ITIS/ITCS 4180/5180 Mobile Application Development

Midterm

Date Posted: 03/14/2013 at 18:00

Due Date: 03/22/2013 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. This exam is composed of 3 parts, answer all of the parts.
7. If any problems arise during this exam, email me (mshehab (at) uncc.edu).
8. Please download the support files provided with the Midterm and use them when implementing your project.
9. **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.
10. 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.
11. **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 Midterm you will develop a poll system. Users will be able to view a number of polls, answer them, and look at the overall poll results. The application will communicate with several server side APIs, to retrieve the polls, to submit the user's answer, and to delete the user's answer(s). All data retrieved from the APIs will be in XML format; hence you will need to parse the data appropriately. The Midterm is divided into 3 parts.

Initial Setup and API Description

- You are provided with a java class "**Config.java**", import it to your project.
- Edit the "**Config.java**", by setting the "**YOUR49ERUSERNAME**" to your UNCC user name, please the username listed on Moodle. If your email is "bsmith@uncc.edu" then set the variable to "bsmith". To retrieve the "uid" you should use the "Config.getUid()" method.
- All the API calls must be through POST method, all the parameters must be passed within the post body.
- There are four main APIs
  1) **getPolls** API: used to retrieve all the polls provided by the server.
  2) **submitPollAnswer** API: used to submit a poll answer.
  3) **deletePollAnswer** API: used to delete the answer to a specific poll.
  4) **deleteAllPollAnswers** API: used to delete all the submitted poll answers.
- 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/polls/index.php

getPolls API:

- *URL*: http://cci-webdev.uncc.edu/~mshehab/polls/getPolls.php
- *Arguments*: (POST method)
  - *uid* : the uid value returned by calling "Config.getUid()"
- Description: This API will retrieve all the polls available on the server. 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 data returned is a list of polls in XML format. The root of the XML returned is a **response** element. The first child of the root element is an **error** element, which will include an error **code**. By reading the error code you will be able to identify if any errors have occurred, and what type of error it was (See Errors Section). The following shows the response data returned:
- Each **poll** element will have "pid" attribute, which represents the poll id.
- Each **poll** element will have a "completed" attribute, which represents if the user has already answered this poll or no. If the "completed" attribute is set "1" this indicates that the user with the specified uid has already answered this poll question, and if it is set to "0" this indicates that the user did not answer this question.
- The "youranswer" attribute, is included only if the user completed this poll. The value of the "youranswer" attribute will include the answer id (aid) selected by the user. In the above example for pid "0" the "youranswer" attribute is set to "2" which indicates that the user previously selected answer with aid "2", which is the "Never tried" answer.

```
<response>
        <error code="0"/>
        <poll pid="0" completed="1" youranswer="2">
                <question>Do you like curry?</question>
                <answer aid="0" percent="71.4">Yes</answer>
                <answer aid="1" percent="25.0">No</answer>
                <answer aid="2" percent="3.6">Never tried</answer>
        </poll>
        <poll pid="1" completed="0">
                <question>Do you like to use post-it notes?</question>
                <answer aid="0">Yes</answer>
                <answer aid="1">No</answer>
                <answer aid="2">I dont know</answer>
        </poll>
        ... ...
</response>
```

- Each **poll** element has multiple children **answer** elements. Each answer element has an "aid" attribute, which represents the answer id.
- The **answer** elements of completed polls ("completed" attribute set "1") will also include a "percent" attribute, which represents the overall percentage of users who have chosen a specific answer for this poll. The "percent" attribute is only available for completed polls.

submitPollAnswer API:

- *URL*: http://cci-webdev.uncc.edu/~mshehab/polls/submitPollAnswer.php
- *Arguments*: (POST method)
  - *uid* : the uid value returned by calling "Config.getUid()".
  - *pid* : the poll id.
  - *aid* : the chosen answer id.
- Description: This API will submit the user's answer to a specific poll to 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 poll answered with the answer percentages also included. A successful response is as follows:

```
<response>
        <error code="0"/>
        <poll pid="5" completed="1" youranswer="1">
                <question>Do you chew your pens and pencils?</question>
                <answer aid="0" percent="37.5">Yes</answer>
                <answer aid="1" percent="62.5">No</answer>
        </poll>
</response>
```

deletePollAnswer API:

- *URL*: http://cci-webdev.uncc.edu/~mshehab/polls/deletePollAnswer.php

- *Arguments*: (POST method)
  - *uid* : the uid value returned by calling “Config.getUid()”.
  - *pid* : the poll id.
- Description: This API will delete the user’s pervious answer to a specific poll. Failing to pass the required parameters will result in error response from the server. A successful response is as follows:

```
<response>
        <error code="0">Done deleting poll response</error>
</response>
```

deleteAllPollAnswers API:

- *URL*: http://cci-webdev.uncc.edu/~mshehab/polls/deleteAllPollAnswers.php
- *Arguments*: (POST method)
  - *uid* : the uid value returned by calling “Config.getUid()”.
- Description: This API will delete all the user’s pervious answers to all the polls. Failing to pass the required parameters will result in error response from the server. A successful response is as follows:

```
<response>
        <error code="0">Done deleting all your poll responses</error>
</response>
```

Errors:

- If an error is detected, you should show a Toast with the corresponding error message.
- The following are the possible error codes and their corresponding messages. Note that an error code of 0, which means no error has occurred.

```
<error code="0"/>
<error code="0">Done deleting poll response</error>
<error code="-1">Missing parameters</error>
<error code="-2">uid not found</error>
<error code="-3">pid not found</error>
<error code="-4">aid or pid not found</error>
<error code="-5">Database error</error>
```

Part A: Home Activity (40 Points):

This activity is displayed when the user starts the application. It is also responsible for loading all the polls from the server using the **getPolls** API. The Home Activity's proposed layout is displayed below.

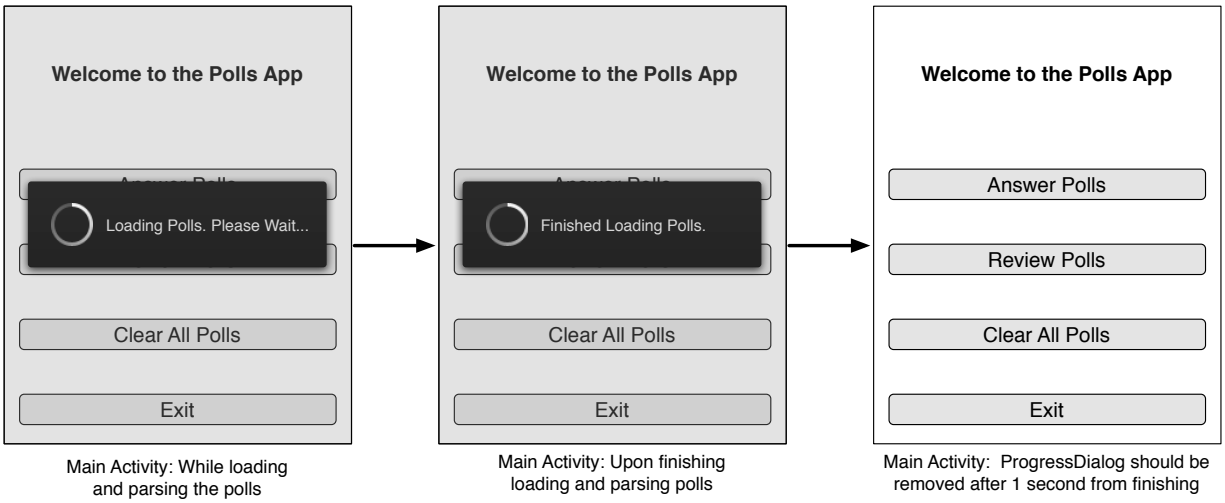

Figure 1, Main Activity: Loading Polls

Requirements:

- When this activity is started it should load the XML file from the server and display a ProgressDialog indicating that the polls are being loaded as indicated in Figure 1. Note that, when the polls are loaded and parsed the ProgressDialog message is changed to "Finished Loading Polls.", after that the ProgressDialog is displayed for 1 second and then removed.
- You should use a separate thread to perform the retrieval and parsing of the XML from the server. Do not use the Main Thread to perform these tasks. Also you should use a "Handler" to pass messages between the Child thread and the Main Thread. Or you can use AsyncTask.
- The recommended method for calling the APIs is through Apache's **HttpClient** class (See the "POST Params" example in the class notes). To retrieve the uid make sure to call "Config.getUid()".
- You should load and update the list of polls whenever you resume the Home activity (This allows you to get the most recent poll statistics).
- When the "Answer Polls" button is tapped it should send the Poll Activity a list of unanswered polls, and start the Poll Activity.
- When the "Review Polls" button is tapped it should send the Review Activity a list of answered polls, and start the Review Activity.
- The "Answer Polls" and "Review Polls" buttons should be enabled only when the Child thread is done retrieving and parsing the poll questions from the server.
- When the "Clear All Polls" button is tapped the activity should use the **deleteAllPollAnswers** to delete all the poll answers from the server. You should use a separate thread to perform this task. Similar to the poll loading, a ProgressDialog should be used to indicate that the polls are being cleared as indicated in Figure 2. Note that, when the response is successfully loaded and

parsed the ProgressDialog message is changed to “Finished Clearing Polls.”, after that the ProgressDialog is displayed for 1 second and then removed. Note, that if all the poll answers are cleared then the previously retrieved polls should be updated to not answered.

- Tapping the “Exit” button should quit the entire application.
- If an error occurs when receiving or parsing the XML, you should show the proper error message via a Toast.

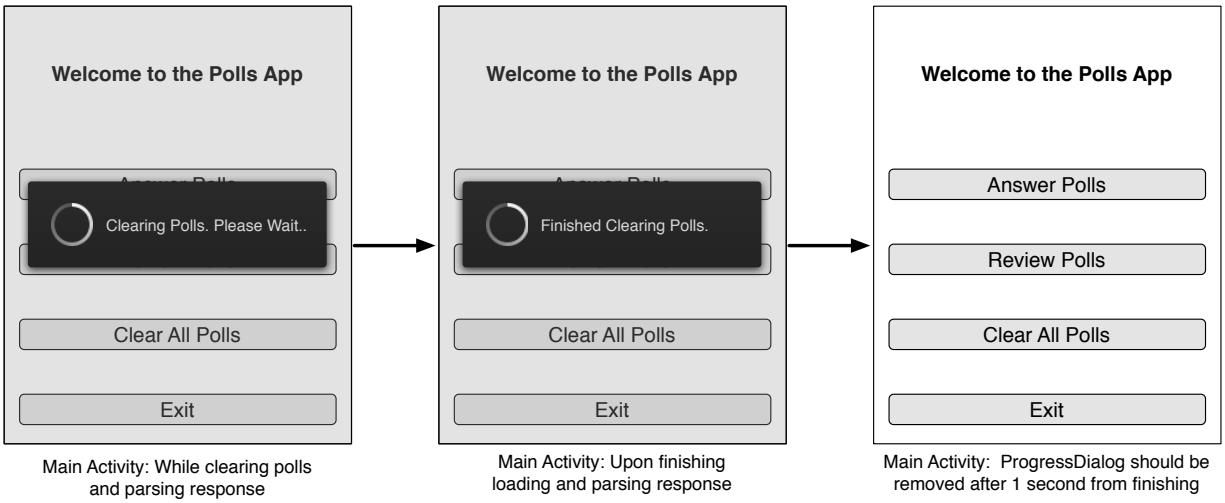

Figure 2, Main Activity: Clearing Polls

Part 2: Poll Activity (40 Points)

The Poll Activity is used to display unanswered polls (complete = 0). The Poll Activity will receive a list of unanswered polls from the Home Activity. The Poll activity should display the poll question using a text view and the set of answers using radio buttons.

Requirements:

- This activity should display the poll question and the list of possible answers, using TextView and RadioButtons respectively, as indicated in Figure 3.
- Once the user selects one of the answers, and taps the “Next” button, the activity should display the next question and the list of possible answers. The activity should maintain the user’s answers to the displayed questions. If the user attempts to tap on the “Next” button without selecting an answer generate a Toast message indicating that an answer should be selected.
- Upon answering the last question and tapping the “Next” button, the activity should submit the recorded answers using the **submitPollAnswer** API. The answers should be submitted synchronously such that the request to submit an answer is submitted after the response from the previously submitted answer is received.
- The recommended method for calling the APIs is through Apache’s **HttpClient** class (See the “POST Params” example in the class notes). Note that this API requires the answer id (aid), poll id (pid) and user id (uid). To retrieve the uid make sure to call “Config.getUid()”. You should use a separate thread to submit

the poll answer and to parse the returned XML. Do not use the Main Thread to perform these tasks. You should use handlers or AsyncTask address the messaging between the child and main threads. Do not block the Main thread. If an error occurs when submitting an answer to the server, you should show the proper error message via a Toast.

- Display a ProgressDialog indicating the progress submitting the answers to the server using the API (See Figure 3). The ProgressDialog should show the percentage completed answer submissions. Hint, using AsyncTask you could use the publishProgress() method to update the ProgressDialog progress. Upon submitting the last answer the ProgressDialog should be removed and the "Next" button should be disabled.
- Tapping on the "Home" button should take the user to the Home Activity. Note that if the user taps the "Home" button before answering the last question then none of the answers are submitted to the server.

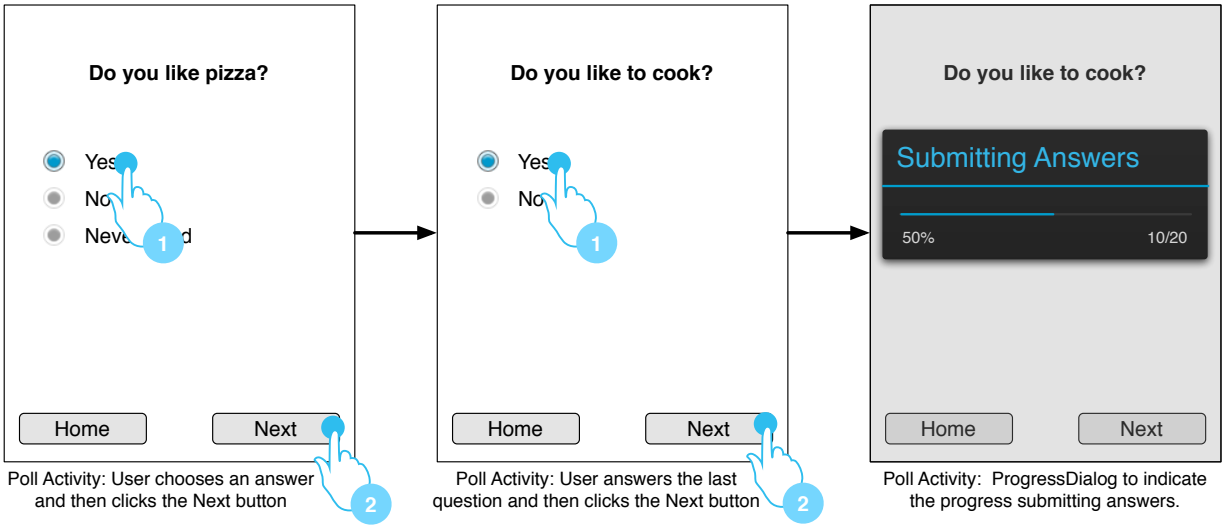

Figure 3, Poll Activity: Answer submission and answer stats

Part 3: Review Activity (20 Points):

The Review Activity is used to display the poll questions that were previously answered by the user. The activity will display the poll question, and the percentage responses for each answer of the poll question. The Review Activity will receive a list of answered polls from the Home Activity.

Requirements:

- The Review activity should display the poll questions and the set of questions using text views. In addition, the percentage response should be displayed beside each answer.
- The user's answer to this poll should be displayed in blue as indicted in the activity's wireframe, Figure 4.
- Upon tapping the "Next" button the next poll question, answers and percentages

should be displayed. When the last poll question is reached, upon tapping “Next” button, the Review Activity should indicate using a Toast message that the last question has been reached.

- Tapping the “Home” button should take the user to the Home Activity.
- Tapping the “Clear” button the activity should use the **deletePollAnswer** API to delete the user’s current poll answer from the server. You should use a separate thread to perform this task. A ProgressDialog should be displayed indicating that the poll answer is being cleared. Upon receiving and parsing the XML from the server, if there is no error (code=0), remove the ProgressDialog and display the next poll question.(see Figure 4).

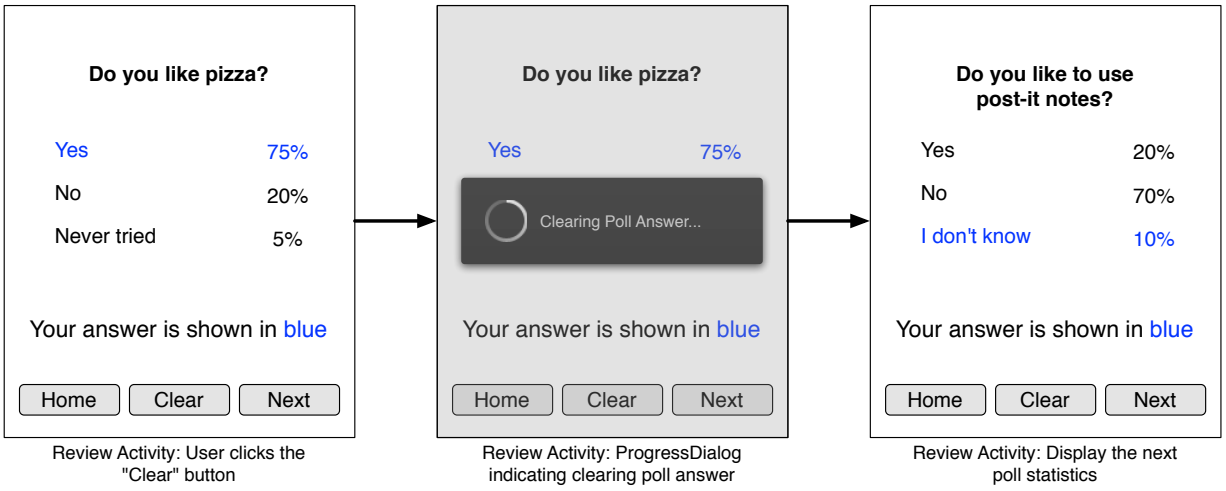

Figure 4, Review Activity: clearing a poll answer