templates文件下

test3是主要标注界面，有一个if else逻辑，就是标注完成提示

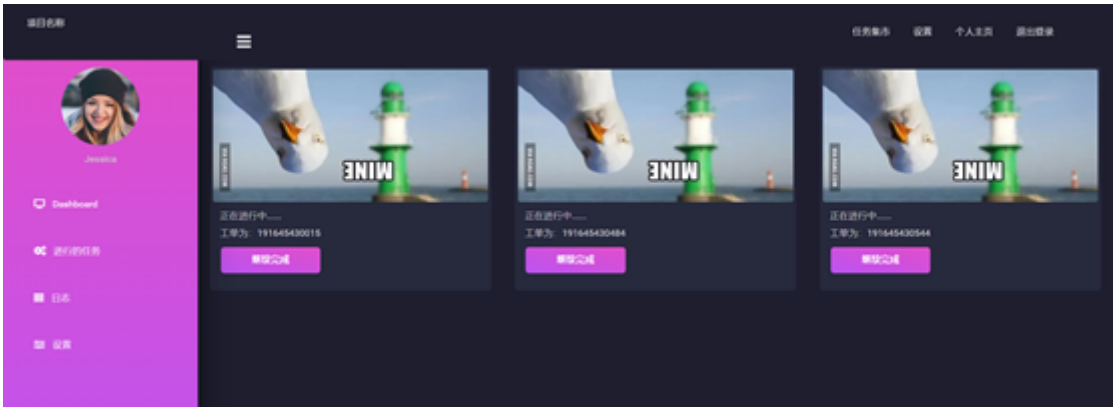
start是接单界面，也有一个if else :如果当前任务已经分配完成，按钮转为红色，用户无法继续领取。

login2 和 signup2是登录注册界面

choosepage2是模型选择界面

chart record setting userprofile 是个人主页的内容

userprofile显示还没有完成的任务



主要的flask页面是together4:

这个真的是屎山(╥╯^╰╥)太菜了 sooooorry

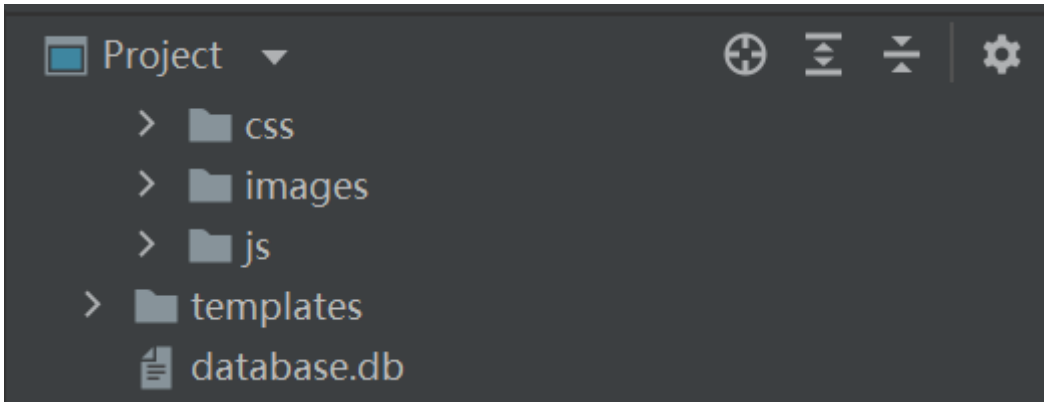
首先这一堆是关于log的，记录都在sample.log里面

```
logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)

formatter = logging.Formatter('%(asctime)s:%(levelname)s:%(message)s')

file_handler = logging.FileHandler('sample.log')
file_handler.setLevel(logging.INFO)
file_handler.setFormatter(formatter)

stream_handler = logging.StreamHandler()
stream_handler.setFormatter(formatter)

```

```
logger.addHandler(file_handler)
logger.addHandler(stream_handler)
```

数据库路径配置

```
app.config['SECRET_KEY'] = 'Thisissupposedtobesecret!'
app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///F:\\1\\dachuang\\merge_two_text_before\\database.db'
```



数据库初始化成功后将会在这个地址创建数据库；

数据库是sqlite3，可以去菜鸟教程看看怎样创建；

https://www.youtube.com/watch?v=8aTnmsDMldY

当时也看了这个视频创建数据库；

管理用户登录的；

```
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'
```

设计的表：

这个是用户表，用于记录注册的用户；

```
class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(15), unique=True)
    email = db.Column(db.String(50), unique=True)
    password = db.Column(db.String(80))
```

用户登录时使用数据库

```
@app.route('/', methods=['GET', 'POST'])
def login():
    form = LoginForm()

    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        print(user)
        if user:
            if check_password_hash(user.password, form.password.data):
```

```
                login_user(user)  # 有了这个之后才会变到choosepage 创建用户session
                return redirect(url_for('begin'))  # 这里写的要是函数名不是路径名

        return '<h1>Invalid username or password</h1>'
        # return '<h1>' + form.username.data + ' ' + form.password.data + '</h1>'

    return render_template('login2.html', form=form)
```

总的文件表，记录了所有上传的文件

```
class Files(db.Model):
    filenumber = db.Column(db.Integer, primary_key=True)
    filename = db.Column(db.String(50))
    userid = db.Column(db.Integer)
    picture_url = db.Column(db.String(100), unique=True)  # 先试试string类型能不能
```

userid是上传文件者的用户名

filenumber	filename	userid	picture_url
1	abandoned_ship_	9	/static/images/te
2	aerial_ladder_truc	9	/static/images/te
3	alley_cat_s_00001	9	/static/images/te
4	alley_cat_s_0000:	9	/static/images/te

用于图表显示:

```
class Chart(db.Model):
    datanumber = db.Column(db.Integer, primary_key=True)
    userID = db.Column(db.Integer)
    count = db.Column(db.Integer)
    loss = db.Column(db.Float)
    single = db.Column(db.Float)
    double = db.Column(db.Float)
    three = db.Column(db.Float)
```

userID便于查找每个用户的数据，count记录训练标注图片数量

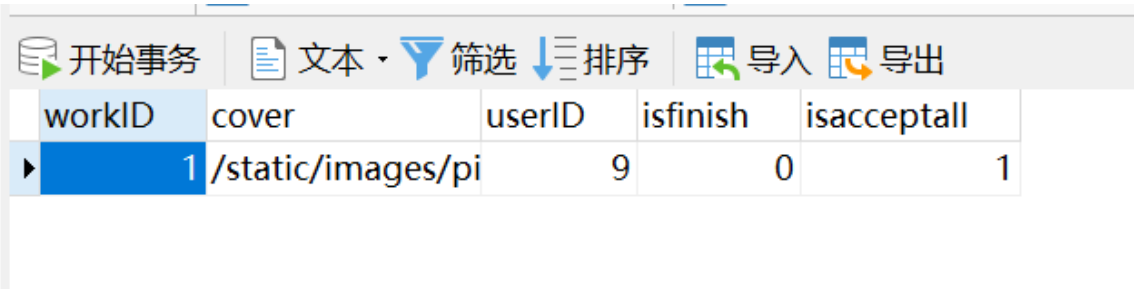
对象 | files @main (1) - 表 | chart @main (1) - 表

开始事务 | 文本 | 筛选 | 排序 | 导入 | 导出

datanumber	userID	count	loss	single	double	three
1	9	0	95752	1490116	0.2	0.3
2	9	8	43188	1490116	0.2	0.3
3	9	16	29272	1490116	0.2	0.3
4	9	24	33936	1490116	0.2	0.3
5	9	32	17664	1490116	0.2	0.3
6	9	40	75403	1490116	0.2	0.3

工单表，商家上传的每个任务对应一个工单，cover是这个任务的封面，workfile表记录了所有商家上传的文件，可以算是file的副本，它与work是多对一的关系，一个工单可以对应多个工作文件，故workID作为workfile的外键。iffinish=1是表明这项工作已经全部完成（完整逻辑暂未实现），isacceptall=1表明改工作已经被全部用户领取（完整逻辑已经实现）。此时任务选择界面显示


```
class Work(db.Model):
    workID = db.Column(db.Integer, primary_key=True)
    cover = db.Column(db.String(100))
    userID = db.Column(db.Integer)
    isfinish = db.Column(db.Integer)
    isacceptall = db.Column(db.Integer)
    workfile = db.relationship('Workfile', backref=db.backref('work'))
```



workfile表含有两个外键，work表的主键work.ID与stateofwork表的主键workIDofuser，workIDofuser是用户工单号，用户每次领取一个任务就会被分发一个工单号。

```
class Workfile(db.Model):
    filenumber = db.Column(db.Integer, primary_key=True)
    filename = db.Column(db.String(50))
    picture_url = db.Column(db.String(100), unique=True)
    workID = db.Column(db.Integer, db.ForeignKey('work.workID'))
    workIDofuser = db.Column(db.Integer,
db.ForeignKey('stateofwork.workIDofuser'))
```

workIDofuser在没有被分配前的值为NULL

filenumber	filename	picture_url	workID	workIDofuser
1	abandoned_ship_	/static/images/te	1	191645430015
2	aerial_ladder_tru	/static/images/te	1	191645430015
3	alley_cat_s_00001	/static/images/te	1	191645430015
4	alley_cat_s_0000	/static/images/te	1	191645430015
5	american_saddle_	/static/images/te	1	191645430015
6	appaloosa_s_000	/static/images/te	1	191645430015
7	blenheim_spaniel	/static/images/te	1	191645430015
8	frog2.png	/static/images/te	1	191645430015
9	frog7.png	/static/images/te	1	191645430015
10	deer0.png	/static/images/te	1	191645430015
11	deer1.png	/static/images/te	1	191645430484
12	deer2.png	/static/images/te	1	191645430484
13	automobile0.png	/static/images/te	1	191645430484
14	automobile1.png	/static/images/te	1	191645430484

记录了所有用户领取的任务图片

```
class Fileofuser(db.Model):
    ID = db.Column(db.Integer, primary_key=True)
    workID = db.Column(db.Integer)
    picture_url = db.Column(db.String(100), unique=True)
    filename = db.Column(db.String(50))
    userID = db.Column(db.Integer)
    workIDofuser = db.Column(db.Integer)
```

ID	workID	picture_url	filename	userID	workIDofuser
1	1	/static/images/te	abandoned_ship_	9	191645430015
2	1	/static/images/te	aerial_ladder_tru	9	191645430015
3	1	/static/images/te	alley_cat_s_00001	9	191645430015
4	1	/static/images/te	alley_cat_s_0000	9	191645430015
5	1	/static/images/te	american_saddle_	9	191645430015

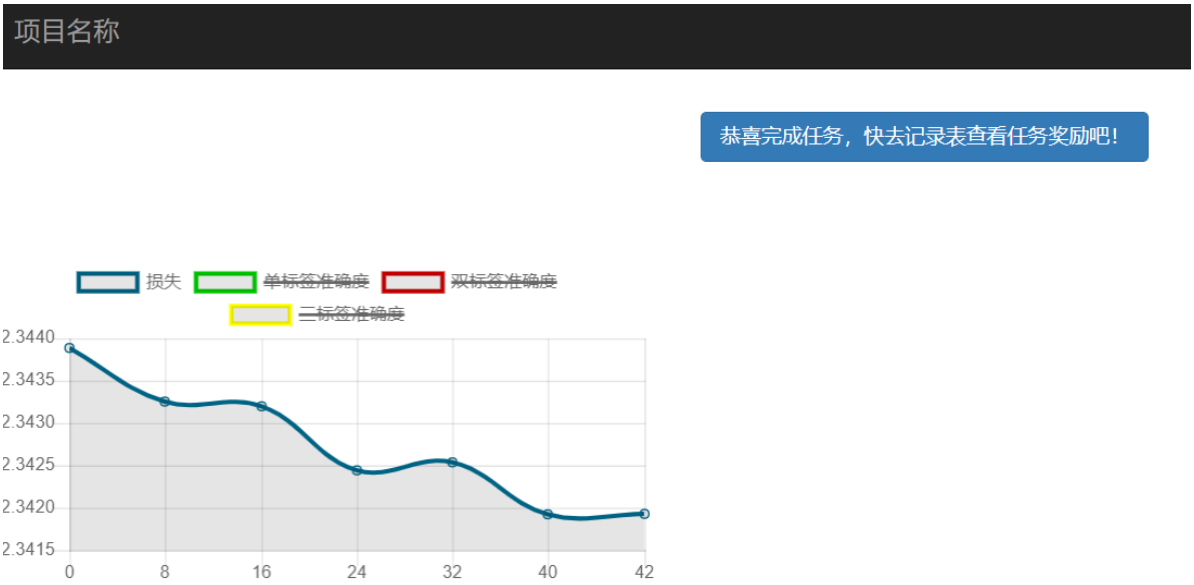
记录了用户任务状态，与workfile是一对多的关系

```
class Stateofwork(db.Model):
    workIDofuser = db.Column(db.Integer, primary_key=True)
    userID = db.Column(db.Integer)
    state = db.Column(db.Integer)
    cover = db.Column(db.String(100))

    workID = db.Column(db.Integer)
    workfileofuser = db.relationship('Workfile',
backref=db.backref('stateofwork'))
```

开始事务 文本 筛选 排序 导入 导出

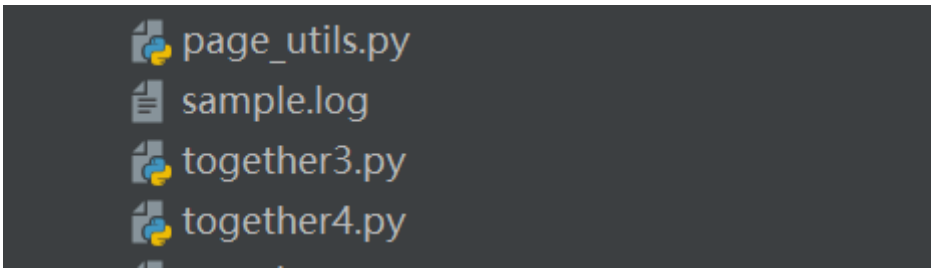
workIDofuser	userID	state	cover	workID
191645430015	9	0	/static/images/pi	1
191645430484	9	0	/static/images/pi	1
191645430544	9	0	/static/images/pi	1

完成任务使state=1,显示提醒

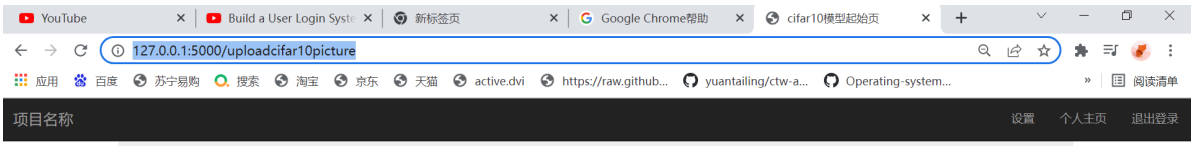


注意！！！：

由于没有实现商家界面，所以文件上传部分，主要是上传到file表里(由原来的版本实现)：



点击运行together3,登录后直接在浏览器上转到uploadcifar10界面，不要进入choosepage点击，**因为这个页面被我改成dogshit了。**



在这个界面可以选择上传的文件，我上传了一张，可以在file表中看见

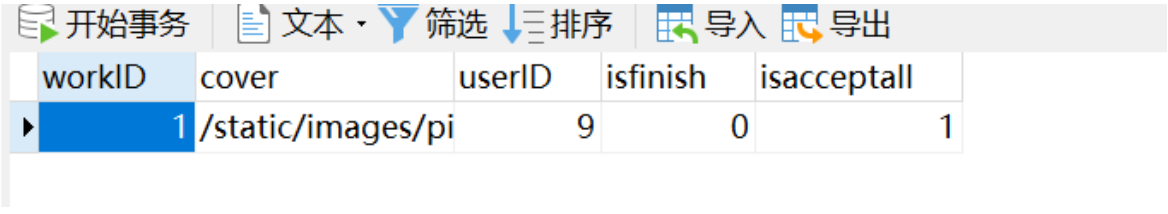
30	bird5.png	9	/static/images/te
31	appaloosa_s_002.	9	/static/images/te

然后再拷贝到workfile表中就可以。

主要函数：

这个是任务选择界面的函数

```
@app.route('/start', methods=['GET', 'POST'])
@login_required
def begin():
    workset = Work.query.all()

    worksetfromdb = []
    for b in workset:
        data = [b.workID, b.cover, b.isacceptall]
        worksetfromdb.append(data)
    print(worksetfromdb)  # ["xx", "xx", "xx", "xx"]


    return render_template("start.html", workset=worksetfromdb)
```



打开work表，然后把信息传到任务选择界面。

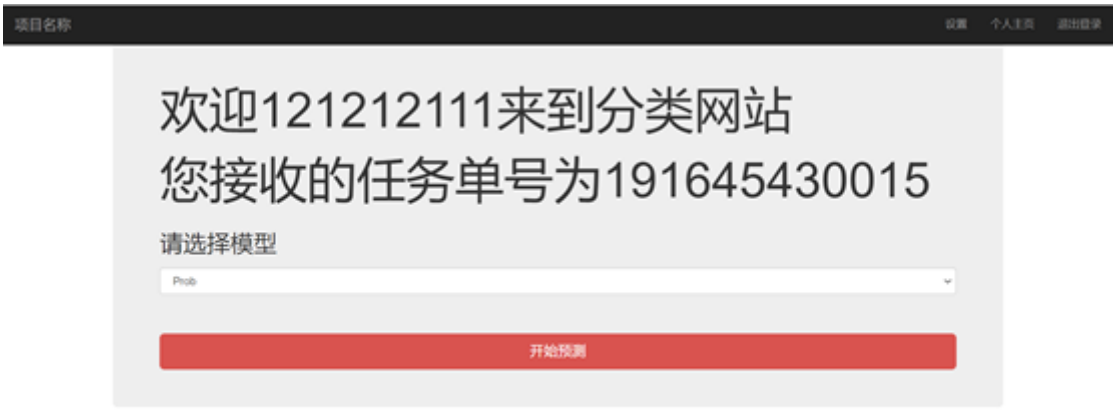
这个函数是模型选择界面的函数：

假设商家设置每个用户每次只能领取10个文件。

```
# 选择哪个模型部分
@app.route('/choosepage/<workid>', methods=['GET', 'POST'])
@login_required
def choosepage(workid):
```

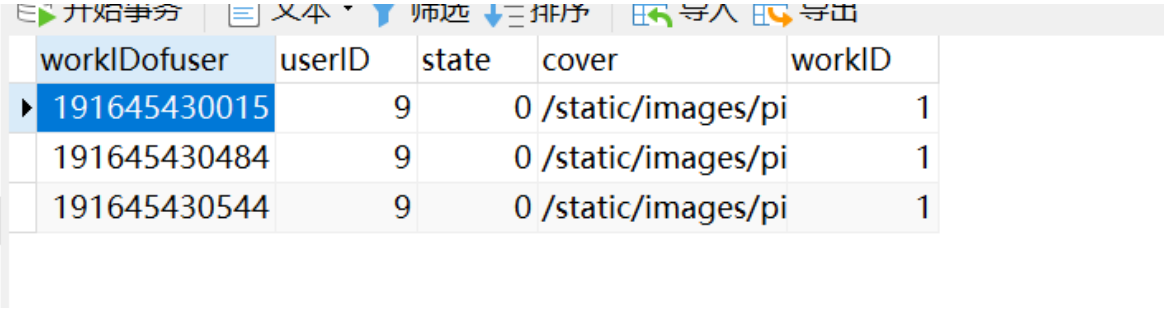
```
    #假设商家设置的每个uwork的文件数量为10

    filenumofuwork = 10

    workfiles = Workfile.query.filter_by(workID=workid).all()
    work = Work.query.filter_by(workID=workid).first()
    cover = work.cover
    print(cover)

    ts = calendar.timegm(time.gmtime())
    i = 0 #记录所有已经被领取过的文件数量
    j = 0 #记录当前领取的文件数量

    for b in workfiles:
        if b.workIDofuser:
            i=i+1
            continue#跳过已经领取的文件
        else:
            j=j+1
            i=i+1
            url=b.picture_url
            filename=b.filename
            workidofuser = int (str(workid)+str(current_user.id)+str(ts)) #同一个用户接同一个商家的任务会报错 #改了，加了时间戳
            print(workidofuser)
            b.workIDofuser=workidofuser
            new_file = Fileofuser(userID=current_user.id, picture_url=url, filename=filename,workID=workid,workIDofuser=workidofuser)#把领取的图片加入Filepfuser文件表
            db.session.add(new_file)
            db.session.commit() #这个好像只用提交一次？
            if j==10:
                break
    if i==len(workfiles):#改任务已经被全部领取
        work.isacceptall=1

    new_work = Stateofwork(userID=current_user.id, state=0, cover=cover, workIDofuser=workidofuser, workID=workid)#记录用户领取的工作状态
    db.session.add(new_work)
    db.session.commit()


    return render_template("choosepage2.html", name2=workidofuser, name1=current_user.username)
```

在模型选择页面，点击开始预测后进入此函数：



根据用户工单加载所有属于该工单的图片，然后把图片的路径名，文件名传到前端标注页面（用于以后的训练），还要把当前任务的状态，与工单传到前端。

stateofwork表中state=0表示当前任务还未完成。



workIDofuser	userID	state	cover	workID
191645430015	9	0	/static/images/pi	1
191645430484	9	0	/static/images/pi	1
191645430544	9	0	/static/images/pi	1

```python
@app.route("/begintoexersize/<workidofuser>", methods=['GET', 'POST'])  # post隐式提交，get显示提交
def predict(workidofuser):


    print(workidofuser)
    dataset1 = Fileofuser.query.filter_by(workIDofuser=workidofuser).all()
    print(dataset1)
    datasetfromdb = []
    for b in dataset1:
        data = [b.picture_url, b.filename]
        datasetfromdb.append(data)
    print(datasetfromdb)  # ["xx", "xx", "xx", "xx"]
    print(type(datasetfromdb))  # <class 'list'>

    pager_obj = Pagination(request.args.get("page", 1), len(datasetfromdb), request.path, request.args,
                           per_page_count=8)#一个页面显示8张图
    print(request.path) #/begintoexersize
    print(request.args)

    indexlist = datasetfromdb[pager_obj.start:pager_obj.end]

    html = pager_obj.page_html()

    widu = workidofuser

    dataset2 = Stateofwork.query.filter_by(workIDofuser=workidofuser).first()
    state = dataset2.state
```

```
28
29      return render_template('test3.html', index_list=indexlist, html=html,
name=widu, name2=state)
```

用户上传标签函数:

```
1  #这里有bug只能按着顺序标
```

```
1  #用户上传标签部分
2
3  @app.route('/labelcifar10/<workidofuser>', methods=['GET', 'POST'])
4  @login_required
5  def labelcifar10(workidofuser):
6      dataset1 = Fileofuser.query.filter_by(workIDofuser=workidofuser).all()
7      print(dataset1)
8      datasetfromdb = []
9      for b in dataset1:
10         data = [b.picture_url, b.filename, b.ID]
11         datasetfromdb.append(data)
12     print(datasetfromdb)  # ["xx", "xx", "xx", "xx"]
13     print(type(datasetfromdb))  # <class 'list'>
14
15     pager_obj = Pagination(request.args.get("page", 1), len(datasetfromdb),
request.path, request.args,
16                            per_page_count=8)
17     print(request.path)
18     print(request.args)
19
20     indexlist = datasetfromdb[pager_obj.start:pager_obj.end]
21
22     labels = request.form.getlist('label')#获取用户上传的标签 这是一个list
23     #如果没有标注，跳过训练环节
24     if labels:
25
26         print(labels)
27         print(indexlist)
28         i = 0
29
30         res =
Chart.query.filter_by(userID=current_user.id).order_by(Chart.count.desc()).
first()
31
32         if res == None:
33             count = 0
34
35         else:
36             count = res.count #如果chart表中的count数据存在，则继续上传的记录计数
37
38         print("count:{}".format(count))
39
40         for data in indexlist:#载入标记的图片
41
42             upload_path = os.path.join(basedir,
secure_filename(data[1]))#data = [b.picture_url, b.filename, b.ID]
43
44             img_to_save = cv2.imread(upload_path)
```

```
45
46              save_path = os.path.join(traindir, labels[i]) #把需要训练的图加入
文件夹
47
48              photoname = data[1]
49
50              if os.path.isdir(save_path):
51                  cv2.imwrite(os.path.join(save_path, photoname),
img_to_save)
52              else:
53                  os.makedirs(save_path)
54                  cv2.imwrite(os.path.join(save_path, photoname),
img_to_save)
55
56
57              #从用户文件表中删除已经标注的文件
58              datasetfromdb.remove(data)
59
60              delete_data = Fileofuser.query.filter_by(ID=data[2]).one()
61              db.session.delete(delete_data)
62              db.session.commit()
63
64              logger.info('user:{} labled {} as {}
'.format(current_user.username, data[1], labels[i]))
65              i = i + 1
66              print('i的值为：{}'.format(i))
67
68              count = count + 1
69              print('count:{}'.format(count))
70              print('lengthlen:{}'.format(len(indexlist)))
71              #8张8张地训练模型
72              if i == len(indexlist):
73                  countlist.append(count)
74                  print(countlist)
75
76                  x_train = trainmodel.newx_train(traindir)
77                  y_train = trainmodel.newy_train(traindir)
78
79                  trainmodel.re_train(x_train, y_train, modelcifar10,
modelpath)
80
81                  #shutil.move(save_path2, olddir)  # 这里有小bug olddir不能事
先存在这里应该用个for循环，循环遍历一个文件夹
82                  move_file(traindir,olddir)#把文件移到已经训练过的文件表中
83
84      model2 = load_model("VGG16.h5")
85
86      eval = trainmodel.evaluate(model2, x_test, y_test, y_foracc)  # 返回的顺
序是：损失函数有多大，单标签准确度，双标签准确度，三标签准确度
87
88      new_chartdata = Chart(userID=current_user.id, count=count,
loss=float(eval[0]), single=float(eval[1]),
89                            double=float(eval[2]), three=float(eval[3]))
90      db.session.add(new_chartdata)
91      db.session.commit()
92      #更新chart表
93      pager_obj = Pagination(request.args.get("page", 1), len(datasetfromdb),
request.path, request.args,
```

```
                                per_page_count=8)
    print(request.path)
    print(request.args)
    #更新返回前端的图片，这里先没有管机器预测环节，如果要加预测，直接用if-else跳过精确度高的图片即可
    indexlist = datasetfromdb[pager_obj.start:pager_obj.end]

    html = pager_obj.page_html()
    temp = 0
    #如果没有图片需要返回前端，则把stateofwork表中的状态置为1，将状态返回前端，前端显示“任务完成”
    if datasetfromdb==[]:
        stateset = Stateofwork.query.filter_by(workIDofuser=workidofuser).first()
        stateset.state = 1
        temp = stateset.state
        db.session.commit()
    print(temp)
    return render_template('test3.html', index_list=indexlist, html=html, eval=eval, name=workidofuser, name2=temp)
```

这里用到了ajax技术实时显示图表

```
@app.route('/data', methods=["GET", "POST"])
def data():
    model2 = load_model("VGG16.h5")



    countlist = []
    losslist = []
    singlelist = []
    doublelist = []
    threelist = []


    data = Chart.query.filter_by(userID=current_user.id).first()
    if data:
        results = Chart.query.filter_by(userID=current_user.id).all()

        for one in results:
            countlist.append(one.count)
            losslist.append(one.loss)
            singlelist.append(one.single)
            doublelist.append(one.double)
            threelist.append(one.three)
    else:
        count = 0
        eval = trainmodel.evaluate(model2, x_test, y_test, y_foracc)  # 返回的顺序是：损失函数有多大，单标签准确度，双标签准确度，三标签准确度
        new_chartdata = Chart(userID=current_user.id, count=count, loss=float(eval[0]), single=float(eval[1]),
                              double=float(eval[2]), three=float(eval[3]))
        db.session.add(new_chartdata)
        db.session.commit()

        countlist = [0]
```

```
        losslist = [float(eval[0])]
        singlelist = [float(eval[1])]
        doublelist = [float(eval[2])]
        threelist = [float(eval[3])]

    return jsonify({'count': countlist, 'accuracy': losslist,
'single_lable': singlelist, 'double_lable': doublelist,
                    'three_lable': threelist})
```

个人主页部分：

这里用到了stateofwork表，加载还没有完成的任务

```
@app.route('/profile', methods=['GET', 'POST'])
@login_required
def profile():
    stateofworks = Stateofwork.query.filter_by(userID=current_user.id).all()
    dataset=[]
    for w in stateofworks:
        if w.state==0:
            data = [w.cover, w.workIDofuser]
            dataset.append(data)

    return render_template("userprofile2.html", data=dataset)
```

剩下的比较简单，主要就是套静态模板

```
@app.route('/chart', methods=['GET', 'POST'])
@login_required
def chart():
    return render_template("chart.html")

@app.route('/setting', methods=['GET', 'POST'])
@login_required
def setting():
    return render_template("setting.html")

@app.route('/record', methods=['GET', 'POST'])
@login_required
def record():
    return render_template("record.html")
```

登入注册登出：

```
@app.route('/', methods=['GET', 'POST'])
def login():
    form = LoginForm()

    if form.validate_on_submit():
        user = User.query.filter_by(username=form.username.data).first()
        print(user)
        if user:
            if check_password_hash(user.password, form.password.data):
```

```
                login_user(user)  # 有了这个之后才会变到choosepage 创建用户session
                return redirect(url_for('begin'))  # 这里写的要是函数名

        return '<h1>Invalid username or password</h1>'
        # return '<h1>' + form.username.data + ' ' + form.password.data + '</h1>'

    return render_template('login2.html', form=form)


@app.route('/signup', methods=['GET', 'POST'])
def signup():
    form = RegisterForm()

    if form.validate_on_submit():
        hashed_password = generate_password_hash(form.password.data, method='sha256')
        new_user = User(username=form.username.data, email=form.email.data, password=hashed_password)
        db.session.add(new_user)
        db.session.commit()

        return render_template('warn.html')

    return render_template('signup2.html', form=form)


@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect('/')
```

比较重要的路径

```
basedir = os.path.abspath(r"F:\1\dachuang\merge_two_text_before\static\images\testpicture")  # 用户上传的图片都会先存进去
traindir = os.path.abspath(r"F:\1\dachuang\merge_two_text_before\static\images\newlable")  # 用来训练的数据
testdir = os.path.abspath(r"F:\1\dachuang\merge_two_text_before\static\images\test")  # 用来测试的数据
olddir = os.path.abspath(r"F:\1\dachuang\merge_two_text_before\static\images\old-labled")
```