CSCI 4140 Spring 2024
Assignment 2 - Chrome Extension for Web Instagram

Deadline: 23:59, 2024 Mar 21 (Thur)
TA-in-charge: Jiaxi Jiang
jxjiang23@cse.cuhk.edu.hk

1 Introduction

In the first assignment, you have implemented the back-end component of a Web Instagram application, which allows multiple users to host their images online. However, the Web Instagram application is a toy application that supports limited functionalities, *e.g.*, uploading images and applying filters. You will enhance the functionality of the Web Instagram through client side techniques in the second assignment.

Specifically, you will implement a Chrome extension. With the extension, a user will be able to perform advanced editing on her/his images. For example, the user will be able to adjust the brightness, contrast, saturation, *etc.*, of an image, and to apply advanced preset filters. Further, it enables a user to save any image she/he likes to her/his album from any website she/he is browsing.

You will not implement the extension for your own Web Instagram implementation, which may differ significantly from others' implementations. Instead, you will implement the extension for an open-source image hosting application – Chevereto[1]. Specifically, your extension should:

- support the free version of Chevereto hosted on any server;
- enhance the photo editing feature of Chevereto using CamanJS[2]; and
- allow a user to upload web image(s) with right click(s).

1.1 Requirements

This assignment has the following requirements.

- You have to use the CamanJS image editing JavaScript library to implement the image editor.
- You can use any extra libraries (*e.g.*, jQuery) besides CamanJS in this assignment.
- You must implement a Google Chrome Extension.

2 Implementation, Grading, and Constraints

- You are recommended to implement a Manifest V3 extension, since Manifest V2 is deprecated in 2024.
- To submit your work, compress your codes and documents into one single file named as "SID_CSCI4140_assignment2.zip", and then submit it through blackboard.
- Do not include the *.git* repository and any other temporary files in your submission.
- Your extension should be able to run on top of Google Chrome 100.0 or above.
- There must not be any JavaScript error thrown to the Developer Console from your extension (from the service-worker script or background and popup pages, and content scripts). Otherwise, at least 5 points would be deducted.

3 Task 1: Setting up Chevereto in Your Local Machine

Chevereto is an 'Instagram-like' open-source image hosting application. In this assignment, Chevereto is required to be deployed on a local machine for testing the functionality of your Chrome extension. You can find the installation instruction of Chevereto at https://github.com/Chevereto/Chevereto-Free.
Hint: Here we recommend using docker-compose to build Chevereto. Try using the template provided at https://github.com/gilgamsh/csci4140-assignment2. Just use the command `docker-compose up -d`. Otherwise you may need to install MySQL or a compatible DBMS software on your local machine. You can also use one hosted in the cloud.

[1] https://github.com/Chevereto/Chevereto-Free/
[2] http://camanjs.com/

4 Task 2: Extension Setting

Your Chrome extension should use the *Browser Action* so that a user can upload images on any website to Chevereto. Since the extension needs to interact with an instance of Chevereto, the specific host name of the Chevereto instance must be configured on the configuration file of the extension. The default value needs to be set to http://localhost/, *i.e.*, by default, Chevereto shall locate at the index page of the localhost server. The user should be able to change this default value to any other valid URL by clicking a *Setting* button in the *Popup* of the browser action of your extension. This is to make sure the extension works correctly no matter where Chevereto is hosted. In particular, you should direct the user to a popup page to configure the Chevereto URL. Figure 1 gives an example of the popup page that is used for settings.

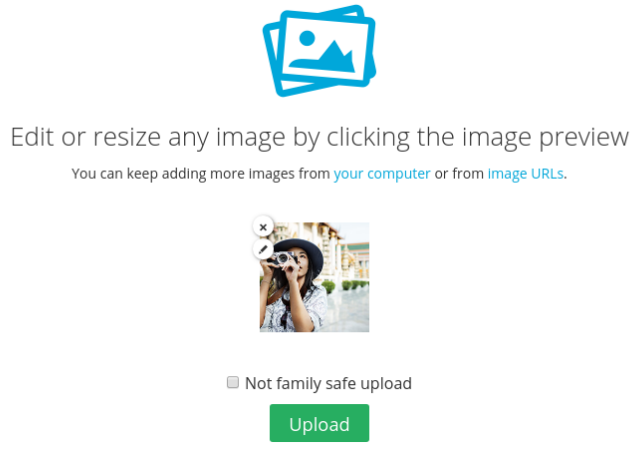
Figure 1: An Example of the Popup Page

Hint:

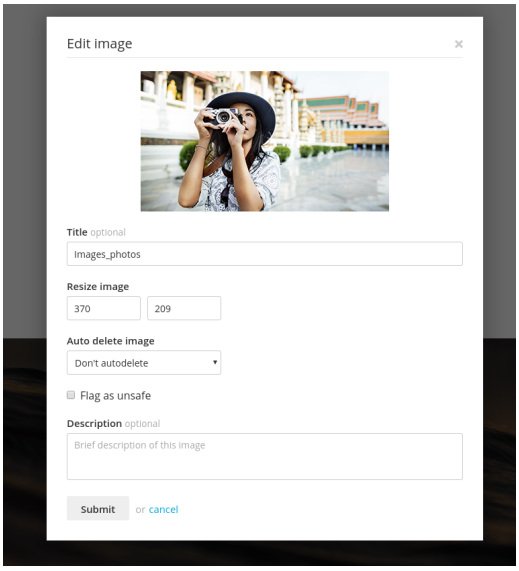
- The setting should be saved into the browser storage, *i.e.*, using the *chrome.storage* API.
- You can inject content scripts to all pages and check with the service-worker script (or background scripts) if the current URL matches the Chevereto server URL configured by the user to determine if the interface code should be executed. Alternatively, you can programmatically inject the content scripts from the service-worker script (or background scripts).

5 Task 3: Enhancing Chevereto's Photo Editing Feature

When a user clicks the *Upload* button in the main page, Chevereto allows a user to upload images from a user's computer or from the Internet by providing image URLs in its *Image Upload* interface/page. It then directs the user to an *Image Preview* page, shown in Figure 2(a), that allows the user to edit or resize the image by clicking the image preview. The original image editing page is shown in Figure 2(b).



(a) Image Preview Page



(b) Original Image Editing Page of Chevereto

Figure 2: Original Interfaces of Chevereto

In this task, you are required to enhance the photo editing feature of Chevereto by leveraging CamanJS (http://camanjs.com). Specifically, you need to insert a new *Edit with CamanJS* button next to each image displayed in the Image Preview page. When a user clicks on such a button, an image editor interface is displayed such that the user can edit the corresponding image with the CamanJS library. Note the image editor interface should be displayed in the same web page instead of a new page. Figure 3 shows an example of the enhanced image preview page.

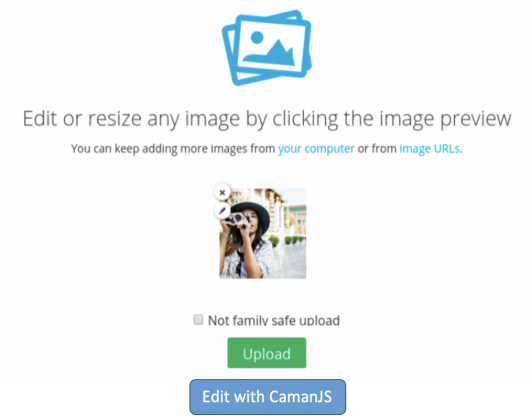


Figure 3: An Example of the Enhanced Image Preview Page

Your are required to implement the brightness, contrast and saturation filters, as demonstrated in http://camanjs.com/examples/. You do **NOT** have to implement other filters in the Interactive/Preset Examples. In particular, you should add an *Original Image* button such that the user can discard all the edits on an image.

When the user finishes editing, the user can click a *Done* button in the image editor interface to save all the work done on an image and go back to the *Image Preview* page. If the user clicks the *Edit with CamanJS* button again for the same image, the user would start editing from the original image, i.e., all settings are reset and you do **NOT** have to save the last edit. The user can also click a *Cancel* button in the image editor interface to discard all edits made in the *current* editing session, and also return to the *Image Preview* page.

Finally, the user can click the *Upload* button in the Image Preview page of Chevereto to finish image uploading. The edited images should be uploaded to Chevereto. Figure 4 gives an example of the image editor interface (you only implement the aforementioned 3 filters).

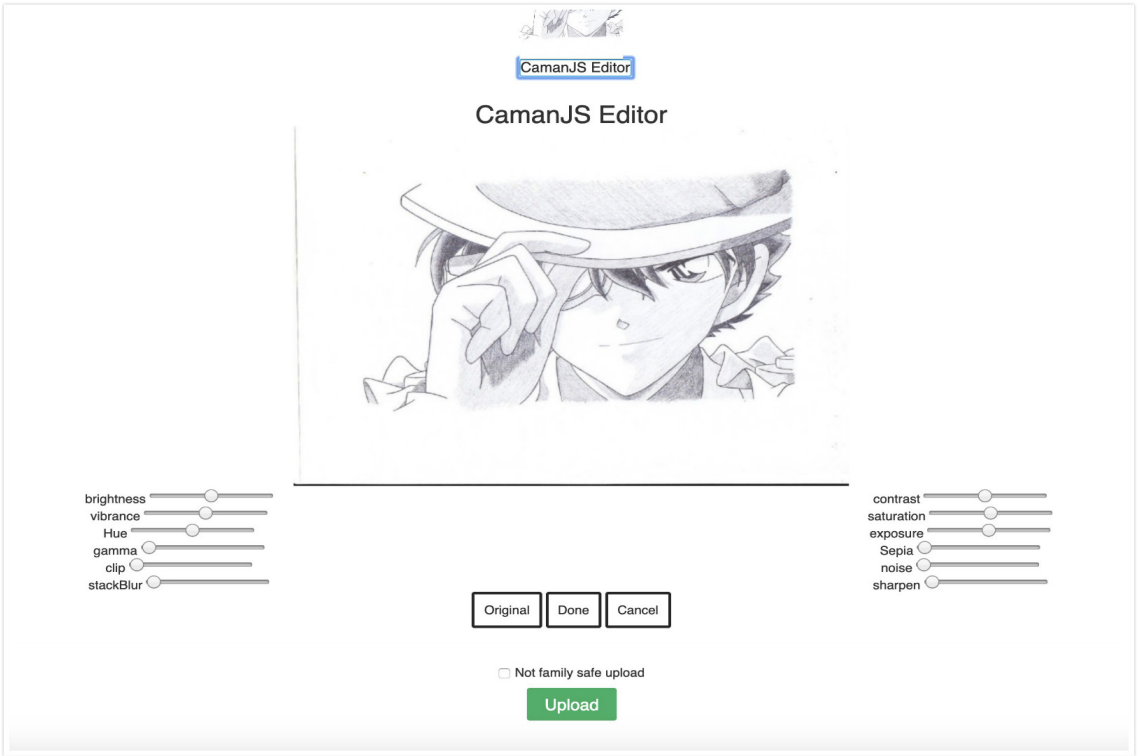


Figure 4: An Example of the Image Editor Interface

Hint. You can use HTML5 Canvas for editing the images. Canvas allows you to draw images using JavaScript. For example, you can use `<canvas id="myCanvas" width="200" height="100"></canvas>` to define a canvas with specified `id`, `width` and `height`. Further, you can import an image to the canvas like this:

```
<script type="text/javascript">
  var c=document.getElementById("myCanvas"); // find the canvas element by ID
  var cxt=c.getContext("2d"); // this gives a built-in HTML5 object
  var img=new Image();
  img.src="flower.png"
  cxt.drawImage(img,0,0); // draw the image on the canvas
</script>
```

You can then use CamanJS library to manipulate the images. For instances,

```
Caman("#your-canvas-id", function () {
  this.brightness(40); // adjust brightness
  this.contrast(-10); // edit contrast
  this.sinCity();
  this.render();
});
```

More examples are available here: http://camanjs.com/guides/. For hints on uploading edited images, please refer to the next section.

6 Task 4: Uploading Web Images with Right Click

Your extension should insert a new option – *Upload to Chevereto* – into the context menu when a user right clicks an image in a web page. When the user clicks this new option, a new tab will be opened, and the user will be redirected to the host page of Chevereto. Then, the content script should automatically select the file to upload, and show the *Image Preview* page in that new tab. The user can then edit those images with the features implemented in Task 3 (§5). **Following are some hints.**

- You need to inject a new `<script>` element into the Chevereto page to interact with the Chevereto's scripts from your content scripts.
- You shall convert the edited image in your editor canvas to a `File` object. Then replace `CHV.fn.uploader.files[0]` object with the edited File object and the original metadata stored in the object.
- If you want to directly load an image instead of using its source URL, you can load the remote cross-origin image in your extension's service-work script (or background/popup scripts) and convert the image into dataURL. Then load the dataURL image in that canvas of the web page.
- If the source URL is a base64 encoded dataURL, you do not need to support it with the context menu upload feature.
- If you cannot access certain DOM element, *e.g.*, the `queue-item`, you may try to wait for some time by calling `setTimeout` before you access it.

7 Milestones

Task 1	**0%**
Task 2	**15%**
Task 3	**58%**
Injecting *Edit with CamanJS* button and showing the *Photo Editor* for the selected image	10%
Photo Editor - 3 Filters	18%
Photo Editor - Done and Cancel	15%
Uploading the edited images to Chevereto server	15%
Task 4	**27%**
Adding option in context menu	12%
Uploading to Chevereto	15%