Catch Similar

HCI-LAB2: IMAGE RETRIEVAL

Catch Similar Design Document

Author: 1852448 Li Yuanfeng

Email: 952445590@qq.com

CATALOG

1. REQUIREMENTS 2

2. DESIGN OVERVIEW 3

2.1. FORMULATION 3

2.2. INITIATION OF ACTION 5

2.3. REVIEW OF RESULTS 7

2.4. REFINEMENT 8

2.5. USE 9

2.6. ADDITIONAL FUNCTION: SEARCH HISTORY 12

3. INTERFACE SPECFICATION 14

1. Requirements

No.	Requirement	Description
Requirements – Search Image		
1	Upload image	Users can upload an image to the website. The site needs to remind the user of the supported formats. The user can upload another image again, which will automatically overwrite the original image. The user can also cancel the upload.
2	Preview the image	Users can see a preview image of the uploaded image so they know if the upload was successful.
3	Search image	Users can search for similar images through the website. The website will display the search results to the user.
4	Download results	Users can download selected images to local.
5	Show image tag	The website will tag the search results so that users can visually know what type of images the results are.
6	Filter	The user can select certain tags to filter the images.
Requirements – Favorites		
7	Add to favorite	Users can add image search results to their favorites. Users will be able to see the favorite images every time they open the website.
8	Delete from favorite	Users can delete favorite images.
9	Download favorite image	Users can download selected favorite images to local.
Requirements – Search History		
10	Save history	Whenever a user searches for an image, the site keeps a search history. And the history will be shown to users.
11	Clear history	Users can clear all history.

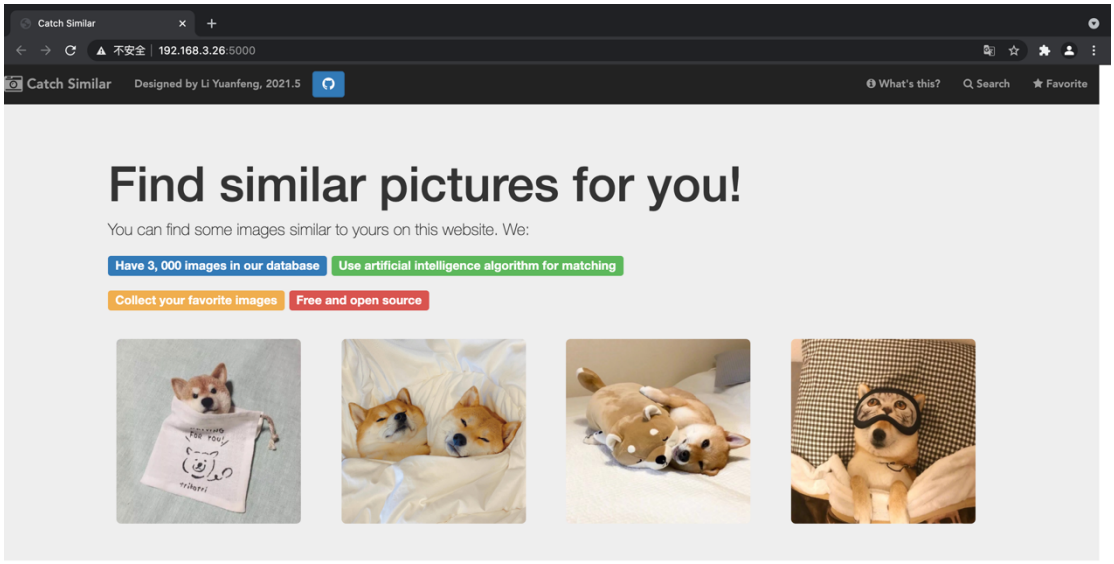
12	Download images in history	Users can download images in history to local.

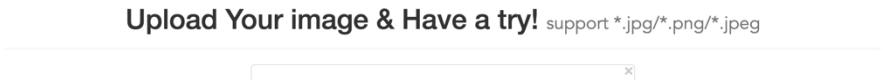
2. Design Overview

2.1. Formulation

➢ **Introduction Panel**

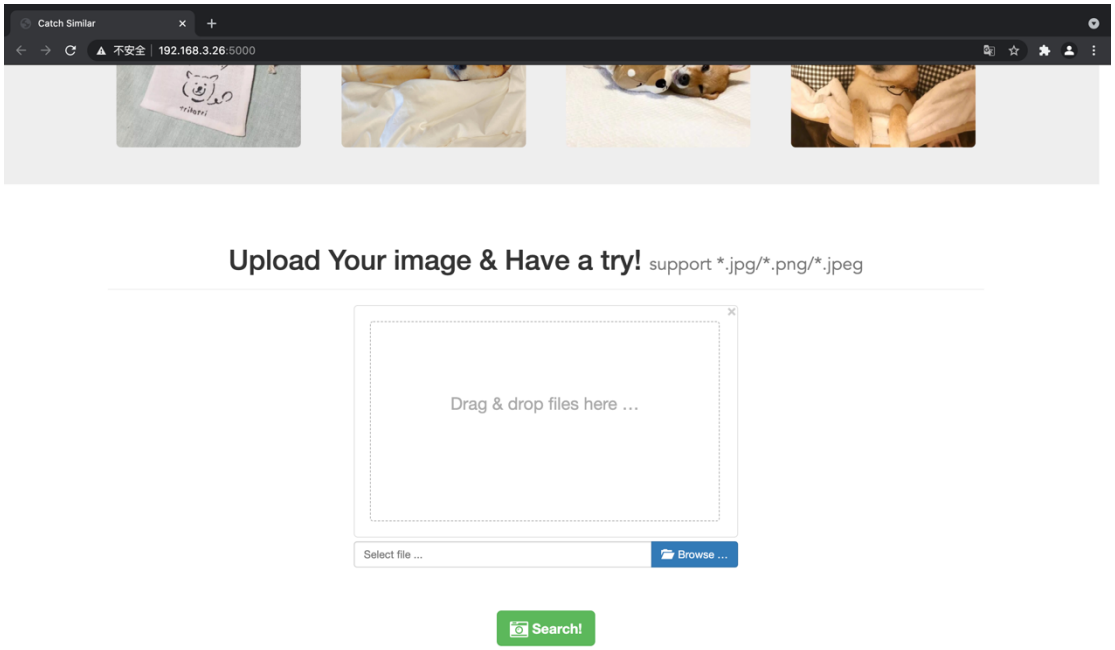
First, after entering the website, users will see the introduction panel. This introduction panel states the features of the website in detail. This panel is implemented in plain html.



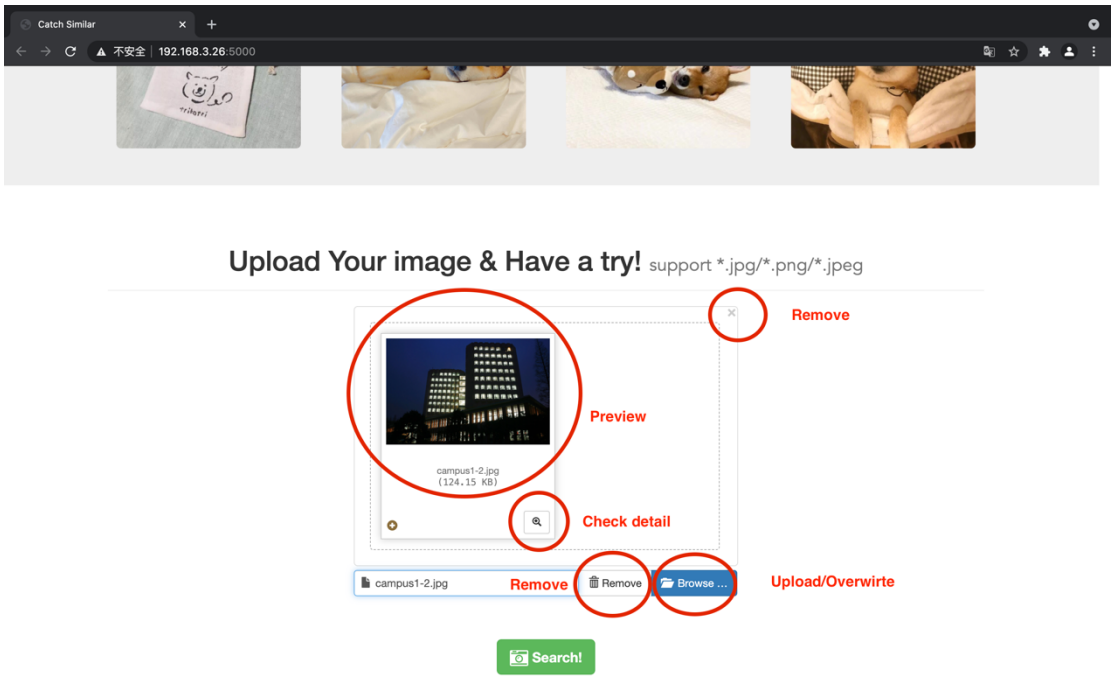


➢ **Component - upload image**

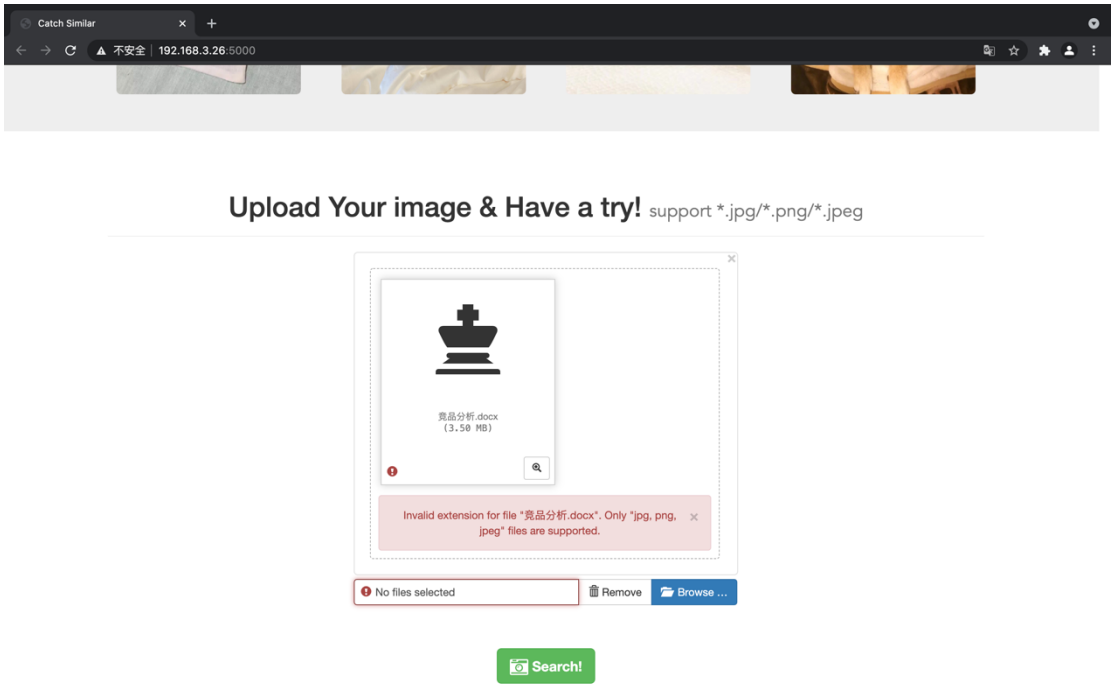
By component “bootstrap-fileInput”, Catch similar provides a complete image upload function. Its functions include:



- Upload, cancel upload, overwrite upload and preview the uploaded.



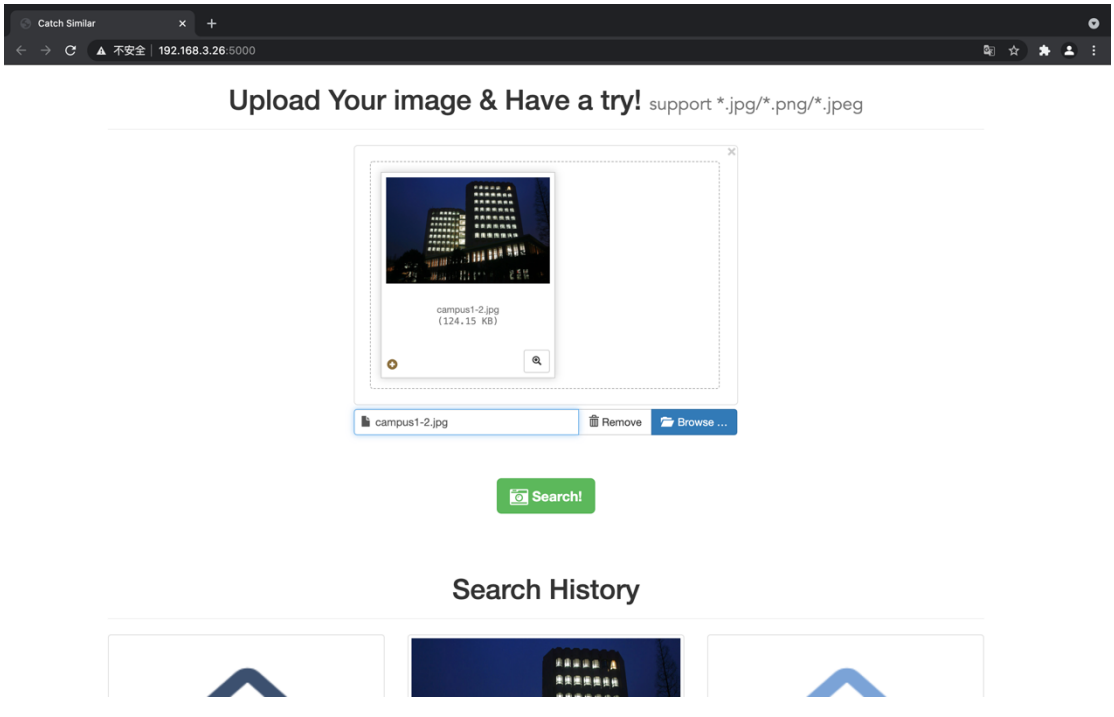
- Verify file formats



2.2. Initiation of action

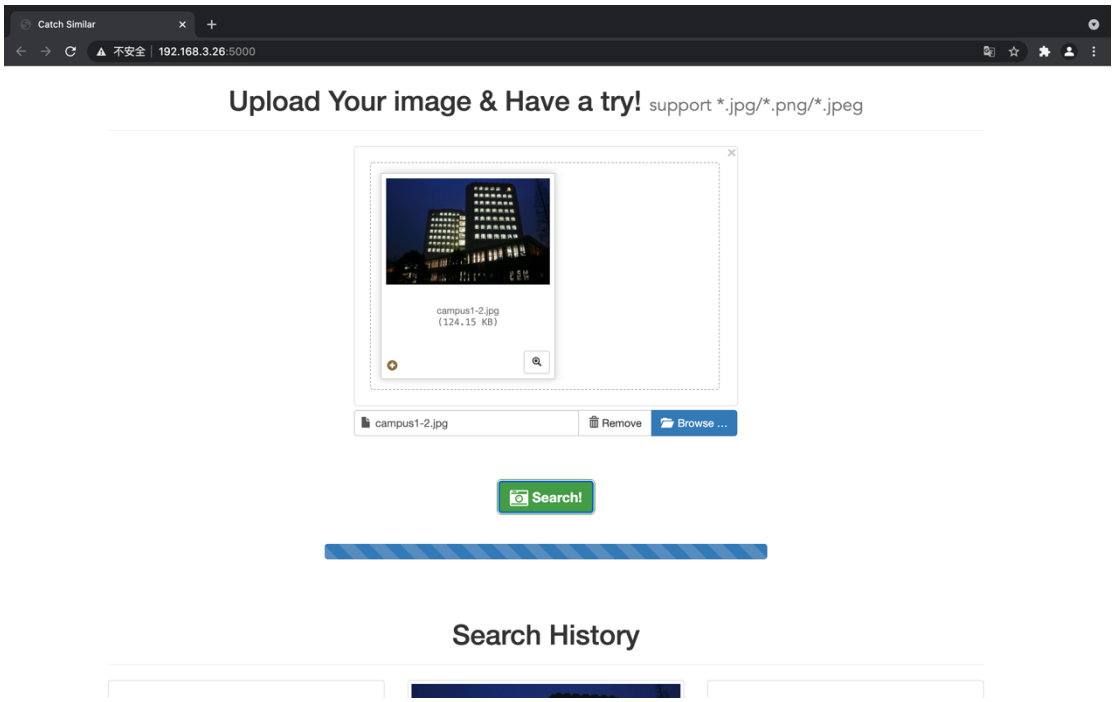
➢ **Search button**

The site has a built-in search button that is bound to a function linked to the backend. When the button is pressed, the front-end will send an ajax(post) request to the back-end and send a file to the back-end. The backend will receive the file and will call search.py to get the image. When the fetching is done, the front-end will send an ajax(get) request to get the image returned by the backend.



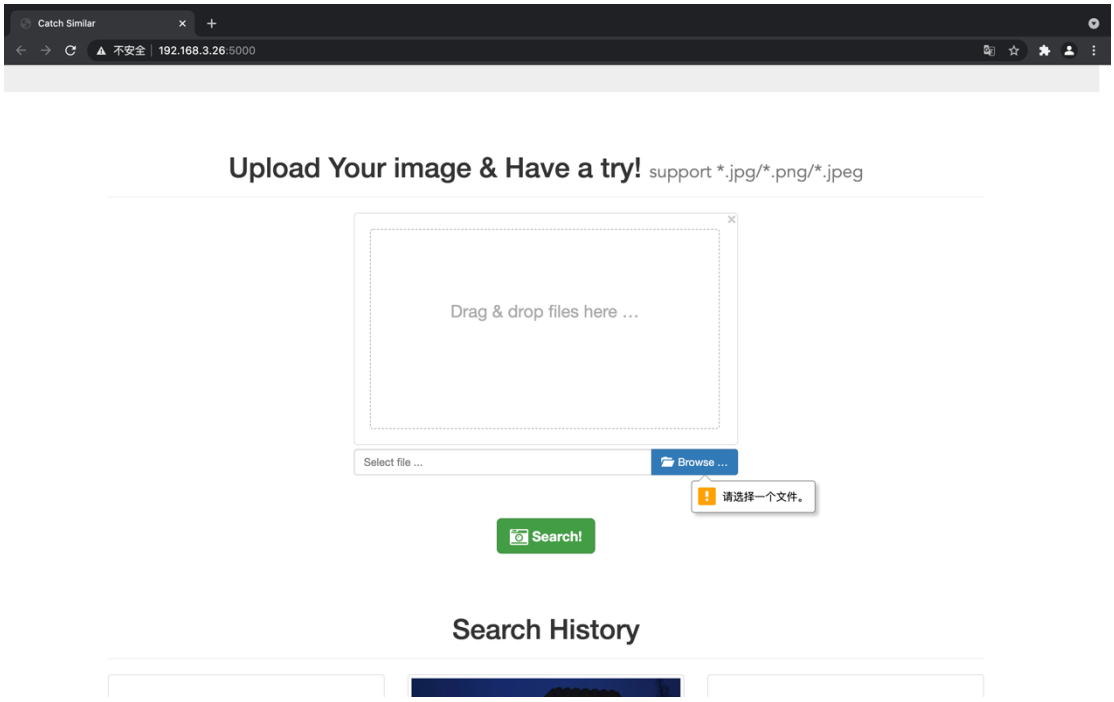
➢ **Progress**

The site is designed with a progress bar control via js. The progress bar will be displayed to alert the user if a search is started.



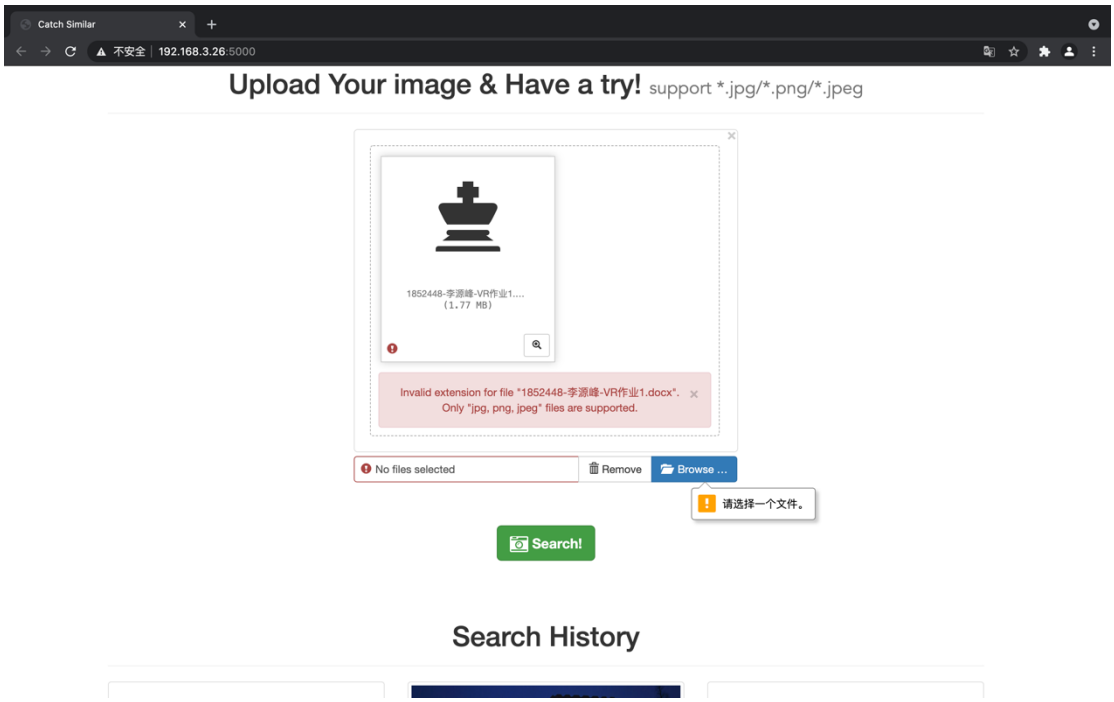
➢ **Exception handling - pressing the button without selecting the image**

If a user presses the button without uploading an image, the site will give an alert and the search will not start.



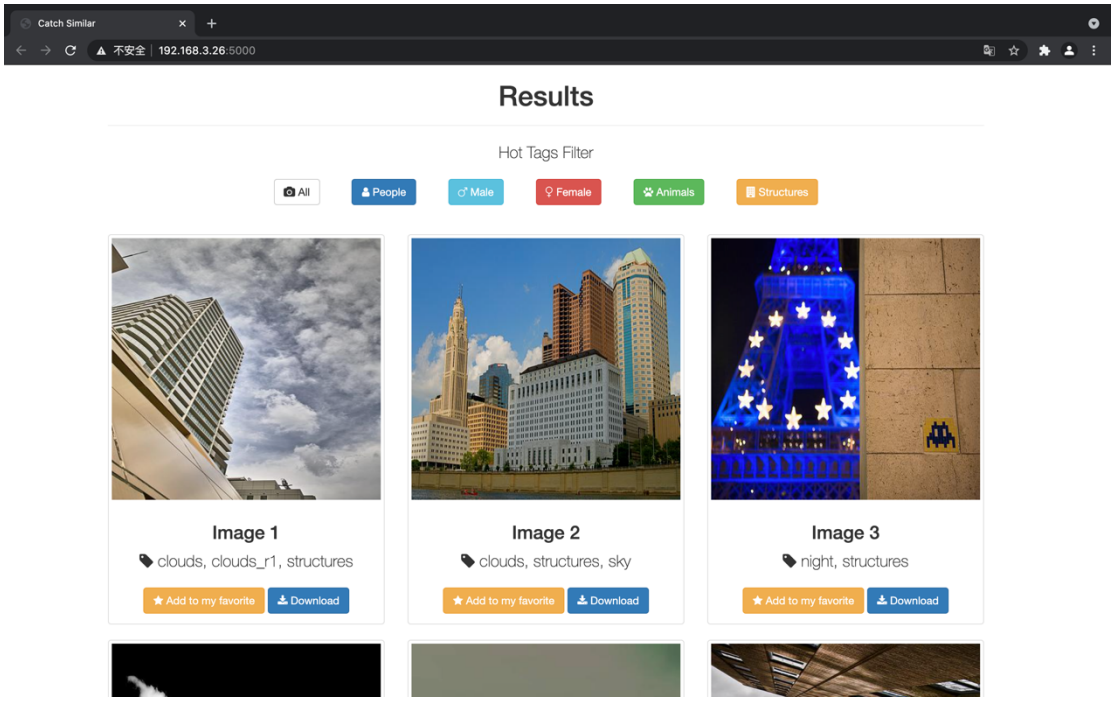
➢ **Exception handling - pressing the button with wrong file**

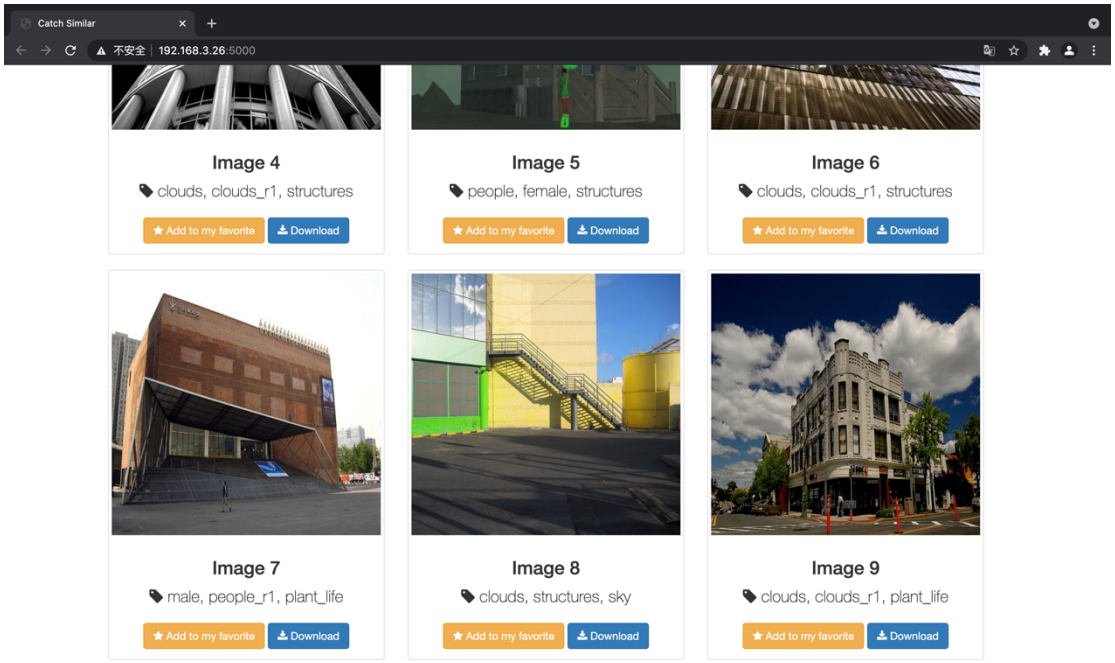
If the user uploaded unsupported file and press the button, the site will give an alert and the search will not start.



2.3. Review of Results

The images returned through the backend are displayed on the website as cards by JS. Each card contains pictures, labels and corresponding actions.

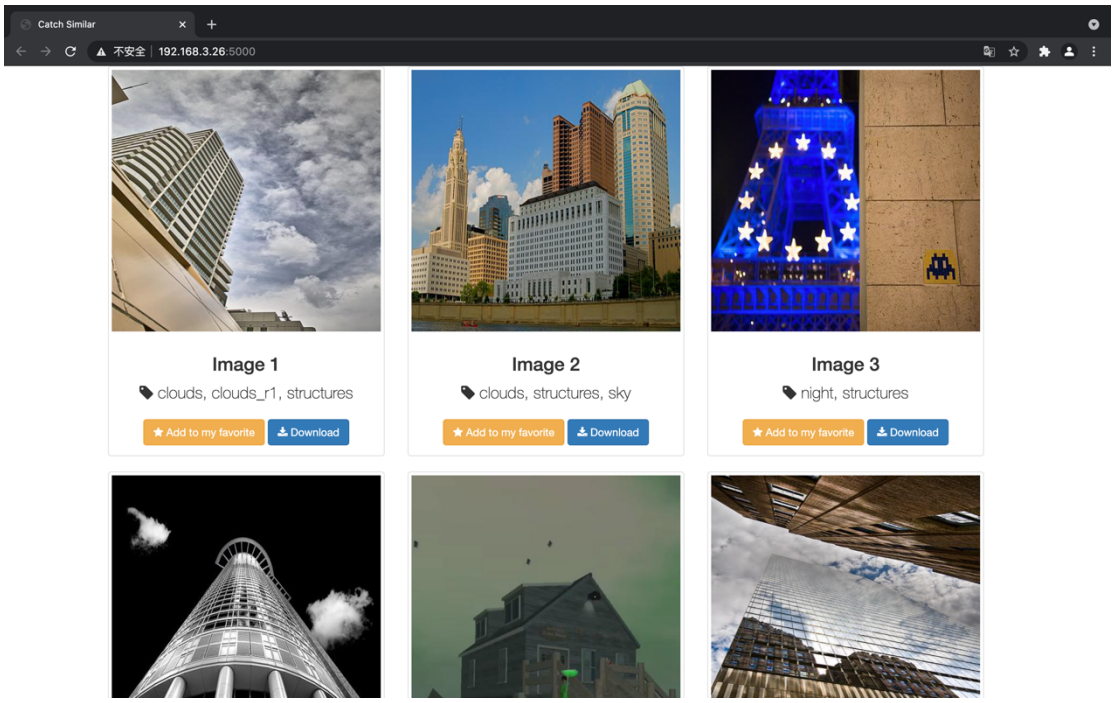




2.4. Refinement

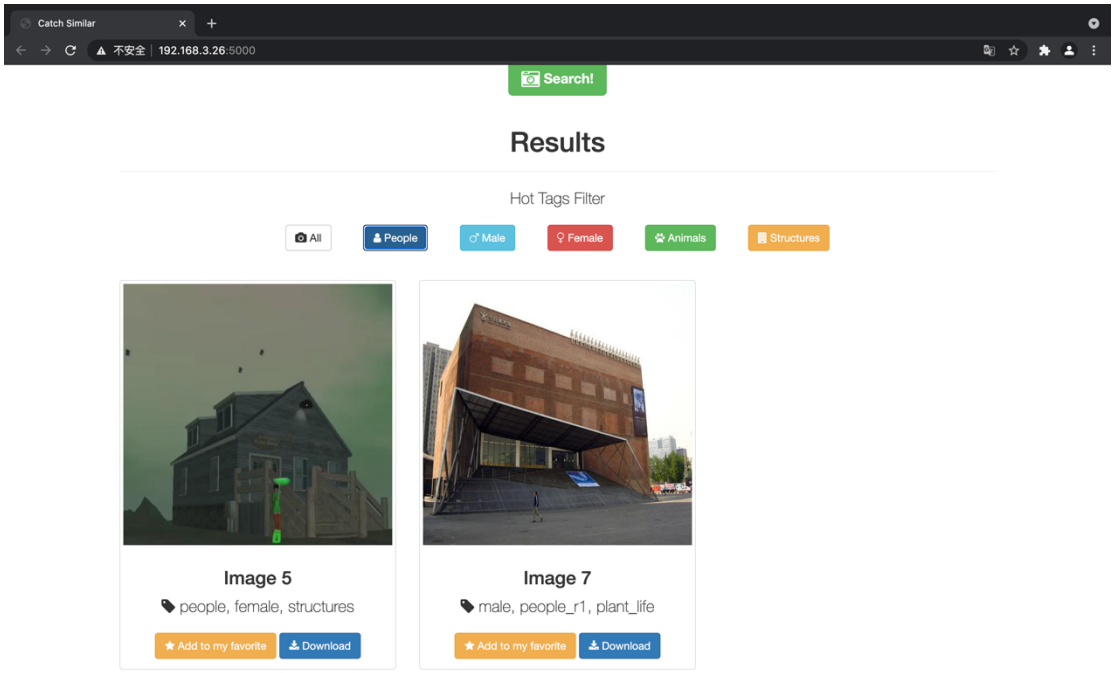
➢ Tag the result

After getting similar images, I also added the operation of tagging the images. I have designed a function tags_search() in search.py. If no tags file exists for the current image, it will search the tags information in the database according to the image name in turn and store the result in a txt file. To shorten the search time, I design each image to have at most 3 tags.



➢ Filter the image by tag

The site supports using popular tags to filter the search results. This function is implemented through the JS function my_filter(). When the filter button is clicked, my_filter() will match the tag corresponding to the button with the image tag as a string, and then hide the images that do not match the requirements.

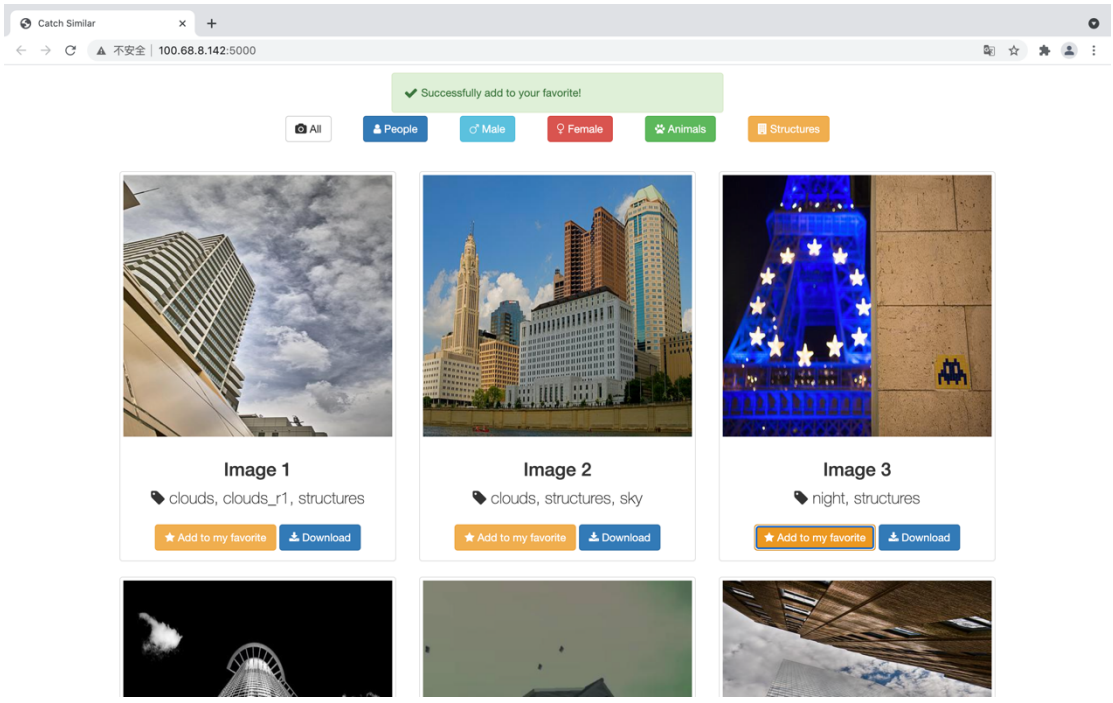


2.5. Use

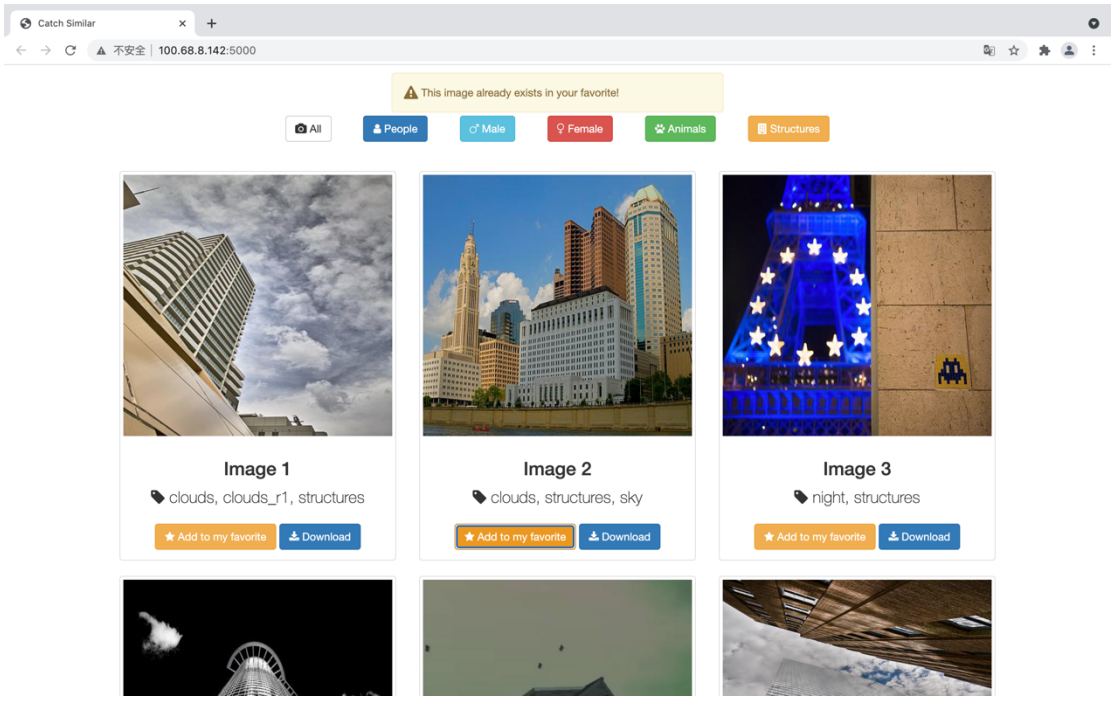
➢ **Add to favorite**

Users can add the results to their own favorites through the Add to “my favorite” button. When the button is pressed, the front-end sends a post request to the back-end. After sending the request, the backend will store the image in the "static/favorite" folder according to the path of the image from the frontend.

If the addition is successful, the web page will send a prompt to user.

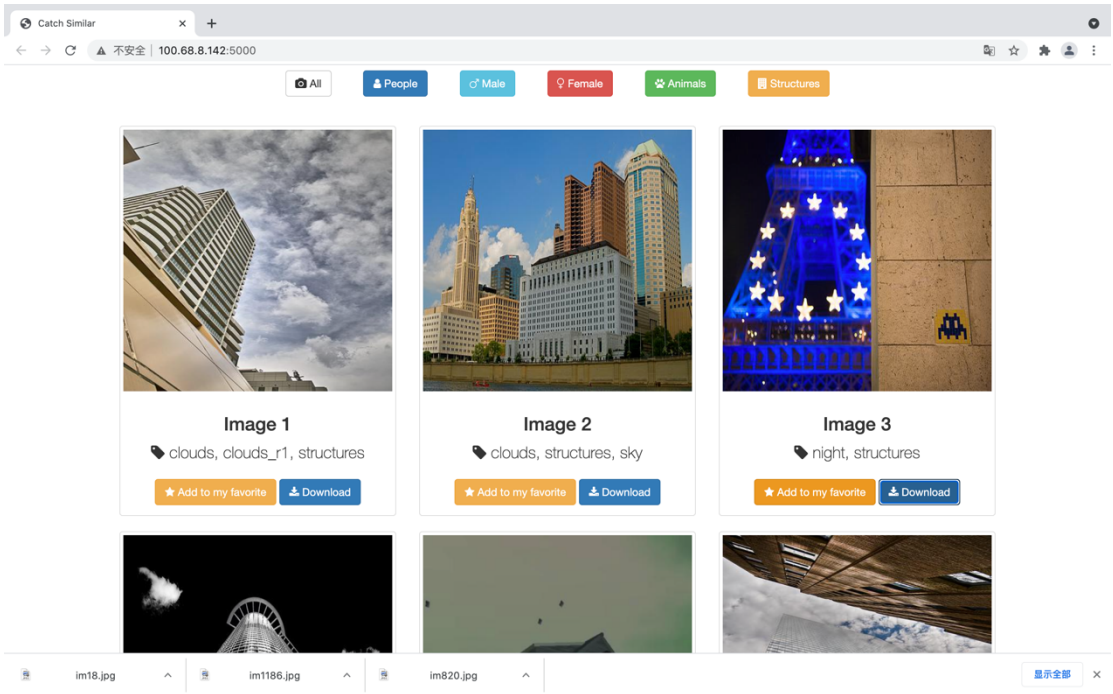


If the image already exists in “static/favorite” folder, the backend will return an error code, and the frontend will send a corresponding prompt.



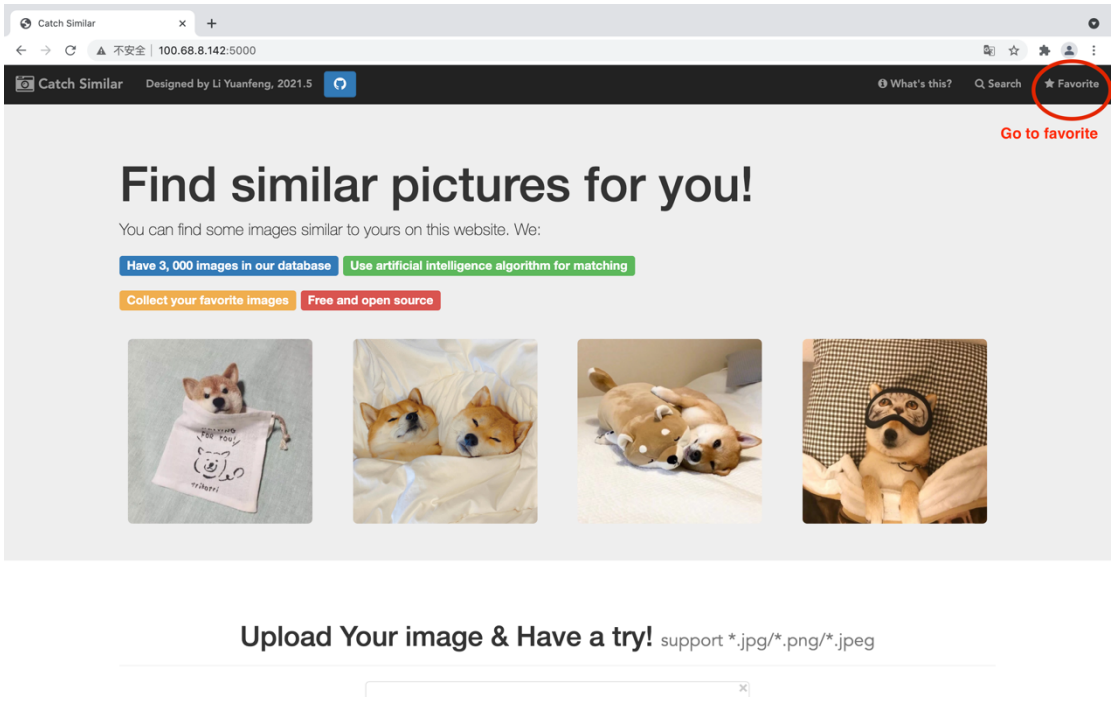
- **Download**

  Users can click “Download” button to download the results. This is implemented by the html element “a”. When the user clicks the button, JS creates a <a> tag, sets the corresponding attributes, and simulates a click event to complete the download.

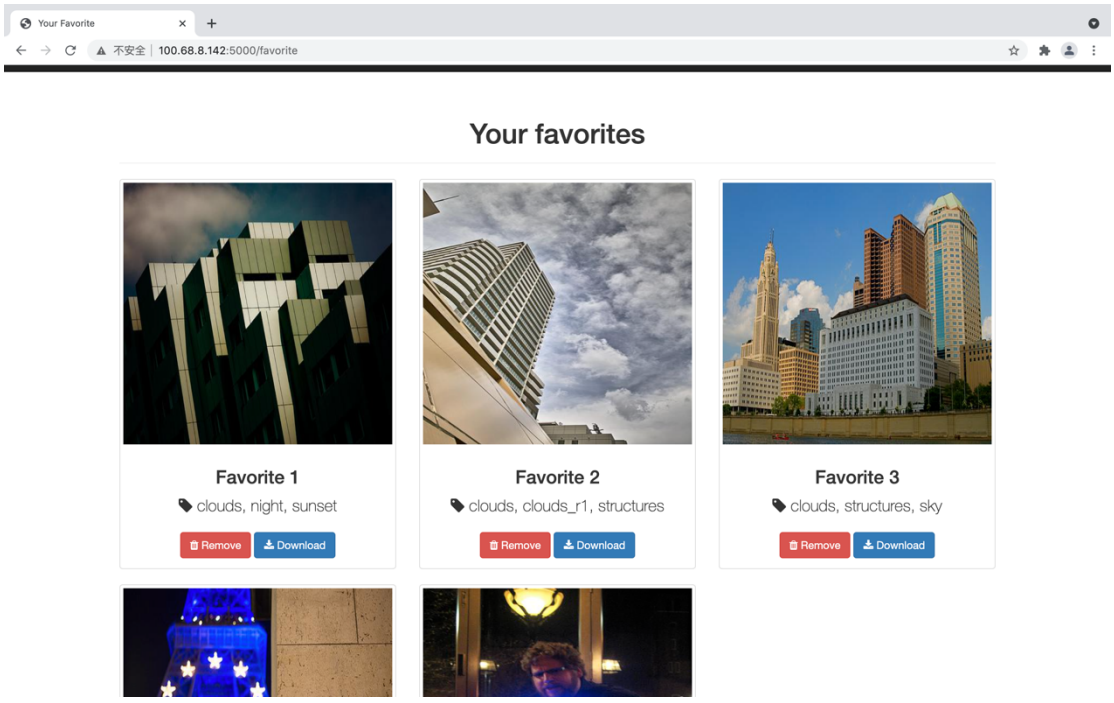


➢ **Manage favorites**

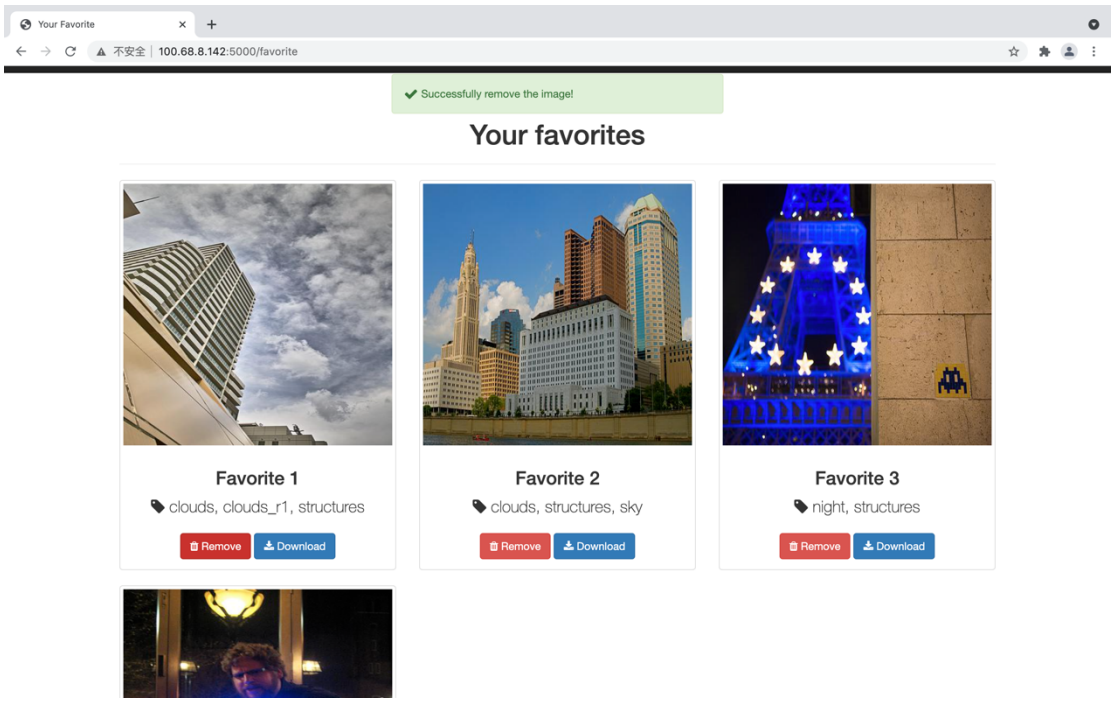
Users can go to their favorites by clicking on the "favorite" tab in the navigation.



After entering, all the favorites will show in the page.



Users can remove a favorite. The example removes “Favorite 1”.



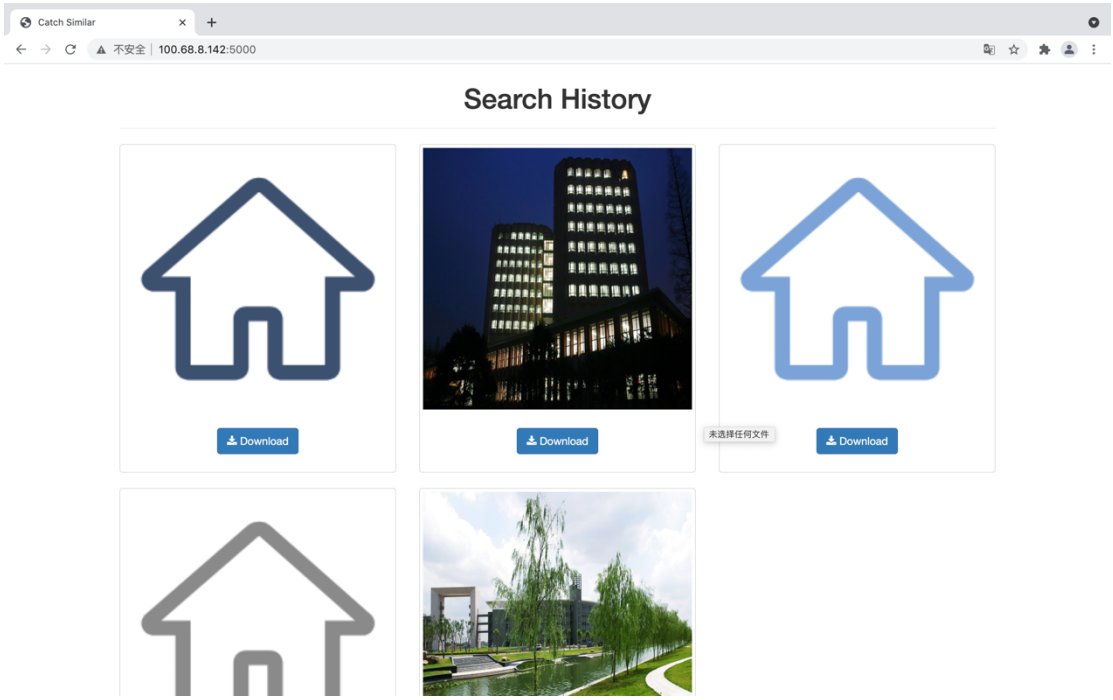
Users can download a favorite. This function is the same as before.

2.6. Additional function: Search history

- **Save history**

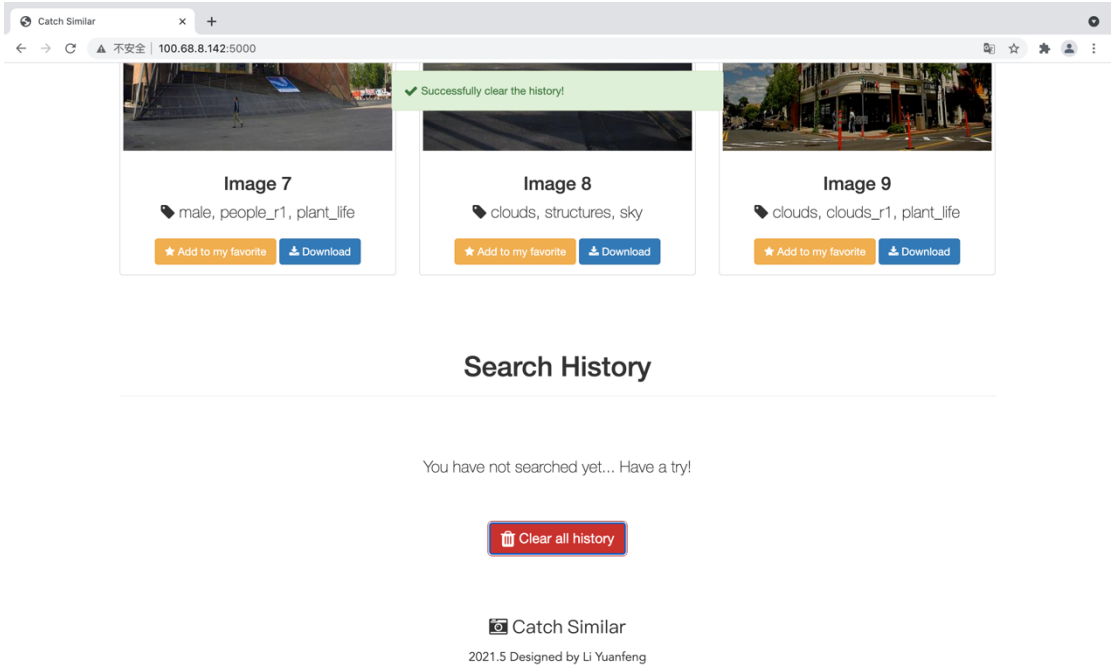
  Whenever a user uploads an image and performs a search, the site automatically sends a request to the backend and stores the image in the "static/history" folder. The site will

display the search history to the user for their convenience. The site will not store images with the same name.



➢ **Clear all history**

Users can click the "Clear all history" button to delete all history records.



3. Interface Specification

URL	Description	Frontend request method
/imgUpload	upload an image	POST
/Add	add to favorite	POST
/Remove	remove from favorite	POST
/RequireFavorite	require all the images in users' favorites	GET
/Require history	require all the images in history	GET
/Clear	clear the history	POST
/favorite	route to favorite page	GET
/	route to main page	GET