ECE 1778 - Creativity and Programming for Mobile Devices
February 2014
Programming Assignment P4, for Programmers

Threads and Databases

The goal of this assignment is to learn the how to offload non-user interface tasks into separate threads, and to get a handle on the basics of a simple on-device database.

1 Reading

Read the following sections from the course texts, if you are developing on Android:

i. Pages 201-211 ("The WebView Widget") of the **The Busy Coder's Guide to Android Development**, version 5.4.
ii. Pages 381-393 ("Dealing with Threads") of the **The Busy Coder's Guide to Android Development**, version 5.4.
iii. Pages 483-502 ("SQL Lite Databases") of the **The Busy Coder's Guide to Android Development**, version 5.4.

For iPhone: from **Beginning iPhone 6 Development Exploring the iOS SDK** by Mark, Nutting, LaMarche and Olsson and the web**,** read:

i. Review the UIWebview Class – at this link.
ii. Lookup the method initWithContentsOfURL in the iPhone Documentation..
iii. Chapter 15 ("Grand Central Dispatch, Background Processing, and You").
iv. Chapter 13 ("Basic Data Persistence"), the section on Using SQLite3.

2 Assignment

NOTE: As in previous assignments, when writing your code for this assignment, please be sure to follow 'Braiden Brousseau's Guide To Quality Apps' that was given as part of Assignment P1. Part of your grade will be assigned for fulfilling them.

You are to write an application that creates a small SQL database that is populated from a file (that contains a list of names) located at a specified URL on the Internet. It will then search for those names on the Internet, and display the results. The application will do two different pieces of work (populating the database and searching) by spawning two separate threads from the main UI activity. Here is the specification in more detail:

When the application launches, it should present the user with three interface widgets:

1. A text field (call it DBURL) that defaults to the following string: 'http://www.eecg.utoronto.ca/~jayar/PeopleList' This string should be changeable by the user.
2. A button that is labeled 'Populate' that is initially active.
3. A button that is labeled 'Search' that is initially **inactive**. (i.e. touching it causes nothing to happen).

When the *Populate* button is touched, the App should spawn a thread that goes to the DBURL (which is a simple text file of names, one per line) and populate the database with the list of names there. While this is happening, the UI should display some kind of 'I know you're waiting' visual – such as a progress bar or spinning wheel. At this point it should not be possible to activate the *Search* button.

When the database is populated the first thread should send a message to the main UI process that it has finished (and then it should terminate), and the UI should both stop displaying the wait motif, and should emit a 'toast' (Android) or an 'alert' (iPhone) that indicates the database is loaded. At this point the second button ('Search') should become active.

For iPhone Developers

When the search button is touched a UIWebView should be populated with successive Google query's for each of the names in the database. Each display of the results should stay on the screen until the user pushes the 'back' button, and then the result for the next name displayed.

For Android Developers

When the search button is touched a new activity should be launched which implements a SectionsPagerAdapter to implement a standard android UI framework called "Scrollable Tabs with Swipe" (When creating a new Eclipse project you can select "Blank Activity" and then select "Scrollable Tabs with Swipe" as a navigation type to see a complete example of this UI type).

You will then create a single Fragment class with an android WebView widget in its layout that accepts (at least) a string as an input argument. For each name in the database you will create an instance of this fragment with a name as the input argument and associate it with a section of the SectionsPagerAdapter. Thus as you scroll to the right from page to page you should see successive searches of the names in your database. Pressing the 'back' button should cause the app to go back to the first activity with the 'populate' and 'search' buttons.

Although it will not be enforced or marked if you would like a small technical challenge you can think about how one might ensure that the searches for only the pages to the left and right of the currently visible page could be completely finished loading before the user chooses to scroll to it.

For All Developers

1. We would like everyone to load the web searches inside their app, for this assignment please refrain from launching a separate browsing app on the phone.
2. The official Google programmatic search API is no longer useful as it only allows 25 searches a day. For the purposes of this assignment a Google search can be done by simply loading your Webview with a URL that looks something like www.google.ca/search?q=this+is+my+search

3. **Due date**: Tuesday February 25th, 6pm, marked out of 10, 0.5 marks off every hour late. Submit your solution on Blackboard portal, uploading a file under the Assignment P4 item.

What to submit:

1. Android developers: a zip file containing your final Android application file (.apk); use your student number as the filename. Also submit the complete eclipse project directory in a separate zip file.
2. iPhone developers: you must submit the complete project directory, including source, in a zip file. Use your student number as the filename. Please do your development on the 5.0.2 version of the SDK, and make sure that you haven't included any files by reference. In fact, please test your submitted zip file before sending it in.