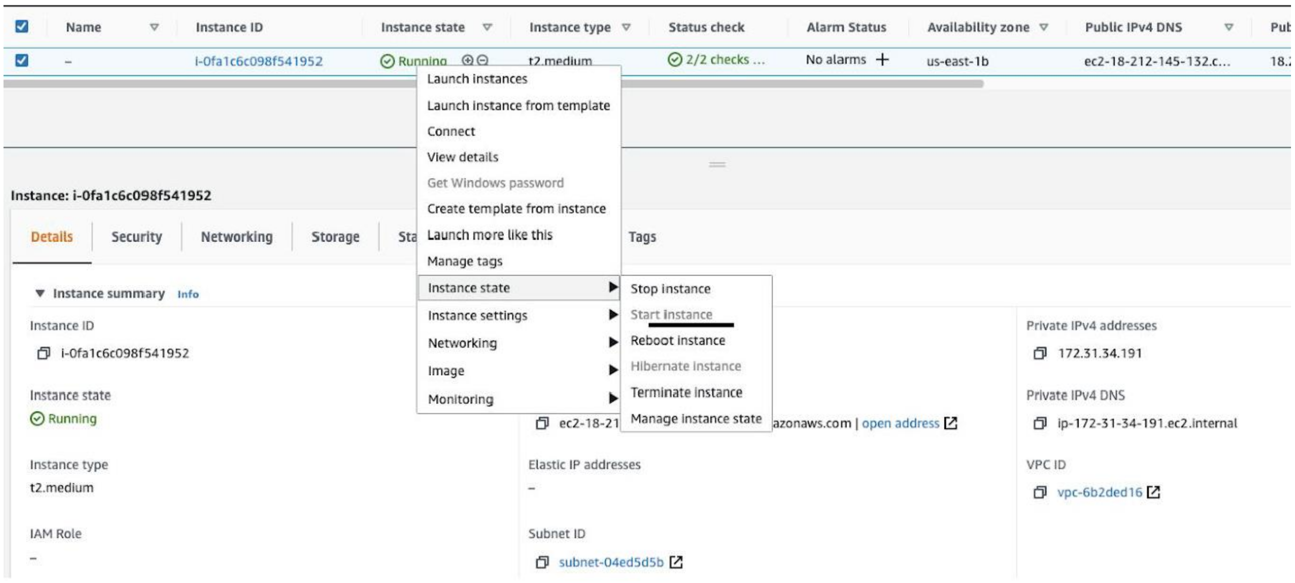
ECE1779 Assignment1

Group members:
Qin Deng 1006714799
Ziyan Zhao 1007212927

How to log into EC2 and start the program

1. Extract the keypair.pem

   rar x a1_submit.rar
2. Make sure the EC2 instance with ID: **i-05297cc1f0b1acde2** is running. The following figure shows how to start an instance.



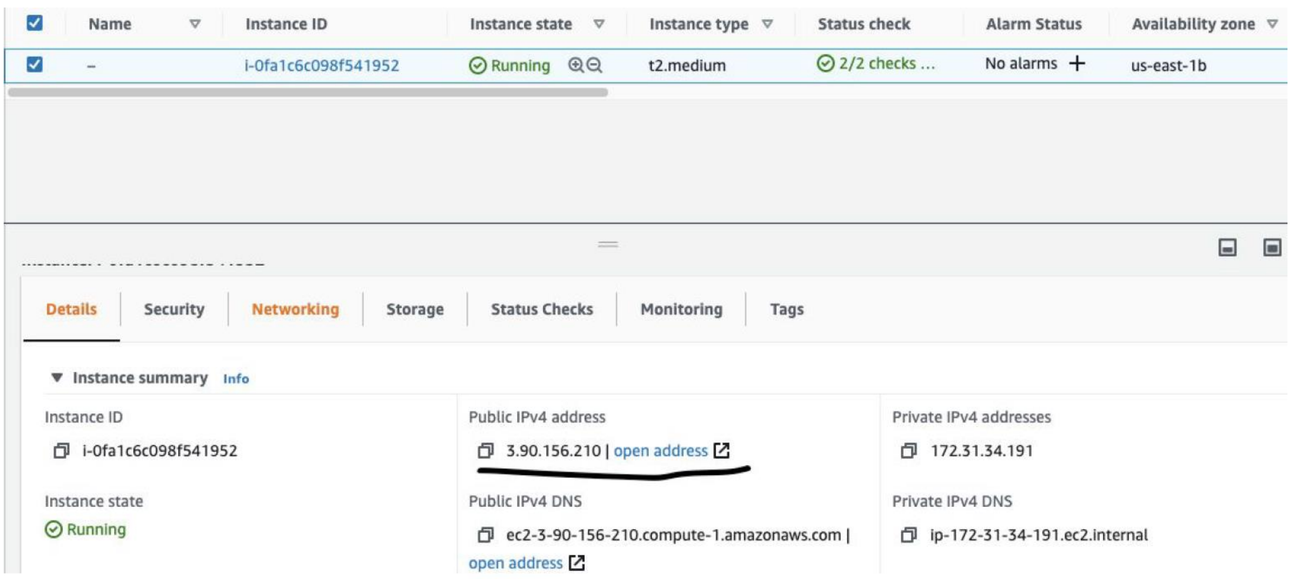
Name	Instance ID	Instance state	Instance type	Status check	Alarm Status	Availability zone	Public IPv4 DNS	Pub
–	i-0fa1c6c098f541952	Running	t2.medium	2/2 checks ...	No alarms +	us-east-1b	ec2-18-212-145-132.c...	18.

Instance: i-0fa1c6c098f541952

Instance ID: i-0fa1c6c098f541952
Instance state: Running
Instance type: t2.medium
IAM Role: –
ec2-18-21...azonaws.com | open address
Elastic IP addresses: –
Subnet ID: subnet-04ed5d5b
Private IPv4 addresses: 172.31.34.191
Private IPv4 DNS: ip-172-31-34-191.ec2.internal
VPC ID: vpc-6b2ded16

3. Put keypair.pem under your ~/.ssh directory. If "~/.ssh" doesn't exist, mkdir ~/.ssh
4. Once pem file is in your .ssh directory, check the public IPv4 address in your aws console

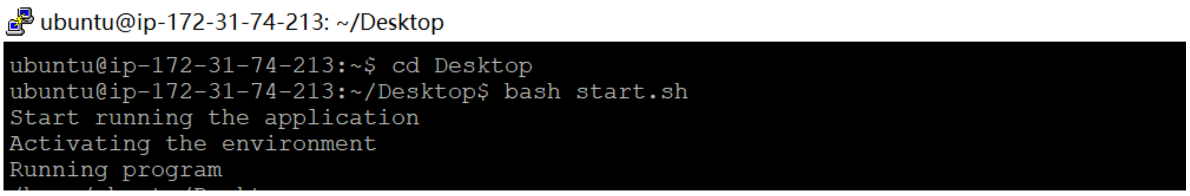
   (For example, ip address in the following figure is: 3.90.156.210)



	Name	Instance ID	Instance state	Instance type	Status check	Alarm Status	Availability zone
☑	–	i-0fa1c6c098f541952	Running	t2.medium	2/2 checks ...	No alarms	us-east-1b

Instance ID: i-0fa1c6c098f541952
Public IPv4 address: 3.90.156.210 | open address
Private IPv4 addresses: 172.31.34.191
Instance state: Running
Public IPv4 DNS: ec2-3-90-156-210.compute-1.amazonaws.com | open address
Private IPv4 DNS: ip-172-31-34-191.ec2.internal

5. Run the following command to ssh into the EC2 instance.
   ssh -i ~/.ssh/keypair.pem ubuntu@ec2_ip_address
6. Once you ssh into the EC2 instance. We need to cd into code repository directory:
   cd Desktop
7. Run the following command in your command line.
   bash start.sh
8. The website should be running like the following figure. The website is running on port 5000, so you can access the website by url: 3.90.1562.210(your public IPv4 address):5000



```
ubuntu@ip-172-31-74-213: ~/Desktop
ubuntu@ip-172-31-74-213:~$ cd Desktop
ubuntu@ip-172-31-74-213:~/Desktop$ bash start.sh
Start running the application
Activating the environment
Running program
```

App Instruction

1. Login panel

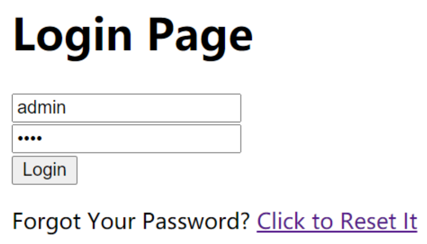
FaceMaskDetection

- Login

Click Login will bring you to the login page.

2. Login

Initially, there is only one administrator (**username:admin, password:test**) in database. Administrators are able to create and delete users later. New users can also be registered by API endpoint '/api/register'



Login Page

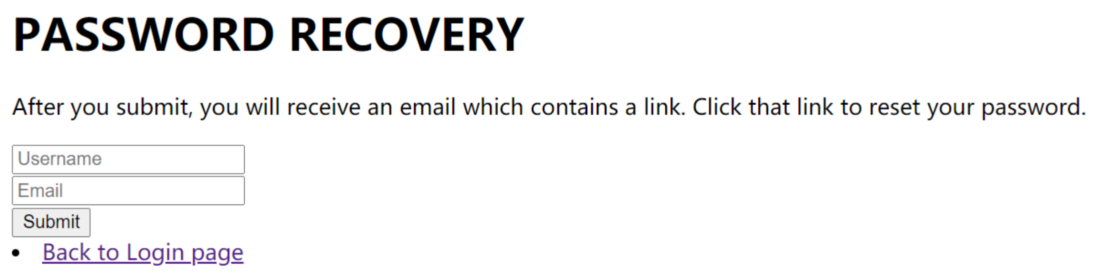
admin

••••

Forgot Your Password? Click to Reset It

If users forget their password, they can reset it using the email they provide in registration form.

3. Password Recovery

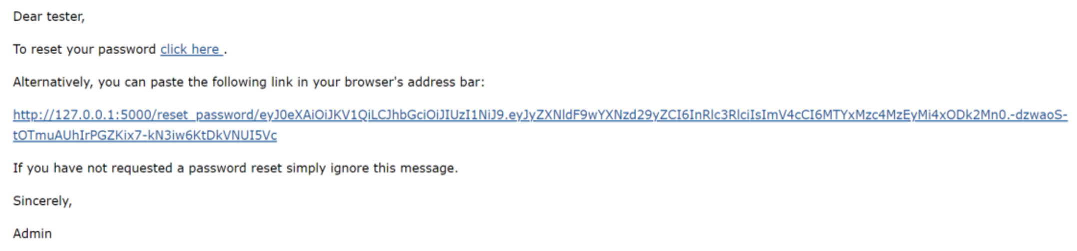


PASSWORD RECOVERY

After you submit, you will receive an email which contains a link. Click that link to reset your password.

- Back to Login page

By providing the username and email associated with it, user will get a link in their email to reset their password.



Dear tester,

To reset your password click here .

Alternatively, you can paste the following link in your browser's address bar:

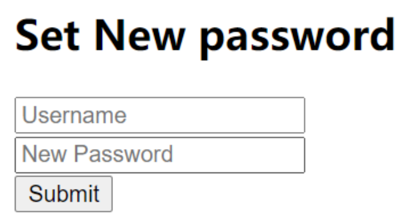
http://127.0.0.1:5000/reset_password/eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJyZXNldF9wYXNzd29yZCI6InRlc3RlciIsImV4cCI6MTYxMzc4MzEyMi4xODk2Mn0.-dzwaoS-tOTmuAUhIrPGZKix7-kN3iw6KtDkVNUI5Vc

If you have not requested a password reset simply ignore this message.

Sincerely,

Admin

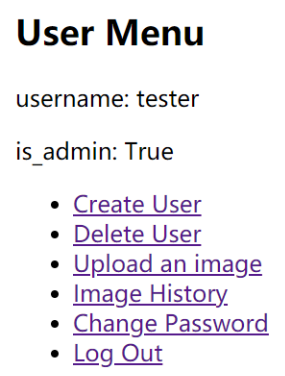
Users can reset their password by clicking this link.



Set New password

4. User management

Administrators are able to add and delete user, while regular users are unauthorized to do that.



User Menu

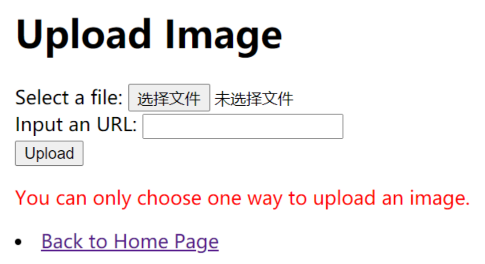
username: tester

is_admin: True

- Create User
- Delete User
- Upload an image
- Image History
- Change Password
- Log Out

Both users and administrators can change password， upload images and view history.

5. Mask detections



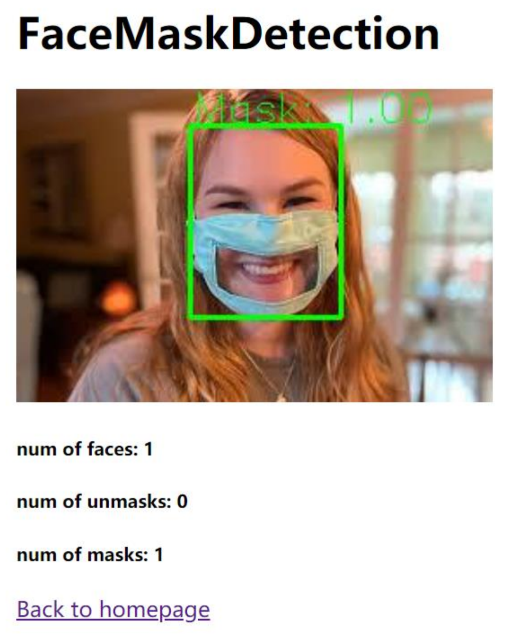
Upload Image

Select a file: 选择文件 未选择文件
Input an URL:
Upload

You can only choose one way to upload an image.

- Back to Home Page

The application allows user to upload an image from local file system or by URL, but every time only one picture will be processed.
The input url should end with an image file like:
https://ss1.bdstatic.com/70cFvXSh_Q1YnxGkpoWK1HF6hhy/it/u=1740406959,550349782&fm=26&gp=0.jpg
After uploading, submit bottom will bring users to the page displaying the processing result.



FaceMaskDetection

num of faces: 1

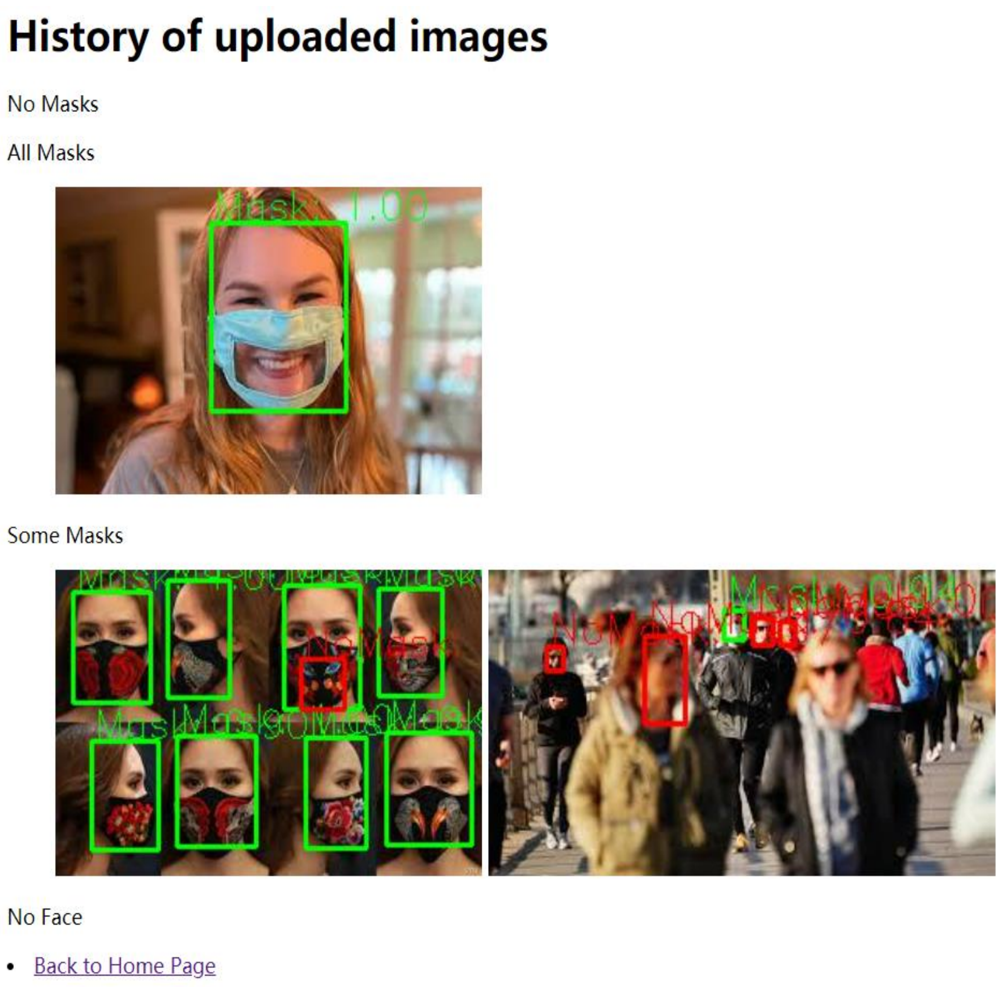
num of unmasks: 0

num of masks: 1

Back to homepage

6. Upload history

Select ‘Image History’, users will see their upload history. All the uploaded image will be split into four groups.



History of uploaded images

No Masks

All Masks

Some Masks

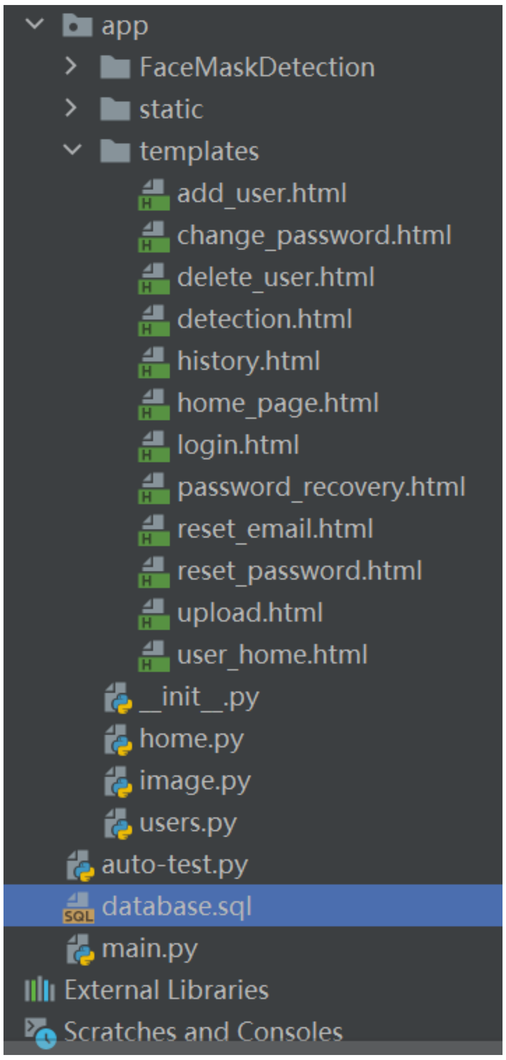
No Face

- Back to Home Page

7. logout

Finally, users can logout application by selecting logout link in user menu. Users can also close the browser and the session will be cleared.

Code Architecture



```
app
  FaceMaskDetection
  static
  templates
    add_user.html
    change_password.html
    delete_user.html
    detection.html
    history.html
    home_page.html
    login.html
    password_recovery.html
    reset_email.html
    reset_password.html
    upload.html
    user_home.html
  __init__.py
  home.py
  image.py
  users.py
auto-test.py
database.sql
main.py
External Libraries
Scratches and Consoles
```

Here is the structure of the code. ‘main.py’ start the application, ‘database.sql’ creates two tables and add an administration at beginning. ‘auto-test’ is primarily for testing API. ‘home.py’, ‘users.py’ and ‘image.py’ are the backend of the application, which connect to mysql and send the processing result to webpage.

/templates store the html templates

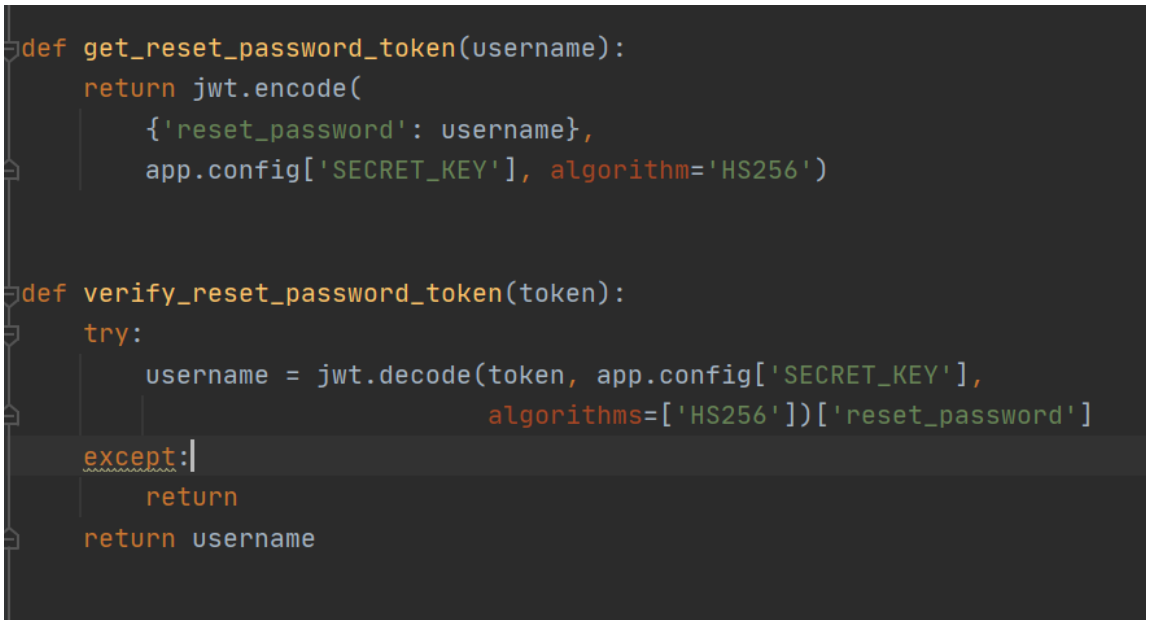
/static store the static image application needs and images uploaded by users.

/FaceMaskDetection By changing the application working directory, the application calls FaceMaskDetection.inference function to process the upload pictures.

Implementation Detail

1. Password Recovery

When a user is not logged in and click the Click to Reset it . The user need to input his/her username and email. Once the username and email are verified, the user will receive a email. We use flask_mail to send email to users. In the email, there is a link containing a token which is generated by function get_reset_password_token(). After the user click the link, user will go to a reset password page. On that page, the user need to input username and new password. After the username is verified by function verify_reset_password_token(). The password will be reset.

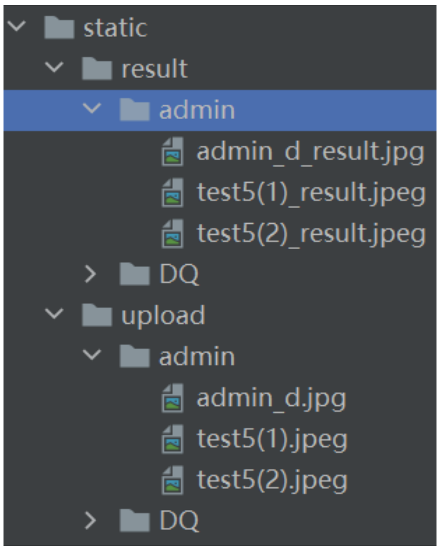


```
def get_reset_password_token(username):
    return jwt.encode(
        {'reset_password': username},
        app.config['SECRET_KEY'], algorithm='HS256')


def verify_reset_password_token(token):
    try:
        username = jwt.decode(token, app.config['SECRET_KEY'],
                              algorithms=['HS256'])['reset_password']
    except:
        return
    return username
```

2. Images

(1) Upload image



```
static
  result
    admin
      admin_d_result.jpg
      test5(1)_result.jpeg
      test5(2)_result.jpeg
    DQ
  upload
    admin
      admin_d.jpg
      test5(1).jpeg
      test5(2).jpeg
    DQ
```

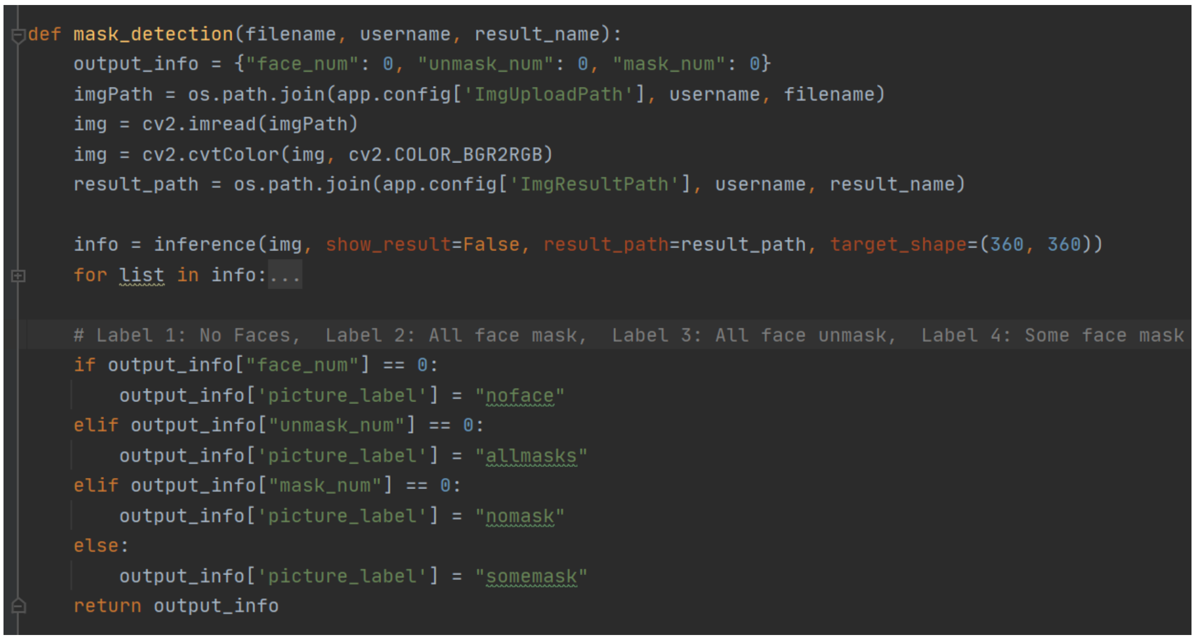
There are two ways to upload image: Select a file from local system and Input an url to make server download an image. We use the package **urllib** to download image from url. The download image will be saved with name: **username_result.XXX(image_type)**. If the image is uploaded by selecting from local system, the image will be saved with its original name.

For each user, we create directory static/upload/<username> to store upload image and static/result/<username> to store images with detection results.

Before we save images, we will find whether the image with same name exists. If an image with the same name exists, we will change the filename to **filename**(num)**.XXX**. The num is a variable will be incremented when there is an image with same name.

(2) FaceMaskDetection

The method mask_detection in image.py calls the FaceMaskDetection.pytorch_infer.inference. The inference method will save the processed image to result_path and will return a dictionary containing the information of detection results.



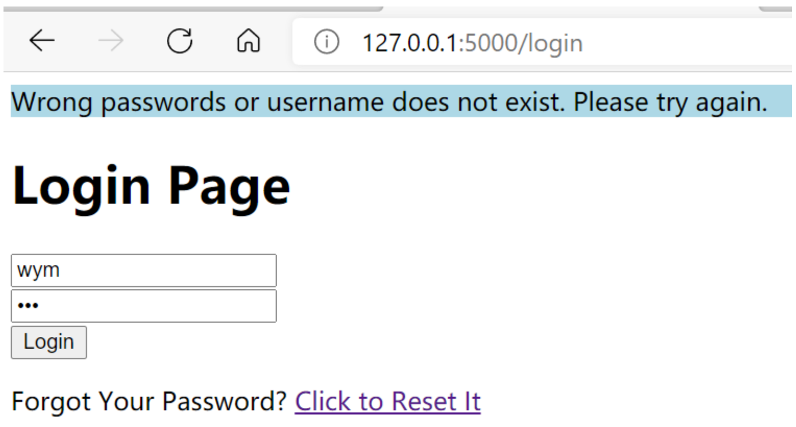
```python
def mask_detection(filename, username, result_name):
    output_info = {"face_num": 0, "unmask_num": 0, "mask_num": 0}
    imgPath = os.path.join(app.config['ImgUploadPath'], username, filename)
    img = cv2.imread(imgPath)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    result_path = os.path.join(app.config['ImgResultPath'], username, result_name)

    info = inference(img, show_result=False, result_path=result_path, target_shape=(360, 360))
    for list in info:...

    # Label 1: No Faces,  Label 2: All face mask,  Label 3: All face unmask,  Label 4: Some face mask
    if output_info["face_num"] == 0:
        output_info['picture_label'] = "noface"
    elif output_info["unmask_num"] == 0:
        output_info['picture_label'] = "allmasks"
    elif output_info["mask_num"] == 0:
        output_info['picture_label'] = "nomask"
    else:
        output_info['picture_label'] = "somemask"
    return output_info
```

3. Error/Success Message

The error/success message of user's operation will be showed by using flash() in flask package. These message will be shown at the top of the page.



127.0.0.1:5000/login

Wrong passwords or username does not exist. Please try again.

Login Page

wym

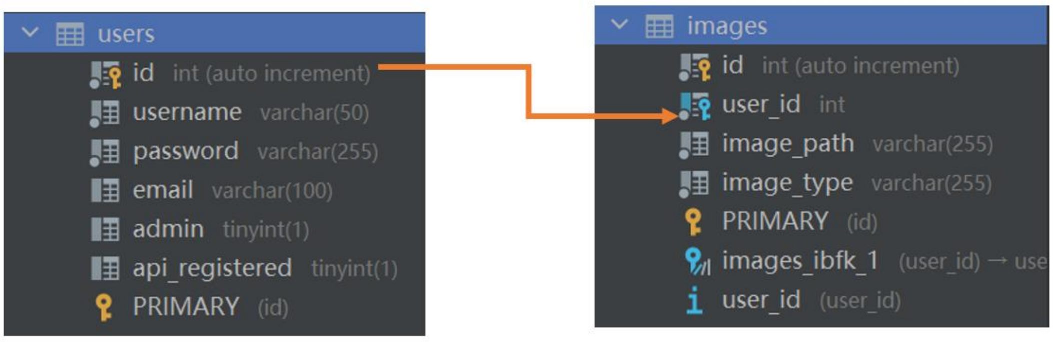
•••

Login

Forgot Your Password? Click to Reset It

Database Schema

The database sql code is in database.sql file.



users
id int (auto increment)
username varchar(50)
password varchar(255)
email varchar(100)
admin tinyint(1)
api_registered tinyint(1)
PRIMARY (id)

images
id int (auto increment)
user_id int
image_path varchar(255)
image_type varchar(255)
PRIMARY (id)
images_ibfk_1 (user_id) → use
user_id (user_id)

Users table contain 6 columns， ‘id‘, ’username‘, ’password‘, ’email‘, ’admin‘(if the user is administrator this column will be 1 else 0)、’api_registered’(if the user was created by api this will be 1 else 0)
Images table contain 4 columns, ‘id’, ‘user_id’(the foreign key to the user table), ‘image_path’(where the image store), ‘image_type’(no_faces, no_masks, some_masks, all_masks)