ITIS/ITCS 4180/5180 Mobile Application Development

Midterm

In-Class Programming Assignment

Basic Instructions:

1. This is the Midterm Exam, which will count for 15% of the total course grade.
2. In every file submitted you MUST place the following comments:
   a. 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. During the exam, you are allowed to use the course videos, slides, and your code from previous homeworks and in class assignments. You are NOT allowed to use code provided by other students or from other sources.
6. Answer all the exam parts, all the parts are required.
7. Please download the support files provided with the Midterm and use them when implementing your project.
8. **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.
9. 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.
10. **Failure to do the above instructions will result in loss of points.**
11. **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 make simple HTTP requests and will parse the iTunes RSS feed data. The Apple iTunes store provides a RSS feed for its different provided media types. In this assignment you will focus the iOS Apps RSS feed. The app is composed of 3 activities, namely Main Activity, Apps Activity and Preview Activity.

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 19**. The app should display correctly on Nexus 5. 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, XML parsing and image downloading should be performed using Threads (or AsyncTask) and your code should not block the main thread.
3. Your implementation should target the most efficient algorithms and data structures. You will be graded based on the efficiency of your implementation.
4. **For this assignment you will be provided with a project folder which includes the activity initial implementations and UI. Import the provided project and add your implementation.**

API Description

- The API used in this assignment is the iTunes RSS API. The api retrieves 100 top grossing applications provided by the iTunes store. Figure 1 shows the XML response and the required fields. To access the XML stream issue a GET request to the below url:
    - https://itunes.apple.com/us/rss/topgrossingapplications/limit=100/xml

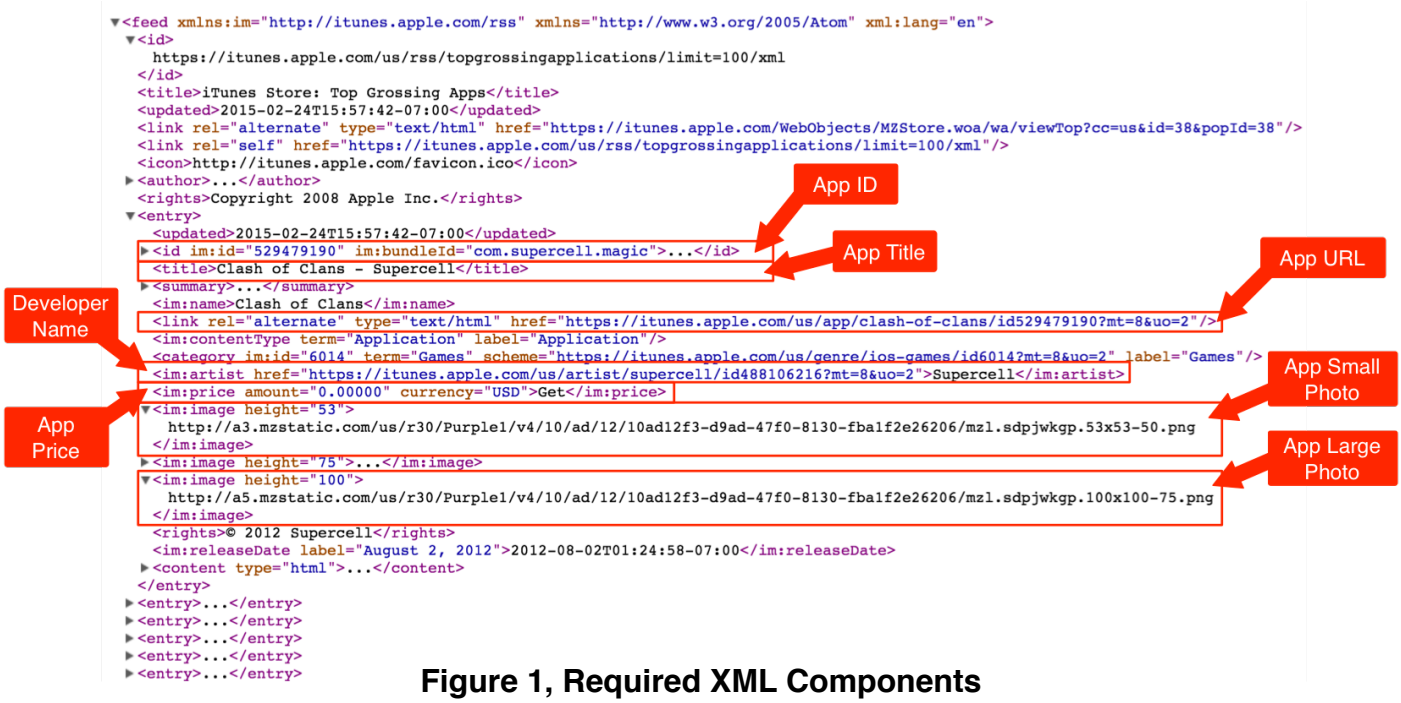


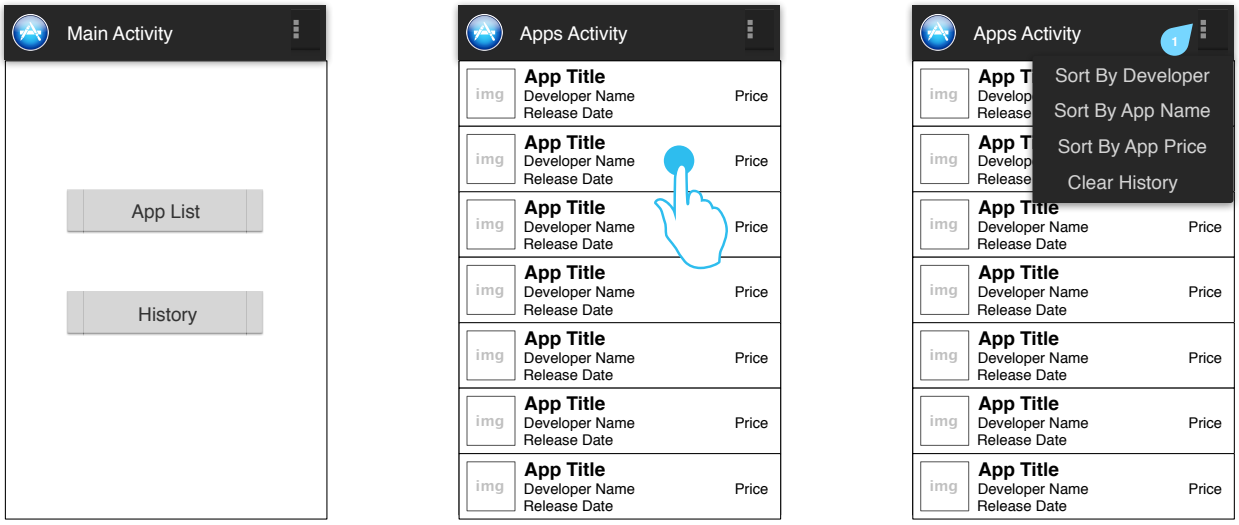
Figure 1, Required XML Components

- You are required to access the api and retrieve for each app the following fields: id, app title, developer name, url, small photo url, large photo url, and app price. The required fields are highlighted in Figure 1.

Part 1 (5 Points): MainActivity

The interface should be created to match the user interface (UI) presented in Fig 2(a). The Main Activity displays 2 buttons. Implementation requirements include:

1. Tapping on the "App List" button should start the Apps Activity, which should display the apps information retrieved directly from the online api.
2. Tapping on the "History" button should start the Apps Activity, which should display the list of apps perviously visited.



(a) Main Activity (b) Apps Activity (c) Menu Options

Figure 2, Wireframes of the Main Activity and Apps Activity

Part 2 (85 Points): Apps Activity

The interface should be created to match the user interface (UI) presented in Figures 2(b) and 2(c). The Apps activity is responsible for the loading the list of apps retrieved from the api or the list of perviously visited apps (history), which is based on the button pressed in the Main Activity. The implementation requirements include:

1. For the apps' list retrieved using the api:

a. Use a thread pool (or AsyncTask) to communicate with the iTunes api and to parse the result. Do not use the main threads to download the api results. Care should be taken while parsing and should take into consideration the namespace.

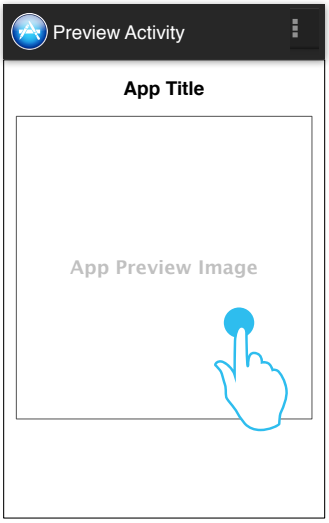
b. The retrieved apps' information should be displayed in a ListView as shown in Figure 2(b).

2. For the apps' list retrieved from the stored history:

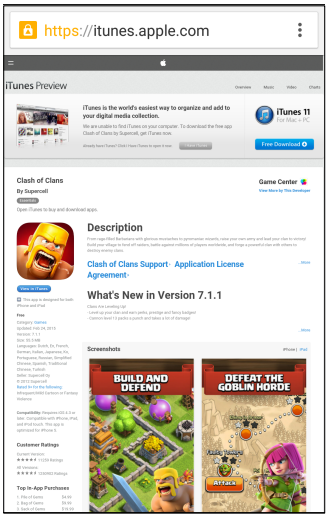
a. Retrieve the list of perviously visited apps from history. You are free to use either **SQLite or Shared Preferences** to implement the required history storage and retrieval functionality. Note that the history should store all the apps information required to display the app list as shown in Figure 2(b).

b. The retrieved apps' information should be displayed in a ListView as shown in Figure 2(b).
c. Please note that the app information should be stored in history each time an app is visited. For simplicity, an app visited multiple time can occur multiple times in the stored history.

3. Create a custom layout and adapter in order to display the required list view items as shown in Figure 2(b).
   a. For image loading/caching you must use the Picasso library.
   b. For the thumbnail image use the Small Photo URL retrieved from the parsed XML for each app item.
   c. For free apps, indicate the price as "0.0".
4. Clicking a list item should display the app details by starting the Preview Activity. Also the history should be updated to store the record of the selected app.
5. **Menu Items (Sorting)**
   a. Clicking the menu item "Sort by Developer" should sort and redisplay the displayed apps list in ascending order based on the developer's name.
   b. Clicking the menu item "Sort by App Name" should sort and redisplay the displayed apps list in ascending order based on the app name.
   c. Clicking the menu item "Sort by App Price" should sort and redisplay the displayed apps list in ascending order based on the app price.
6. **Menu Items (Clear History)**
   a. Clicking the menu item "Clear History" should clear all the stored app visit history. If the current list being displayed is the history list then it should be redisplayed to reflect clearing of the history.



(a) Preview Activity



(b) Native Web-Browser

Figure 3, Preview Activity

Part 3 (10 Points): PreviewActivity

This activity should display the app title, and the application preview image. The Preview Activity should receive the app information from the Apps Activity. Figure 3(a) shows the PreviewActivity. The implementation requirements include:

1. Use the “App Large Photo” retrieved for each app to display the app preview image.
2. Clicking on the “App Preview Image” ImageView should display the app url in the native browser, as shown in Figure 3(b).

Grading Key

- Main Activity (Total : 5 Points)
    - (5 Points): Two buttons display correctly and pass information to the App Activity through intents.

- Apps Activity (Total : 85 Points)
    - (20 Points) : Http connection and XML parsing
    - (20 Points) : Storing and retrieving app visit history through shared preferences or SQLite database.
    - (20 Points) : Extending the Adapter, and displaying the list items correctly.
    - (15 Points) : Correctly sorting the list based on the selected menu item and refreshing the list.
    - (10 Points): Clearing the history and reloading the list when required.

- Preview Activity (Total : 10 Points)
    - (5 Points) : Receives data through intents and correctly displays the app information and app image using Picasso library.
    - (5 Points) : Upon clicking the app image it loads the app url in the native browser.