ITIS/ITCS 4180/5180 Mobile Application Development
Homework 4

Basic Instructions:

---

1. In every file submitted you MUST place the following comments:
    a. Assignment #.
    b. File Name.
    c. Full name of all students in your group.
2. Each group should submit only one assignment. Only the group leader is supposed to submit the assignment on behalf of all the other group members.
3. Your assignment will be graded for functional requirements and efficiency of your submitted solution. You will lose points if your code is not efficient, does unnecessary processing or blocks the UI thread.
4. Please download the support files provided with this assignment and use them when implementing your project.
5. Export your Android project and create a zip file which includes all the project folder and any required libraries.
6. Submission details: The file name should follow the following format:
    Group#_HW04.zip
7. **Failure to follow the above instructions will result in point deductions.**

Homework 4 (100 Points)

In this assignment you will develop an “IMDb App.” It is a Movie Database app where you will be able to search all the movies worldwide using the OMDB API. The app will return a list of movies based on a search term and will display information related to the movie, like Release Date, Genre, Director, etc. In this assignment, you will learn how to parse JSON responses.

Part A: Main Activity (10 Points)

Figure 1 shows the UI for the Main Activity, it contains:

1. An Image (imdb_main.jpg)
2. EditText for searching the movie name (Note: Hint will display “Find Movies” and will go away when you start typing)
3. “Search IMDb.com” button which will start **Search Movies Activity** and send the search value from above Edit Text to the next Activity. For example, tapping on search, as shown in *Figure 1*, will send the value “*Batman”* to the next Activity

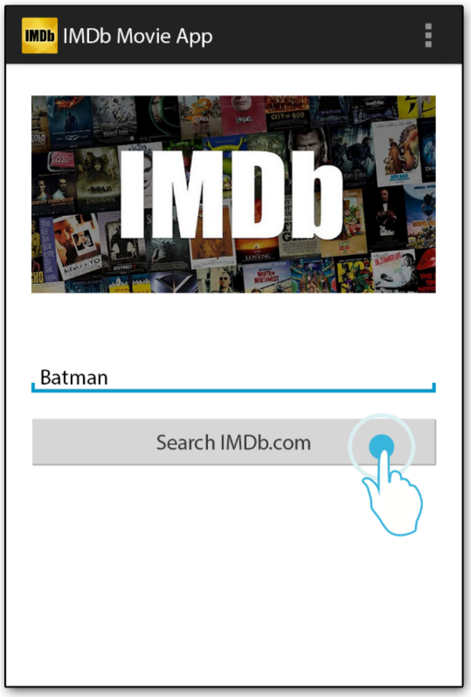


Figure 1, Main Activity

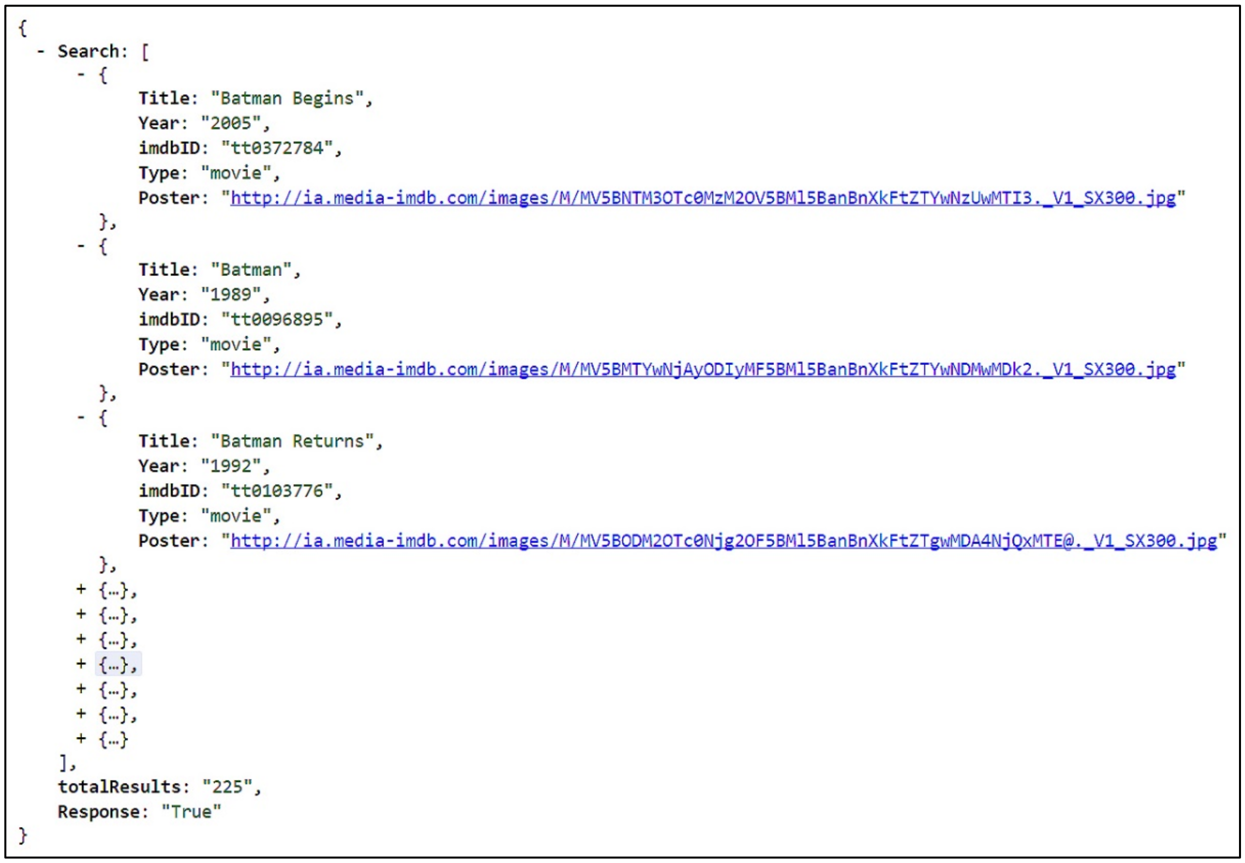
Part B: Search Movie Activity (40 Points)

This activity will get the JSON as response from OMDb API based on the movie name passed from the **Main Activity**. The requirements are as below:

1. Query the OMDb API and retrieve results. The API details is as follows:
    a. Endpoint: http://www.omdbapi.com
    b. Argument (GET Method)

    i. type: should be set to movie
    ii. s: this is the movie title to be searched for example(the value obtained from extra), Batman

2. For example, to get the movies with containing “*Batman,*” the URL should be:
   http://www.omdbapi.com/?type=movie&s=Batman



```
{
  - Search: [
      - {
            Title: "Batman Begins",
            Year: "2005",
            imdbID: "tt0372784",
            Type: "movie",
            Poster: "http://ia.media-imdb.com/images/M/MV5BNTM3OTc0MzM2OV5BMl5BanBnXkFtZTYwNzUwMTI3._V1_SX300.jpg"
        },
      - {
            Title: "Batman",
            Year: "1989",
            imdbID: "tt0096895",
            Type: "movie",
            Poster: "http://ia.media-imdb.com/images/M/MV5BMTYwNjAyODIyMF5BMl5BanBnXkFtZTYwNDMwMDk2._V1_SX300.jpg"
        },
      - {
            Title: "Batman Returns",
            Year: "1992",
            imdbID: "tt0103776",
            Type: "movie",
            Poster: "http://ia.media-imdb.com/images/M/MV5BODM2OTc0Njg2OF5BMl5BanBnXkFtZTgwMDA4NjQxMTE@._V1_SX300.jpg"
        },
      + {…},
      + {…},
      + {…},
      + {…},
      + {…},
      + {…},
      + {…}
    ],
    totalResults: "225",
    Response: "True"
}
```

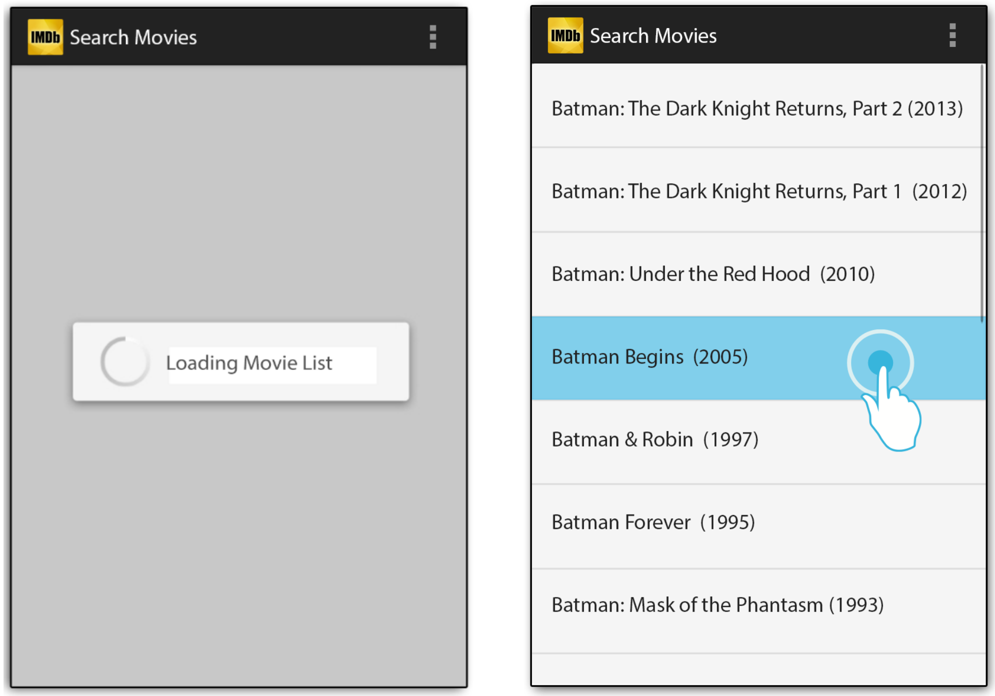
Figure 2, JSON Response

The response will be as follows:

3. The corresponding query request should be sent to the server to retrieve the JSON document stream.
4. You should use a separate thread to perform data retrieval from the server and data parsing. Do not use the Main Thread to perform these tasks. All the JSON parsing and network connections should be performed by the child thread.
5. Display a Progress Dialog during the download and/or parsing of the JSON and dismiss it once these operations are completed as indicated in *Figure 3(a)*.
6. Create a Movie class containing variables - title, year, imdbID, poster, released, genre, director, actors, plot and imdbRating.
7. Parse the JSON stream, storing retrieved information in Movie objects, and store them in a Movie list. Movies must be sorted in **Descending order of year** before being stored to the list.

a. Make sure that you store title, year, imdbID and poster for each movie and set the remaining values to null.

8. The movie items should be displayed in a list, this list should be created using a combination of ScrollView and Linear Layout, where each row is a TextView. *See Figure 3(b).* ***You should not use a ListView in this assignment.*** The goal is expose you to managing a list without using the ListView component.
9. Clicking on a movie item from the displayed List should start the **Movie Details Activity,** which provides more details about the current movie item. The entire Movie list should be sent along with the information for the specific movie clicked.



Batman: The Dark Knight Returns, Part 2 (2013)
Batman: The Dark Knight Returns, Part 1 (2012)
Batman: Under the Red Hood (2010)
Batman Begins (2005)
Batman & Robin (1997)
Batman Forever (1995)
Batman: Mask of the Phantasm (1993)

(b) Loading Movie List (a) List of Movies

Figure 3, Search Movies Activity

Part C: Movie Details Activity (35 Points)

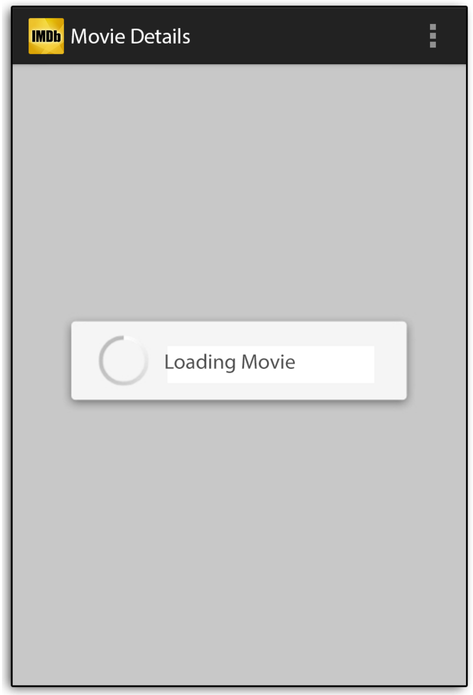
The Movie Details Activity shows details about a selected movie item.

1. Query the OMDb API to fetch all the specific movie details based on the imdbID. The API details is as follows:
   a. Endpoint: http://www.omdbapi.com
   b. Argument (GET Method)
      - i: this is the imdbID of a movie. For example: tt0372784 (for Batman Begins)
      - The URL for the above example should be: http://www.omdbapi.com/?i=tt0372784

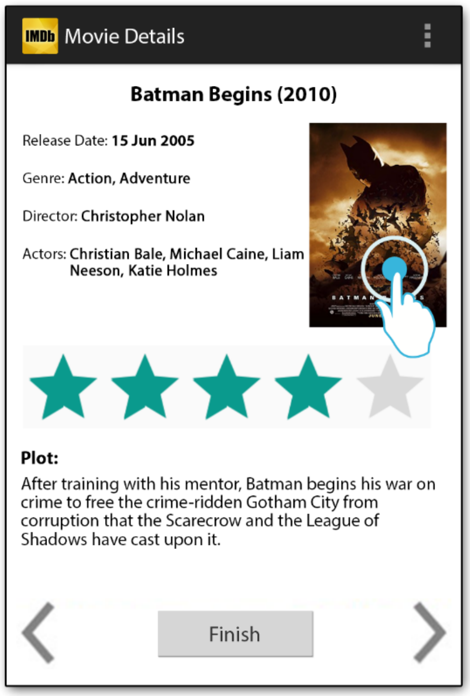
The response will be like the following:

```
{
    Title: "Batman Begins",
    Year: "2005",
    Rated: "PG-13",
    Released: "15 Jun 2005",
    Runtime: "140 min",
    Genre: "Action, Adventure",
    Director: "Christopher Nolan",
    Writer: "Bob Kane (characters), David S. Goyer (story), Christopher Nolan (screenplay), David S. Goyer
    (screenplay)",
    Actors: "Christian Bale, Michael Caine, Liam Neeson, Katie Holmes",
    Plot: "After training with his mentor, Batman begins his war on crime to free the crime-ridden Gotham
    City from corruption that the Scarecrow and the League of Shadows have cast upon it.",
    Language: "English, Urdu, Mandarin",
    Country: "USA, UK",
    Awards: "Nominated for 1 Oscar. Another 15 wins & 64 nominations.",
    Poster: "http://ia.media-imdb.com/images/M/MV5BNTM3OTc0MzM2OV5BMl5BanBnXkFtZTYwNzUwMTI3._V1_SX300.jpg",
    Metascore: "70",
    imdbRating: "8.3",
    imdbVotes: "918,818",
    imdbID: "tt0372784",
    Type: "movie",
    Response: "True"
}
```

2. Use a separate thread to perform data retrieval, display a Progress Dialog (*Figure 4(a)*) while parsing the JSON and update the Movie object with **released, genre, director, actors, plot** and **imdbRating** which were previously set to null.
3. The poster image should be retrieved using a background thread (Thread or AsyncTask).
4. The release date should be formatted as: MMM DD YYYY (Ex: JUN 15 2005)
5. Dismiss the ProgressDialog once these operations are completed.
6. The activity should be updated to display the fetched values, see Figure 4(b).
7. The actions required from the each of the GUI images are listed in *Table 1*. Upon clicking Finish, the activity should be closed and return to **SearchMovieActivity**.
8. **imdbRating** should be converted to a scale of 5 and should be displayed using RatingBar. To know more about RatingBar, please check: http://developer.android.com/reference/android/widget/RatingBar.html
9. All Movies have a URL to preview the movie details at the official IMDB website. Clicking any movie poster image, as shown in Figure 3(b)) should start **Movie Webview Activity** and send **imdbID** to this activity in order to view this URL.



(a) Loading Movie



Batman Begins (2010)

Release Date: 15 Jun 2005

Genre: Action, Adventure

Director: Christopher Nolan

Actors: Christian Bale, Michael Caine, Liam Neeson, Katie Holmes

Plot:
After training with his mentor, Batman begins his war on crime to free the crime-ridden Gotham City from corruption that the Scarecrow and the League of Shadows have cast upon it.

(b) Movie Details Loaded

Figure 4, Movie Detail Activity

Button	Action
<	Show previous Movie from the Movie List of Search Movie Activity
>	Show next Movie from the Movie List of Search Movie Activity

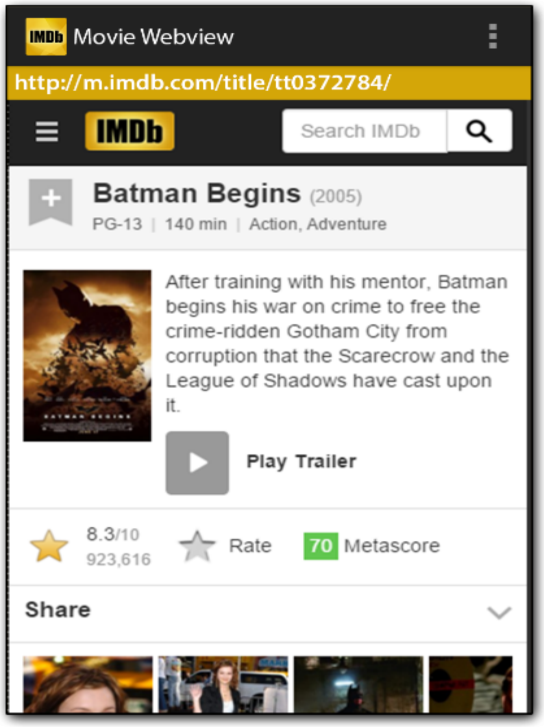
Table 1, GUI Actions

Part D: Movie Webview Activity (15 Points)
This activity should receive the "**imdbID**" information from the Movie Detail Activity. The requirement is as follows:

1. The fetched *imdbID* should be appended to http://m.imdb.com/title/

Example: http://m.imdb.com/title/tt0372784

2. The activity should display the imdb URL in an embedded WebView as indicated in *Figure 5*. The implementation requirements include:
   a. The movie URL should be displayed in the WebView, which is a simple component that is able to display a webpage in the hosting activity.
   b. You should override the WebView functionality to be able to handle redirects, check the documentation provided at http://developer.android.com/guide/webapps/webview.html. The URL should be loaded in the WebView and should handle redirects and other links in the loaded page without opening the native browser.



Movie Webview

http://m.imdb.com/title/tt0372784/

Search IMDb

Batman Begins (2005)

PG-13 | 140 min | Action, Adventure

After training with his mentor, Batman begins his war on crime to free the crime-ridden Gotham City from corruption that the Scarecrow and the League of Shadows have cast upon it.

Play Trailer

8.3/10
923,616

Rate

70 Metascore

Share

Figure 5, Webview Activity