- Description
- Intended User
- Features
- User Interface Mocks
  - Screen 1 - Upcoming Coupons
  - Screen 2 - Past Coupons
  - Screen 3 - Overflow Menu
  - Screen 4 - My Profile
  - Screen 5 - Settings
  - Screen 6 - Coupon View
  - Screen 7 - Tablet View
  - Screen 8 - Widget View
  - Screen 9 - Notification View
- Key Considerations
    - How will your app handle data persistence?
    - Describe any corner cases in the UX.
    - Describe any libraries you'll be using and share your reasoning for including them.
    - Describe how you will implement Google Play Services.
- Next Steps: Required Tasks
  - Task 1: Project Setup
  - Task 2: Create Database
  - Task 3: Create Coupon View Fragment
  - Task 4: Create Coupon List Fragment / Activity
  - Task 5: Create My Profile Fragment
  - Task 6: Create Settings Preferences Fragment
  - Task 7: Handle export or import data to / from drive
  - Task 8: Create widget
  - Task 9: Set up alarms
  - Task 10: Final testing

GitHub Username: darsh2

Coupons Tracker

Description

Coupons Tracker helps you keep track of all currently active offers provided by various merchants. It is an offline solution so you do not need an internet connection to view coupons. There are daily notifications that inform the user about coupons expiring each day.

Often times we get multiple offers from different merchants and we lose track of many. Albeit there are many apps and websites that dynamically update with new offers, there are some offers provided for specific users thereby making them hard to catch. We receive such discounts / offers either via text messages, emails or in app. Hence Coupons Tracker, as coupons can be manually entered to the app and kept track of.

Depending on circumstances, we are likely to not be interested in all available offers, hence the user can enter only those that are of immediate concern. Coupons can also be backed up to Google drive due to which the data can be imported at any time on another device.

Intended User

This app is for anyone who wants to locally keep track of various offers / discount coupons received from different merchants or retailers.

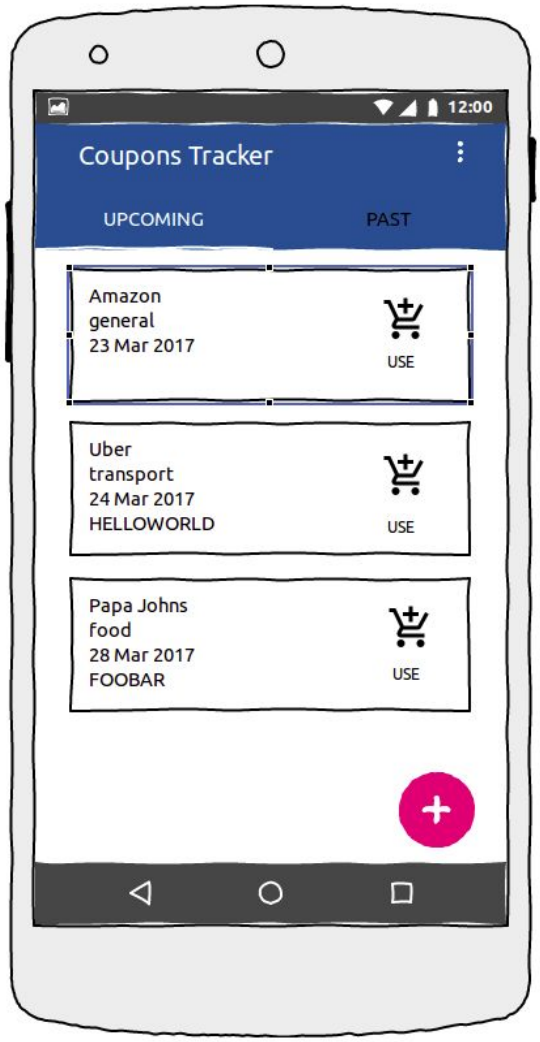
Features

- Locally save offer / discount coupons from various merchants or retailers.
- View upcoming and past coupons.
- View all the coupons used.
- Share, edit or delete any coupon.
- Set daily notification reminders about coupons expiring each day.
- Add widget to quickly browse coupons that can be availed.
- Backup data to Google drive and import the same on another device.

User Interface Mocks

The mockups were created using WireframeSketcher along with Android Lollipop Stencil and Material Design Icons assets.

Screen 1 - UPCOMING COUPONS



The landing screen of the app that lists all valid coupons. There are two tabs:

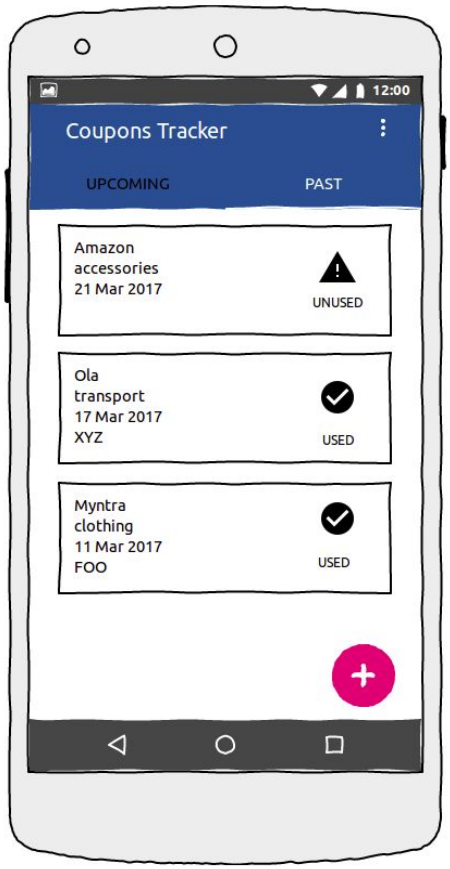
- UPCOMING - for coupons yet to expire
- PAST - for coupons that are no longer available

There is a floating action button to add a new coupon.

In the UPCOMING tab, coupons are sorted by the increasing order of their expiry dates. The layout of the coupon card is as follows:

- Coupon name (Papa Johns)
- Category (food)
- Valid until date (28 Mar 2017)
- Coupon code if present (FOOBAR)
- An add shopping cart image if the coupon has not yet been used or a check circle if a coupon is used

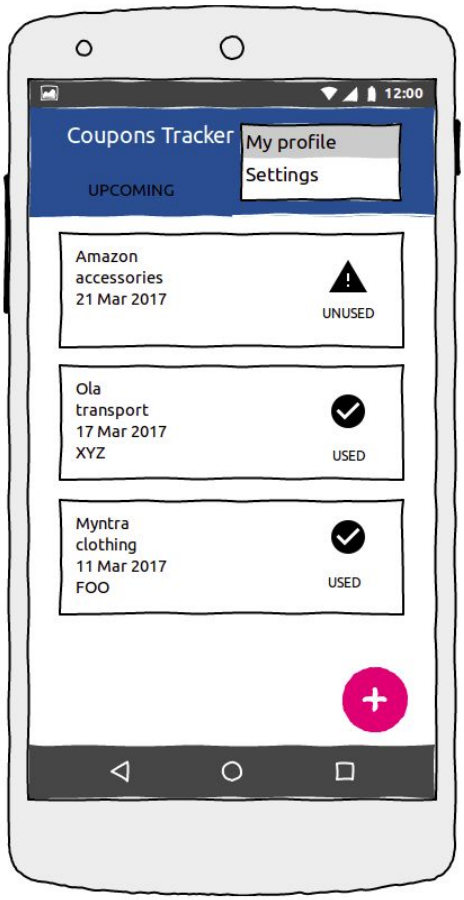
Screen 2 - PAST COUPONS



PAST coupons tab layout is similar to the UPCOMING coupons tab layout. The two differences are:

- Coupons are stored in the decreasing order of their expiry dates.
- For coupons that are unused, an alert image is shown indicating the same.

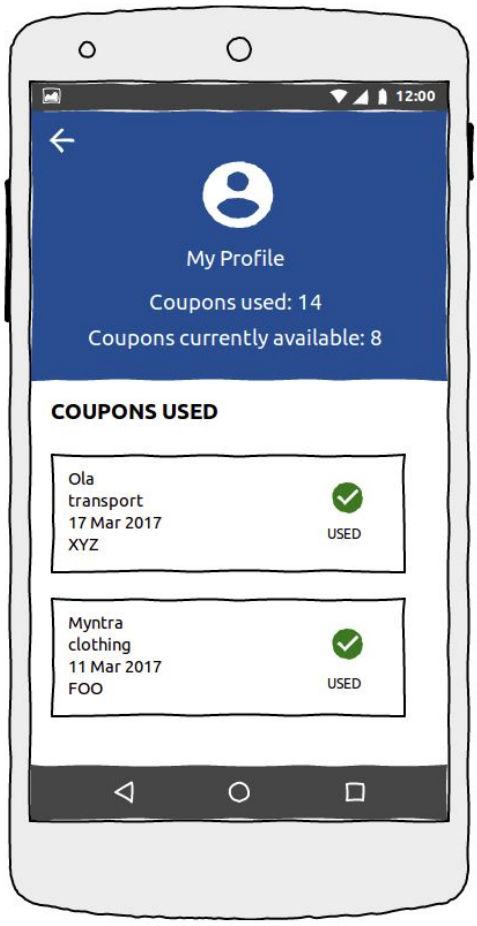
Screen 3 - OVERFLOW MENU



The overflow menu icon has two menu items:

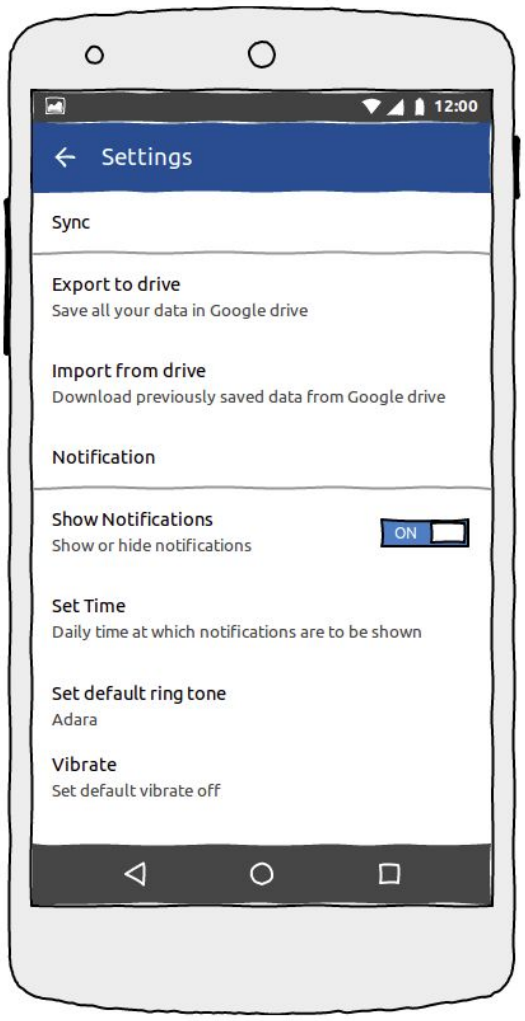
- My profile, that takes user to a profile screen showing activity in app
- Settings, that takes user to the app settings screen

Screen 4 - MY PROFILE



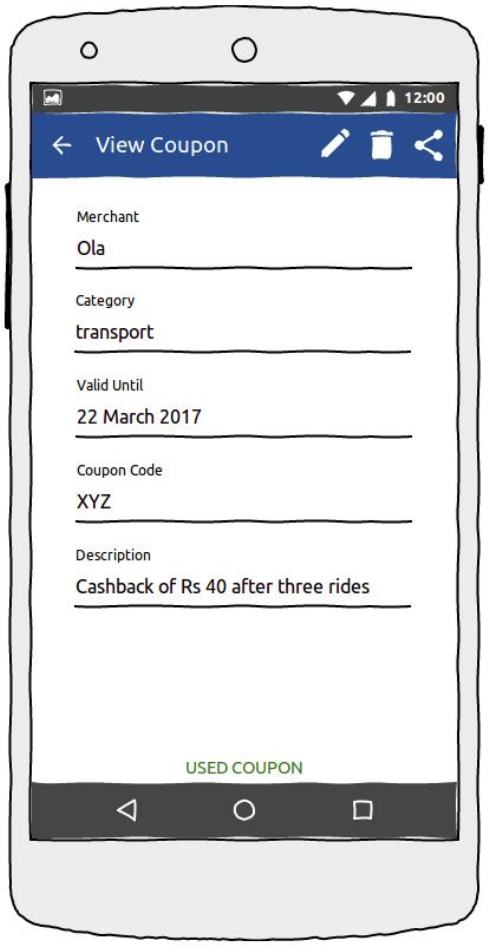
My Profile screen lists the number of coupons the user has used so far and also the number of coupons that are currently active and can be used. It also has a list view of all the used coupons sorted in the decreasing order of the coupon expiry date.

Screen 5 - SETTINGS



The sync settings option allows the user to backup / import previous app data. The notifications can be enabled or disabled. By default notifications are enabled and there is a daily recurring notification at 9 am indicating the number of coupons that expire today. User can change time, default ringtone and vibrate settings for the notification.

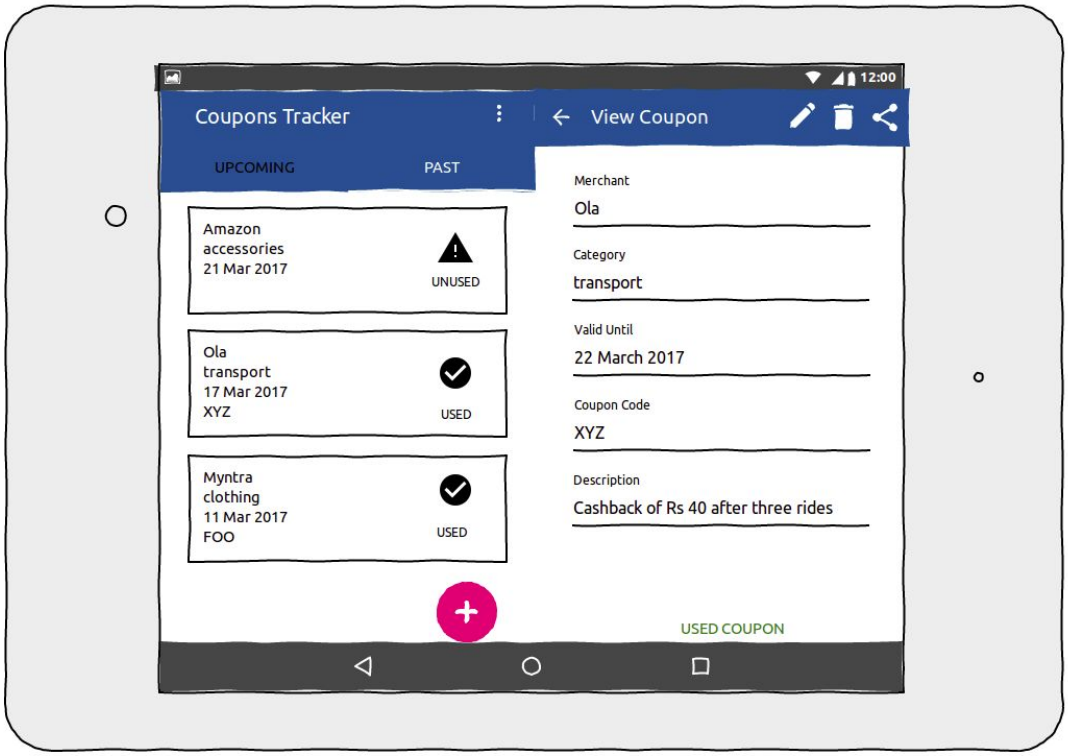
Screen 6 - COUPON VIEW



This gives the details of the coupon like merchant, category etc. There are 3 modes in which this screen can be opened:

- Mode CREATE - to create a new coupon with all the fields. The Toolbar title is ‘Create coupon’ and it will have only one menu item, a ‘check’ icon to save to db.
- Mode VIEW - to view an existing coupon. There will a button aligned to the bottom of the screen, which will have text ‘USE COUPON’ or ‘USED COUPON’ or ‘COUPON UNAVAILABLE’ indicating whether the coupon can be used or has been used. The Toolbar title is ‘View Coupon’ with edit, delete and share actions.
- Mode EDIT - to edit an already existing coupon. The Toolbar title is ‘Create coupon’ and it will have only one menu item, a ‘check’ icon to save to db.

Screen 7 - TABLET VIEW

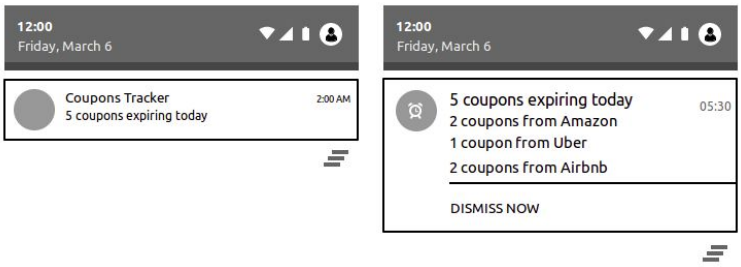


For tablets, all the screens are the same as for phones. The only difference is that tablets have a master - detail screen flow where all the coupon view screen will always occur in the right half of the tablet while other screens will be on the left.

Screen 8 - WIDGET VIEW

Widget consists of a scrollable list of coupons that are expiring today. On clicking any list view item, user is taken to the corresponding coupon fragment in the app.

Screen 9 - NOTIFICATION VIEW



Collapsed and expanded notification view.

Key Considerations

How will your app handle data persistence?

Data persistence is handled using a “coupon” sqlite table and SharedPreferences. The coupon table consists of the following fields:

- _id: Type integer - Row identifier for each coupon in the table. It is an integer primary key that is auto-incremented.
- merchant: Type text - Name of the merchant offering this coupon
- category: Type text - Indicates which category the coupon will belong to. Some merchants provide multiple discount offers and coupons ranging from incentives on shopping to cashback on mobile bill payments. Hence this category field is used to help identify where the coupon can be used.
- valid_until: Type integer - Date of form YYYYMMdd converted to a long to indicate till when the coupon is valid.
- coupon_code: Type text - Optional coupon code that needs to be used to avail the coupon offer.
- description: Type text - Short description of the offer provided by this coupon.

Shared preferences are used for saving notification settings. If notification is enabled, it is also used to note the default notification ringtone, whether device should vibrate and time of daily recurring notifications.

Describe any corner cases in the UX.

Coupon expiry date can only be from the day of adding coupon and after. That is, the date picker will have older dates disabled. This is to avoid users unnecessarily adding invalid entries.

When a coupon is added in the app, user has to specify the date until which the coupon will remain valid. This poses an issue if the device time zone is changed or user manually changes device time. In order to solve this, a date such as 24 Mar 2017 is stored as a long of value 20170324, basically a long obtained from Long.parseLong(“Date string of format YYYYMMdd”) because:

- Using Unix timestamp for date has issues with device time zone being changed.
- Saving date as text and then invoking datetime() sqlite function to sort coupons by date has performance issues.
- Saving it as a long where digits are of form YYYYMMdd is a simple solution that avoids issues with change in device time zone and also sorting based on an integer in sqlite is fast. It is a simple matter to convert it into a date again.

Daily notifications have to be sent at a user specified time. A recurring alarm with an interval set to one day is used to achieve this. Widget has to be updated once a day. Updating it at any random time does not help the user. Hence the widget will be updated every day at 2 am using a daily recurring alarm. However an issue arises when device time (Intent.ACTION_TIME_TICK) or device time zone (Intent.ACTION_TIMEZONE_CHANGED) is changed. To solve this, there will be a BroadcastReceiver listening to changes in device time or device time zone. If a change is detected, the previously set alarms will be cancelled and a new alarm will be set.

On turning off and turning on the device, all alarms set will be lost. Hence there is BroadcastReceiver that listens for Intent.ACTION_BOOT_COMPLETED to reset any alarms if applicable.

Describe any libraries you'll be using and share your reasoning for including them.

1. Butterknife - for binding android views and callbacks to fields and methods
2. RxJava - for asynchronous and event based programming
3. RxAndroid - for RxJava bindings in Android
4. Eventbus - for communication between activities, fragments, services etc
5. Gson - for creating and reading from json
6. Android design support library - for using Material UI elements like floating action button, coordinator layout, tab layout, text input layout etc.
7. CardView - for displaying coupons as cards
8. JUnit4 - for testing java methods
9. Espresso - for android UI testing

Describe how you will implement Google Play Services.

1. Play services drive api (com.google.android.gms:play-services-drive:10.2.0) - to export and import app db. Users can backup a copy of their db to Google drive. If app is installed on another device or so, users can import the previously added coupons by importing the backed up db file.
2. Play services ads api (com.google.android.gms:play-services-ads:10.2.0) - to display interstitial ads in the app. On going back from the profile screen, users will see an interstitial app if it is already loaded.

Next Steps: Required Tasks

Task 1: Project Setup

- Create project in Android studio.

- Add all required dependencies in gradle and configure project build script.

Task 2: Create Database

- Create a DbHelper class to handle create, upgrade and downgrade of database.
- Create CouponTable class and corresponding Contract class.
- Create ContentProvider to interact with db.
- Add tests to check for ContentProvider functionality.

Task 3: Create Coupon View Fragment

- Create coupon view fragment layout.
- Perform user input validation before entering details to db.
- Handle UI changes for different modes, ie, create, edit and view.

Task 4: Create Coupon List Fragment / Activity

- Create tabbed layout for coupon list activity.
- Load data from db and create list for upcoming and past coupons.
- Inflate menu.

Task 5: Create My Profile Fragment

- List all the coupons that the user has used.
- List data of coupons used by user and number of currently available coupons.
- Show interstitial ad on going back from profile fragment.

Task 6: Create Settings Preferences Fragment

- Implement settings preferences screen.
- Appropriately store and retrieve user preferences from SharedPreferences.

Task 7: Handle export and import data to / from drive

- Implement exporting db to drive.
- Implement importing db data from drive.

Task 8: Create widget

- Create widget screen for the app.
- Handle onClick navigation from widget screen to app.

Task 9: Set up alarms

- Set alarm for daily recurring notifications.
- On clicking a daily recurring notification, the user will be taken to a screen with a listview showing the coupons expiring today. An AsyncTask is used to get the notifications that expire today. This list view will be populated using loaders.
- Set alarm for updating widget at 2 am daily.
- Create BroadcastReceiver to ensure alarms are set if necessary after device reboot.

Task 10: Final testing

- Fix any bugs that may be found in the final round of testing.
- Clean project, create signed apk and test installRelease task.