HW4 MovieDB part2

This homework exercises your knowledge of using RESTful API and storing data in the local database. The task examines your ability to use Retrofit to retrieve data from an online API and store the user-modified data in a Room/SQLite database. 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 second part of the Movie DB app, 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. Go to the movieDB website and register a developer account: https://www.themoviedb.org/?language=en-US.
After login in, you can find the API key here: https://www.themoviedb.org/settings/api?language=en-US
You can find a list of API protocols here: https://developers.themoviedb.org/3/getting-started/introduction

2. For the now-playing and upcoming movies, you will need to retrieve them from the movieDB API. If the data is not coming from the movieDB API, you will lose 50 pts.
now playing: https://developers.themoviedb.org/3/movies/get-now-playing
up-coming: https://developers.themoviedb.org/3/movies/get-upcoming
You can use the "try it out" tool to see how the url and returned JSON look like.

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"

}
```

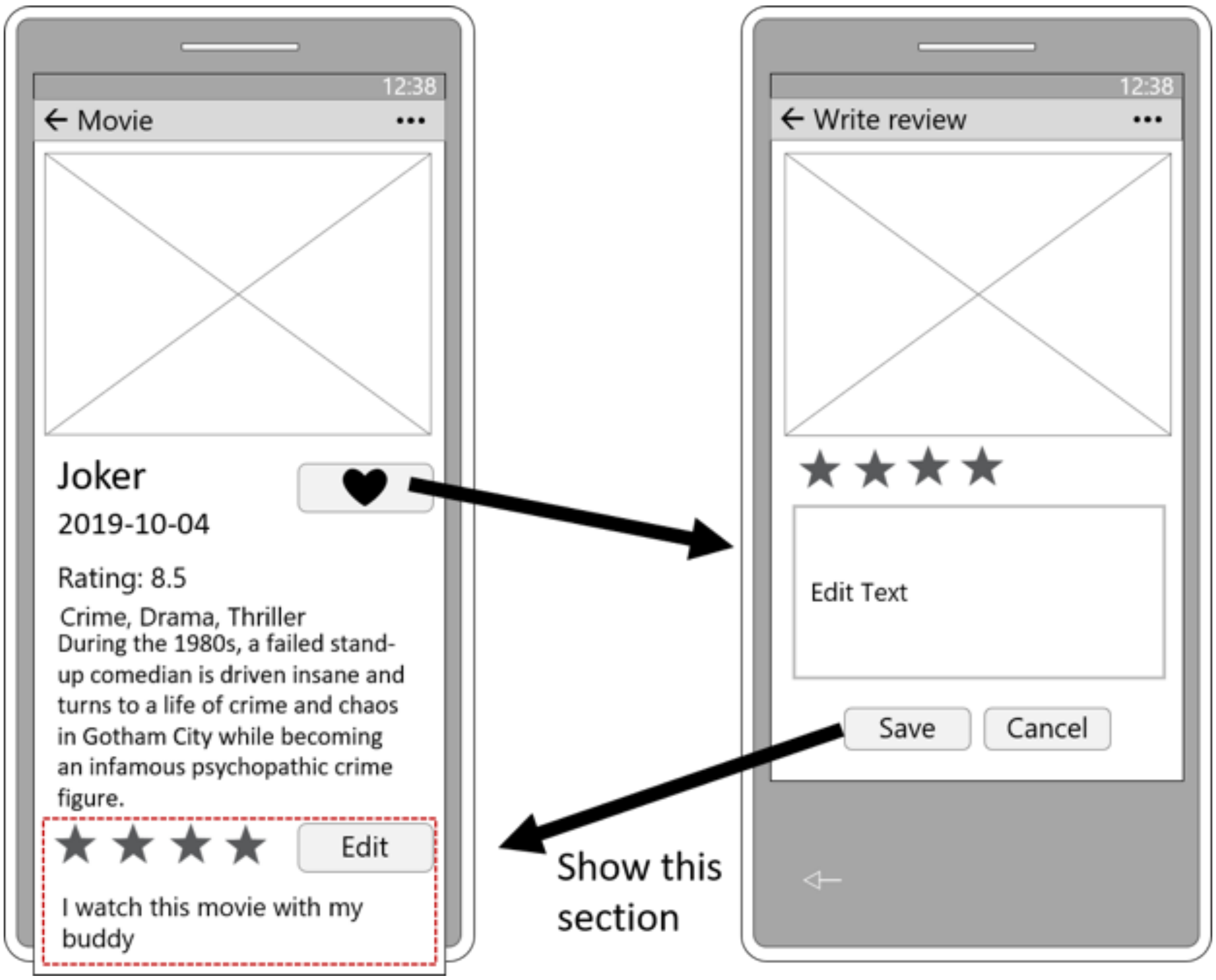
3. The UI requirements are excitedly the same as HW3. The application must have four screens: a list screen showing the now-playing movie list, a list screen showing the upcoming movie list, a detail screen showing information about the selected movie, and a review screen to write a review. The posters need to be showed on the list/detail/edit/saved screens. Refer to HW3 for what functions need to be implemented.

Clarification of the review screen:
If there was no review, the users should not see the rating bar, the edit button, or any reviews on the detail screen. If the user has reviewed before, the previously saved review should show up on the detail screen. After clicking like button, the app shows the review screen, where the user can rate the movie and write the review. During editing, the rating and texts should be preserved after rotation/leaving the app. If the cancel button is clicked. No changes should be saved. But the previously saved reviews should not be removed from the detail screen. If the user kills the app on the review screen, no changes should be saved (equivalent to clicking the cancel button). If the save button is clicked, the reviews (rating bar, edit button, review text) should show up on the detail screen. If the user clicks like/edit button, the last saved rating and reviews should show up in the review screen to support editing. If the review is removed from the saved movie list, the reviews should be removed. So the user cannot see the rating bar, review text, and the edit button on the detail screen.

Hint: to find a movie in the SQLite database by a specific id, you can use:

```
@Query("SELECT * FROM movies WHERE id=:id")
```

```
fun getMovieById(id: String): Movie
```


12:38
← Movie
Joker
2019-10-04
Rating: 8.5
Crime, Drama, Thriller
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.
Edit
I watch this movie with my buddy
Show this section
12:38
← Write review
Edit Text
Save
Cancel

4. The movie list must be refreshed every time the user opens the app or navigates to the now-playing/upcoming screens.

5. 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, navigating to another

screen/app, or killing the app.

For this assignment, all saved movies and their ratings and reviews must be stored in a Room database. Adding/updating/deleting the movie reviews should change the data in the SQLite database as well. If the user closes the app or shut down the phone, the saved movie list should still exist. Downloading the new list should not override or delete any existed saved movies. If the saved movies are not stored in the database, you lose 50 pts.

6. 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 re-editing. If the movie has not been edited, default information should show up on the review screen. If the movie has been saved before, the previous information should show up.

After writing a review, if the cancel or the back button is clicked, the review screen should go back to the detail screen without updating the reviews. But if the user goes to the review screen by clicking the "like" or "edit" button, the last saved review (default information if no review has been added before) should show up. You need to preserve all the saved reviews. Navigating to other screens, rotating the screen, or **killing the app** should not make the saved reviews disappear.

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.) Navigating to different screes should not affect the consistency of the data.

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

9. 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 uninstall the app. But the data must be consistent if the user rotates the screen, temporally leaves the app, or kills the app.

Your homework will be graded based on the following criteria:

- Required GUI and user input is properly designed and implemented (40%)
- Data is consistent throughout the navigation (40%)
- Bug-free (10%)