CMPE 230 Spring 2015 - Project 3

In this project you are going to implement a smart image gallery program that will not only display images but also label (tag) those images automatically. You will implement a Qt based graphical user interface and provide a simple navigation in your program as well. Think of this assignment as an extension to your second project and use the scripts you produced for the second project to complete this project.

Requirements

In order to complete this project you will need the following:

1- An Amazon AWS account
    a. See http://aws.amazon.com
2- AWS command line interface
    b. See http://aws.amazon.com/documentation/cli/
3- An imagga.com API user.
    c. Go to http://imagga.com and sign-up for a free account. The code related to imagga will be provided for you with the description. You will need your own imagga API keys.
4- Perl, WWW::Curl package for Perl
5- Qt

Project Details

Your program will have a simple structure as shown on Figure 1. There will be a top menu, side navigation, a main content area and a status bar at the bottom.

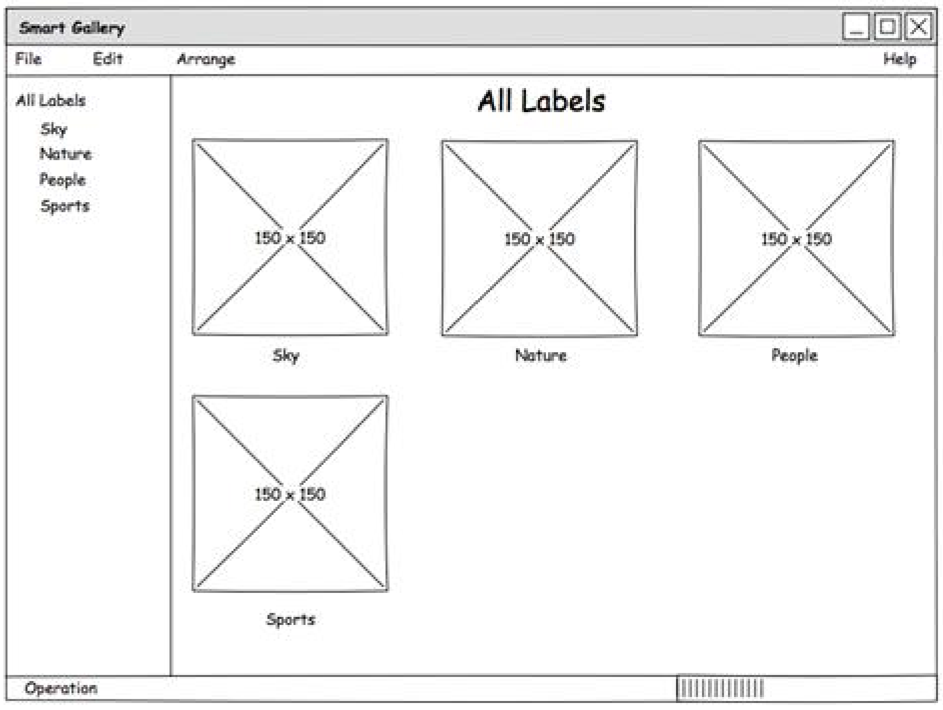

Figure 1 - Basic Structure

The sidebar will have a navigation consisting of an “All Labels” link and links for individual labels in your gallery. If “All Labels” is selected, the main content area will display a collection of images that represent the labels, as shown in Figure 1. These images may be randomly selected from the images in the gallery as well as the sample images retrieved for those labels.

The status bar at the bottom of the interface will display the actions being performed by the program. These actions will be explained later in this text.

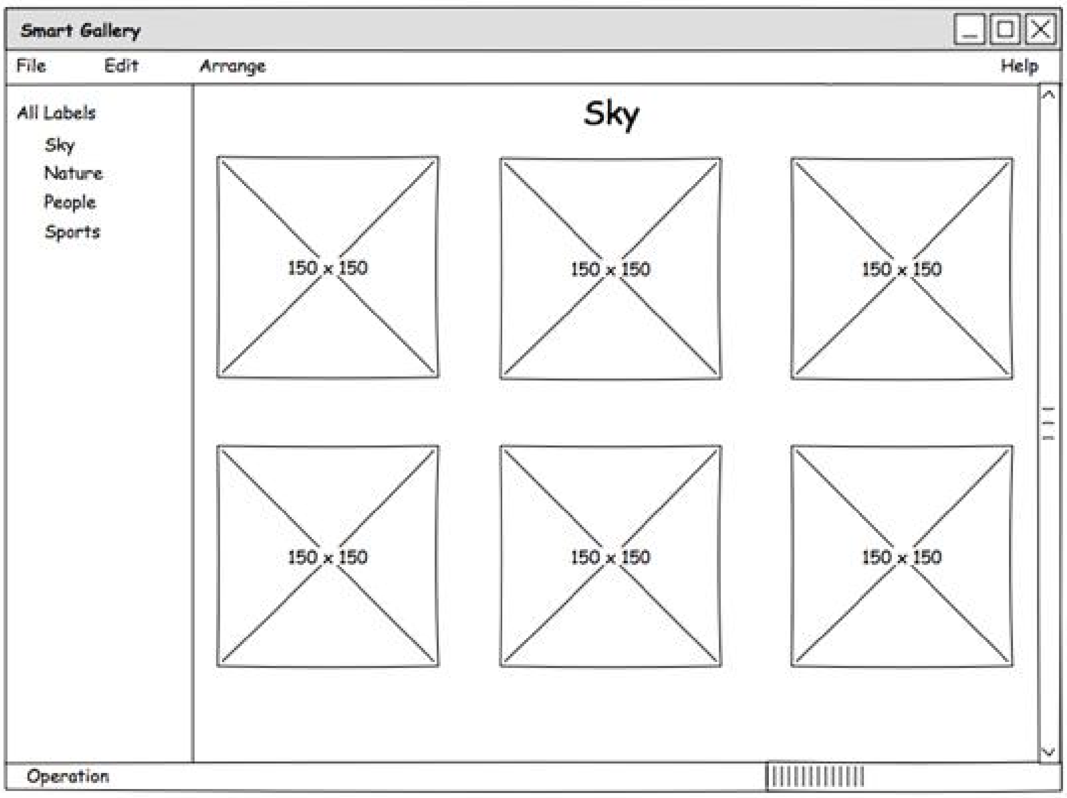

Figure 2 - A Selected Label

When the user clicks on a label from the sidebar, or from the “All Labels” collection in the main content area, the images with the selected label will be displayed in thumbnail form as shown in Figure 2. If there are more images than the main area can handle, you should make the area scrollable. Finally when a user clicks on a listed image, it will be displayed so that it covers the whole main content area. The displayed image should fill the width of the content area. You should protect the aspect ratio (keep the ratio of the width of the image to its height the same) when displaying the image. Figure 3 represents a clicked image.

The contents of the menu items should include at least the following:

- Add File: Add an image file to the gallery. (you can assume .jpg extension)
- Add Folder: Add the images in a folder to the gallery.
- Settings: Open a dialog that contains settings for the program.
- Export: Export the gallery contents to a file.
- Import: Import data into the gallery. (Should merge with existing gallery not overwrite)
- Select All: Select all the items displayed in the main content area (images or labels)
- Remove: Remove the selected items from the gallery (not delete them from the actual storage though, just remove from the gallery)
- Arrange:
  - By Name: Arrange the content in the main area by their name
  - By Date: Arrange the content in the main area by their creation date
  - Sample First: For the label listings, show the sample images first

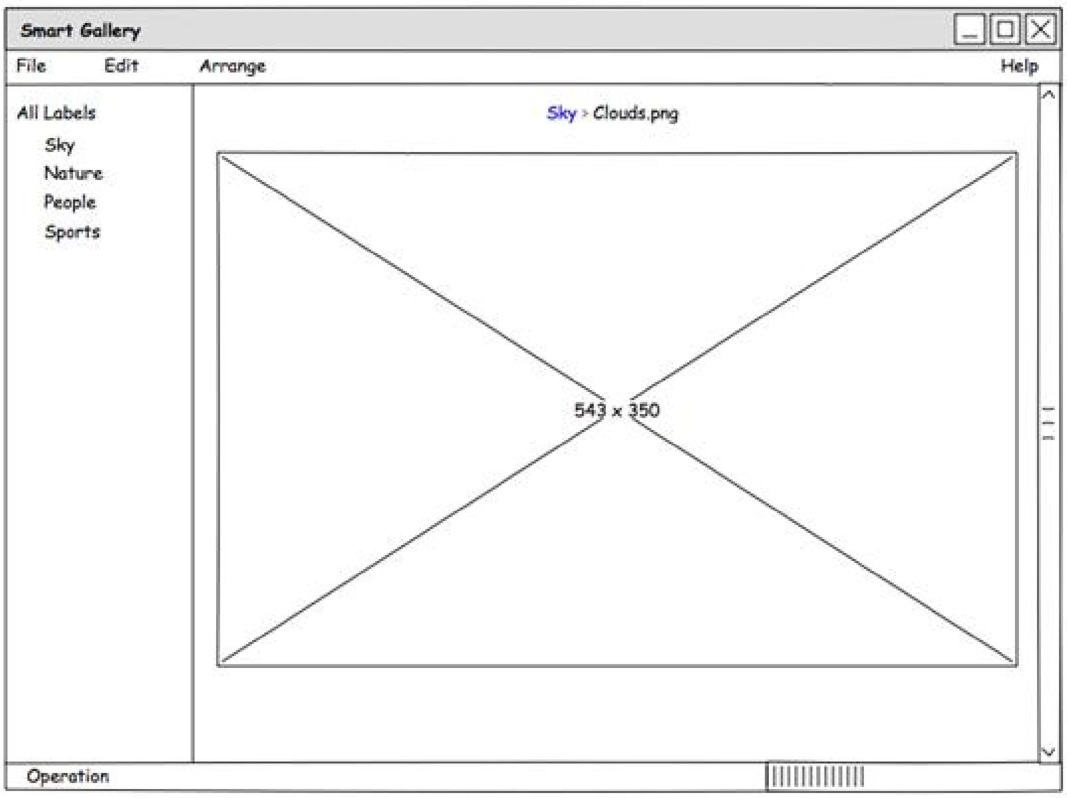

Figure 3 - Detail View of an Image

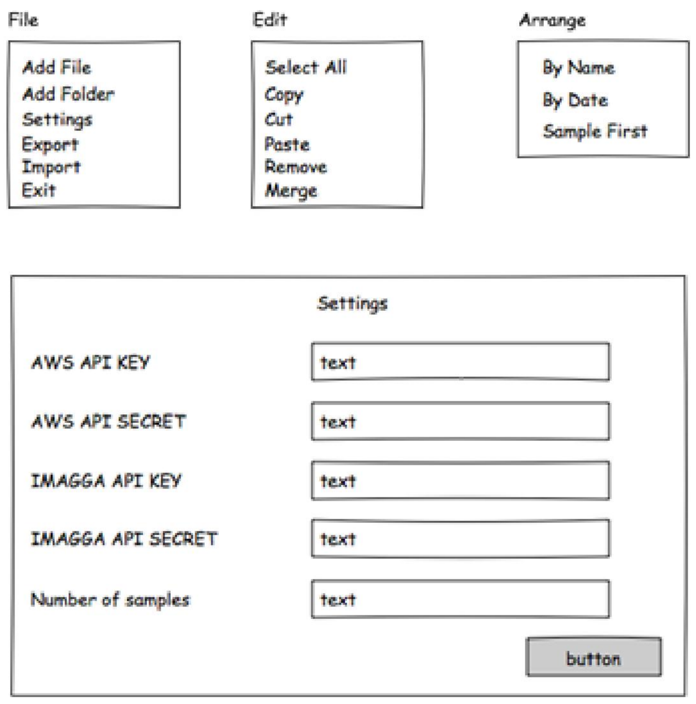

Figure 4 - Possible Arrangement of Menu Items and Settings Dialog

Figure 4 represents a possible arrangement of the menu items and a minimal settings dialog. It is up to you to determine if you need any other dialogs or layouts, and you are allowed and in fact encouraged to include more menu items than listed here.

Implementation Details

In this project you should use the scripts you generated for the previous project. In the previous project you should have generated at least two files that contained label information and label-image correlation. These two files (or more if you used more) will make the database of your gallery program.

Most of the scripts you have produced for the previous project will be used when the user clicks “Add File” or “Add Folder” menu items. Using these commands user will point to images or folders and:

1- An AWS instance will be started if none is running
2- Those images will be uploaded to the AWS instance
3- Those images will be labeled in the instance (labelimage.pl script should be run on the AWS instance)
4- The label information will be downloaded
5- The AWS instance will be stopped if you want (you may want to keep it alive for a while though)
6- Downloaded information will be inserted (merged with) into the local gallery.

You should display these steps on the status bar at the bottom as they are being performed, possibly with a moving indicator, so that we can see the program is working and did not die in the background.

Import and export functionalities must use the same file format so that exported data can be used to import the gallery. When you import data into the gallery, do not overwrite your entire gallery, but merge the new data with the old one.

For the sake of compatibility between each of your projects, you can use the following format:

```
gallery.export

[LABELS]

<LabelName>, <Sample1>, <Sample2>, …, <SampleN>
<LabelName>, <Sample1>, <Sample2>, …, <SampleN>

[IMAGES]

<ImagePath>,<LabelName>,<LabelName>,…,<LabelName>
<ImagePath>,<LabelName>,<LabelName>,…,<LabelName>
<ImagePath>,<LabelName>,<LabelName>,…,<LabelName>
```

Notes:

1- You don't have to display and accurate progress bar at the status bar (as a matter of fact you don't have to display a progress bar at all)
2- It would be desirable to generate a help page if you have more menu items than the minimum requirements.
3- Feel free to produce additional files, menu items, and interface elements (keep the number reasonable though).
4- Look at the QT documentation. http://doc.qt.io/
5- There are various books on https://www.safaribooksonline.com which you can access when inside the campus or using a proxy from outside. Example book "C++ Gui Programming with QT 4"

- Basically your gallery should be synchronized with the AWS instance. You should handle that every image that have been added to the gallery is also uploaded to the AWS instance. The folder structure at the AWS instance is up to you to decide. How you relate local images with the corresponding images on AWS is also up to you to decide.
- You can assume that only a single client program will be synchronized. (This means only one program may change the gallery at a time unlike for example Dropbox which enables multiple changes from multiple locations)

What to submit:

1- Your (well written and commented) code.
2- A brief report that describes the implementation you have done. Keep this report BRIEF: no more than 2 pages, 1 page is preferred.