GitHub Username: maliameer

Smart Shop

Description

“Smart Shop” will allow users to maintain multiple lists of items (like groceries, electronics etc.). This app will use “Walmart” REST APIs to find matching items against each of that saved item in the nearby Walmart Store with below information:

- Image - Price - Aisle Location in Store - Ratings

Thus, helping user to know what options are available in the nearby Store and based on user’s preference user goes to the preferred Store and buy those items.
Since, Aisle Location is also listed in the app for each item, user can quickly reach to the desired item and do shopping quicker.
App will allow user to maintain multiple lists.

NOTE: This app is initially targeting Walmart REST APIs (which is quite nice) for this Capstone Project ONLY. This app can later be extended to hook other REST APIs from other stores like Target etc. Thus, giving user more choice for each of their desired item.

Intended User

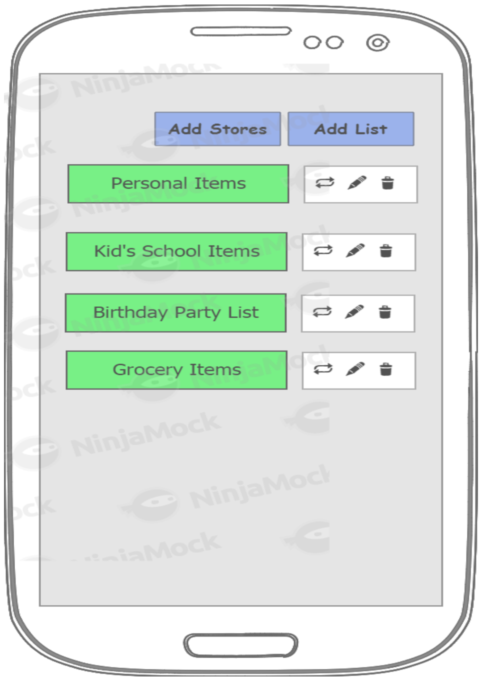
This app is for everyone since everyone does some kind of shopping.

Features

- Create multiple lists of items to be shopped.
- Add nearby Walmart Stores as provided by the Zip Code.
- Suggest available products for each list item in the Favorite Walmart Stores. Each item gets listed with “Image”, “Price”, “Aisle Location in Store” and “Ratings”.

User Interface Mocks

Screen 1



This is the main screen which allows user:

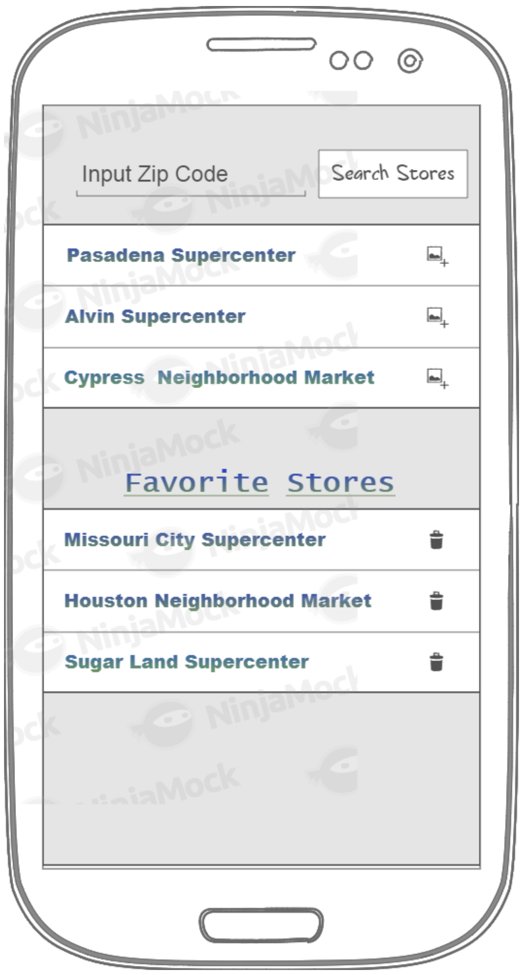
- To navigate to the screen for adding **Favorite Stores** by clicking “**Add Stores**” button **(Screen 2)**.
- To navigate to the screen for adding **Items List** by clicking “**Add List**” button **(Screen 3)**.
- View list of earlier added Items List. Each list has set of icons:

  ⇄ Clicking it will execute the list by navigating to **Screen 4**.

  ✎ Clicking it will edit the list by navigating to **Screen 3**.

  🗑 Clicking it will delete the list.

Screen 2



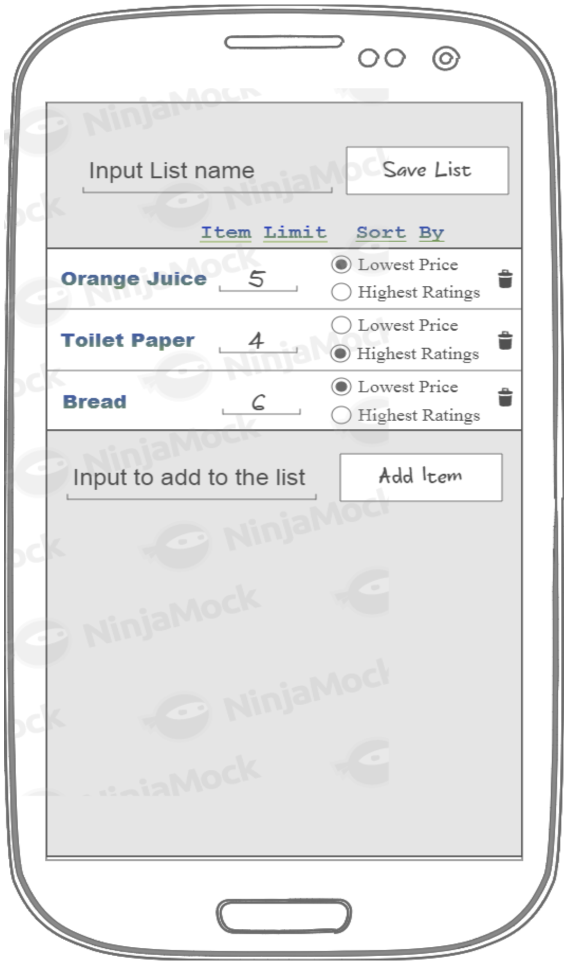
This screen allows user to:

- Search Walmart Stores by using Zip Code. First list shows the Stores as searched by the provided Zip Code.

  Next to each Store is a (**Add**) icon to add this Store as “**Favorite Store”**.
- Second list shows “**Favorite Stores**” earlier added.

  Next to each Store is a (Delete) icon to delete this Store.

Screen 3

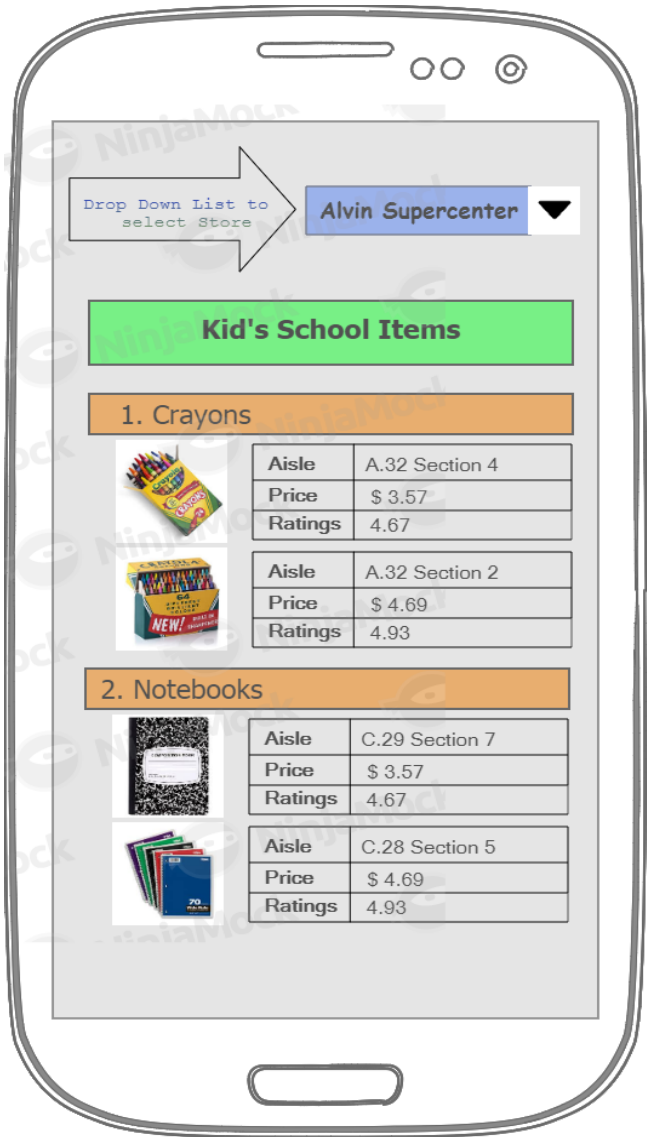


	Item Limit	Sort By
Orange Juice	5	[✓ Lowest Price] Highest Ratings
Toilet Paper	4	Lowest Price [✓ Highest Ratings]
Bread	6	[✓ Lowest Price] Highest Ratings

This screen allows user to add/update the Items List.
User can add as many items s/he likes.
Table in-between shows the list of items user added.
In each row, there is a text-field where user can input a **Limit**, to limit the number of products searched matching to that item.
Next to that text-field is a radio box group, allowing user to **Sort** searched products either by "**Lowest Price**" or "**Highest Ratings**".

NOTE: User doesn't have to type-in "**Item Limit**" and "**Sort By**" for each item. There will be options in App Settings where user can define the default values for both these fields.
Whenever new Item is added those values get set for that item as Defaults.
User can later override it depending upon user's preference for any item.

Screen 4



1. Crayons

Aisle	A.32 Section 4
Price	$ 3.57
Ratings	4.67

Aisle	A.32 Section 2
Price	$ 4.69
Ratings	4.93

2. Notebooks

Aisle	C.29 Section 7
Price	$ 3.57
Ratings	4.67

Aisle	C.28 Section 5
Price	$ 4.69
Ratings	4.93

This screen shows Search Results (“**Item Limit**” & “**Sort By”** applied for each item as earlier defined) against each item in the selected list.
Against each Searched Product, there is a table showing:

- Image
- Aisle Location in Store
- Price
- Ratings

At the top right, there is Drop-Down List for “**Favorite Stores**”, selecting any other Store from the list will pull Search Results for that selected Store.

Key Considerations

How will your app handle data persistence?

To persist information like Store, Item List etc, app will be using SQLite via Android Architecture Component “Room”.

Main Libraries

- Picasso will be used to handle the loading and caching of images.
- Android Architecture Component “Room”.

Using Google Cloud Engine - Endpoint API

Walmart REST API is designed such that it doesn’t support Sorting and it returns results maximum of 50 results at a time.
In to order to search Products matching to each List Item based on desired “**Sort By**” option. App requires a mechanism to pull all the matching Products against each Item (page by page, each consists of 50 records), Sort & Limit them as defined by the user for each item in the list.

This is a heavy-weight process considering it has to be done for each Item in the List and doing this in an Android Application will certainly impact app performance.
To avoid this performance issue, this process will be run at Google Cloud Engine and will be accessed through Google Endpoint API.
So, this project will be implementing a “**backend**” project to fulfill this objective.

When user navigates to **Screen 4**, an asynchronous Endpoint Loader will invoke that End Point API by posting item names and user’s “**Item Limit**” and “**Sort By**” (for each item) in JSON format. Once the call finishes, it will get results in JSON format and will be rendered according to **Screen 4**..

Next Steps: Required Tasks

Below are the main tasks to complete this project:

Task 1: Project Setup

This application will have 2 main Projects:

- **app**: For all UI like Activity, Layouts etc. to support Screen 1 - 4.

- **backend**: A web-application deployed at Google Cloud Engine as End Point API, which invokes Walmart API to search Products, merge the Results, Sort & Limit them and return back as JSON Payload.

Task 2: Implement UI for Each Activity

Build UIs (including Activity, Layout & other resources) for Screen 1 - 4.

Task 3: Implement Android Architecture Component “Room”

To support data as needed by Screen 1 - 4, implement Android Architecture Component “Room”.