### storage.py

# -\*- coding: UTF-8 -\*-

from django.core.files.storage import FileSystemStorage

from django.http import HttpResponse

class ImageStorage(FileSystemStorage):

from django.conf import settings

def \_\_init\_\_(self, location=settings.MEDIA\_ROOT, base\_url=settings.MEDIA\_URL):

super(ImageStorage, self).\_\_init\_\_(location, base\_url)

def \_save(self, name, content):

import os, time, random

ext = os.path.splitext(name)[1]

d = os.path.dirname(name)

origin = os.path.basename(os.path.splitext(name)[0])

fn = time.strftime('%Y%m%d%H%M%S')

fn = origin + fn

name = os.path.join(d, fn + ext)

return super(ImageStorage, self).\_save(name, content)

### urls.py

from django.contrib import admin

from django.urls import path, re\_path

from django.views.generic import TemplateView

from django.views.static import serve

import user.views

from eyes import settings

# from user.views import ChatConsumer

urlpatterns = [

path('admin/', admin.site.urls),

path('login/', user.views.login),

path('register/', user.views.register),

path('receive/', user.views.receive\_origin),

path('upload/', user.views.receive\_upload),

path('getList/', user.views.get\_photo\_list\_for\_doctor),

path('deletePicture/', user.views.delete\_picture),

path('revisePictureName/', user.views.revise\_picture\_name),

re\_path(r'^media/(?P<path>.\*)$', serve, {'document\_root': settings.MEDIA\_ROOT}),

re\_path(r'^', TemplateView.as\_view(template\_name="index.html")),

]

### views.py

import subprocess

import torch

import numpy as np

import cv2

import os

import sys

import json

import time

from django.core.paginator import Paginator, PageNotAnInteger, EmptyPage

from django.utils.encoding import escape\_uri\_path

from django.http import JsonResponse, FileResponse, StreamingHttpResponse

from user.models import Doctor, Patient, Photo

from django.views import View

from eyes.settings import MEDIA\_ROOT

from django.contrib.auth.hashers import make\_password, check\_password

from PIL import Image

from torchvision import transforms

BASEURL ="http://10.251.0.251:8000/media/test/"

def login(request):

if request.method == 'POST':

data\_json = json.loads(request.body)

uname = data\_json.get('username')

pwd = data\_json.get('password')

print(uname)

print(pwd)

user = Doctor.objects.filter(doctor\_username=uname)

if len(user) == 0:

return JsonResponse({'success': False, 'message': '用户不存在'})

if len(user) > 0:

if check\_password(pwd, (user[0]).doctor\_password):

return JsonResponse({'success': True, 'message': '登录成功', 'userID': user[0].doctor\_id,

'role': 'Doctor'})

else:

return JsonResponse({'success': False, 'message': '密码错误'})

else:

return JsonResponse({})

def register(request):

if request.method == 'POST':

data\_json = json.loads(request.body)

new\_name = data\_json.get('username')

temp\_pwd = data\_json.get('password')

new\_email = data\_json.get('email')

new\_ph = data\_json.get('phone')

new\_realname = data\_json.get('realname')

new\_sex = data\_json.get('sex')

print(new\_name)

new\_pwd = make\_password(temp\_pwd, None, 'pbkdf2\_sha1')

print(new\_pwd)

print(new\_email)

print(new\_ph)

print(new\_realname)

print(new\_sex)

if new\_name is None:

return JsonResponse({'success': False, 'message': '未输入'})

else:

space = Doctor.objects.filter(doctor\_username=new\_name)

if len(space) > 0:

return JsonResponse({'success': False, 'message': '用户名已存在'})

else:

new\_d = Doctor()

new\_d.doctor\_username = new\_name

new\_d.doctor\_password = new\_pwd

new\_d.doctor\_email = new\_email

new\_d.doctor\_phone = new\_ph

new\_d.doctor\_realname = new\_realname

if new\_sex == 0 or new\_sex == 1:

new\_d.doctor\_sex = new\_sex

new\_d.save()

return JsonResponse({'success': True, 'message': '注册成功'})

else:

return JsonResponse({})

def get\_photo\_list\_for\_doctor(request):

if request.method == 'POST':

data\_json = json.loads(request.body)

u\_id = data\_json.get('user\_id')

print(u\_id)

pagenum = data\_json.get('pagenum')

pagesize = data\_json.get('pagesize')

title = data\_json.get('title')

result = Photo.objects.filter(photo\_doctor\_\_doctor\_id=u\_id, photo\_realname\_\_contains=title)

apply\_list = list(

result.values('photo\_id', 'photo\_realname', 'photo\_savename', 'photo\_origin', 'photo\_upload',

'photo\_promap', 'photo\_patient\_\_patient\_id'))

total = len(apply\_list)

paginator = Paginator(apply\_list, pagesize)

try:

p\_list = paginator.page(pagenum)

except PageNotAnInteger as e:

p\_list = paginator.page(1)

except EmptyPage as e:

if int(pagenum) > paginator.num\_pages:

p\_list = paginator.page(paginator.num\_pages)

else:

p\_list = paginator.page(1)

return JsonResponse({'success': True, 'message': '返回list成功', 'imageList': list(p\_list), 'total': total},

safe=False, status=200,

json\_dumps\_params={'ensure\_ascii': False})

else:

return JsonResponse({})

def receive\_origin(request):

image = request.FILES.get('pic\_img')

u\_id = request.POST.get('user\_id')

obj = Photo.objects.create(photo\_img=image)

obj.photo\_doctor = Doctor.objects.get(doctor\_id=u\_id)

obj.photo\_realname = os.path.basename(os.path.splitext(image.name)[0])

obj.photo\_savename = os.path.basename(os.path.splitext(obj.photo\_img.path)[0])

obj.photo\_origin = BASEURL + obj.photo\_savename + "\_origin.png"

obj.photo\_promap = BASEURL + obj.photo\_savename + "\_promap.png"

obj.save()

if BASEURL == "http://10.251.0.251:8000/media/test/":

subprocess.Popen("python ./test/test.py" + " " + os.path.basename(obj.photo\_img.path), shell=True)

else:

subprocess.Popen("python ./test/local\_test.py" + " " + os.path.basename(obj.photo\_img.path), shell=True)

return JsonResponse('message="上传成功', safe=False)

def receive\_upload(request):

image = request.FILES.get('pic\_img')

p\_id = request.POST.get('photo\_id')

photo = Photo.objects.get(photo\_id=p\_id)

base\_dir = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

file\_path = os.path.join(base\_dir, 'media', 'test', photo.photo\_savename)

with open(file\_path, 'wb+') as f:

for chunk in image.chunks():

f.write(chunk)

img = cv2.imread(file\_path)

cv2.imwrite('./media/test/' + photo.photo\_savename + '\_upload.png', img, )

os.remove(file\_path)

photo.photo\_upload = BASEURL + photo.photo\_savename + "\_upload.png"

photo.save()

return JsonResponse('message="上传成功', safe=False)

def delete\_picture(request):

if request.method == 'POST':

data\_json = json.loads(request.body)

photo\_id = data\_json.get('photoID')

print(photo\_id)

p = Photo.objects.filter(photo\_id=photo\_id)

if len(p) == 0:

return JsonResponse({'success': False, 'message': '图片'}, status=404)

else:

photo = Photo.objects.get(photo\_id=photo\_id)

save\_name = photo.photo\_savename

try:

if photo.photo\_upload is not None:

os.remove('./media/test/' + save\_name + '\_upload.png')

os.remove('./media/test/' + save\_name + '\_promap.png')

os.remove('./media/test/' + save\_name + '\_origin.png')

except FileNotFoundError as e:

print(e)

photo.delete()

return JsonResponse({'success': True, 'message': '删除成功'}, status=200)

else:

return JsonResponse({})

def revise\_picture\_name(request):

if request.method == 'POST':

data\_json = json.loads(request.body)

new\_name = data\_json.get('newName')

photo\_id = data\_json.get('photoID')

picture = Photo.objects.filter(photo\_id=photo\_id)

if len(picture) == 0:

return JsonResponse({'success': False, 'message': '不存在该图片'}, status=404)

else:

obj = Photo.objects.get(photo\_id=photo\_id)

obj.photo\_realname = new\_name

obj.save()

return JsonResponse({'success': True, 'message': '修改成功'}, status=200)

else:

return JsonResponse({})

from django.db import models

from eyes.storage import ImageStorage

### models.py

class Doctor(models.Model):

doctor\_id = models.AutoField(db\_column='doctor\_ID', primary\_key=True) # Field name made lowercase.

doctor\_username = models.CharField(max\_length=255)

doctor\_password = models.CharField(max\_length=255)

doctor\_realname = models.CharField(max\_length=255, null=True)

doctor\_email = models.CharField(max\_length=255, blank=True, null=True)

doctor\_icon = models.CharField(max\_length=255, blank=True, null=True)

doctor\_phone = models.CharField(max\_length=255, blank=True, null=True)

doctor\_sex = models.SmallIntegerField(blank=True, null=True)

class Meta:

managed = False

db\_table = 'user\_doctor'

class Manager(models.Model):

manager\_id = models.AutoField(db\_column='manager\_ID', primary\_key=True) # Field name made lowercase.

manager\_name = models.CharField(max\_length=20)

manager\_password = models.CharField(max\_length=25)

class Meta:

managed = False

db\_table = 'user\_manager'

class Patient(models.Model):

patient\_name = models.CharField(max\_length=30)

patient\_id = models.AutoField(db\_column='patient\_ID', primary\_key=True) # Field name made lowercase.

patient\_phone = models.CharField(max\_length=11, blank=True, null=True)

patient\_address = models.CharField(max\_length=30, blank=True, null=True)

patient\_email = models.CharField(max\_length=255, blank=True, null=True)

patient\_height = models.DecimalField(max\_digits=3, decimal\_places=3, blank=True, null=True)

patient\_weight = models.DecimalField(max\_digits=3, decimal\_places=3, blank=True, null=True)

patient\_age = models.IntegerField(blank=True, null=True)

patient\_sex = models.CharField(max\_length=1, blank=True, null=True)

class Meta:

managed = False

db\_table = 'user\_patient'

class Photo(models.Model):

photo\_id = models.AutoField(db\_column='photo\_ID', primary\_key=True) # Field name made lowercase.

photo\_realname = models.CharField(max\_length=255)

photo\_img = models.FileField(max\_length=255, upload\_to='test', storage=ImageStorage())

photo\_doctor = models.ForeignKey(Doctor, models.DO\_NOTHING, db\_column='photo\_doctor\_ID', blank=True,

null=True) # Field name made lowercase.

photo\_promap = models.CharField(max\_length=255, blank=True, null=True)

photo\_origin = models.CharField(max\_length=255, blank=True, null=True)

photo\_savename = models.CharField(max\_length=255)

photo\_patient = models.ForeignKey(Patient, models.DO\_NOTHING, db\_column='photo\_patient\_ID', blank=True,

null=True) # Field name made lowercase.

photo\_upload = models.CharField(max\_length=255, blank=True, null=True)

class Meta:

managed = False

db\_table = 'user\_photo'

### test.py

import torch

import numpy as np

import cv2

from PIL import Image

from torchvision import transforms

import os

import sys

unloader = transforms.ToPILImage()

os.environ["CUDA\_VISIBLE\_DEVICES"] = '2,1,0,3'

model = torch.load('./test/DS\_Model\_100.pkl')

def test():

image\_path = './media/test/' + sys.argv[1]

roi\_path = './test/test\_mask.gif'

if hasattr(torch.cuda, 'empty\_cache'):

torch.cuda.empty\_cache()

img = cv2.imread(image\_path)

cv2.imwrite('./media/test/' + os.path.splitext(sys.argv[1])[0] + '\_origin.png', img, )

img = cv2.resize(img, (568, 584))

os.remove(image\_path)

img = np.array(img, np.float32).transpose(2, 0, 1) / 255.0

roi = np.array(Image.open(roi\_path))

roi = cv2.resize(roi, (568, 584))

roi[roi >= 0.5] = 1

roi[roi <= 0.5] = 0

img1 = torch.Tensor(img)

img1 = img1.unsqueeze(0)

img1 = img1.cuda()

if hasattr(torch.cuda, 'empty\_cache'):

torch.cuda.empty\_cache()

pred = model(img1)

pred = (pred.squeeze(0)).squeeze(0)

pred = pred.cpu().detach().numpy()

pred = pred \* roi

pred = Image.fromarray(np.uint8(pred \* 255))

pred.convert('L').save('./media/test/' + os.path.splitext(sys.argv[1])[0] + '\_promap.png')

if \_\_name\_\_ == '\_\_main\_\_':

test()

### router/index.js

import { createRouter, createWebHistory } from "vue-router";

const routes = [

/\* 公共路由\*/

{

/\*用于指定初始界面的路由 \*/

path: "/",

redirect: "/login",

},

{

path: "/register",

name: "Register",

component: () => import("../views/Public/Register"),

},

{

path: "/login",

name: "Login",

component: () =>

import(

/\* webpackChunkName: "login" \*/

"../views/Public/Login"

),

},

{

path: "/home",

name: "Home",

component: () =>

import(

/\* webpackChunkName: "login" \*/

"../views/Public/Home"

),

},

/\* 患者的页面的路由\*/

{

path: "/patient",

name: "Patient",

children: [

{

path: "upload",

name: "/patient/upload",

meta: {

title: "上传",

},

component: () => import("../views/Patient/Upload"),

},

{

path: "user",

name: "/patient/user",

meta: {

title: "个人中心",

},

component: () => import("../views/Patient/User"),

},

],

component: () =>

import(

/\* webpackChunkName: "login" \*/

"../views/Patient/Home"

),

},

/\* 医生的页面的路由\*/

{

path: "/doctor",

name: "Doctor",

children: [

{

path: "user",

name: "/doctor/user",

meta: {

title: "个人中心",

},

component: () => import("../views/Doctor/User"),

},

{

path: "upload",

name: "/doctor/upload",

meta: {

title: "上传",

},

component: () => import("../views/Doctor/Upload"),

},

],

component: () =>

import(

/\* webpackChunkName: "login" \*/

"../views/Doctor/Home"

),

},

/\* 管理员的页面的路由\*/

{

path: "/administrator",

name: "Administrator",

children: [

{

path: "user",

name: "user",

meta: {

title: "个人中心",

},

component: () => import("../views/Administrator/User"),

},

{

path: "admin/patient",

name: "admin/patient",

meta: {

title: "患者管理",

},

component: () => import("../views/Administrator/Patient"),

},

{

path: "admin/doctor",

name: "admin/doctor",

meta: {

title: "医生管理",

},

component: () => import("../views/Administrator/Doctor"),

},

{

path: "admin/drug",

name: "admin/drug",

meta: {

title: "药品管理",

},

component: () => import("../views/Administrator/Drug"),

},

{

path: "admin/operation",

name: "admin/operation",

meta: {

title: "手术管理",

},

component: () => import("../views/Administrator/Operation"),

},

],

component: () =>

import(

/\* webpackChunkName: "login" \*/

"../views/Administrator/Home"

),

},

];

const router = createRouter({

history: createWebHistory("/"),

routes,

});

export default router;

### store/index.js

import { createStore } from 'vuex'

import createPersistedState from "vuex-persistedstate"

export default createStore({

state: {

tagsList: [],

collapse: false,

islogin: false,

isAdministrator: false,

username: "未登录",

user\_id: "",

role: 2,

},

mutations: {

login(state, username) {

state.islogin = true

state.username = username

},

changeName(state,username){

state.username = username

},

storeId(state, userid) {

state.user\_id = userid

},

loginout(state) {

state.islogin = false

state.username = "未登录"

state.user\_id = ""

state.tagsList=[]

},

setRole(state, role) {

state.role = role

},

delTagsItem(state, data) {

state

.tagsList

.splice(data.index, 1);

},

setTagsItem(state, data) {

state

.tagsList

.push(data)

},

clearTags(state) {

state.tagsList = []

},

closeTagsOther(state, data) {

state.tagsList = data;

},

closeCurrentTag(state, data) {

for (let i = 0, len = state.tagsList.length; i < len; i++) {

const item = state.tagsList[i];

if (item.path === data.$route.fullPath) {

if (i < len - 1) {

data

.$router

.push(state.tagsList[i + 1].path);

} else if (i > 0) {

data

.$router

.push(state.tagsList[i - 1].path);

} else {

data

.$router

.push("/");

}

state

.tagsList

.splice(i, 1);

break;

}

}

},

// 侧边栏折叠

hadndleCollapse(state, data) {

state.collapse = data;

}

},

actions: {},

modules: {},

plugins: [createPersistedState()]

})

### axios.js

"use strict";

import axios from "axios";

// Full config: https://github.com/axios/axios#request-config

// axios.defaults.baseURL = process.env.baseURL || process.env.apiUrl || ''

// axios.defaults.headers.common['Authorization'] = AUTH\_TOKEN

// axios.defaults.headers.post['Content-Type'] = 'application/x-www-form-urlencoded'

let config = {

baseURL: "http://10.251.0.251:8000/",

//baseURL: "http://localhost:8000/",

timeout: 60 \* 1000, // Timeout

withCredentials: true, // Check cross-site Access-Control

};

// 创建Axios对象

const Axios = axios.create(config);

// 请求拦截器

Axios.interceptors.request.use(

function(config) {

// Do something before request is sent

return config;

},

function(error) {

// Do something with request error

return Promise.reject(error);

}

);

// 响应拦截器

Axios.interceptors.response.use(

function(response) {

// Do something with response data

return response;

},

function(error) {

// Do something with response error

return Promise.reject(error);

}

);

export default (app) => {

// 挂载axios到Vue对象

app.config.globalProperties.$http = Axios;

};

### request.js

import axios from 'axios';

const service = axios.create({

// process.env.NODE\_ENV === 'development' 来判断是否开发环境

// easy-mock服务挂了，暂时不使用了

// baseURL: 'https://www.easy-mock.com/mock/592501a391470c0ac1fab128',

timeout: 5000

});

service.interceptors.request.use(

config => {

return config;

},

error => {

console.log(error);

return Promise.reject();

}

);

service.interceptors.response.use(

response => {

if (response.status === 200) {

return response.data;

} else {

Promise.reject();

}

},

error => {

console.log(error);

return Promise.reject();

}

);

export default service;

### main.js

import { createApp } from 'vue'

import App from './App.vue'

import router from './router'

import store from './store'

import installElementPlus from './plugins/element'

import './assets/css/icon.css'

import installAxios from './plugins/axios'

import \* as echarts from 'echarts'

const app = createApp(App)

installElementPlus(app)

installAxios(app)

app.config.globalProperties.$echarts = echarts

app

.use(store)

.use(router)

.mount('#app')

### index.html

<!DOCTYPE html>

<html lang="">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width,initial-scale=1.0">

<title>眼底血管分割云平台</title>

<link rel="stylesheet" href="https://at.alicdn.com/t/font\_830376\_qzecyukz0s.css">

</head>

<body>

<noscript>

<strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't work properly without JavaScript enabled.

Please enable it to continue.</strong>

</noscript>

<div id="app"></div>

<!-- built files will be auto injected -->

</body>

</html>

### App.vue

<template>

<keep-alive>

<router-view v-if="$route.meta.keepAlive" />

</keep-alive>

<router-view v-if="!$route.meta.keepAlive" />

</template>

<script>

export default {};

</script>

<style>

@import "./assets/css/main.css";

@import "./assets/css/color-dark.css";

</style>

### color-dark.css

.header {

background-color: #242f42;

}

.login-wrap {

background: #324157;

}

.plugins-tips {

background: #eef1f6;

}

.plugins-tips a {

color: #20a0ff;

}

.el-upload--text em {

color: #20a0ff;

}

.pure-button {

background: #20a0ff;

}

.tags-li.active {

border: 1px solid #409eff;

background-color: #409eff;

}

.message-title {

color: #20a0ff;

}

.collapse-btn:hover {

background: rgb(40, 52, 70);

}

### main.css

\* {

margin: 0;

padding: 0;

}

html,

body,

#app,

.wrapper {

width: 100%;

height: 100%;

overflow: hidden;

}

body {

font-family: 'PingFang SC', "Helvetica Neue", Helvetica, "microsoft yahei", arial, STHeiTi, sans-serif;

}

a {

text-decoration: none

}

.content {

width: auto;

height: calc(100vh - 70px);

padding: 10px;

background: #f0f0f0;

overflow-y: scroll;

box-sizing: border-box;

}

.content-collapse {

left: 65px;

}

.container {

padding: 30px;

background: #fff;

border: 1px solid #ddd;

border-radius: 5px;

}

.crumbs {

margin: 10px 0;

}

.el-table th {

background-color: #f5f7fa !important;

}

.pagination {

margin: 20px 0;

text-align: right;

}

.plugins-tips {

padding: 20px 10px;

margin-bottom: 20px;

}

.el-button+.el-tooltip {

margin-left: 10px;

}

.el-table tr:hover {

background: #f6faff;

}

.mgb20 {

margin-bottom: 20px;

}

.move-enter-active,

.move-leave-active {

transition: opacity .1s ease;

}

.move-enter-from,

.move-leave-to {

opacity: 0;

}

/\*BaseForm\*/

.form-box {

width: 600px;

}

.form-box .line {

text-align: center;

}

.el-time-panel\_\_content::after,

.el-time-panel\_\_content::before {

margin-top: -7px;

}

.el-time-spinner\_\_wrapper .el-scrollbar\_\_wrap:not(.el-scrollbar\_\_wrap--hidden-default) {

padding-bottom: 0;

}

/\*Upload\*/

.pure-button {

width: 150px;

height: 40px;

line-height: 40px;

text-align: center;

color: #fff;

border-radius: 3px;

}

.g-core-image-corp-container .info-aside {

height: 45px;

}

.el-upload--text {

background-color: #fff;

border: 1px dashed #d9d9d9;

border-radius: 6px;

box-sizing: border-box;

width: 360px;

height: 180px;

text-align: center;

cursor: pointer;

position: relative;

overflow: hidden;

}

.el-upload--text .el-icon-upload {

font-size: 67px;

color: #97a8be;

margin: 40px 0 16px;

line-height: 50px;

}

.el-upload--text {

color: #97a8be;

font-size: 14px;

text-align: center;

}

.el-upload--text em {

font-style: normal;

}

/\*VueEditor\*/

.ql-container {

min-height: 400px;

}

.ql-snow .ql-tooltip {

transform: translateX(117.5px) translateY(10px) !important;

}

.editor-btn {

margin-top: 20px;

}

/\*markdown\*/

.v-note-wrapper .v-note-panel {

min-height: 500px;

}

### header.vue

<template>

<div class="header">

<!-- 折叠按钮 -->

<div class="logo">眼底血管分割云平台</div>

<div class="header-right">

<div class="header-user-con">

<!-- 消息中心 -->

<div class="btn-bell">

<el-tooltip effect="dark" placement="bottom">

<router-link to="/Notification">

<i class="el-icon-bell"></i>

</router-link>

</el-tooltip>

</div>

<!-- 用户头像 -->

<div class="user-avator">

<router-link to="/user"> </router-link>

</div>

<!-- 用户名下拉菜单 -->

<el-dropdown class="user-name" trigger="click" @command="handleCommand">

<span class="el-dropdown-link">

{{ username }}

<i class="el-icon-caret-bottom"></i>

</span>

<template #dropdown>

<el-dropdown-menu v-if="islogin">

<el-dropdown-item divided command="loginout"

>退出登录</el-dropdown-item

>

</el-dropdown-menu>

<el-dropdown-menu v-else>

<router-link to="/login">

<el-dropdown-item>前往登录</el-dropdown-item>

</router-link>

</el-dropdown-menu>

</template>

</el-dropdown>

</div>

</div>

</div>

</template>

<script>

export default {

data() {

return {

fullscreen: false,

name: "linxin",

message: 2,

};

},

computed: {

username() {

let username = this.$store.state.username;

return username;

// return username ? username : this.name;

},

collapse() {

return this.$store.state.collapse;

},

islogin() {

return this.$store.state.islogin;

},

},

methods: {

// 用户名下拉菜单选择事件

handleCommand(command) {

if (command == "loginout") {

this.$store.commit("loginout");

this.$router.push("/login");

}

},

// 侧边栏折叠

collapseChage() {

this.$store.commit("hadndleCollapse", !this.collapse);

},

},

mounted() {

if (document.body.clientWidth < 1500) {

this.collapseChage();

}

},

};

</script>

<style scoped>

.header {

position: relative;

box-sizing: border-box;

width: 100%;

height: 70px;

font-size: 22px;

color: #fff;

}

.collapse-btn {

float: left;

padding: 0 21px;

cursor: pointer;

line-height: 70px;

}

.header .logo {

float: left;

width: 250px;

margin-left:50px;

line-height: 70px;

}

.header-right {

float: right;

padding-right: 50px;

}

.header-user-con {

display: flex;

height: 70px;

align-items: center;

}

.btn-fullscreen {

transform: rotate(45deg);

margin-right: 5px;

font-size: 24px;

}

.btn-bell,

.btn-fullscreen {

position: relative;

width: 30px;

height: 30px;

text-align: center;

border-radius: 15px;

cursor: pointer;

}

.btn-bell-badge {

position: absolute;

right: 0;

top: -2px;

width: 8px;

height: 8px;

border-radius: 4px;

background: #f56c6c;

color: #fff;

}

.btn-bell .el-icon-bell {

color: #fff;

}

.user-name {

margin-left: 10px;

}

.user-avator {

margin-left: 20px;

}

.user-avator img {

display: block;

width: 40px;

height: 40px;

border-radius: 50%;

}

.el-dropdown-link {

color: #fff;

cursor: pointer;

}

.el-dropdown-menu\_\_item {

text-align: center;

}

</style>

### Home/index.vue

<template>

<div class="about">

<v-header />

<div class="content">

<router-view v-slot="{ Component }">

<transition name="move" mode="out-in">

<keep-alive :include="tagsList">

<component :is="Component" />

</keep-alive>

</transition>

</router-view>

<!-- <el-backtop target=".content"></el-backtop> -->

</div>

</div>

</template>

<script>

import vHeader from "../../../components/Doctor/Header";

import vSidebar from "../../../components/Doctor/Sidebar";

export default {

components: {

vHeader,

vSidebar,

},

computed: {

tagsList() {

return this.$store.state.tagsList.map((item) => item.name);

},

collapse() {

return this.$store.state.collapse;

},

},

};

</script>

### Upload/index.vue

<template>

<div>

<div class="body" style="min-width:1000px">

<el-row>

<!-- 左侧的固定快捷按钮 -->

<el-col :span="4">

<el-affix :offset="120">

<upload-left

:props="props"

@downloadAllImage="downloadAllImage"

@deleteAllImage="deleteAllImage"

@getImageList="getImageList"

/>

</el-affix>

</el-col>

<!-- 中间主体部分 -->

<el-col :span="16" style="min-width:600px;">

<upload-middle

:props="props"

@downloadAllImage="downloadAllImage"

@deleteAllImage="deleteAllImage"

@getImageList="getImageList"

/>

</el-col>

<!-- 右边的分页 -->

<el-col :span="4">

<el-affix :offset="120">

<div style="margin-left:4vw;">

<el-row>

<el-select

v-model="props.pagesize"

placeholder="Select"

style="width:100px;margin-left:5px"

@change="handleSizeChange(props.pagesize)"

>

<el-option label="1条/页" :value="1"> </el-option>

<el-option label="3条/页" :value="3"> </el-option>

<el-option label="5条/页" :value="5"> </el-option>

<el-option label="10条/页" :value="10"> </el-option>

</el-select>

</el-row>

<el-row style="margin-left:5px; inline-block;">

<el-button-group>

<el-button

style="width:50px; inline-block;"

icon="el-icon-arrow-left"

:disabled="props.pagenum <= 1"

@click="handleCurrentChange(props.pagenum - 1)"

></el-button>

<el-button

style="width:50px; inline-block;"

icon="el-icon-arrow-right"

:disabled="props.pagenum >= props.total / props.pagesize"

@click="handleCurrentChange(props.pagenum + 1)"

>

</el-button>

</el-button-group>

</el-row>

<el-row>

<el-pagination

:currentPage="props.pagenum"

:page-size="props.pagesize"

layout="pager"

:total="props.total"

@current-change="handleCurrentChange"

>

</el-pagination>

</el-row>

</div>

</el-affix>

</el-col>

</el-row>

</div>

</div>

</template>

<script>

import { ElMessageBox, ElMessage } from "element-plus";

import UploadMiddle from "./components/Middle.vue";

import UploadLeft from "./components/Left.vue";

export default {

name: "Upload",

components: {

UploadLeft,

UploadMiddle,

},

data() {

return {

props: {

loadingNewPicture: false,

pagenum: 1,

pagesize: 5,

searchTitle: "",

total: 0,

imageList: [],

newName: "",

//baseURL: "http://localhost:8000/",

baseURL:"http://10.251.0.251:8000/"

},

};

},

created() {

this.getImageList();

},

methods: {

handleSizeChange(size) {

this.props.pagesize = size;

this.getImageList();

},

handleCurrentChange(current) {

this.props.pagenum = current;

this.getImageList();

},

downloadAllImage() {

var i, len;

for (i = 0, len = this.props.imageList.length; i < len; i++) {

this.downloadASetOfImage(this.props.imageList[i]);

}

},

downloadASetOfImage(singleImage) {

this.downloadIamge(

singleImage.photo\_origin,

singleImage.photo\_realname + "\_origin"

);

this.downloadIamge(

singleImage.photo\_upload,

singleImage.photo\_realname + "\_upload"

);

this.downloadIamge(

singleImage.photo\_promap,

singleImage.photo\_realname + "\_promap"

);

},

deleteAllImage() {

ElMessageBox.confirm(

"本操作无法撤回，您确认要清空本页图片吗？",

"Warning",

{

confirmButtonText: "OK",

cancelButtonText: "Cancel",

type: "warning",

}

)

.then(() => {

var i, len;

for (i = 0, len = this.props.imageList.length; i < len; i++) {

this.deleteASetOfImage(this.props.imageList[i]);

}

ElMessage({

type: "success",

message: "删除成功",

});

})

.catch(() => {

ElMessage({

type: "info",

message: "删除被取消",

});

});

},

async deleteASetOfImage(singleImage) {

await new Promise((resolve) => {

this.$http

.post(

"/deletePicture/",

JSON.stringify({ photoID: singleImage.photo\_id })

)

.then((res) => {

{

if (res.data.message === "删除成功") {

this.getImageList();

} else {

this.$message.error("删除失败");

}

}

});

resolve();

}).then(() => {});

},

async getImageList() {

this.props.loadingNewPicture = true;

await new Promise((resolve) => {

this.$http

.post(

"/getList/",

JSON.stringify({

title: this.props.searchTitle,

user\_id: this.$store.state.user\_id,

pagenum: this.props.pagenum,

pagesize: this.props.pagesize,

})

)

.then((res) => {

{

if (res.data.message === "返回list成功") {

this.props.imageList = res.data.imageList;

this.props.total = res.data.total;

} else {

this.$message.error("获取列表失败");

}

this.props.loadingNewPicture = false;

}

});

resolve();

}).then(() => {});

},

downloadIamge(imgsrc, name) {

//下载图片地址和图片名

let image = new Image();

// 解决跨域 Canvas 污染问题

image.setAttribute("crossOrigin", "anonymous");

image.onload = function() {

let canvas = document.createElement("canvas");

canvas.width = image.width;

canvas.height = image.height;

let context = canvas.getContext("2d");

context.drawImage(image, 0, 0, image.width, image.height);

let url = canvas.toDataURL("image/png"); //得到图片的base64编码数据

let a = document.createElement("a"); // 生成一个a元素

let event = new MouseEvent("click"); // 创建一个单击事件

a.download = name || "photo"; // 设置图片名称

a.href = url; // 将生成的URL设置为a.href属性

a.dispatchEvent(event); // 触发a的单击事件

};

image.src = imgsrc;

},

},

};

</script>

<style>

@import "style.css";

</style>

### Upload/style.css

#img {

font-size: 15px;

border-radius: 25px;

border: 2px solid #004b97;

padding: 20px;

width: 500px;

height: 50px;

}

.inline-block {

display: inline-block;

margin-right: 30px;

}

.card-header {

display: flex;

justify-content: space-between;

align-items: center;

}

.el-pager {

display: flex;

flex-direction: column;

}

.el-pager > li {

width: 100px;

}

### Middle.vue

<template>

<div class="container">

<!-- 顶部操作模块 -->

<upload-head

:props="myProps"

@downloadAllImage="downloadAllImage"

@deleteAllImage="deleteAllImage"

@getImageList="getImageList"

/>

<!-- 显示图片卡片 -->

<div v-loading="myProps.loadingNewPicture">

<div v-if="myProps.imageList.length > 0">

<div

v-for="singleImage in myProps.imageList"

:key="singleImage"

style="margin-top: 20px"

>

<photo-card

:singleImage="singleImage"

:props="myProps"

@getImageList="getImageList"

/>

</div>

</div>

<div v-else>

<el-empty description="暂未上传图片" />

</div>

</div>

</div>

</template>

<script>

import UploadHead from "./middle/Head.vue";

import PhotoCard from "./middle/PhotoCard.vue";

export default {

props: ["props"],

components: {

UploadHead,

PhotoCard,

},

data() {

return {

myProps: this.props,

};

},

methods: {

getImageList() {

this.$emit("getImageList");

},

downloadAllImage() {

this.$emit("downloadAllImage");

},

deleteAllImage() {

this.$emit("deleteAllImage");

},

},

};

</script>

### Left.vue

<template>

<div style="margin-left:6vw;">

<el-row>

<el-tooltip

class="item"

effect="dark"

content="上传新的眼底血管图片"

placement="right"

>

<el-upload

class="upload-demo"

:action="myProps.baseURL + 'receive/'"

:data="{ user\_id: $store.state.user\_id }"

:on-success="handleAvatarSuccess"

:on-error="handleUploadError"

list-type="false"

:show-file-list="false"

:before-upload="checkFile"

name="pic\_img"

>

<el-button

show-file-list="false"

type="primary"

icon="el-icon-upload"

circle

></el-button>

</el-upload>

</el-tooltip>

</el-row>

<el-row>

<el-tooltip

class="item"

effect="dark"

content="下载本页所有图片"

placement="right"

>

<el-button

style="margin-top:25px"

type="success"

icon="el-icon-download"

@click="downloadAllImage()"

circle

></el-button>

</el-tooltip>

</el-row>

<el-row>

<el-tooltip

class="item"

effect="dark"

content="清空本页所有图片"

placement="right"

>

<el-button

style="margin-top:25px"

@click="deleteAllImage()"

type="danger"

icon="el-icon-delete"

circle

></el-button>

</el-tooltip>

</el-row>

<el-row>

<el-tooltip

class="item"

effect="dark"

content="搜索图片"

placement="right"

>

<div>

<el-popover placement="right" :width="400" trigger="click">

<template #reference>

<el-button

style="margin-top:25px"

type="info"

icon="el-icon-search"

circle

></el-button>

</template>

<div>

<el-input

v-model="myProps.searchTitle"

@keyup.enter="getImageList"

placeholder="请输入图片名"

clearable

>

<template #append>

<el-button

@click="getImageList"

icon="el-icon-search"

></el-button>

</template>

</el-input>

</div>

</el-popover>

</div>

</el-tooltip>

</el-row>

</div>

</template>

<script>

export default {

props: ["props"],

data() {

return {

myProps: this.props,

};

},

methods: {

getImageList() {

this.$emit("getImageList");

},

handleUploadError(){

this.myProps.loadingNewPicture = false;

this.$message.error("上传失败");

},

checkFile(file) {

this.myProps.loadingNewPicture = true;

var fileExtension = file.name.substring(file.name.lastIndexOf(".") + 1);

if (file.type.startsWith("image")) {

if (fileExtension === "gif") {

this.$message.error("不支持上传gif图片");

this.myProps.loadingNewPicture = false;

return false;

}

} else {

this.$message.error("请上传图片");

this.myProps.loadingNewPicture = false;

return false;

}

},

handleAvatarSuccess() {

this.$message({

message: "上传成功",

type: "success",

});

this.myProps.loadingNewPicture = true;

setTimeout(() => {

this.myProps.imageList = [];

this.getImageList();

}, 10000);

},

downloadAllImage() {

this.$emit("downloadAllImage");

},

deleteAllImage() {

this.$emit("deleteAllImage");

},

},

};

</script>

### Edit.vue

<template>

<el-container style="text-align:center">

<el-main style="text-align:center">

<div :id="'tui-image-editor'+singleImage.photo\_id+index" style="margin:0 auto"></div>

</el-main>

<el-footer>

<el-button size="mini" @click="downloadImg">下载</el-button>

</el-footer>

</el-container>

</template>

<script>

import "tui-image-editor/dist/tui-image-editor.css";

import "tui-color-picker/dist/tui-color-picker.css";

const ImageEditor = require("tui-image-editor");

const locale\_zh = {

ZoomIn: "放大",

ZoomOut: "缩小",

Hand: "手掌",

History: "历史",

Resize: "调整宽高",

Crop: "裁剪",

DeleteAll: "全部删除",

Delete: "删除",

Undo: "撤销",

Redo: "反撤销",

Reset: "重置",

Flip: "镜像",

Rotate: "旋转",

Draw: "画",

Shape: "形状标注",

Icon: "图标标注",

Text: "文字标注",

Mask: "遮罩",

Filter: "滤镜",

Bold: "加粗",

Italic: "斜体",

Underline: "下划线",

Left: "左对齐",

Center: "居中",

Right: "右对齐",

Color: "颜色",

"Text size": "字体大小",

Custom: "自定义",

Square: "正方形",

Apply: "应用",

Cancel: "取消",

"Flip X": "X 轴",

"Flip Y": "Y 轴",

Range: "区间",

Stroke: "描边",

Fill: "填充",

Circle: "圆",

Triangle: "三角",

Rectangle: "矩形",

Free: "曲线",

Straight: "直线",

Arrow: "箭头",

"Arrow-2": "箭头2",

"Arrow-3": "箭头3",

"Star-1": "星星1",

"Star-2": "星星2",

Polygon: "多边形",

Location: "定位",

Heart: "心形",

Bubble: "气泡",

"Custom icon": "自定义图标",

"Load Mask Image": "加载蒙层图片",

Grayscale: "灰度",

Blur: "模糊",

Sharpen: "锐化",

Emboss: "浮雕",

"Remove White": "除去白色",

Distance: "距离",

Brightness: "亮度",

Noise: "噪音",

"Color Filter": "彩色滤镜",

Sepia: "棕色",

Sepia2: "棕色2",

Invert: "负片",

Pixelate: "像素化",

Threshold: "阈值",

Tint: "色调",

Multiply: "正片叠底",

Blend: "混合色",

Width: "宽度",

Height: "高度",

"Lock Aspect Ratio": "锁定宽高比例",

};

const customTheme = {

"common.bi.image": "", // 在这里换上你喜欢的logo图片

"common.bisize.width": "0px",

"common.bisize.height": "0px",

"common.backgroundImage": "none",

"common.backgroundColor": "#f3f4f6",

"common.border": "1px solid #444",

// header

"header.backgroundImage": "none",

"header.backgroundColor": "#f3f4f6",

"header.border": "0px",

// load button

"loadButton.backgroundColor": "#fff",

"loadButton.border": "1px solid #ddd",

"loadButton.color": "#222",

"loadButton.fontFamily": "NotoSans, sans-serif",

"loadButton.fontSize": "12px",

"loadButton.display": "none", // 可以直接隐藏掉

// download button

"downloadButton.backgroundColor": "#fdba3b",

"downloadButton.border": "1px solid #fdba3b",

"downloadButton.color": "#fff",

"downloadButton.fontFamily": "NotoSans, sans-serif",

"downloadButton.fontSize": "12px",

"downloadButton.display": "none", // 可以直接隐藏掉

// icons default

"menu.normalIcon.color": "#8a8a8a",

"menu.activeIcon.color": "#555555",

"menu.disabledIcon.color": "#434343",

"menu.hoverIcon.color": "#e9e9e9",

"submenu.normalIcon.color": "#8a8a8a",

"submenu.activeIcon.color": "#e9e9e9",

"menu.iconSize.width": "24px",

"menu.iconSize.height": "24px",

"submenu.iconSize.width": "32px",

"submenu.iconSize.height": "32px",

// submenu primary color

"submenu.backgroundColor": "#1e1e1e",

"submenu.partition.color": "#858585",

// submenu labels

"submenu.normalLabel.color": "#858585",

"submenu.normalLabel.fontWeight": "lighter",

"submenu.activeLabel.color": "#fff",

"submenu.activeLabel.fontWeight": "lighter",

// checkbox style

"checkbox.border": "1px solid #ccc",

"checkbox.backgroundColor": "#fff",

// rango style

"range.pointer.color": "#fff",

"range.bar.color": "#666",

"range.subbar.color": "#d1d1d1",

"range.disabledPointer.color": "#414141",

"range.disabledBar.color": "#282828",

"range.disabledSubbar.color": "#414141",

"range.value.color": "#fff",

"range.value.fontWeight": "lighter",

"range.value.fontSize": "11px",

"range.value.border": "1px solid #353535",

"range.value.backgroundColor": "#151515",

"range.title.color": "#fff",

"range.title.fontWeight": "lighter",

// colorpicker style

"colorpicker.button.border": "1px solid #1e1e1e",

"colorpicker.title.color": "#fff",

};

export default {

props: ["singleImage", "index"],

data() {

return {

instance: null,

imgVisible: false,

src: "",

};

},

created() {

if (this.index == "origin") {

this.src = this.singleImage.photo\_origin;

} else if (this.index == "upload") {

this.src = this.singleImage.photo\_upload;

} else {

this.src = this.singleImage.photo\_promap;

}

},

mounted() {

this.instance = new ImageEditor(

document.querySelector("#tui-image-editor"+this.singleImage.photo\_id+this.index),

{

includeUI: {

loadImage: {

path: this.src,

name: "SampleImage",

},

menu: [

"resize",

"crop",

"rotate",

"draw",

"shape",

"icon",

"text",

"filter",

],

initMenu: "filter",

uiSize: {

width: "1000px",

height: "700px",

},

menuBarPosition: "bottom",

theme: customTheme,

locale: locale\_zh,

},

cssMaxWidth: 700,

cssMaxHeight: 500,

selectionStyle: {

cornerSize: 20,

rotatingPointOffset: 70,

},

}

);

},

methods: {

downloadImg() {

const base64String = this.instance.toDataURL();

this.downloadIamge(

base64String,

this.singleImage.photo\_realname + "\_" + this.index

);

},

downloadIamge(imgsrc, name) {

//下载图片地址和图片名

let image = new Image();

// 解决跨域 Canvas 污染问题

image.setAttribute("crossOrigin", "anonymous");

image.onload = function() {

let canvas = document.createElement("canvas");

canvas.width = image.width;

canvas.height = image.height;

let context = canvas.getContext("2d");

context.drawImage(image, 0, 0, image.width, image.height);

let url = canvas.toDataURL("image/png"); //得到图片的base64编码数据

let a = document.createElement("a"); // 生成一个a元素

let event = new MouseEvent("click"); // 创建一个单击事件

a.download = name || "photo"; // 设置图片名称

a.href = url; // 将生成的URL设置为a.href属性

a.dispatchEvent(event); // 触发a的单击事件

};

image.src = imgsrc;

},

},

};

</script>

<style lang="css" scoped>

.container {

height: 90px;

text-align: center;

}

.dialog-bottom {

text-align: right;

padding-top: 80px;

padding-right: 20px;

}

</style>

### SinglePhoto.vue

<template>

<div>

<el-card :body-style="{ padding: '0px' }">

<el-image

lazy

style="width: 100%;"

:src="src"

class="image"

:preview-src-list="list"

>

<!-- 图片加载出错的情况 -->

<template #error>

<div class="image-slot">

<el-image

lazy

style="width: 100%;"

:src="require('../../../../../assets/img/loading.gif')"

class="image"

/>

<div style="text-align:center;">

<h3 style="text-align:center">

图片加载中，请稍后刷新

</h3>

<el-button @click="getImageList">强制刷新</el-button>

</div>

</div>

</template>

</el-image>

<div style="padding: 14px">

<span>{{ name }}</span>

<div class="bottom">

<el-button

type="text"

class="button"

@click="downloadIamge(src, singleImage.photo\_realname + type)"

>下载图片</el-button

>

<el-button

type="text"

class="button"

@click="showEdit"

>编辑图片</el-button

>

</div>

</div>

</el-card>

<el-dialog v-model="editVisible" :fullscreen="true" style="text-align:center" >

<my-edit :singleImage="singleImage" :index="index"/>

</el-dialog>

</div>

</template>

<script>

import MyEdit from "../Edit.vue"

export default {

props: ["singleImage", "index"],

components: {

MyEdit

},

data() {

return {

myProps: this.props,

list: [],

name: "",

type: "",

src: "",

editVisible:false,

};

},

created() {

this.initialize();

},

methods: {

showEdit(){

this.editVisible=true

},

initialize() {

if (this.index == "origin") {

this.list = [

this.singleImage.photo\_origin,

this.singleImage.photo\_upload,

this.singleImage.photo\_promap,

];

this.name = "原始图片";

this.type = "\_origin";

this.src = this.singleImage.photo\_origin;

} else if (this.index == "upload") {

this.list = [

this.singleImage.photo\_upload,

this.singleImage.photo\_promap,

this.singleImage.photo\_origin,

];

this.name = "黄金标准";

this.type = "\_upload";

this.src = this.singleImage.photo\_upload;

} else {

this.list = [

this.singleImage.photo\_promap,

this.singleImage.photo\_origin,

this.singleImage.photo\_upload,

];

this.name = "生成图片";

this.type = "\_promap";

this.src = this.singleImage.photo\_promap;

}

},

getImageList() {

this.$emit("getImageList");

},

downloadIamge(imgsrc, name) {

//下载图片地址和图片名

let image = new Image();

// 解决跨域 Canvas 污染问题

image.setAttribute("crossOrigin", "anonymous");

image.onload = function() {

let canvas = document.createElement("canvas");

canvas.width = image.width;

canvas.height = image.height;

let context = canvas.getContext("2d");

context.drawImage(image, 0, 0, image.width, image.height);

let url = canvas.toDataURL("image/png"); //得到图片的base64编码数据

let a = document.createElement("a"); // 生成一个a元素

let event = new MouseEvent("click"); // 创建一个单击事件

a.download = name || "photo"; // 设置图片名称

a.href = url; // 将生成的URL设置为a.href属性

a.dispatchEvent(event); // 触发a的单击事件

};

image.src = imgsrc;

},

},

};

</script>

### PhotoCard.vue

<template>

<el-card shadow="hover">

<!-- 卡片头栏 -->

<template #header>

<div class="card-header">

<span>

<span>{{ singleImage.photo\_realname }}</span>

<!-- 修改名字 -->

<el-popover placement="right" width="400" trigger="click">

<template #reference>

<el-button

style="margin-right:10px"

icon="el-icon-edit"

type="text"

circle

@click="myProps.newName = singleImage.photo\_realname"

></el-button>

</template>

<el-input

v-model="myProps.newName"

@keyup.enter="revisePictureName(singleImage)"

placeholder="请输入修改后的图片名称"

clearable

>

<template #append>

<el-button

@click="revisePictureName(singleImage)"

icon="el-icon-check"

></el-button>

</template>

</el-input>

</el-popover>

</span>

<span>

<!-- 下载图片 -->

<el-button

@click="downloadASetOfImage(singleImage)"

style="margin-right:10px"

icon="el-icon-download"

>下载全部</el-button

>

<!-- 删除图片 -->

<el-button

@click="deleteAGroupOfImage(singleImage)"

type="danger"

icon="el-icon-delete"

>删除全部</el-button

>

</span>

</div>

</template>

<!-- 三张图片 -->

<el-row>

<el-col :span="8">

<single-photo :singleImage="singleImage" index="origin" />

</el-col>

<el-col :span="8">

<single-photo

v-if="singleImage.photo\_upload && singleImage.photo\_upload.length > 0"

:singleImage="singleImage"

index="upload"

/>

<div v-else>

<el-upload

class="avatar-uploader"

:action="myProps.baseURL+'upload/'"

:data="{ photo\_id: singleImage.photo\_id }"

:show-file-list="false"

:on-success="handleAvatarSuccess"

:before-upload="checkFile"

:on-error="handleUploadError"

list-type="false"

name="pic\_img"

>

<el-card :body-style="{ padding: '0px' }">

<el-image

lazy

style="width: 100%;"

src="https://img0.baidu.com/it/u=101714381,3578597708&fm=253&fmt=auto&app=138&f=JPEG?w=667&h=500"

class="image"

>

</el-image>

<span>点击以上传</span>

<div style="padding: 14px">

<span>黄金标准</span>

</div>

</el-card>

</el-upload>

</div>

</el-col>

<el-col :span="8">

<single-photo :singleImage="singleImage" index="promap" />

</el-col>

</el-row>

</el-card>

</template>

<script>

import { ElMessageBox, ElMessage } from "element-plus";

import SinglePhoto from "./SinglePhoto.vue";

export default {

props: ["props", "singleImage"],

components: {

SinglePhoto,

},

data() {

return {

myProps: this.props,

};

},

methods: {

handleUploadError(){

this.myProps.loadingNewPicture = false;

this.$message.error("上传失败");

},

checkFile(file) {

this.myProps.loadingNewPicture = true;

var fileExtension = file.name.substring(file.name.lastIndexOf(".") + 1);

if (file.type.startsWith("image")) {

if (fileExtension === "gif") {

this.$message.error("不支持上传gif图片");

this.myProps.loadingNewPicture = false;

return false;

}

} else {

this.$message.error("请上传图片");

this.myProps.loadingNewPicture = false;

return false;

}

},

handleAvatarSuccess() {

this.$message({

message: "上传成功",

type: "success",

});

this.myProps.loadingNewPicture = true;

setTimeout(() => {

this.myProps.imageList = [];

this.getImageList();

}, 2000);

},

getImageList() {

this.$emit("getImageList");

},

downloadASetOfImage(singleImage) {

this.downloadIamge(

singleImage.photo\_origin,

singleImage.photo\_realname + "\_origin"

);

this.downloadIamge(

singleImage.photo\_upload,

singleImage.photo\_realname + "\_upload"

);

this.downloadIamge(

singleImage.photo\_promap,

singleImage.photo\_realname + "\_promap"

);

},

downloadIamge(imgsrc, name) {

//下载图片地址和图片名

let image = new Image();

// 解决跨域 Canvas 污染问题

image.setAttribute("crossOrigin", "anonymous");

image.onload = function() {

let canvas = document.createElement("canvas");

canvas.width = image.width;

canvas.height = image.height;

let context = canvas.getContext("2d");

context.drawImage(image, 0, 0, image.width, image.height);

let url = canvas.toDataURL("image/png"); //得到图片的base64编码数据

let a = document.createElement("a"); // 生成一个a元素

let event = new MouseEvent("click"); // 创建一个单击事件

a.download = name || "photo"; // 设置图片名称

a.href = url; // 将生成的URL设置为a.href属性

a.dispatchEvent(event); // 触发a的单击事件

};

image.src = imgsrc;

},

async deleteASetOfImage(singleImage) {

await new Promise((resolve) => {

this.$http

.post(

"/deletePicture/",

JSON.stringify({ photoID: singleImage.photo\_id })

)

.then((res) => {

{

if (res.data.message === "删除成功") {

this.getImageList();

} else {

this.$message.error("删除失败");

}

}

});

resolve();

}).then(() => {});

},

deleteAGroupOfImage(singleImage) {

ElMessageBox.confirm(

"本操作无法撤回，您确认要清空这三张图片吗？",

"Warning",

{

confirmButtonText: "OK",

cancelButtonText: "Cancel",

type: "warning",

}

)

.then(() => {

this.deleteASetOfImage(singleImage);

ElMessage({

type: "success",

message: "删除成功",

});

})

.catch(() => {

ElMessage({

type: "info",

message: "删除被取消",

});

});

},

async revisePictureName(singleImage) {

if (this.props.newName.trim().length == 0)

return this.$message.error("图片命名不能为空");

await new Promise((resolve) => {

this.$http

.post(

"/revisePictureName/",

JSON.stringify({

photoID: singleImage.photo\_id,

newName: this.props.newName,

})

)

.then((res) => {

{

if (res.data.message === "修改成功") {

singleImage.photo\_realname = this.props.newName;

this.$message.success("图片名修改成功");

} else {

this.$message.error("修改失败");

}

}

});

resolve();

}).then(() => {});

},

},

};

</script>

### Head.vue

<template>

<div class="card-header">

<div>

<!-- 上传新图片的按钮 -->

<el-tooltip

class="item"

effect="dark"

content="上传新的眼底血管图片"

placement="top"

>

<el-upload

class="upload-demo inline-block"

:action="props.baseURL + 'receive/'"

:data="{ user\_id: $store.state.user\_id }"

:on-success="handleAvatarSuccess"

list-type="false"

:show-file-list="false"

:before-upload="checkFile"

:on-error="handleUploadError"

name="pic\_img"

>

<el-button

show-file-list="false"

type="primary"

icon="el-icon-upload"

>上传</el-button

>

</el-upload>

</el-tooltip>

<!-- 下载全部图片的按钮 -->

<el-tooltip

class="item"

effect="dark"

content="下载本页所有图片"

placement="top"

>

<el-button

type="success"

@click="downloadAllImage()"

icon="el-icon-download"

>下载</el-button

>

</el-tooltip>

<!-- 清空全部图片的按钮 -->

<el-tooltip

class="item"

effect="dark"

content="清空本页所有图片"

placement="top"

>

<el-button

style="margin-left:30px"

@click="deleteAllImage()"

type="danger"

icon="el-icon-delete"

>清空</el-button

>

</el-tooltip>

</div>

<div>

<!-- 搜索框 -->

<el-input

v-model="myProps.searchTitle"

@keyup.enter="getImageList"

placeholder="请输入图片名"

clearable

style="width:20vw"

>

<template #append>

<el-button @click="getImageList" icon="el-icon-search"></el-button>

</template>

</el-input>

</div>

</div>

</template>

<script>

export default {

props: ["props"],

data() {

return {

myProps: this.props,

};

},

methods: {

getImageList() {

this.$emit("getImageList");

},

handleUploadError(){

this.myProps.loadingNewPicture = false;

this.$message.error("上传失败");

},

checkFile(file) {

this.myProps.loadingNewPicture = true;

var fileExtension = file.name.substring(file.name.lastIndexOf(".") + 1);

if (file.type.startsWith("image")) {

if (fileExtension === "gif") {

this.$message.error("不支持上传gif图片");

this.myProps.loadingNewPicture = false;

return false;

}

} else {

this.$message.error("请上传图片");

this.myProps.loadingNewPicture = false;

return false;

}

},

handleAvatarSuccess() {

this.$message({

message: "上传成功",

type: "success",

});

this.myProps.loadingNewPicture = true;

setTimeout(() => {

this.myProps.imageList = [];

this.getImageList();

}, 10000);

},

downloadAllImage() {

this.$emit("downloadAllImage");

},

deleteAllImage() {

this.$emit("deleteAllImage");

},

},

};

</script>

### Login/index.vue

<template>

<div class="login-wrap">

<div class="ms-login">

<div class="ms-title">眼底血管分割云平台</div>

<el-form

:model="param"

:rules="rules"

ref="login"

label-width="0px"

class="ms-content"

>

<el-form-item prop="username">

<el-input v-model="param.username" placeholder="username">

<template #prepend>

<i class="el-icon-user">用户名</i>

</template>

</el-input>

</el-form-item>

<el-form-item prop="password">

<el-input

type="password"

placeholder="password"

v-model="param.password"

@keyup.enter="submitForm()"

>

<template #prepend>

<i class="el-icon-lock">密码</i>

</template>

</el-input>

</el-form-item>

<div class="login-btn">

<el-button type="primary" @click="submitForm()">登录</el-button>

</div>

<h>当前还未注册？</h>

<h class="login-tips"

><router-link to="/register">前往注册</router-link></h

>

</el-form>

</div>

</div>

</template>

<script>

export default {

data() {

return {

param: {

username: "",

password: "",

},

rules: {

username: [

{ required: true, message: "请输入用户名", trigger: "blur" },

],

password: [

{ required: true, message: "请输入密码", trigger: "blur" },

],

},

};

},

created() {

this.$store.commit("clearTags");

},

methods: {

submitForm() {

this.$refs.login.validate(async (valid) => {

if (valid) {

await new Promise((resolve) => {

this.$http

.post("/login/", JSON.stringify(this.param))

.then((res) => {

if (res.data.success === false)

return this.$message.error(res.data.message);

this.$message.success("登录成功");

this.$store.commit("login", this.param.username);

this.$store.commit("storeId", res.data.userID);

console.log(res.data.userID);

this.$router.push({ path: "/doctor/upload" });

});

resolve();

}).then(() => {

console.log("成功啦post啦");

});

} else {

return;

}

});

},

},

};

</script>

<style scoped>

@import "style.css";

</style>

### Login/style.css

.login-wrap {

position: relative;

width: 100%;

height: 100%;

background-image: url(../../../assets/img/login.jpg);

}

.ms-title {

width: 100%;

line-height: 50px;

text-align: center;

font-size: 20px;

color: rgb(141, 139, 151);

border-bottom: 1px solid rgb(87, 107, 163);

}

.ms-login {

position: absolute;

left: 50%;

top: 50%;

width: 350px;

margin: -190px 0 0 -175px;

border-radius: 5px;

background: rgba(255, 255, 255);

overflow: hidden;

}

.ms-content {

padding: 30px 30px;

color: rgb(29, 29, 31);

}

.login-btn {

text-align: center;

}

.login-btn button {

width: 100%;

height: 36px;

margin-bottom: 10px;

}

.login-tips {

line-height: 30px;

color: rgb(29, 29, 31);

}

### Register/index.vue

<template>

<div class="register-wrap">

<el-scrollbar style="height: 100%">

<div class="ms-register">

<div class="ms-title">眼底血管分割云平台</div>

<el-form

:model="param"

:rules="rules"

ref="register"

class="demo-form-inline"

>

<el-form-item prop="username" label="用户名">

<el-input v-model="param.username" placeholder="请输入用户名">

</el-input>

</el-form-item>

<el-form-item prop="password" label="密码">

<el-input

type="password"

placeholder="请输入密码"

v-model="param.password"

@keyup.enter="submitForm()"

>

</el-input>

</el-form-item>

<el-form-item prop="password" label="确认密码">

<el-input

type="password"

placeholder="请确认密码"

v-model="param.tempPassword"

@keyup.enter="submitForm()"

>

</el-input>

</el-form-item>

<el-form-item prop="realname" label="真实姓名">

<el-input v-model="param.realname" placeholder="请输入真实姓名（选填）">

</el-input>

</el-form-item>

<el-form-item prop="email" label="邮箱">

<el-input v-model="param.email" placeholder="请输入邮箱（选填）">

</el-input>

</el-form-item>

<el-form-item prop="phone" label="手机号">

<el-input v-model="param.phone" placeholder="请输入手机号（选填）">

</el-input>

</el-form-item>

<el-form-item label="性别" prop="sex">

<el-radio-group v-model="param.sex" size="medium">

<el-radio border label="1">男</el-radio>

<el-radio border label="0">女</el-radio>

</el-radio-group>

</el-form-item>

<div class="register-btn">

<el-button type="primary" @click="submitForm()">注册</el-button>

</div>

<h>已有帐号？</h>

<h class="h">

<router-link to="/login">立即登录</router-link>

</h>

</el-form>

</div>

</el-scrollbar>

</div>

</template>

<script>

export default {

data() {

var checkEmail = (rule, value, callback) => {

const regEmail =

/(^([a-zA-Z0-9\_-])+@([a-zA-Z0-9\_-])+((.[a-zA-Z0-9\_-]{2,3}){1,2})+$)?/;

if (regEmail.test(value)) {

return callback();

}

callback(new Error("请输入合法的邮箱"));

};

var checkUsername = (rule, value, callback) => {

const regUsername = /(^[0-9]{11}$)\*/;

if (regUsername.test(value)) {

return callback();

}

callback(new Error("请输入合法的用户名"));

};

var checkPhone = (rule, value, callback) => {

const regPhone = /(^[0-9]+$)\*/;

if (regPhone.test(value)) {

return callback();

}

callback(new Error("请输入合法的手机号"));

};

return {

param: {

username: "",

password: "",

email: "",

phone: "",

sex: "",

},

rules: {

username: [

{ required: true, message: "请输入用户名", trigger: "blur" },

{

min: 1,

max: 20,

validator: checkUsername,

message: "用户名长度在1-20之间",

trigger: "blur",

},

],

password: [

{ required: true, message: "请输入密码", trigger: "blur" },

{

min: 6,

max: 15,

message: "密码长度在6到15个字符",

trigger: "blur",

},

],

email: [

{

validator: checkEmail,

trigger: "blur",

},

],

phone: [

{

validator: checkPhone,

trigger: "blur",

},

],

sex: [{ requierd: false, message: "请选择性别", trigger: "blur" }],

},

};

},

created() {

this.$store.commit("clearTags");

},

methods: {

submitForm() {

this.$refs.register.validate(async (valid) => {

if (valid) {

if (this.param.password !== this.param.tempPassword) {

return this.$message.error("输入的两次密码不一致");

}else {

await new Promise((resolve) => {

this.$http

.post("/register/", JSON.stringify(this.param))

.then((res) => {

console.log(res);

if (res.data.success === false)

return this.$message.error(res.data.message);

this.$message.success("注册成功");

this.$router.push({ path: "/login" });

});

resolve();

}).then(() => {

console.log("成功啦post啦");

});

}

} else {

return;

}

});

},

},

};

</script>

<style scoped>

@import "style.css";

</style>

### Register/style.css

.register-wrap {

position: relative;

width: 100%;

height: 100%;

background-image: url(../../../assets/img/login.jpg);

}

.ms-title {

width: 100%;

line-height: 50px;

text-align: center;

font-size: 20px;

color: rgb(141, 139, 151);

border-bottom: 1px solid rgb(87, 107, 163);

}

.ms-register {

margin: 0 auto;

width: 50%;

height: 50%;

border-radius: 5px;

background: rgba(255, 255, 255);

padding: 30px;

}

.register-btn {

text-align: center;

}

.register-btn button {

width: 100%;

height: 36px;

margin-bottom: 10px;

}

.register-tips {

font-size: 12px;

line-height: 30px;

color: rgb(29, 29, 31);

}

.h {

color: rgb(129, 109, 83);

}