Lab 2 Information Retrieval

## 1. The requirements of an image search task

### 1) Formulation

The searching interface has to contain an input box to upload an image, and users can preview the query image in the searching window.

### 2) Initiation

The searching interface should have a search button.

### 3) Review

We should provide an overview of the results.

### 4) Refinement

The searching interface should allow changing search parameters when reviewing results.

### 5) Use

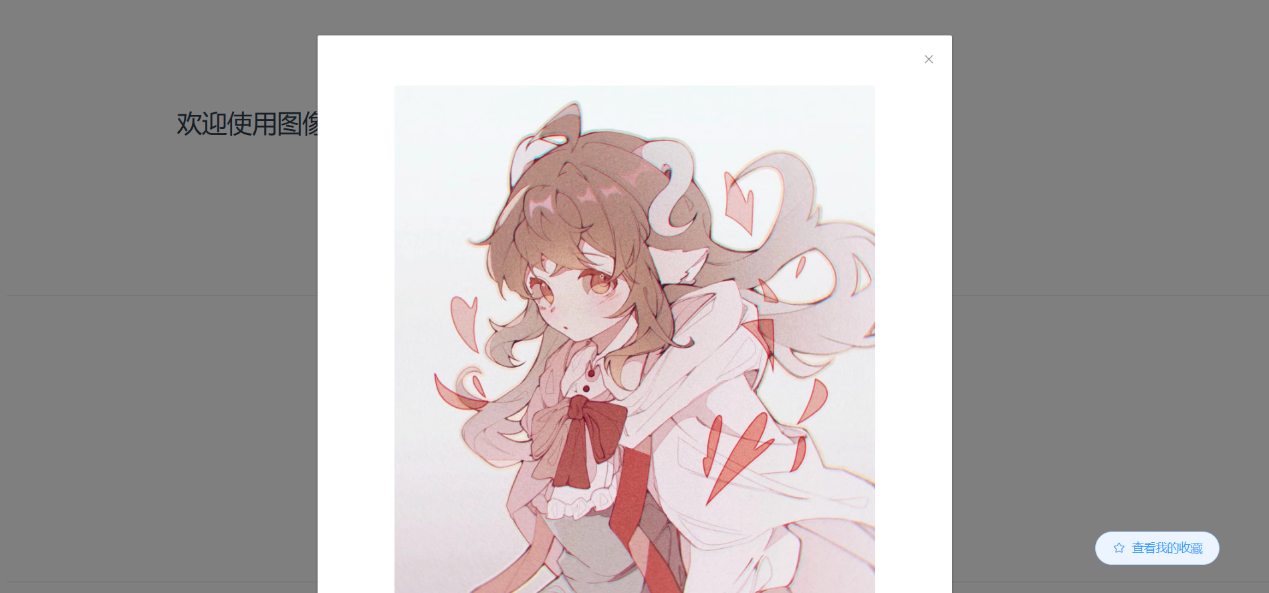
Users can take some actions, e.g. add selected images to a favorite list.

## 2. The demonstration on my project

### 1) Formulation

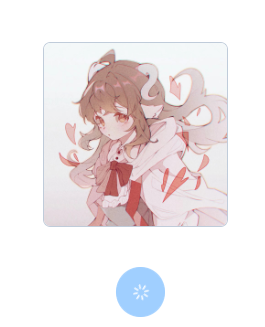
Users can upload an image and preview it.





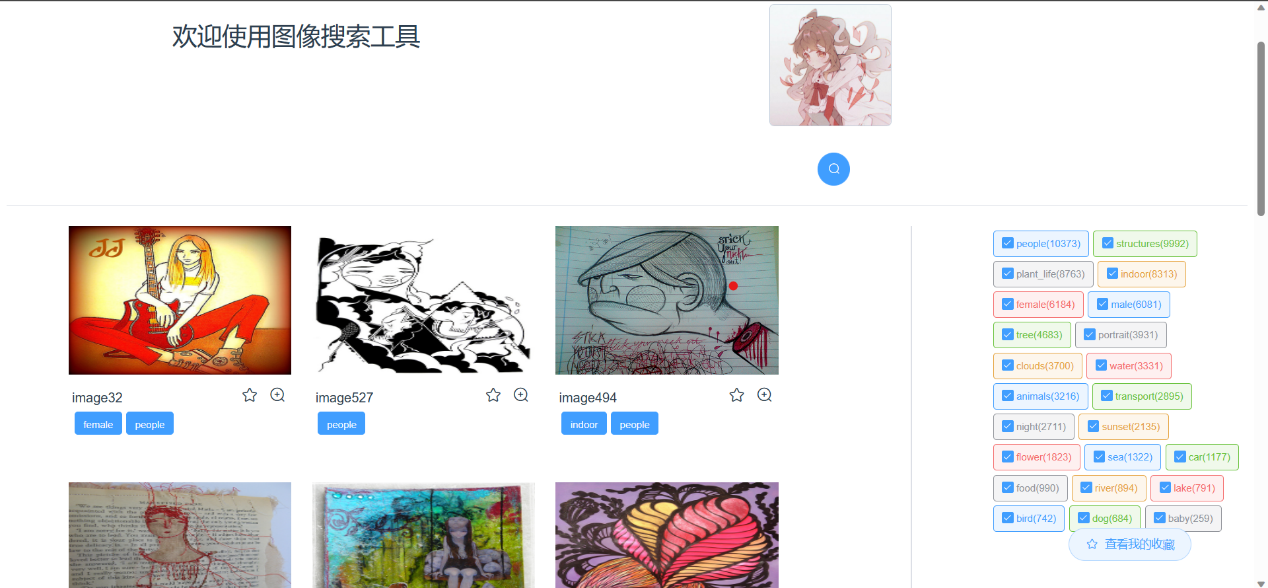
### 2) Initiation

There’s a search button below the uploaded image, and it takes a while to get result after clicking the search button.



### 3) Review

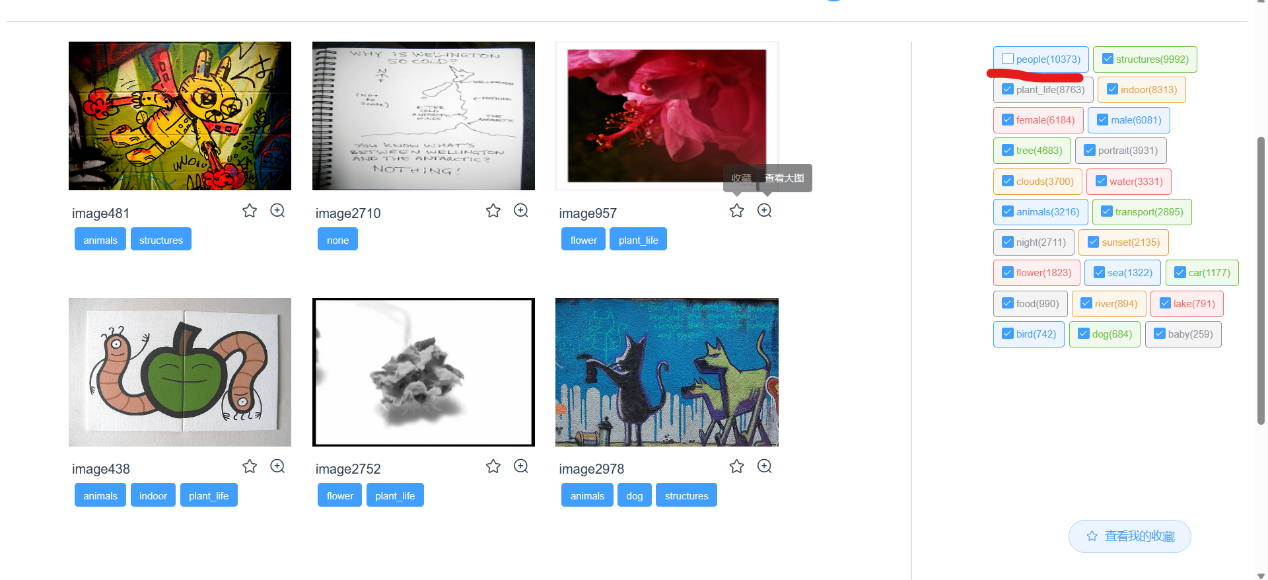
Users can have a rough review on the results.





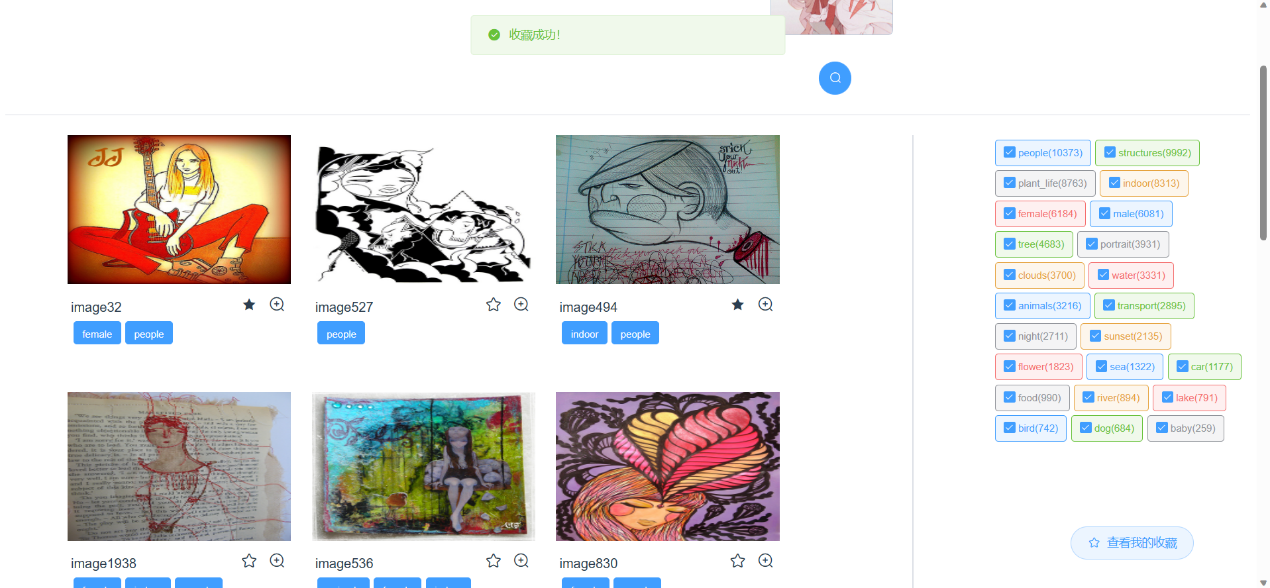
### 4) Refinement

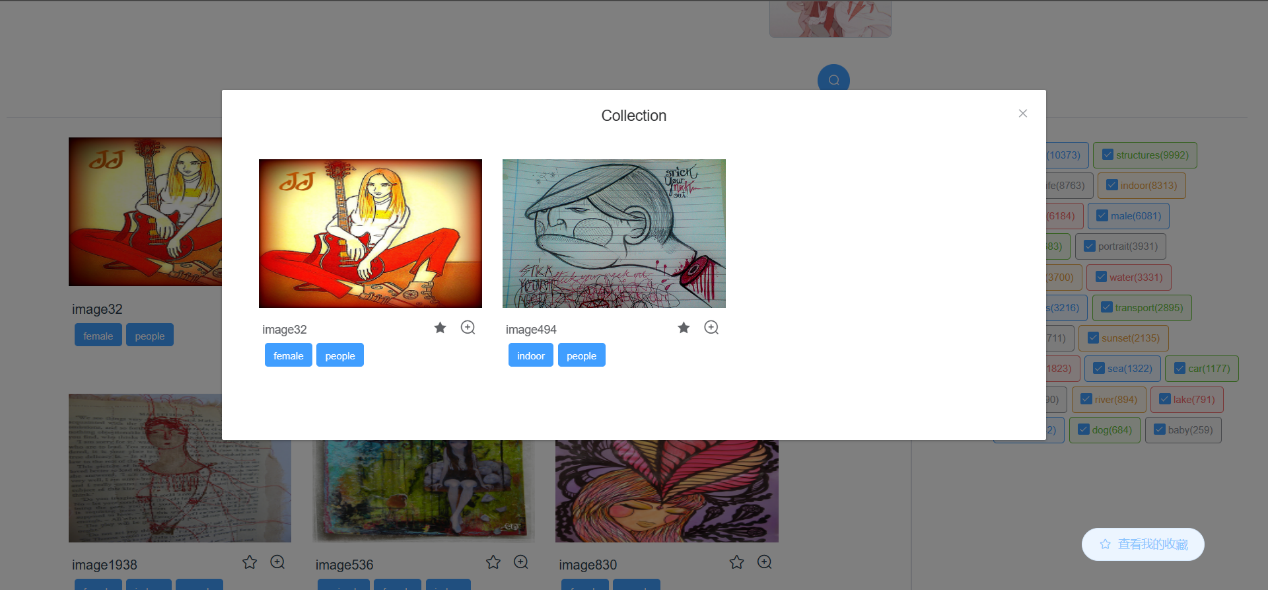
By selecting or deselecting the tags on the right, users can filter the results and get what they want.



### 5) Use

User can collect the results they like and those images can be viewed in collection.





## 3. Brief description on implementation

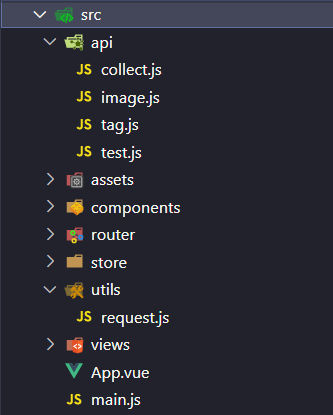
In this project, I used Vue as frontend framework, and the Flask framework in Python is used to collaborate with frontend.

For the backend, Flask has to recognize a folder in which frontend resources are kept, and in this project it is assigned to “./frontend/dist”, which contains the results of running the command “npm run build” in frontend.



There are two ways of running this project: one is to directly visit <http://127.0.0.1:5000/>, this is where backend is deployed, and since we have created a connection through folder “./frontend/dist”, it is ok to visit backend website directly; another is to start the backend server, then move into frontend folder and run the command “npm run dev”, which directs to <http://localhost:8080/>, this is where frontend is deployed.

In the frontend, I encapsulated axios into a request service, and it provides a unified approach to access apis. The request service is in util folder, and unified apis are in api folder.





Those apis are connected respectively to backend, where Flask provides a standard to create apis.



Back to frontend, the main page displayed to users is Home.vue, and it contains two components which are created by me: ImageCard and UploadImage. ImageCard defines how each image is displayed, it includes an image, the name of the image, collect button and zoom-in button. UploadImage provides input box for users as well as the function of previewing the searching image.

