USC CSCI-571: Web Technologies (Spring 2021)

Homework - 9
USC Films Android App

1. Objectives

- The objective is to create an Android application as specified in the document below and in the video.
- Become familiar with Java, JSON, Android Lifecycle, and Android Studio for Android app development and build a good-looking Android app.
- Learn the essentials of Google's Material design rules for designing Android apps.
- Learn to use the TMDB API and the Android SDK.
- Get familiar with third-party libraries for Android 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/.

2.3 Amazon Web Services (AWS)

AWS is Amazon's implementation of cloud computing. Included in AWS is Amazon Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud, and AWS Elastic Beanstalk, an even easier way to quickly deploy and manage applications in the AWS cloud. You simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server, PHP, and Python, Passenger for Ruby, IIS for .NET, and Apache Tomcat for Java.
The Amazon Web Services homepage is available at http://aws.amazon.com/

3. Prerequisites

This homework requires the use of the following components:

- Download and install Android Studio. Technically, you may use any 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, Hints are going to be your best friends!

Tutorials to help you get started with Android App Development:

Below is a list of a few YouTube tutorials that you might find handy to get started with Android App Development. You need not watch all of them, but just whatever you might find useful.

1. Android Development for Beginners - Full Course
2. Android Development Tutorial 2020 For Java Developers
3. Android App Development in Java All-in-One Tutorial Series (4 HOURS!)
4. Android App Development Tutorial For Beginners 2021 [Master Android Today]

4. High Level Design

This homework is an extended mobile app version of Homework 6 and Homework 8. In this exercise, you will develop an Android application to browse movies/TV Shows and look at the detailed information about them. Additionally, users can add their favorite movies/tv-shows to a watchlist for viewing later. These added items will be retained in the watchlist even after the user closes the app and opens it again. The user will also have an option to remove items from the watchlist.
The application will consist of 4 main screens:

1) Home Screen - The user can toggle between viewing movies and TV shows on the home screen.
2) Search Screen - Search for movies and TV shows.
3) Watchlist Screen - Lists all movies and TV shows added to watchlist
4) Details Screen - Shows more information about a selected movie or TV show.

The API calls will be the same as that of HW8, so the same Node.js backend can be used. You can test the app with a local backend before deploying it to AWS.

You may develop the app using **Java** or **Kotlin**. Please note that **you will not receive any development support from the teaching team if you choose Kotlin**. If you choose Kotlin, make sure the app is identical or similar to the one shown here.

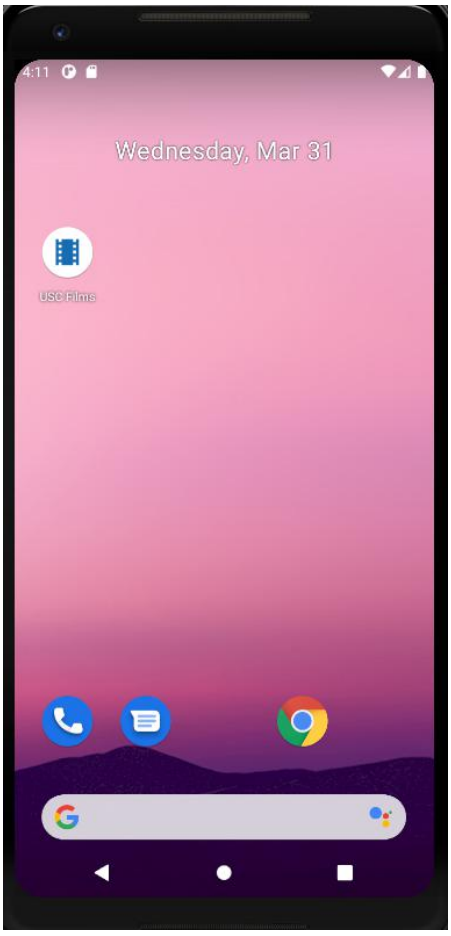
This app has been designed and implemented in a **Pixel 2XL emulator** by using **SDK API 30. It is highly recommended that you use the same virtual device and API level to ensure consistency.**

5. Implementation

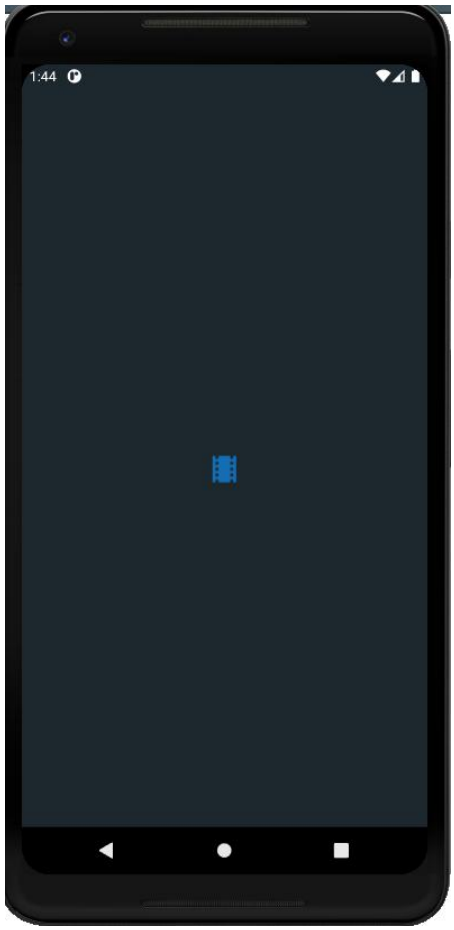
5.1 Main Activity

In order to get the app icon/image, please see the **hints** section.

The app begins with a welcome screen which displays the icon provided in the **hints** section. This screen is called the Splash Screen and 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.



App Icon



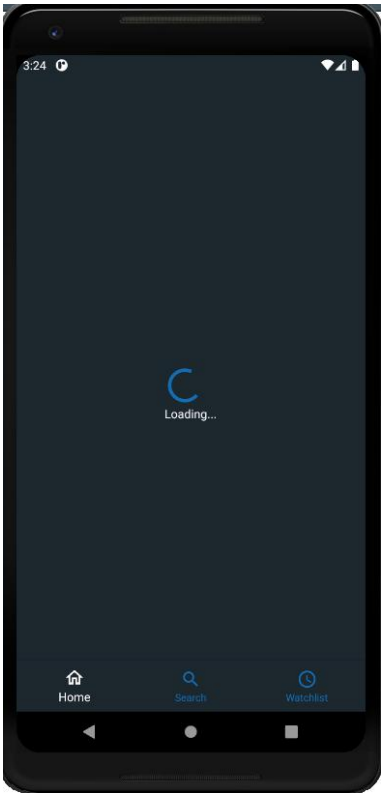
Splash Screen

5.1.1 Home Fragment

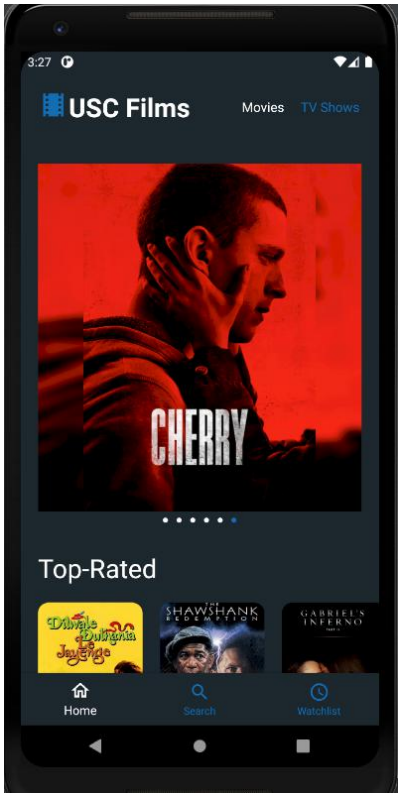
When you open the app, there will be an **initial spinner** while the data is being fetched using Volley. After the main screen opens up, there will be a **navigation bar** at the bottom with three options to select from: Home, Search, and Watchlist (in this order). These three screens can be implemented using fragments. To see how to implement a fragment, please see the hints section. The app will first open with the Home Screen.

The home screen will have the name of the app - **USC Films** displayed at the top alongside the logo, with **two tabs** on the **right-hand side** - **Movies and TV Shows**. The selected tab will be highlighted in white.

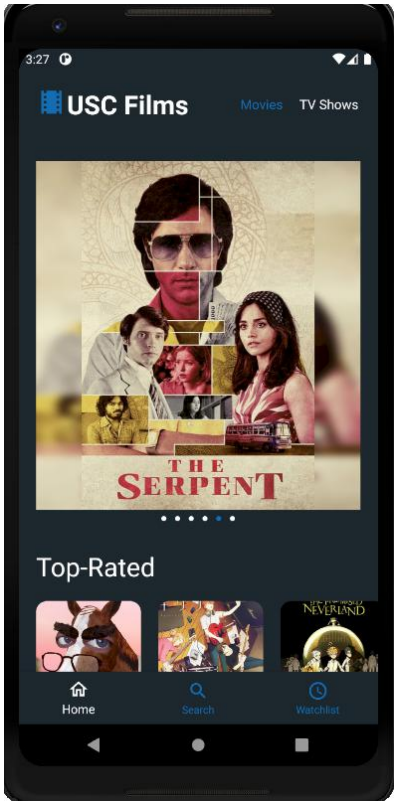
The home screen will initially be populated with the movies tab. The user will be able to toggle between the movies and tv show views for the home screen using these tabs. Both views of the home screen (movies and tv) are shown in figures below.



Loading Screen-Spinner



Movie view



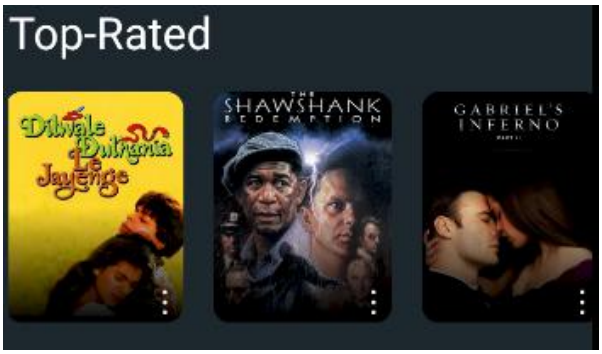
TV Shows view

5.1.1.1 Movie view:

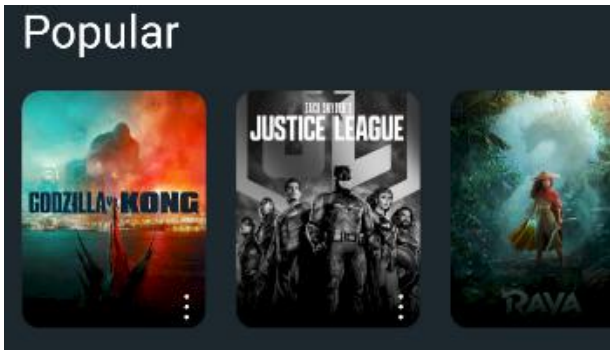
The movie view consists of a title bar containing the main heading at the top and two tabs to switch between the movie view and the tv view. This title bar should stay in place and should not move when the user scrolls down.

Below that, there is an image slider with a total of **six** images, for a duration of **600 ms** each. These movies are also superimposed on a blurred out version of the same image. Please refer to the hints section to see how to implement the image slider and how to form a blurred out version of images.
The **images** are obtained from the **movie/now_playing** endpoint. Following is the template of the API Call:
https://api.themoviedb.org/3/movie/now_playing?api_key=<<api_key>>&language=en-US&page=1

Next, we come to the horizontal scroll views.



Horizontal scroll view (Top - Rated) for movies



Horizontal scroll view (Popular) for movies

There are two horizontal scroll views beneath the image slider - Top Rated and Popular.
Every card on the horizontal scroll view consists of the poster_path of the concerned movie/tv show, and a 'show menu' button at the bottom right corner. The image also has an image gradient as a part of this foreground, so that the menu button stands out. To see how to implement an image gradient in the foreground of an image, check out the hints section.

Top Rated Movies scroll view displays a list of Top-Rated Movies from Top-Rated Movies endpoint.

Following is the template of API call :
https://api.themoviedb.org/3/movie/popular?api_key=<<api_key>>&language=enUS&page=1

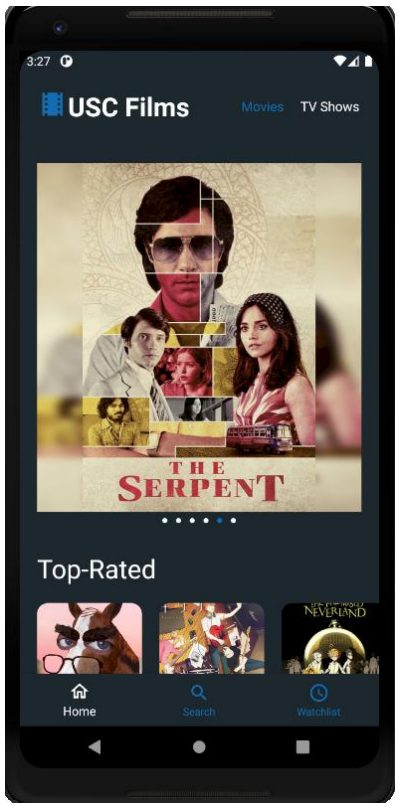
Popular Movies scroll view displays a list of Popular Movies from Popular Movies endpoint. Following is the template of API call:
https://api.themoviedb.org/3/movie/popular?api_key=<>&language=enUS&page=1

Rest of the field information is the same as HW8. The cards in both these views have to be populated by just the image returned by the image found in the **'poster_path'** field of both, the 'Top Rated' and 'Popular' endpoints. Both the scroll views should have **10 cards** each.

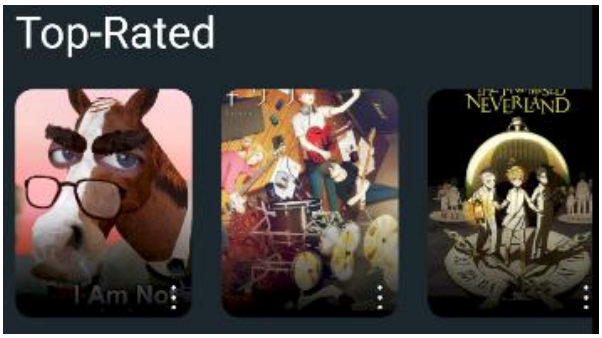
Each card should further have a **single click functionality** and have a button to view a **pop up menu** (Hint: this button can be implemented using a TextView instead of a button, to make things easier). To see how to implement an onClickListener, check out the **hints section**. On single click, a screen with all the details of the particular movie should be displayed. More information on the Details Activity is available in the **Details Activity section**. On pressing the menu option, a menu consisting of four options will appear. The functionality of this menu is described in the **Pop Menu Section.**

5.1.1.2 TV View:

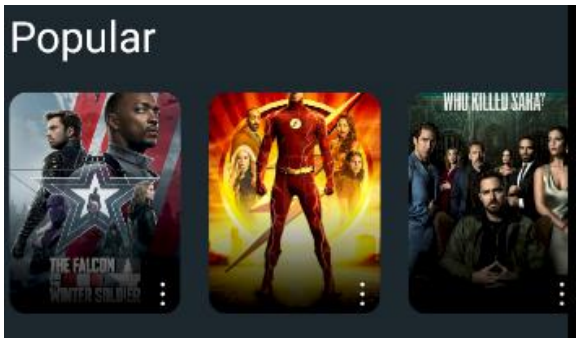
The TV view will have similar functionalities, with just the difference of API endpoints.



The images used in the **image slider** of the **TV view** is obtained from the **/tv/trending** endpoint of the TMDB API. Following is the template of the API call:
https://api.themoviedb.org/3/tv/trending?api_key=<>&language=enUS&page=1



Horizontal scroll view (top rated) - tv shows



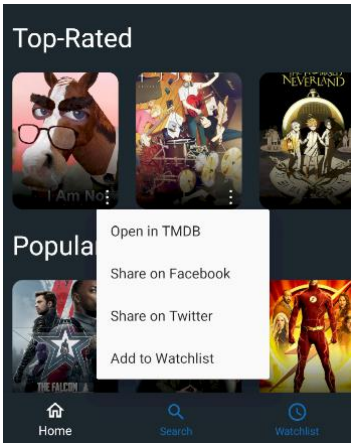
Horizontal scroll view (popular) - tv shows

Top Rated TV scroll view displays a list of Top-Rated TV shows from Top-Rated TV shows endpoint. Following is the template of API call:
https://api.themoviedb.org/3/tv/popular?api_key=<<api_key>>&language=enUS&page=1

Popular TV scroll view displays a list of Popular TV shows from Popular tv shows endpoint. Following is the template of API call:
https://api.themoviedb.org/3/tv/popular?api_key=<>&language=enUS&page=1

5.1.1.3 Pop up menu:

The pop up menu will look as follows:



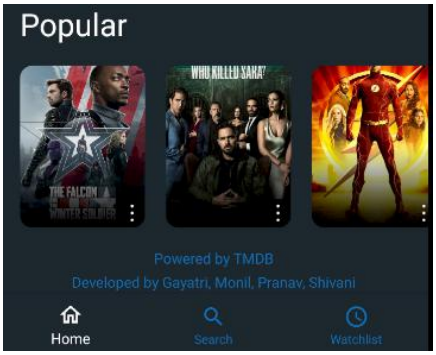
Upon pressing the menu buttons on each of the cards in the horizontal scroll views (in both, the movie and tv views), a pop up menu having four options will appear. These **four options** will be:

1) Open in TMDB - This option will take the user to the official tmdb page of the movie/TV show.
2) Share on Facebook - This option will enable the user to share a link to the official tmdb page of the concerned movie/tv show to facebook, in a browser.
3) Share on Twitter - This option will enable the user to share a link to the official tmdb page of the movie/tv show to facebook, in a browser.
4) Add to Watchlist - This option will enable the user to add the movie/tv show to the watchlist section of the application for viewing it later.

For more information on how to implement pop up menus and how to open external urls in the browser inside the app and pass information to them, please refer to the **hints** section.

Footer:
Both the Movie and TV views will have a footer at the bottom, after the 'popular' horizontal scroll views. The footer will have **two text views**:

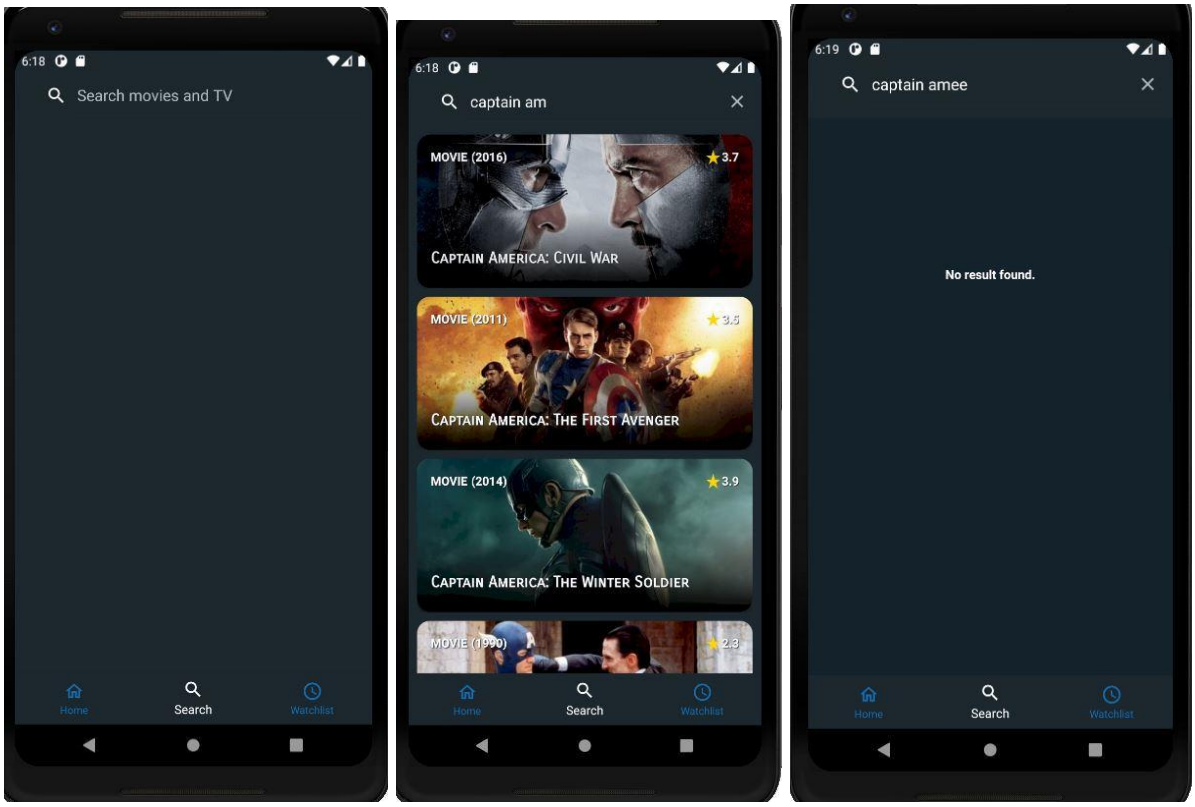
1) Powered by TMDB (clickable): The text to be displayed is 'Powered by TMDB'. Upon clicking, the user should be redirected to the following url: https://www.themoviedb.org/
2) Developed by <student_name>: The text to be displayed is: 'Developed by <student_name>



Note:
All the images and icons used in the app are image/vector assets created in Android Studio. Each icon can be added inside an ImageView. Play around with different types of logos that Android Studio provides and learn how to use them. More about how to use them in the **hints section.**

5.1.2 Search Fragment

In this fragment, you will implement a search feature that will display search result cards of movies and TV shows when you type on the toolbar on top of the screen, as shown in the picture below.



To implement the search fragment, you will be using a **Toolbar** that contains a **SearchView** widget inside it. The SearchView widget should contain the hint - "**Search movies and TV**", and should **automatically get activated with the cursor on the screen** once you navigate to the search fragment. The search result cards below are implemented using **RecyclerView**. Show a maximum of **20** search results at once. If there are no search results for a query, display "**No result found.**"

As you type the search query in the text field, an API call to the TMDB Multi Search Endpoint is made to get the list of movies and TV shows (excluding people results) for that query. This data is pushed into the RecyclerView that will populate the following information for each search card (as shown in the image):

- backdrop_image: Background image of the search card
- media_type (year): Top left of the search card
- title: Bottom left of the search card
- Rating - Top right of the search card

The search cards should get updated **every time a new letter is typed/removed** from the search query. When you click on a particular search card, it should open the Details

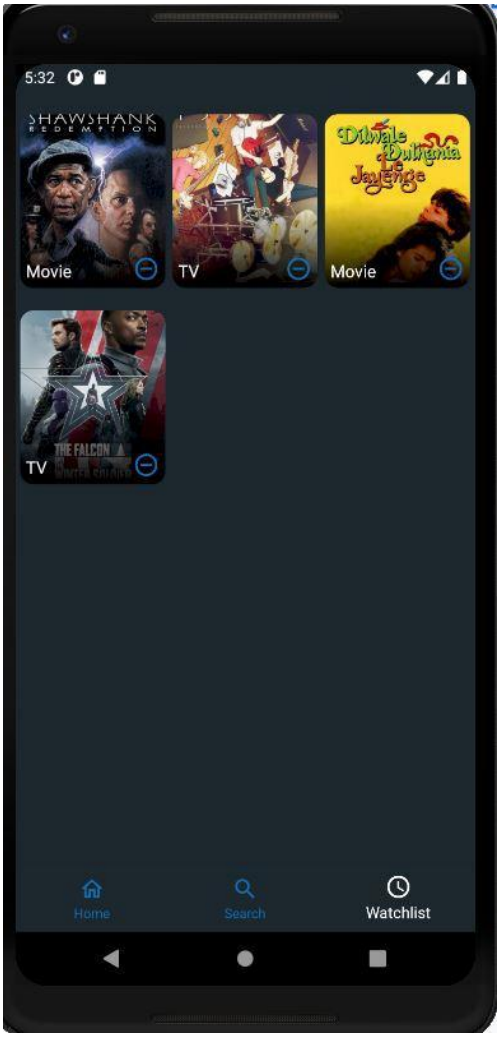
Activity for that movie or TV show. Notice the **foreground gradient** applied to the search card to make the text pop and the **shadows** on the text.

Hints to Implement Search Feature:

- RecyclerView - See Hints Section.
- SearchView:
    - How to implement SearchView in fragment without ActionBar
    - SearchView.OnQueryTextListener

5.1.3 Watchlist Fragment

The idea is to give users the ability to add a movie/TV-show to the watchlist. The watchlisted items **must** persist even after the user has closed the app. The watchlist is implemented using a RecyclerView. The RecyclerView should be displayed using the **GridLayoutManager** with **CardView** Layout. The figure below shows the watchlist fragment.



Every watchlisted card is clickable to go to the Details Activity. The **watchlisted cards must persist even after the application is closed and restarted**.

5.1.3.1 Adding to Watchlist:

The user can add the movie/TV-show to the watchlist from the Home fragment and the Details Activity.

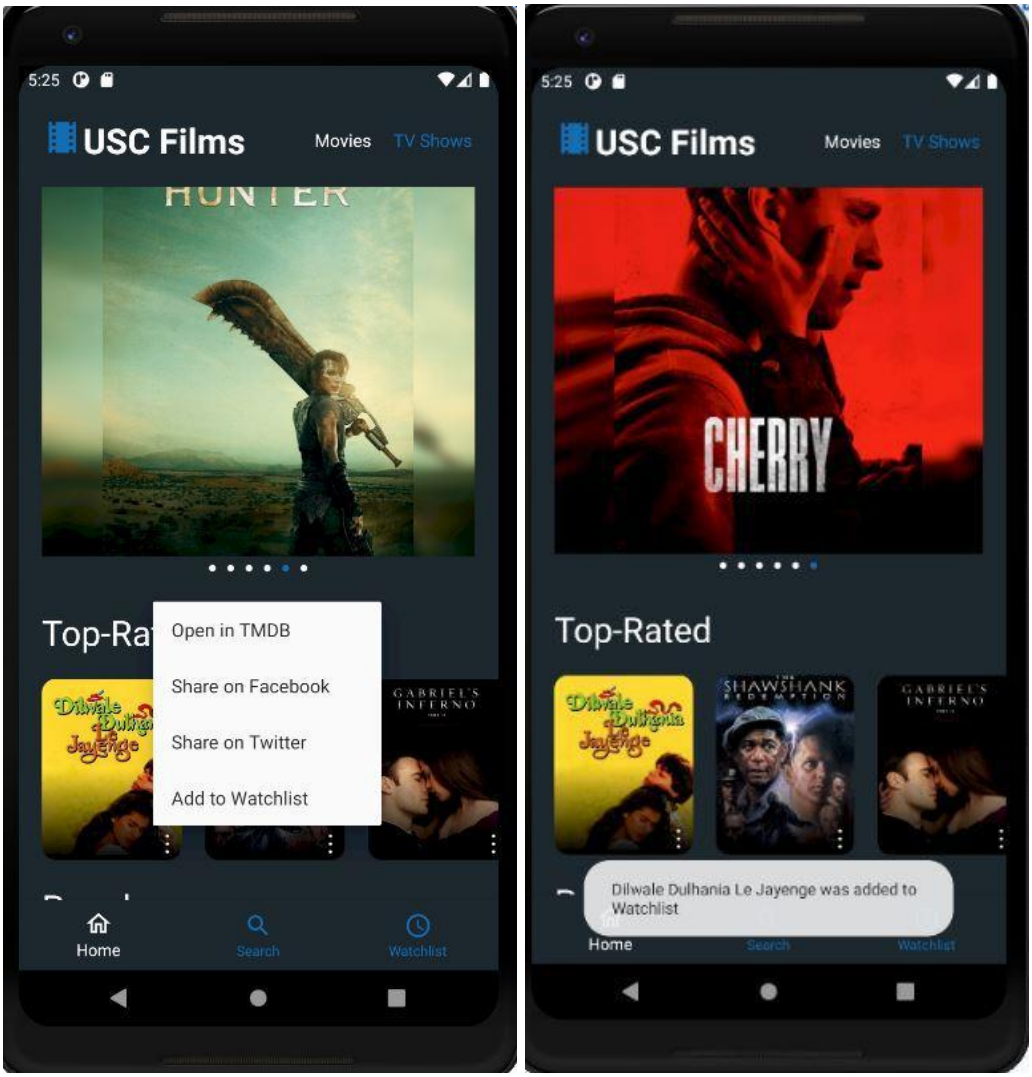


Figure:- Pop up menu showing add to watchlist option and toast showing the movie is added to watchlist.

5.1.3.2 Removing from Watchlist:

The user can remove the movie/TV-show from the Watchlist fragment and the Home fragment. Following figures show a movie being removed from the watchlist along with a Toast message.

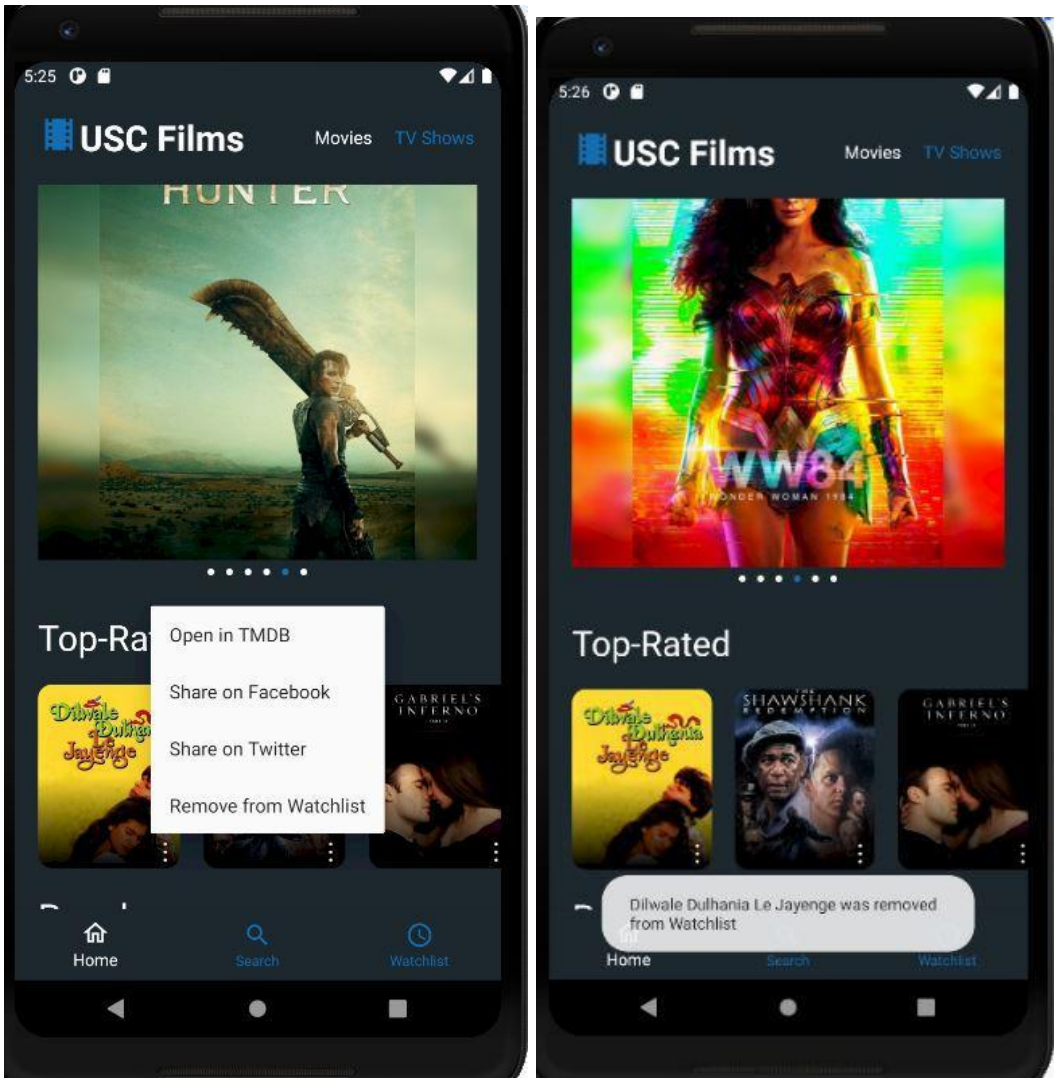
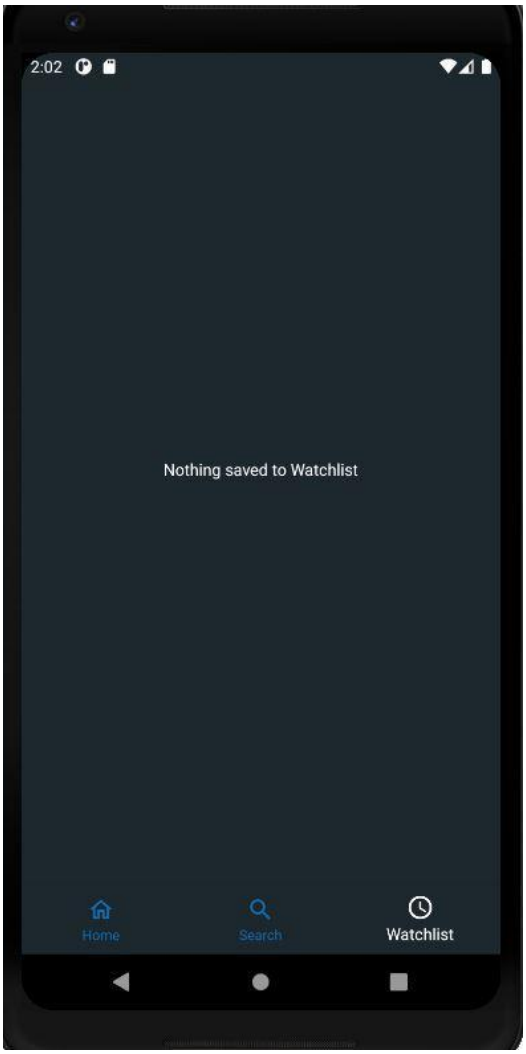


Figure:- Pop up menu showing remove from watchlist option and toast when item is successfully removed.

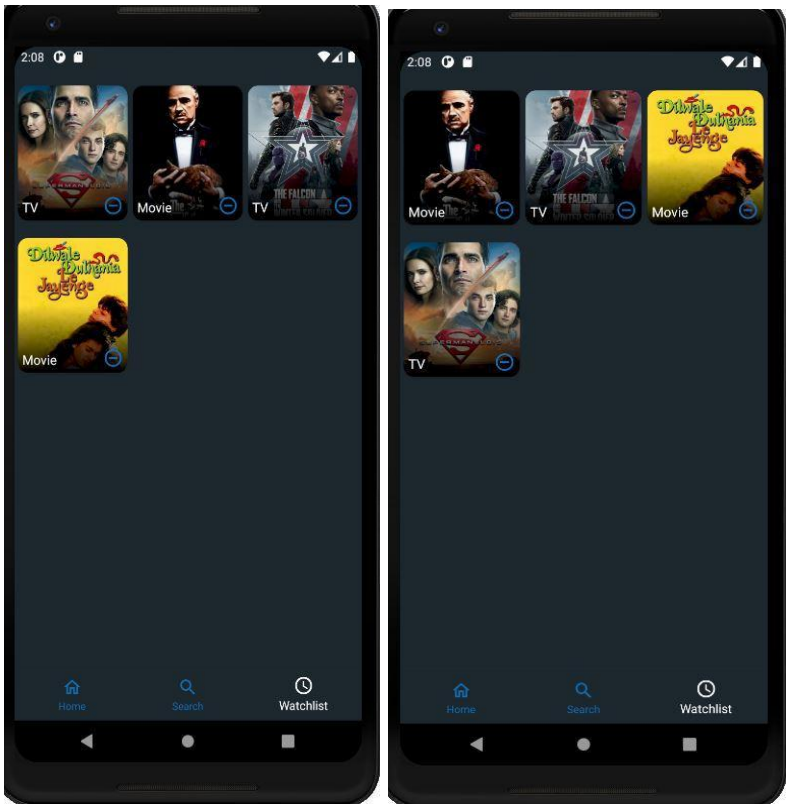
5.1.3.3 Empty Watchlist fragment:

The following figure shows an empty watchlist fragment:



5.1.3.4 Drag and drop feature:

The cards in the watchlist fragment should be draggable and droppable to reorder them.



Following figures shows the drag drop feature where any item can be placed on any other items position.

NOTES:-

1. Display **“<Movie / TV-Show Title>”** was added to / removed from watchlist using a TOAST anytime an item is either added / removed from watchlist anywhere throughout the application. HINTS are provided for implementing TOAST.
2. Make sure that the add to / remove from watchlist text in the pop menu on Home screen toggles correctly.
3. The watchlist items must persist across sessions i.e. opening and closing should retain the view. You can use Shared Preferences to do this. (like localStorage in HW8).
4. The Movie / TV label should be displayed correctly.
5. On clicking the minus button, the card should get removed immediately from the screen.
6. The Drag and Drop feature should work just like displayed in the video.

5.2 Details Activity

This Activity displays the complete details of a particular TV Show or Movie. This activity can be accessed by clicking on a particular TV Show/Movie card in the application.

This activity can be accessed in the following ways -

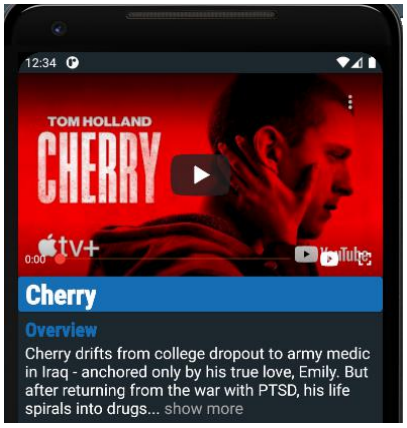
1. From the home page (Click on Image Slider, or any other movie / TV show card to open it's details.
2. From the search bar (Clicking on any of the search results)
3. From Watchlist (Clicking on a watchlist item will direct you to its respective details activity.

Movie Details

TV Show Details

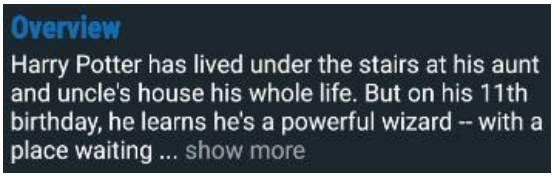
5.2.1 Components present:

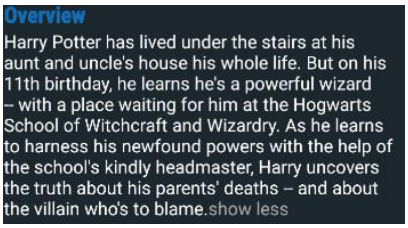
1. **Trailer**
   This contains the Youtube trailer for a selected item. This is fetched using TMDB's Get Video (For Movie, For TV Show) Endpoint. In the scenario where a trailer is not present, you should display the '**backdrop_path**' (of that movie or TV show) obtained from the details endpoint. The trailer should be able to pause and play and will require a small youtube icon which opens that trailer on Youtube.



Trailer Present Trailer absent (backdrop_image used)

2. **Movie/TV Show Name**
   Displays the name of the item, received from the details endpoint. Notice the background color. The text has been placed over a card to obtain the desired look.
3. **Overview**
   Displays the overview for a particular item. Notice the 'show more' text. Only **four** lines of the text to be displayed initially, On clicking show more, the whole overview should be displayed and on clicking show less, the original state should be achieved.





Show more Show less

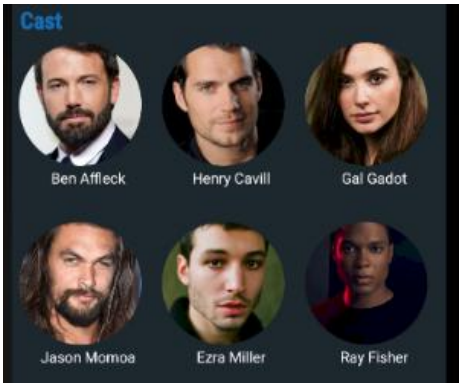
4. **Genres**
   Displays the Movie Genres obtained from the details endpoint.

5. **Share Options**
   Sharing is possible in two ways - Facebook or Twitter. On clicking these icons, a Facebook page opens up with a post ready or a Twitter page opens up with some preloaded content. (See Hints section)

6. **Add to Watchlist/Remove from Watchlist**
   By clicking on the '+' icon, the item gets added to the watchlist and the icon changes to a '-' icon, representing that the movie/TV show is already present in the watchlist. The state of the icon should be dependent on that item being in the watchlist. It should not change on revisiting the activity or restarting the app. Appropriate toasts should be displayed when added to or removed from the watchlist.

Components 5,6

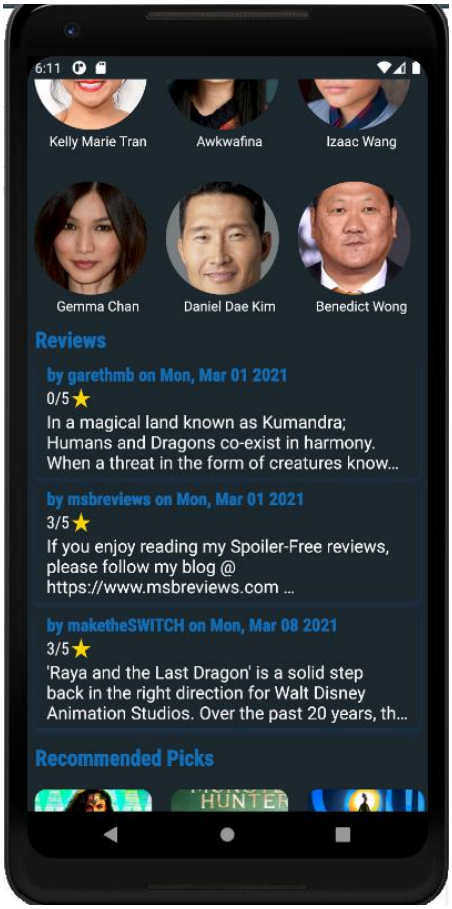
7. **Cast**
   The first **6** cast members fetched from the Credits API (For Movie, For TV Show) should be displayed here. **If there are less than 6 returned, only display those**. If there are 0 cast members returned, completely remove the cast section. The name of the cast member should be displayed right below their image.



Cast

8. **Reviews**

   First **three** reviews are displayed here, which are obtained from the Reviews API (For Movie, For TV Show). If more than 3 are returned, display the first three. If less than three are returned, display them, if 0 are returned, completely remove the review section. Each review item is a card, on which the review text (rating out of 5, a star, username, date reviewed, and the first three lines of the review content) is displayed. The card can be clicked to open the Review Activity and the complete content of the review will be visible. Notice the styling used for displaying the reviews. Make sure to use shadows and appropriate elevation attributes to obtain such a result.

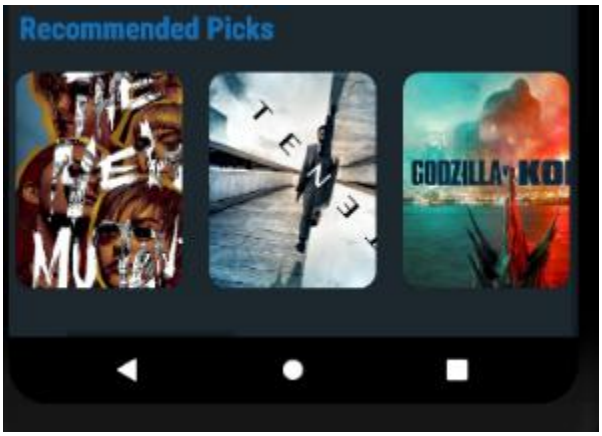


Review Cards

9. **Recommended Picks**

   This section shows TV Shows/ Movies which are similar to the one that is being viewed. These recommended picks are received from the Recommended API

(For Movie, For TV Show) and the first 10 are displayed. These cards are horizontally scrollable.



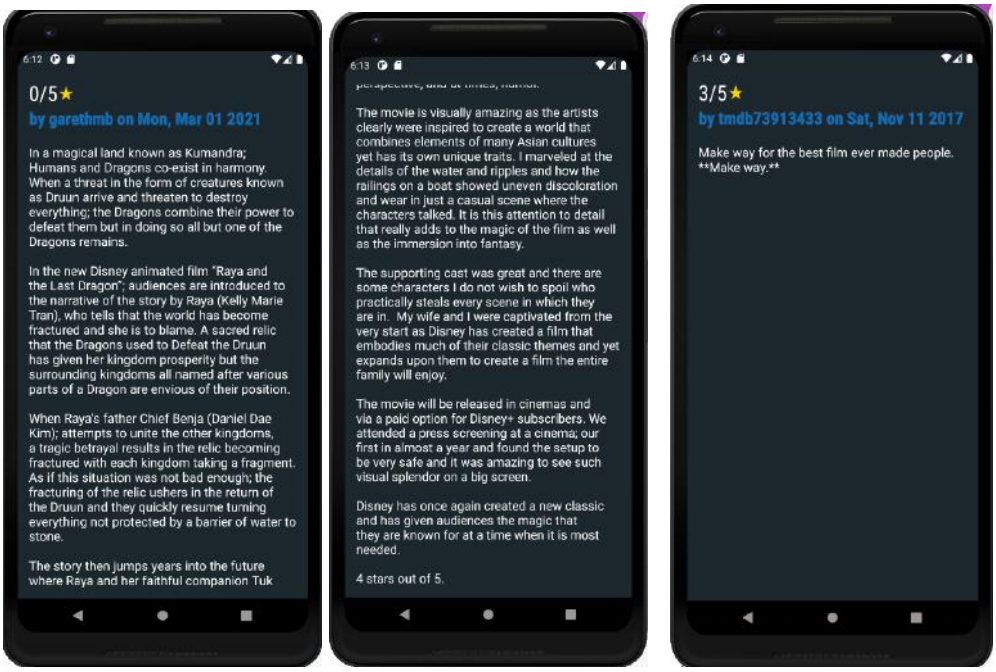
Recommended Picks

5.2.2 Endpoints Used:

1.Details Endpoint (movies, TV)
   a. trailer (video_id)
   b. backdrop_path (if trailer is absent)
   c. title
   d. overview
   e. genres
   f. release_date (for movies) / first_air_date (for tv shows)
   g. id
2.Credits Endpoint (movies, TV)
   a. image
   b. name
3.Reviews Endpoint (movies, TV)
   a. rating
   b. username
   c. content
   d. created_date
4.Recommended Endpoint (movies, TV)
   a. id
   b. poster_path

5.3 Reviews Activity

When a review card in the Details Activity is clicked, the Reviews Activity opens up. The rating out of 5, username, date created, and contents of the review is displayed as shown below. Long reviews should be scrollable.



Review Activity Review Scrolled Down Another Review

6. Implementation Hints

6.1 App icons and images:

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. The drawable folder with the xml files has been included further down in this section.

6.2 Third party libraries:

This assignment will help you gain familiarity with essential third party libraries for tasks like:

6.2.1 Volley HTTP requests

Volley can be helpful with asynchronous http request to load data. You can learn more about it here:

Volley overview

6.2.2 Picasso

Picasso is a powerful image downloading and caching library for Android. You can use the latest version of Picasso.
Picasso

6.2.3 Glide

Glide is also a powerful image downloading and caching library for Android. You can also use glide for blurring out an image in the image slider. It is similar to Picasso.
Glide v4 : Fast and efficient image loading for Android
Tutorials Glide — Custom Transformations

6.2.4 Smarteist Image Slider

Smarteist is one of the most popular third party libraries used for implementing image sliders.
Here is a tutorial on how to use this library to implement sliders: Auto Image Slider in Android with Example

6.2.5 YouTube Player

The following library is used for embedding a youtube video player inside the app.
https://github.com/PierfrancescoSoffritti/android-youtube-player

6.2.6 Show More/Show Less functionality

This library is used to show more, show less text in the overview -
https://github.com/bravoborja/ReadMoreTextView

6.2.7 Circle Image View

This library is used to show the icons in a circular view.
https://github.com/hdodenhof/CircleImageView

6.3 Pop Up Menus:

PopupMenu
Popup Menu in Android With Example

6.4 Implementing Splash Screen:

There are many ways to implement a splash screen. This blog highlights almost all of them with examples:
The (Complete) Android Splash Screen Guide | by Elvis Chidera | AndroidPub

6.5 Adding app icons:

How to create adaptive icons for Android using Android Studio

6.6 Implementing RecyclerView:

Create dynamic lists with RecyclerView
RecyclerView Android Studio | Beginner's Guide
Android RecyclerView Tutorial. ViewHolder. Handles the individual… | by Carlos Rucker
Updating data in an Android RecyclerView | by Suragch | Medium
RecyclerView Using GridLayoutManager With Example In Android Studio | Android Material Design Tutorial

6.6 Passing Variables to an Intent:

How to start an Intent by passing some parameters to it?

6.7 Opening Links in Browser:

How can I open a URL in Android's web browser from my application?

6.8 Using recyclerview inside scrollview:

8 RecyclerView inside ScrollView is not working

6.9 Working with fragments and bottom navigation bars:

BottomNavigationView with Fragments - Android Studio Tutorial
Fragments

6.10 Adding Toasts

How to display Toast in Android?

6.11 To implement Watchlist

SharedPreferences
Android Shared Preferences Example Tutorial

GridView Tutorial With Examples In Android
Next Android RecyclerView Drag and Drop

6.12 Creating Image/Vector assets:

How to create vector drawables for android?
Create app icons with Image Asset Studio
How to add vector image in android Studio ?

6.13 Adding gradient as a foreground to an image:

Add gradient to imageview

6.14 Date format

Java Code: Convert Date to String in Java

6.15 Icons used

All the icons used in the app are zipped together here as vector assets (xml files). Copy these to your res/drawable folder in android studio and they will be ready to use.

Developed by
Gayatri, Monil, Pranav, Shivani