ICS4U

Gordon Roller

Jan 23, 2017

Mohamed Amadou

Samuel Jones

Find It! Painless Shopping

Software Design Document

# Table of Contents

1.0 Client

1.1 Client Description

1.2 Client Requirements

1.3 Software Platforms and APIs

2.0 User’s Guide

2.1 Search for Items

2.2 Use the Item Cart

2.3 Change your Language

2.4 Change your Display Theme

3.0 UML Diagrams

4.0 Detailed Explanation of Functional Code Blocks

4.1 Website Search Classes

4.2 Handler Classes

4.3 Adapters

4.4 Item Class

4.5 Language Code

4.6 Activity Code

Appendix A - Code

Appendix B - Beta Testing and Reporting

Appendix C - Reflection of Learning

# 

# 1.0 Client

## 1.1 Client Description

Sophie is a student here at Earl. Sophie’s into Art, Books and Fashion. She enjoys shopping but often finds herself wasting time going from store to store searching for the item that she wants. On average, she’ll visit the centrum shopping centre twice a week, which, over time, adds up.

## 1.2 Client Requirements

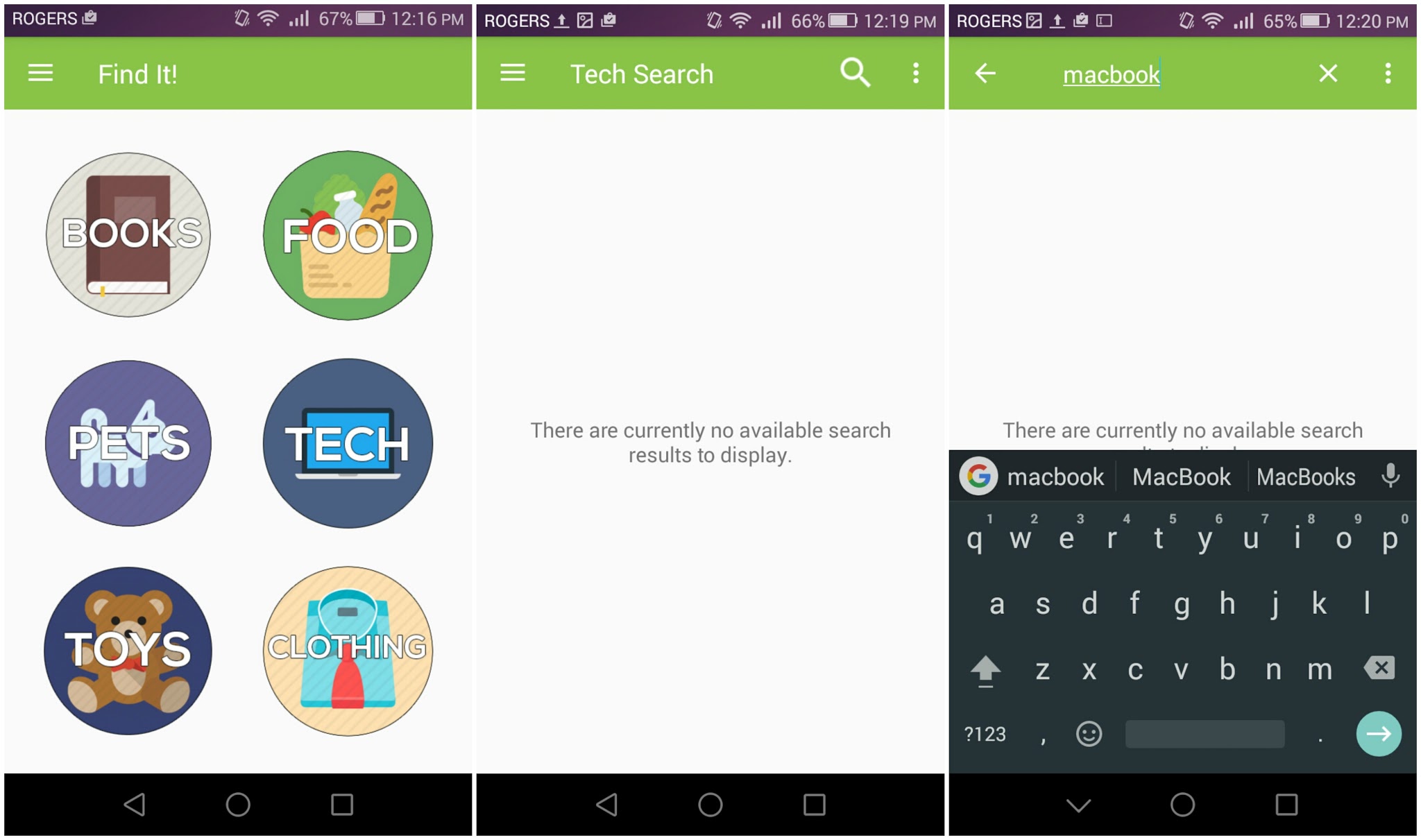
Knowing Sophie personally, she was the first person that came to mind when thinking of a client. Her requirements for the app were: android compatibility, working for ottawa, and analyzing all of the basic details of queried items. These were all solid criteria and we were very lucky to have been given clear responsibilities by our client. With these needs in mind we created a comprehensive list of the core functionalities of the app.

## 1.3 Software Platforms and APIs

* Android Studio
  + It is an IDE or integrated development environment that makes use of a lightweight, sleek UI to streamline the development experience. To top it off, it also has Git built in, which made updating and maintaining the app easier than any developer could wish for. We used Android Studio to code, test, and design our user interface as well as maintain and update our code using the built in git client.
* Android SDK
  + The Android Software Development Kit provided us with all of the tools necessary for working with Android devices in all of their versions. The difference between Android Studio and Android SDK lies in that Android Studio is an IDE, meaning we do our coding on it, whereas Android SDK contains a library that allows Android Studio to read and interpret our code.
* Adobe Photoshop CC and Illustrator CC
  + Photoshop is a part of the Adobe Creative Suite and is used primarily for image editing although it can be used for digital art as well as logo design. Along with Adobe Illustrator, Photoshop was my goto whenever we needed visuals done. The search icons were done in photoshop and the app logo was retouched in photoshop.
  + Illustrator revolves around vectors and works with svg’s or scalable vector graphics. The benefit to this is that designs created in illustrator can be scaled to very large degrees and retain 100% of their quality. However the downside to this is that because it relies on shapes and paths to produce its images, it is difficult to make particularly intricate designs. Because of this, Illustrator’s most effective use lies in logos and simple designs.
  + Now to anyone wishing to work with either Illustrator or Photoshop in the future, it is important to note that most graphic designers use both in conjunction with each other. You’ll have the people who swear by illustrator and the ones that swear by photoshop, and there’s nothing wrong with that per say, but each of them have their own uses and well-rounded design tends to come from a combination of the two.
* Jsoup
  + Jsoup is a Java Library that allows users to read and manipulate the HTML content of web pages. This allowed us to gather information from web searches of specific websites. No 2 websites were the same however, so we had to create a unique method of searching the site for each of them. Jsoup is great because it’s very version friendly. It doesn’t require a lot of resources and can run on very low Android versions.
* GitHub
  + GitHub is a Git based hosting service that we used to host our code and use for version control. It works on a system of branches, commits, merges, etc. which makes it easy for multiple people to work on the same project without overwriting each other’s code.

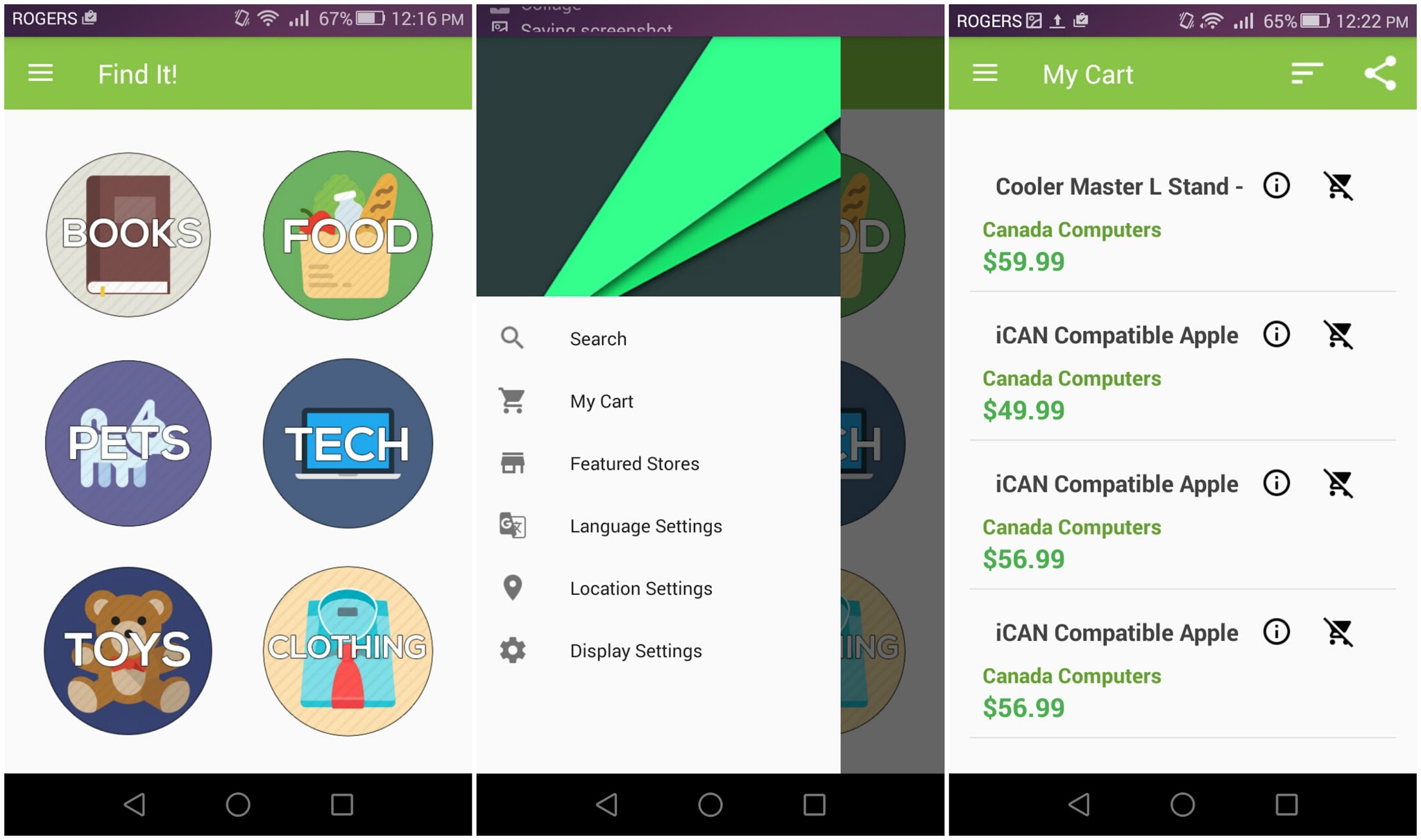
# 2.0 User’s Guide

## 2.1 Search for Items



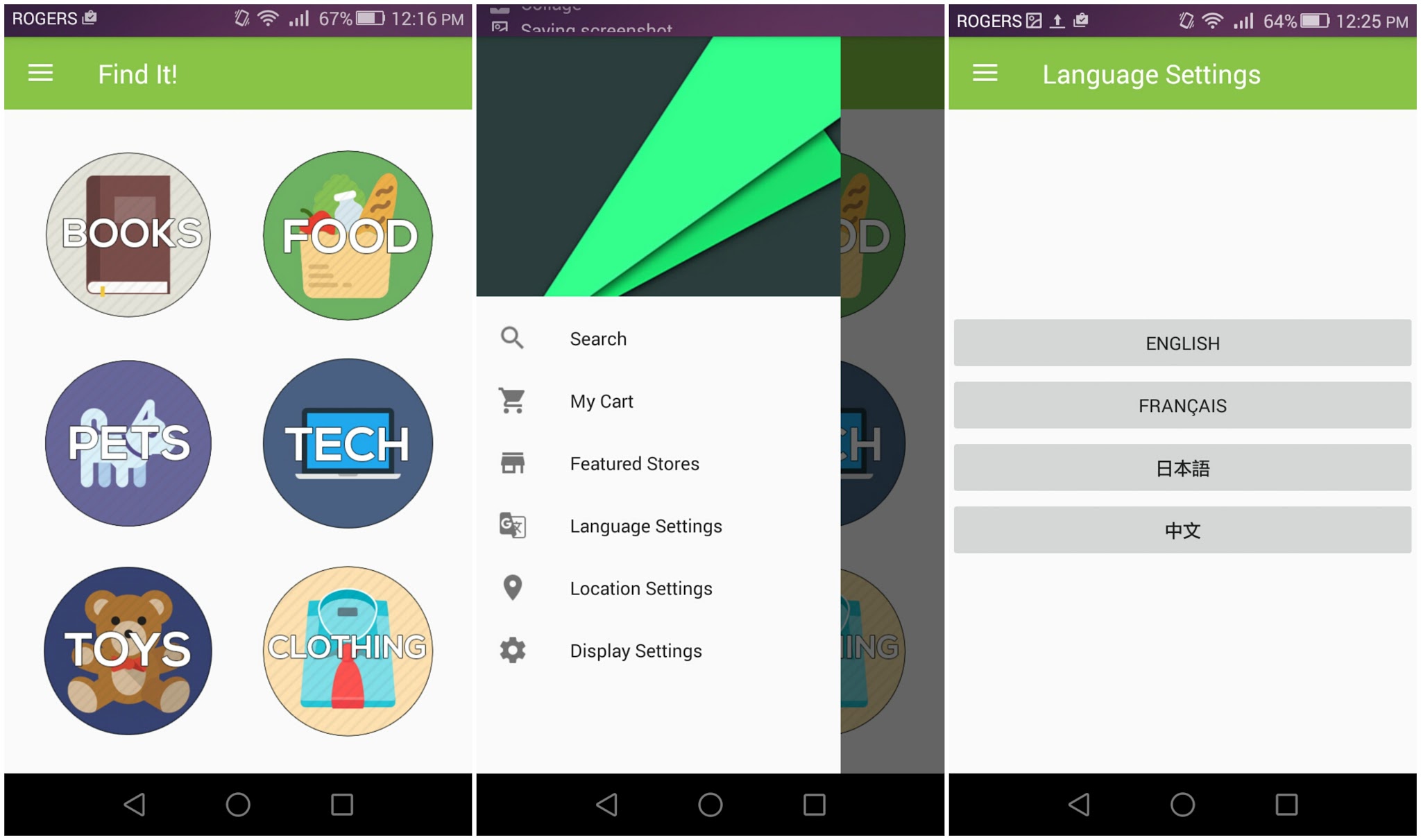
1. When on the “Search” screen click the icon corresponding to the search type you would like to make.
2. Once the corresponding activity open click the magnifying class to open up text field to be able to type in your query.
3. Once the enter button is clicked your search will be carried out and the items will appear on screen.
4. Once the search has been completed you may tap the three dots in the corner to sort the results. Other options include pressing the i button to open the link to the item or the cart to add it your cart.

## 2.2 Use the Item Cart



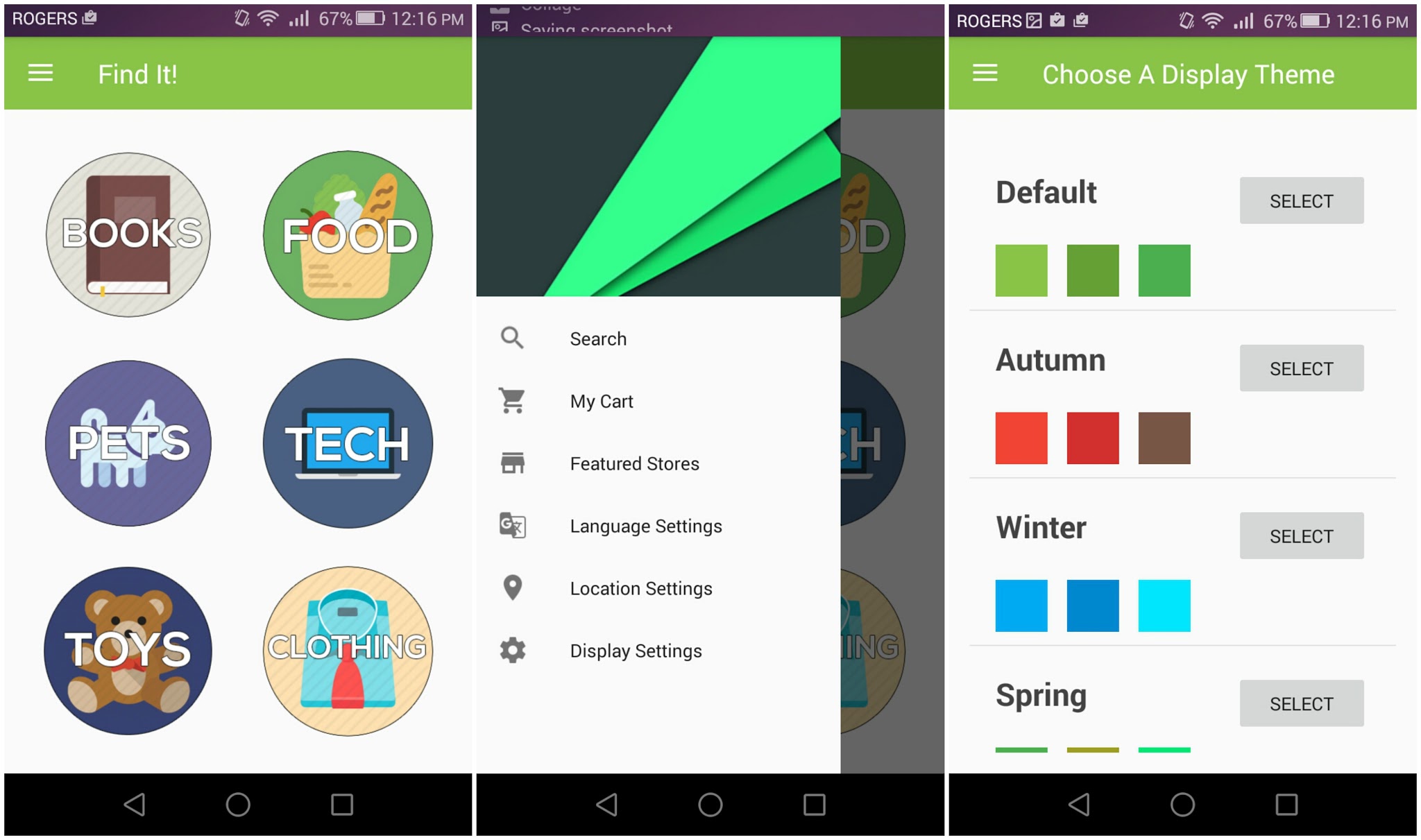
1. When on the “My Cart” screen you will view the items that you have saved.
2. Once the search has been completed you may tap the three dots in the corner to sort the results. Other options include pressing the i button to open the link to the item or the cart to add it your cart. The share button in the corner will also allow you to export your cart and send it to friends.

## 2.3 Change your Language



1. When on any screen click the drawer icon in the corner to open the screen drawer. From there click “Language Settings” to open that activity.
2. Once the activity has been opened, click the language for which you would like to see Find It! display.

## 2.4 Change your Display Theme



1. When on any screen click the drawer icon in the corner to open the screen drawer. From there click “Display Settings” to open that activity.
2. Once the activity has been opened, click the select button for the colors for which you would like to see Find It! display.

# 3.0 UML Diagrams

|  |
| --- |
| BestBuySearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| BookSearch |
| * context : Context * dialog : ProgressDialog * Items : ArrayList<Item> * listView : ListView * adapter : CustomAdapter * text : TextView * doc : Document |
| * onCreate(Bundle savedInstanceState) : void * onNewIntent(Intent intent) : void * handleIntent(Intent intent) : void * onCreateOptionsMenu(Menu menu) : boolean * onOptionsItemSelected(MenuItem item) : boolean * sortList(String type) : void * onBackPressed() : void * onNavigationITemSelected(MenuItem item) : boolean |

|  |
| --- |
| CanadaComputersSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| CartAdapter |
| * dataSet : ArrayList<Item> * mContext : Context * listView : View |
| * CartAdapter (ArrayList<Item> data, Context context, View lv) * onClick(View v) : void * getView(int position, View convertView, ViewGroup parent) : View |

|  |
| --- |
| CartInfoProvider |
| * cart : ArrayList<Item> |
| * importCart(ArrayList<Item> i) : void * addToCart(Item i) : void * removeFromCart(Item i) : void * getCart() : ArrayList<Item> |

|  |
| --- |
| ChaptersIndigoSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| ClothingSearch |
| * context : Context * dialog : ProgressDialog * Items : ArrayList<Item> * listView : ListView * adapter : CustomAdapter * text : TextView * doc : Document |
| * onCreate(Bundle savedInstanceState) : void * onNewIntent(Intent intent) : void * handleIntent(Intent intent) : void * onCreateOptionsMenu(Menu menu) : boolean * onOptionsItemSelected(MenuItem item) : boolean * sortList(String type) : void * onBackPressed() : void * onNavigationITemSelected(MenuItem item) : boolean |

|  |
| --- |
| CustomAdapter |
| * dataSet : ArrayList<Item> * mContext : Context * listView : View |
| * CartAdapter (ArrayList<Item> data, Context context, View lv) * onClick(View v) : void * getView(int position, View convertView, ViewGroup parent) : View |

|  |
| --- |
| Display |
| * window : Window * mViewPager : ViewPager |
| * onCreate(Bundle savedInstanceState) : void * onBackPressed() : void * onCreateOptionsMenu(Menu menu) : boolean * onOptionsItemSelected(MenuItem item) : boolean * onNavigationItemSelected(MenuItem item) : boolean |

|  |
| --- |
| EBGamesSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| FeaturedScreen |
|  |
| * onCreate(Bundle savedInstanceState) : void * onBackPressed() : void * onNavigationItemSelected(MenuItem item) : boolean |

|  |
| --- |
| FeaturedStoresAdapter |
| * mContext : Context * mThumbIds : Integer[] * mLinks : String[] |
| * FeaturedStoresAdapter(Context c) * getCount() : int * getItem(int position) : Object * getItemId(int position) : long * getView(final int position, View convertView, ViewGroup parent) : View |

|  |
| --- |
| FoodSearch |
| * context : Context * dialog : ProgressDialog * Items : ArrayList<Item> * listView : ListView * adapter : CustomAdapter * text : TextView * doc : Document |
| * onCreate(Bundle savedInstanceState) : void * onNewIntent(Intent intent) : void * handleIntent(Intent intent) : void * onCreateOptionsMenu(Menu menu) : boolean * onOptionsItemSelected(MenuItem item) : boolean * sortList(String type) : void * onBackPressed() : void * onNavigationITemSelected(MenuItem item) : boolean |

|  |
| --- |
| HnMSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| HomeScreen |
|  |
| * onCreate(Bundle savedInstanceState) : void * onCreateOptionsMenu(Menu menu) : boolean * onOptionsItemSelected(MenuItem item) : boolean * onBackPressed() : void * onNavigationITemSelected(MenuItem item) : boolean |

|  |
| --- |
| ImageAdapter |
| * mContext : Context * mThumbIds : Integer[] |
| * ImageAdapter(Context c) * getCount() : int * getItem(int position) : Object * getItemId(int position) : long * getView(int position, View convertView, ViewGroup parent) : View * onClick(View v) : void |

|  |
| --- |
| Item |
| * title, store, link : String * price : Double |
| * Item (String t, String s, Double p, String l) * setTitle(String t) : void * setStore(String s) : void * setPrice(Double p) : void * getTitle() : String * getPrice() : Double * getStore() : String * toString() : String * sortItems(ArrayList<Item> unsorted, String sort) : ArrayList<Item> |

|  |
| --- |
| LangSettings |
| * appCurrentLanguage : String |
| * onCreate(Bundle savedInstanceState) : void * onCreateOptionsMenu(Menu menu) : boolean * onOptionsItemSelected(MenuItem item) : boolean * onBackPressed() : void * onNavigationITemSelected(MenuItem item) : boolean |

|  |
| --- |
| LanguageHandler |
| * defaultLang : String * lang : String |
| * getLang() : String * setLang(String l) : void |

|  |
| --- |
| LoblawsSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| MarksSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| MastermindToysSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| MyCart |
| * Items : ArrayList<Item> * listView : ListView * adapter : CartAdapter |
| * onCreate(Bundle savedInstanceState) : void * onCreateOptionsMenu(Menu menu) : boolean * shareIt() : void * onOptionsItemSelected(MenuItem item) : boolean * sortList(String type) : void * onBackPressed() : void * onNavigationItemSelected(MenuItem item) : boolean |

|  |
| --- |
| OpeningScreen |
| * firstLaunch * KEY\_PREFS\_NAME, KEY\_FIRST\_LAUNCH : String |
| * onCreate(Bundle savedInstanceState) : void |

|  |
| --- |
| PetSearch |
| * context : Context * dialog : ProgressDialog * Items : ArrayList<Item> * listView : ListView * adapter : CustomAdapter * text : TextView * doc : Document |
| * onCreate(Bundle savedInstanceState) : void * onNewIntent(Intent intent) : void * handleIntent(Intent intent) : void * onCreateOptionsMenu(Menu menu) : boolean * onOptionsItemSelected(MenuItem item) : boolean * sortList(String type) : void * onBackPressed() : void * onNavigationITemSelected(MenuItem item) : boolean |

|  |
| --- |
| PetSmartSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| RCSuperstoreSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| RootsSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| Search |
| * context : Context * dialog : ProgressDialog * Items : ArrayList<Item> * listView : ListView * adapter : CustomAdapter * text : TextView * doc : Document |
| * onCreate(Bundle savedInstanceState) : void * onNewIntent(Intent intent) : void * handleIntent(Intent intent) : void * onCreateOptionsMenu(Menu menu) : boolean * onOptionsItemSelected(MenuItem item) : boolean * sortList(String type) : void * onBackPressed() : void * onNavigationITemSelected(MenuItem item) : boolean |

|  |
| --- |
| SearchQuery |
| * doc : Document * results : Elements * description : String * price : double * status : int * dialog : ProgressDialog * BOOK\_SEARCH, FOOD\_SEARCH, CLOTHING\_SEARCH, PETS\_SEARCH, TECH\_SEARCH, TOYS\_SEARCH : int |
| * SearchQuery() * SearchQuery(Context c, String q, int type) |

|  |
| --- |
| SearchQueueHandler |
| * BOOK\_QUEUE, CLOTHING QUEUE, FOOD\_QUEUE, PETS\_QUEUE, TECH\_QUEUE, TOYS\_QUEUE : int * BOOK\_SEARCH, FOOD\_SEARCH, CLOTHING\_SEARCH, PETS\_SEARCH, TECH\_SEARCH, TOYS\_SEARCH : int |
| * makeRequest(Context c, ArrayList<Item> processed, int type) : void |

|  |
| --- |
| SportChekSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| StaplesSearch |
| * resultsEven : Elements * finalDoc : Elements * processed : Elements * uiHandler : Handler * status : int * c : Context |
| * BestBuySearch(Context context, String query) * crunchResults(Elements e) : ArrayList<Item> * getStatus() : int |

|  |
| --- |
| TechSearch |
| * context : Context * dialog : ProgressDialog * Items : ArrayList<Item> * listView : ListView * adapter : CustomAdapter * text : TextView * doc : Document |
| * onCreate(Bundle savedInstanceState) : void * onNewIntent(Intent intent) : void * handleIntent(Intent intent) : void * onCreateOptionsMenu(Menu menu) : boolean * onOptionsItemSelected(MenuItem item) : boolean * sortList(String type) : void * onBackPressed() : void * onNavigationITemSelected(MenuItem item) : boolean |

|  |
| --- |
| ThemeAdapter |
| * dataSet : ArrayList<Item> * mContext : Context * listView : View * colorPrimary, colorPrimaryDark, colorAccent : int * themeIDs : Integer[] |
| * ThemeAdapter(Context context, View lv) * getCount() : int * getItem(int i): Object * getItemId(int i) : long * getView(int position, View convertView, ViewGroup parent) : View |

|  |
| --- |
| ThemeHandler |
| * THEME\_DEFAULT : int * theme : int |
| * setTheme(int themeID) : void * getTheme() : int |

|  |
| --- |
| ToySearch |
| * context : Context * dialog : ProgressDialog * Items : ArrayList<Item> * listView : ListView * adapter : CustomAdapter * text : TextView * doc : Document |
| * onCreate(Bundle savedInstanceState) : void * onNewIntent(Intent intent) : void * handleIntent(Intent intent) : void * onCreateOptionsMenu(Menu menu) : boolean * onOptionsItemSelected(MenuItem item) : boolean * sortList(String type) : void * onBackPressed() : void * onNavigationITemSelected(MenuItem item) : boolean |

# 4.0 Detailed Explanation of Functional Code Blocks

Our code is very well-structured in terms of its modularity. The Website Search classes are all tied together by a SearchQuery class controlled by a Handler class that executes them all together. Every dynamic list (A list that is loaded depending on a user action, i.e. a sorted list or a generated search results page) All app preferences are loaded up using File IO which passes information to the Handler classes that effectively save the preferences. All activity classes are intertwined through a navigation drawer class that allows users to travel between them. As you can see, our code is very modular and allows for the maximum amount of adaptability and additions.

## 4.1 Website Search Classes

The website search classes will be explained using the search code for **Best Buy**. The search class for Best Buy can be found at the following link:

|  |
| --- |
| https://github.com/FindItPS/Find-It-Android/blob/master/src/main/java/com/painlessshopping/mohamed/findit/BestBuySearch.java |

Functional Code Blocks

|  |
| --- |
| public BestBuySearch(Context context, String query) |

This is the constructor method for the search class. It takes in two parameters(A context, and a search query in the form of a String) and executes the search. An Android WebView with an embedded Javascript-HTML Interface is used to allow the search code to support JavaScript-generated webpages.

|  |
| --- |
| protected class JSHtmlInterface |

This nested class allows the Android WebView class to interact with JavaScript (where it would normally only read HTML). It is activated by using the setJavaScriptEnabled() method on the WebView and then attaching the JSHTMLInterface by using the addJavaScriptInterface() method on the WebView.

|  |
| --- |
| class fetcher extends AsyncTask<String, Void, Elements> |

AsyncTasks are an Android class that allows processes to run in the background without stalling the user interface. An AsyncTask consists of 4 methods; a constructor, a preExecute, a doInBackground, and an onPostExecute which is typically used to communicate with the ui thread.

|  |
| --- |
| public fetcher(Context context) |

The constructor for our AsyncTask really only serves the purpose of obtaining the context from the search activity. In order to post a progress dialog to the user interface, the AsyncTask needs a context, or window if you will, to attach itself to.

|  |
| --- |
| protected void onPreExecute() |

Our pre-execute starts a progress dialog before the task begins so that the user knows that the app is working.

|  |
| --- |
| protected Elements doInBackground(String... strings) |

The doInBackground is the main component of the AsyncTask. In our doInBackground, we load in the website, and search for and return all elements containing search results. The parameter array ‘strings’ is the link for the website.

|  |
| --- |
| protected void onPostExecute(Elements result) |

As mentioned previously, this is where the AsyncTasks communicates with the thread on which the user interface is again. Our onPostExecute calls the crunchResults method which parses the returned elements, then adds all of the results to the dynamic list being displayed on the screen and refreshes the list so that they show up. Finally, it dismisses the progress dialog so that the user knows that data collection is complete and sends a request to our custom-written SearchQueueHandler.

|  |
| --- |
| public ArrayList<Item> crunchResults(Elements e) |

This method reads in the raw results in the form of Jsoup Elements objects. It extracts all relevant details about the objects and stores them in Item objects. Once complete, it returns the formatted results in the form of an Item ArrayList.

## 4.2 Handler Classes

The handler classes will be explained using the code for the **SearchQueueHandler**. The code for SearchQueueHandler can be found at the following link:

|  |
| --- |
| https://github.com/FindItPS/Find-It-Android/blob/master/src/main/java/com/painlessshopping/mohamed/findit/SearchQueueHandler.java |

A Word on Static Variables

|  |
| --- |
| public static int BOOK\_QUEUE = 4 , CLOTHING\_QUEUE = 4, FOOD\_QUEUE = 2, ...; |

The queues are made into public static variables so that they can be accessed by any class without having to instantiate a SearchQueueHandler object. They represent the total amount of searches to be done in a certain category before displaying a ‘search complete’ message.

|  |
| --- |
| public static final int BOOK\_SEARCH = 1, FOOD\_SEARCH = 2, PETS\_SEARCH = 3, ...; |

These numbers are set as public static final variables so that they cannot be changed dynamically. They simply serve as identifiers for each search type so that the SearchQueueHandler knows which queue to subtract from.

|  |
| --- |
| public static ArrayList<Item> masterList = new ArrayList<Item>(); |

This contains the master list of all Items compiled from the individual website search classes.

Functional Code Blocks

|  |
| --- |
| public static void makeRequest(Context c, ArrayList<Item> processed, int type) |

This is the bulk of the code for our SearchQueueHandler. It reads in a context to display the ‘search complete’ message to (The search screen) as well as the processed results from a single search and an identifier for the type of search. (BOOK\_SEARCH, FOOD\_SEARCH, etc..) It then adds these processed results to the master list and decreases the queue for the designated type of search. If the search queue has now reached 0, it will display a search complete message including the amount of relevant results that were retrieved.

## 4.3 Adapters

The logic behind the adapter classes will be explained using the code for **CartAdapter**. The code for CartAdapter can be found at the following link:

|  |
| --- |
| https://github.com/FindItPS/Find-It-Android/blob/master/src/main/java/com/painlessshopping/mohamed/findit/CartAdapter.java |

|  |
| --- |
| private static class ViewHolder |

This nested class allows us to effectively create a container that holds all of the components of the cart item layout. It contains TextViews and ImageViews that are titled appropriately to show their purpose.

|  |
| --- |
| public CartAdapter (ArrayList<Item> data, Context context, View lv) |

Much like the constructor method for our AsyncTasks, this constructor simply reads in parameters and stores them in local variables so that the rest of the class can interact with them.

|  |
| --- |
| public void onClick(View v) |

This onClick method is the listener for the more info button. When it is clicked it opens a link in the web browser that goes directly to the item’s info page.

|  |
| --- |
| public View getView(int position, View convertView, ViewGroup parent) |

This is an override method from the ArrayAdapter. It essentially reads the cart item in the position position and allows it to be edited. This method is called when the List is instantiated.

|  |  |
| --- | --- |
|  | public void onClick(View v) |

This onClick is the first onClick nested within the removeFromCart button. It removes an item from cart and creates a snackbar message that allows users to undo the last action.

|  |  |  |
| --- | --- | --- |
|  |  | public void onClick(View v) |

This onClick is the onClick for the snackbar message’s ‘UNDO’ button. When it is clicked it re-adds the item to the cart and dismisses the snackbar message.

## 4.4 Item Class

The code for the Item class can be found at the following link:

|  |
| --- |
| https://github.com/FindItPS/Find-It-Android/blob/master/src/main/java/com/painlessshopping/mohamed/findit/Item.java |

Note: Most of the code in this class is extremely simplistic and will not be explained (All of the getter and setter methods)

Functional Code Blocks

|  |
| --- |
| public static ArrayList<Item> sortItems(ArrayList<Item> unsorted, String sort) |

This method is a static method that sorts an ArrayList of items according to a parameter sort that details the type of sort to be carried out. It overrides the default Collections.sort with sort methods specified in the ItemSort enum.

|  |
| --- |
| enum ItemSort implements Comparator<Item> |

This enum overrides the default Comparator included with Collections class. This allows to define our own sorting methods and easily sort a list of Items.

## 4.5 Language Code

The language code will be explained using the **OpeningScreen** class. The code for OpeningScreen can be found at the following link:

|  |
| --- |
| https://github.com/FindItPS/Find-It-Android/blob/master/src/main/java/com/painlessshopping/mohamed/findit/OpeningScreen.java |

Note: The OpeningScreen class is used for the sake of simplicity, every Activity class follows the same method.

|  |
| --- |
| protected void onCreate(Bundle savedInstanceState) |

The onCreate for every Activity class includes the processes that said Activity undergoes upon starting up. In the opening screen specifically, it checks the saved user data to see if it is the user’s first time launching the app. If so, it will set the default theme and language using the ThemeHandler and LanguageHandler classes. On future launches, a saved theme or language will be loaded on startup.

## 4.6 Activity Code

The Activity code will be explained using the **BookSearch** class. The code for BookSearch can be found at the following link:

|  |
| --- |
| https://github.com/FindItPS/Find-It-Android/blob/master/src/main/java/com/painlessshopping/mohamed/findit/BookSearch.java |

Functional Code Blocks

|  |
| --- |
| protected void onCreate(Bundle savedInstanceState) |

This method is the code for what happens when the activity starts. It is where everything is instantiated and methods are called.

|  |
| --- |
| protected void onNewIntent(Intent intent) |

This is an override method for the Searchable interface. It allows the activity to handle search intents.

|  |
| --- |
| private void handleIntent(Intent intent) |

This is an override method for the Searchable interface. It tells the activity what to do with an intent when received. In the case of this activity, it creates a Search Query for the BookSearch website search classes.

|  |
| --- |
| public boolean onCreateOptionsMenu(Menu menu) |

This is another override method for the Searchable interface. It simply adds a clickable search button the action bar for searching.

|  |
| --- |
| public boolean onOptionsItemSelected(MenuItem item) |

This method executes code when the sort button in the top right corner is clicked. It creates a pop-up message asking a user how they would like to sort their data. It then executes the sortList method with the corresponding type.

|  |
| --- |
| public void sortList(String type) |

Uses the sortItems method from the Item class to sort the list depending on the type of search specified.

|  |
| --- |
| public void onBackPressed() |

Controls what happened when the back button is pressed on the Android screen. In this case, it will close the navigation drawer if open or navigate to the previously opened activity if not.

|  |
| --- |
| public boolean onNavigationItemSelected(MenuItem item) |

This method controls what happens when an item from the navigation drawer is selected. It will simply close the drawer if the current activity is selected and if another activity is selected it will open an instance of that activity.

# Appendix A - Code

<https://github.com/findit2016/Find-It-Android>

# Appendix B - Beta Testing and Reporting

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement Specification Reference** | **Software Features** | **Code Block(s) Tested** | **Status**  **(Pass/Fail w/ explanation)** | **Date Tested** |
| 1.1 | Must work for the entire Ottawa Area | SearchQuery.java | Fail  While an original attempt was made to use Google Shopping, something that could search all of the stores in the Ottawa area, efforts failed. Searches are now done by store and we are working to incorporate local stores within the Centrum area.  Technically though it does work in the entire Ottawa area, just only for chains of stores that have a store in Centrum. | 8/1/2017 |
| 2.1 | Retrieve Item Prices from Store Website | All Website Search Classes | Pass  For every store that we currently have a search class for, results show an accurate price of an item as listed on the website. | 8/1/2017 |
| 2.2 | Format and Display Prices of the Screen | CustomAdapter.java | Pass  Prices that are retrieved from the websites are displayed under the item titles in a clean looking fashion. | 8/1/2017 |
| 2.3 | Must have price comparison function for 3 items | None as they do not yet exist. | Fail  This function has been a low priority and has not yet been addressed. | 8/1/2017 |
| 3.1 | Sort in alphabetical order | sortItems() in the Item class | Pass  For every search that is made, results can easily and successfully be sorted from A-Z | 8/1/2017 |
| 3.2 | Sort in reverse alphabetical order | sortItems() in the Item class | Pass  For every search that is made, results can easily and successfully be sorted from A-Z | 8/1/2017 |
| 3.3 | Sort by price (High to Low) | sortItems() in the Item class | Pass  For every search that is made, results can easily and successfully be sorted from High-Low Price | 8/1/2017 |
| 3.4 | Sort by price (Low to High) | sortItems() in the Item class | Pass  For every search that is made, results can easily and successfully be sorted from Low-High Price | 8/1/2017 |

# Appendix C - Reflection of Learning

Overall this course has been a great learning experience that followed the expectations that I initially had.

* Expectations - what did you expect to learn?

Going into this course I expected to learn more of the project management side of software development since I already had prior experience developing independently.

* What did you learn?

I did end up learning a lot about project management in the context of a software development project, however I did also end up learning a lot about Android and the way that it works. I learned about the different types of SDLCs and their benefits as well as Gantt Charts for scheduling and GitHub for Version Control. Since I was developing independently before, I had only used a local server ran from my computer to store my files. However, being forced to use GitHub opened my eyes to the benefits of using GitHub even for independent projects.

* What about the course did you enjoy?

One of my favorite parts of the course was the Emerging Technologies Assignment. I learned a lot about a lot of really cool breakthroughs in technology and it incited a lot of excitement in me for the technology industry.

* What didn’t you enjoy?

There was nothing I really disliked in this course. If I had to concentrate on one thing that I didn’t enjoy as much as the rest it would’ve been the group work. Group work has never been my strong suit because I always feel as though two people can never work at the same pace and I am a perfectionist. But, that being said, I feel that a lot of the group work related challenges I faced throughout the duration of the project helped me to overcome some aspects of my perfectionism. So all in all, there was nothing about the course that I didn’t enjoy in some way or the other.

* What went well in the development of the app?

A lot of things went well in the development of the app. We didn’t really have trouble coming up with ideas and deciding on the most actionable one. We were also both able to accustom ourselves with Android relatively quickly for first-time developers. I had a lot of freedom working with the UI and I think that showed in the final product. In terms of working with GitHub, there was never a time that we were both working on the same file so we never had a problem merging our directories.

* What could you have done differently to streamline development process

In all honesty, the biggest thing that we could have done to streamline the development process would have been to more thoroughly plan out the coding stage. I don’t believe this to be a realistic expectation though considering neither of us had worked with Android before. We knew that Android worked off of Java so we planned according to our knowledge of Java, not knowing that Android is essentially completely different in some regards.

* How did the course prepare you for post-secondary/workforce?

This course provided a great sort of ‘preview’ into what the daily routines of someone working in software development looked like. In terms of post-secondary, I want to go to university so I don’t feel as though it has prepared me all too much for that considering that at most universities undergraduate computer science is very conceptual and different from project development. However, it helped me to understand the realities of working in computer science so if I wanted to take a gap year or skip university entirely I’d know where to start in building a hireable portfolio.

* How would a similar job in the software industry be different from your experience in class

The key difference would be that in the software industry, a lot more of our time would be dedicated to the development of the project whereas in school we have 3 other classes to worry about which take away from our development time. We’d also be working in a much larger team in the software industry so the correct use of version control would be paramount. Another big difference is that a lot more overtime would most likely be put in in the software industry do to its competitive nature.