ITIS/ITCS 4180/5180 Mobile Application Development
Homework 3
Date Posted: 09/22/2015 at 23:55
Due Date: 09/29/2015 at 23:55

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 student in the group is required to submit the assignment on Moodle.
3. Please download the support files provided with this assignment and use them when implementing your project.
4. Export your 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 project for this assignment 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. When exporting 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.
5. Submission details:
   a. All the group members should submit the same zip file.
   b. The file name is very important and should follow the following format: **Group#_HW03.zip**
   c. You should submit the assignment through Moodle: Submit the zip file.
6. **Failure to follow the above instructions will result in point deductions.**

Homework 3 (100 Points)

In this assignment you will develop a Trivia quiz. You will get familiar with AsyncTasks/ Thread, and making Http connections. This project is composed of four Activities.

Important App Requirements:

1. All strings should be read from your strings.xml, all dimensions from dimens.xml, and all images from drawable-ldpi. The string values used for the text labels and button labels should be read from the `strings.xml` file, not be hardwired in the layout file.
2. All network access and image downloading should be performed using Threads (or AsyncTask) and your code should not block the main thread.

Initial Setup and API Description:

1. In order to access the API, your group needs to obtain a group identifier by using the form provided at http://dev.theappsdr.com/apis/trivia_fall15/index.php. Using the form you will be required to enter your group name as indicated under your moodle profile. Upon completing the form all the group members will be sent an email containing the Group ID to be use to call the APIs in this assignment.

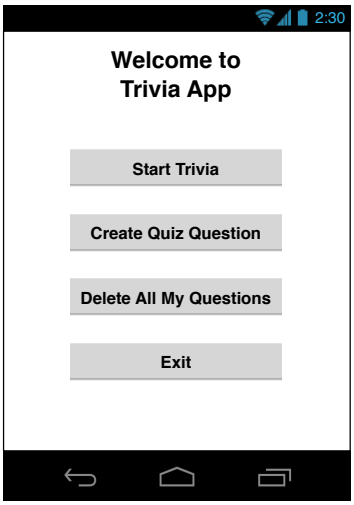
API Description:

You are provided with a test website which includes a form that will allow you to test these APIs, http://dev.theappsdr.com/apis/trivia_fall15/index.php. The different API descriptions are listed below.

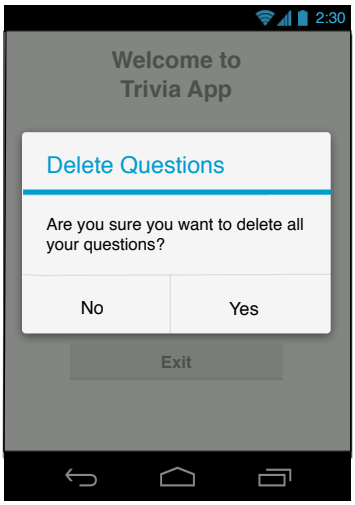
- **getAll API:**
  - URL: http://dev.theappsdr.com/apis/trivia_fall15/getAll.php
  - No Parameters.
  - Method: GET
  - Description: this API returns the list of questions stored in the online database. These questions include all the questions submitted by all the other groups in the class. The api returns each question in a single line, and the question attributes are semicolon separated as will be described in assignment description.
- **checkAnswer API:**
  - URL: http://dev.theappsdr.com/apis/trivia_fall15/checkAnswer.php
  - Parameters:
    - "gid": the group id that was provided through email
    - "qid": the question id
    - "a": the provided answer to the question
  - Method: POST
  - Description: this API returns a 0 if the provided answer is incorrect, or a 1 if the answer is correct. This should be called whenever a trivia question response is submitted. In case of any error the API returns -1.
- **uploadPhoto API:**
  - URL: http://dev.theappsdr.com/apis/trivia_fall15/uploadPhoto.php
  - Parameters:
    - "uploaded_file": a photo to be uploaded to the server.
  - Method: POST

- Description: this API allows a photo to be uploaded, and returns the file URL of the uploaded photo, which can be submitted as part of a new question. In case of error the api will return -1.

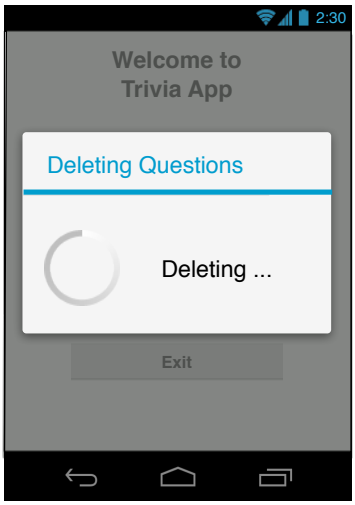
- **saveNew API:**
  - URL: http://dev.theappsdr.com/apis/trivia_fall15/saveNew.php
  - Parameters:
    - "gid": the group id that was provided through email
    - "q": Question formatted in a single line, which is semicolon separated.
  - Method: POST
  - Description: this API allows a group to create and store a new question on the online database. Please note that the question will be visible to all other groups. If the question is successfully created the API returns -1 other wise it returns -1.
- **deleteAll API:**
  - URL: http://dev.theappsdr.com/apis/trivia_fall15/deleteAll.php
  - Parameters:
    - "gid": the group id that was provided by the TA.
  - Method: POST
  - Description: this API deletes all questions belonging to the group "gid" provided. If the delete was successful the API returns -1 other wise it returns -1.



(a) Main Activity



(b) When Delete Button Pressed



(c) Progress bar after the user confirms the delete

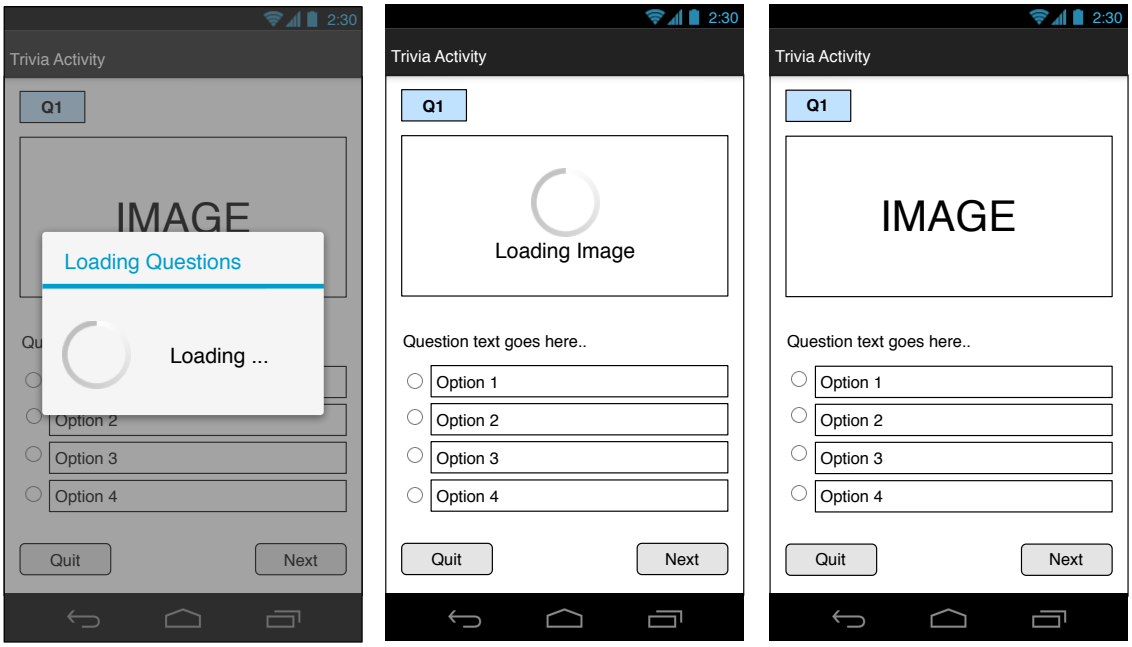
Figure 1, MainActivity

Part 1: MainActivity (15 Points)

The Main activity is shown in Figure 1(a). The activity requirements are as follows:

1. Clicking on the "Start Trivia" button starts the Trivia Activity, where the current quiz questions are loaded, parsed, and displayed.
2. Clicking the "Create Quiz Question" button starts the CreateQuizQuestion Activity, where users can submit their own questions.
3. Clicking the "Delete All My Questions" button displays an alert dialog where the user is prompted to confirm their choice to delete their questions, see Figure 1(b).

4. If the user confirms the deletion of the questions then the deleteAll API should called. A progress dialog should be displayed while the questions are being deleted and should be dismissed after the deletion is completed. Figure 1(c), shows the progress dialog being displayed while the API is being called.



(a) Loading Questions **(b) Loading Image** **(c) Loaded Question**

Figure 2, Trivia Activity

Part 2: Trivia Activity (50 Points)

The trivia activity retrieves, and displays the trivia questions as indicated in Figure 2. The activity requirements are as follows:

1. The questions should be retrieved using the getAll api, when the trivia activity is started. The Trivia Activity should use an AsyncTask or Thread to retrieve the questions and answers at the URL provided above. While the questions are being retrieved the Activity should display a progress bar as indicated in Figure 2(a). The progress dialog should be dismissed when the questions are retrieved. The getAll api will return all the questions. Each question is formatted in a separate line with entries separated by the semicolon ";". An example question is listed below:

```
100;Sheldon is very dedicated to his work but he also has outside interests and
hobbies. Which of these is one of them?;Bird watching;Open mic night stand-up
comedy;Model trains;dancing;http://dev.theappsdr.com/lectures/trivia/photos/Q2.jpg;
```

2. The question line is divided as follows: the first item is the question id, followed by the question itself, then the list of possible answers, followed by the question image

url (which is optional). You should note that the number of possible answers can vary, in the above example there are 4 possible answers, but in other questions might vary. Questions should have at least 2 possible listed choices. The below figure shows and example question with no image url.

```
12;Sheldon is very dedicated to his work but he also has outside interests and
hobbies. Which of these is one of them?;Bird watching;Open mic night stand-up
comedy;Model trains;dancing;;
```

3. You should create a Question Class that should hold the parsed question, possible answers, and url. The AsyncTask or Thread should return a List of Question objects. Do not pre-download any of the images and do not pre-store the images in the Question Class.
4. The activity shows the question number, a question text, and the set of answer options. The Quiz activity also shows the current question's image. Note that not all questions will have a question image, so if there is no image, the url entry will be left as "" to indicate no image. You should display an image to indicate that there was no image found for this question. The number of possible answers is variable, your interface should be able to handle a variable number of answer choices.
5. If the current question has an image, then the image should be downloaded from the specified image url indicated in the question using a separate thread (or AsyncTask) and not using the main thread. Your activity should ensure that the ***downloaded image is displayed only when its question is currently displayed not when other questions are displayed***.
6. While the question image is still loading you should display a progress bar indicating the image is loading, as indicated in Figure 2(b).
7. When displaying the next question you should not use a new Question activity instead update the layout of the Quiz activity to display the new question. Note that the number of choices for each question varies, so the views representing the choices should be dynamically generated in your code and should not be statically created in the layout xml.
8. If the user clicks the "Quit" button the activity is finished and the user is sent back to the Main Activity.
9. When the user answers a question by responding and clicking next, you should detect whether the selected answer was correct or not, by calling the checkAnswer API, which will retrieve use an AsyncTask or Thread to retrieve the correct answer for that question . Once it is retrieved, check the result returned by the API and indicate with a Toast whether it is correct or incorrect. Keep track the number of correctly answered questions.
10. Upon answering the last question, the Quiz activity should start the Result Activity and send it the required information in the intent to display the result.

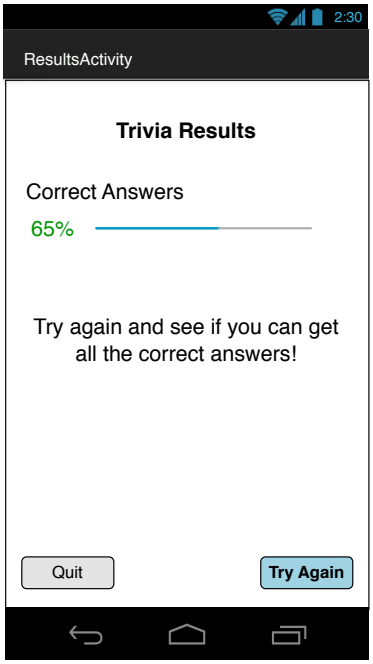


Figure 4, Results Activity

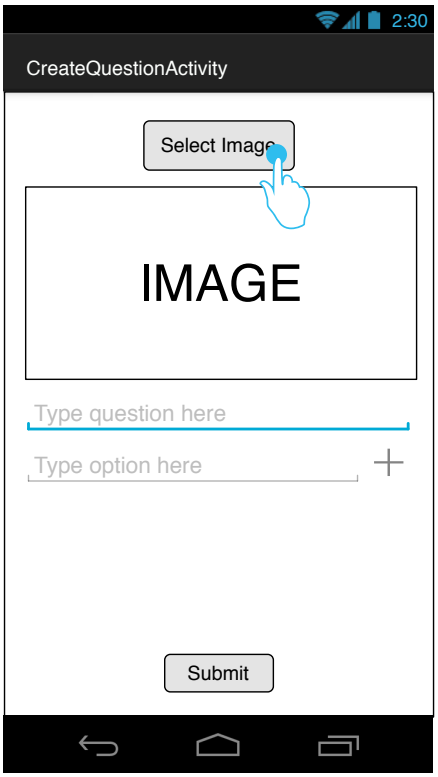
Part 3: Result Activity (10 Points)

Figure 4 shows the wireframe for the Result Activity. This activity shows the user the percentage correctly answered quiz questions. Clicking the "Quit" button should send the user to the Main Activity. Clicking the "Try Again" button should send the user back to the Trivia activity, should reload the questions from the server and display the first quiz question to enable the user to retry the quiz from the first question.

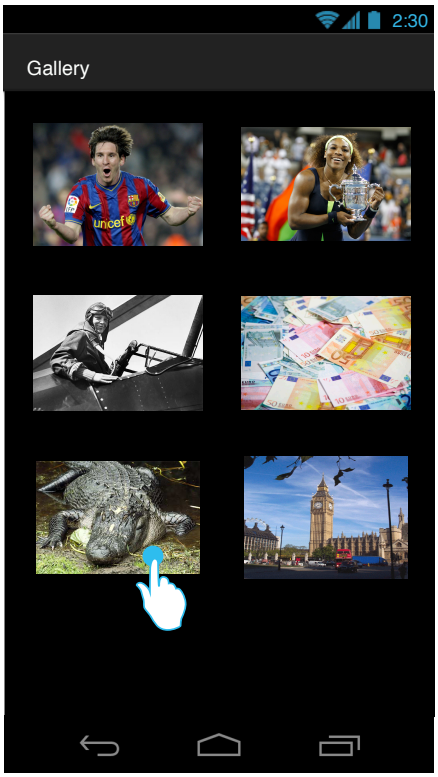
Part 4: Create Question Activity (25 Points)

Figure 5 shows the wireframe for the CreateQuestion Activity. This activity allows the user to submit their own question to the online repository. The activity requirements are as follows:

1. It is optional for question to include an image. Clicking the "Select Image" button should launch an intent to the Gallery, where an image will be chosen, see Figure 5(b), and should be displayed in as shown in Figure 5(c). **Warning: Be careful. Any inappropriate images (including pornography, racially or religiously insensitive images) will not be tolerated. If you post any violating material your assignment will not be graded and your group will receive ZERO on the assignment. Be sensitive of others and only submit appropriate images.**
2. There should be an EditText for the question input, an EditText to input an answer option, with a "+" button to add additional options, and another EditText for the question image URL, see Figure 5a.
3. To submit a question, it should have question text and 2 or more answer options. In order to add answer options to the question, the user must enter the option text in the options EditText and then click the "+" button, see Figure 5c. This will dynamically add a radio button for the option. One of the radio buttons for the options must be selected in order to indicate which is the intended correct answer,

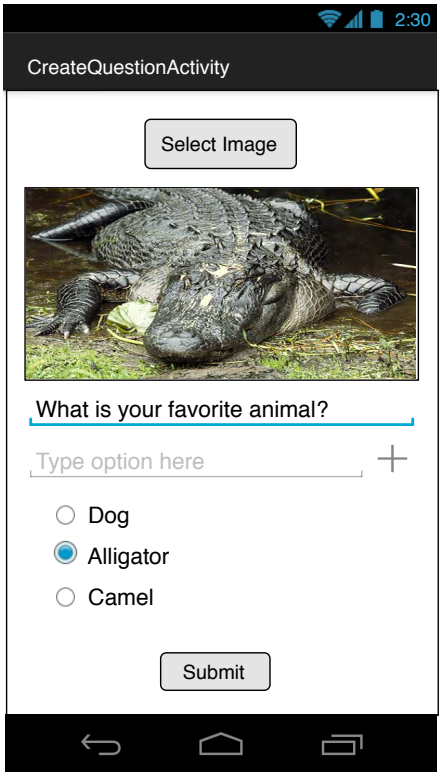


(a) Empty CreateQuestion fields



(b) Choosing an image from gallery

(c) Populating Question and answer fields



(d) After input of options and image

Figure 5, CreateQuestion Activity Wireframe

see Figure 5d.

4. Once all the necessary info as been input, then submitting the content involves two stages:
   (a) If an image is to be included with the question, then first call the uploadPhoto API which uploads the photo to the server and returns the URL of the photo on the server in the response. Then submit the question content, including the URL that was returned for the uploaded image, using the saveNew API as a single, semicolon-separated string, in the format shown below. It is similar to the format as the question entries that were downloaded in Trivia Activity, **except there is no question id and the last parameter is the correct answer index**. Include the randomly-generated group ID in the submission as a POST parameter, or the submission will not be successful. Display a ProgressDialog until it has successfully been submitted, and then the activity should return to Main Activity.

```
Sheldon is very dedicated to his work but he also has outside interests and hobbies.
Which of these is one of them?;Bird watching;Open mic night stand-up comedy;Model
trains;dancing;http://dev.theappsdr.com/lectures/trivia/photos/Q2.jpg;2;
```

   (b) If no image is to be included just follow the second step indicated above, and the formatted question should be as follows:

```
Sheldon is very dedicated to his work but he also has outside interests and hobbies.
Which of these is one of them?;Bird watching;Open mic night stand-up comedy;Model
trains;dancing;;2;
```

Note that we are allowing groups to share input, so this might result in errors in provided questions. Your application should be able to address these errors and should be able to check for errors in the formatting of the received questions. Questions that do not match the line formatting described above should not be displayed.