

# Web-based Image Management System

CSCI4140 - Individual assignment #1

Version 1.0, 2014 January 20.

## Abstract

Online photo management systems are abundant in the Internet, such as Picasa and Flickr. In this assignment, you are going to implement a simplified system with the techniques introduced during the lectures.

## 1 Your Task

Your task is to implement a private photo album management system. The flow of the system is described in Figure 1. We will not spend any context to further explain how the system works because Figure 1 is clear and is self-explanatory. Basically, the system can be summarized into the following points.

- The system only allows one user to log in and there is only one album. Though you can enhance the system to support more users or more albums, there is no bonus marks for extra work done.
- With a successful login attempt, the user is allowed to:
  - view photos stored in the album.
  - add, or upload, photos into the album.
  - remove any existing photos in the album.



A logout action will remove the last two privileges from the users, i.e., the user can still view photos in the album.

- The system adopts two different kinds of permanent storage:
  - The database. It should, at least, stores the upload time as well as the pathname to every stored photo.
  - The album folder. It should, at least, stores the uploaded photos.

## 2 Development and Demonstration Environment

We require you to work with the following constraints:

- You should implement the system using **Perl** and **HTML** only. For the sake of fairness, no JavaScript is allowed.
- For Perl, on top of the basic features, you are only allowed to use the **CGI** and the **DBI** as well as the relevant **DBD** modules.
- For HTML, you are welcome to write **ugly** interfaces. Remember, beautiful and fancy interfaces do not result in high marks, and you are not suggested wasting the time in producing fancy interfaces.
- Our primary development environment is OpenShift - <http://openshift.com>. Please use the Perl cartridge with the MySQL database support as your development platform. Please make sure that your finalized work can work on OpenShift even though you are not using OpenShift as the development environment.

**Demonstrations will be our only grading process.** During the demonstration, we will restore your submitted git repo, HTML files, etc. into an OpenShift Perl cartridge.

Your system should be running on (1) Firefox version 4.0 or above or (2) Google Chrome 32.0 (latest stable release at the time of this writing) during the development process and

the demonstration. Last but not least, during the demonstration, we may be using Mac, Linux or Windows as the client platform.

## 3 Task Description

Before we describe all the tasks involved in Figure 1 one by one, we introduce you to the installation script requirement.

### 3.1 Installation Script

An installation script (or the re-initialization script) is a Perl program which will initialize your Photo album system. It is a typical practice in the open-source realm that a web-based system comes with an installation script.

Note that since that script can re-initialize the entire system, it is not wise to add a URL on any page in the system. Rather, we will hard code the URL to the script's interface as follows.

`http://[URL to Your System]/reinit.html`

#### 3.1.1 Interface of “reinit.html”

An installation script would not be loaded immediately; a confirmation page will be shown and is depicted in Figure 2.

- When the “Yes” button is clicked, the system will be re-initialized.
- When the “No” button is clicked, the browser will be sent back to the “Login Interface”.

#### 3.1.2 The installation script

Basically, the installation script will create a fresh system, and its function includes:

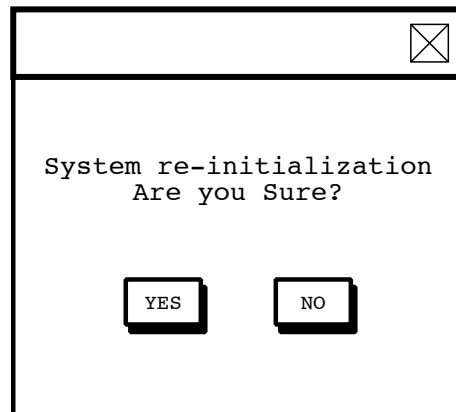


Figure 2: Confirmation page before the system is re-initialized.

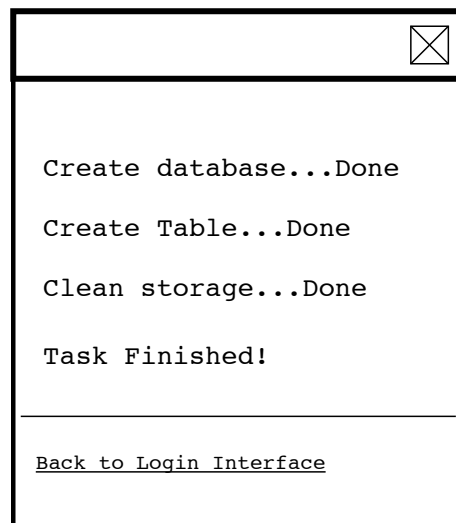


Figure 3: Re-initialization is done.

- Drop all the tables of the database (if it exists) of the photo album system.

Note that you are recommended to create a configuration file which is for storing the database login name and password. Then, during the demonstration, we can modify the least number of files in order for your system to run.

- If a database of the photo album system does not exist, create it.
- Create all the required table.

- If your system needs seed data in the database, e.g., the database rows for the default login credentials, then insert them.
- Clean the permanent file storage on OpenShift.

After the installation has been completed, the page should provide a link for the user to go back to the login interface, as shown in Figure 3.

## 3.2 Login Interface

A diagram of a login interface window. It has a title bar with a close button (X). The main area contains labels "Username" and "Password" next to input fields. The "Username" field contains the text "csci4140". The "Password" field contains four dots. Below these is a "Submit" button. At the bottom, there is a link labeled "View album (read-only)".

Figure 4: The minimum requirement for the login interface.

The login interface should allow the user to provide his/her credential in order to log in the system. The minimum requirement for the login interface is depicted in Figure 4.

## 3.3 Login CGI program

Although the login CGI program is not explicitly shown in Figure 1, the box “Correct Credential” implies a CGI program that decides whether the login attempt is legitimate or not.

<b>Program input</b>	At least, login name and password; they cannot be empty.
<b>Program output</b>	You have the freedom to choose your own way to implement output.

Note that the login CGI program should behave in the way described in Figure 1 on page 2. On top of that, the program should avoid the following two problems, which were mentioned during the lectures:

1. Multiple login requests generated by reloading.
2. Embedding login or session information in URLs.

### 3.4 Display panel

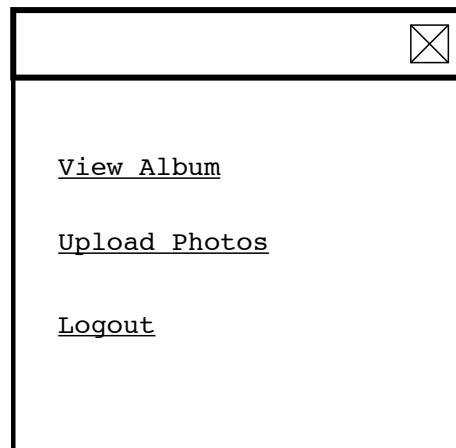


Figure 5: The minimum requirement for the display panel interface.

The display panel should contain a list of links that directs the user to three different components or functions of the system. Let's take Figure 5 as an example:

1. The link "View Album" takes the user to the "*Album display interface*".
2. The link "Upload Photos" takes the user to the "*File picking interface*".

3. The link “Logout” logs the user out.

Note that Figure 5 is just a minimum setting.

### 3.5 Album display interface

Dimension  x

Sort By   
File size  
Name  
Upload time

Order   
Ascending  
Descending

☒ img\_001.jpg ☐ img\_002.jpg ☐ img\_003.jpg ☐ img\_004.jpg

☒ img\_005.jpg ☐ img\_006.jpg ☐ img\_007.jpg ☐ img\_008.jpg

This is a description.

Page  of 10

Figure 6: A sample design of the album display interface.

The album display interface provides two functions: to view the album and to remove photos from the album. Figure 6 is just an example layout. Although you have the freedom to implement your own one, your implementation is subject to a set of requirements.



- **Your system has to validate every input provided by the user through the interface only.**

- **Display photo array and its dimension.**

The album display interface should display all the available photos in an array. The number of rows and the number of columns are allowed to be changed. The valid range for both numbers is from **1** to **9**, inclusively.

**Hint.** You can choose any types of “input” element for the input to the dimension.

- **Pagination.**

The maximum number of photos shown in the interface is defined by the “Row input” and the “Column Input”. E.g., in Figure 6, the maximum number of photos shown is 8.

If an album contains more photos than the maximum number, then an album should be broken into pages. Let us consider Figure 6, if there are 76 photos in the album, then the album will be broken into 10 pages.

The interface should tell the user which page he/she is currently look at and the interface should provide a feature for the user to navigate to different pages.

- **Sort the display.**

The photos are allowed to be sorted by three of their characteristics:

- sorted by size, measured in bytes;
- sorted by photo name, lexicographically; or
- sorted by upload time. For the sake of convenience, you can store the time in terms of the Unix timestamp, i.e., the number of seconds since the Epoch (00:00:00 UTC, Jan 1, 1970).

On top of that, the photos can be sorted in either ascending order or descending order. Note that you are free to implement any tie-breaking rules.

Together with the pagination function and the array dimension control, you have to implement **two extra requirements**.

- After any change in the sorting configuration or any change in the array dimension, the album display interface will be showing the first page.
- Changing the page number must not change the sorting configuration nor the array dimension.

- **Display thumbnails only.**

In the display array, you are required to display a thumbnail of the target photo only, instead of resizing the display of the target photo. That is to say you have to create a file that is smaller than the original photo. The maximum width and the maximum height of a thumbnail are both 100 pixels, using the aspect ratio of the original photo.

- **Display description of the photo.**

When the user moves the mouse over a photo, a description box should be shown with the text uploaded in the “*File picking interface*”. The parameter “**title**” of the “**img**” tag should be used. Note that the description may be empty.

- **Select photos to remove.**

You are required to allow the user to select photos, which are showing in the display array, to be removed. You have to freedom to handle any funny scenarios, e.g., changing the page display after the user has selected a photo, without really deleting it. When the user chooses to remove the selected photos, you have to remove every data about that photo.

- **Click and display the target photo.**

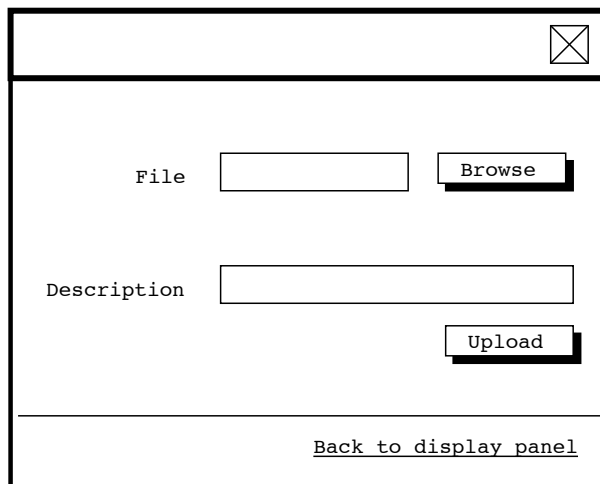
Last but not least, clicking on the thumbnail of a photo will display the original photo.

### 3.6 Album display interface (read-only)

The system should support a read-only mode. Comparing with the original version, there are two differences:

1. The read-only mode will not provide the photo-removing function.
2. When going back to the previous page, the read-only mode will lead the user back to the login page, instead of the display panel.

### 3.7 File picking interface



The figure shows a wireframe of a file picking interface. It is enclosed in a rectangular border. In the top right corner, there is a small square button with an 'X' inside. The main area contains two rows of controls. The first row has the label 'File' on the left, a rectangular text input field in the middle, and a 'Browse' button on the right. The second row has the label 'Description' on the left, a wider rectangular text input field in the middle, and an 'Upload' button on the right. At the bottom of the interface, there is a horizontal line, and below it, the text 'Back to display panel' is centered.

Figure 7: A sample design of the file picking interface.

The file picking interface can be reached from the display panel, and Figure 7 shows a sample design. In addition to the file-choosing function, the file picking interface should also allow the user to input a text description of the photo. The restrictions on the text description are as follows.

1. The maximum length is 50 characters.

2. You have to guarantee a viewable, verbatim output. E.g, if a user inputs the string “>\_<”, then you should convert the string to “&gt;\_&lt;”. For simplicity, we display the characters that you have to convert as follows.

- ‘<’ → “&lt;”
- ‘>’ → “&gt;”
- ‘”’ → “&quot;”
- ‘, ’ → “&#39;”
- ‘&’ → “&amp;”

When a user chooses to upload a photo, this interface will drive the actual uploading process. Although it is not explicitly shown in Figure 1 on page 2, such a process is implemented as a CGI program. The program should work with the following set of requirements:

- The maximum file size of each photo is restricted to be 1,000,000 bytes. The upload process fails when such a requirement cannot be fulfilled.
- The filename of the uploaded file must only carry one of the following extensions: jpg, gif, or png. The upload process fails when such a requirement cannot be fulfilled. By the way, no animated GIF nor animated PNG files would be uploaded.

To keep things simple, the filename of the photo only contains lowercase-alphabet, digits, and underscores. During the demonstration, you will not be provided with filenames violating the above restrictions. Yet, for your convenience sake, you can add extra handling over the name, e.g., converting all filenames to lower-case.

Last but not least, every upload image will be stored, using the filename provided.

- The program should also check whether the upload file is really an image or not. The Linux command “identify” can help you and it is available inside the Perl cartridge. The upload process fails when such a requirement cannot be fulfilled.

- The CGI program should generate a thumbnail of the original photo. The Linux command “`convert`” can help you and is available in the Perl cartridge. There is no naming restriction imposed on the filename of a thumbnail.

### 3.8 Duplication handling interface

Figure 8: A sample design of the duplication handling interface.

This page allows a user to be notified the case when a file carrying a **duplicated filename** is uploaded to the system.

- First, the interface is triggered only when there is a file with the same filename exists in the system.
- During the duplication check, **only the filename is considered** even though the existing image and the uploaded image may carry different content.

The user has three options to choose from, as depicted in Figure 8.

1. **Overwriting the existing file.** This requires the system to remove the existing image as well as the corresponding thumbnail.

Since it is possible that the user will keep on encountering duplicated files when he/she chooses to rename the filename of the uploading image, it is needed to display the duplicated name involved. Then, the overwriting operation will be overwriting the existing file displayed.

Certainly, the description of the existing file will be replaced by the new one.

2. **Renaming the uploading file.** You have to provide an input field for the user to input the new filename of the uploading file. Note that since the uploading file has passed the “file-type checking”, it is not wise to allow the user to change the file extension. As shown in Figure 8, the file extension is not allowed to change.

For the file description, you are free to design whether to use a new one or the previously-uploaded one.

3. **Canceling the upload.** This allows the user to withdraw the upload request.

## 3.9 Permanent storage

The permanent storage means two things: a database and an album folder.

### 3.9.1 Database

You are suggested saving the following things into the database:

- Credential of the user.
- Information of every photo uploaded, including its filename, its file size, its upload time, its description, and the pathname of the corresponding thumbnail.
- Session information.

You are suggested to use MySQL with the DBI module in Perl. MySQL as well as its management interface, `phpmyadmin`, are available in OpenShift.

### 3.9.2 Album folder

The album folder is just a directory storing the uploaded photos and the corresponding thumbnails. Note that OpenShift has created the storage folder for each cartridge. Please refer to our tutorials for details.

## 3.10 Session management

A feature that is not shown in Figure 1 on page 2 is the session management. Note that you **are highly recommended to make use of the HTTP cookies** as the approach to implement session management.

### 3.10.1 The “*kick-you-out*” requirement

While the user is visiting the system, the system has to guarantee the presence of the session information. The list of the involved parts of the system is given as follows.

1. Display panel;
2. Album display interface; and
3. File picking interface.

If the session information is missing, then the system should refer the user to the login interface directly, i.e., *to kick him/her out*. Then, the login interface should ask for the correct credential from the user.

### 3.10.2 The “*welcome-back*” requirement

Nevertheless, if a user carrying a valid HTTP cookie, visits the login interface, then the login interface should lift the login process, i.e., *to welcome the returned user*. Then, the followup action is up to your implementation, the system can either forward the user to the display panel or display a link to the display panel. On the other hand, you are free to set the

duration of the expiry time of the cookie held by all users. Yet, for the sake of a smooth demonstration, please set the cookie expiry to a reasonable time period, say one hour. Last but not least, the cookie expiry time **must be set**. The system must be able to welcome returning using after the browser is re-launched.

Note that the read-only version of the album display interface does not require the presence of the session information.

### 3.10.3 The “*multiple login*” requirement

It is required that the system allows the same user to log in from different locations or the same location with different browser instances.

### 3.10.4 The “*logout*” requirement

The logout action should remove the corresponding login session information from both the client and the system of the corresponding browser instance.

## Assignment Submission

Please follow the submission guideline posted on the course homepage:

<http://www.cse.cuhk.edu.hk/csci4140/>

**Deadline - 23:59, February 28, 2014 (Fri).**



<b>Milestones</b>
<b>0. Installation script - 5%</b>
Does the confirmation interface function normally?
Does the installation script function as expected?
<b>1. Login interface - 5%</b>
Does the interface contain all necessary component?
Failed login attempt.
Successful login attempt.
<b>2. Album display interface - 25%</b>
Photo array and dimension changing
Page breaking
Sorting
Thumbnail
Photo description
Photo display
Read-only mode
<b>3. Photo removal - 15%</b>
Photo selection
Are selected photos deleted?
<b>4. File picking interface - 30%</b>
File size, extension, and type checking
Description checking
Is the photo uploaded?
Is the description uploaded?
Is the thumbnail generated?

<b>5. Duplication handling interface - 10%</b>
Overwriting an existing file.
Renaming the uploaded file.
Canceling the upload request.
<b>6. Session management - 10%</b>
Kick-you-out
Welcome-back
Multiple login
Logout

—END—

## Change Log

Time	Affected Version(s)	Details
2014 Jan 20	NIL	Release version 1.0.