

### Objectives:

In this assignment, you are to make use of what you have learnt in this module to design and develop a *Movie Viewer Application* on the Android platform.

### Scenario:

*PopCornMovie* is a company that allow their users to view movies details. They are looking for a mobile application that allows users to view top rated and popular. They have decided to hire you to develop this application on the Android platform.

### Deadline

Week 15 Friday - **27 Jan 2023 (Fri) 23:59**

### Weightage:

This assignment is 100 marks and constitutes **30%** of your final ICA.



**Students caught plagiarizing from other source (Internet, friends, etc..) will cause their submission to be voided.**

### Deliverables:

Listed below are the deliverables that is expected.

1. Zipped file with source codes that fulfils the Basic requirements.
2. Zipped file with source codes that fulfils the Intermediate requirements.
3. Zipped file with source codes that fulfils the Advanced requirements.

Format of file name :

*[Admin number]\_[AdvancedMaterial/Advanced/Intermediate/Basic].zip*

e.g. **22212A\_Advanced.zip**

Instructions:

- This is an INDIVIDUAL assignment. Students are to submit their assignment online
- Marks will be deducted for the following conditions.
  - Applications that cannot run upon first time installation
  - Late submissions
  - Not submitting work based on instructions given in the assignment
- The application should be able to run in the following environment
  - Emulator: [Pixel 2](#)
  - Minimal SDK: [API 27: Android 8.1](#) (Google API)
- You will be given three starter projects.

## Part 1: Basic [40 marks]

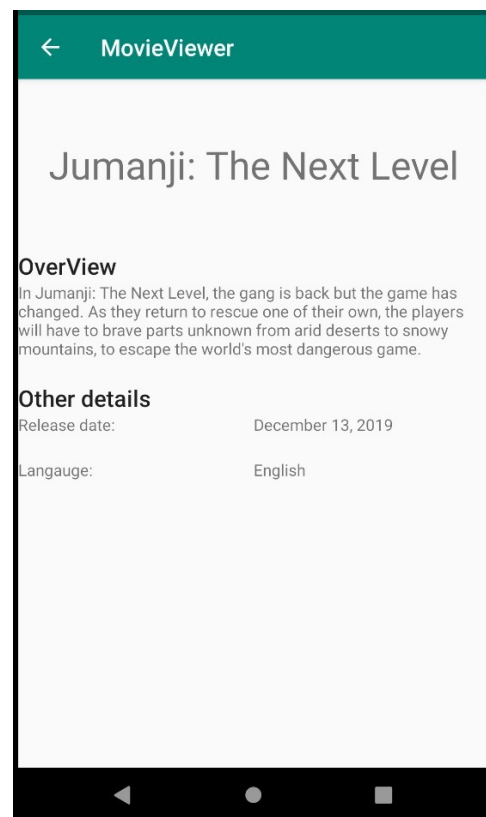
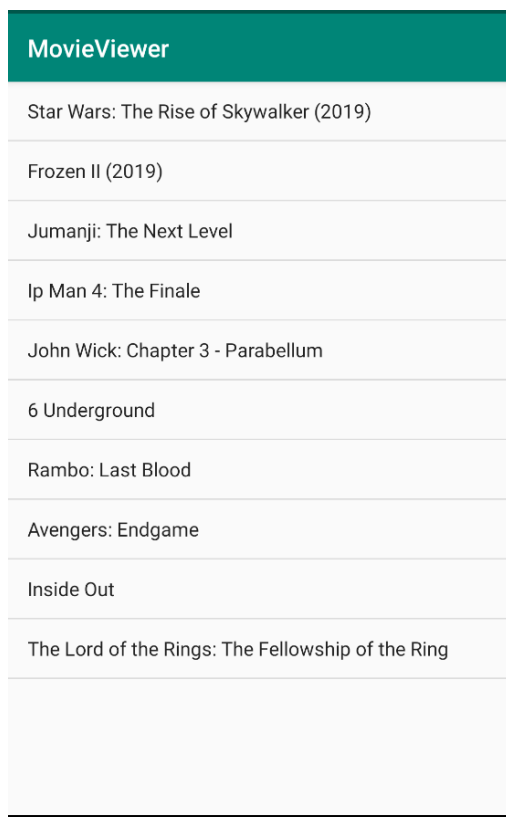
For this part, you are to make use of the **MovieViewerBasic\_Starter** project provided.

### List and detail view [10 marks]

Populate the Listview in **SimpleViewListOfMoviesActivity**.

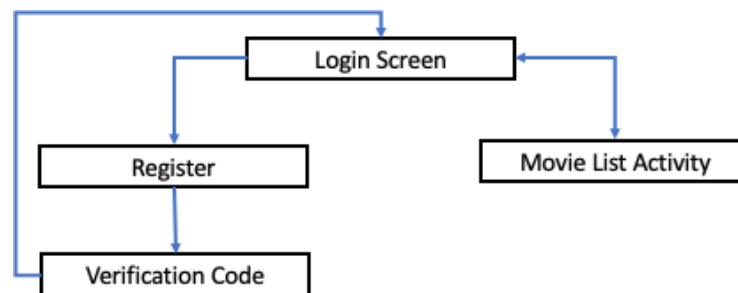
You are to make use of the sample data provided in **SimpleMovieSampleData**.

Upon tapping on a movie item, open **SimpleItemDetailActivity** and populate it with details of the selected movie.



### Login [15 marks]

Create the login sequence as stated below with the following activities:



### LoginActivity

To verify login, hardcode the correct username and password for this part of the assignment.

Login name: **testuser**

Password: **testuser**

If wrong login name or password, display a toast *"Login Error"*.

If login successfully, navigate to **ViewListOfMoviesActivity**. Alternatively, a new user can tap the *"Register"* button to navigate to **RegistrationActivity**.

### RegistrationActivity

Registration requires the following information:

- Login name
- Password
- Email
- Admin number
- Pem Group

All fields are compulsory. Perform validation by displaying error messages.

If all fields are entered, display a toast that displays all the information entered by the user before navigating to the next screen **VerificationCodeActivity**.

### VerificationCodeActivity

You can assume that verification code is a series of numbers sent to you via email.

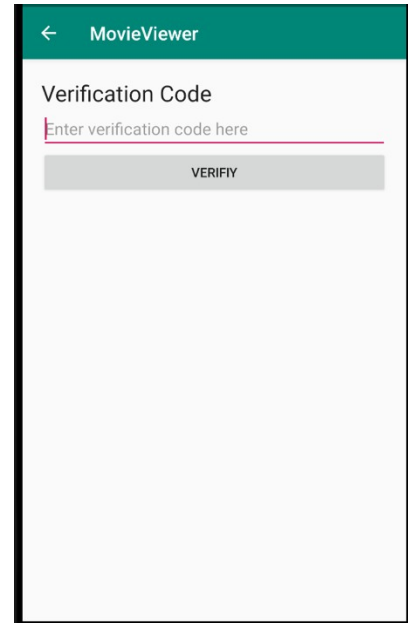
Perform validation by displaying an error message *"Code cannot be empty"*.

To verify the code, hardcode the correct code for this part of the assignment.

Verification Code: **1234**

If wrong verification code, display a toast *"Code Error"*.

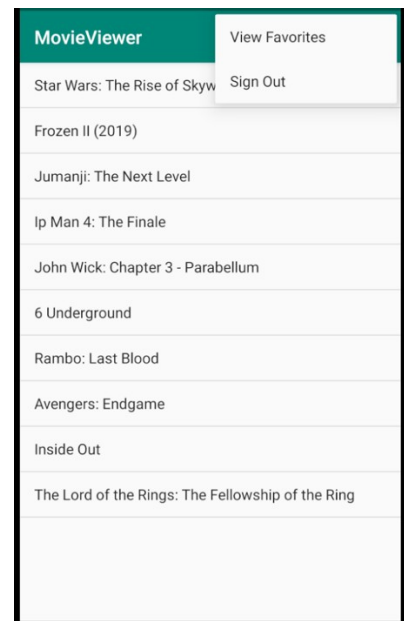
When correct code is entered, display a toast "Code verified" and navigate to **LoginActivity**.



### Sign Out

Create a *"Sign Out"* option menu item in

**SimpleViewListOfMoviesActivity**. When tapped, navigate to **LoginActivity**.

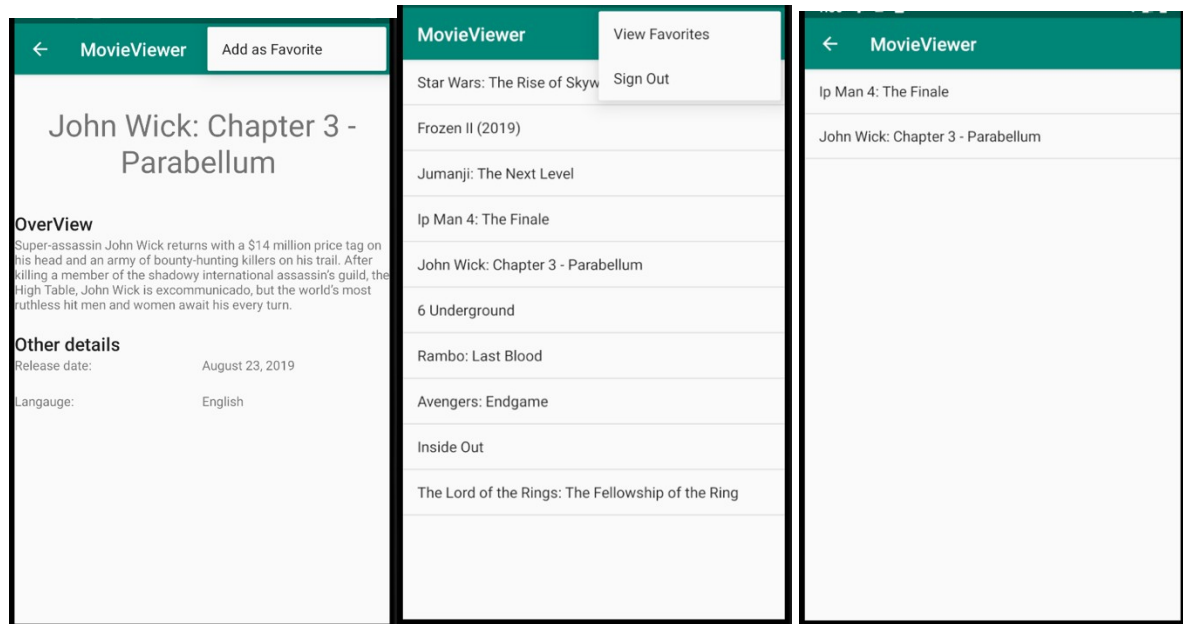


### Save Favorite [15 marks]

In **SimpleItemDetailActivity**, add an action button *"Add as Favourite"* that allows users to save the movie item as favorite in the **local SQLite database**.

In **SimpleViewListOfMoviesActivity**, add an options menu item *"View Favourites"* to navigate to a new screen **FavMoviesActivity**.

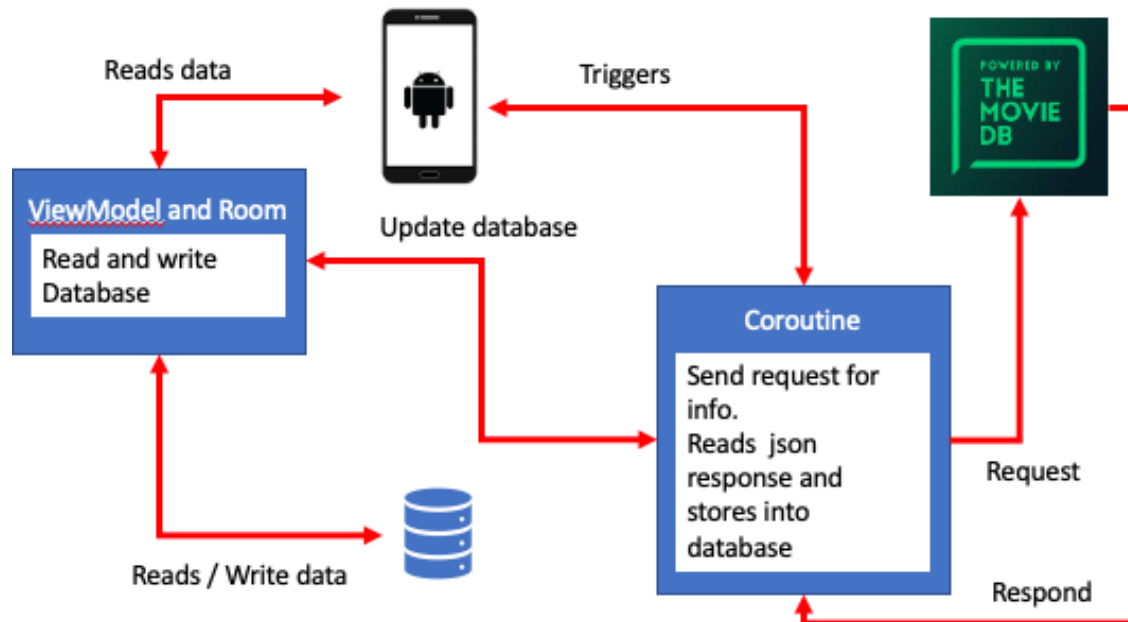
Create the new **FavMoviesActivity** that will display all the movie items that was saved as favorite in the local database. The list view will look like the **SimpleViewListOfMoviesActivity** interface.



## Part 2: Intermediate [30 marks]

For this part, you are to make use of the **MovieViewerIntermediate\_Starter** project provided for you.

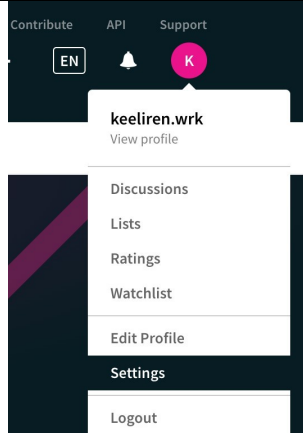
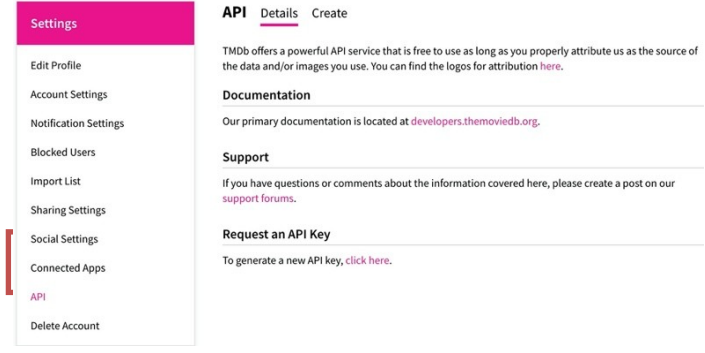
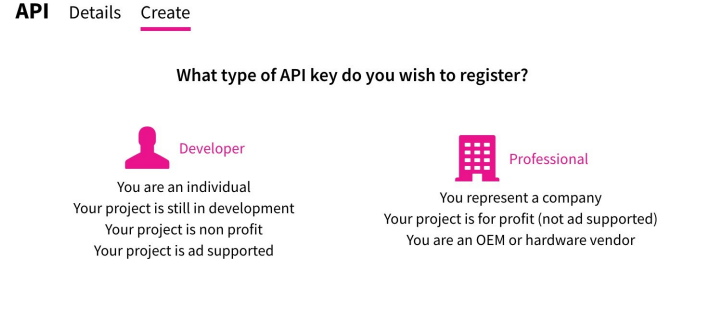
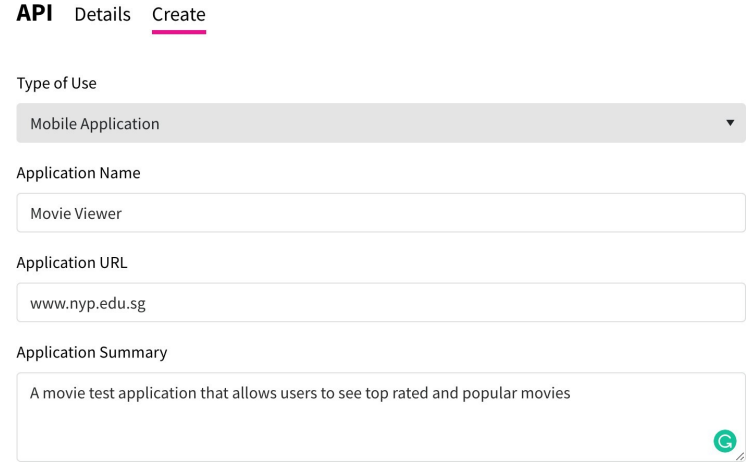
### Overview



The application that you will be modifying will request from "TheMovieDB" the information that it requires. "TheMovieDB" will response with the data in JSON format. Your app should then make use of a **coroutine** to retrieve and store the data into the database using the **view models and room**.

## API Key

Head to the following website: <http://www.themoviedb.org> and register yourself.

	<ul style="list-style-type: none"> <li>• Login and click on "Settings"</li> </ul>
	<ul style="list-style-type: none"> <li>• Click on "API"</li> <li>• Click on "click here" to request an API Key</li> </ul>
	<ul style="list-style-type: none"> <li>• Click on "Developer"</li> </ul>
	<ul style="list-style-type: none"> <li>• Fill in the following details and your other personal details.</li> </ul>



Once registered, follow the following instructions, and download the v3 Auth API key.

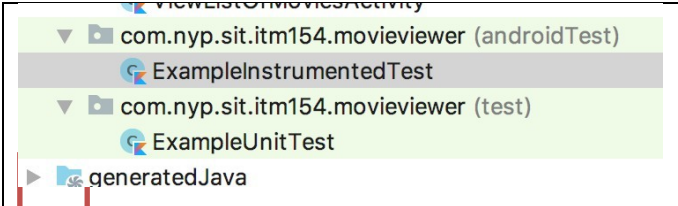
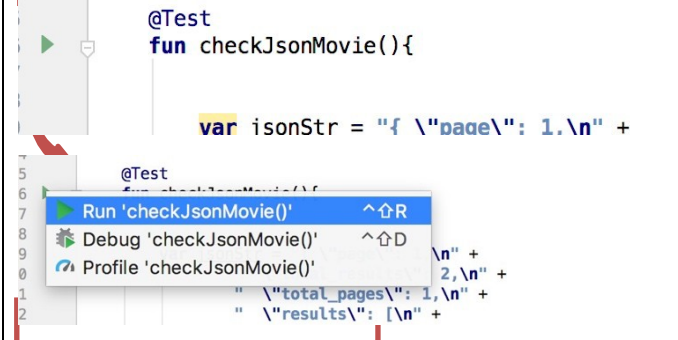
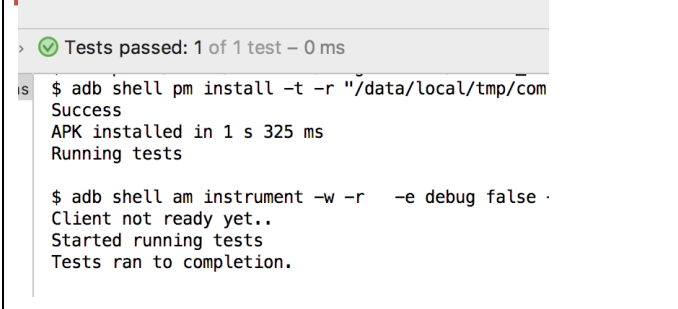
Open your project and place this key into strings.xml in your project. You should set it as the value of "moviedb\_api\_key".

```
<resources>
  <string name="app_name">MovieViewer</string>
  <string name="menu_sort_popular">Sort by Popularity</string>
  <string name="menu_sort_top_rated">Sort by Ratings</string>
  <string name="moviedb_api_key">XXXXXXXXXX</string>
</resources>
```

### JUnit Testing Json parsing [5 marks]

Before you retrieve the movies information from "TheMovieDB", update the **getMovieDetailsFromJson** method in **movieDBJsonUtils** class file to parse the downloaded json into an array of movie items.

To pass this section, you will need to run the test method.

	<p>To test this, you are to make use of the <b>ExampleInstrumentedTest</b> file. Look for it in the "androidTest" folder</p>
	<p>Run the <b>checkJsonMovie</b> test method. You are NOT allowed to change the <b>jsonStr</b> value.</p>
	<p>Make sure you pass all the test.</p>

### Data response retrieval [15 marks]

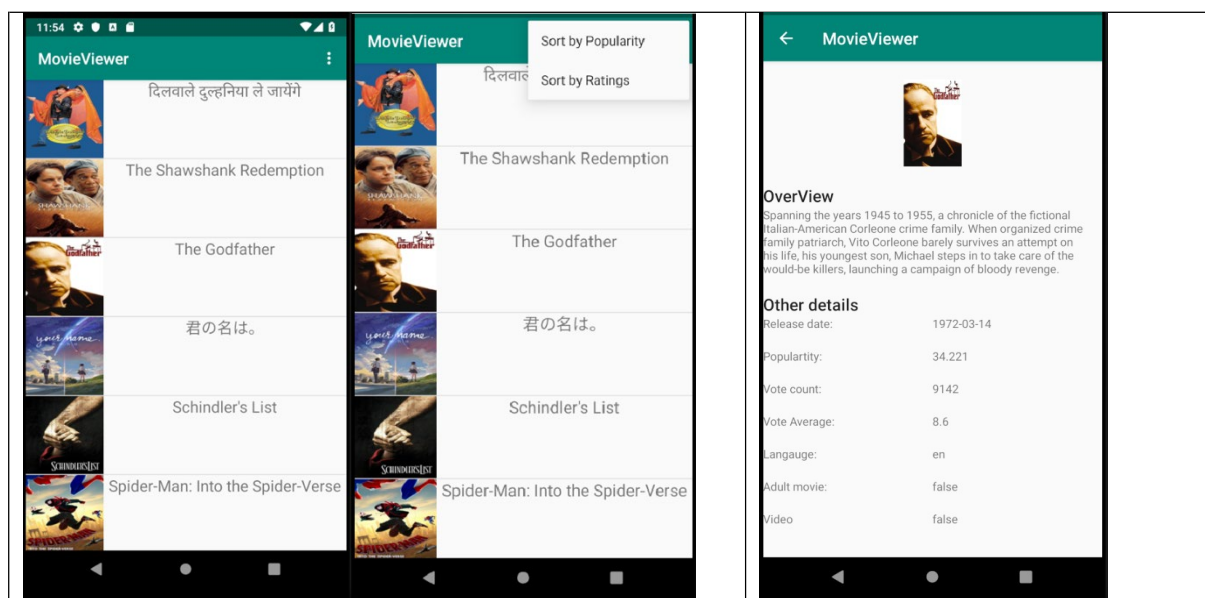
- In **ViewListOfMoviesActivity**, create and trigger a coroutine to do a HTTP request from “TheMovieDB” and store the information in the database via ViewModel and Room. The default list should be the top-rated list. You are to make use of **NetworkUtils** to do http request and getting the response.
- Update the Listview to display the list of movie items as shown below.
- User should be able to toggle between top-rated and popular list via the options menu.
- <https://developers.themoviedb.org/3/movies/get-top-rated-movies>
- <https://developers.themoviedb.org/3/movies/get-popular-movies>
- Each time the data is updated, the previous data is to be removed from the database.

Note :

- There is no need to update how favorite movie items are being saved currently. i.e. you will only need to use ViewModels and Room for data retrieved from “TheMovieDB”.

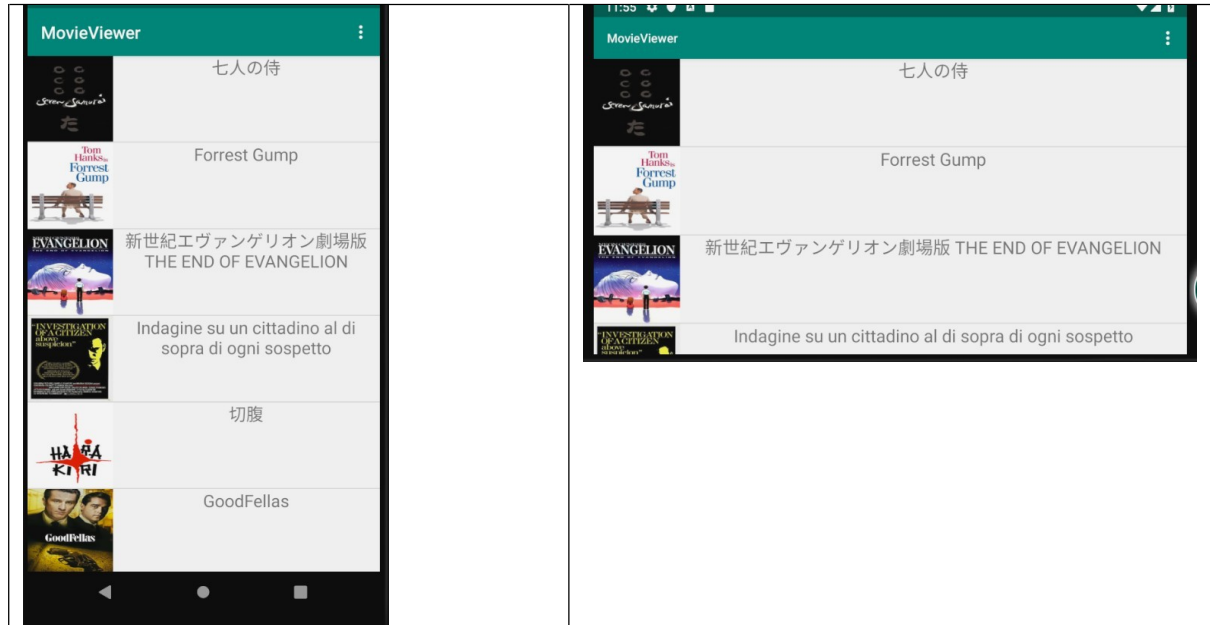
### Item retrieval [5 marks]

- Use the **Picasso API** to display all the movie images.
- Update **ItemDetailActivity** to display the details of the selected movie item.



Configuration change [5 marks]

- Currently the list will shift back to the first item each time the device's orientation changes. Update **ViewListOfMoviesActivity** to display the correct item each time the device change orientation.



### Part 3: Advanced [30 marks]

For this part, you are to make use of the **MovieViewerAdvanced\_Starter** project provided for you.

You will need to refer to implementation you have done for *Basic* and *Intermediate* to complete this section. When copying files between projects, remember to modify the package name and AndroidManifest file accordingly.

#### Cloud connectivity

The *Basic* application uses hardcoded values for authentication and local SQLite database for favorite movie items.

For *Advanced*, make use of **Amazon Web services** to perform the following:

- Registration and Login using Cognito
- Implementation of favorites and storage into DynamoDB

The starter project has been setup to connect to Amazon's Cognito and Services. You are to make use of the setup provided by the project. The app should **NOT** connect to another AWS project or account.

To make use of the DynamoDB services, you will need to authenticate the user first.

#### The Movie DB

Refer to the *Intermediate* application whereby **ViewListOfMoviesActivity** retrieve movies from <http://www.themoviedb.org>. Implement the same for *Advanced*.

### Registration and Login using Cognito [10 marks]

Refer to the login sequence in *Basic* to register and login via Cognito. There is NO need to implement Verification Code activity.

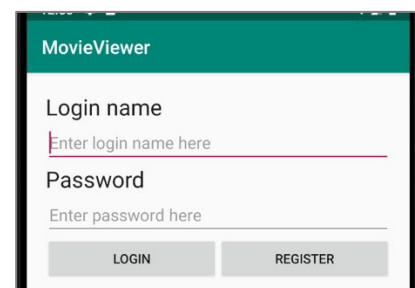
On startup, the mobile app will launch the **LoginActivity**.

The app will automatically log in user if Cognito returns that user's status as logged in and launch **ViewListOfMoviesActivity** instead.

#### LoginActivity

If login successfully to Cognito, navigate to **ViewListOfMoviesActivity**.

Alternatively, a new user can tap the *"Register"* button to navigate to **RegistrationActivity**.



### RegistrationActivity

For registration, the user is required to enter similar information as the *Basic* Application.

The **password** must confirm to the following **password policy**:

- Minimum length of **8**
- Require **number**
- Require **uppercase** letter
- Require **lowercase** letter

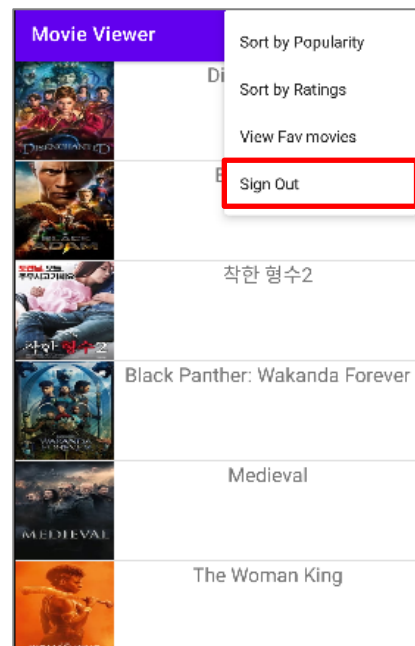
In addition to **userId** and **password**, the additional **Cognito attributes** to be stored are stated below:

- Email address -> **email**
- Admin number -> **custom:AdminNumber**
- Pem Group -> **custom:PemGrp**

Upon successful registration, navigate back to **LoginActivity**.

### Sign Out

User can choose to logout from Cognito via **ViewListOfMoviesActivity** by tapping on the *"Sign Out"* option menu item.



### Saving Favorite Movie into DynamoDB [10 marks]

Update saving of favorite movie items to save and load items from DynamoDB.

The details of the DynamoDB table are as follows:

- Table Name: **UserData**
- Each row item has 2 attributes "*id*" and "*favMovie*".

Attribute name	Attribute data type	Description
<b>id</b>	String	Stores login username to uniquely identify each row item
<b>favMovie</b>	List of <b>MovieItem</b>	Stores a list of favourite movie items of the user

- Each item in the list of favorite movie items should have all the attributes of a **MovieItem**:

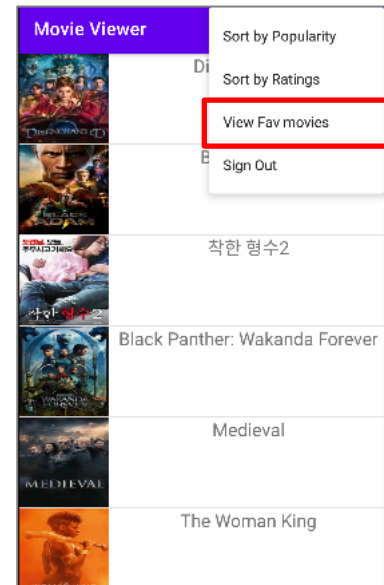
Attribute name
post_path
adult
overview
release_date
genre_ids
original_title
original_language
title
backdrop_path
popularity
vote_count
video
vote_avg

- Since each row is identified by the unique username *id*, a different favourite movie list is stored for each user.

### ViewListOfMoviesActivity

Tap on options menu item *"View Fav movies"* to navigate to **FavMoviesActivity**.

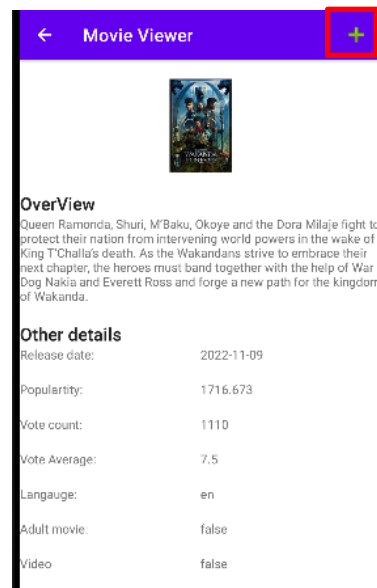
Tap on each item to navigate to **ItemDetailActivity**.



### ItemDetailActivity

Tap on options menu item *"+"* to add the movie to DynamoDB table **UserData**.

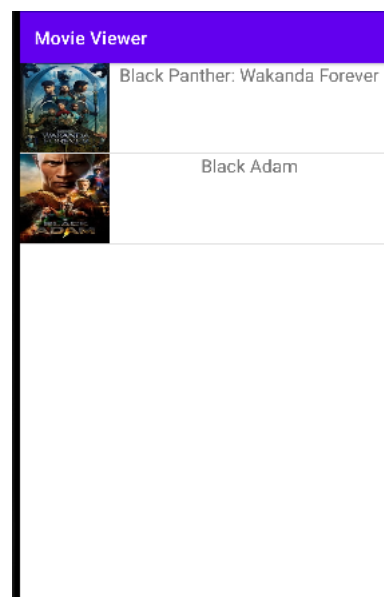
Retrieve the Cognito login username to save to the correct user data identified by *id*.



### FavMoviesActivity

Load the list of favourite movies by reading from the DynamoDB table **UserData**.

Retrieve the Cognito login username to load the correct user data identified by *id*.



### Enhanced UI [10 marks]

Update the current UI using **Material Design** components to enhance the user experience of the application.

Note :

- Marks are given based on the innovative and unique usage of different material components to enhance the user experience.
- Uniqueness is scoped within the work done by students in this module.