HW3 MovieDB part1

This homework exercises your knowledge on ViewModel and practices save instances. The task examines your ability to create and manage LiveData. The application to be implemented is a movie app that shows the now-playing and up-coming movies and allows the users to save movies and write reviews. This is the first part of the Movie DB app, in part 2, you will need to fetch the data from the https://www.themoviedb.org/ and store the reviewed movies in the database. The requirements are:

1. Load the two JSON files from the Moodle. One file has a list of now-playing movies and the other has a list of upcoming movies. These movies are the example JSON returned from the MovieDB website. The JSON files should be placed in the res/raw. You will need to use JSONObject and JSONArray to decode the raw file. A movie JSON object looks like this:

{

"popularity": 365.799,

"vote_count": 232,

"video": false,

"poster_path": "/lbGzEyESjANpOeD48aROlc3X7aa.jpg",

"id": 475557,

"adult": false,

"backdrop_path": "/pLO4qJdQxhAMPaFJu7q8bgme6R3.jpg",

"original_language": "en",

"original_title": "Joker",

"genre_ids": [

80,

18,

53

],

"title": "Joker",

"vote_average": 8.5,

"overview": "During the 1980s, a failed stand-up comedian is driven insane and turns to a life of crime and chaos in Gotham City while becoming an infamous psychopathic crime figure.",

"release_date": "2019-10-04"

}

Read these articles about how to parse JSON in Kotlin:

https://blog.mindorks.com/parsing-json-in-android

https://www.javatpoint.com/kotlin-android-json-parsing-using-url

2. The application must have four screens: a list screen showing the now-playing movie list, a list screen showing the upcoming movie list, a detailed screen showing information about the selected movie, and a review screen to write a review.

3. The now-playing and upcoming should be the same Fragment with different movie list loaded in a RecyclerView. By default, the app shows the now playing list. Note that some movies in the now-playing and upcoming movies may are the same. Each movie has a unique id. In the RecycelrView, each item must contain a small poster image, the movie name, and the genre list. See the attachment for the genre codings.

4. The user can navigate to the detail screen by clicking on the movie on either of the list screens (now playing and upcoming). The user can go back to the list screen by clicking back button or the menu items in the action bar. The detail screen must have two sections: movie information and user-added information. The movie information must contain a poster picture, movie name, date released, rating, and overview. There must be a button to navigate to the review screen.

The poster picture must be downloaded from the MovieDB API (code provided). You need to enable internet permission in Manifest file. The poster is retrieved by the "poster_path" of each movie. The URL of the poster is created by concatenating http://image.tmdb.org/t/p/w185 and the poster_path from the JSON. Use this method to load the image:

```
PosterLoader.getInstance().loadURL(m.poster_url, imgView)
```

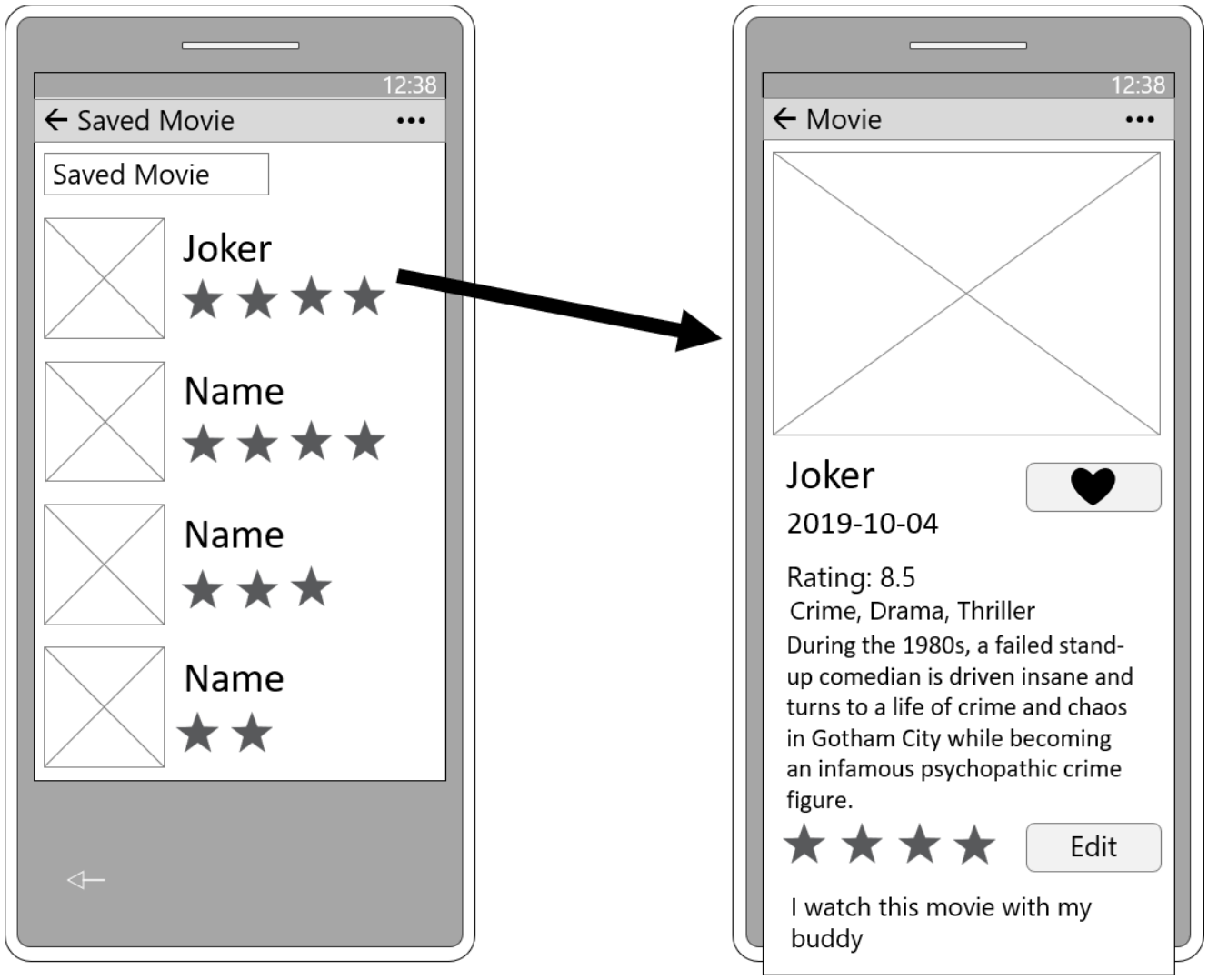
5. After clicking the "like" function, the app shows the review screen. The review screen asks the users to write a review. The review screen must contain the poster, a rating bar, a review text field, and buttons to save/cancel the review. After save, the review should show on the detail screen. The review screen must preserve the unsaved information after screen rotation or navigating to another screen.

If the save button is clicked, the detail screen shows the saved review (rating and text) in the user-added information section. Note that the review must be saved either when the user starts from the now-playing or the upcoming movie screen. Use the ScrollView and LinearLayout to make sure all information is visible in both land and port

mode: https://www.tutlane.com/tutorial/android/android-scrollview-horizontal-vertical-with-examples. Clicking the review button on the detail screen should open the review screen and allow editing. If the movie has not been edited, default information should show up on the editing screen. If the movie has been edited before, the previous information should show up.

After writing the review, if the cancel or the back button is clicked, the review screen should go back to the detail screen without showing the added-information section. But if the user goes to the review screen by clicking the back button or the "like" button, the last saved/unsaved review and rating should show up. You need to preserve all the unsaved reviews for all edited movies. Navigating to other screens or rotating the screen should not make the saved/unsaved reviews disappear.

6. The action bar should contain a button to show a list of saved movies. Clicking it shows a list of movies sorted by their ratings. Click the movie opens the corresponding detail screen. Swiping to the right must delete the saved record. The list must be preserved after screen rotation or navigating to another screen.



7. The action bar must contain menu buttons to facilitate navigation. Clicking the menu button shows menu items to go to the now playing screen, up-coming screen, and saved list screen. The action bar should also show the name of the current screen and a back button to go to the previous screen. (Alternatively, you are encouraged to use BottomNavigationView to implement the navigation UI.)

8. The navigation graph and GUI of the app must be properly designed and implemented. Provide instructions or hints by using Toast or Dialogs for improper interaction. The navigation of different Fragment should be direct and intuitive.

9. All required screens (Fragments) must be accessible from one to another. Make sure the user is not stuck at one fragment and won't be able to navigate to other screens.

10. The app must properly handle screen rotation. All views must be fully visible and interactive in both landscape and portrait modes.

11. The data, including but not limited to, the movie list, selected movie, ratings saved reviews, and unsaved reviews must be consistent throughout the app usage. You can assume the user does not press the power button or kill your app. But the data must be consistent while the user rotates the screen or temporally leave the app.

Your homework will be graded based on the following criteria:

- Required GUI and user input is properly designed and implemented (30%)
- Data is consistent throughout the navigation (30%)
- User navigation and configuration change are properly implemented (30%)
- Bug-free (10%)