ITIS/ITCS 4180/5180 Mobile Application Development

Midterm

In-Class Programming Assignment

Basic Instructions:

---

1. This is the Midterm Exam, which will count for 30% of the total course grade.
2. In every file submitted you MUST place the following comments:
   a. Midterm.
   b. File Name.
   c. Your Full Name.
3. This Midterm is an individual effort. Each student is responsible for her/his own Midterm and its submission.
4. Once you have picked up the exam, you may not discuss it in any way with anyone until the exam period is over.
5. This exam is composed of 3 parts, answer all of the parts.
6. Please download the support files provided with the Midterm and use them when implementing your project.
7. **Export your Android project as follows:**
   a) From eclipse, choose "Export…" from the File menu.
   b) From the Export window, choose General then File System. Click Next.
   c) Make sure that your Android project for this Midterm is selected. Make sure that all of its subfolders are also selected.
   d) Choose the location you want to save the exported project directory to. For example, your Desktop or Documents folder.
   e) The first time you export, make sure you select Create directory structure for files.
   f) Click Finish, and then go to the directory you exported the project to. Make sure the exported directory contains all necessary files, such as the .java and resource files.
8. Submission details:
   a) When you submit the Midterm, compress your exported Android project into a single zip file. The format of compressed file name is:
      i. Midterm.zip
   b) You should submit the Midterm through Moodle:
      i) Submit the zip file.
9. **Failure to do the above instructions will result in loss of points.**
10. **Any violation of the rules regarding consultation with others will not be tolerated and will result disciplinary action and failing the course.**

Midterm (100 Points)

In this assignment you will develop an app based on the Rotten Tomatoes api. The application should be able to load information about box office, in theater, opening, and upcoming movies. For this app you use JSON parsers.

Important App Requirements:

Points will be deducted if your application does not follow the below requirements:

1. The required Android Virtual Device (AVD) should have **minimum SDK version set to 14 and target SDK at least 17**. The app should display correctly on 3.2" QVGA (ADP2) (320x480: mdpi). Your assignment will not be graded if it does not meet these requirements, and you will not be granted any points on your submission.
2. All API calls, JSON parsing and image downloading should be performed using Threads (or AsyncTask) and your code should not block the main thread.
3. Your code should use standard naming conventions, such as, uppercase class names, and lower case variable/method names. Also your variable and method names should be descriptive of the data or action performed.
4. All AsyncTask/Runnable and Parsing Classes should not be implemented as inner classes in an Activity instead should be implemented as separate files.
5. Your implementation should target the most efficient algorithms and data structures. You will be graded based on the efficiency of your implementation.
6. For further information about the rotten tomatoes API check the documentation available at http://developer.rottentomatoes.com. In addition the Rotten Tomatoes api provides the following test interface http://developer.rottentomatoes.com/iodocs

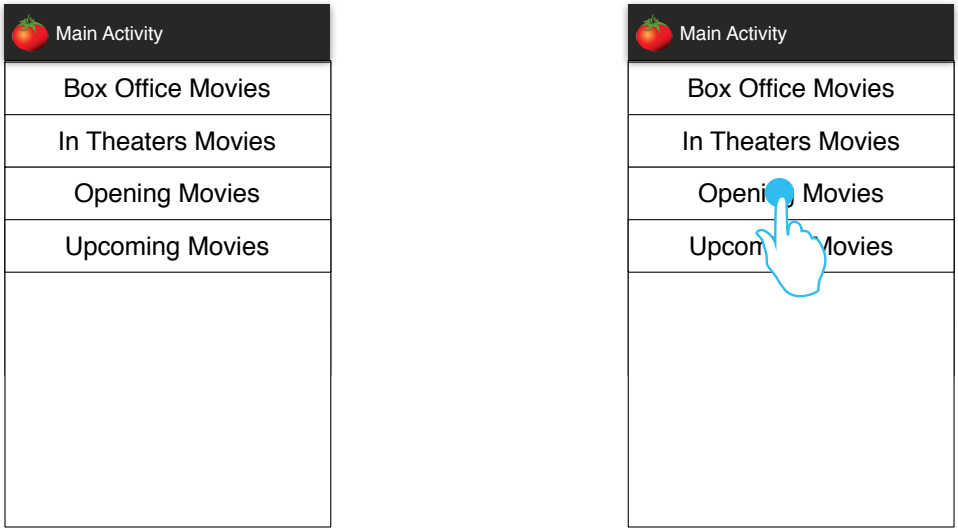
Initial Setup:

1. Go to http://developer.rottentomatoes.com/ and create a new account.
   a. As your application name use Miderm App.
   b. As your application URL use http://www.uncc.edu
2. After creating an account, you will receive an email from rotten tomatoes you need to confirm your account by clicking on the link in the received email.
3. Click on My Account (top right corner), it will display your new app and display the key which is required to be passed when access the rotten tomatoes api.
4. To access the different APIs check the documentation:
   a. Box office movies: http://developer.rottentomatoes.com/docs/read/json/v10/Box_Office_Movies
   b. In Theaters movies: http://developer.rottentomatoes.com/docs/read/json/v10/In_Theaters_Movies
   c. Opening Movies: http://developer.rottentomatoes.com/docs/read/json/v10/Opening_Movies
   d. Upcoming Movies: http://developer.rottentomatoes.com/docs/read/json/v10/Upcoming_Movies
5. In all your api calls set the **page_limit** parameter to 20.
6. All the api responses provided by rotten tomatoes is in JSON format, use the chrome inspect element console feature to inspect the returned results.

Part A: Main Activity (10 Points)

The Main activity displays a ListView with 4 options to the user, which include “Box Office Movies”, “In Theaters Movies”, “Opening Movies”, and “Upcoming Movies”. Figures 1(a) and 1(b) show the Main activity. The requirements include:

1. Clicking on any of these options should create an intent and start the Movies Activity. The intent extras should only include the selected option and send it to the Movies activity. You will not be given any points if you use a static variable to share data between activities.



(a) ListView displaying options (c) Clicking the ListView Item

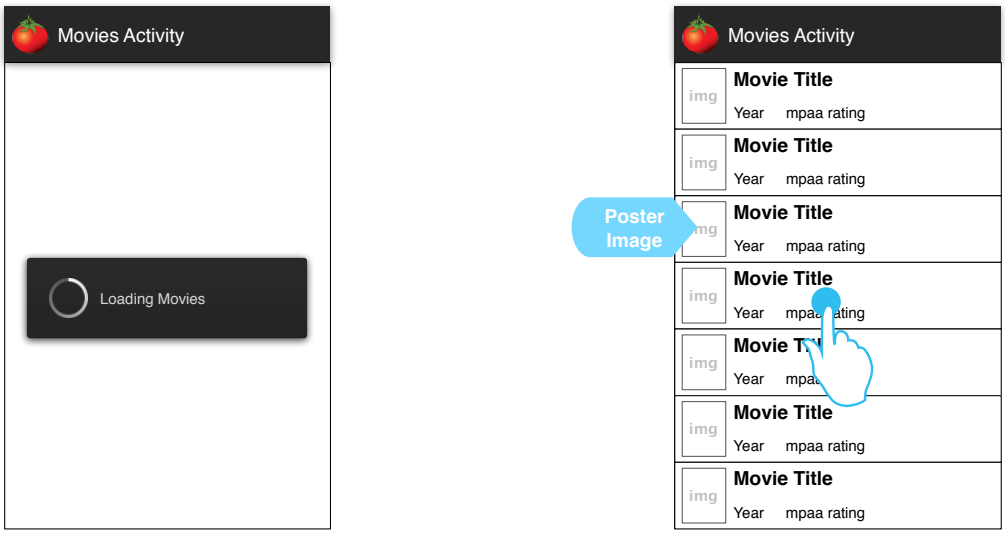
Figure 1, The Main Activity Wireframe

Part B: Movies Activity (60 Points)

The Movies activity is responsible for the loading the list of movies based on the selected option in the Main activity. The Movies activity wireframe is shown in Figure 2. The requirements include:

1) If the Main activity, the “Box Office Movies”, “In Theaters Movies”, “Opening Movies”, or “Upcoming Movies” options are selected then the corresponding JSON rottentomatoes APIs should be used. Check the API documentation (http://developer.rottentomatoes.com/docs).
2) The content of the JSON API should be retrieved by establishing a HTTP connection to the service, you should request at least **20 movies**. All the parsing and HTTP connections should be performed by a worker thread or an AsyncTask and should not be performed by the main thread. While, the movies JSON is being loaded you should indicate that using the progress dialog as indicated in Figure 2(a).
3) The progress dialog should be dismissed after the parsing is completed and the ListView is setup and displayed.

4) The retrieved movies should be displayed in a ListView. The ListView items should be displayed to include the movie poster thumbnail, title, year, and MPAA rating, as shown in Figure 2(b).
5) You should create your custom adapter and a custom layout for the ListView items. You should reuse the views provided by the ListView adapter from the (scarp) recycle pool and should provide the synchronization needed to ensure the loading of the correct thumbnails.
6) The images should be retrieved by the adapter using a separate thread or AsyncTask. The images should be downloaded only when needed, your implementation should not pre-download all the images. At any given time only maintain a reference to the displayed photo. Do not store the downloaded images in memory as this is not efficient and will crash your application.
7) Clicking a movie item should display the movie details by starting the Movie activity. The intent should include the required movie information to be sent to the Movie activity using extras. You will not be given any points if you use a static variable to share data between activities.



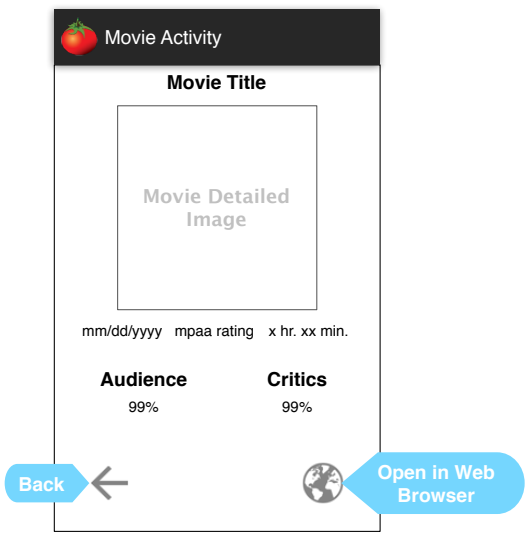
(a) Movies Activity Loading Movies (b) ListView showing the retrieved movies

Figure 2, The Movies Activity Wireframe

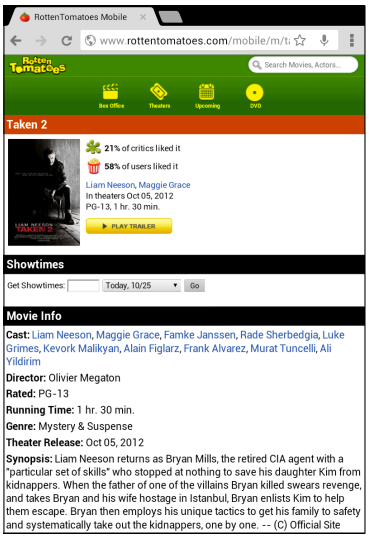
Part C: Movie Activity (30 Points)

The Movie activity will display more details related to the movie selected in the Movies activity. The Movie activity wireframe is shown in Figure 3(a). The requirements include:

1) The movie details displayed include, movie title, movie detailed poster image, movie release date, mpaa rating, movie runtime, audience rating, and critiques rating.
2) Loading the movie detailed image should not be performed by the main thread, you should use another thread or AsyncTask. If no poster is found for the displayed movie use the provided "poster_not_found.png" image. The images should be downloaded only when needed, your implementation should not pre-download all the images. At any given time only maintain a reference to the displayed photo. Do not store the downloaded images in memory as this is not efficient and will crash your application.
3) Upon clicking the back icon (left arrow icon), the Movie activity should exit.
4) Upon clicking on the globe icon, the movie website should be loaded in the mobile's web browser, as indicated in Figure 3(b).



(a) Movie Activity



(b) Movie Website

Figure 3, The Movie Activity Wireframe