



GOOD DEAL



Pranay Bajpai (pb1643)

Puneet Chugh (pc1837)

Jian-Wei Huang (jwh391)

INSTRUCTOR: Jeffrey Bickford

INDEX

Problem Overview	2
Current Solution and Existing Work	3
Solution	7
Implementation	9
Views and UI Drawings	18
Android Components and Technologies Used	20
Project Task Distribution	22
Goals for Each milestone	23
Feedback and Conclusion	24

PROBLEM OVERVIEW

Motivation

When we go to a store, we like something say a TV we find the one we like but we have to spend a good amount to buy it. We are hesitant about buying the TV as we do not know if it's worth the money. The TV may have design flaws; it could be using outdated technology or it could be simply overpriced. We need someone to reassure us to make the purchase.

We end up searching in forum discussions, opening review websites and end up being confused as reviews for all the products are not easy to find and also because of the people's mixed opinions. It becomes hard for us to reach a decision. So, the confusion still persists whether to proceed with the deal or not.

Good Deal app will let you know if that is a good deal by checking the price difference and rating of the product.

The other main objective of Good Deal is to be able to find you the store that can get you the best deals. Generally, we go to a store to buy a product or look online to find the price. But the problem is that there's no common portal that compares the price of same product from different store like Walmart, BestBuy, Target, eBay, etc. under one umbrella.

Imagine a scenario that you want to buy a 20 Liter Whirlpool Microwave. The first step would be to find out the reviews for this microwave and then reach a decision if it's worth buying. In case you decide to buy it, your next step would be to find out the store that sells it for the lowest price. All these functions will be performed by Good Deal. When you enter the name and price of the microwave, you will get to know if it's a good deal or not, if there's any other store/online that sells the same product at a lower price. All you need to do is to enter the name of the product and price. Further you can also save the product and its prices for a later search.

Good Deal will help you make an informed decision taking into account product rating and price at which the product is sold.

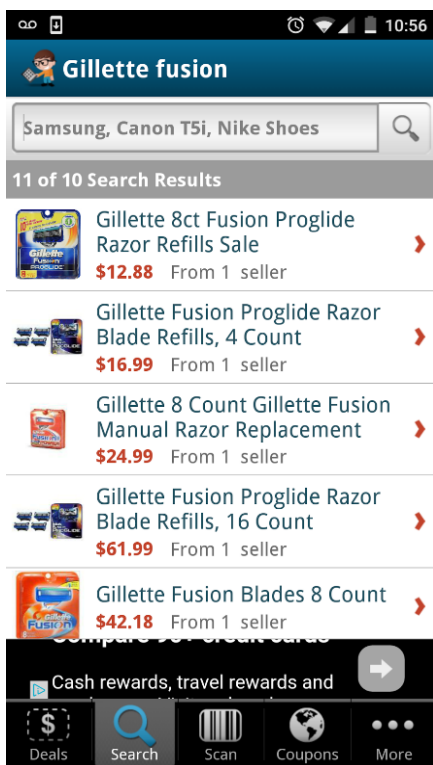
CURRENT SOLUTION AND EXISTING WORK

First let us perform comparison of Good Deal app with some other apps and then reach a conclusion:

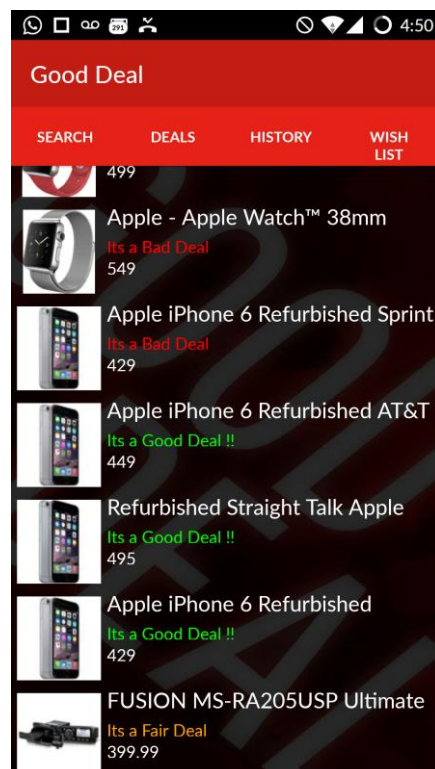
Comparison with BuyVia app

There are some apps that search the products, but they don't tell the user if it's a good deal. The user has to read the reviews to find out the product. However, Good Deal app uses information like product rating and price parameters to reach to the conclusion if it's a Good Deal, Fair Deal or Bad Deal. An app "BuyVia" is capable of finding the products but does not have the discretion to tell if one should go ahead with the deal.

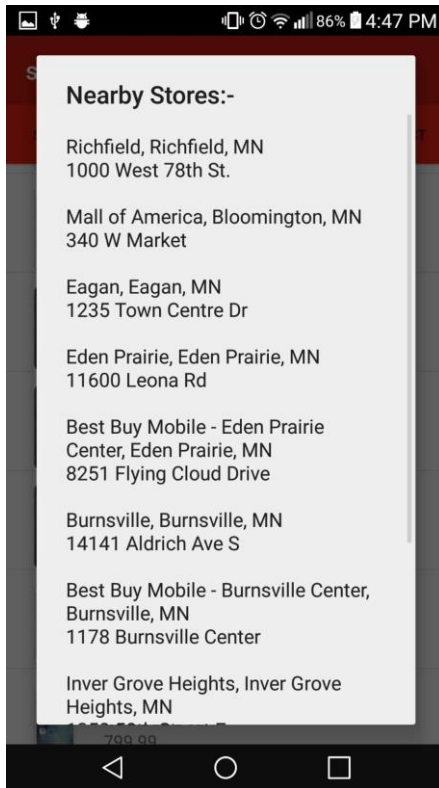
BuyVia App



Good Deal App



The other add-on feature in Good Deal app is that it lets you find the nearby stores for any product.

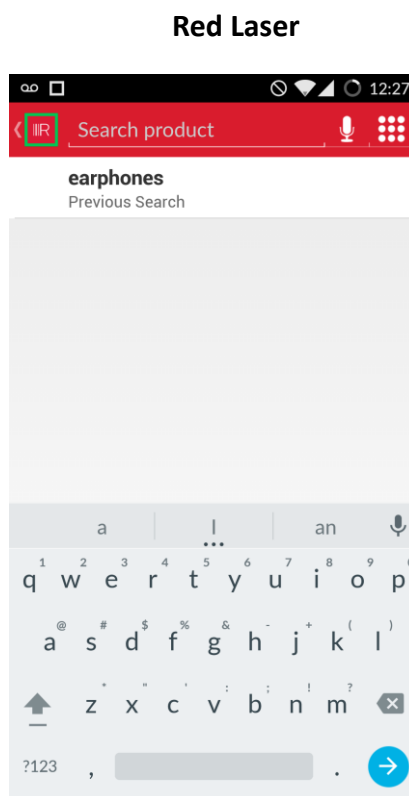
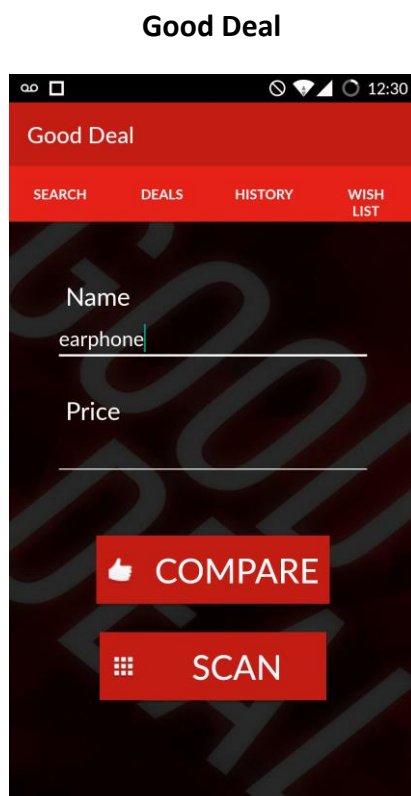


Another feature that Good Deal has got is that it can scan the Barcode and QR code to find the name of the product and search the product.

Comparison with “Red Laser”

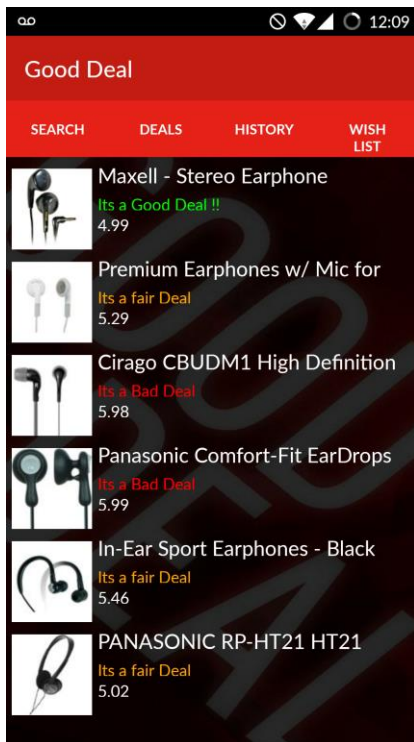
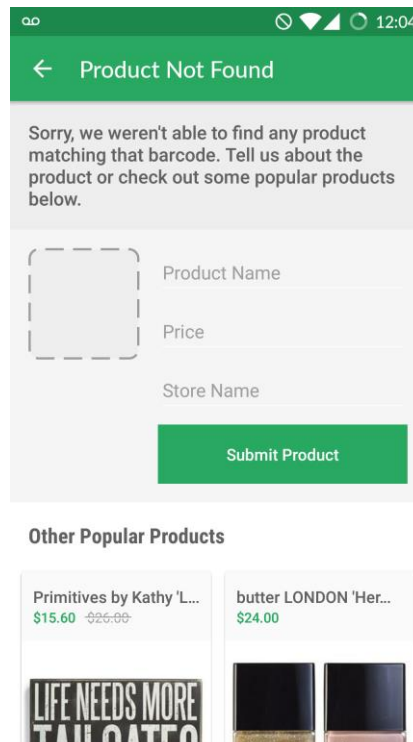
However, there’s an existing app “Red laser” that searches the products based on the name entered. But it does not let us know if it's a good deal or not and the results do not even show the price of the product.

Also the interface of Red laser does not take full advantage of Android’s design guidelines and is not as intuitive as good deal



Comparison with “Shop Savvy”

“Shop Savvy” app gives you the option of scanning the barcode of the product, but the app itself says that search results are best if products are of more than \$10. So, when we tried to scan the barcode of earphones priced a mere 5 dollars, ShopSavvy did not bring any results. And on the contrary, Good Deal indeed brought about some good results.

Good Deal	Shop Savvy																					
 <p>The screenshot shows the Good Deal app interface. At the top, there's a red header with the app name "Good Deal" and a navigation bar with "SEARCH", "DEALS", "HISTORY", and "WISH LIST". Below this, a list of earphones is displayed with their images, names, and prices. Each item has a label indicating its deal status: "It's a Good Deal !!", "It's a fair Deal", or "It's a Bad Deal".</p> <table><tr><th>Product Name</th><th>Price</th><th>Deal Status</th></tr><tr><td>Maxell - Stereo Earphone</td><td>4.99</td><td>It's a Good Deal !!</td></tr><tr><td>Premium Earphones w/ Mic for</td><td>5.29</td><td>It's a fair Deal</td></tr><tr><td>Cirago CBUDM1 High Definition</td><td>5.98</td><td>It's a Bad Deal</td></tr><tr><td>Panasonic Comfort-Fit EarDrops</td><td>5.99</td><td>It's a Bad Deal</td></tr><tr><td>In-Ear Sport Earphones - Black</td><td>5.46</td><td>It's a fair Deal</td></tr><tr><td>PANASONIC RP-HT21 HT21</td><td>5.02</td><td>It's a fair Deal</td></tr></table>	Product Name	Price	Deal Status	Maxell - Stereo Earphone	4.99	It's a Good Deal !!	Premium Earphones w/ Mic for	5.29	It's a fair Deal	Cirago CBUDM1 High Definition	5.98	It's a Bad Deal	Panasonic Comfort-Fit EarDrops	5.99	It's a Bad Deal	In-Ear Sport Earphones - Black	5.46	It's a fair Deal	PANASONIC RP-HT21 HT21	5.02	It's a fair Deal	 <p>The screenshot shows the Shop Savvy app interface. At the top, there's a green header with a back arrow and the text "Product Not Found". Below this, a message states: "Sorry, we weren't able to find any product matching that barcode. Tell us about the product or check out some popular products below." Below the message is a form with fields for "Product Name", "Price", and "Store Name", and a green "Submit Product" button. At the bottom, there's a section titled "Other Popular Products" showing two product cards: "Primitives by Kathy 'L...' for \$15.60 (original price \$26.00) and "butter LONDON 'Her...' for \$24.00.</p>
Product Name	Price	Deal Status																				
Maxell - Stereo Earphone	4.99	It's a Good Deal !!																				
Premium Earphones w/ Mic for	5.29	It's a fair Deal																				
Cirago CBUDM1 High Definition	5.98	It's a Bad Deal																				
Panasonic Comfort-Fit EarDrops	5.99	It's a Bad Deal																				
In-Ear Sport Earphones - Black	5.46	It's a fair Deal																				
PANASONIC RP-HT21 HT21	5.02	It's a fair Deal																				

Apart from the capability of categorizing the products, the Good Deal app has many features that would otherwise require four-five apps. For example, Barcode and QR code scanning capability, finding nearby stores and so on.

SOLUTION

Proposed Solution

The user goes to a store and finds a product he wants to buy and he knows the price. Now , he is not sure if he wants to buy the product, he wants to know if he is getting a good deal. He pulls out his phone and starts Good Deal app which allows him to:-

- Search for the items and know the lowest cost
- Enter the price of the item and compare it to the price of the product elsewhere.
- Find weather they can find the item cheaper or online or in another store.
- Get Directions to the nearby store or a link to the website where they can buy it online.
- Save item for later search.
- Get warning if the item is a lemon/defective or has bad reviews.

Thus, the app provides a sense of reassurance to the users telling him if he should buy the product or not.

Work Flow

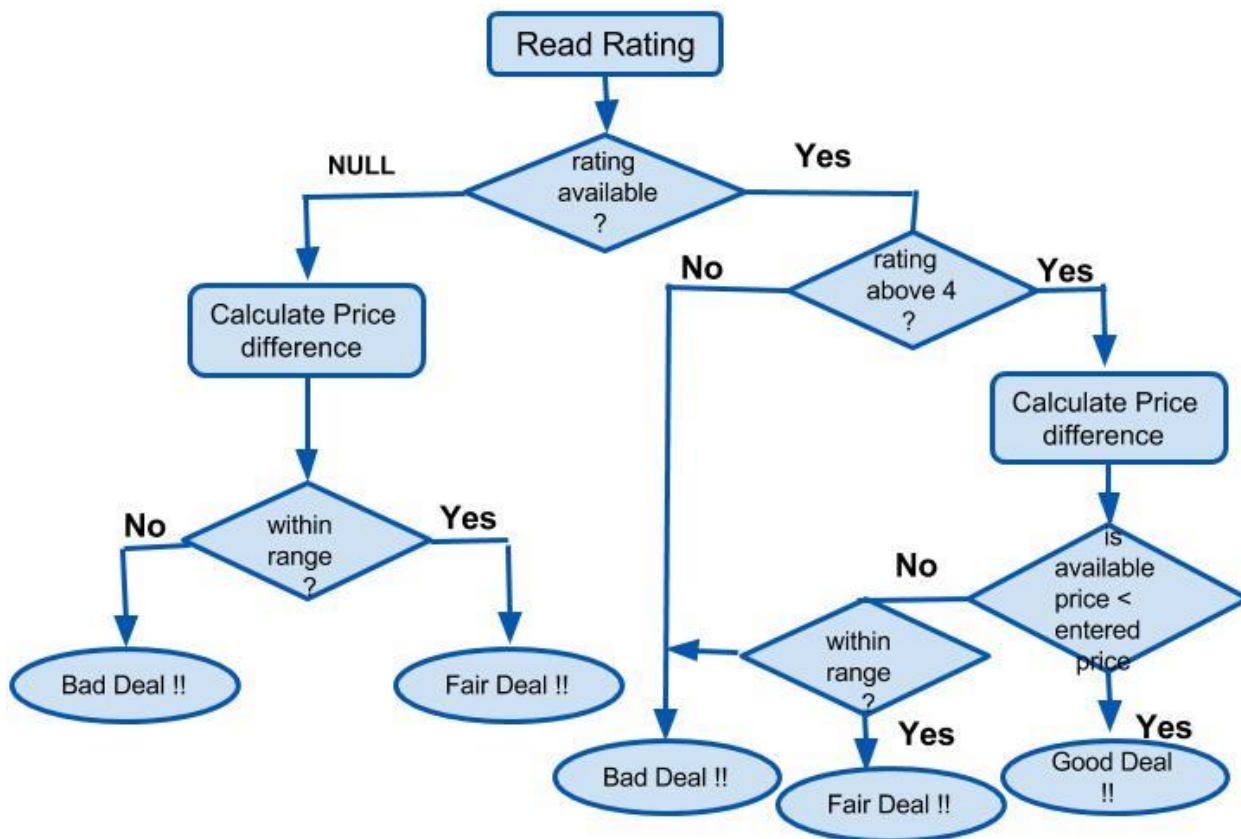
This app categorizes the products based on two factors namely:

- 1) The difference of available price and entered price
- 2) Product Rating

The Good Deal app categorizes the products into three categories:

- 1) Good Deal – For which you don't need to hesitate in buying. The product has a rating above 4 and available product price is less than entered product price.
- 2) Fair Deal – You can wait for it to get cheaper or go ahead with the deal. The available price is little above(within a range) what you want or/and rating is not available.
- 3) Bad Deal – You may want to abstain from buying such a product. Product price is way above the price you want it for and rating is also not good.

So, a product for which the rating is not available cannot be called Good Deal. It will be in Fair Deal if the price is within the range or Bad Deal otherwise.



Flow Diagram

IMPLEMENTATION

The Algorithm



Pseudo - code for the central logic of Good Deal

The fair deal price range is dependent on different percentages for different price categories, the following PERCENTAGE_POINTS are applied to different price categories:

1. PERCENTAGE_POINT_1 = 10%
2. PERCENTAGE_POINT_2 = 5%
3. PERCENTAGE_POINT_3 = 4%
4. PERCENTAGE_POINT_4 = 3%
5. PERCENTAGE_POINT_5 = 2%
6. PERCENTAGE_POINT_6 = 1.5%
7. PERCENTAGE_POINT_7 = 1%

1. **compareAndSetText()** method checks the price range and calls checkCategory() to find out the category to which the product belongs to.

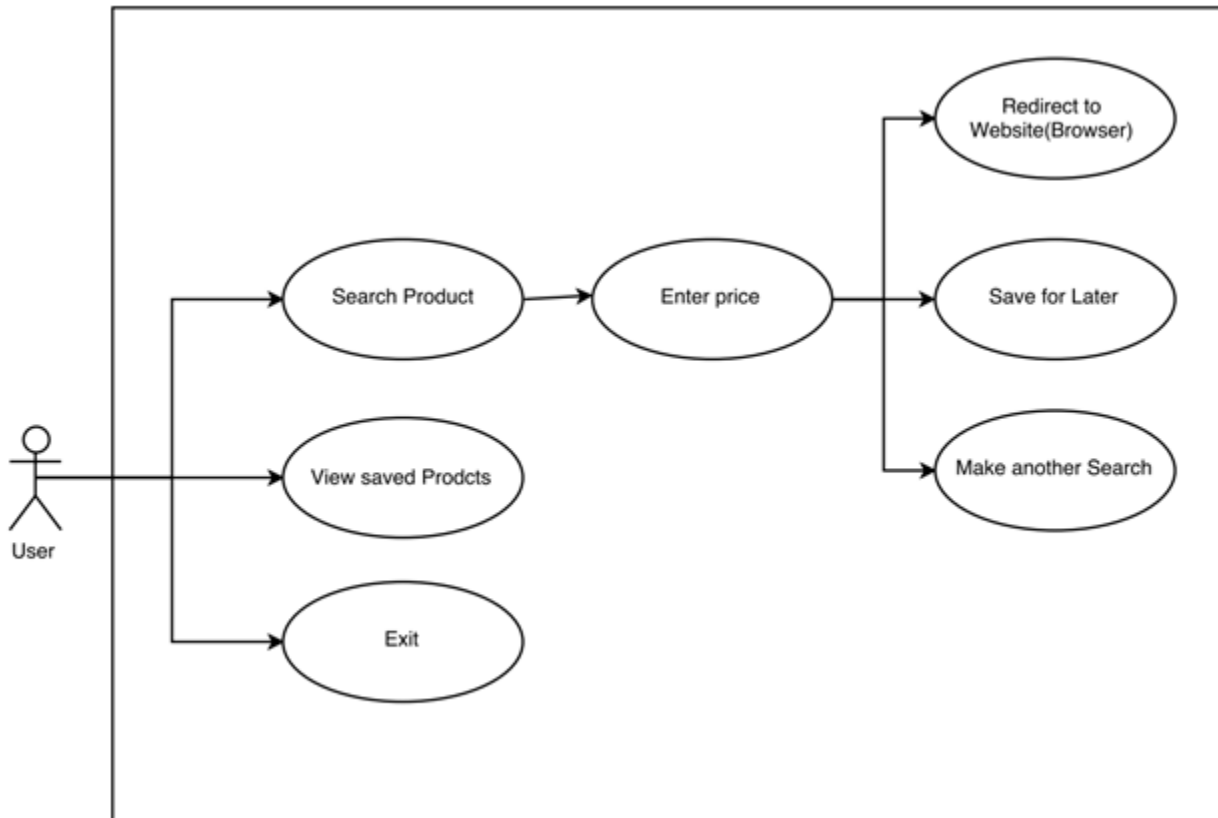
```
compareAndSetText(){  
  
    if(Product Price < 20){  
        checkCategory(productPrice, enteredPrice, productRating)  
    }  
  
    if(Product Price >= 20 && Product Price < 40){  
        checkCategory(productPrice, enteredPrice, productRating)  
    }  
  
    if(Product Price >= 40 && Product Price < 120){  
        checkCategory(productPrice, enteredPrice, productRating)  
    }  
  
    if(Product Price >= 120 && Product Price < 300){  
        checkCategory(productPrice, enteredPrice, productRating)  
    }  
  
    if(Product Price >= 300 && Product Price < 600){  
        checkCategory(productPrice, enteredPrice, productRating)  
    }  
  
    if(Product Price >= 600 && Product Price < 1000){  
        checkCategory(productPrice, enteredPrice, productRating)  
    }  
  
    if(Product Price >= 1000){  
        checkCategory(productPrice, enteredPrice, productRating)  
    }  
  
}
```

2) checkCategory(Product Price, Entered Price, Product Rating)

This method categorizes the product into Good Deal, Fair Deal or Bad Deal.

```
checkCategory(Product Price, Entered Price, Product Rating){  
  
    if((Product Price < Entered Price) && (Product Rating >= 4.0))  
        Then "Good Deal"  
  
    else if((Product Price < Entered Price) && (Product Rating == 0.0))  
        Then "Fair Deal"  
  
    else if((Product Price > Entered Price) && (difference) <= appropriate PERCENT_POINT)  
        Then "Fair Deal"  
  
    else "Bad Deal"  
  
}
```

Use Cases



Search Product

When the actor needs to find a product and wants to compare the prices, the first step is to Search the products button on the main page. Then the actor will have to enter the name of The product, which will be auto completed if entered correctly. If the product is found, a list with brands and prices will show up.

Enter Price

The user can or cannot enter a price for the product. If the user enters a price then the screen he will be redirected to will show the information about the products as well as assuring the user to buy the product with good reviews. The app will fetch bad reviews from the website and give the user a Warning message if the deal is bad or the product is faulty.

Redirect to Store

If the user likes an online web store where he can buy the item he can click on the item and he will be redirected to the vendor website. The app will be paused and the user will be moved to a web browser where he can make the purchase or know more details about the item.

View saved products

This is the list of products that the user saved for later use. The user can go to each of these products and a list with brands and prices will be displayed.

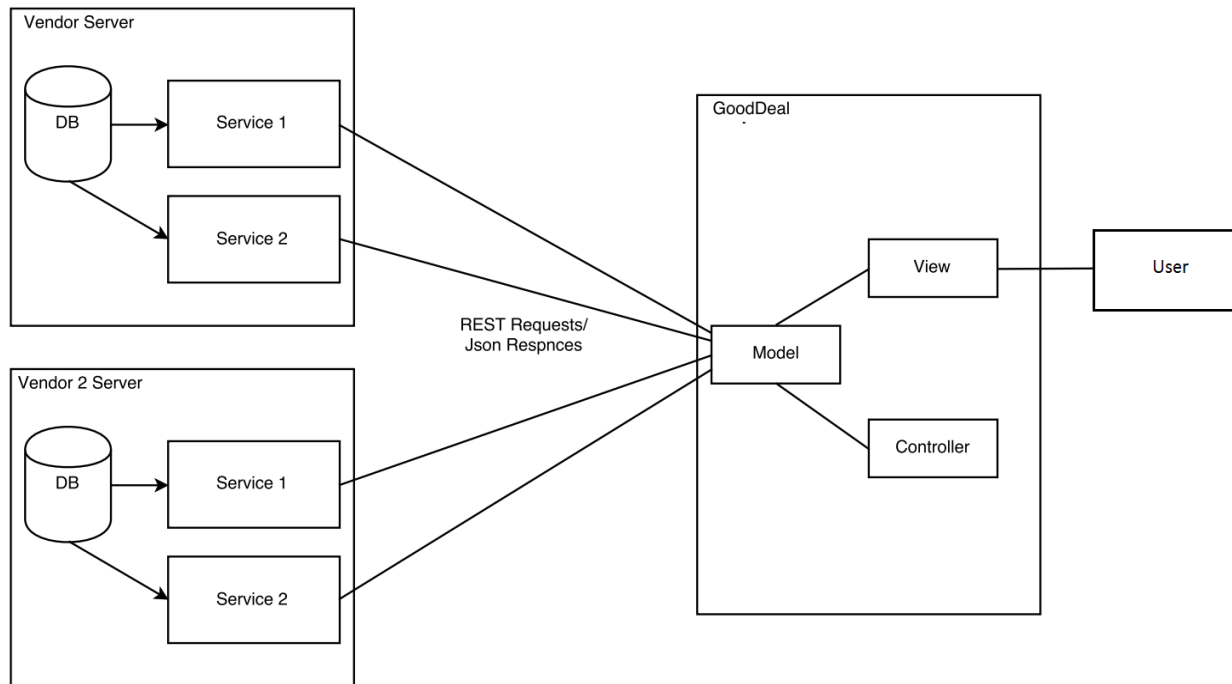
Save for Later

Helps the user to save items for later searches and retrieve an updated search if the user wants to buy the item later.

Make another search

Redirects the user to the main page of the app where he can enter another product with its price and make another search

Architecture Diagram



Communication with the server:-

The vendor servers basically contain a set of services identified by URL's. Those services obtain information from different databases and provide it to someone asking through a REST API (JSON/XML) over http or https.

There are several authentication mechanisms that these server may use. For instance, Server 2 may use an API key (OAuth Mechanism) which is unknown by other servers. There are several different Authentication mechanisms that are used by different servers.

MVC Framework

This section describes how MVC Framework is implemented in Good Deal app. It explains the functionalities performed by different activities using the Product, History and WishList models. It also has the screenshots of all the views.

Model:

Product:

- int id - The id of the item you want to buy in the database
- String productId - The id of the item you want to buy in the vendor store
- String productName - The name of the item you want to buy
- String productPrice - The price of the item you want to buy
- String productAvgRating - The average rating of the item you want to buy
- String productImageUrl - The image url of the item you want to buy

History:

- int id - The id of the item you want to buy in the database
- String name - The name of the item you want to buy
- String price - The price of the item you want to buy

WishList:

- int id - The id of the item you want to buy in the database
- String name - The name of the item you want to buy
- String imageUrl - The image url of the item you want to buy
- String price - The price of the item you want to buy

Controller:

MainActivity:

This is the very first activity that runs when you start the application. It provides you the EditText space to enter the name and the price of the item you want to buy. We also included the barcode scanner. After entering all the information, when you click the compare button the activity will send the information to SearchActivity which includes the vendor API for getting the product information and to compare the entered price with the vendor stores. This page also gives you the option of QR/Barcode scanning.

SearchActivity:

The activity gets input from MainActivity and pass it to API for more information to make comparison. API returns item's name, item's id, item's picture url, item's price and item's customer rating. After

getting all information we need, we pass it to our price-comparison algorithm to know whether the product is good deal or bad deal. All this information will be stored in a database for later use.

Fragment:

SlidingTabLayout:

To be used with ViewPager to provide a tab indicator component which give constant feedback as to the user's scroll progress

SlidingTabStrip:

To be used for the design of the tab strip.

ViewPagerAdapter:

Adapter for fragment elements which help application to initialize the sliding tab view.

SearchTab:

Control over the SearchTab fragment which provide the input of product name and product price. Both input will send an intent to searchActivity to make the search and do the comparison.

DealTab:

Control over the DealTab fragment to provide the result of the search. It will show a list of product returned from the vendor api with a good deal or bad deal icon after the product price goes through our comparison algorithm. DealTab also provide function for searching the product in the online store, search nearby store and save it in the wish list.

HistoryTab:

Control over the HistoryTab fragment to provide a list of search history. You can also delete the product in the history fragment.

WishListTab:

Control over the WishListTab fragment to provide a list of product saved in the wish list. You can also delete the product in the wish list fragment.

Adapter:

CustomAdapter:

This is the adapter responsible for displaying the list in DealTab. The adapter generate the list for all deals and pass the product to comparison algorithm for good deal or bad deal result.

HistoryAdapter:

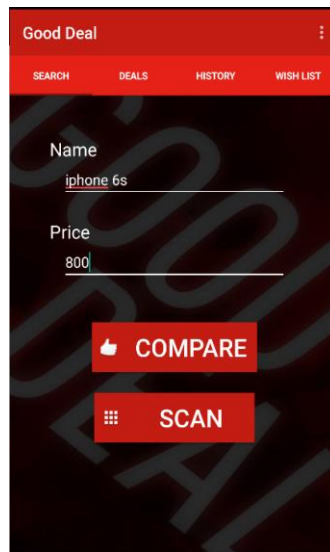
This is the adapter responsible for populating the list in HistoryTab. The adapter generates the list for all history item.

WishListAdapter:

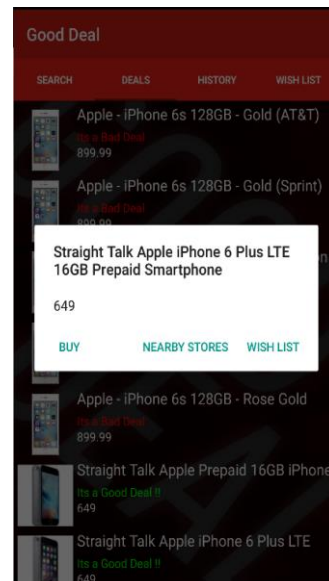
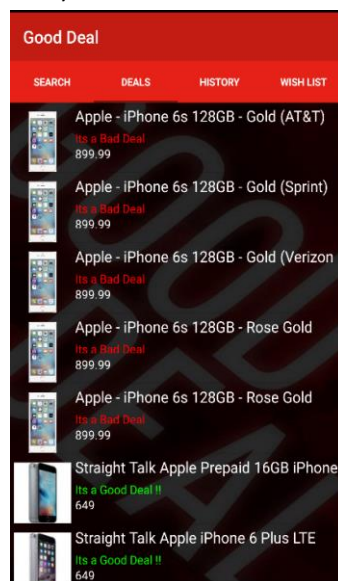
This is the adapter responsible for populating the list in WishListTab. The adapter generates the list for all wish list item.

VIEW & UI DRAWINGS

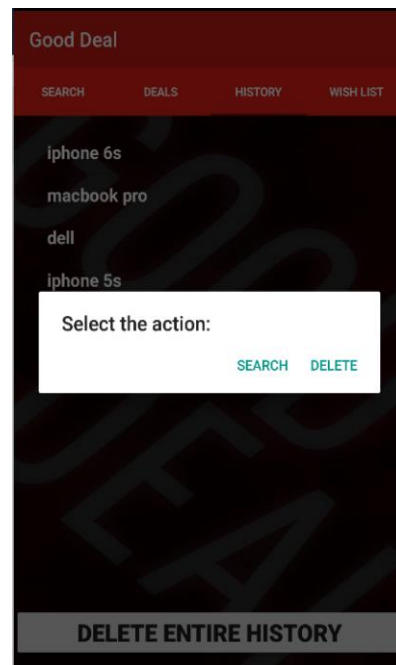
Main (SearchTab):



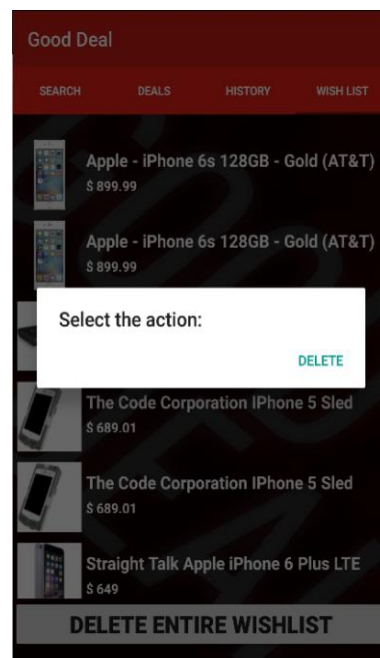
Deal List (DealTab):



History List (History Tab):



Wish List (WishList Tab):



ANDROID COMPONENTS AND TECHNOLOGIES USED

Vendor APIs (BestBuy, Walmart)

The Vendor APIs help us to search the product online. These APIs allow us to search the product, determine its price and fetch the user reviews. These user reviews will further help us in recommending the product to the user.

We pass the search keyword to the APIs and make Asynchronous calls get the result which are stored in the database. This is explained further in the document. We then apply the Algorithm previously explained to the determine the deal.

We have also used the Android Vendor Store APIs to get the latitude and longitude of all the stores in a particular radius for the item selected. The user's location is taken from the GPS and passed to the API to get the nearby stores which are displayed in the app.

We use the Google maps to pass get the directions for the store where the item is available by passing the coordinates of the user and the store.

The APIs allows you to query for place information on a variety of categories, such as: establishments, prominent points of interest, geographic locations, and more. We are using this API to search for locations on the basis of the category of item such as Electronics, Clothing etc.

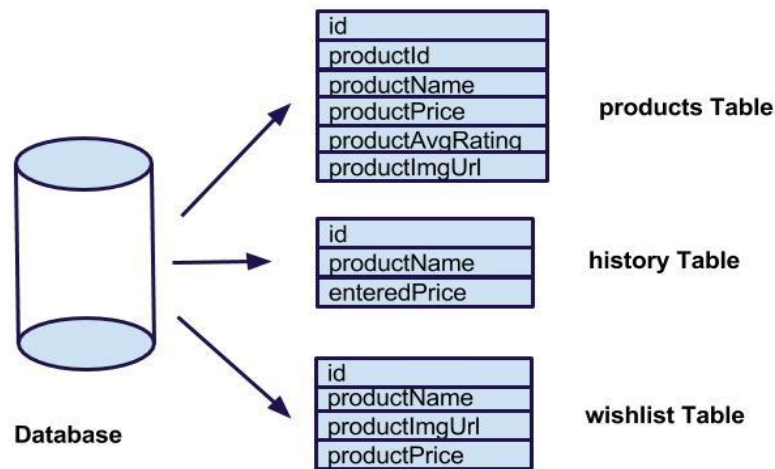
SQLite

SQLite is an Open Source database embedded into every Android device We are using SQLite to store the data about the lists of the items the user has saved for later.

The database in Good Deal has three tables:

- 1) **Products** - This table stores all the information of the products temporarily so that it's available to other activities for display. The results could be seen in the Deals Tab and when you click on the product, there are options available for buying, putting the product into wishlist and finding nearby stores. Everytime a new product search is made, the previous records are first deleted from this table and new results are stored.

- 2) **history** - This table stores the history of the searched products. This table persists as long as user wants. The records in the history table could be seen in the History Tab where user gets the option to delete an individual record or even the entire table
- 3) **wishlist** - This table stores the products the user puts in the wishlist from the DealsTab. The records could be seen in wishlist tab where user gets the option to delete an individual entry or even the entire table



Using ASYNC Task

Android implements single thread model and whenever an android application is launched, a thread is created. As in our app we are doing network operation ie making an HTTP request and receiving data back and storing in into the database on Compare button click in our application. On button click a request would be made to the server and response will be awaited. Due to single thread model of android, till the time response is awaited our screen is non-responsive. So we should avoid performing long running operations on the UI thread. This includes database and network access.

To overcome this we created new threads and implemented execute method to perform the various API calls, so UI remains responsive.

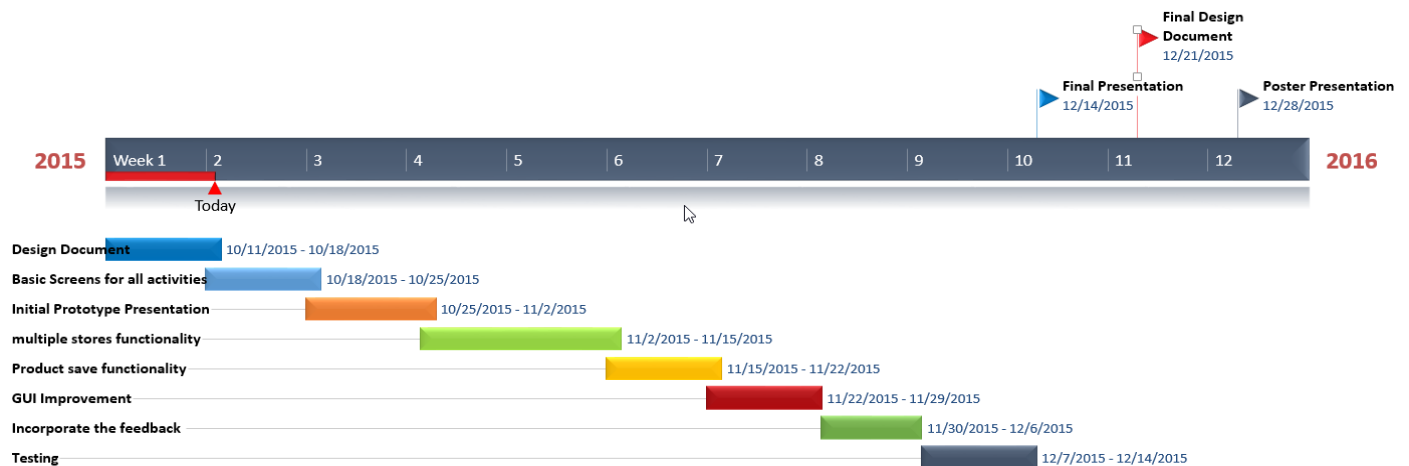
AsyncTask is ideal to be used for API call operations as they take only a few seconds.

PROJECT TASK DISTRIBUTION

All team members will work equally towards the design, logic and implementation and successful execution of project. Team members will collaborate over emails, skype and in person meetings on several occasions to discuss problems, approach and ideas.

Task Description	Actors
Idea Presentation	Pranay
Team formation	Puneet, Pranay, Tony
Project Discussion	Puneet, Pranay, Tony
Approach, idea and possible solutions	Puneet, Pranay, Tony
UI implementation	Tony
Setting up API Access	Pranay, Tony
Setting up SQ Lite	Puneet
Problem with logic and Algorithm	Puneet
Redesign and implementation	Puneet, Pranay, Tony
Final Implementation for Project	Puneet, Pranay, Tony
Testing	Puneet, Pranay, Tony
Final Review on code	Puneet, Pranay, Tony
Project report	Puneet, Pranay, Tony
Final review	Puneet, Pranay, Tony

GOALS FOR EACH MILESTONE



10/18/2015	Design Document
10/25/2015	Basic Screens for all activities should be working. To be able to enter the name of product and save the product for a later search.
11/2/2015	Initial Prototype Presentation. To be able to search a product from one major store and display on the screen.
11/15/2015	To be able to search a product from multiple stores and display on the screen
11/22/2015	To be able to Save the product and get to know good deals in future
11/29/2015	Improve the GUI for better presentation Test the app according to Test-cases and fix the crashes
11/30/2015	Poster Presentation There will be a working demo for the presentation at NYU or AT&T for the audience feedback on the app.
12/6/2015	Incorporate the feedback
12/10/2015	Test the app according to Test-cases and fix the crashes.
12/14/2015	Final demo
12/20/2015	Final Design Document

FEEDBACK RECEIVED AND CONCLUSION

We have tried to enlist the suggestions that we got in the poster-board presentation. So, the following list gives you the ideas that we would like to include in our future development:

1. Deals List in ascending order of price.
2. Faster search time and more efficient integration of APIs
3. Notification to the user if the product in the wish list sees a drop in price.
4. Slider tab for deletion of product from history/wish list
5. Better Algorithm design to determine the deal - using Machine learning.
6. UI Improvements, better use of Material UI fragments.
7. Integration of APIs from more Stores.
8. Improving the overall app stability.
9. Make the result from the search more accurate to user's need
10. Using web-scraping to find out the product rating from the product reviews where product rating is not explicitly given.

After implementing these functionalities, the app would be more equipped to let the users find a Good Deal for any product quickly and make the Deals List display even more presentable.