Assignment 1 - Web Instagram

Deadline: 23:59, 2018 Feb 13 (Tue)

TA-in-charge: Yue Wang

Introduction

Your task is to implement an online photo editor as well as an album for photos uploaded to the system. Let us call that the "**Web Instagram**". The goals of this assignment are as follows.

- To demonstrate your understanding in implementing web-based applications through CGI.
- To learn how to write web-based applications with Python.
- To make use of open-source, readily available image processing libraries in order to make your task easy.

Note that we are going to implement a minimal, yet viable, system. We aim for an easy task for whom who did not have prior experiences. If you are an experienced web application programmer, please bear with the restrictions of this assignment as we aim for a common implementation for all students, with varying level of experience.

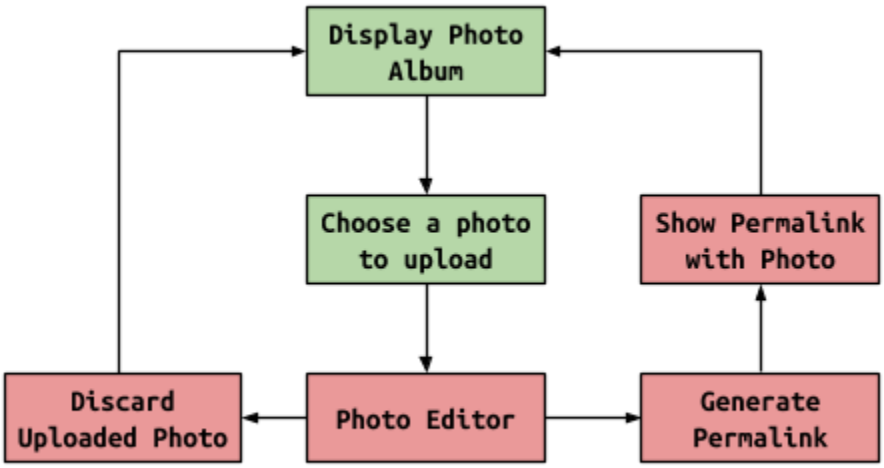


Figure. Flow of the "Web Instagram" System. Red boxes are system states that are accessible only after the user has uploaded a photo. Green boxes are system states that are accessible before a user uploads any photos.

The above figure describe the basic workflow of the system. To be specific:

- **Login is required**. Although the system is open to any users, only users with login can upload and edit the photos. More detail requirements about access control will be specified later.
- The web application starts with the album showing all the uploaded photos with permalinks. (logged-in user can view all public photos and his/her uploaded private photos, while non logged-in user can view public photos)
- Then, a user can upload a photo and specify the upload mode, either private or public.
- After a user has uploaded photo, the user can edit the photo with the **Photo Editor**. We will describe the requirements for the photo editor later.
- Inside the photo editor, the user can choose either to:
  1. edit the photo;
  2. discard the uploaded / edited photo; or
  3. generate a permanent link for the (edited) photo.
- If the user chooses to discard the uploaded photo, the photo will be permanently removed from the server.
- Else if the user chooses to generate a permalink for the photo, the photo will be kept on the server permanently with a URL pointing to the photo. Plus, the photo cannot be edited any further.

In addition to the basic workflow of the system, you are required to implement **a user access control and session manage** module.

Implementation, Grading, and Constraints

- Implement the system with Python and HTML only. For the sake of fairness, no JavaScript is allowed on the frontend or the backend sides. Also, server-side application frameworks such as Flask and django are not allowed.
- You are welcome to use packages you like for managing the database, e.g. MySQL and Sqlite3.
- For the HTML-based client, you are welcome to write **UGLY** interfaces. Remember, we grade your work based on the completion of features, not how beautiful your interface is. Of course, user experience is not put into consideration.
- Our primary development and grading environment is **OpenShift**. Please follow our tutorial in setting up a working environment. Although you are allowed to use other environments, you will only grade your work on OpenShift.
- You need to apply for a private repository github account (https://education.github.com/pack) and then place all your codes for this assignment on it. Also set the webhooks to your website in openshift server (then any changes on your github repo will be showed in the website, which will be taught in coming tutorial).
- For grading your submitted assignment, we will use two ways (**both**):

1. We will restore your submitted github repository into an OpenShift image and test the required feature. You need to package your project into a zip file named as "SID_CSCI4140_assignment1.zip" and submit it via the blackboard system.
2. You need to setup your website in openshift server and make sure it can be visited by others. Keep your github repo for this assignment unchanged after deadline (no more commits after deadline) and add the TA as the collaborator (Yue's github account is "ayueei"). We will grade by reading through your readme.md file and inspect your code, and test your website via the link you provide.

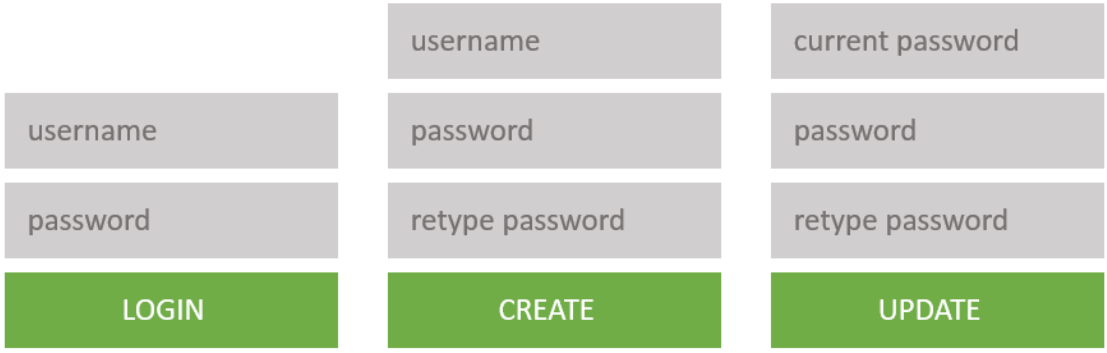
- On the client side, your system should be able to run on (1) Firefox 4.0 or above or (2) Google Chrome 40.0 or above.
- We will be using Windows 10 or Mac OS X as the client platform.

Task 1: Access Control & Session Management

In this task, you need to implement an account registration, a login, and password change CGI programs. Also, you are asked to use Cookies to authenticate a user session.

Access control

You need to create three pages for the user access control, which are account registration, user login, and password change. Examples are given as follows:



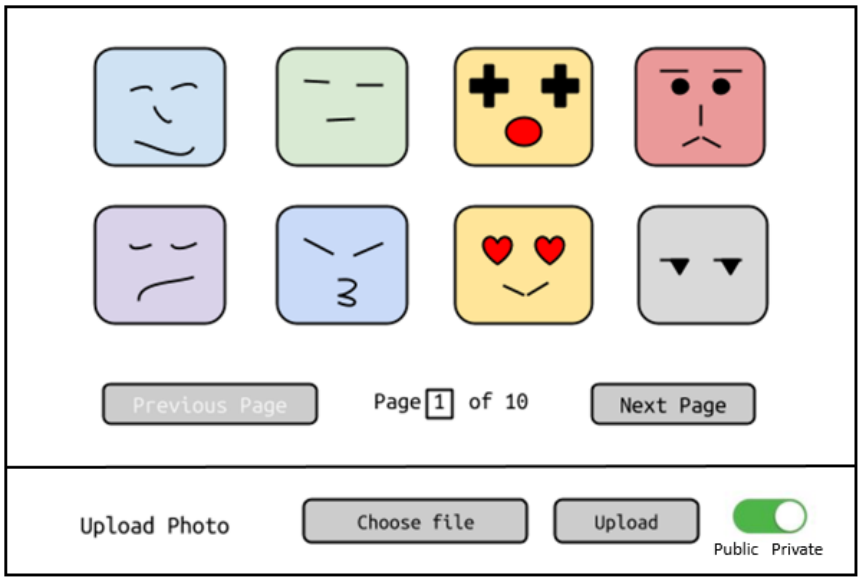
- Requests to these CGI programs should be sent using HTTP POST requests.
- If the username has been used by others, the system will give error prompts.
- You are not allowed to use existing account management package. In other words, you need to build these from scratch.
- To make it simple, we do not require you to use email to valid the account registration.
- After successful login, the user will be forwarded to the index page.

Session Management

You need to implement one cookie that is set if the user successfully logs in and becomes invalid if the user logs out. In the index page, you need to provide a link for login/logout based on whether a user has authenticated or not.

- The index page should include the username for a logged-in user.
- If the login fails, the system will give error prompts and lead back to the index page showing the public photos.
- Note that a user cannot view others' private images.

Task 2: Index page



Index Page. It is responsible to display an album of stored photos with permalinks (upper part of the figure) as well as to provide the interface to upload a photo (lower part of the figure).

The index page is a CGI program providing two functions:

1. Display the stored photos with permalinks, and
2. Display an interface that allows the user to upload a photo.

Task 2.1 - Index page: photo album

There are two main functions: displaying resized version of uploaded photos with permalinks and linking those resized photos to the original photos.

Sort and display photos

The photos are sorted by the creation time, i.e., the time that it is permanent stored in the server with permalinks, in a descending order. That means the latest photo is always on the top.

The album page displays 8 photos at most in any display format you like. (grid or simple one image per line)

Displaying a resized photos

Each photo in the index page is a resized version of the original photo. The maximum width and the maximum height of a resized photo are both 200 pixels, using the aspect ratio of the original photo.

Clicking on the resized photo will show the user the original photo (and it is okay for the browser zooming out automatically when the photo is over-sized).

Note that it is up to you whether to generate a separate thumbnail of an uploaded photo or not.

Pagination

If there are fewer than or equal to 8 stored photos in the server, the index page can display all the resized photos.

If there are more than 8 stored photos in the server, then the album should be broken into pages. Let us consider the above figure, if there are 76 photos in the album, then the album will be broken into 10 pages.

The index page 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.

Task 2.2 - Index page: file upload

The file upload interface is generated by the index page CGI program. When a user uploads a file to the server, a corresponding CGI program will be triggered. Let us call it the "upload CGI program".

Assumptions

- We will upload only one photo at a time. The file must be of the following extensions: jpg, gif, and png.
- There is no need for supporting animated GIF or animated PNG files.
- The name of an uploaded file contains upper-case characters, lower-case characters, digits, hyphen (-), underscore (_), or space characters only.

File type checking

- The upload CGI program first checks if the uploaded file is really a photo. If the file content does not match with the file extension, then the upload CGI program must reject the upload and must notify the user.
- Our file type checking requirement is:

Extension	File Type
.jpg	JPEG image file or MIME type: image/jpeg
.gif	GIF image file or MIME type: image/gif
.png	Portable network graphics (PNG) image file or MIME type: image/png

- Invalid file type examples:

Filename	File Type
abc.jpg	Content is a text file.
abc.gif	Content is a JPEG file.

Hint: The command identify from the ImageMagick can help you a lot!

Specify the upload mode

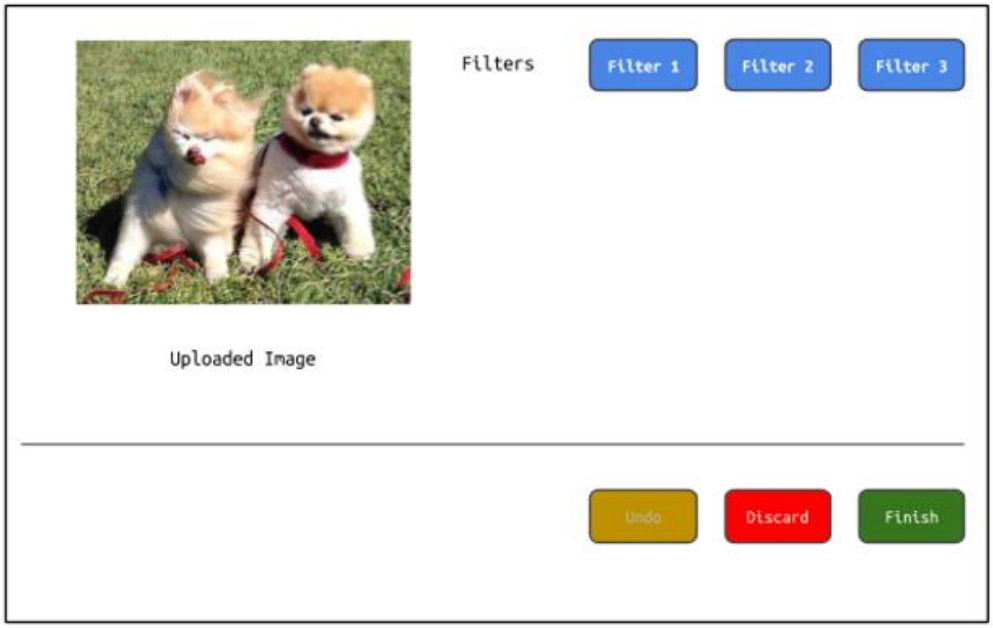
Before clicking the upload button, you need to choose upload mode either "public" or "private". Note that for private photos, they can only be viewed by yourself.

Photo and session storage

Then, we go on with the photo storage if the uploaded photo can pass the file type checking.

- The photo must be stored on the server so as to let the user use the photo editor.
- You can save the image into whatever name. Note that the server may have multiple users uploading their photos, the server must be able to uniquely identify the users with their corresponding photos.
- Last, the upload CGI program should guide the user to the photo editor.

Task 3: Photo Editor



The photo editor is an CGI program. Let us call it the "editor CGI program". This CGI program is only triggered after a user has successfully uploaded a photo.

Editor Display

The editor is depicted in the above figure.

- The editor first shows the original photo. If a filter is applied, then the editor shows the modified photo.
- We provide several filters for editing to the photo. Filters that apply to the entire photo. To make it simple, you can apply one filter for an image. You can make use of the ImageMagick package.

Applying filters

There are 5 filters to be included in this editor. Those filters are implemented by calling the commands "convert" or "composite" in the ImageMagick package.

- **Border**. Adding a black border around the photo.

```
convert <in> -bordercolor <color> -border <size> <out>
```

- **Lomo**. Pretending to be a photo taken by the lomography camera.

```
convert <in> -channel R -level 33% -channel G -level 33% <out>
```

- **Lens Flare**. Adding a flare effect to the photo. Note that this filter requires a pre-defined flare image. It is specified as "lensflare.png" in the following command.

```
convert lensflare.png -resize <width>x tmp.png
composite -compose screen -gravity northwest tmp.png <in> <out>
```

- **Black White**. Turning the photo into a black-and-white photo. Note that this filter requires a pre-defined gradient image. It is specified as "bwgrad.png" in the following command.

```
convert <in> -type grayscale <itm>
convert bwgrad.png -resize <width>x<height>\! tmp.png
composite -compose softlight -gravity center tmp.png <itm> <out>
```

- **Blur**. Adding a low-pass filter to the photo.

```
convert <in> -blur 0.5x2 <out>
```

Undo Edits

The photo editor allows users to undo the effects done by either separate filter / annotation. If a user does not like his/her applied filter / annotation, he/she can undo such an edit by clicking the "Undo" button.

If there is no edits done, the "Undo" button should be disabled. Otherwise, the server reports errors after the user clicks on the "Undo" button.

Note 1. After each edit or undo action, the photo editor should display the most recent photo.

Note 2. During the grading, we will not click the "Back" button to undo the recent edit.

Discarding uploaded photo

- In case that the user wants to remove the uploaded photo, he / she should click the "Discard" button.
- Then, all related storage of the uploaded photo will be removed.
- Last, the user will be forwarded to the index page again.

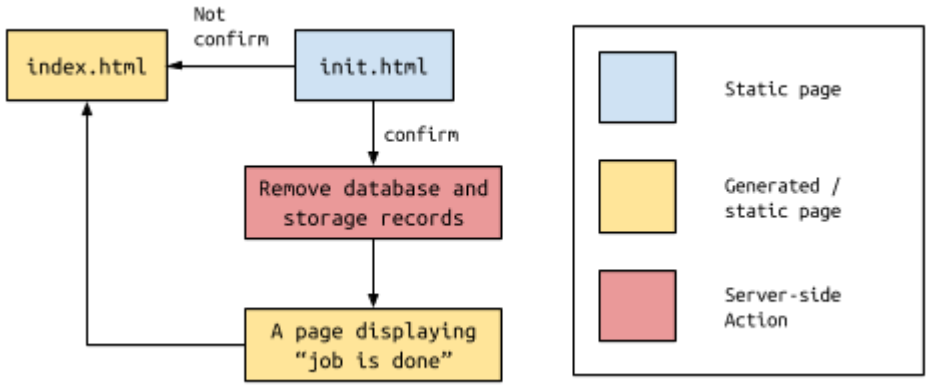
Store the final photo and generate the permalink

- In case that the user wants to confirm the edits of the uploaded photo, he / she should click the "Finish" button.
- Then, the uploaded photo will not be allowed to be edited further.
- Next, a permalink to the uploaded photo should be generated. **The requirements of a permalink are given as follows.**

  - A permalink is not a URL pointing to any CGI programs.
  - A permalink is a normal URL pointing to the finalized photo itself.
  - The permalink link is assumed to be referring to the same server.
  - You are allowed to store the photos with any filename. The only requirement is that you must preserve the original file extension.
- On top of that, the "Finish" button takes the user away from the photo editor. Then, the next page is to display the finalized photo with the permalink printed on the page. Last, there should be a button taking the user to the index page.
- All stored photos with permalinks should be displayed in the index page. Note that since the index page puts the most recent photos in the first page, the photo which is just finalized with a permalink should be able to be seen in the index page unless there are many photos stored permanently at the same time.
- By the way, you are not required to remove any photos with permalinks.

Task 4: System Initialization

Imagine that you are allowing your friends to freely deploy your system. Yet, how can your friends deploy your system without looking into your codes? The answer is to provide a system initialization feature.



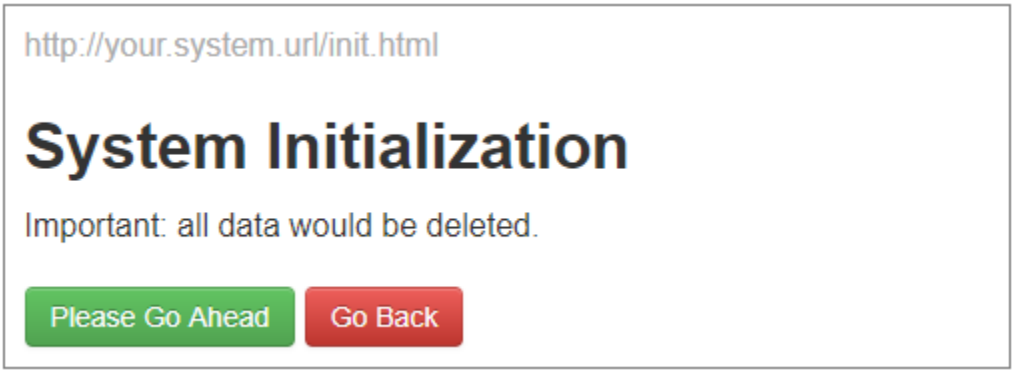
You are required to provide a link that allows the user to initialize the web application. The link should be:

http://your.system.url/init.html

The tasks of the system initialization feature are given as follows.

- The system should be initialized with an “Admin” account. In the first time, the user is asked to change the default password of the Admin account in the initialization step. After the first initialization step, no one except for the Admin with the correct password can access the initialization system.

- Provide an interface for the user to confirm the initialization procedure, e.g.:



- You have to provide a link / button for the user to confirm. It is the "Please Go Ahead" button in the example. Once it is confirmed, the system is re-initialized.
- You have to provide another link / button for the user not to proceed. It is the "Go Back" button in the example. Once it is selected, the user will be taken back to the "index page" of the system.
- The initialization action includes:
  1. Remove all stored photos, if any.
  2. Delete all database records related to the stored photo, if any.
  3. Remove all photos as well as database records related to editing photos.
  4. Create the database tables, if missing.
  5. Create initial database records, if needed.
- Last but not least, after the initialization is completed, your system should provide a page telling the user that the job is done. Also, add a link on that page in order to take the user back to the "index page" of the application.

Additional Requirements

Input Validation

- Your system should validate every input from the browser. Since we are not allowing the use of JavaScript, either the page uses HTML5 input validation or the server validates the inputs.
- If a user attempts to create a new account with a used username, this attempt should be refused with some error prompts.
- For example, you should reject the input "hello world" if an input field is asking for a positive integer.
- For another example, in the index page, the system should display an error message if a user clicks on the "Upload" button when no file is chosen to be uploaded.
- Note 1. We will only input invalid inputs through your web application. That means we are not going to input invalid inputs by carving any HTTP requests.

- Note 2. We would not use the browser's development tool, modifying your code in order to break your system's functionalities / validations.

Readme file

You need to provide a readme file to introduce your project, including:

- The link to your designed website on the openshift server. **(important!!)**
- What each directory stores and their corresponding functionality (for each directory and files).
- Briefly introduce the procedure of building your system, including some key components, e.g. which database package you use.
- Which parts you finish very well and want to request for some bonus and which parts you have not fully completed and give some reason.

Other notes are:

- You need to write it in the form of Markdown so that it can be well viewed at github.
- Most importantly, you need to use a **private** github repository to hold all your codes and set the webhooks to your designed website on the openshift server. Remember to **add me (ayueei) as your collaborators.** (After the deadline**, no commits are allowed** for your private github repo!!)

Milestones

Milestone 1 - Access Control and Session Management	**25%**
Account registration, login and password change	10%
Cookies to authenticate a user session	5%
A link for login/logout and show the username after login	5%
Access control on private images	5%
Milestone 2 - Index Page & File Upload	**25%**
Sorted album layout and Pagination	10%
Resized images w/ clickable image links	5%

Photo uploading and file type validation	10%
Milestone 3 - Photo Editor	**30%**
Filters	15%
Undo & Discard & Permalink	15%
Milestone 4 - System Initialization	**10%**
Input Validation & Readme	**10%**
Input Validation	5%
Readme	5%

Note that input validation should be done in all milestones. When grading, we will try some invalid inputs to test this feature. A full mark for this part will be given if your system can handle these properly.