WinterI Progress Report
UniversityGear

February 16, 2017

Devin Foulger
Hector Trujillo
Bryan Liauw

Abstract

This document will recap what we have done over the term and will describe what we plan to do in the following year. It will go into detail about the goals we have set and how we plan to accomplish them. It will also include a summary of the project status as it is now. It will also include a retrospective of what has happened throughout the term.

Contents

1 Introduction

Throughout the term, we have been able to accomplish and learn a lot. We were able to design an application that will help people purchase college merchandise. We were able to come up with requirements and analyze the situation in a meaningful way. The team also analyzed many different technologies that we will be using to complete the project.

2 Project goals and recap

The goal of the project is to build an Android application that will allow users to purchase college related merchandise. This means that users will be able to search for many items depending a lot of user provided information. They can search for specific schools and their colors as well as condition of item, or the price. We also set out to accomplish how to create documents in which we could effectively describe our project.

Here our recaps from each term for each week.

2.1 Fall Term

Week 2
During week 2, our group had just been in contact with our client, Luther. We discussed what was expected of the project and what eBay wanted. With this knowledge, we began work on our problem statement. This document outlined what was to be expected from the application that we would be developing. The team also set up a GitHub repo to hold all our documents and code.

Week 3
We completed our problem statement during week 3. After many emails with our client, we came to an agreement with the problem statement.

Week 4
During week 4, we had began to make plans on what needed to be revised in the problem statement. The team had also created developer accounts on the eBay website and began looking at the APIs we would be using. We had also received a small rundown on what requirements will need to be met. Finally, we also began the weekly meetings with our TA.

Week 5
This week, we worked towards completing our requirements document. We had figured out what would need to be accomplished. Our client had also provided us with a mock up of what our application could look like. This helped with the creation of what requirements must be met.

Week 6
Over the course of week 6, the team accomplished a lot. We finished our requirements document and met with our client in Portland. After meeting with the team at eBay, it really felt like they wanted us to succeed. They helped us to better define the requirements that needed to be completed.

Week 7
During week 7, we began work on the tech review. We had split the requirements into three major parts: Searching and gathering data, UI and presenting data, and purchasing the items. After that, each team member focused on the parts that they had been assigned.

Week 8
Over week 8, the team did a mock design of what the application would look like. This was the beginning of the design document. Each member had also completed their part of the tech review. The review featured many different technologies that we would be using, discussing the benefits of each one.

Week 9

Large portions of the design document were completed over week 9. The group created a UML to demonstrate what classes might be used during development.

Week 10

The final week of the term, we completed our design document. However, we still needed to begin work on the fall progress report.

Overall, we set out to accomplish many goals over the term. The end result of each of our documents provides us with a good start for development.

2.2 Winter Term

Week 1

During week 1, we talked about when we were going to meet to do some paired programming. We also uploaded our blank Android project to GitHub to get started. We also spent the time earlier during the term and the break learning how most other website, such as Google and Yahoo, employ their search function, which we ideally want to mimic. However, it turns out Google, which is the benchmark of search engines, employs a Levenshtein Distance method that is used alongside an artificial machine learning that takes in what people often search as probability. This advanced method is not viable for us because we would not have as much traffic, so we decided to stick with Levenshtein Distance only.

Week 2

This week we accomplished creating our search activity, however the class does not currently work with eBay's APIs. Basically what we have done is a basic search function with a textbox that gets user input. We also have a radio button that virtually does nothing but pass in the string that it contains. The input that the users give does nothing yet. We have yet to integrate any functionality from the eBay API. However, the text box does have limited Levenshtein Distance feature, in that we do not have a lot of keywords that user's input can be changed to. This early version of Levenshtein Distance has a lot of bugs that happens because the toleration level for mistakes is too high, which we have corrected.

Week 3

For this week, we were able to get eBay's search API integrated with our search class allowing for us to make searches.This is done through passing in a HTTP requests using an asynchronous task class that is integrated inside the activity. After this milestone, we can finally use the result from the search activity to start working on other activity. However, this API needed authentication, which we arbitrarily pass in. This is not viable for beta release since the authentication, which utilizes oAuth keys, changes every two days. The filters also pass in words instead of actually filtering. On top of that, we also started the list view activity and the item class. The item class will allow us to create items in any class. The list view allows us to view the items by using a recycler view.

Week 4

During week 4, we have been connecting everyone's pieces. This means that we can now go from searching to the list view correctly displaying search results. The list view correctly displays the title, image, and price of the item. The item class is also continually being added to, depending on what new variables we will be needing.

Week 5

We added secondary filter activity as well as a method that gives new oAuth token every time it is called. For now, a new oAuth token is going to be retrieved every time a search is being submitted. The filters now utilize the filter options provided by the eBay API such as the price range and item condition. We also provided item categories to limit the search for the user. We also changed some filter categories and how a search works. Now, prior to actually searching for the item, we asked the user for a particular school first. This is done through typing in the school name in the text box or pressing one or more of the toggle buttons provided and then pressing the next button. It will pass in the value as an extra message for intent. We have not given the ability to pass in both value from the text box and the toggle buttons because we feel that it is not necessary for the user to do. After going through selecting the school, we provide the user with a text box and several check boxes for user to limit their search. Submitting it will send the keywords and filters to eBay's server, which we will return the result afterwards. This process is from one of the examples that eBay actually provided us with.

3 Project status

The design for our application, UniversityGear, has been fairly complete. We have detailed the functional requirements of our project. We have also completed a technology review in which we discovered what technologies would be the most efficient with our application. Last, we created a design document to detail how we were going to implement the technologies we have found into our application. After many weeks of communication with our client, we have arrived at the development phase.

We have developed over half of our application. So far, we have implemented the ability to search, view, and select single items. This alone is a large part of the project. In the coming weeks, users will be able to purchase items, as well as use several filters to find the items that they need. These classes we have created also interact with eBay's APIs. So far, we have used their Browse API, but now we are going to be using the Order API to allow users to purchase the items that they want.

3.1 Search Class Development

The search class is the first activity that the users are greeted by. They will interact with this class first. as it is required to get any results. This search activity will also function as our home page which consists of a simple design that contains a search text field and a search button. Once the user presses the search button, the Search Activity takes the input and searches' eBay's database for related items. We were also able to implement a progress bar that shows up when the user clicks on Submit and disappears when the search results are ready to be presented. This is important because it lets the user know that the application is processing the request.

3.2 Notification's Development

Have implemented the Alert notification that lets the user know that their keyword/s did not return any results. This message is important because it lets the user know that the application is working as it should, and there were simply no items containing the keyword/s that they used.

3.3 Item Class Development

The item class is what allow us to build specific items or a list of items. First, the class will create a list of items that will be displayed to the user in a list item view. Once an item is selected from the list, the item class will build another single item, with much more information pertaining to the item selected. Development for this class is something that will be ongoing throughout the project. This is because more information may need to be grabbed depending on what kind of item is being selected. However, development was rather straight forwarded, as we grabbed the specific information we needed.

It was important that this was one of the first classes that needed to be created. This is because other parts of the project will heavily depend upon this class. This class used the standard JSON methods that are present in Java. These tools allowed us to parse and gather the correct information that we needed. All of the data returned by eBay's APIs are in JSON format. So far, the class will return many things like the items title, price, description, color, material, pattern, type, and category.

3.4 List View Development

Successfully completed the presentation of the search results in a list view. This list view displays the item title, image, and item price for every image that is returned. The user is allowed to scroll through the different items until they find one that catches their attention. We also set up the ”onClickListener” for each item so that when a user clicks on an item that they like, they are presented with more details on that specific item.

3.5 Single Item Activity

The single item activity class is what the users use to view the single item they have selected. This activity will display the pertinent information of an item, like its title, color, price, description, material, pattern, and any other item specific information. This class made use of eBay's Browse API. Specifically, it used the get item method, which retrieves data for a given item.

This activity is started after a user selects a single item from the list view. From there, it passes in the item id to the single item activity. The single item activity then uses that item id in the eBay API. The API then returns a JSON file with all of the information on the item. When searching for things like mugs, it won't contain the same contents as a jacket would. A mug might only show the fields material, while a jacket could have a pattern and a material field. So the content displayed to the user is somewhat dynamic.

The UI for the single item activity resembles the style of the eBay app. The layout is rather simple and it is easy to follow. It is broken into several sections. The first section is the image, title, and price of the item. From there, we have the description and the purchase button. Then there is the about section. It contains the condition of the item, its quantity, category, brand, and other important information. The last section is the shipping and return info. This section states what kind of return policy the item has and how long a user will have to return the item. It also specifies who pays for returns on the item.

3.6 Location-based Recommendation

An implementation that we would like to consider as a stretch goal is using Androids GPS to get user's location and give a list of recommended school based on this information. This could potentially increase the usability of our application. This might not be a hard method to implement as the location is mostly available in one of the Androids built-in function. The difficulty would lie in assigning the location to the schools. This would be time-consuming. Also, this function is definitely not a requirement so we can put this off.

3.7 Utilizing Storage

Storage is not fully utilized yet. Although this does not affect performance as much, we would like to increase the speed on how user will be able to search. While we have a storage method in the search activity, it is a stub currently and does nothing but saving what user has inputted. We need to be able to find a way to integrate this to either filter function or text box function. Also, we need to test the

3.8 Keyword Bank for Levenshtein Distance

We would like to improve on is the Levenshtein Distance keywords list. For now, we have a limited list of keywords. This, while not essentially hindering usage of the app, limits a lot of what could potentially help user in the searches. Currently, our implementation relies on an array that consists of only several words. We would like to use an internal storage to store a list of keywords as well as school names.

3.9 Bug Testing

We also need to test this activity to ensure that it is bug-free. For now, we would want to try passing in empty string and making sure it will not crash. Another error we think might possibly happen is when we return to the previous activity. This will usually happen because the stack is not what we thought it is. This is one of the places where we think it is vulnerable.

4 Impediments and solutions

Over the course of the term, our team managed to avoid many impediments. However, we did need to revise our problem statement. In order to solve this, we re-wrote the problem statement with the revisions provided by the professors of the capstone class.

Another problem is the design documents. The format that is required is not something we are familiar with. There is a lot of questions regarding what to put where. After looking at several examples, as well as asking people, we managed to get it done in the end.

Initially, we had hoped to get started on the project during winter break. It proved difficult to find the energy to work on it after a grueling Fall term. Some of the items that we had hoped to accomplish during this time was to become familiar with eBay's APIs and how to use them. Another goal that we had for the break was to get familiar with the technology that we were going to be using to implement the list item view and the Android notification/alert systems. The failure to work on these goals during the break, didn't have a large impact on the project. The main consequence was that we had more work to do in a shorter period. Also, it did slow down development just a bit as we familiarized ourselves with the technologies being used, however we were able to recover from this slow start to get back on schedule.

We have run into several problems trying to integrate the eBay API. We do not know exactly know what variable they want at first for authentication. Since they have two different oAuth keys, it turns out we were using the wrong oAuth key the whole time. Another problem is that we pass in an incorrect value due to our method of encoding. We thought the encoding would use a default encoding when the value is actually somewhat modified. We managed to solve both these problems by communicating with our clients.

A problem that still remains unsolved is that one of our members broke his gradle build. This becomes a huge problem because he is unable to open changes that we made into his own computer, therefore relying on meetings to catch up with the rest of our work. However, despite this, he can work around it by working on one activity at a time. Since we have our meetings pretty frequently, we can rectify this pretty easily.

5 Remaining Items

5.1 Item Filter Development

One of the larger pieces that still needs to be developed is the implementation of the Search filters. These filters will help the user narrow down their search results to display items that are truly relevant to what they are looking for. For the alpha release we will have the front end complete. This will consist of having the filters available for the user to select, but the will not do anything until after the alpha release.

5.2 Internet Connectivity Verification

Another small, but important item that still needs to be implemented is to have the application check to ensure an active internet connection. If the user does not have an active internet connection, we will display an alert stating that a connection to the internet is a requirement to using the application. This will help in avoiding scenarios where the user tries to search for items, but nothing happens because they are not connected to the internet.

5.3 Purchase Class Development

One of the larger classes, the purchase class, still needs to be developed before the end of the term. This class is what will allow users to purchase an item of their choice. In order to implement this class, we will need to use eBay's Order API. The Order API allows us to use guest checkout, meaning that any user will be able to use our app without a log in.

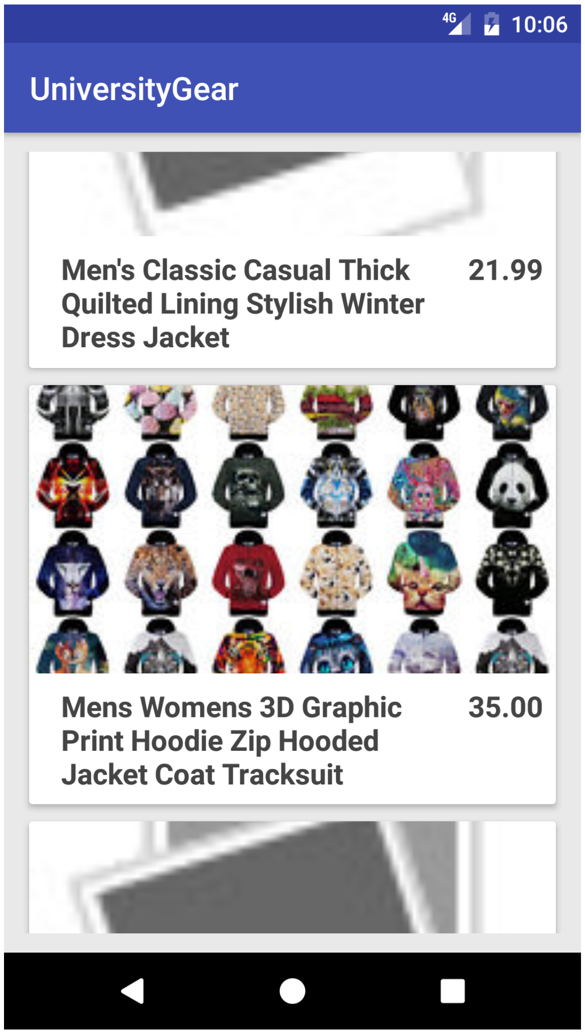
5.4 Known Bugs

Improve or fix the way images are being presented within the current list view results. There are certain items who's images are cropped out because they are either too big or are in portrait mode. Ensuring that the images fit into their intended area will be important so that the users can see the entire image and thus get a good idea of the product that is for sale. There is also a bug in the list view results display that we need to track down and fix. The bug results in images not displaying at all when the user is viewing the search results in a list view. However, the item displays as it should in the single item view. The part that makes it harder to identify is the fact that this bug is only present when working in the sandbox environment. When testing the application on eBay's production servers, the images display properly on both the list view and single item view. It could come down to being an issue with the data in eBay's sandbox environment. To determine this, we will need to do plenty of testing, and discuss with our client eBay potential causes for the different behavior in the different environments.

6 Project Images

This image demonstrates what is displayed to the user after they successfully complete a search. This will list all possible searches. Each item card will display the title, image and price.

Fig. 1. List view of items

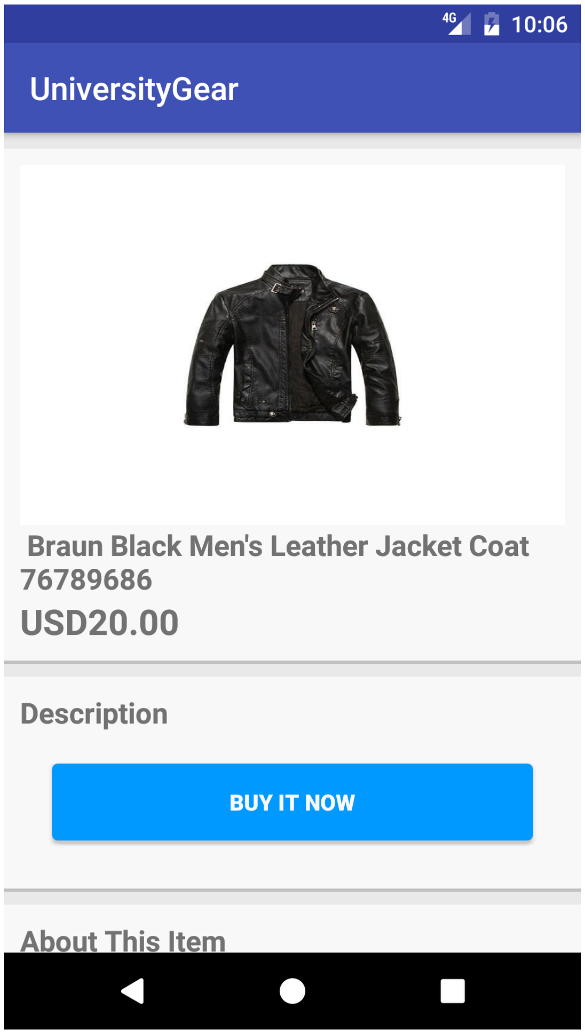


Men's Classic Casual Thick Quilted Lining Stylish Winter Dress Jacket 21.99

Mens Womens 3D Graphic Print Hoodie Zip Hooded Jacket Coat Tracksuit 35.00

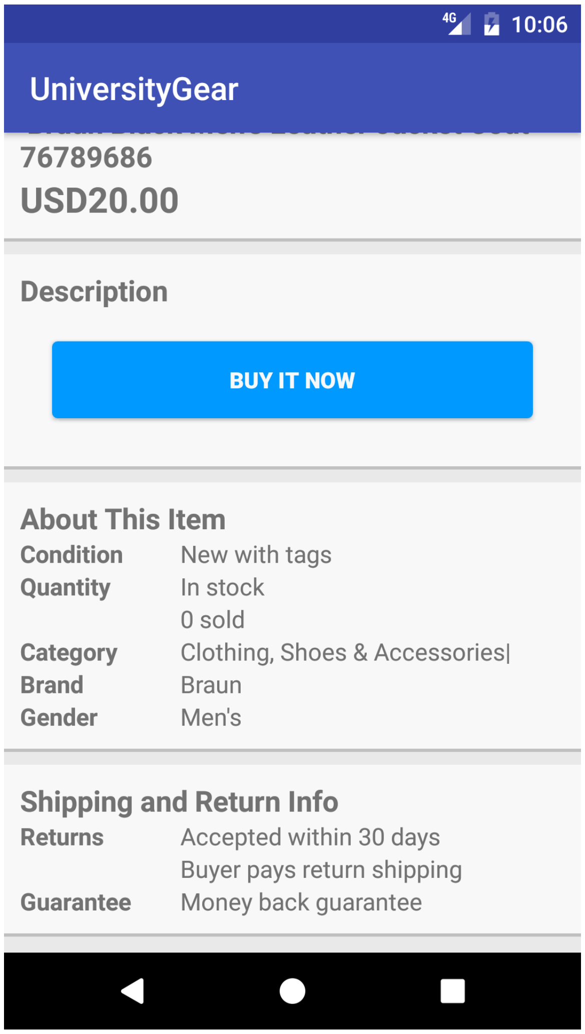
This is an image of what a single item would look like when being viewed. It contains the title, price, and other important information that pertains to the item.

Fig. 2. Image of single item



This is what the rest of the single item view would display to the user. It shows more details about the item as well as shipping and return info.

Fig. 3. Item details of a single item



UniversityGear

76789686

USD20.00

Description

BUY IT NOW

About This Item

Condition	New with tags
Quantity	In stock
	0 sold
Category	Clothing, Shoes & Accessories|
Brand	Braun
Gender	Men's

Shipping and Return Info

Returns	Accepted within 30 days
	Buyer pays return shipping
Guarantee	Money back guarantee

7 Retrospective

TABLE 1
Retrospectives

Positives	Deltas	Actions
Completed the problem statement	Didn't meet the criteria	Rewrote the report to define the problem better
Met with the client in Portland	Could meet with clients more frequently	Arrange more meetings on-line
Completed the requirements document	Not quantifying goals, spelling errors, and missing an abstract	Proofreading it more and asking for more clarifications prior to submitting it
Completed the technology review	Writing could be improved	Go to the Writing Center in order to see missed mistakes
Completed the design document	Can start earlier	Talk more to TA and instructor on clarifying some matters

8 Conclusion

During the course of the term we have learned many things. This includes items from different IEEE formats used in the real world to best practices when gathering project requirements. We also learned how to look at projects from the 10,000 foot level instead of trying to focus on specific details. We will begin development on our project during winter term. However, we are hoping to make progress during winter break. Over half of winter term and we have already accomplished half of what we have set out to do.