ITIS/ITCS 5180 Mobile Application Development
In Class Assignment 5

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 loose points if your code is not efficient, does unnecessary processing or blocks the UI thread.
4. Export your Android project and create a zip file which includes all the project folder and any required libraries.
5. Submission details:
   a. Only a single group member is required to submit on canvas for each group.
   b. The file name is very important and should follow the following format: **Group#_InClass05.zip**
   c. You should submit the assignment through Canvas: Submit the zip file.
6. **Failure to follow the above instructions will result in point deductions.**

In Class Assignment 05 (100 Points)

In this assignment we are going to make a Photo Gallery application, which will use a URL that retrieves a text file containing a dictionary of keywords and image URLs related to the associated keyword. The application consists of a single activity that enables the user to download and view online photos.

APIs:

- (Get Keywords API) To get the list of possible keywords:
  - URL : http://dev.theappsdr.com/apis/photos/keywords.php
  - Method : GET
  - Output : returns the possible keywords separated by ";". For example: android;aurora;uncc;winter;wonders
- (Get Urls API) To get photos for a given keyword:
  - URL : http://dev.theappsdr.com/apis/photos/index.php
  - Method : GET
  - Parameter:
    - keyword : is the keyword selected by the user.
  - Output: returns the photo urls for images related to this keyword with each url on a separate line. For example for a keyword parameter set to winter the following output is returned

    https://c2.staticflickr.com/6/5538/11966402046_fbd73d52e9_c_d.jpg
    https://c1.staticflickr.com/5/4010/4255115508_e21a09138c_z_d.jpg
    https://c2.staticflickr.com/4/3746/11948232454_bd5cfd5b8f_c_d.jpg
    https://c1.staticflickr.com/9/8474/8413293701_4c63cb5ff8_c_d.jpg
  - Note: The retrieved text contains a url in each separate line. The "keyword" is the search keyword. For example, "weather" returns 4 URLs associated with it. Other keywords may have more or less.

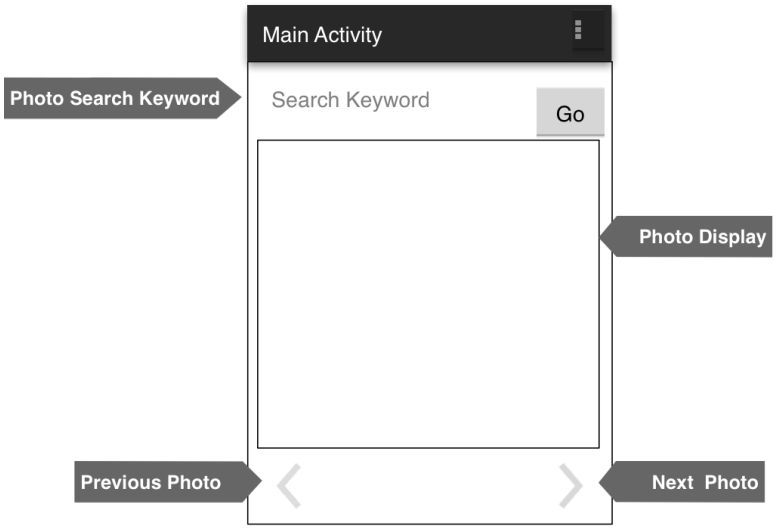
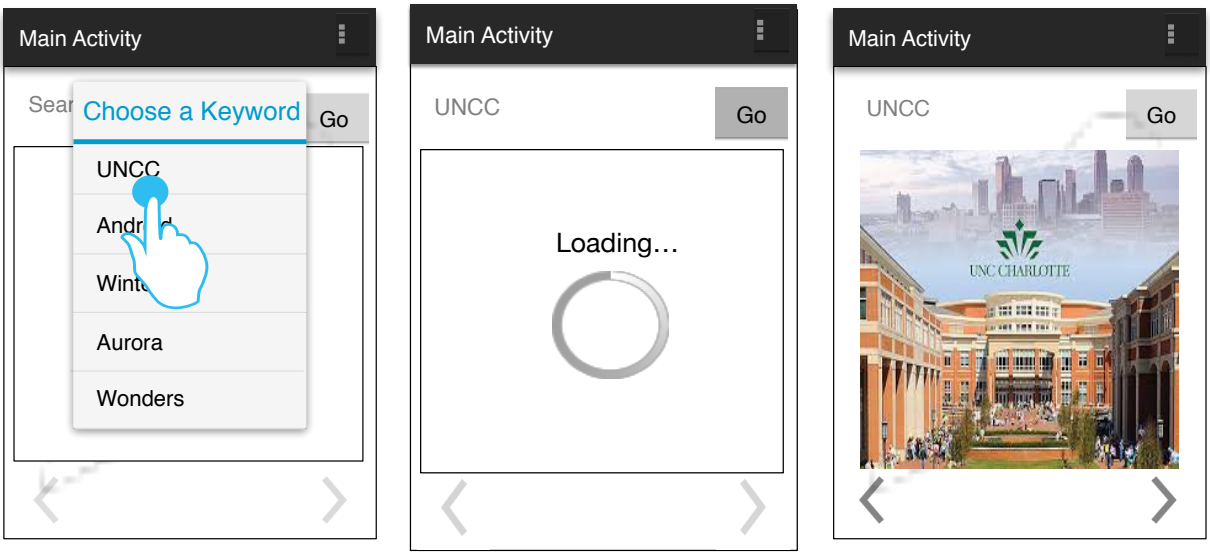
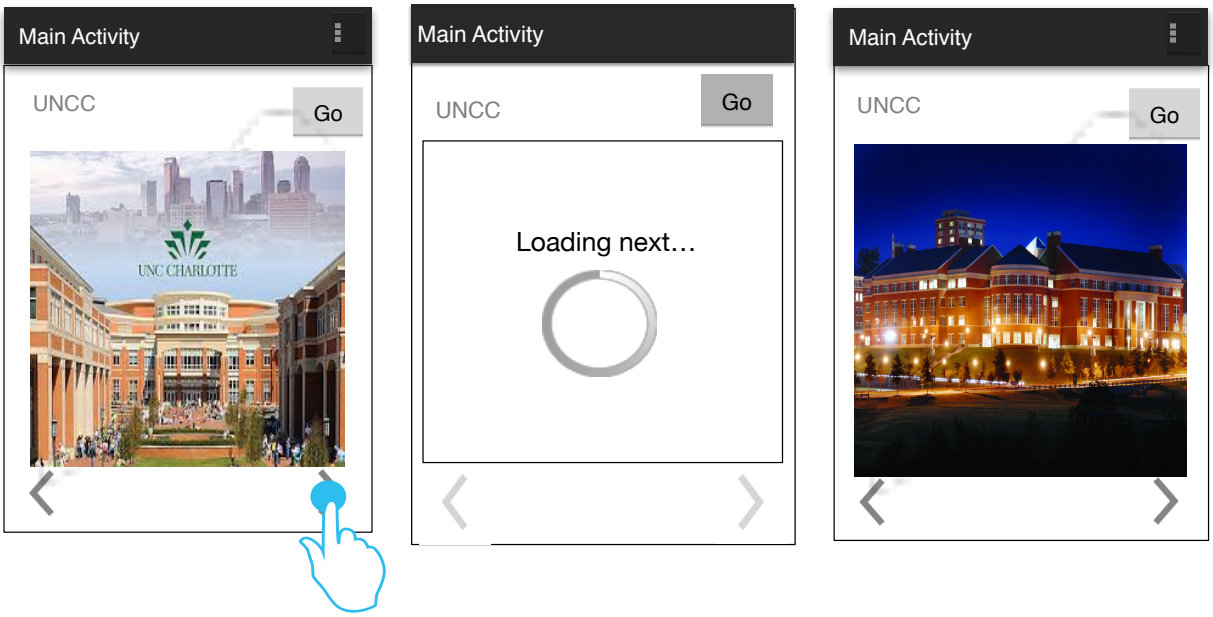


Figure 1, Application Wireframe



(a) Select Keyword (b) Loading the first image (c) First image

Figure 2, App Wireframe



(a) Next Image (b) Loading Photo (c) Show Next Photo

Figure 3, App Wireframe

Loading Images based on Keywords

The interface should be created to match the user interface (UI) presented in Figure 1. You will be using layout files, and strings.xml to create the user interface. Perform the following tasks:

1. When the activity starts, the keyword api should be called to get the list of possible keywords. The keyword api should be called using an AsyncTask.
2. Clicking on the "Go" Button should display a list of keywords as shown in Figure 2(a). You can either Alert Dialog or Spinner to implement it.
3. Clicking on a Keyword should do the following:
   a) The TextView will hold the search keyword clicked by the user.
   b) Use AsyncTask/Thread to connect to the Get URLs API and retrieve the list of image urls related to the selected keyword.
4. When the api data is retrieved:
   a) Use another AsyncTask/Thread to retrieve the first image associated with the keyword and display it in the ImageView.
   b) The AsyncTask/Thread class should be in a separate file/class not inner to the main thread. i.e. You should manage passing parameters to the class and then pass back the result image downloaded to the UI.
   c) While the image is being retrieved, you should display a Progress Bar as indicated in Figure 3(b).
5. The Next and Previous photo icons should be disabled when the application is launched, and enabled after the first photo is displayed. The buttons will also remain disabled in the case there is only 1 image or there are no images corresponding to a keyword. Use icons provided in Support Files for setting the image icons (**next.png, prev.png**)
6. **Do not** store the photos, simply download and display the retrieved photos. Also do not attempt to download all the URLs receive, and your app should only download and display a single photo at any given time.
7. Upon clicking the "Next Photo" icon, you should download the next photo in list of URLs retrieved (following the of order of appearance in the URL list retrieved). You should call the AsyncTask/ Thread used to download the next photo.
   a) If the currently displayed photo happens to be the last photo, you should download and retrieve the photo at index 0 (the first photo) when the Next icon is pressed.
8. Clicking the "Previous Photo" icon, you should download the previous photo. If the currently displayed photo happens to be the first photo, you should download and retrieve the photo at the last index position (last photo).
9. If the api returns an empty list of urls, then clear the currently displayed ImageView and display an error Toast message "No Images Found".
10. Your application should download the requested photo only if there is an established internet connection. If there is no internet connection you should display a Toast message indicting that there is no internet connection and do not attempt to send the HTTP request.