ITIS/ITCS 4180/5180 Mobile Application Development

Midterm

Date Posted: 10/24/2013 at 19:30

Due Date: 10/31/2013 at 23:55

Basic Instructions:

1. This is the Midterm Exam, which will count for 20% 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. Once you have picked up the exam, there is no limit as to how much time you may spend working on it, until the return date/time mentioned above.
6. A 10% penalty will be subtract from the midterm's score for each late day. No credit will be given for a midterm turned in more than 7 days past the due date.
7. This exam is composed of 6 parts, answer all of the parts.
8. If any problems arise during this exam, email me (mshehab (at) uncc.edu).
9. Please download the support files provided with the Midterm and use them when implementing your project.
10. **<u>Export your Android project as follows:</u>**
    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.
11. 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.
12. **Any violation of the rules regarding consultation with others will not be tolerated and will result disciplinary action and failing the course. Failure to do the above instructions will result in loss of points.**

Midterm (100 Points)

In this assignment you will develop a simple Rotten Tomatoes client application. The application should be able to load information about box office, in theater, opening, and upcoming movies. Users of the application will be able to add/remove movies to their favorites. The favorites should be stored on the server using the provided favorites API. Finally, you will present the favorite statistics using the Google Charts API in a WebView. For this homework you will be using both JSON and XML parsers.

Notes:

1. The recommended Android Virtual Device (AVD) should have minimum SDK version set to 11 and target SDK at least 17. The app should display correctly on 3.2" QVGA (ADP2) (320x480: mdpi).
2. All API calls, JSON or XML parsing and image downloading should be performed using Threads (or AsyncTask) and your code should not block the main thread.
3. 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
4. For information about the Google Charts API specifically the Bar Chart Visualization go to https://google-developers.appspot.com/chart/interactive/docs/gallery/barchart

Initial Setup and Favorites API Description

- You are provided with a java class "**Config.java**", import it to your project.
- The "Config" Class provided contains a method "String getUid(String username)", this method is used to retrieve the user id (uid). The username is your UNCC user name, please the username listed on Moodle. If your email listed on moodle is "bsmith@uncc.edu" then your username is "bsmith". To retrieve the "uid" you should call the method Config.getUid("bsmith").
- You are provided with a test website which includes a form that will allow you to test these APIs, http://cci-webdev.uncc.edu/~mshehab/api-rest/favorites/

addToFavorites API:

- *URL*: http://cci-webdev.uncc.edu/~mshehab/api-rest/favorites/addToFavorites.php
- *Arguments*: (**POST method**)
  - *uid* : the uid value returned by calling "Config.getUid(username)", where the username is retrieved from the Shared Preferences.
  - *mid* : the movie id to be added to the user favorites.
- *Description*: This API will add a movie to the user's favorites on the server. Failing to pass the required parameters will result in error response from the server. If all parameters are passed successfully, the server will return a response that includes the an error message, and the user's id. A successful response is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <error>
      <id>0</id>
      <message>Done adding favorite 12452094382</message>
   </error>
   <user>
      <id>444ac1a0bbbfa1cf258b1eeeeb71ff420</id>
   </user>
</response>
```

getAllFavorites API:

- *URL*: http://cci-webdev.uncc.edu/~mshehab/api-rest/favorites/getAllFavorites.php
- *Arguments*: (**POST method**)
  - *uid* : the uid value returned by calling "Config.getUid(username)", where the username is retrieved from the Shared Preferences.
- *Description*: This API will retrieve all the favorites stored on the server for the specified user id. The "uid" parameter should be passed to the API to enable the server to identify the user. Failing to pass the required parameters properly will result in error response from the server. The root of the XML returned is a **response** element. The first child of the root element is an **error** element, which will includes an error id and message. The error elements will enable you to identify if any errors have occurred. Each favorite element will have an "id" element, which represents the id of the movie stored in favorites. Below is an example response XML returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <error>
      <id>0</id>
      <message>Done getting all favorites</message>
   </error>
   <user>
      <id>444ac1a0bbbfa1cf258b1eeeeb71ff420</id>
   </user>
   <favorites>
      <favorite>
         <id>111111111</id>
      </favorite>
      <favorite>
         <id>22222222</id>
      </favorite>
   </favorites>
</response>
```

deleteFavorite API:

- *URL*: http://cci-webdev.uncc.edu/~mshehab/api-rest/favorites/deleteFavorite.php
- *Arguments*: (**POST method**)
  - *uid* : the uid value returned by calling "Config.getUid(username)", where the username is retrieved from the Shared Preferences.
  - *mid* : the movie id to be deleted from the user favorites.
- *Description*: This API will delete a favorite movie from the user's stored favorites. Failing to pass the required parameters will result in error response from the server. A successful response is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <error>
      <id>0</id>
      <message>Done deleting favorite 12452094382</message>
   </error>
   <user>
      <id>444ac1a0bbbfa1cf258b1eeeeb71ff420</id>
   </user>
</response>
```

deleteAllFavorites API:

- *URL*: http://cci-webdev.uncc.edu/~mshehab/api-rest/favorites/deleteAllFavorites.php
- *Arguments*: (**POST method**)
  - *uid* : the uid value returned by calling "Config.getUid(username)", where the username is retrieved from the Shared Preferences.
- *Description*: This API will delete all the user's favorites stored on the server. Failing to pass the required parameters will result in error response from the server. A successful response is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <error>
      <id>0</id>
      <message>Done deleting all your favorites</message>
   </error>
   <user>
      <id>444ac1a0bbbfa1cf258b1eeeeb71ff420</id>
   </user>
</response>
```

isFavorite API:

- *URL*: http://cci-webdev.uncc.edu/~mshehab/api-rest/favorites/isFavorite.php
- *Arguments*: (**POST method**)
  - *uid* : the uid value returned by calling "Config.getUid(username)", where the username is retrieved from the Shared Preferences.
  - *mid* : the movie id to be checked in favorites.
- *Description*: This API will check if the provided movie id is present in the user's favorites that stored on the server. The result will include an isFavorite element that will be set to true or false. Failing to pass the required parameters will result in error response from the server. A successful response is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <error>
      <id>0</id>
      <message>Done checking is favorite 12452094382</message>
   </error>
   <user>
      <id>444ac1a0bbbfa1cf258b1eeeeb71ff420</id>
   </user>
   <favorites>
      <favorite>
         <id>123</id>
         <isFavorite>true</isFavorite>
      </favorite>
   </favorites>
</response>
```

getFavoriteStats API:

- *URL*: http://cci-webdev.uncc.edu/~mshehab/api-rest/favorites/getFavoriteStats.php
- *Arguments*: (**POST method**)
  - *uid* : the uid value returned by calling "Config.getUid(username)", where the username is retrieved from the Shared Preferences.
- Description: This API will return the top 5 favorite movie id's and the favorite count for each of the movie. Failing to pass the required parameters will result in error response from the server. A successful response is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
    <error>
        <id>0</id>
        <message>Done getting favorite stats</message>
    </error>
    <user>
        <id>444ac1a0bbbfa1cf258b1eeeeb71ff420</id>
    </user>
    <favorites>
        <favorite>
            <id>770785616</id>
            <count>2</count>
        </favorite>
        <favorite>
            <id>771238418</id>
            <count>5</count>
        </favorite>
        <favorite>
            <id>770678819</id>
            <count>3</count>
        </favorite>
        <favorite>
            <id>771303549</id>
            <count>7</count>
        </favorite>
        <favorite>
            <id>771235120</id>
            <count>20</count>
        </favorite>
    </favorites>
</response>
```

Errors:

- Each error has an associated id and an error message explaining the error. If an error is detected you should show a Toast message displaying the corresponding error message. Not that, an error id of 0, means no errors have occurred.

Important Programming Requirements:

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

1) All network connections, JSON/XML parsing, and image downloading should be performed using Threads (or AsyncTask) and should not block the main thread.
2) 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.
3) All AsyncTask/Runnable and Parsing Classes should not be implemented as inner classes in an Activity instead should be implemented as separate files.
4) You should not share any data between activities using static variables, and instead should use intent extras and/or starting activities for results.

5) Your implementation should target the most efficient algorithms and data structures. You will be graded based on the efficiency of your implementation.
6) In this app you will use the Android Shared Preferences to store the username locally. For information go to http://developer.android.com/guide/topics/data/data-storage.html#pref

Part A: Main Activity (10 Points)

The main activity displays a ListView with options to the user, which include “My Favorite Movies”, “Box Office Movies”, “In Theaters Movies”, “Opening Movies”, “Upcoming Movies”, and “Favorite Statistics”, see Figure 1(a). Tapping on “My Favorite Movies”, “Box Office Movies”, “In Theaters Movies”, “Opening Movies”, and “Upcoming Movies” options should start the Movies Activity. Tapping the “Favorite Statistics” should start the Statistics Activity. Also note that clicking the options menu should show the “Clear All Favorites”, “Setup Username”, and “Exit” options, see Figure 1(b). Clicking the “Clear All Favorites” should delete all favorites stored on the server, clicking the “Setup Username” should start the Username Activity and clicking the “Exit” option should exit the application. The Main Activity requirements include:

1) When the Main Activity starts it should first check the Shared Preferences for a stored username, if there is no stored username then the Username Activity should be started by the Main Activity to set a username. Upon returning from the Username Activity the Main Activity should recheck the Shared Preferences for a stored username, and if there is no stored username then the Username Activity should be started by the Main activity to set a username.

(a) Main Activity Starting (b) The menu options (c) The Username Activity

Figure 1, The Main Activity and Username Activity Wireframe

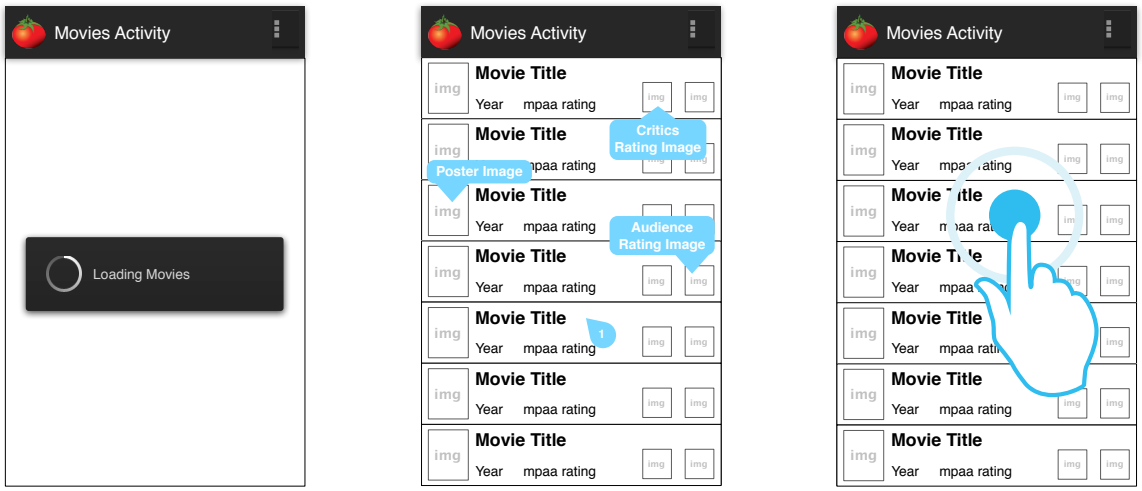
Part B: Username Activity (5 Points)

The Username activity is responsible for helping the user set and store their username in the Shared Preferences.

1) Upon loading the Username activity the Shared Preferences should be checked to retrieve the stored username, if there is a username stored then it should be

displayed in the username EditText, if there is no username previously stored in the Shared Preferences then the EditText should be cleared.

2) Upon clicking the “Set Username” button, the username entered in the EditText should be stored in the Shared Preferences, and the Username Activity should finish. Your code should check the username input to ensure that it is not an empty string, and should display an error message to the user indicating that an error string is not allowed, and request from the user to re-enter the username.



(a) Movies Activity Loading Movies

(b) ListView showing the different movies

(c) Long Press removes a favorite movie

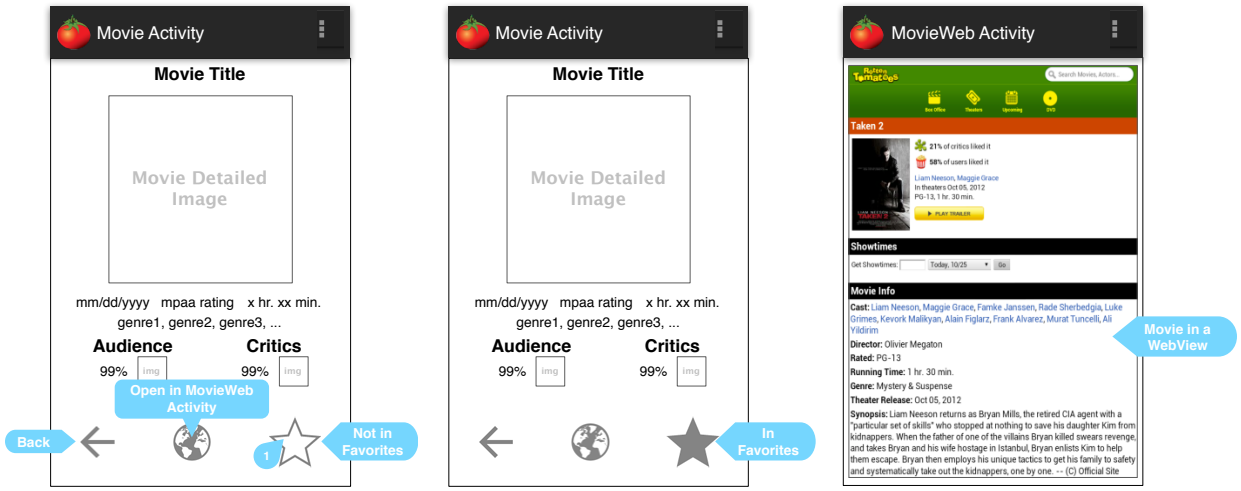
Figure 2, The Movies Activity Wireframe

Part C: Movies Activity (40 Points)

The Movies activity is responsible for the loading the list of movies based on the selected option in the Main activity.

1) If “My Favorite Movies” was selected then the movies list should be loaded by using the favorites API to retrieve the ID’s of the favorite movies. Then the Rotten Tomatoes API should be contacted to retrieve the details of retrieved movie ID’s. If 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 **50 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/XML is being loaded you should indicate that using the progress alert as indicated in Figure 2(a).
3) The loaded movies should be displayed in a ListView. The ListView items should be displayed to include the movie poster thumbnail, title, year, MPAA rating, critics rating image and the audience rating image, as shown in Figure 2(b). The audience and critic rating images are provided in the support files. ***Your adapter implementation should reusing 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.***

4) Clicking a movie item should display the movie details by starting the Movie activity. If the Movies activity is displaying the Favorite Movies, then upon returning from the Movie activity the ListView should be updated if the selected movie was removed from the favorites. Do not reload the whole list, instead only update (remove) the corresponding movie.
5) When the Movies activity is displaying the Favorite Movies, upon long pressing any of the favorite movies should remove this movie from the favorites by contacting the corresponding favorite API and should refresh the ListView to indicate this change, see Figure 2(c).
6) To retrieve the username the SharedPreferences should be used.



(a) Movies Activity Loaded for movie not in favorites

(b) Movies Activity Loaded for movie in favorites

(c) Movie website displayed in a WebView

Figure 3, The Movie and MovieWeb Activity Wireframe

Part D: Movie Activity (20 Points)

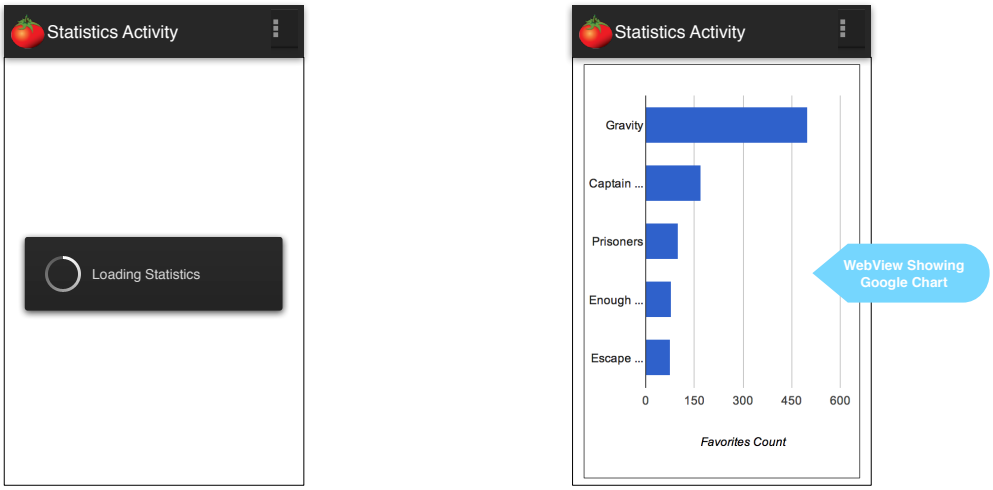
The Movie activity will display more details related to the movie selected in the Movies activity. The Movie activity wireframe is shown in Figures 3(a) and 3(b).

1) Loading the movie detailed image should be performed by a worker thread and not by the the main thread. If no poster is found use the “poster_not_found.png” image.
2) The audience and critics images are similar to ones used in the Movies activity. Note that you are also required to display a comma separated string showing the movie’s different genres, which would require using the Rotten Tomatoes Movies Info api.
3) The loaded movie is either in the favorite list or not. If the movie is not in the favorites this should be indicated by using the icon in Figure 3(a), and if it was in the favorites it should be indicated using the icon in Figure 3(b). Clicking not in favorites icon should add the current movie to the favorites and should change the icon to the in favorites icon. Similarly clicking the in favorites icon should remove the current movie from the favorites and should change the icon to the not in favorites icon.
4) Upon clicking the back icon (left arrow icon), the Movie activity should exit.
5) Upon clicking on the globe icon, the MovieWeb Activity should be started, and the movie website (alternate url) should be passed in the Intent extras. This loads the MovieWeb Activity to show the movie website in a WebView.
6) To retrieve the username the SharedPreferences should be used.

Part E: MovieWeb Activity (5 Points)

The MovieWeb activity will display the movie's rottentomatoes website in a WebView. The MovieWeb activity should be passed the movie url from the Movie Activity. The MovieWeb activity wireframe is shown in Figure 3(c).

1) The WebView should be setup to enable JavaScript and handle url redirects.
2) For information related to WebViews please check the Android Developers site: http://developer.android.com/reference/android/webkit/WebView.html



(a) Loading Statistics

(b) Displaying the Google Chart showing the statistics

Figure 4, Statistics Activity Wireframe

Part F: Statistics Activity (20 Points)

The MovieWeb activity will display a Google Charts bar chart showing the favorite statistics for the top 5 favorite movies. The bar chart used to visualize the favorite counts for 5 top favorited movies, this bar chart is displayed in a WebView running the JavaScript needed to display the Google Charts bar chart. The Activity should use getFavoriteStats API should to retrieve the top 5 favorited movies, then the statistics should be formatted and sent to the WebView to be visualized. The Statistics activity wireframe is shown in Figure 4.

1) The loading of the statistics should be performed in an AsyncTask or Thread.
2) The WebView should be setup to enable JavaScript and handle url redirects.
3) For information related to WebViews please check the Android Developers site: http://developer.android.com/reference/android/webkit/WebView.html
4) For information and examples related to the Google Charts bar charts api please check the https://google-developers.appspot.com/chart/interactive/docs/gallery/barchart
5) Hint: you should investigate mechanisms to pass JavaScript to a WebView in order to pass the required data from your Activity to the the JavaScript running in the WebView.