Homework 2--Image Search System

- Student ID: 1851202
- Name: Lei Hong
- Date: 4th Jun, 2021
- Class Time: 10:00 - 11:35, Friday

Project Description

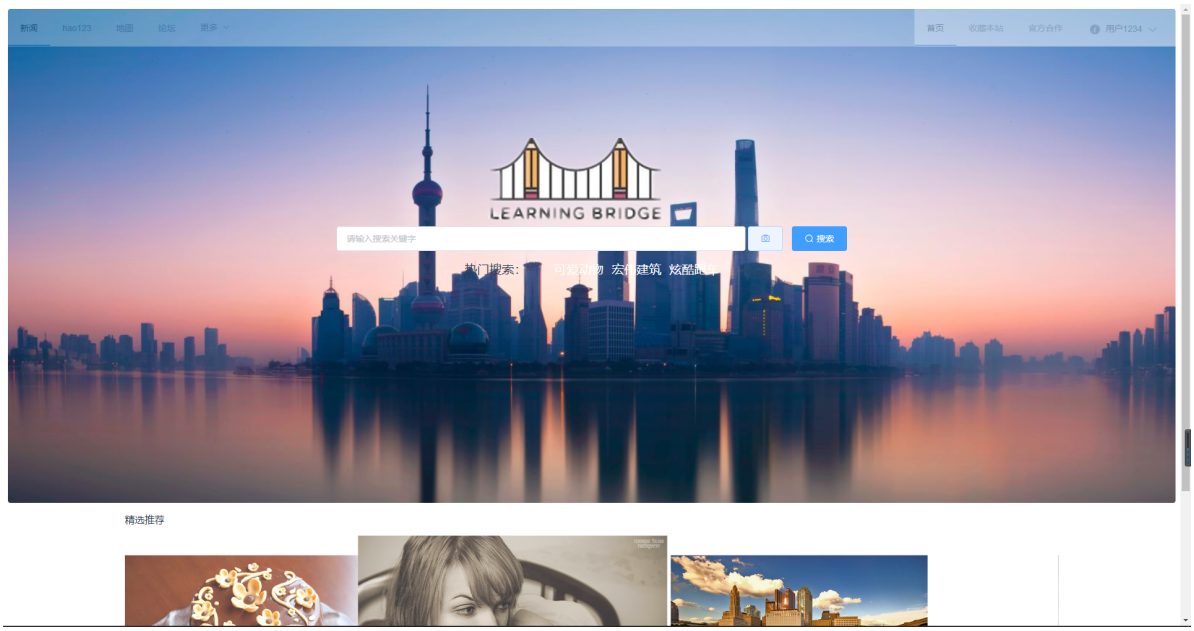
- [x] Implemented a user-friendly **image recommendation search system**

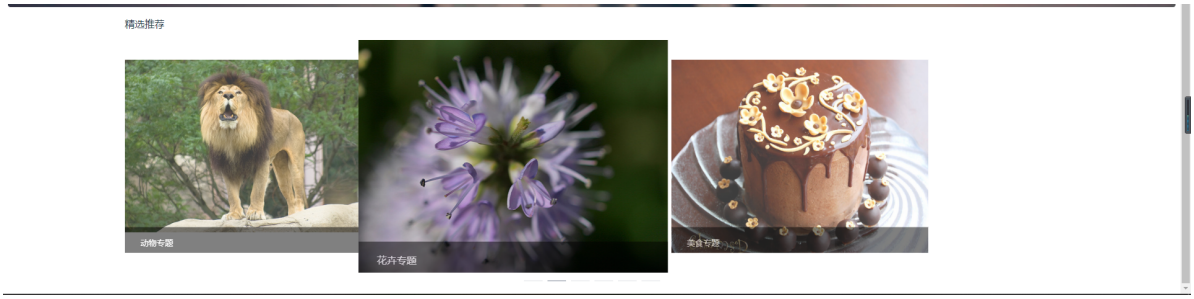
Users can use the system to:

- upload a picture, the system will identify similar pictures from the database and return them to the user as search results according to the user's parameter settings
- browse the related links of the other sites
- enter a keyword to search for related images with that keyword, the system will search corresponding pictures by keyword from the database and return them to the user as search results according to the user's parameter settings
- get some popular search suggestions, which appear at the bottom of the search input box. And this suggestions will be presented to users in the form of links to buzzwords or a quick light.
- download beautiful pictures they prefer by clicking the button "下载"

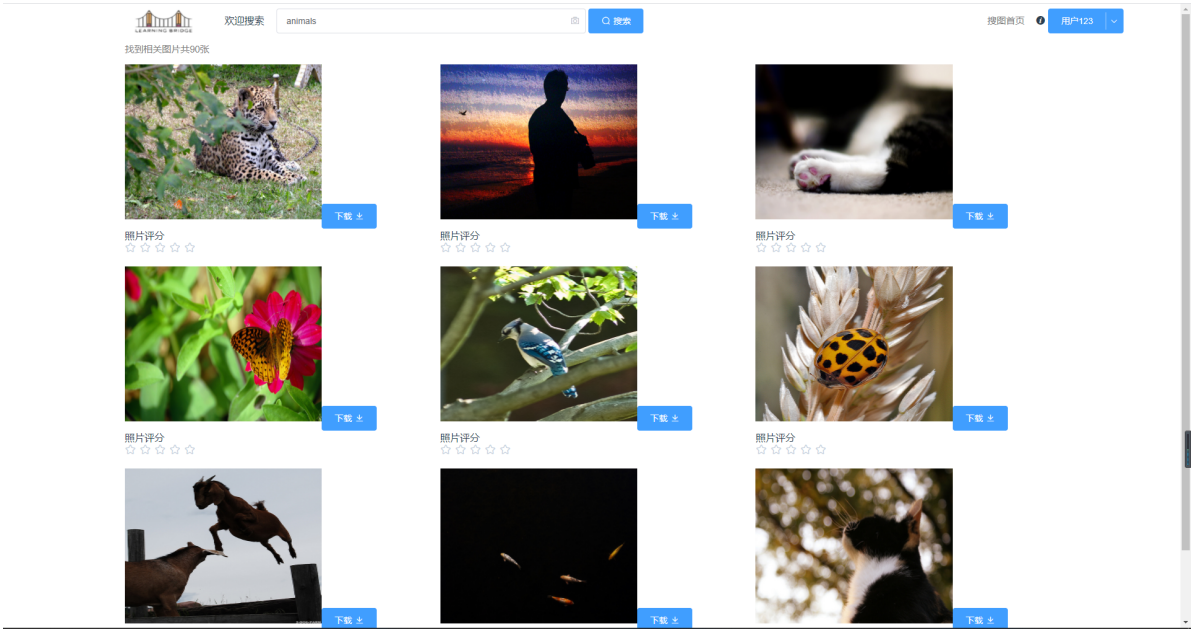
User Interface:

Homepage:





Search Result:



Development Environment:

This system adopts the architecture of front and back end separation to develop.(How to open the system can read the corresponding 'README.txt');

`Front End: Vue 3`

`Back End: Flask`

System Analysis —— requirements of an image search task

1. User Interaction

A usable, user-friendly image search system should include the following features:

- search by keyword
  - Users can enter keywords and get pictures of the content associated with them.
- upload images
  - Users can browse local files and select an image to upload
  - Users can preview the images to be searched
  - Users can reset images to be uploaded
- get result
  - The system will transmit the uploaded pictures to the back end, and the back end will return pictures with similar characteristics by calling similar image recognition algorithm model
- show the results
  - Users can view the search results

2. Image Recommendation

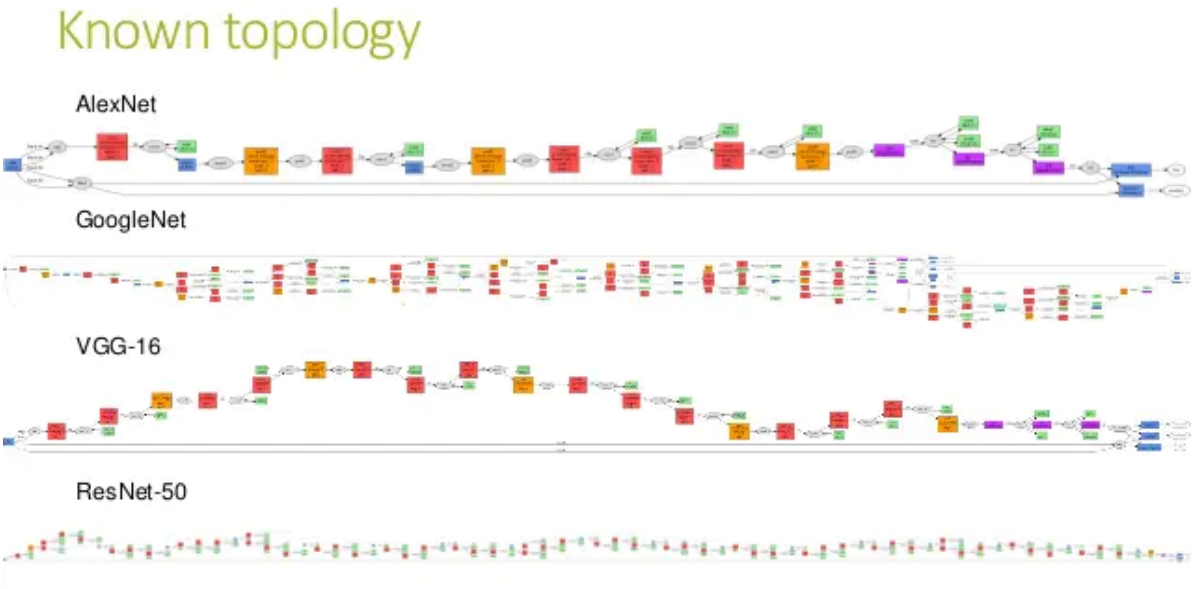
To identify whether two images are similar, we might first distinguish between a person, a landscape, etc..... The corresponding scenery is the blue sky or the sea..... Do a series of categories.

From the perspective of machine learning, the first step is to extract the features of images, classify these features, train and build models, and then identify them.

But let the computer to distinguish between these images respectively is what kind of is very not easy, but the computer can know the image pixel values, therefore, in the process of image recognition and identification by color features are similar to the picture is our common (and, of course, its features and texture features, shape features and spatial relationships, etc., these are divided into histogram, color, color board, Aggregate vectors, correlative graphs, etc., to calculate color features).

In order to get two similar pictures, the similarity of the pictures is calculated by the following simple calculation methods:

- Histograms calculate the similarity of images
- Hamming distance is calculated by the hash value
- Calculated by the cosine distance of the picture
- Calculated by picture structure metrics



System Design —— five-stage search framework

Design:

Main functions:

Ⅰ. Formulation

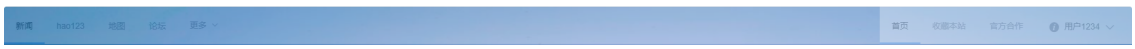
1. System's name and the main function. It contains an input box to upload an image.

2. Recognize phrases to allow entry of names, such as “animals”

3. Provide suggestions, hints, and common sources

(This is a Carousel that can change themes on a regular basis)

4. Other links, toolbars, quick jumps, and so on



5. Users can click the camera button to visit the intelligent image recognition interface, and select pictures to upload.

6. Users can preview the query image in the searching window.

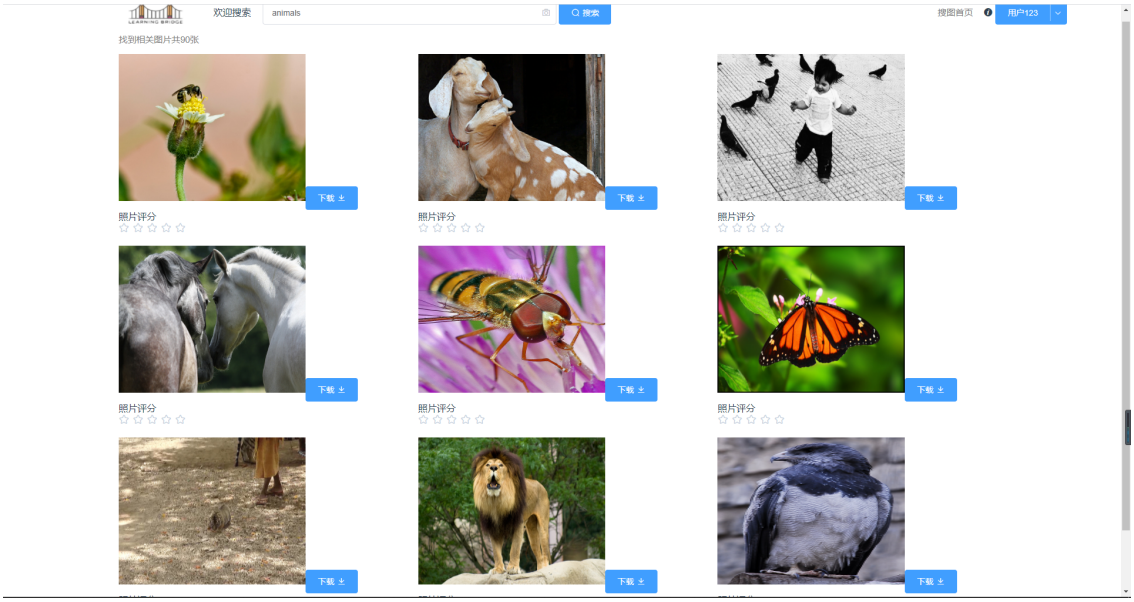
Ⅱ. Initiation of Action

1. It has a search button .
2. Users can also reset images to be uploaded through clicking the cross icon at the right top of the picture.

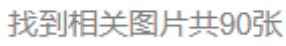
At the position of "Reset", when the mouse cursor moves over this area, a small cross appears that can be clicked to delete the uploaded image.

Ⅲ. Review of Results

1. Provide an overview of the results



2. The system provides an overview of the results, which includes the total number of the result.

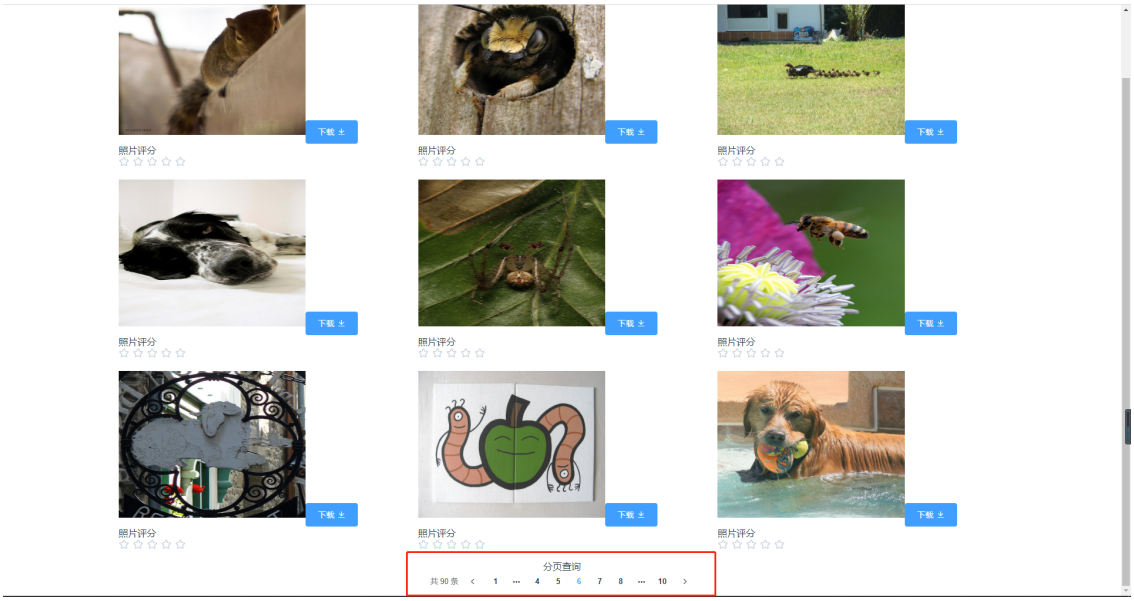


Ⅳ. Refinement

1. Allow changing search parameters (e.g. select certain category/tag) when reviewing results

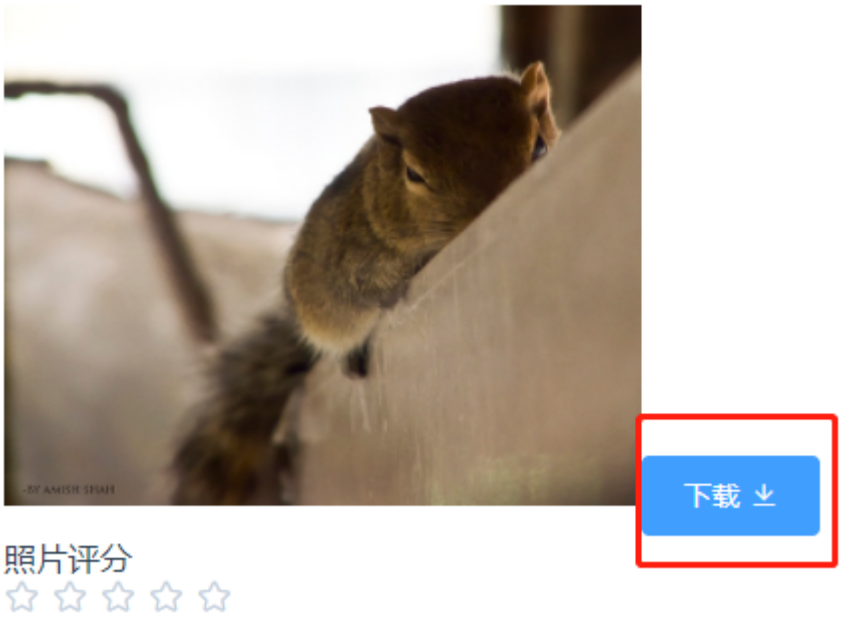
- [x] (Relevant search)

2. At the bottom of the page there is a paging button, the user can use the paging button to page query, get results on different pages.



V. Use

1. Users can download the images they like locally by pressing the *"Download"* button



2. Users can rate each image according to their preferences

System Implementation:

1. Front end Main functions:

Ⅰ. Search for images by keyword

```
     findimages(val){
          Vue.axios.post('/api/searchByKey?
key='+val+'&page=1').then((response)=>{
              console.log(response);
              this.$router.push({path:'/result',query:
{'result':response.data,'q':val}});
          }).catch((error)=>{
        console.log(error);
        this.$router.push({path:'/result',query:{'result':"no",'q':val}});
      });
      },
      searchImg(){
          if(!this.input){
          this.$router.push('/');
          return;
           }
           this.findimages(this.input);
      }
```

Ⅱ. Search for images by a similar picture

```
    submitPicture(){
           this.$refs.upload.submit();
      }
```

```
    <el-dialog
  title="智能识图"
  :visible.sync="dialogVisible"
  width="50%"
  :before-close="handleClose">
```

```
  <el-tooltip class="item" effect="dark" content="智能识图：识别人物、搜索服饰、寻找高清素材、浏览相似美图，尽在智能识图！   上传一张图片，我们将会为您展示相似图片。"
placement="top" :enterable="false" >
          <i class="el-icon-question"></i>
  </el-tooltip>
  <el-upload
  class="upload-demo"
  drag
  ref="upload"
  action="/api/imgUpload"
  multiple
  :on-preview="handlePreview"
  :on-remove="handleRemove"
  :auto-upload="false"
   :limit="1"
   :on-exceed="handleExceed"
  :on-success="handleSuccess"
  :file-list="fileList"
   list-type="picture">

   <p>本地上传</p>
  <i class="el-icon-upload"></i>
  <div class="el-upload__text">拖拽图片到此处试试，或<em>点击上传</em></div>
  <div class="el-upload__tip" slot="tip">只能上传jpg/png文件，且不超过500kb</div>
   </el-upload>
  <span slot="footer" class="dialog-footer">
    <el-button @click="dialogVisible = false">取 消</el-button>
    <el-button type="primary" @click="submitPicture">识 图 一 下</el-button>
  </span>
   </el-dialog>
```

Ⅲ. Download File

```
doDownload(val){
        if(val == 1){
          this.downloadByBlob(this.image1,'img1');
        }
        else if(val == 2){
          this.downloadByBlob(this.image2,'img2');
        }
        else if(val == 3){
          this.downloadByBlob(this.image3,'img3');
        }
        else if(val == 4){
          this.downloadByBlob(this.image4,'img4');
        }
        else if(val == 5){
          this.downloadByBlob(this.image5,'img5');
        }
        else if(val == 6){
          this.downloadByBlob(this.image6,'img6');
        }
        else if(val == 7){
          this.downloadByBlob(this.image7,'img7');
        }
        else if(val == 8){
          this.downloadByBlob(this.image8,'img8');
```

```
        }
        else if(val == 9){
          this.downloadByBlob(this.image9,'img9');
        }
      },
      download(href, name) {
  let eleLink = document.createElement('a')
  eleLink.download = name
  eleLink.href = href
  eleLink.click()
  eleLink.remove()
},
      downloadByBlob(url,name) {
    let image = new Image()
    image.setAttribute('crossOrigin', 'anonymous')
    image.src = url
    image.onload = () => {
      let canvas = document.createElement('canvas')
      canvas.width = image.width
      canvas.height = image.height
      let ctx = canvas.getContext('2d')
      ctx.drawImage(image, 0, 0, image.width, image.height)
      canvas.toBlob((blob) => {
        let url = URL.createObjectURL(blob)
        download(url,name)
        // 用完释放URL对象
        URL.revokeObjectURL(url)
      })
    }
  }
```

2. Back end Main functions:

Ⅰ. Search for images by keyword

```
@app.route('/imgUpload', methods=['GET', 'POST'])
#def allowed_file(filename):
#    return '.' in filename and \
#           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

def upload_img():
    print("image upload")
    result = 'static/result'
    if not gfile.Exists(result):
          os.mkdir(result)
    shutil.rmtree(result)

    if request.method == 'POST' or request.method == 'GET':
        print(request.method)
        # check if the post request has the file part
        if 'file' not in request.files:
            print('No file part')
            return redirect(request.url)

        file = request.files['file']
```

```
        print(file.filename)
        # if user does not select file, browser also
        # submit a empty part without filename
        if file.filename == '':

            print('No selected file')
            return redirect(request.url)
        if file:# and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            inputloc = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            recommend(inputloc, extracted_features)
            os.remove(inputloc)
            image_path = "/static/result"
            image_list =[os.path.join(image_path, file) for file in
os.listdir(result)
                                if not file.startswith('.')]
            images = {
            'image0':image_list[0],
            'image1':image_list[1],
            'image2':image_list[2],
            'image3':image_list[3],
            'image4':image_list[4],
            'image5':image_list[5],
            'image6':image_list[6],
            'image7':image_list[7],
            'image8':image_list[8]
              }
            return jsonify(images)
```

Ⅱ. Search for images by a similar picture

```
@app.route('/searchByKey', methods=['GET', 'POST'])

def findImg():
    result = 'static/result'
    if not gfile.Exists(result):
        os.mkdir(result)
    else:
        shutil.rmtree(result)
        os.mkdir(result)

    if request.method == 'POST' or request.method == 'GET':
        print(request.method)
        key = request.args.get("key")
        page = request.args.get("page")
        print(page)
        page = int(page)
        file = "database/tags/" + key + ".txt"
        imgList = []
        try:
            f = open(file, mode='r')
            f.readline()

            #分页，如果是第2页，就pre=9，先空读9个
            pre = (page-1)*9
            for j in range(pre):
```

```
                f.readline()

            for i in range(9):
                line = f.readline()
                line = line.strip('\n')
                line = "im"+line+".jpg"
                imgList.append(line)
            f.close()
        except FileNotFoundError:
            print("file does not exist")
            return redirect(request.url)

        for item in imgList:
            copyfile("database/dataset/" + item, "static/result/" + item)

        image_path = "/static/result"
        image_list = [os.path.join(image_path, file) for file in
os.listdir(result)
                      if not file.startswith('.')]

        images = {
            'image0': image_list[0],
            'image1': image_list[1],
            'image2': image_list[2],
            'image3': image_list[3],
            'image4': image_list[4],
            'image5': image_list[5],
            'image6': image_list[6],
            'image7': image_list[7],
            'image8': image_list[8]
        }
        return jsonify(images)
```