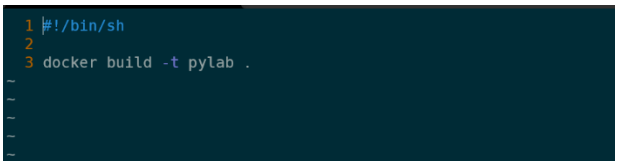
بنام خدا

گزارش پروژه آز پایتون

توضیح فایل های سمت server :



```
#!/bin/sh

docker build -t pylab .
```

فایل او dockerbuild.sh است که شامل کد یک خطی زیر است که یک image داکر را با تگ pylab میسازد و نحوه ساخت آنرا از آدرسی که در آن است میگیرد که در Dockerfile ذخیره شده است.

توضیح Dockerfile :

```
FROM ubuntu:latest

ENV DEBIAN_FRONTEND noninteractive

RUN apt-get update
RUN apt-get -y upgrade
RUN apt-get -y install python3 python3-pip python3-opencv
RUN pip install opencv-python numpy hug
RUN groupadd -r hug && useradd -r -g hug hug

ADD main.py /app/main.py
WORKDIR /home/hug
EXPOSE 8585
CMD hug -f /app/main.py -p 8585

```

خط اول به معنای این هست که این سرور مجازی که با داکر میخواهیم بسازیم آخرین ورژن اوبونتو هست و در خط سوم متغیر DEBIAN_FRONTEND ، مقدارش به noninteractive تغییر میکند تا دستور های خطوط بعدی که با استفاده از apt هستن بدون نیاز به گرفتن ورودی اجرا شوند.

در خط پنجم نمایه مخزن نرم افزار هارا به روز رسانی میکنیم و در خط بعدی مواردی که با توجه به خط قبل نیاز به آپدیت دارند را آپدیت میکنیم.

در خط هفتم پایتون را نصب میکنیم برای اجرای کد پایتونی که داریم و pip را برای نصب کتابخانه هایی که استفاده میکنیم و opencv برای بخش تشخیص چهره!

در خط هشتم هم سه کتابخانه مورد نیاز را نصب میکنیم.

در خط نهم کاربری با نام hug و گروه hug را میسازیم .

در خط یازدهم فایل پایتون main.py را از سرور فیزیکی به آدرس نوشته شده در سروری که با داکر میسازیم کپی میکنیم.

در خط دوازدهم دایرکتوری که در سرور میخواهیم در آن کار کنیم را مشخص میکنیم.

در خط سیزدهم پورت 8585 از سروری که با داکر میسازیم را باز میکنیم.

و نهایتا در خط آخر command که توسط این سرور اجرا می شود را تعیین میکنیم که سرویس api ، با استفاده از hug است که چگونگی آن در فایل main.py مشخص می شود و خروجی آن به پورت 8585 فرستاده می شود.

پس از ساخت image داکر با استفاده از فایل های بالا ، به سراغ فایل dockerrun.sh می رویم:

```
1 #!/bin/sh
2
3
4 for c in `docker ps -a --no-trunc | grep 'hug' | awk '{print $1}'`; do
5   docker kill $c
6   docker rm -f $c
7 done
8
9
10 int_port=8585
11 ext_port=5050
12 if [ ! -d "$1" ]
13 then
14         echo "This volume does not exist, buddy."
15         exit 1
16 fi
17 cd $1
18 d=`pwd`
19 cd -
20
21 docker run -dit -v $d:/home/hug  -p 127.0.0.1:$ext_port:$int_port pylab
```

در قسمت اول یعنی خطوط 4 تا 7 سایر سرویس هایی که از این میسر hug در حال استفاده هستند را kill میکنیم و فایل آن ها را پاک میکنیم.

سپس پورت های ورودی و خروجی را مشخص میکنیم و در ادامه اجرای این فایل dockerrun.sh آدرس دایرکتوری تصاویرمان را میدهیم که خط 12 تا 16 بررسی میکند که ورودی بعدی یک دایرکتوری باشد.

سپس در خط 17 به آن مسیر میرویم و آدرسش را در متغیر d ذخیره میکنیم.

نهایتا سروری را run میکنیم که آدرس فایل تصاویر که ذخیره کردیم را متناظر میکند با آدرس /home/hug در سرور که نیازی به کپی کردن فایل تصاویر نباشد و از روی سرور فیزیکی خوانده شوند همچنین در ورودی های بعدی تناظر پورت ها و image که سرور از روی آن ساخته می شود را مشخص کردیم که همان pylab قسمت قبل است.

حال به سراغ توضیح کد پایتون main.py سمت سرور میرویم :

```python
import hug, cv2
import glob, base64

def face_detector(picname):
    image = cv2.imread(picname)
    image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades+\
                                         'haarcascade_frontalface_default.xml')
    faces = face_cascade.detectMultiScale(image_gray)

    for x, y, width, height in faces:
        cv2.rectangle(image, (x, y), (x + width, y + height), color=(255, 0, 0), thickness=2)
    if len(faces) == 0:
        return (0,picname)
    else :
        cv2.imwrite(f"/tmp/{picname}", image)
        return (len(faces), f"/tmp/{picname}")

@hug.get('/')
def get_image(pic_name, func):
    if func == 'Serial':
        if pic_name in img_list:
            b64img = base64.b64encode(open(pic_name,'rb').read())
            return { 'status':'OK', 'content':b64img, 'msg':'Your picture is finded!' }
        else:
            return {'status':'NOK', 'content':'Sorry, but this image does not exist :(\n'}
    elif func == 'FaceDetection':
        if pic_name in img_list:
            n, picStr = face_detector(pic_name)
            b64img = base64.b64encode(open(picStr,'rb').read())
            return { 'status':'OK', 'content': b64img, 'msg':'There is a face!' if n else 'No face detected!' }
        else:
            return {'status':'NOK', 'content':'Sorry, but this image does not exist :(\n'}
    else:
        return {'status':'dorost nis!', 'content':'Not valid func'}


img_list = glob.glob('*.png')
img_list += glob.glob('*.jpg')

```

خط اول کتابخانه های hug و cv2 هستند که برای api و تشخیص چهره از آنها استفاده میکنیم.
در خط دوم کتابخانه های glob و base64 هستند که برای پیدا کردن پترن های فایل های عکس و تبدیل دیتا باینری به دیتا ASCII قابل نمایش و برعکس استفاده می شوند.

در ادامه دو تابع در کد هستند که تابع برای تشخیص چهره با استفاده از کتابخانه cv2 هست و خروجی تعداد چهره های detect شده و تصویر مورد نظر هست و تابع دوم که با استفاده از decorator به پاسخ فراخوانی hug با / بر میگردد با توجه به درخواست اگر تصویر موجود باشد خود تصویر یا تصویری که در آن چهره ها مشخص شده اند را بر میگرداند که در ادامه تست میکنیم.

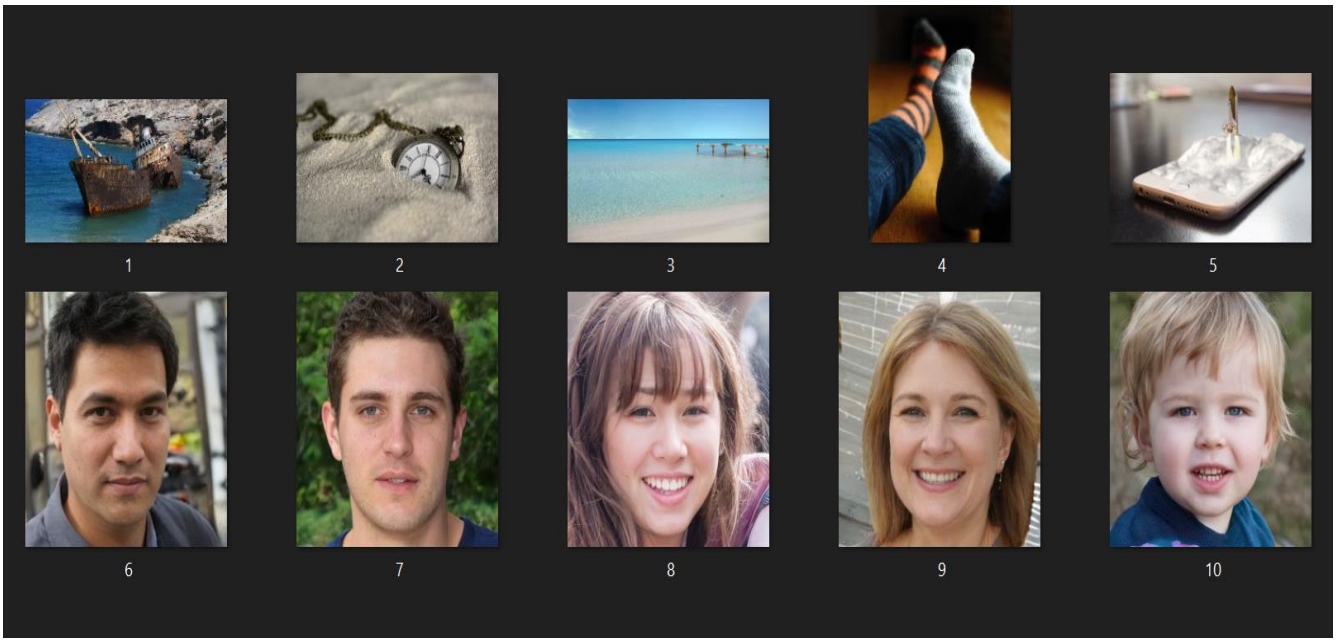
و نهایتا در سمت client کد پایتون زیر را داریم :

```python
import requests
import base64
print('For exit enter -1 :')
while True :
    pic_num = input('Enter your picture number:')
    if pic_num == '-1' :
        break
    mode = input('Enter your reques, Serial or FaceDetection?')
    data = {'pic_name':f'{pic_num}.jpg', 'func':mode}
    r = requests.get('http://localhost:5050/', data=data)
    res = r.json()
    if res['status'] == 'OK':
        f = open('res.png','wb')
        f.write(base64.b64decode(res['content']))
        print(res['msg'])
    else:
        print(res['content'])

```

کد بالا که از ورودی شماره تصویر مورد نظر و نوع درخواست که Serial و یا FaceDetection است را میگیرد و پاسخ را برمیگرداند و نهایتا با دریافت ورودی 1- پروسه به پایان میرسد.

در فولدر img ده تصویر زیر قرار دارند :



ابتدا dockerbuild.sh و سپس dockerrun.sh را با ورودی img که فولدر تصاویر است در ترمینال اجرا میکنیم و از مسیر server خارج می شویم :

```
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project cd server
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project/server ./dockerbuild
.sh
Sending build context to Docker daemon  2.759MB
Step 1/11 : FROM ubuntu:latest
 ---> ba6acccedd29
Step 2/11 : ENV DEBIAN_FRONTEND noninteractive
 ---> Using cache
 ---> 45998185e323
Step 3/11 : RUN apt-get update
 ---> Using cache
 ---> 24f3077fe834
Step 4/11 : RUN apt-get -y upgrade
 ---> Using cache
 ---> 18f253dcc183
Step 5/11 : RUN apt-get -y install python3 python3-pip python3-opencv
 ---> Using cache
 ---> 6b8e74b2e26b
Step 6/11 : RUN pip install opencv-python numpy hug
 ---> Using cache
 ---> e636bafa998e
Step 7/11 : RUN groupadd -r hug && useradd -r -g hug hug
 ---> Using cache
 ---> 2ac4a08fe0e2
Step 8/11 : ADD main.py /app/main.py
 ---> Using cache
 ---> e47003e93911
Step 9/11 : WORKDIR /home/hug
 ---> Using cache
```

```
Step 6/11 : RUN pip install opencv-python numpy hug
 ---> Using cache
 ---> e636bafa998e
Step 7/11 : RUN groupadd -r hug && useradd -r -g hug hug
 ---> Using cache
 ---> 2ac4a08fe0e2
Step 8/11 : ADD main.py /app/main.py
 ---> Using cache
 ---> e47003e93911
Step 9/11 : WORKDIR /home/hug
 ---> Using cache
 ---> 91538c508045
Step 10/11 : EXPOSE 8585
 ---> Running in 55e24092055d
Removing intermediate container 55e24092055d
 ---> 788b296adc11
Step 11/11 : CMD hug -f /app/main.py -p 8585
 ---> Running in 105c61603be2
Removing intermediate container 105c61603be2
 ---> 0ae20a2a93cc
Successfully built 0ae20a2a93cc
Successfully tagged pylab:latest
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project/server ./dockerrun.s
h img
f37d2e53f283fce22c01f42e62c3247b9f05a1aa5a9f20d16c5b4f178d7edf4e
f37d2e53f283fce22c01f42e62c3247b9f05a1aa5a9f20d16c5b4f178d7edf4e
/mnt/c/Users/mamal/Desktop/dars/az-python/project/server
31a6be810acb7b483cfee8aa73556f2f3129cc4066c3b3896ca8977e70338539
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project/server cd ..
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project
```

حال در سمت client کد client.py را اجرا میکنیم :

و درخواست تصویر اول را به صورت Serial وارد میکنیم :

```
 ---> 2ac4a08fe0e2
Step 8/11 : ADD main.py /app/main.py
 ---> Using cache
 ---> e47003e93911
Step 9/11 : WORKDIR /home/hug
 ---> Using cache
 ---> 91538c508045
Step 10/11 : EXPOSE 8585
 ---> Running in 55e24092055d
Removing intermediate container 55e24092055d
 ---> 788b296adc11
Step 11/11 : CMD hug -f /app/main.py -p 8585
 ---> Running in 105c61603be2
Removing intermediate container 105c61603be2
 ---> 0ae20a2a93cc
Successfully built 0ae20a2a93cc
Successfully tagged pylab:latest
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project/server ./dockerrun.s
h img
f37d2e53f283fce22c01f42e62c3247b9f05a1aa5a9f20d16c5b4f178d7edf4e
f37d2e53f283fce22c01f42e62c3247b9f05a1aa5a9f20d16c5b4f178d7edf4e
/mnt/c/Users/mamal/Desktop/dars/az-python/project/server
31a6be810acb7b483cfee8aa73556f2f3129cc4066c3b3896ca8977e70338539
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project/server cd ..
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project python client.py
For exit enter -1 :
Enter your picture number:1
Enter your reques, Serial or FaceDetection?Serial
Your picture is finded!
Enter your picture number:
```

پیام Your picture is finded! نمایش داده می شود و در کنار فایل client.py تصویر زیر با نام res به وجود می آید.

در ادامه درخواست تصویر سوم را به صورت FaceDetection میفرستیم :

```
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project python client.py
For exit enter -1 :
Enter your picture number:3
Enter your reques, Serial or FaceDetection?FaceDetection
No face detected!
Enter your picture number:
```

خروجی بالا نمایش داده می شود و تصویر زیر ذخیره می شود :

به همین صورت چند درخواست دیگر مطابق تصویر زیر ارسال میکنیم و تصاویر خروجی را به ترتیب می آوریم :

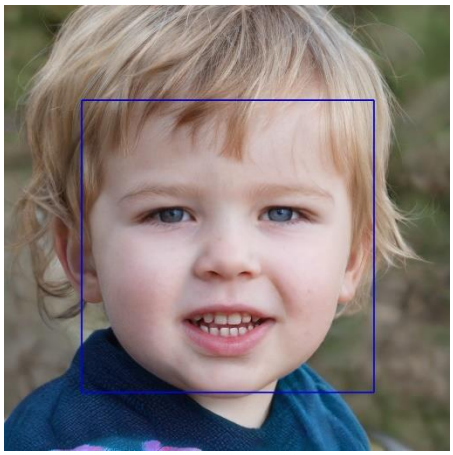
```
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project python client.py
For exit enter -1 :
Enter your picture number:8
Enter your reques, Serial or FaceDetection?Serial
Your picture is finded!
Enter your picture number:10
Enter your reques, Serial or FaceDetection?FaceDetection
There is a face!
Enter your picture number:5
Enter your reques, Serial or FaceDetection?FaceDetection
No face detected!
Enter your picture number:14
Enter your reques, Serial or FaceDetection?Serial
Sorry, but this image does not exist :(

Enter your picture number:15
Enter your reques, Serial or FaceDetection?FaceDetection
Sorry, but this image does not exist :(

Enter your picture number:6
Enter your reques, Serial or FaceDetection?FaceDetection
There is a face!
Enter your picture number:2
Enter your reques, Serial or FaceDetection?Serial
Your picture is finded!
Enter your picture number:-1
(anaconda3) srinivasazarei@LAPTOP-441JB163 /mnt/c/Users/mamal/Desktop/dars/az-python/project
```

پاسخ درخواست عکس 8 به صورت Serial :

پاسخ در خواست عکس 10 به صورت FaceDetection :



پاسخ در خواست عکس 5 به صورت FaceDetection :

که پیام No face detected! نمایش داده شد در ترمینال

درخواست تصاویر 14 و 15 به درستی با پیام ☹ Sorry, but this image does not exist همراه شدند زیرا در مسیر img تنها تصاویر یک تا ده وجود دارند.

و نهایتا درخواست تصویر 6 به صورت FaceDtection و تصویر 2 به صورت Serial :