HOMEWORK ASSIGNMENT 4

CSCI 571 – Fall 2023

Abstract

Android Studio, JSON, MongoDB, Node.js, Google photos API, and eBay API

Marco Papa

papa@usc.edu

CSCI 571: Web Technologies
Assignment 4: Product Search Android App

Table of Contents

1. Objectives **2**

2. Background **3**

2.1 Android Studio 3
2.2 Android 3

3. Prerequisites **4**

4. High Level Design **5**

5. Implementation **6**

5.1 App Icon and Splash Screen 6
5.2 Search Form 7
5.3 Search Results 11
5.4 Product Details 14
5.4.1 Product Details Tab *16*
5.4.2 Shipping Tab: *18*
5.4.3 Google Tab *21*
5.4.4 Similar Products Tab *22*
5.5 WishList 25
5.6 Summary of detailing and error handling 28
5.7 Additional 30

6. Implementation Hints **31**

6.1 Icons 31
6.2 Getting current location 31
6.3 Third party libraries 32
6.3.1 Google Play services *32*
6.3.2 Volley HTTP requests *32*
6.3.3 Picasso *32*
6.3.4 Glide *32*
6.3.5 CircularScoreView *33*
6.4 Implementing a Gallery view 33
6.5 Implementing Sorting techniques 33
6.6 Working with the AutoCompleteTextView 33
6.7 Implementing a Splash Screen 33
6.8 Dynamic coloring using DrawableCompat 33
6.9 String manipulation in Java 33
6.10 User location using Emulator 34

1. Objectives

- Become familiar with Java, JSON, Android Lifecycle and Android Studio for Android app development.
- Build a good-looking Android app.
- Learn the essentials of Google's Material design rules for designing Android apps
- Learn to use the Google Maps APIs and Android SDK.
- Get familiar with third party libraries like Picasso, Glide and Volley.

2. Background

2.1 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android application development, based on IntelliJ IDEA - a powerful Java IDE. On top of the capabilities you expect from IntelliJ, Android Studio offers:

- Flexible Gradle - based build system.
- Build variants and multiple apk file generation.
- Code templates to help you build common app features.
- Rich layout editor with support for drag and drop theme editing.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- ProGuard and app-signing capabilities.
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

More information about Android Studio can be found at:
http://developer.android.com/tools/studio/index.html

2.2 Android

Android is a mobile operating system initially developed by Android Inc., a firm purchased by Google in 2005. Android is based upon a modified version of the Linux kernel. As of Nov 2018, Android was the number 1 mobile OS, in unit sales, surpassing iOS, while iOS was still the most profitable platform.

The Official Android home page is located at:
http://www.android.com/
The Official Android Developer home page is located at:
http://developer.android.com/

3. Prerequisites

This Assignment requires the use of the following components:

- Download and install Android Studio. Technically, you may use any other IDE other than Android Studio such as Eclipse, but the latest SDKs may not be supported with Eclipse.

We will not be providing any help on problems arising due to your choice of alternate IDEs.

- You must use the **emulator**. Everything should just work out of the box.
- If you are new to Android Development, please refer to the hints section at the bottom.

4. High Level Design

This Assignment is a mobile app version of Assignment 3. In this exercise, you will develop an Android application, which allows users to search for the products on eBay, look at information about them, save some to Wishlist and post on Facebook about the same. You should reuse the Node.js backend service you developed in Assignment 3 and follow the same API call requirements.

5. Implementation

5.1 App Icon and Splash Screen

In order to get this icon/image of the size of your choice, go to the icons web page specified in the section 6.1 and search for an item called “Shopping”. Using the advanced export option, set the colors and a correct size to download the PNG icon.

The app begins with a welcome screen **(Figure 2**) which displays the icon downloaded above. This screen is called a splash screen. This screen can be implemented using many different methods. The simplest is to create a resource file for the launcher screen and add it as a style to AppTheme.Launcher (see hints).

This image is also the app icon as shown in **Figure 1**.

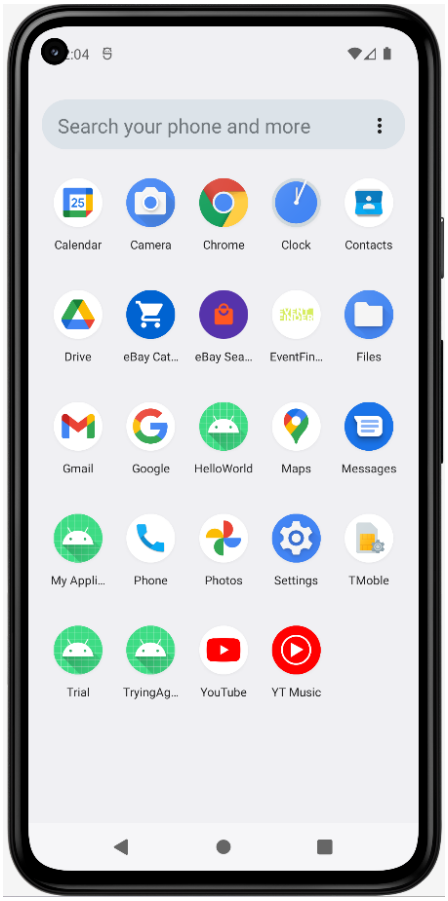

Figure 1: App Icon

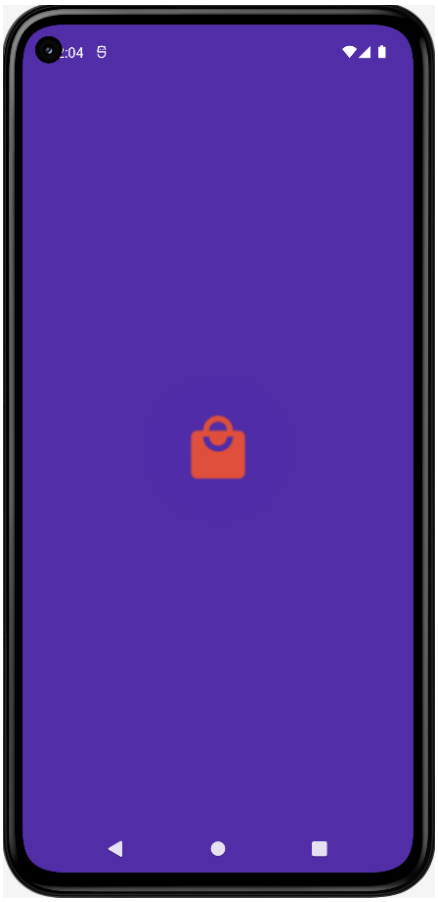

Figure 2: Splash Screen

5.2 Search Form

The initial interface is shown in **Figure 3**. There are two tabs in this interface: search and WishList.

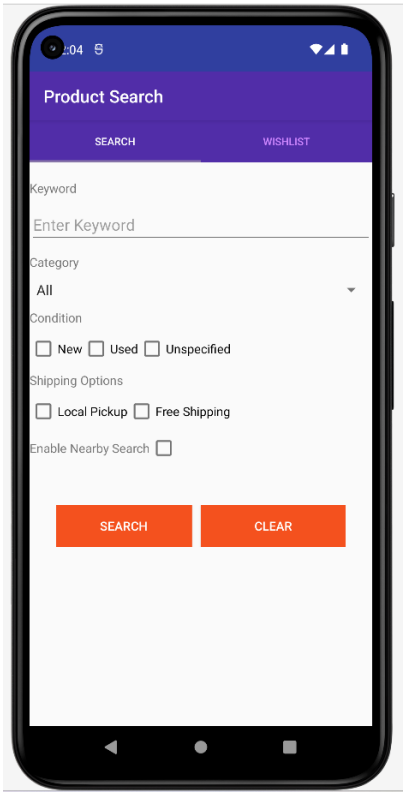

Figure 3: The eBay Product Search App

For the search tab, it has the following fields:

- **Keyword**: An EditText component - allowing the user to enter the keyword.
- **Category**: A Spinner view allowing the user to choose a category. When the user taps on this field, a dropdown list should display for selecting a category, as shown in **Figure 5**. Make sure you include all the categories in Assignment 3.
- **Condition**: It is a set of checkboxes indicating the condition of the item the user wants to buy – new/used/unspecified.
- **Shipping Options** –A set of checkboxes indicating the shipping options the user is interested in - Local pickup/Free shipping.
- **Enable Nearby search**: A checkbox – selecting this will open further distance-based options as shown in **Figure 4** (which are initially hidden as in **Figure 3**) at the bottom. The values of distance-based search considered for eBay query only when this checkbox is selected. The options are as follows:

- **Miles From**: An EditText (type: number) allowing the user to enter the distance and the default value is 10.
- **From**: Two Radio Buttons to select “Current Location” or “Zipcode”.
  - **Current Location:** You can get the location from your emulator (see hints) or from ip-api calls, for users that choose “Here”.
  - **Zip Code**: An AutoCompleteTextView - It provides the autocomplete function as shown in **Figure 6**. Make sure you use the same API as Assignment 3. See hints. When the keyboard is opened, the layout should slide up so that the zip code textview is not hidden by the keyboard.

- **Search**: A button to get the input information of each field, after validation. If the validation is successful, then the products would be fetched from the server. However, if the validation is unsuccessful, appropriate messages should be displayed and no further requests would be made to the server.
- **Clear:** A button to clear the input fields and reset them to default values when applicable. It should also remove any validation error messages.

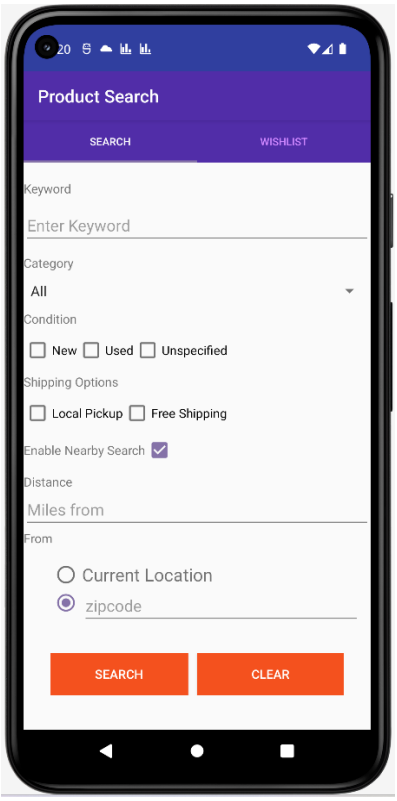

Figure 4: Distance based options

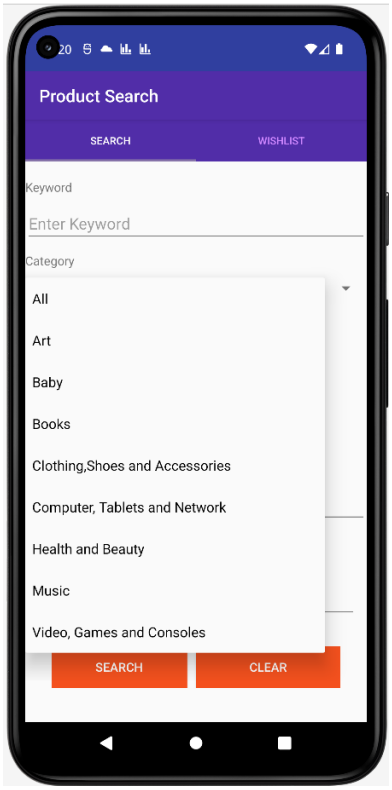

Figure 5: Category Spinner

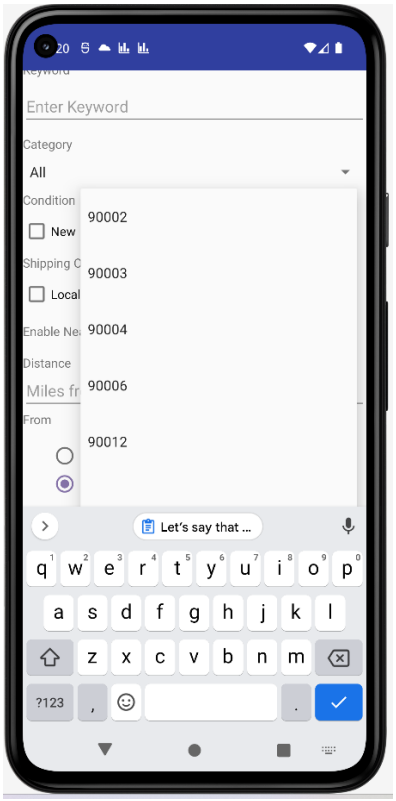

Figure 6: Autocomplete for zipcode

The validation for an **empty keyword** has to be implemented. If the user does not enter anything in the EditText or just enters some empty spaces, when he/she presses the Search button you need to display an appropriate message to indicate the error, as shown in **Figure 7**. The same should be done when "**Zipcode**" location is not entered, and that option is enabled using the radio button as shown in **Figure 8**.

Whenever a validation error is found, along with the error form, the error message should also be displayed on the bottom of the screen using a "Toast" message as shown in the figure.

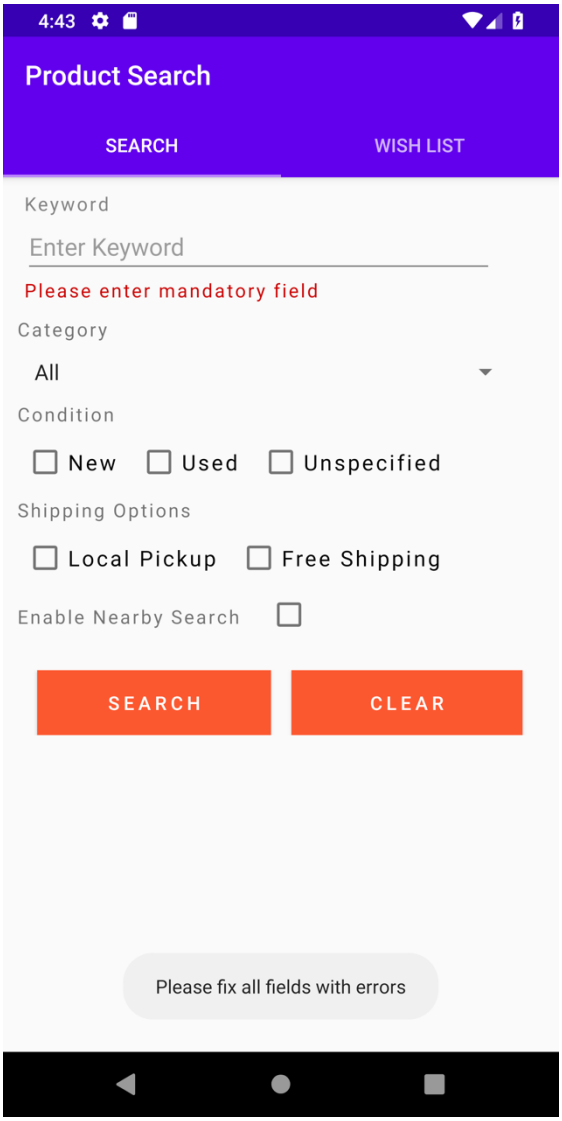

Figure 7: Validation error messages

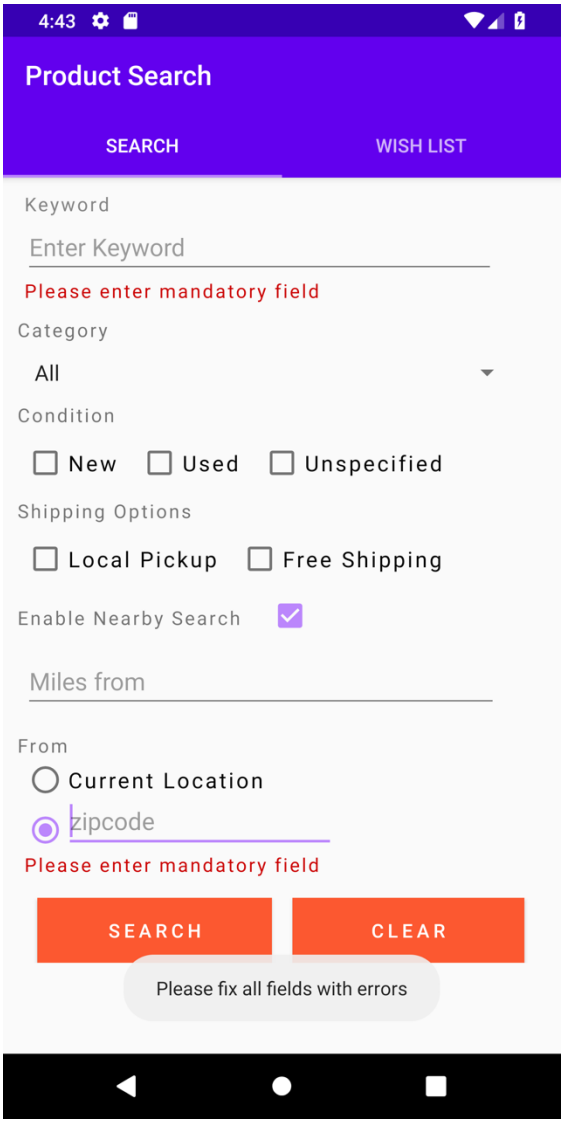

Figure 8: Validation error messages

f5.3 Search Results

When the user taps the SEARCH button and all validations pass, your app loads the search results page.

Before you get the data from your backend server, a progress bar should display on the screen as indicated in **Figure 9**.

After you get the data from your backend, hide the ProgressBar and display the result page as a list using RecyclerView, as shown in **Figure 10.**

The RecyclerView must be scrollable. There should also be a ‘back button’ to navigate back the search/wishlist interface.

The list view should be displayed using GridLayoutManager with CardLayout. Each of the item in the list should have the following fields:

- Product Image
- Title of the product
- Zip Code
- Shipping cost
- Product cost
- Condition
- A button to add to wishlist

See Assignment 3 for more details about these fields.

You can fix the size of the card to any size, but all the cards should be the **same size**. To do this you MUST do the following:

- Fix the size of the image and wishlist icon on the card
- All the images should be centered inside the ImageView and should NOT be cropped
- Fix the title to span 3 lines (or any fixed number) – if there are less than 3 lines then append/prepend a ‘\n’ characters. Fix the layout of zip code and shipping information lines
- If any of zip code/shipping info/price/condition are missing – it should show N/A
- Ensure that the price is the last line and is shown at the bottom of the card

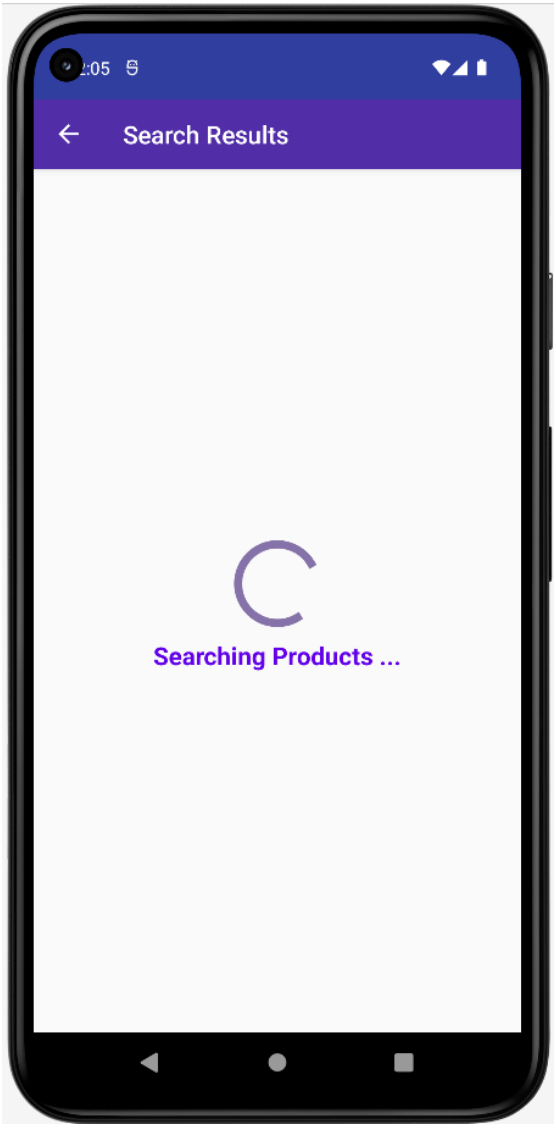

Figure 9: ProgressBar while fetching results

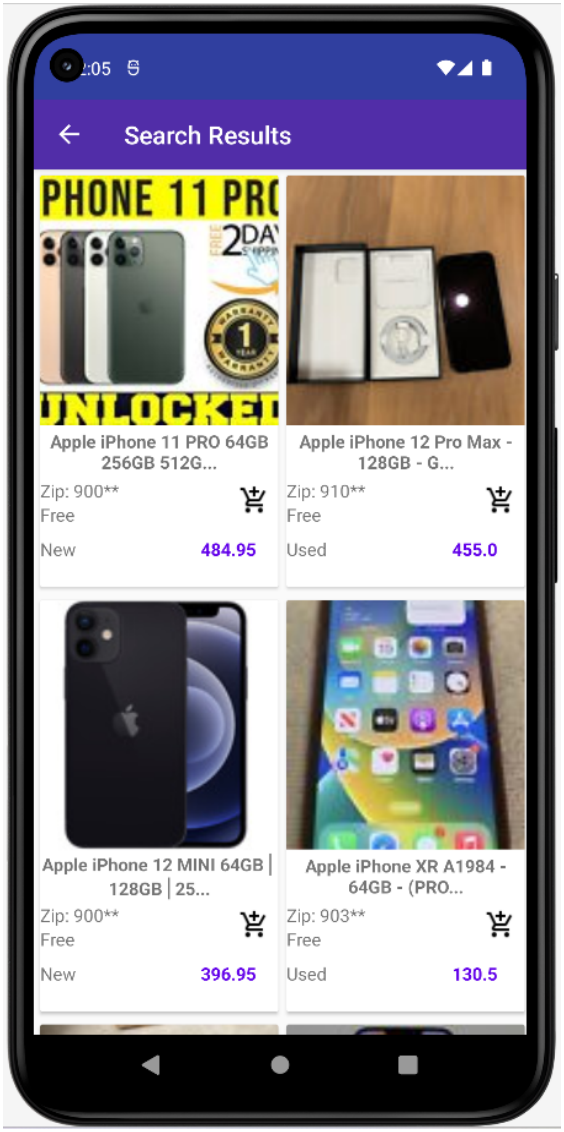

Figure 10: List of search results

Tapping the Cart button adds the corresponding product into the wishlist, and a message is displayed at the bottom of the app using a Toast, as shown in **Figure 11**. Tapping the button again would remove that product from the wishlist, and a similar message should also be displayed to indicate the product has been removed from the wishlist, as shown in **Figure 12**.

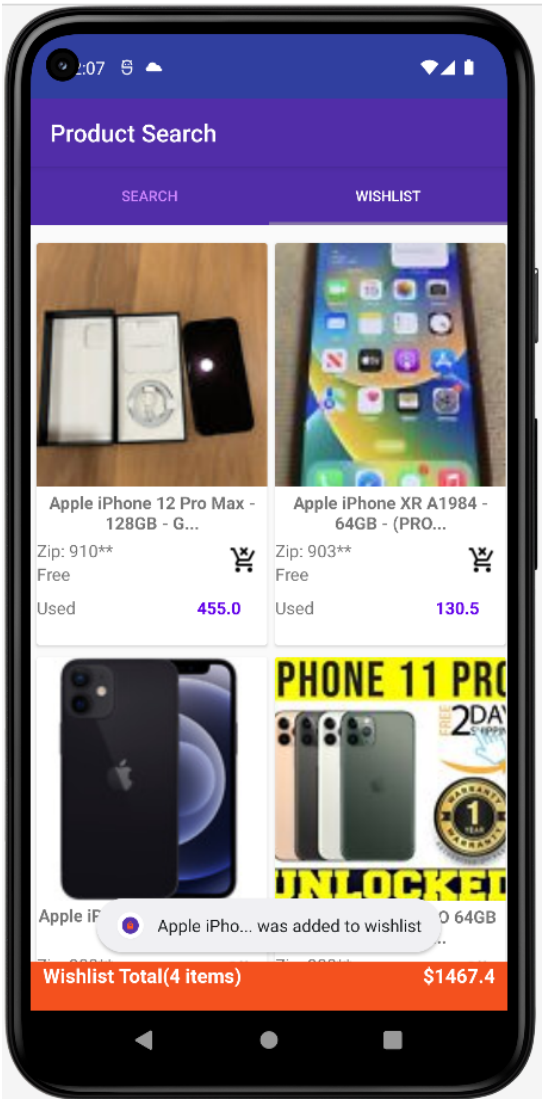

Figure 11: Message for adding to wishlist

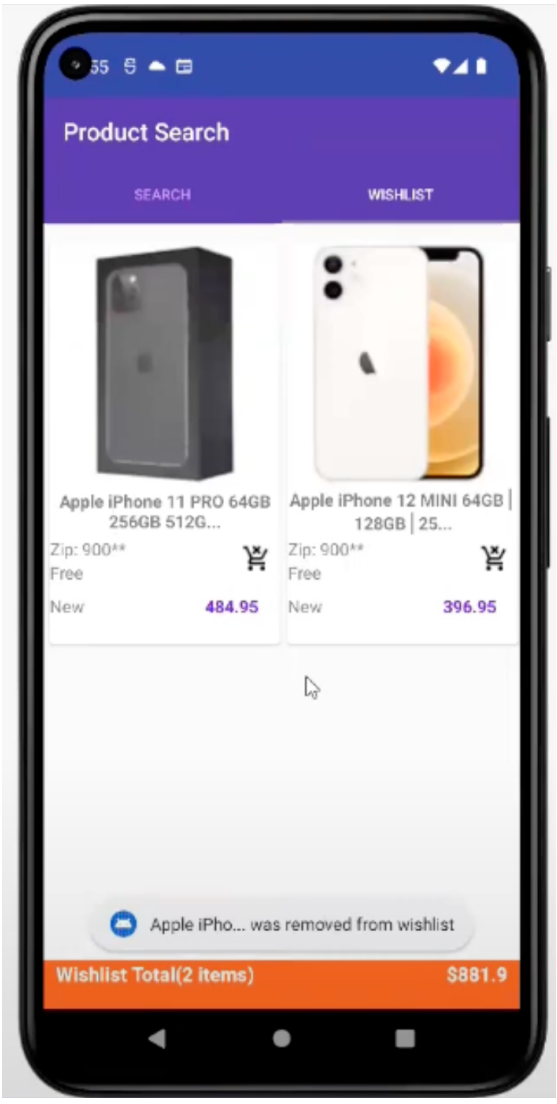

Figure 12: Message for removing from wishlist

5.4 Product Details

Tapping on an item in the result list should show details of that product with four tabs.

There are 4 tabs displayed for every product:

1. Product Details Tab
2. Shipping Tab
3. Google images Tab
4. Similar Products Tab

Note that the Progress spinner should be shown on each tab before you are ready to display the corresponding tab, as shown in **Figure 13A.** Replace “Product details” in the progress bar message with the respective tab names for other tabs: “Shipping details”, “Photos” and “Similar products”.

The tabs should be attached to the ActionBar and a ViewPager should be used to host all the tabs, as shown in **Figure 13B**. Users should be able to switch between tabs by both swiping and tapping on a tab. Please refer to the video to see this in action.

The ActionBar should also include the following elements:

- **Back button:** Navigates back to the search results list WITHOUT page refresh. See video for more details.
- **Title**: which is the name of the product.
- **Facebook button**: to share the product details on Facebook. Once the button is tapped, a web page should open to allow the user to share the product information on Facebook, as shown in **Figure 14**. This should work the same as Assignment 3. Note the additional hashtag that is added to the Facebook post.
- **Add to wishlist button:** This button will add/remove the product to/from the wishlist. This can be implemented using a FloatingActionButton. The icon of the button should also change based on whether the item currently belongs to the wishlist or not. For more details see video.

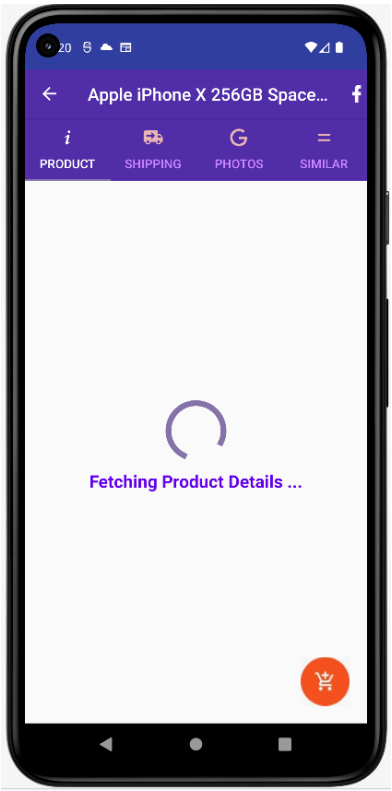

Figure 13A: Product details Loading Spanner

Figure 13B: Product details

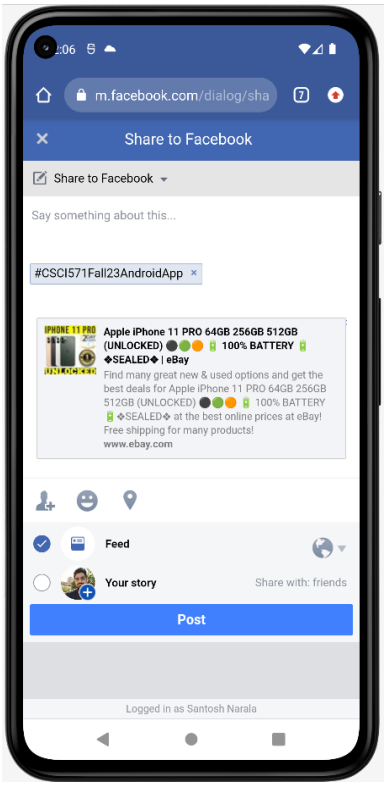

Figure 14: Share Product on Facebook

5.4.1 Product Details Tab

This tab contains the following sections (See **Figure 15** and **Figure 16**):

- **Product images:**
  - PictureURL property in json.
  - This is a minimal version of the image gallery. The user of the app will be able to browse through the images provided by eBay. This can be implemented by using a HorizontalScrollView (see hints section). Please note there are many other ways to implement this, you can use any of those.
- **Product Title:**
  - Title property in json
  - A textview with a big font which contains the title of the product.
- **Price and shipping:**
  - CurrentPrice and ShippingCost properties in json
  - A textview with Price and shipping cost. If the Shipping cost is $0 then display “With Free Shipping”, otherwise show “With $XX Shipping” with XX being the shipping cost
- **Highlights:**

- Price: price of the product - CurrentPrice property in json
- Brand: Brand of the product – Brand property in json

- **Specifications:**
  - Display all the values inside ItemSpecifics of the JSON from eBay.
  - Brand has to be the first value in this list.
  - For each value capitalize the first letter.
- **Notes:**
  - **JSON property names are written for reference. For exact structure please refer to Assignment 3.**
  - **Each of these sections are separated with a faint gray horizontal line.**
  - **If any of the details are missing, skip that row.**
  - **If for a particular section <u>ALL the fields</u> are missing then skip the entire section.**
  - **If <u>ALL sections</u> are missing, then display an appropriate no results message.**

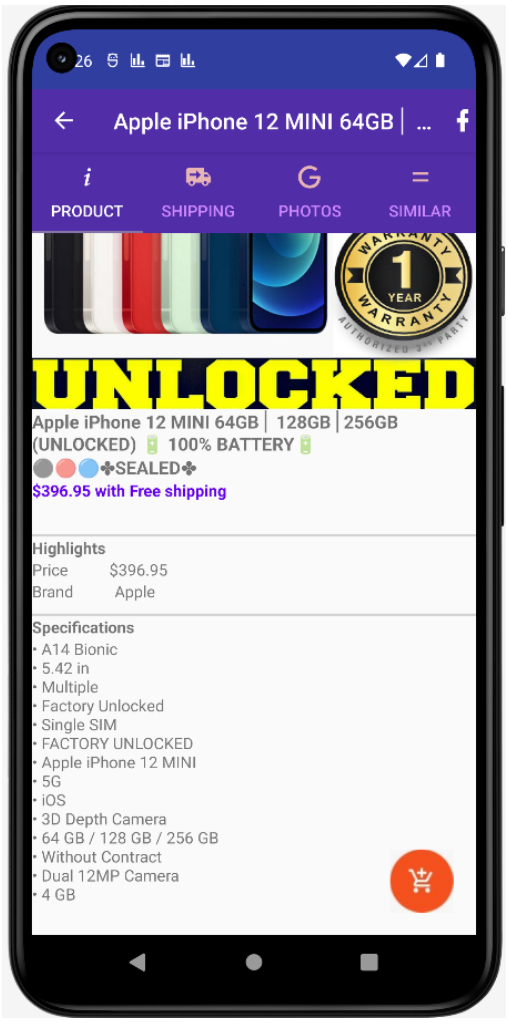

Figure 16: Specifications

Figure 15: Product Tab

5.4.2 Shipping Tab:

This tab contains the following sections each of which can be implemented using a TableLayout (see **Figure17** and **18**)

- **Sold By:** This section contains 4 fields
  - **Store Name:**
    - StoreName and StoreUrl properties in json.
    - Clicking on this should open a browser page redirecting to the store URL.
  - **Feedback score:**
    - FeedbackScore property in json
  - **Popularity**:
    - PositiveFeedbackPercent property in json.
    - This can be implemented using an external library called CircularScoreView See hints.
  - **Feedback star**:
    - FeedbackRatingStar property in json.
    - This section displays a small icon indicating the seller's feedback star value. The meaning of each of the values can be found here.
    - The stars have to be styled based on their name – the name indicates the color that the star will be displayed in, the icon can be taken from the icons table in section 6.1.
    - You can define multiple images/xmls for each color or you can set the colors dynamically (see hints)

- **Shipping info:** This section contains 4 fields
  - **Shipping Cost:**
    - Needs to be passed from the Search Result Activity to this. This can be done by intent. Display "Free shipping" if the cost is 0 else display $X
  - **Global Shipping:**
    - GlobalShipping property in json. Display Yes/No instead of true/false
  - **Handling Time:**
    - HandlingTime property in json.
    - If 0 or 1 display use "day" else use "days" as suffix
  - **Condition:**
    - ConditionDescription property in json

- **Return Policy:** This section contains 4 fields
    - **Policy:**
        - ReturnsAccepted property in json
    - **Returns within:**
        - ReturnsWithin property in Json
    - **Refund Mode:**
        - Refund property in Json
    - **Shipped by:**
        - ShippingCostPaidBy property in json

- **Notes:**
    - **JSON property names are written for reference. For exact structure please refer to Assignment 3.**
    - **Each of these sections are separated with a faint horizontal line.**
    - **If any of the details are missing, skip that row.**
    - **If for a particular section, <u>ALL the fields</u> are missing then skip the entire section.**
    - **If <u>ALL sections</u> are missing, then display an appropriate no results message.**

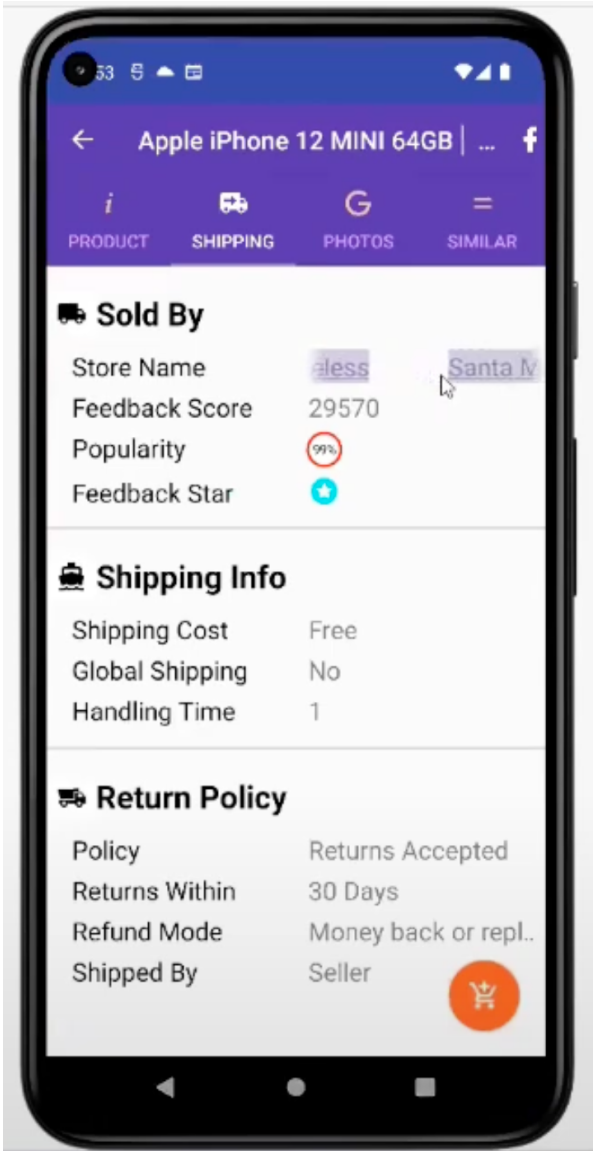

Figure 17: Shipping Tab

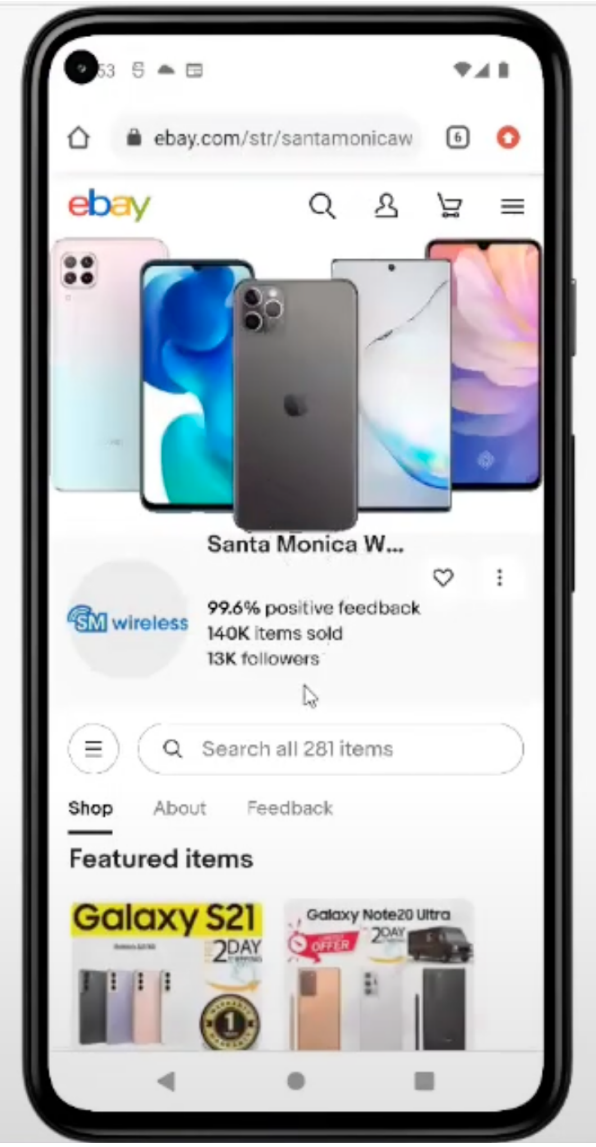

Figure 18: Navigate to store url

5.4.3 Google Tab

The content displayed is the same as in Assignment 3. Show 8 photos from *Google Custom Search API*. These photos are fetched using the title of the product received in the eBay response and **not** from the keyword that was entered by the user. See **Figure 19** and **Figure 20**.

You can use Volley Network ImageView, Picasso or Glide to load the image. The images should **not** be cropped and must fit and be centered on the designated ImageView.

See the hints and some useful links in section 6.3.5.

Figure 19: Google Photos

Figure 20: Google Photos

5.4.4 Similar Products Tab

This tab displays the similar products received from eBay, same as Assignment 3.

The Similar Products are shown in a list using a RecyclerView. Note that each of the cells can be tapped and then a web page should be opened and navigated to the eBay page, see video for more detail and **Figure 21**.

- **Product image:**
  - imageUrl property in json
- **Title:**
  - title property in json
- **Shipping**:
  - shipping costs value property in json
  - (display Free Shipping if the cost is 0)
- **Days Left**:
  - timeLeft property in json.
  - The days can be extracted by taking the characters between P and D in the specified field. See hints for some helpful String methods in Java.
- **Price**:
  - buyItNowPrice's value property in json.

As shown in **Figure 21** and **22**, you should use two Spinners to switch sort parameters (Name/Price/Days/Default) and also its order (Ascending/Descending). When "Default" is selected, the Ascending or Descending spinner should be disabled, and the original order should be restored.

There are many ways to handle sorting depending on how you store your data. The recommended way is to create a Java Class to store all the fields of each row and use a comparator/comparable to sort the list and notify the adapter (see hints for examples)

- **Notes:**
  - **JSON property names are written for reference. For exact structure please refer to Assignment 3**
  - **If any of the details are missing, you can either skip that row or show "N/A"**
  - **If there are no similar products – display "No results" and both spinners should be disabled**

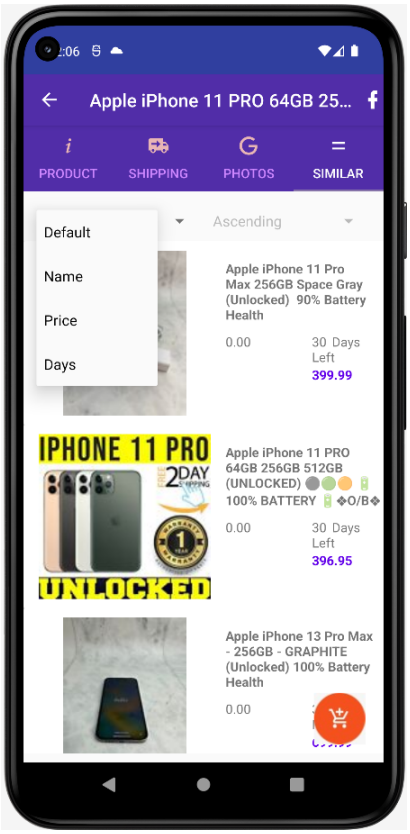

Figure 21: Sort category spinner

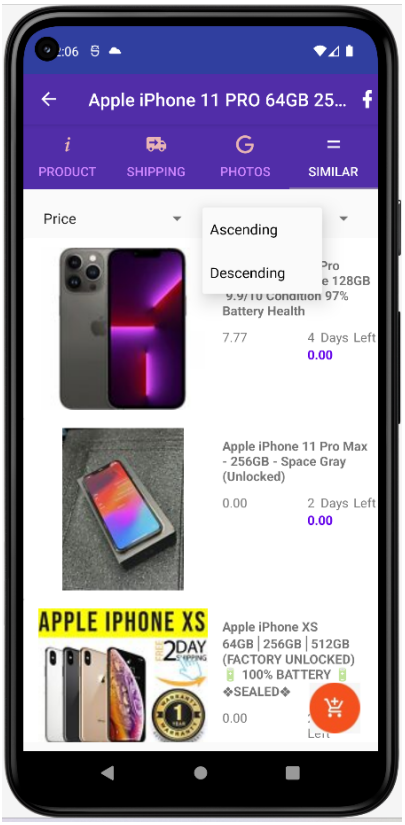

Figure 22: Sort Order Spinner

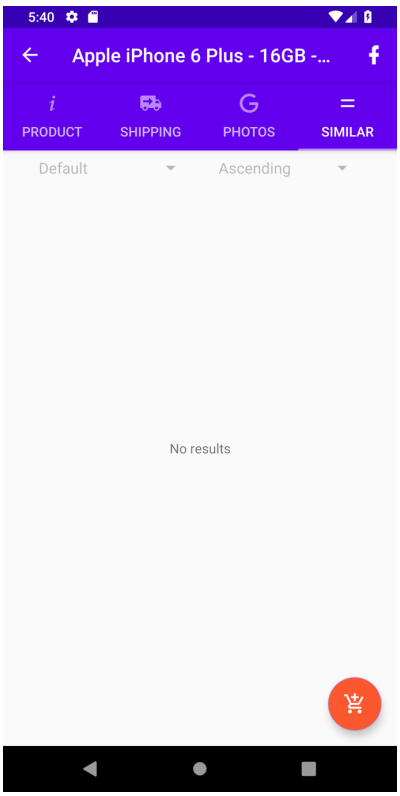

Figure 24: No Records

5.5 WishList

Use Tabs with a ViewPager on the main screen to switch between the search page and the wishlist page. The wishlisted products should be displayed in a list using a RecyclerView. You can use the same view that you used in the search results. See **Figure 25**.

If there are no wishlisted products, "No items in Wishlist" should be displayed at the center of the screen, as shown in **Figure 27**.

Like in search results, pressing the cart icon here should remove the product from the wishlist. See video for more detail.

Like in Assignment 3, the product info of wishlist should be persisted in Mongodb on the cloud platform (GCP/AWS/Azure).

The bottom of this page displays the summary of the wishlist. This view should contain 2 things:

- “Wishlist total (X items)” with X being the number of items in the wishlist
- The sum of prices of all products should be displayed on the right end
- Adding/removing items should update this view dynamically.
- You can achieve this by updating the text every time you call this component’s adapter’s notifyAdapterDataSetChanged method.

See **Figure 25** and **Figure 26**.

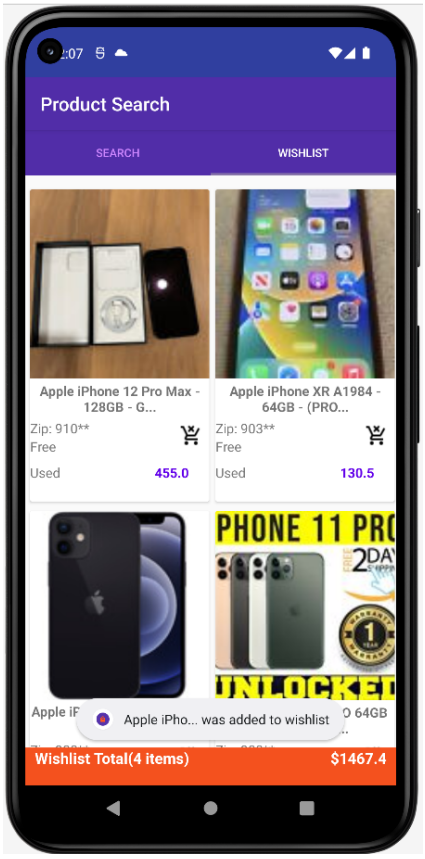

Figure 25: Wishlist page

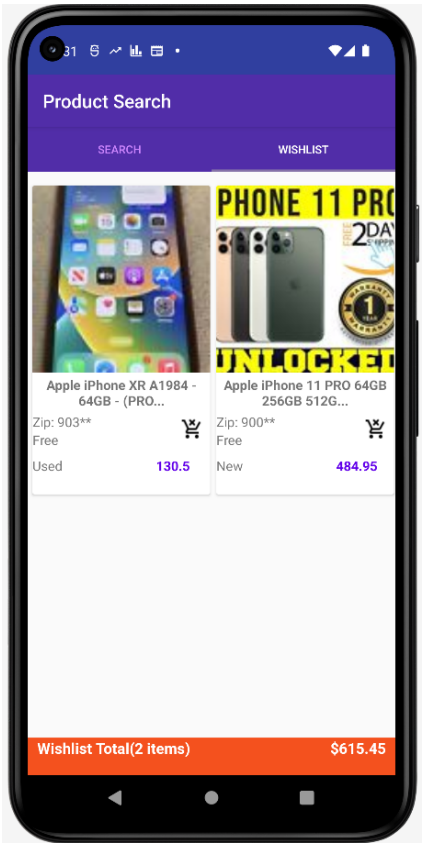

Figure 26: Wishlist with changed totals

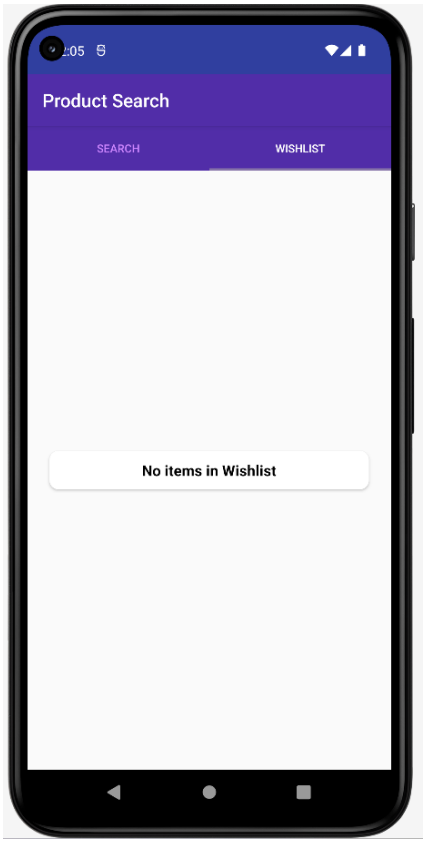

Figure 27: No Wishes

5.6 Summary of detailing and error handling

1. If no products are found given a keyword, a “No Results” message should be displayed. Similarly, appropriate messages should be displayed for no similar products, and for the empty wishlist. See **Figure 27** for an example.
2. If for any reason an error occurs (no network, API failure, cannot get location, etc.), an appropriate error message should be displayed at the bottom of the screen using a Toast.
3. For any cards (search results and wishlist)mj if any field is missing, then display N/A.
4. For any missing data on detailed tabs pages, skip that row. If all rows are missing in a section, skip that entire section and if all sections are missing, then display a message similar to 1 above.
5. The name title of the product should span EXACTLY 3 lines in the list page.
6. Everywhere the shipping cost should be displayed as “Free Shipping” if the cost is 0.
7. Colors and icons for wishlist should be dynamic.
8. Brand should be the first value under Specifications section.
9. Show the number of search results on top.
10. Display 8 images in the Google tab.
11. No photos should be cropped anywhere, center the image inside the ImageView.
12. Feedback star’s color and icon should be dynamic.
13. Sorting for price and days should work on the numeric values and not String comparisons.
14. Facebook posts must include Hashtag.
15. Total cost and number of items in the wishlist should update immediately.
16. The keyboard should not hide the Zip Code textview while typing.
17. Distance should only be considered if the location checkbox is enabled.
18. Clear should clear all errors and form values.

5.7 Additional Info

For things not specified in the document, grading guideline, or the video, you can make your own decisions. But keep in mind about the following points:

- Always display a proper message and don’t crash if an error happens.
- Always display a loading message if the data is loading.
- You can only make HTTP requests to your backend Node.js on AWS/GAE/Azure.
- All HTTP requests should be asynchronous and should not block the main UI thread. You can use third party libraries like Volley to achieve this in a simple manner.

6. Implementation Hints

6.1 Icons

The images used in this Assignment are taken from https://materialdesignicons.com/ .

Here are the names and details of each page. You can choose to work with xml/png/jpg versions. We recommend using xml as it is easy to modify colors by setting the Fill Colors.

Icon Name	**Usage**
Shopping	App Icon and welcome screen
information-variant	Product info tab icon
truck-delivery	Shipping tab icon
Google	Google photos tab icon
equal	Similar products tab icon
facebook	for Facebook share button
Wrench	for specifications section in product details
information	for highlights section in product details
cart-plus	Add to wishlist
cart-remove	Remove from wishlist
star-circle	for shooting seller feedback in Shipping tab
star-circle-outline	for normal seller feedback in Shipping tab
truck	for Sold By section in Shipping tab
dump-truck	for Returns Policy section in Shipping tab
ferry	for Shipping info section in Shipping tab

6.2 Getting current location

In order to get the current location, you can use either ip-api or location services.
For your location fetching code to work, you must request permission from the user. You can read more about requesting permissions here:
https://developer.android.com/training/permissions/requesting.html

You may need to mock the location in your emulator. This can be done from the emulator settings.

6.3 Third party libraries

Almost all functionality of the app can be implemented without using third party libraries, but sometimes using them can make your implementation much easier and quicker. Some libraries you may want to use are listed in the following sections.

6.3.1 Google Play services

You will need this for various features like getting the current location and using Google Maps in your app.

You can learn about setting it up here:
https://developers.google.com/android/guides/setup

6.3.2 Volley HTTP requests

Volley can be helpful with asynchronous http requests to load data. You can also use Volley network ImageView to load photos in the Google tab. You can learn more about them here:

https://developer.android.com/training/volley/index.html

6.3.3 Picasso

Picasso is a powerful image downloading and caching library for Android.
http://square.github.io/picasso/

If you decide to use RecycleView to display the photos with Picasso Please, use version 2.5.2 since the latest version does not support RecycleView well.
https://github.com/codepath/android_guides/wiki/Displaying-Images-with-the-Picasso-Library

6.3.4 Glide

Glide is also a powerful image downloading and caching library for Android. It is similar to Picasso. You can also use Glide to load photos in the Google tab.
https://bumptech.github.io/glide/

6.3.5 CircularScoreView

This is a third-party library that can be used to display the feedback rating of a seller. This can be found here https://github.com/wssholmes/CircularScore. Add this library to your Gradle and you can directly use it in the resource files. Set the appropriate value and the color.

6.4 Implementing a Gallery view

A simple way of implementing a gallery view is to use a HorizontalScrollView which basically contains the collection of image views of fixed size. A sample can be found here:
https://questdot.com/android-image-gallery-horizontalscrollview-tutorial/

6.5 Implementing Sorting techniques

https://stackoverflow.com/questions/45790363/how-to-sort-recyclerview-item-in-android
Do not forget to call NotifyDataSetChanged() method wherever needed.

6.6 Working with the AutoCompleteTextView

Working with the AutoCompleteTextView to show the suggestions might be a little challenging. This tutorial goes over how it is done so that you get an idea of how to go about it.
https://www.truiton.com/2018/06/android-autocompletetextview-suggestions-from-webservice-call/

6.7 Implementing a Splash Screen

There are many ways to implement a splash screen. This blog highlights almost all of them with examples:
https://android.jlelse.eu/the-complete-android-splash-screen-guide-c7db82bce565

6.8 Dynamic coloring using DrawableCompat

https://medium.com/@hanru.yeh/tips-for-drawablecompat-settint-under-api-21-1e62a32fc033

6.9 String manipulation in Java

https://www.guru99.com/java-strings.html

6.10 User location using Emulator

If you choose to use location services to get the current location, then you can use the following emulator settings.

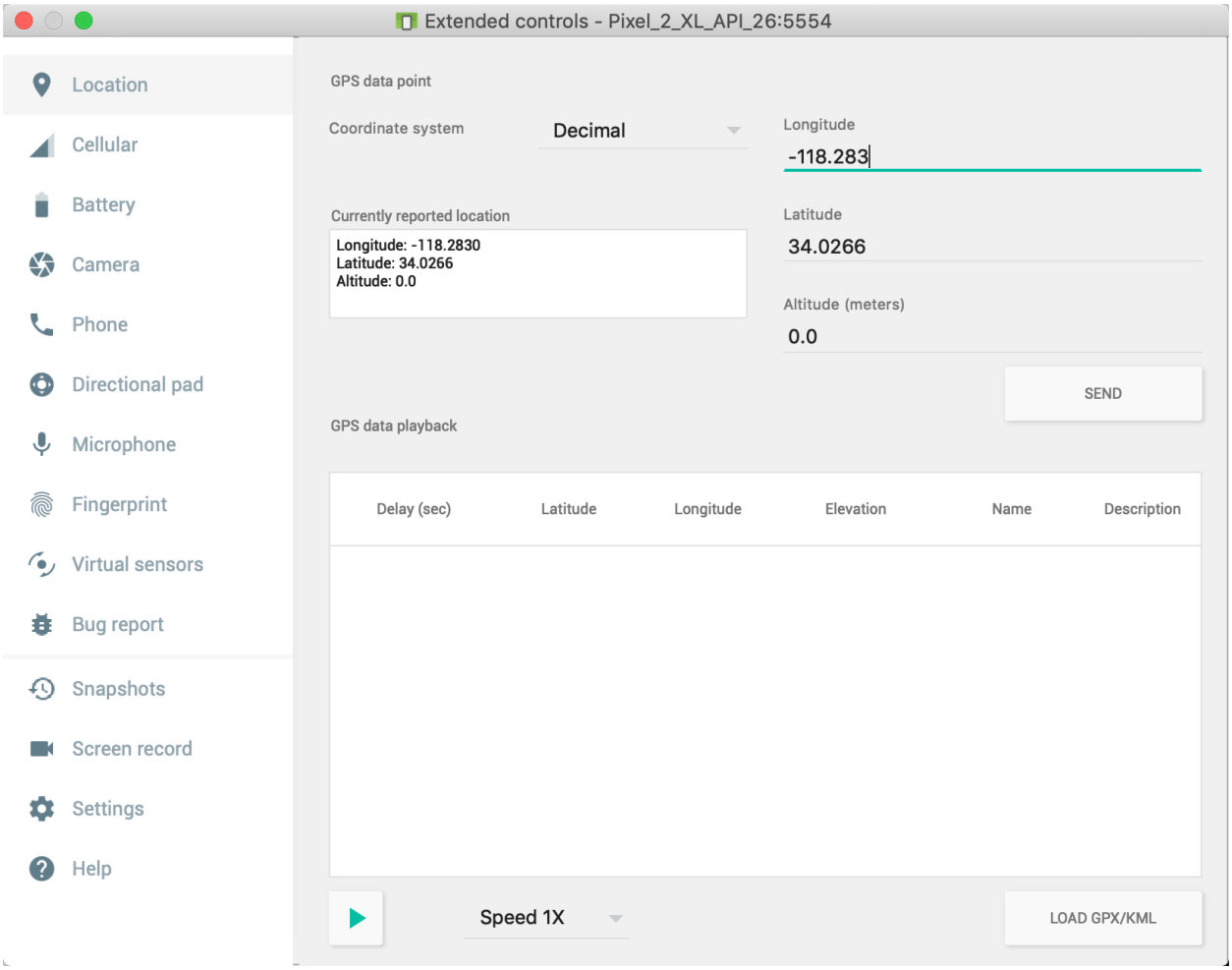

Figure 28: Location Setting of Emulator

6.11 MongoDB with Android

For wishlist items, all information should be persisted in Mongodb. Please see section 5.6 from Assignment 3 for more details.

7. What to Upload to D2L

You should ZIP all your source code top level folder (the java/ and res/ directories) and submit the resulting ZIP file by the end of the demo day.

Unlike other exercises, you will have to demo your submission **via zoom recording**. You will upload the Zoom recording MP4 in addition to the source code ZIP.

.

Details and logistics for the demo will be provided on D2L. **The demostration is done an a laptop/ notebook/MacBook or Windows PC using the emulator, and not a physical mobile device.**

****IMPORTANT****

All videos are part of the Assignment description. All discussions and explanations on Piazza related to this Assignment are part of the Assignment description and will be accounted into grading. So please review all Piazza threads before finishing the assignment.