section 5.3.

In conclusion, with a dynamic architecture and a set of features that pick up current trends such as social network and GPS services, the outcomes of this report and the prototype have provided a good foundation for future implementation of iTrak in a full scale.

### 5.2 Related work

As mentioned earlier, one of the advantages of iTrak is to avoid the hassle of maintaining separate applications for different activities a traveler may perform. The following applications are examples that demonstrate the related features supported in iTrak as well as the features in iTrak that these applications do not have.

#### 5.2.1 EverNote

EverNote is an emerging note taking and activity planning application that is available in various systems. The application allows a user to easily take notes of everyday activity. The user can access the notes from different devices such as personal computers, smart phones, or tablets. EverNote will assure the changes made to the notes are synchronized. However, EverNote does not associate with GPS location data and live map view. Therefore, travelers who use EverNote would need to use a separate application for the trip tracking purpose. [Eve].

#### 5.2.2 Video journal devices and apps

Nowadays, travelers often bring a voice recorder or a video recorder, along with the traditional notebook, to record their traveling experience. These recorders can be found on the Internet in the form of portable digital camcorder, such as the Pure Digital PSV-351. User can also use default media recorder applications available on many smart phones to make video or voice journals. However, these devices and applications work independently and require travelers to take extra effort to maintain them.

#### 5.2.3 RunKeeper

RunKeeper is one of the few applications that have tracking ability similar to the iTrak’s vision. The application allows a user to plan a workout activity and then keep track of how much the user has completed the plan based on the GPS tracking data. RunKeeper also allows the user to share the activity with people on the user’s social networks like Facebook. However, RunKeeper does not support note taking or journal recording activities. At the moment, RunKeeper offers RunKeeper Live service which is similar to the live map view in iTrak. However, users would have to pay monthly service fees to use this feature [Run].

### 5.3 Future work

The prototype of iTrak covers only a small set of features as a proof of concept. More work still needs to be done to complete the application as in the iTrak vision. In addition to the main development of iTrak, other features and activities to improve iTrak functionality and usability can be considered. Some of them are discussed in the following sections.

#### 5.3.1 Detect the item’s color automatically

HTML5 is an emerging language for building website. Many web browsers have already supported HTML5. There are major changes in HTML5 to support media elements and scalable vector graphics (SVG). It is considerable to migrate the iTrak web app to HTML5 in the future for enhanced graphical presentation.

#### 5.3.2 Integration with social networks such as Facebook

We were able to login to Facebook but we did not have enough time to implement sharing the outfit on Facebook because sharing images of the outfit is a complicated. Given that this app requires displaying pictures, Twitter may not be a good social network to share, Instagram may be a better choice.

Currently the iTrak prototype only supports Facebook. It is important to expand the feature to include other social networks such as Tweeter to attract more users.

#### 5.3.3 Implement speech-to-text to fill in required fields

#### 5.3.4 Integrate with Calendar to suggest based on schedule. Add support for travel by letting customer choose what to wear in the future (by recommending based on future forecast). Integrate with ESPN. Look at doc about the ClosetStylist for more.

The in-app media functionalities such as recording video or voice journals are based on the Android SDK, which does not support incremental recording. Therefore, iTrak user currently cannot achieve a true voice or video journal. It would be impossible to continue recording a previously saved video or voice journal. Integration with an open-source codec such as JCodec [Jcod] would allow merging multiple media files into a single record and make it a true journal.

#### 5.3.4 Make media players compatible with multiple platforms

Due to the limitation of the Android SDK, iTrak can offer only MP4 as the format for both video and sound records. It would be essential to also support other common formats such as MP3 for sound records and OGG, WEBM, or SWF for video records [W3s]. Similar to customizing the media recorder, this can be achieved by employing an open-source codec in a server client, which will handle converting uploaded media into different formats.

#### 5.3.5 Improve GPS data and map overlay

As showed in Chapter 4, the quality of the GPS data needs to be improved. Implementation of sophisticated filters for GPS data could be considered to address this issue. Algorithms for normalizing GPS data can also be developed to adjust the recorded GPS data and allow the map overlay fit in the actual geographical area.

## Appendix: Balsamiq Mockups

Mockups used in this report were created by using the Balsamiq Mockups software. Balsamiq Mockups is a commercial application that offers a user friendly interface for creating mockups. The interface of the application is intuitive and fast. The application also provides a good set of features to export the mockups into different formats, including XML, PNG, or PDF. Figure A.2 shows the Balsamiq interface with the iTrak mobile app mockups loaded.

The trial version of the software lasts only seven days. After the trial period, user would not be able to save and load the mockups. However, Balsamiq still allows loading a design through the XML importing/exporting interface. Therefore, for academic purpose, a student can continue using Balsamiq after the trial period has expired, by manually saving the design as in the following steps:

1. To save a design, export the design to XML and save the XML onto a file. If changes are made later for the design, export the XML again and overwrite the content of the file with the new XML.
2. To load a design, open the saved XML file and copy the content to the clipboard. Then, paste the XML to the import dialog in Balsamiq. Figure A.1 shows the import interface with the XML content pasted into the import dialog.

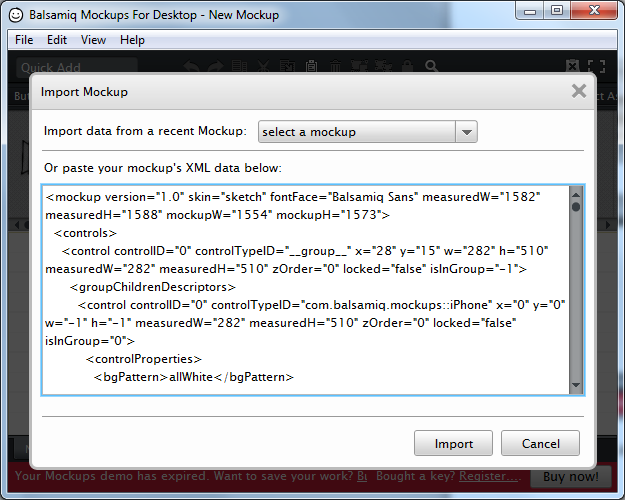
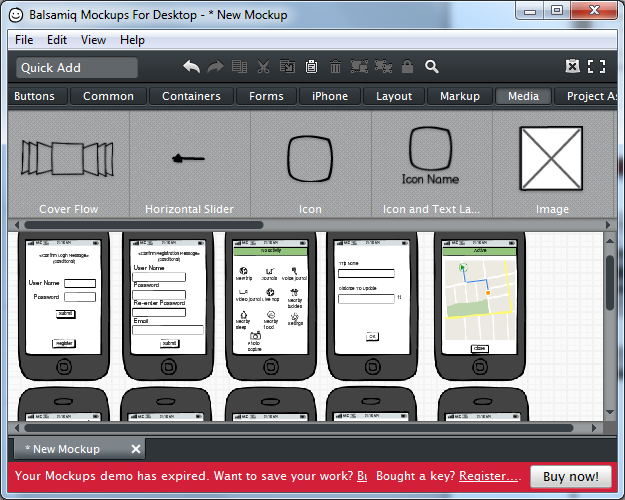


Figure A.1: Balsamiq Mockups XML import interface screenshot.



Tools ribbon to build up mockups

iTrak mockups

Figure A.2: Balsamiq Mockups screenshot.

## References

[TLNF2003] Tjøstheim, I.; Lous, J.; Nordlund, B.; Fuglerud, K.S. 2003. Travelers and Location-information in the Mobile Environment – Consumer Attitudes and a Prototype of a Service for Early Adopters of Mobile Internet: *Norsk Regnesentral, Norway*.

[AV] Axup, J.; Viller, S. 2005. Augmenting travel gossip: design for mobile communities: *OZCHI, Proceedings of the 17th Australia conference on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future, pp. 1-4*

[Cur] Curry, E. 2004. Adaptive and Reflective Middleware: *Middleware for Communications: John Wiley & Sons, pp. 43-47.*

[Mid] Langendorfer, P.; Maye, O.; Dyka, Z.; Sorge, R.; Winkler, R. Kraemer, R. 2004. Middleware for Location-based Services: Design and Implementation Issues: *Middleware for Communications: John Wiley & Sons, pp. 306-329.*

[TOHS] Tarr, P.; Ossher, H.; Harrison, W.; Sutton, S.M. 1999. N degrees of separation: multi-dimensional separation of concerns: *ICSE, Proceedings of the 21st international conference on Software engineering, pp. 107-119.*

[Sol] Soley, R. 2000. Model Driven Architecture: *The Object Management Group, White Paper, Nov 27, 2000.*

[Soap] Box, D.; Ehnebuske, D.; Kakivaya, G.; Layman, A.; Mendelsohn, N.; Nielsen, H.F.; Thatte, S.; Winer, D. 2000. Simple Object Access Protocol (SOAP) 1.1: *W3C Note, 08 May 2000.*

[Https] RFC2818. 2000. HTTP Over TLS: *Network Working Group.*

[Sta429] Stallings, W. 2011. Cryptography and Network Security, 5th edition: *Prentice Hall, pp. 429-437.*

[KP] Krashner, G.E.; Pope, S.T. 1988. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System: *Journal of Object-Oriented Programming archive, Volume 1 Issue 3, Aug./Sept. 1988, pp. 26-49.*

[Fapi] Facebook SDK for Android. [Online].

<https://developers.facebook.com/android/>

[Json] JSON. [Online].

<http://www.json.org/>

[Gson] Google GSON. [Online].

<http://code.google.com/p/google-gson/>

[Bry] Bryce, T. 2005. Re-Inventing Business Process Design. [Online].   
<http://it.toolbox.com/blogs/irm-blog/reinventing-business-process-design-3238>

[Amb] Ambler, Scott. UML 2 Activity Diagramming Guidelines. [Online].

<http://www.agilemodeling.com/style/activityDiagram.htm>

[Sto] Storage Option. [Online].

<http://developer.android.com/guide/topics/data/data-storage.html>

[Bal] Balsamiq Mockups Software. [Online].

<http://www.balsamiq.com>

[Wae] Waele, D. 2010. Understanding the Location Listener in Android. [Online].

<http://blog.doityourselfandroid.com/2010/12/25/understanding-locationlistener-android/>

[GAE] Google App Engine. [Online].

<https://developers.google.com/appengine/>

[Earth] Google Earth API. [Online].

<https://developers.google.com/earth/>

[Map] Google Maps API. [Online].

<https://developers.google.com/maps/>

[KML] Keyhole Markup Language. [Online].

<https://developers.google.com/kml/documentation/>

[Ecl] Eclipse. [Online].

<http://www.eclipse.org/downloads/>

[AWeb] Android Web View. [Online].

<http://developer.android.com/reference/android/webkit/WebView.html>

[Mrec] Media Recorder. [Online].

<http://developer.android.com/reference/android/media/MediaRecorder.html>

[Vjs] VideoJS

<http://videojs.com/>

[Agps] Android GPS. [Online].

<http://developer.android.com/reference/android/location/package-summary.html>

[Eve] EverNote. [Online].

<http://evernote.com/evernote/>

[Run] RunKeeper. [Online].

<http://runkeeper.com/index>

[Cpro] CodePro Analytix. [Online].

<https://developers.google.com/java-dev-tools/codepro/doc/>

[Gcon] Google Maps to GPX conversion. [Online].

<http://www.elsewhere.org/journal/gmaptogpx/>

[Asbla] Assembla. [Online].

<https://www.assembla.com/>

[Jcod] JCodec. [Online].

<http://jcodec.org/>

[W3s] Video formats. [Online].

<http://www.w3schools.com/html/html_videos.asp>

[Agps] AGPS. [Online].

<http://en.wikipedia.org/wiki/Assisted_GPS>

[Act] Android Activity. [Online].

<http://developer.android.com/reference/android/app/Activity.html>

[SWA] Surviving with Android

<http://www.survivingwithandroid.com/2013/05/build-weather-app-json-http-android.html>