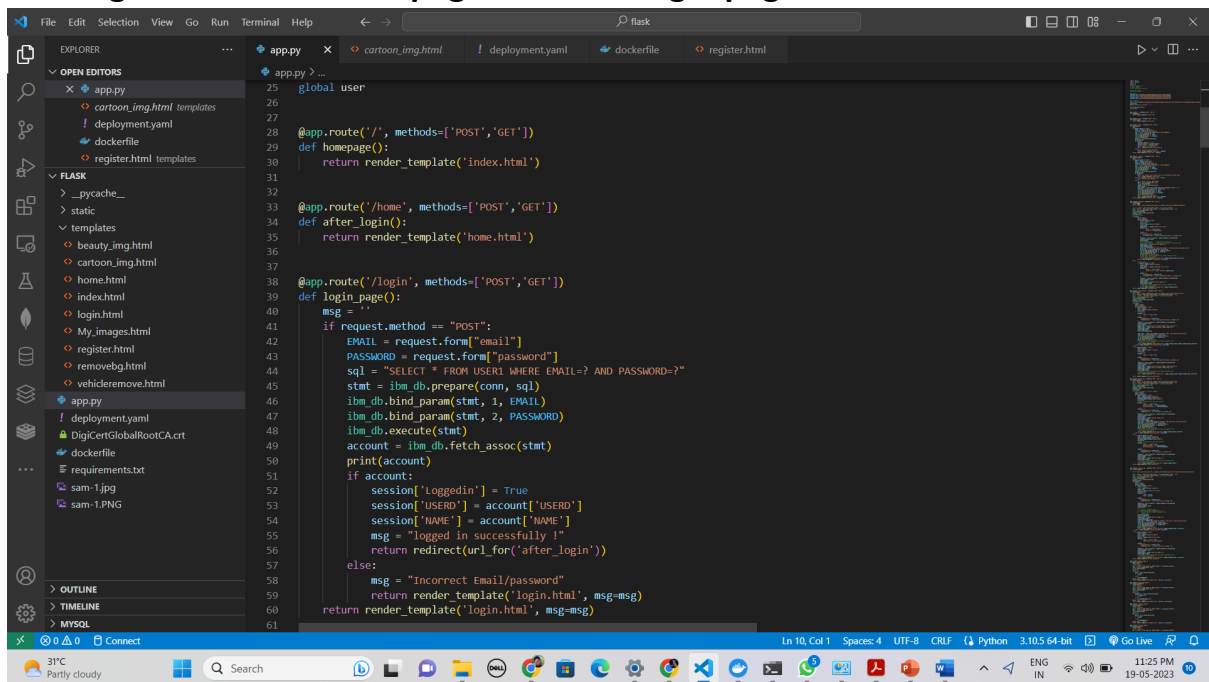


Connecting UI And DB With Flask

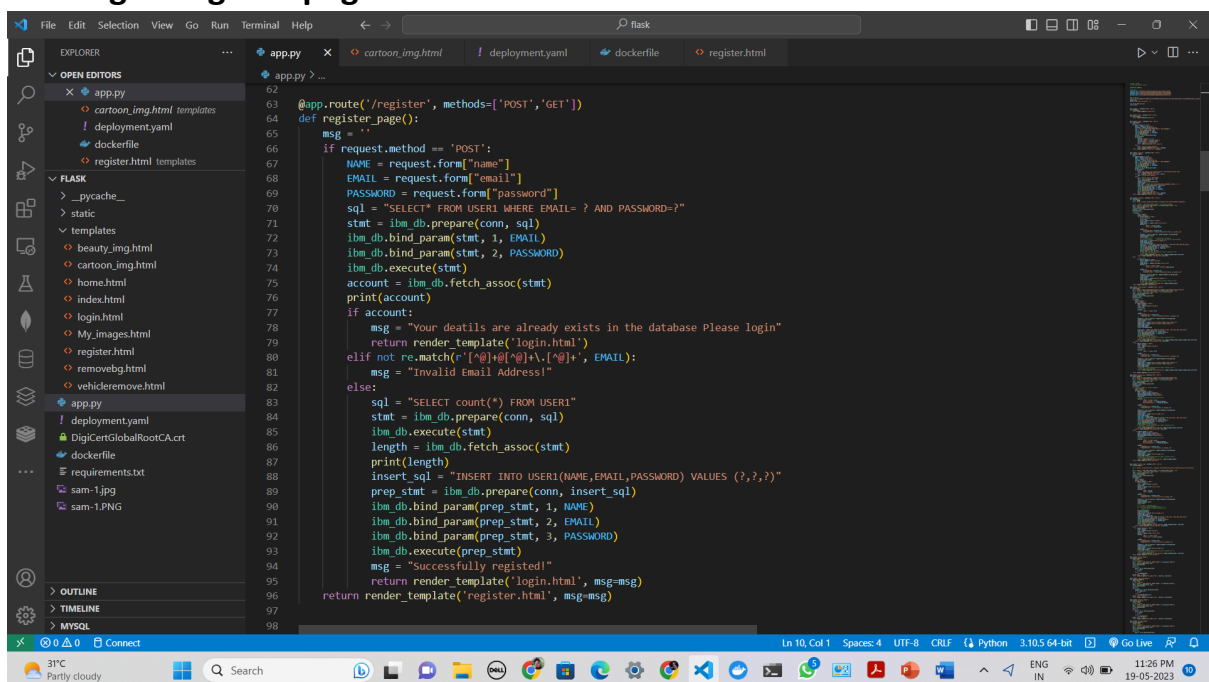
Team ID	NM2023TMID16640
Project Name	Pixel Perfection: Transforming your photos with our cutting-edge image editing platform

Routing to the dashboard page and User login page



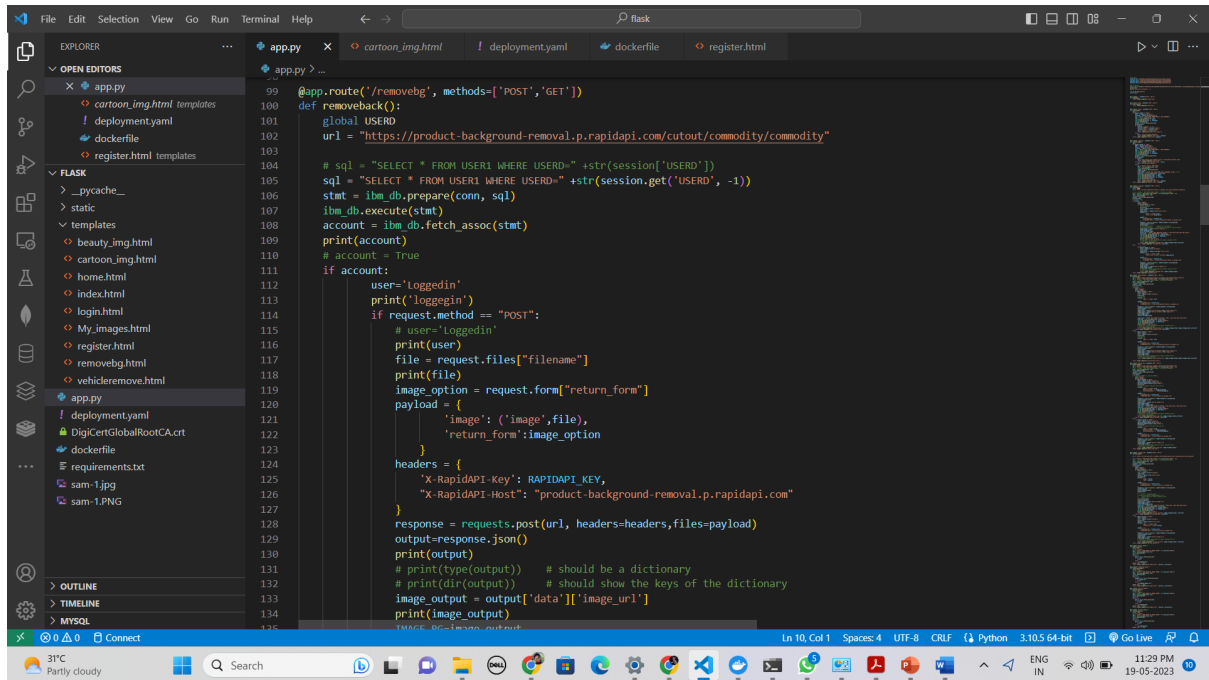
```
app.py > ...
25 global user
26
27
28 @app.route('/', methods=['POST','GET'])
29 def homepage():
30     return render_template('index.html')
31
32
33 @app.route('/home', methods=['POST','GET'])
34 def after_login():
35     return render_template('home.html')
36
37
38 @app.route('/login', methods=['POST','GET'])
39 def login_page():
40     msg = ''
41     if request.method == "POST":
42         EMAIL = request.form["email"]
43         PASSWORD = request.form["password"]
44         sql = "SELECT * FROM USER1 WHERE EMAIL=? AND PASSWORD=?"
45         stmt = ibm_db.prepare(conn, sql)
46         ibm_db.bind_param(stmt, 1, EMAIL)
47         ibm_db.bind_param(stmt, 2, PASSWORD)
48         ibm_db.execute(stmt)
49         account = ibm_db.fetch_assoc(stmt)
50         print(account)
51         if account:
52             session['loggedin'] = True
53             session['USERID'] = account['USERID']
54             session['NAME'] = account['NAME']
55             msg = "logged in successfully !"
56             return redirect(url_for('after_login'))
57         else:
58             msg = "Incorrect Email/password"
59             return render_template('login.html', msg=msg)
60     return render_template('login.html', msg=msg)
61
```

Routing to register page



```
app.py > ...
62
63 @app.route('/register', methods=['POST','GET'])
64 def register_page():
65     msg = ''
66     if request.method == "POST":
67         NAME = request.form["name"]
68         EMAIL = request.form["email"]
69         PASSWORD = request.form["password"]
70         sql = "SELECT * FROM USER1 WHERE EMAIL=? AND PASSWORD=?"
71         stmt = ibm_db.prepare(conn, sql)
72         ibm_db.bind_param(stmt, 1, EMAIL)
73         ibm_db.bind_param(stmt, 2, PASSWORD)
74         ibm_db.execute(stmt)
75         account = ibm_db.fetch_assoc(stmt)
76         print(account)
77         if account:
78             msg = "Your details are already exists in the database Please login"
79             return render_template('login.html')
80         elif not re.match(r'^(.+)@(.+)\.(.+)$', EMAIL):
81             msg = "Invalid Email Address!"
82         else:
83             sql = "SELECT count(*) FROM USER1"
84             stmt = ibm_db.prepare(conn, sql)
85             ibm_db.execute(stmt)
86             length = ibm_db.fetch_assoc(stmt)
87             print(length)
88             insert_sql = "INSERT INTO USER1(NAME,EMAIL,PASSWORD) VALUES (?,?,?)"
89             prep_stmt = ibm_db.prepare(conn, insert_sql)
90             ibm_db.bind_param(prep_stmt, 1, NAME)
91             ibm_db.bind_param(prep_stmt, 2, EMAIL)
92             ibm_db.bind_param(prep_stmt, 3, PASSWORD)
93             ibm_db.execute(prep_stmt)
94             msg = "Successfully registered!"
95             return render_template('login.html', msg=msg)
96     return render_template('register.html', msg=msg)
97
98
```

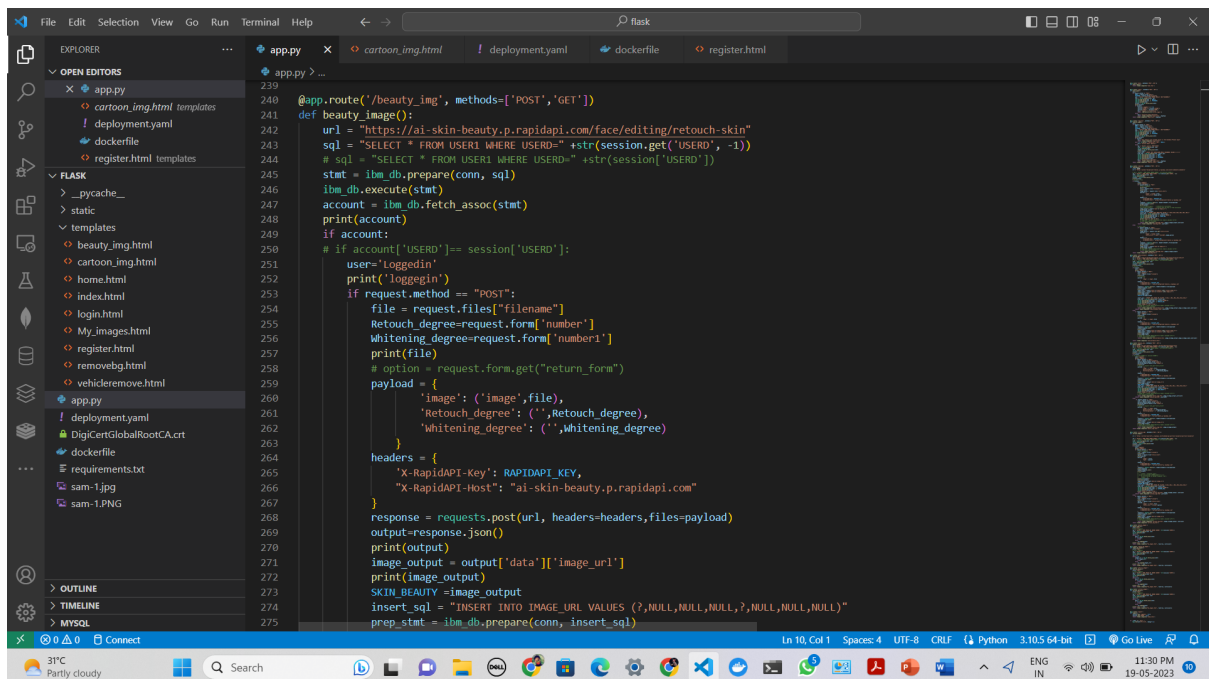
Routing to Remove Background page



The screenshot shows the Visual Studio Code editor with the Flask application code. The Explorer panel on the left shows the project structure, including the 'app.py' file. The main editor displays the code for the 'removebg' endpoint, which is a POST method. The code includes a global 'USERID' variable, a database connection, and a query to fetch user information. It then checks if the user is logged in and if the request method is POST. If so, it processes the file upload and sends a POST request to the 'product-background-removal.p.rapidapi.com' API. The response is printed and the image URL is extracted from the output.

```
99 @app.route('/removebg', methods=['POST','GET'])
100 def removeback():
101     global USERID
102     url = "https://product-background-removal.p.rapidapi.com/cutout/commodity/commodity"
103
104     # sql = "SELECT * FROM USER1 WHERE USERID=" +str(session['USERID'])
105     sql = "SELECT * FROM USER1 WHERE USERID=" +str(session.get('USERID', -1))
106     stat = ibm_db.prepare(conn, sql)
107     ibm_db.execute(stat)
108     account = ibm_db.fetch_assoc(stat)
109     print(account)
110     # account = True
111     if account:
112         user='loggedin'
113         print('loggedin')
114         if request.method == "POST":
115             # user='loggedin'
116             print(user)
117             file = request.files["filename"]
118             print(file)
119             image_option = request.form["return_form"]
120             payload = {
121                 'image': ('image',file),
122                 'return_form':image_option
123             }
124             headers = {
125                 'X-RapidAPI-Key': RAPIDAPI_KEY,
126                 'X-RapidAPI-Host': "product-background-removal.p.rapidapi.com"
127             }
128             response = requests.post(url, headers=headers,files=payload)
129             output=response.json()
130             print(output)
131             # print(type(output)) # should be a dictionary
132             # print(dir(output)) # should show the keys of the dictionary
133             image_output = output["data"]["image_url"]
134             print(image_output)
```

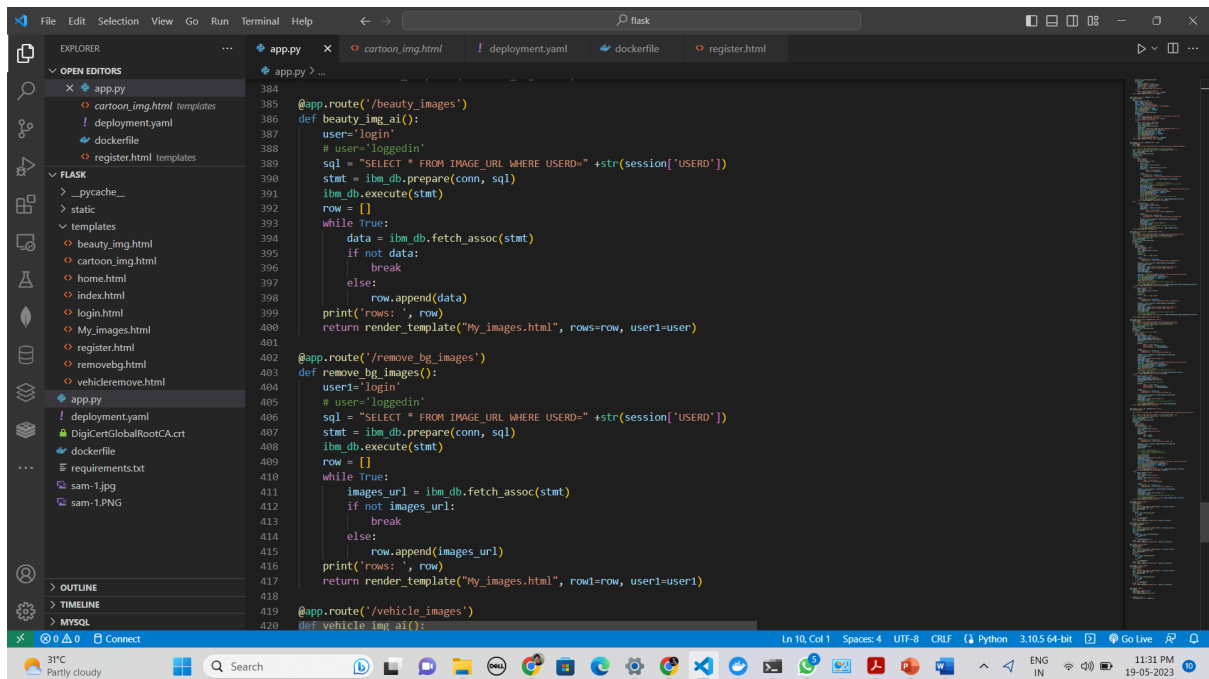
Routing to the AI Skin Beauty page



The screenshot shows the Visual Studio Code editor with the Flask application code. The Explorer panel on the left shows the project structure, including the 'app.py' file. The main editor displays the code for the 'beauty_img' endpoint, which is a POST method. The code includes a global 'USERID' variable, a database connection, and a query to fetch user information. It then checks if the user is logged in and if the request method is POST. If so, it processes the file upload and sends a POST request to the 'ai-skin-beauty.p.rapidapi.com' API. The response is printed and the image URL is extracted from the output.

```
239 @app.route('/beauty_img', methods=['POST','GET'])
240 def beauty_image():
241     url = "https://ai-skin-beauty.p.rapidapi.com/face/editing/retouch-skin"
242     sql = "SELECT * FROM USER1 WHERE USERID=" +str(session.get('USERID', -1))
243     # sql = "SELECT * FROM USER1 WHERE USERID=" +str(session['USERID'])
244     stat = ibm_db.prepare(conn, sql)
245     ibm_db.execute(stat)
246     account = ibm_db.fetch_assoc(stat)
247     print(account)
248     if account:
249         # if account['USERID']== session['USERID']:
250         user='loggedin'
251         print('loggedin')
252         if request.method == "POST":
253             file = request.files["filename"]
254             Retouch_degree=request.form["number"]
255             Whitening_degree=request.form["number1"]
256             print(file)
257             # option = request.form.get("return_form")
258             payload = {
259                 'image': ('image',file),
260                 'Retouch_degree': ('',Retouch_degree),
261                 'Whitening_degree': ('',Whitening_degree)
262             }
263             headers = {
264                 'X-RapidAPI-Key': RAPIDAPI_KEY,
265                 'X-RapidAPI-Host': "ai-skin-beauty.p.rapidapi.com"
266             }
267             response = requests.post(url, headers=headers,files=payload)
268             output=response.json()
269             print(output)
270             image_output = output["data"]["image_url"]
271             print(image_output)
272             SKIN_BEAUTY=image_output
273             insert_sql = "INSERT INTO IMAGE_URL VALUES (?,NULL,NULL,?,NULL,NULL,NULL)"
274             prep_stat = ibm_db.prepare(conn, insert_sql)
```

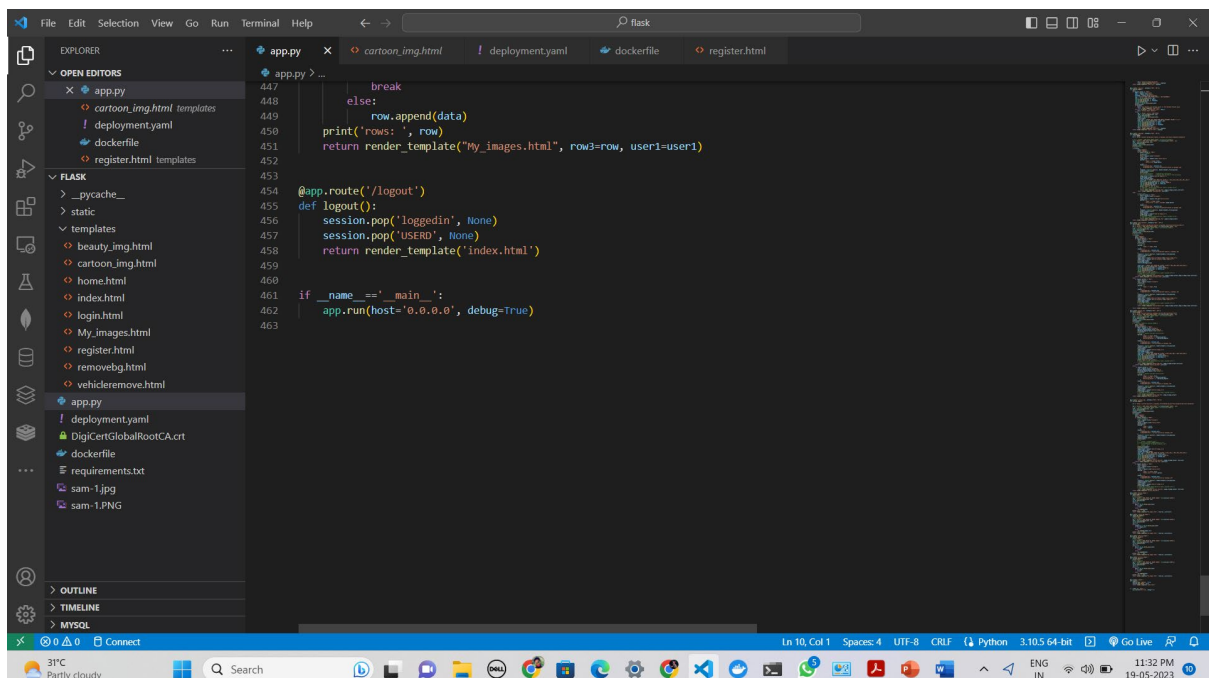
Routing to display previous images



The screenshot shows the Visual Studio Code editor with the `app.py` file open. The file contains two routes for displaying previous images. The first route, `@app.route('/beauty_images')`, calls `def beauty_img_ai()` and uses a SQL query to fetch image data from the `IMAGE_URL` table where the user is logged in. It then iterates through the data and renders the `My_images.html` template. The second route, `@app.route('/remove_bg_images')`, calls `def remove_bg_images()` and uses a similar SQL query to fetch image data. It then iterates through the data and renders the `My_images.html` template. The file also includes a `def vehicle_img_ai()` function.

```
384
385 @app.route('/beauty_images')
386 def beauty_img_ai():
387     user='login'
388     # user='loggedin'
389     sql = "SELECT * FROM IMAGE_URL WHERE USERID="+str(session['USERID'])
390     stat = ibm_db.prepare(conn, sql)
391     ibm_db.execute(stat)
392     row = []
393     while True:
394         data = ibm_db.fetch_assoc(stat)
395         if not data:
396             break
397         else:
398             row.append(data)
399     print('rows: ', row)
400     return render_template("My_images.html", rows=row, user1=user)
401
402 @app.route('/remove_bg_images')
403 def remove_bg_images():
404     user='login'
405     # user='loggedin'
406     sql = "SELECT * FROM IMAGE_URL WHERE USERID="+str(session['USERID'])
407     stat = ibm_db.prepare(conn, sql)
408     ibm_db.execute(stat)
409     row = []
410     while True:
411         images_url = ibm_db.fetch_assoc(stat)
412         if not images_url:
413             break
414         else:
415             row.append(images_url)
416     print('rows: ', row)
417     return render_template("My_images.html", row1=row, user1=user1)
418
419 @app.route('/vehicle_images')
420 def vehicle_img_ai():
```

main function



The screenshot shows the Visual Studio Code editor with the `app.py` file open. The file contains a `def logout()` function that removes the user from the session and renders the `index.html` template. The file also includes a `if __name__ == '__main__':` block that runs the application on `host='0.0.0.0'` with `debug=True`.

```
447         break
448     else:
449         row.append(data)
450     print('rows: ', row)
451     return render_template("My_images.html", row3=row, user1=user1)
452
453
454 @app.route('/logout')
455 def logout():
456     session.pop('loggedin', None)
457     session.pop('USERID', None)
458     return render_template('index.html')
459
460
461 if __name__ == '__main__':
462     app.run(host='0.0.0.0', debug=True)
463
```