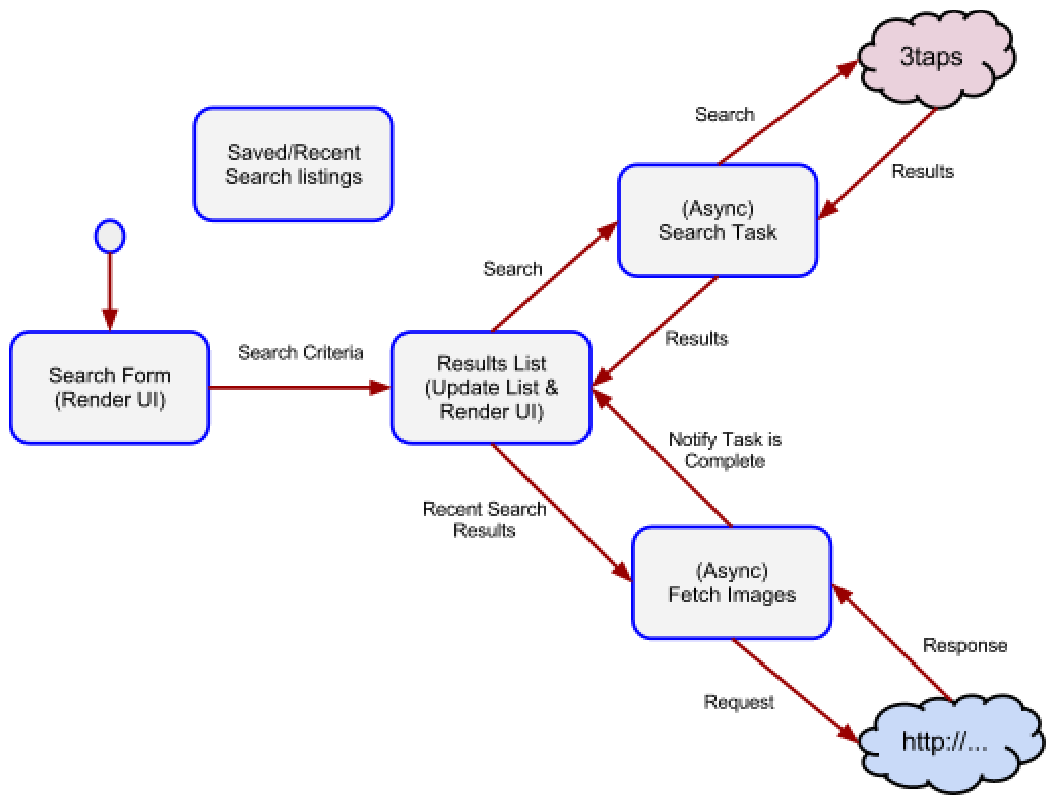
CSI319-51
Assignment 3
Due 19 November 2012

Assignment 3

Continuing from the previous assignment, you should have the following workflow implemented:



By now you should be comfortable with the development environment, framework, and APIs. Going forward, it only makes sense to let you use your best judgement in *some* of the design decisions.

Part 1

Having to continuously fill out the search form sucks and would piss off almost any user. We should at a minimum, provide a means to save the search criteria.

To keep this simple, we need a new Activity (MainActivity.java will do) to present a list of saved searches. The way you present this View is up to you. I chose an Expandable List View as I store Recent as well as Saved searches. I suggest using a List View to get started.

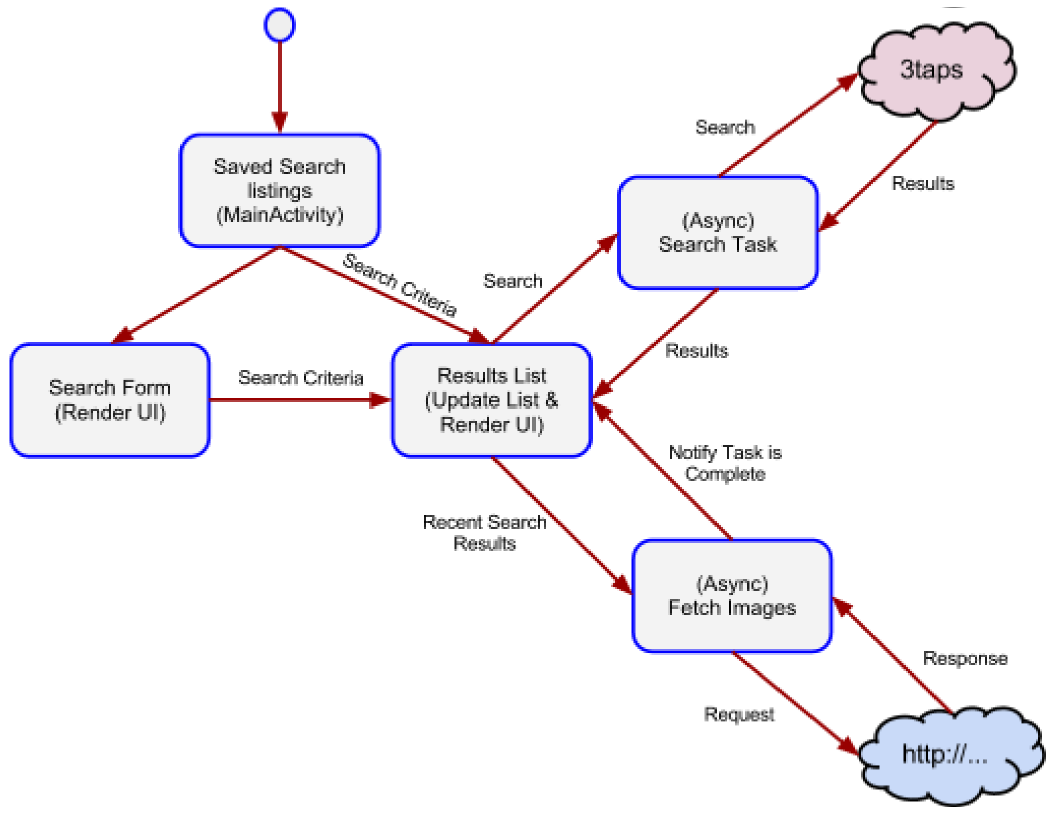
You will want to make sure your application's entry point is MainActivity which contains a list of saved searches. Within this View, provide a means to get to the SearchForm for when the user want to perform a new search. You can supply a button, use tabs, or go down the path of using the experimental ViewPager... it is your choice.

Sooo... What do you need?

- Activity to display Saved Searches (MainActivity.java)
- Layout for the Saved Searches

No Fragment?!?!? Yeah, no fragment. This will eventually serve as *the* Activity for your application when on a tablet layout. For this assignment, this Activity will use an Intent to start the SearchForm Activity if the user wants to create a new search, or start the ResultsList Activity if the user wants to execute a saved search.

A high level workflow is as follows:



Part 2

Now you should have an empty list of saved searches with some way of getting to the SearchForm Activity. In the SearchForm, you need to add another button to allow the user to "Save" the search. I suggest "Save & Search" and "Search Only" buttons. You must provide a way for the user to choose to save the search or search only.

Upon either onClick event, the `ResultsListActivity` will be started with an Intent. You will implement the "save" action in Part 3.

Part 3

For this assignment, you will only be required to store "saved" searches.

Create a sub-package called repository (edu.champlain.csi319.findstuff.repository). This is where you will provide the means to interact with your datasource (SQLite in this case). You will need to create an SQLite database with a single table. Within this table, you need to provide a column for a unique id as well as each property in `SearchCriteria` that is needed to repeat a search. More than likely this will include keyword, category, subcategory, country, and state/province.

Create a sub-package called provider (edu.champlain.csi319.findstuff.provider). This is where you will provide the methods/mechanism for your application to access the data through your repository. Leverage Android's `ContentProvider` to expose methods to insert, query, update, and delete items from your database.

The benefit of using a `ContentProvider` is that it allows you to later share your application's data with other applications. With that said, you will need to supply a Contract class which simply defines constants for the Uri, Table, and Column names.

Use a `ContentResolver` to use the methods implemented by your `ContentProvider` in SearchFormFragment for saving a search, and into the MainActivity from Part 1 for displaying saved searches.

Insert a saved search with something like:

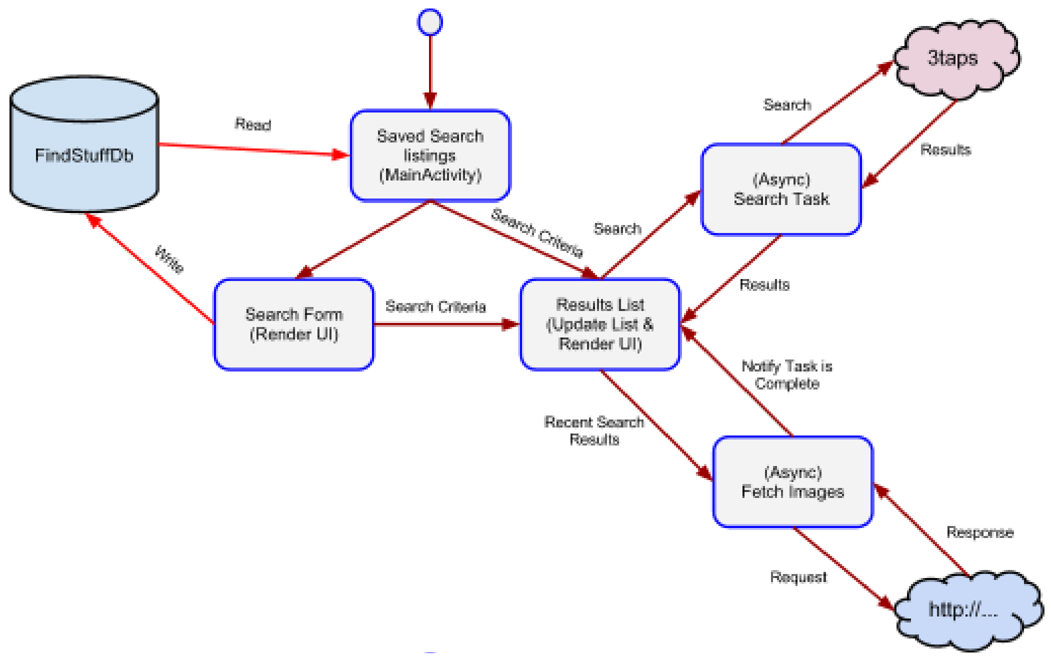
```
getContentResolver().insert(FindStuffContract.SavedSearches.CONTENT_URI, criteria);
```

Where criteria is of type `ContentValues` and has been mapped from `SearchCriteria`.

Get saved searched with a `Cursor` with something like:

```
Cursor cursor = getContentResolver().query(ContentUris.withAppendedId(SavedSearches.CONTENT_URI, id),
null, null, null, null);
```

A high-level workflow:



By now, your project should look something like this:

edu.champlain.csi319.findstuff.content
- Categories.java
- Locations.java
- LoadImageUrlTask.java
- SearchTask.java

edu.champlain.csi319.findstuff.model
- SearchCriteria.java
- SearchResult.java

```
*edu.champlain.csi319.findstuff*.provider
    FindStuffContentProvider.java
    FindStuffContract.java
*edu.champlain.csi319.findstuff*.repository
    FindStuffDb.java
    SavedSearchesTable.java
*edu.champlain.csi319.findstuff*.ui
    *edu.champlain.csi319.findstuff*.searchform
        SearchFormActivity.java
        SearchFormFragment.java
    *edu.champlain.csi319.findstuff*.resultslist
        ResultsListActivity.java
        ResultsListFragment.java
        ResultsListItemAdapter.java
    MainActivity.java
*edu.champlain.csi319.findstuff*.util
    HttpImageLoader.java
*edu.champlain.csi319.findstuff*.widget
    HashMapAdapter.java
```

Part 4 (25% Extra Credit)

Currently, the SearchResult object is Parcelable and contains a Bitmap for the listing's thumbnail along with other data. By making this Parcelable, we can serialize the SearchResult object and pass it to another process (like another Activity). We can also serialize it when saving state (like when another app comes to the foreground).

Unfortunately this is careless. The amount of memory used to store 100 72x72 pixel Bitmaps plus any other String objects can become huge... much more than what an embedded system would want to place on the stack.

Also, if your application went to the background long enough, or while a resource-hungry app came to the foreground, this List<SearchResults> would eventually be garbage collected. That means that when the user resumes your app, all those images would have to be re-fetched.

This is where caching comes in.

To keep this simple, implement Memory caching to store the Bitmaps. You can leverage Android's LruCache<Key, Value> and use the image url as the key and Bitmap as the Value. Once you have caching up and working, remove (comment out) the calls in SearchResult.java which read and write the bitmap as a Parcelable:

```
        thumbnail = in.readParcelable(Bitmap.class.getClassLoader());
        dest.writeParcelable(thumbnail, 0);
```

Assignment Submission

Submit an exported archive (zip or tar) of your project in Angel. Please name the archive as follows:
***lastname*_assignment3.tar** or ***lastname*_assignment3.zip**

Do not hesitate to ask questions!