

# **ASSIGNMENT-3**

**S.DIVYA THARSHINI**

**VV COLLEGE OF ENGINEERING**

**III-YEAR CSE**

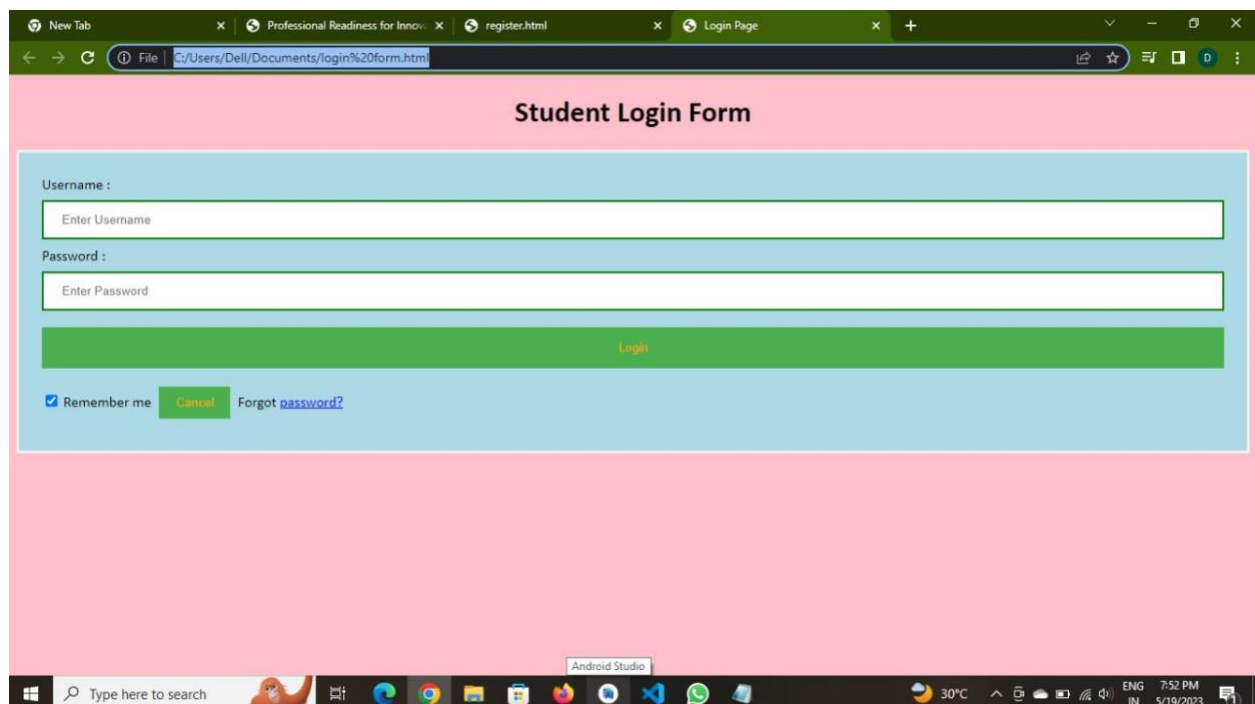
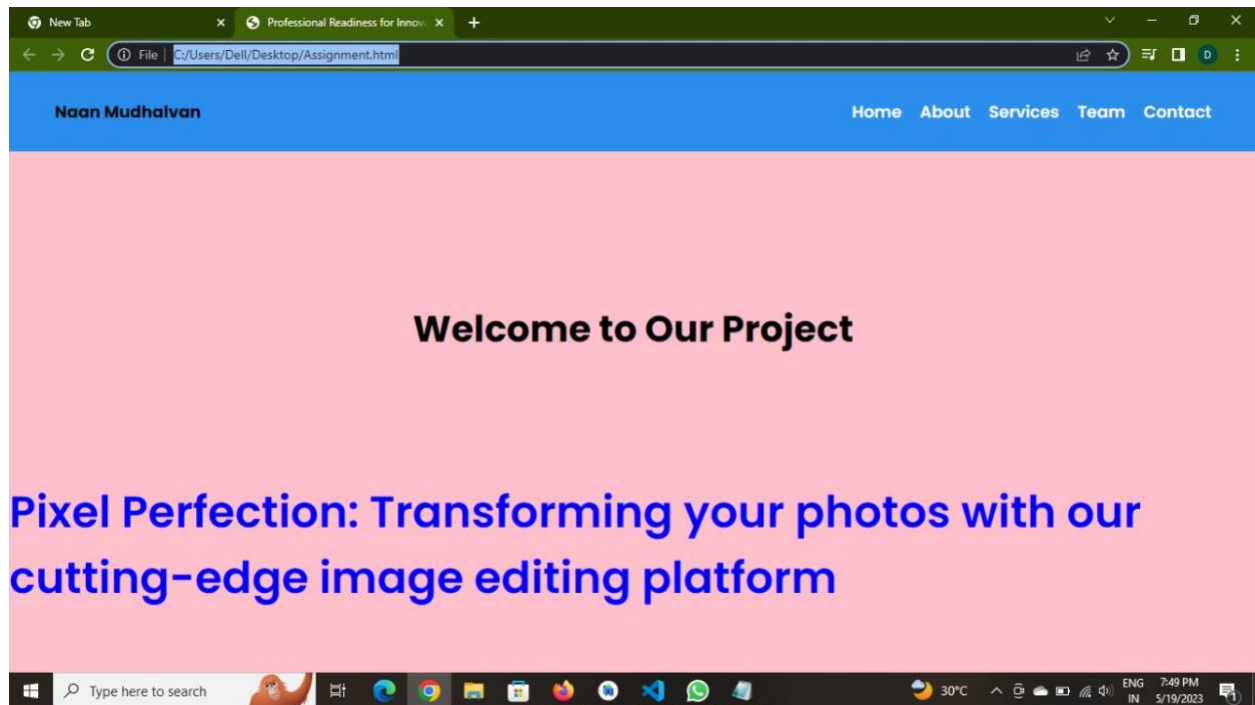
In the Assignment, you created 3 HTML files (index, profile and log-in), CSS files and stored the registration data in IBM db2 and did login validation.

In this assignment follow these tasks

**Task 1:** You have to containerize the whole application by using the docker

**Task 2:** Upload this docker image to the IBM container registry

**Task 3:** Then orchestrate the docker container using Kubernetes.



New Tab x Professional Readiness for Innov... x register.html x Login Page x +

File | C:/Users/Dell/Documents/register.html

## Register

Please fill in this form to create an account.

**Email**

Enter Email

**Password**

Enter Password

**Repeat Password**

Repeat Password

By creating an account you agree to our [Terms & Privacy](#).

Register

Already have an account? [Sign In](#).

Type here to search

30°C

ENG IN 7:52 PM 5/19/2023

Inbox (53) - divya.r140007@gmail... x HTML Code for Student Profile P... x Profile Page x +

File | C:/Users/Dell/Desktop/profile.html

## Profile

29 50

About me  
Student

Profile

Settings

### Personal Details

Name	: DIVYA.R
Gender	: Female
Age	: 21
Institution	: VV College Of Engineering
Department	: CSE
Email ID	: divya@gmail.com
Phone Number	: 9876543210
Skill	: C,HTML,PYTHON
Hobbies	: Drawing,Playing
State	: Tamilnadu
Language Known	: English,Tamil

### SOCIAL MEDIA

Facebook Twitter Instagram WhatsApp Messenger

Type here to search

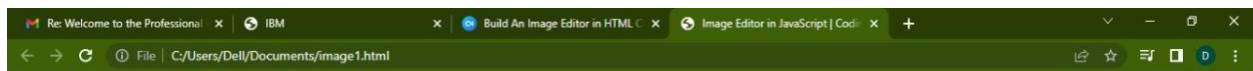
36°C

ENG IN 1:25 PM 5/21/2023



Click on the "Choose image" button to upload a image:

No file chosen



## Easy Image Editor

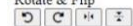
Filters

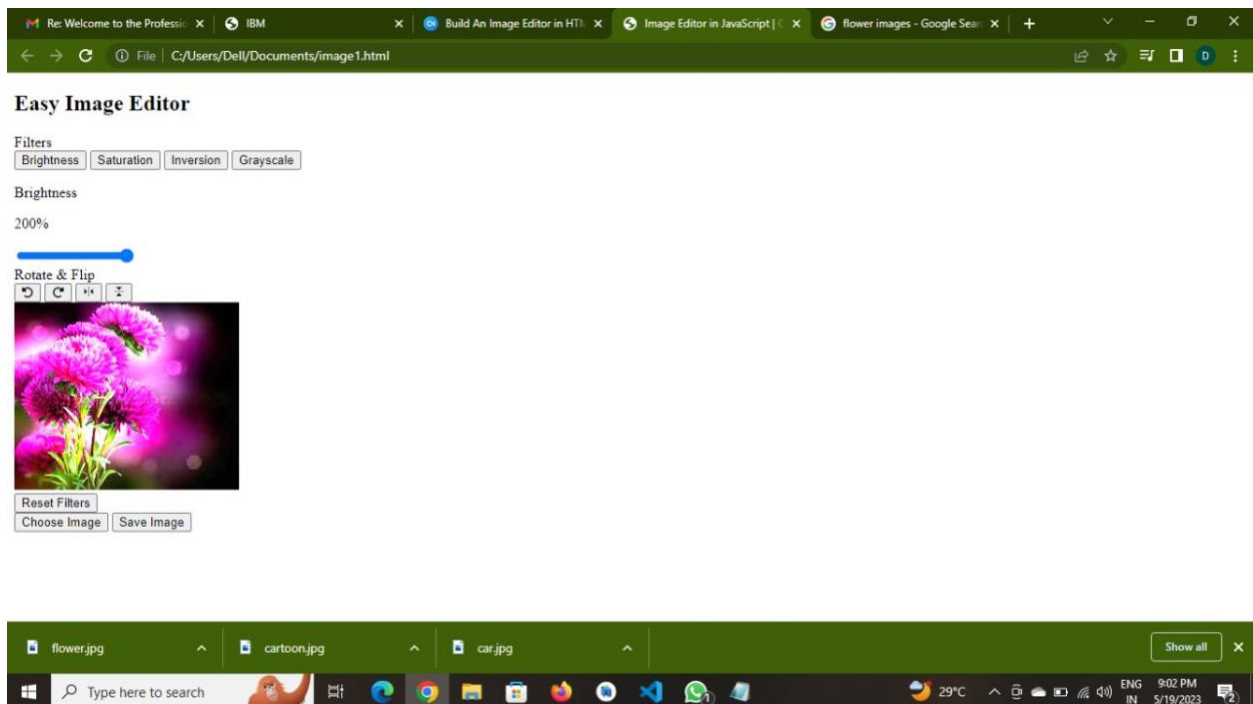
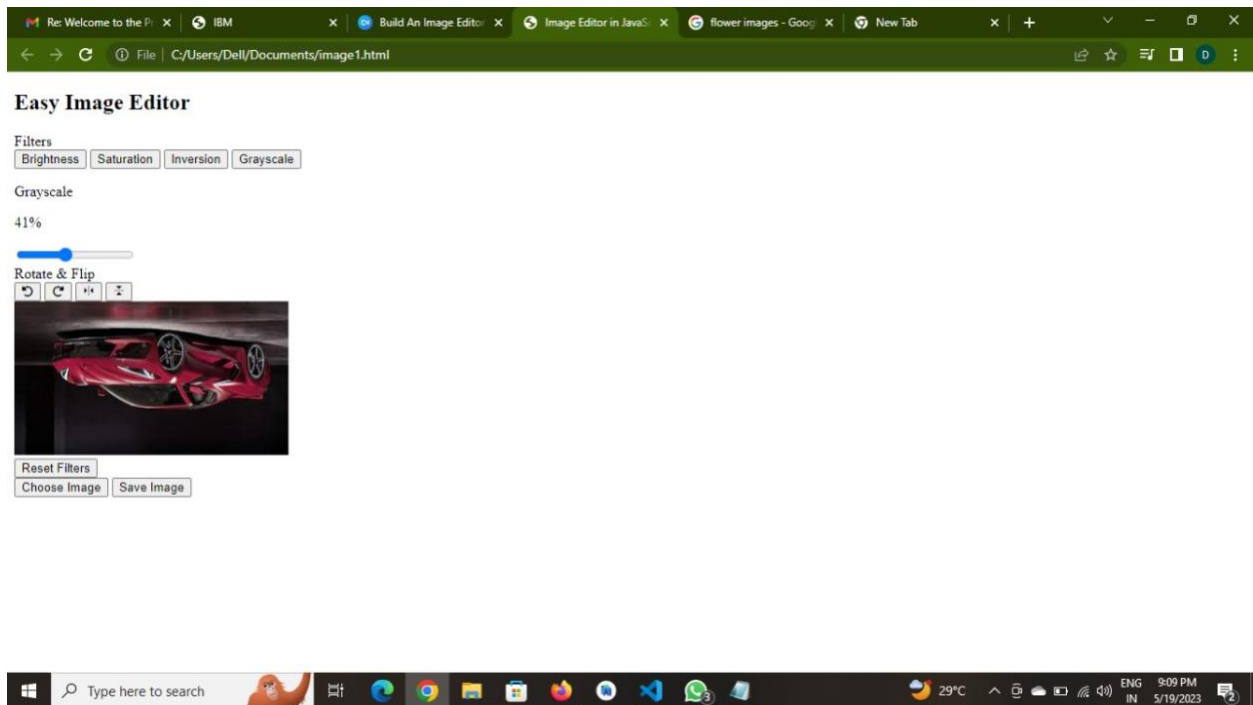
Brightness

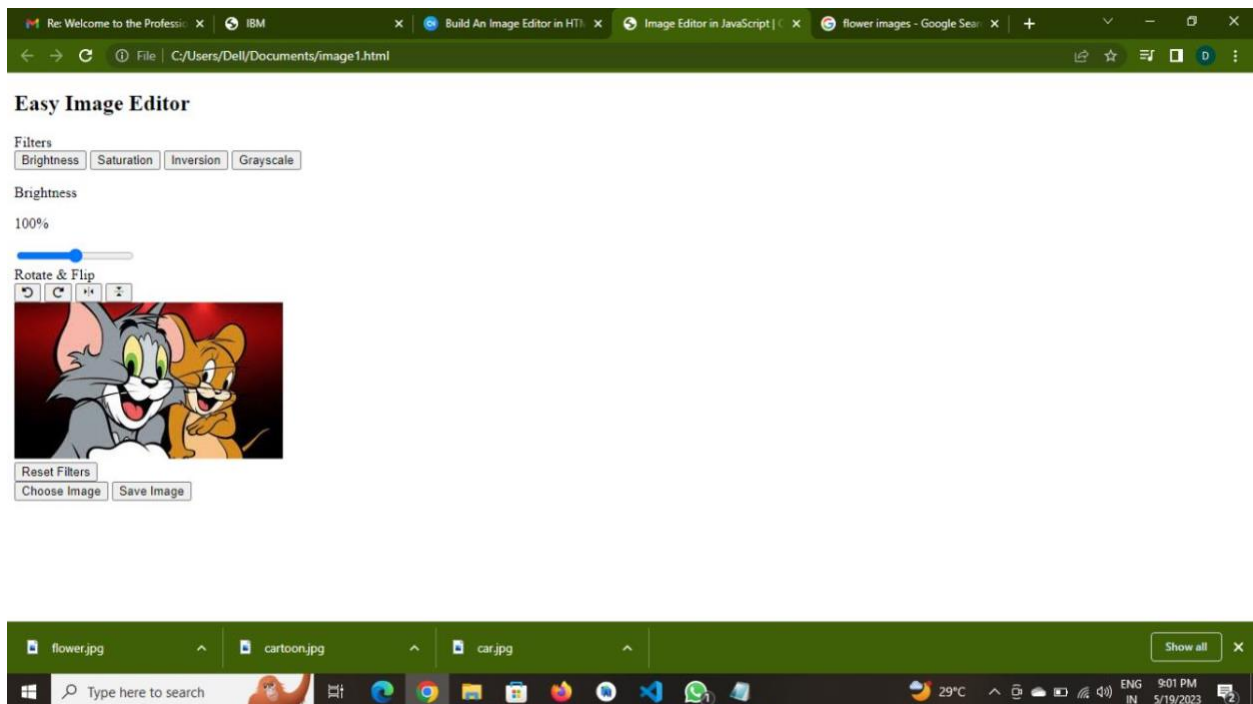
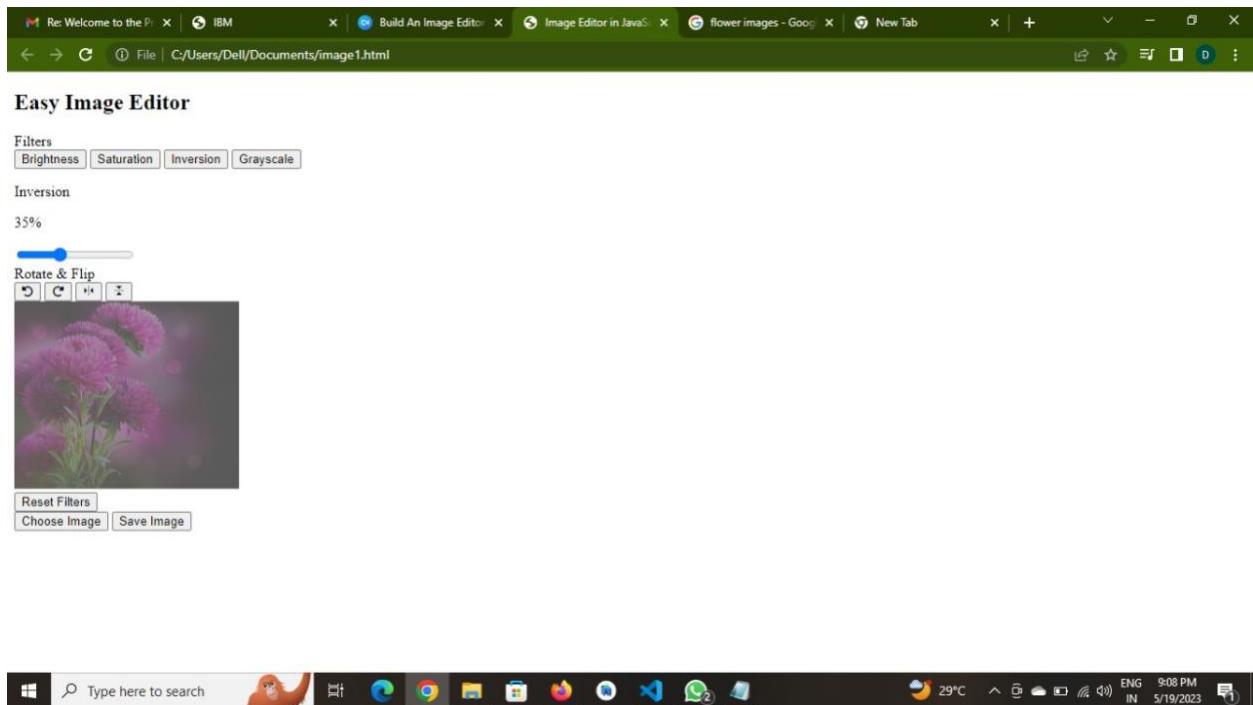
100%

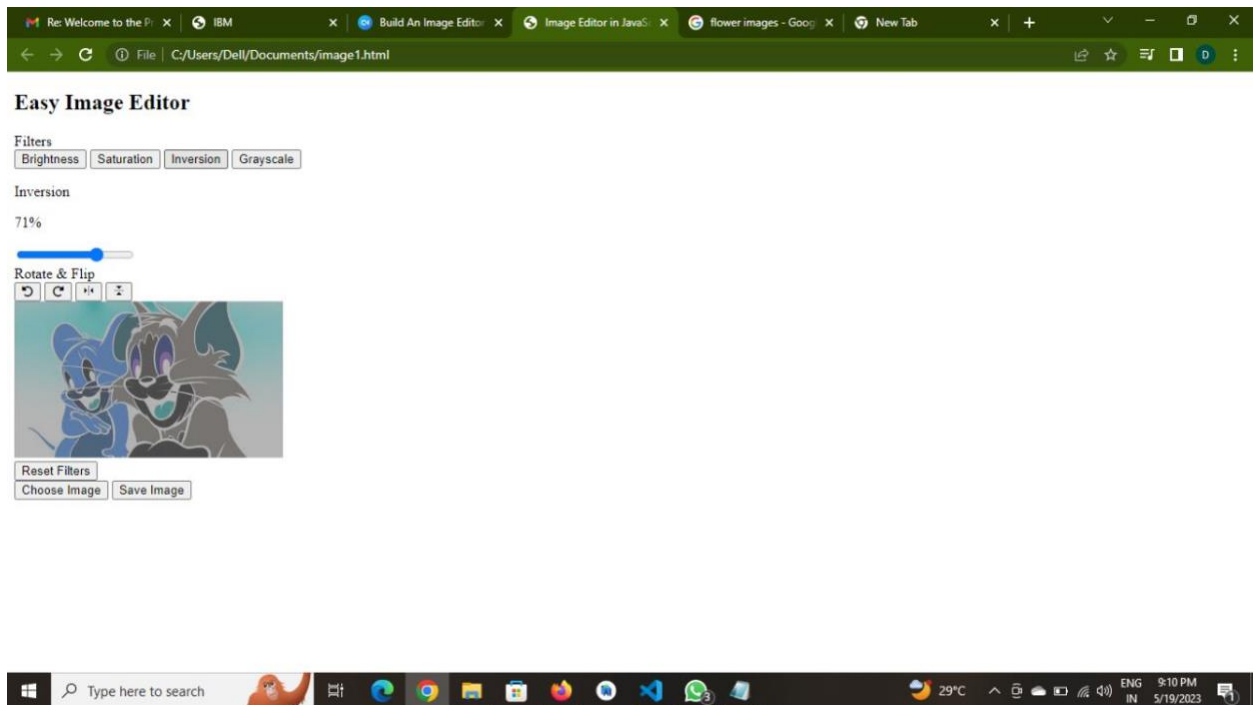


Rotate & Flip









## Table definition



USER1

No statistics available.

Name	Data type	Nullable	Length	Scale	
NAME	VARCHAR	Y	32	0	👁
EMAIL	VARCHAR	Y	32	0	👁
PASSWORD	VARCHAR	Y	32	0	👁
USERD	INTEGER	Y		0	👁

FNC49721.USER1

Back

Export to CSV

NAME	EMAIL	PASSWORD	USERD
test1	test1@gmail.com	test1	1
test2	test2@gmail.com	test2	2



Table definition

IMAGE\_URL

No statistics available.

Name	Data type	Nullable	Length	Scale	
USERD	INTEGER	Y		0	
IMAGE_BG	LONG VARCHAR	Y	32700	0	
VEHICLE_BG	LONG VARCHAR	Y	32700	0	
CARTOON_IMG	LONG VARCHAR	Y	32700	0	
SKIN_BEAUTY	LONG VARCHAR	Y	32700	0	
SKIN_DISEASE	LONG VARCHAR	Y	32700	0	
IMG_MOTION	LONG VARCHAR	Y	32700	0	
3DCARTOON_IMG	LONG VARCHAR	Y	32700	0	

FNC49721.IMAGE\_URL

Back

Export to CSV

USERD	IMAGE_BG	VEHICLE_BG	CARTOON_IMG	SKIN_BEAUTY	SKIN_DISEASE	IMG_MOTION	3DCARTOON_IMG
1	https://ai-result-rapidapi.aialabtools.com/cutout/segmentCommodity/2023/02/28/100436-8394c102-0827-70fb-6c42-10ea39c96cbb-1677578676.png						
1			https://ai-result-rapidapi.aialabtools.com/faceBody/generateHumanAnimeStyle/2023/03/05/084320-680f116c-ee52-f2b8-8414-a55d8bb322db-1678005800.png				
1	https://ai-result-rapidapi.aialabtools.com/cutout/segmentCommodity/2023/03/05/094439-6dc63a0d-a844-aec9-05e2-0a65ba9a07c6-1678009479.png						
1	https://ai-result-rapidapi.aialabtools.com/cutout/segmentCommodity/2023/03/05/094051-0fed12e6-d567-c314-4e2f-aa6ab31b11eb-1678009251.png						
1		https://ai-result-rapidapi.aialabtools.com/cutout/segmentVehicle/2023/03/04/171555-4bcb69e2-35dd-5608-3be4-e2004a54e98f-1677950155.png					
1				https://ai-result-rapidapi.aialabtools.com/faceBody/reftouchSkin/2023/03/03/185451-			



```
1 from flask import Flask, render_template, request, redirect, url_for, session
2
3
4 app = Flask(__name__)
5 app.secret_key = 'a'
6
7 @app.route("/")
8 def home():
9     return "Hello world"
10
11 if __name__ == '__main__':
12     app.run(host='0.0.0.0', debug=True)
```



```
1 #routing to home page
2 @app.route('/', methods=['POST', 'GET'])
3 def homepage():
4     return render_template('index.html')
5
6 #routing to home page after login of user
7 @app.route('/home', methods=['POST', 'GET'])
8 def after_login():
9     return render_template('home.html')
```

```
1
2 #user login code
3 @app.route('/login', methods=['POST','GET'])
4 def login_page():
5     msg = ''
6     if request.method == "POST":
7         EMAIL = request.form["email"]
8         PASSWORD = request.form["password"]
9         sql = "SELECT * FROM USER1 WHERE EMAIL=? AND PASSWORD=?"
10        stmt = ibm_db.prepare(conn, sql)
11        ibm_db.bind_param(stmt, 1, EMAIL)
12        ibm_db.bind_param(stmt, 2, PASSWORD)
13        ibm_db.execute(stmt)
14        account = ibm_db.fetch_assoc(stmt)
15        print(account)
16        if account:
17            session['Loggedin'] = True
18            session['USERD'] = account['USERD']
19            session['NAME'] = account['NAME']
20            msg = "logged in successfully !"
21            return redirect(url_for('after_login'))
22        else:
23            msg = "Incorrect Email/password"
24            return render_template('login.html', msg=msg)
25    return render_template('login.html', msg=msg)
26
```

```

1  #user registartion code
2  @app.route('/register', methods=['POST','GET'])
3  def register_page():
4      msg = ''
5      if request.method == 'POST':
6          NAME = request.form["name"]
7          EMAIL = request.form["email"]
8          PASSWORD = request.form["password"]
9          sql = "SELECT* FROM USER1 WHERE EMAIL= ? AND PASSWORD=?"
10         stmt = ibm_db.prepare(conn, sql)
11         ibm_db.bind_param(stmt, 1, EMAIL)
12         ibm_db.bind_param(stmt, 2, PASSWORD)
13         ibm_db.execute(stmt)
14         account = ibm_db.fetch_assoc(stmt)
15         print(account)
16         if account:
17             msg = "Your deatils are already exists in the database Please logi
18 n"
19             return render_template('login.html')
20         elif not re.match(r'^[^\s@]+@[^\s@]+\.[^\s@]+', EMAIL):
21             msg = "Invalid Email Address!"
22         else:
23             sql = "SELECT count(*) FROM USER1"
24             stmt = ibm_db.prepare(conn, sql)
25             ibm_db.execute(stmt)
26             length = ibm_db.fetch_assoc(stmt)
27             print(length)
28             insert_sql = "INSERT INTO USER1 VALUES (?, ?, ?, ?)"
29             prep_stmt = ibm_db.prepare(conn, insert_sql)
30             ibm_db.bind_param(prepare_stmt, 1, NAME)
31             ibm_db.bind_param(prepare_stmt, 2, EMAIL)
32             ibm_db.bind_param(prepare_stmt, 3, PASSWORD)
33             ibm_db.bind_param(prepare_stmt, 4, length['1']+1)
34             ibm_db.execute(prepare_stmt)
35             msg = "Successfully registered!"
36             return render_template('login.html', msg=msg)
37         return render_template('register.html', msg=msg)

```



```
1  # for logout the user start
2  @app.route('/logout')
3  def logout():
4      session.pop('loggedin', None)
5      session.pop('USERD', None)
6      return render_template('index.html'
    )
```

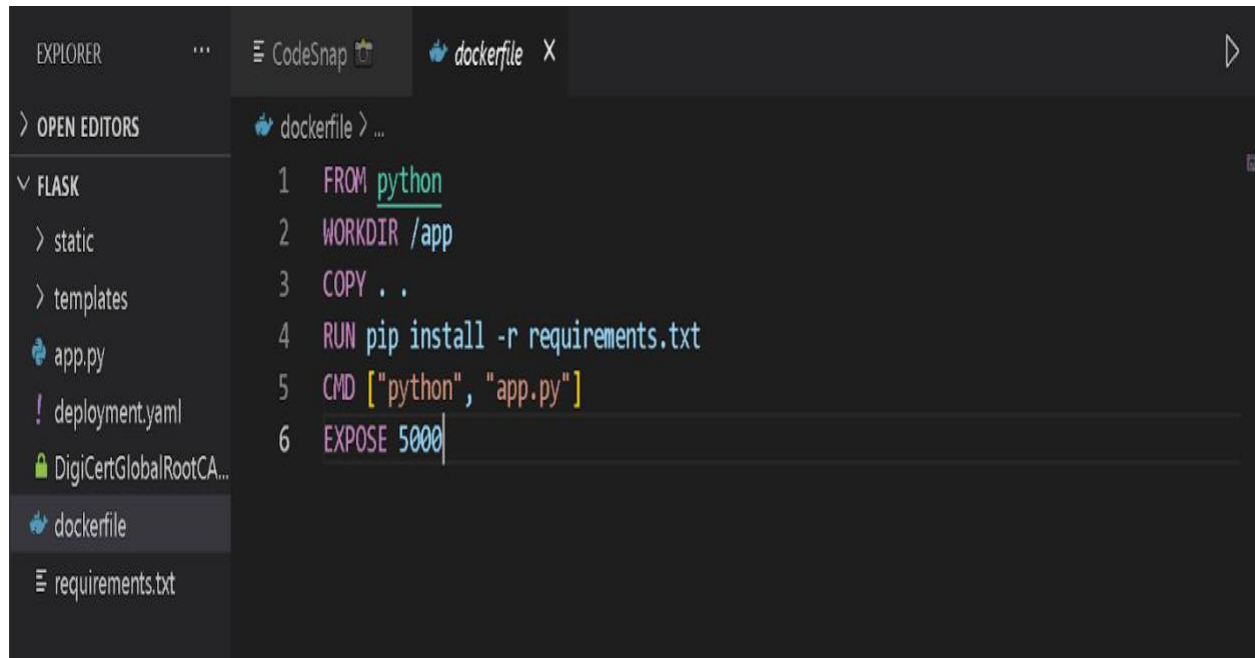


```
1  # this lines of code for displaying the images retrived from db2 sta
2  @app.route('/beauty_images')
3  def beauty_img_ai():
4      user='login'
5      # user='loggedin'
6      sql = "SELECT * FROM IMAGE_URL WHERE USERD=" +str(session['USERD'
7  ]) stmt = ibm_db.prepare(conn, sql)
8      ibm_db.execute(stmt)
9      row = []
10     while True:
11         data = ibm_db.fetch_assoc(stmt)
12         if not data:
13             break
14         else:
15             row.append(data)
16     print('rows: ', row)
17     return render_template("My_images.html", rows=row, user1=user)
```



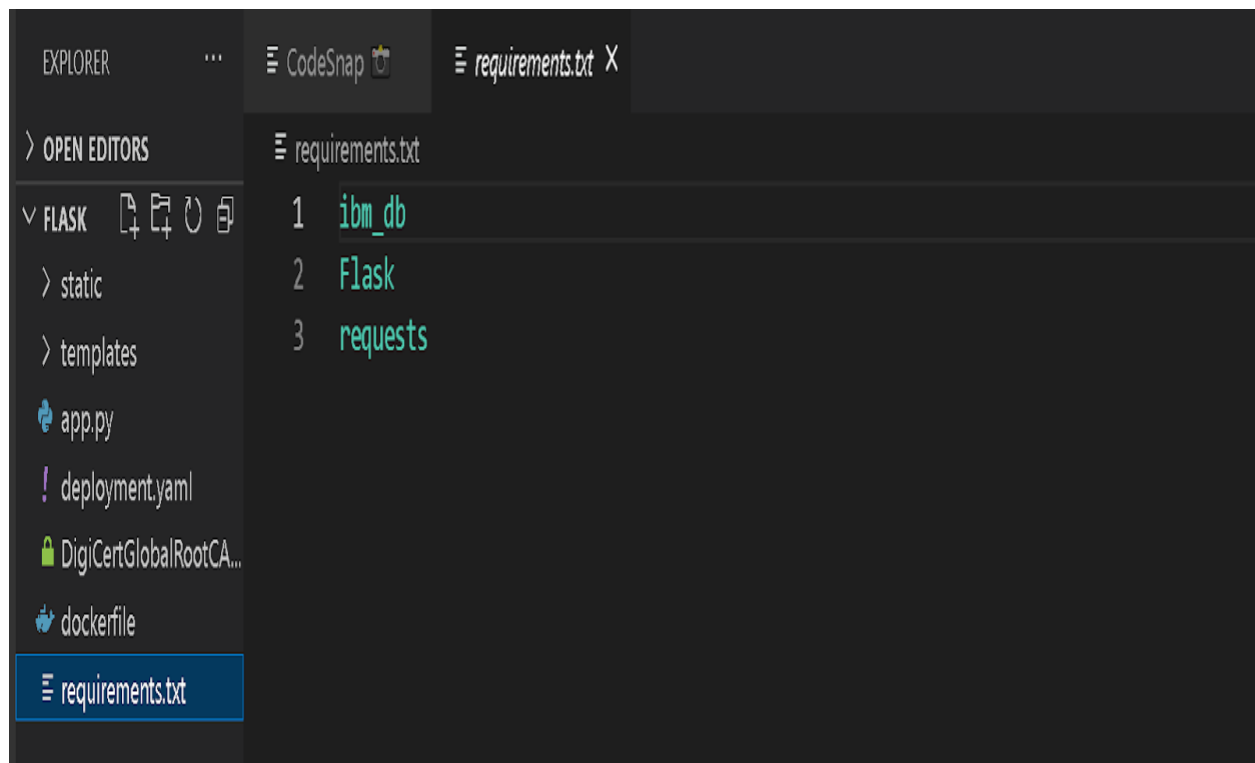
```
1  # main function start
2  if __name__=='__main__':
3      app.run(debug=True,host='0.0.0.0'
    )
```

# Task 1:



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left displays a project structure with folders 'static' and 'templates', and files 'app.py', 'deployment.yaml', 'DigiCertGlobalRootCA...', 'dockerfile', and 'requirements.txt'. The 'dockerfile' file is selected and open in the editor. The editor window shows the following Dockerfile content:

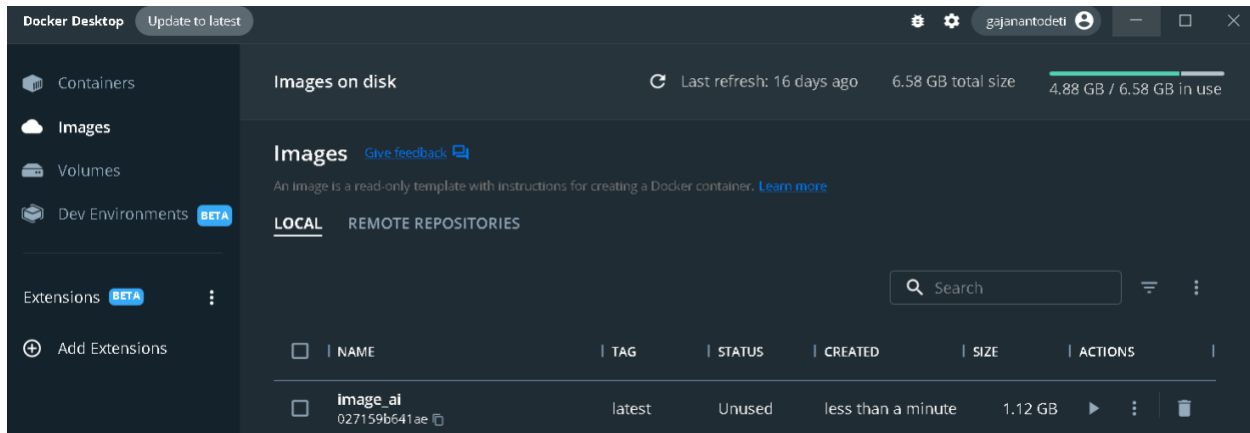
```
1 FROM python
2 WORKDIR /app
3 COPY . .
4 RUN pip install -r requirements.txt
5 CMD ["python", "app.py"]
6 EXPOSE 5000
```



The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left is the same as the previous image. The 'requirements.txt' file is selected and open in the editor. The editor window shows the following content:

```
1 ibm_db
2 Flask
3 requests
```

## Task 2:



```
C:\Users\DELL> ibmcloud login
API endpoint: https://cloud.ibm.com
Region: eu-de

C:\Users\DELL> ibmcloud plugin install container-registry
Looking up 'container-registry' from repository 'IBM Cloud'...
Plug-in 'container-registry[cr] 1.0.6' found in repository 'IBM Cloud'
ac504d030134: Pushing [=====>] 160.3MB/528.8MB
5' or not? [y/N] > y
Attempting to download the binary file...
11.91 MiB / 11.91 MiB [-----] 100.00% 5s
12484608 bytes downloaded
ac504d030134: Pushing [=====>] 158.7MB/528.8MB
OK
Plug-in 'container-registry 1.0.6' was successfully installed into C:\Users\DELL\bluemix\plugins\container-registry. Use 'ibmcloud p

C:\Users\DELL> ibmcloud cr region-set us-south
The region is set to 'us-south', the registry is 'us.icr.io'.

ac504d030134: Pushing [=====>] 157.5MB/528.8MB

C:\Users\DELL> ibmcloud cr namespace-add myimage_ai
No resource group is targeted. Therefore, the default resource group for the account ('Default') is targeted.
Adding namespace 'myimage_ai' in resource group 'Default' for account gajanan todeti's Account in registry us.icr.io...
Successfully added namespace 'myimage_ai'

ac504d030134: Pushing [=====>] 147.6MB/528.8MB
ac504d030134: Pushing [----->] 157MB/528.8MB
Logging 'docker' in to 'us.icr.io'...
ac504d030134: Pushing [----->] 155.9MB/528.8MB

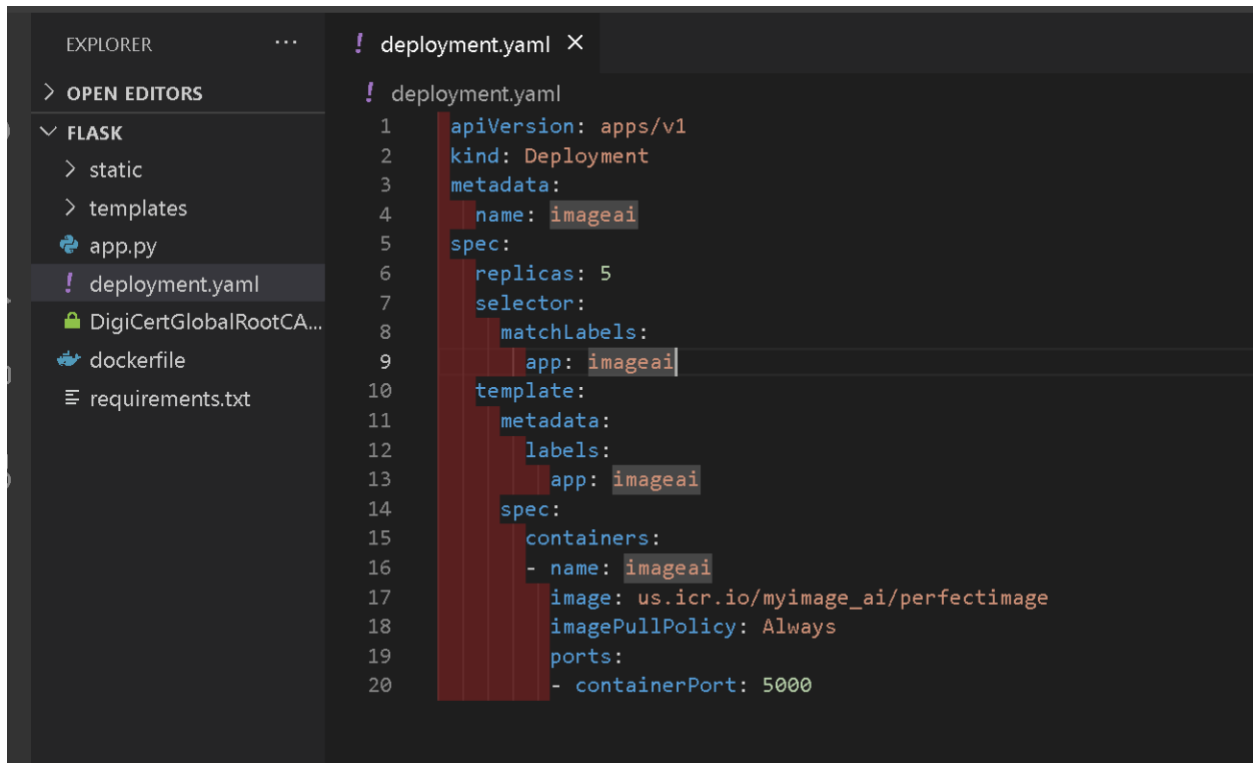
C:\Users\DELL> docker tag image_ai us.icr.io/myimage_ai/perfectimage

C:\Users\DELL> docker tag push us.icr.io/myimage_ai/perfectimage
non response from daemon: No such image: push:latest

C:\Users\DELL> docker push us.icr.io/myimage_ai/perfectimage
sing default tag: latest
The push refers to repository [us.icr.io/myimage_ai/perfectimage]
d1e28035ea: Pushed
050e41efaf9: Pushed
4dd8426b147: Pushed
7985d454104: Pushed
041520c0f1e: Pushed
F706e6a9f98: Pushed
baa05faf31a: Pushed
c504d030134: Pushing [=====>] 145.4MB/528.8MB
```



# Task 3:



```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: imageai
5  spec:
6    replicas: 5
7    selector:
8      matchLabels:
9        app: imageai
10   template:
11     metadata:
12       labels:
13         app: imageai
14     spec:
15       containers:
16       - name: imageai
17         image: us.icr.io/myimage_ai/perfectimage
18         imagePullPolicy: Always
19         ports:
20         - containerPort: 5000
```

kubernetes

default

Search

