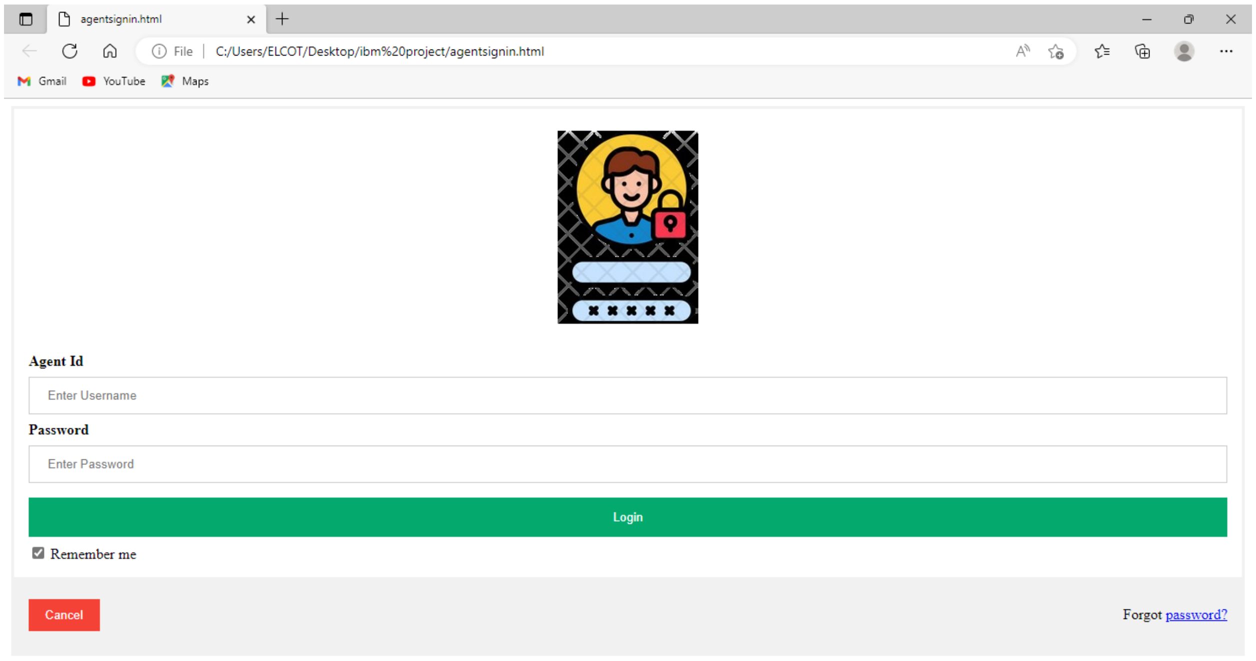
CUSTOMER CARE REGISTRY

TEAM ID:PNT2022TMID37726

AGENT LOGIN. HTML:

```
<!DOCTYPE
html>
            <html>
            <head>
            <link rel="stylesheet" href="https://static86.s3.jp-tok.cloud-object-
            storage.appdomain.cloud/mystyle1.css">
            </head>
            <body>

            <form action="action_page.php" method="post">
              <div class="imgcontainer">
                <img src="https://static86.s3.jp-tok.cloud-object-
            storage.appdomain.cloud/img_avatar2.png">
              </div>

              <div class="container">
                <label for="uname"><b>Agent Id</b></label>
                <input type="text" placeholder="Enter Username" name="uname" required>

                <label for="psw"><b>Password</b></label>
                <input type="password" placeholder="Enter Password" name="psw" required>

                <button type="submit">Login</button>
                <label>
                  <input type="checkbox" checked="checked" name="remember"> Remember me
                </label>
              </div>

              <div class="container" style="background-color:#f1f1f1">
                <button type="button" class="cancelbtn">Cancel</button>
                <span class="psw">Forgot <a href="#">password?</a></span>
              </div>
            </form>
            </body>
            </html>
```

AGENT LOGIN PNG:


agentsignin.html

C:/Users/ELCOT/Desktop/ibm%20project/agentsignin.html

Gmail YouTube Maps

Agent Id

Enter Username

Password

Enter Password

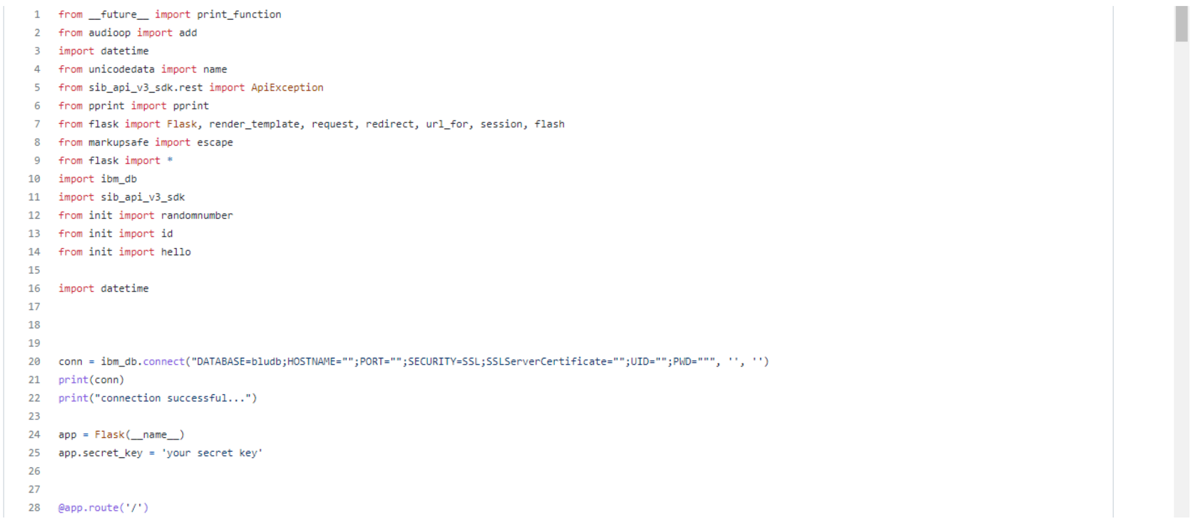
Login

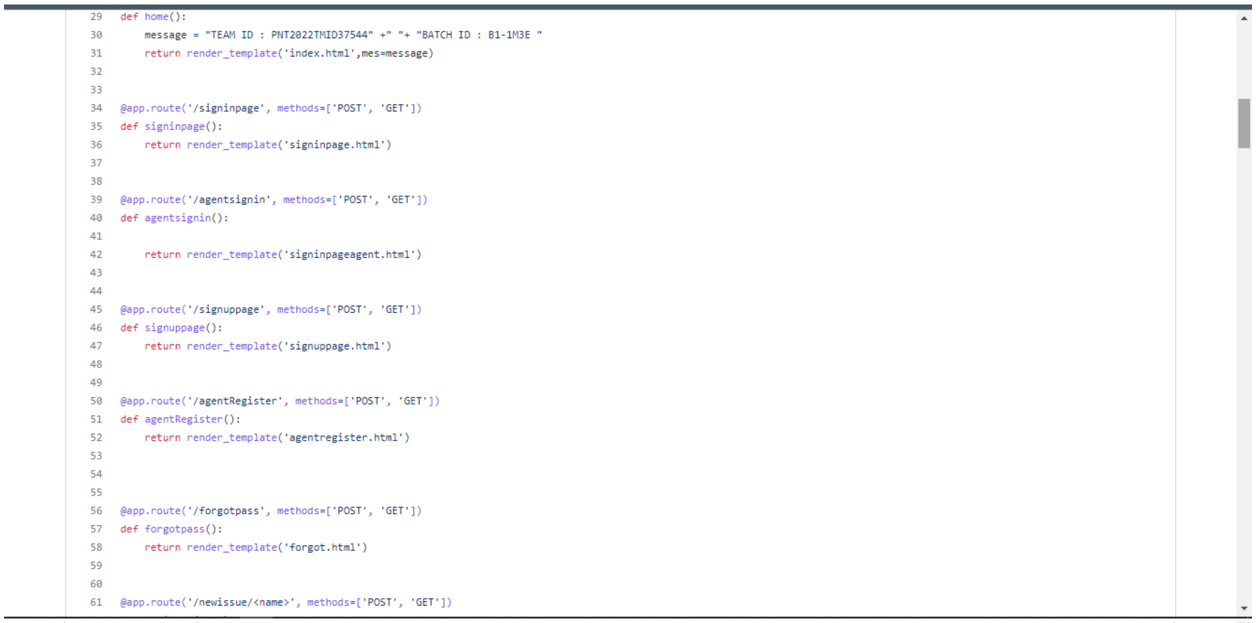
Remember me

Cancel

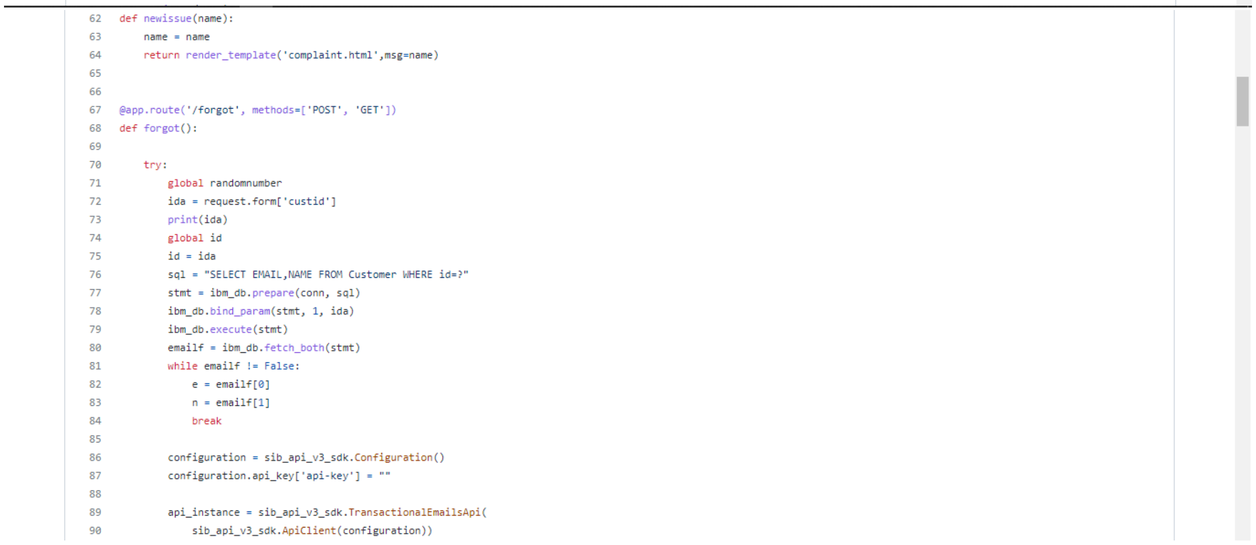
Forgot password?

APP.PY:

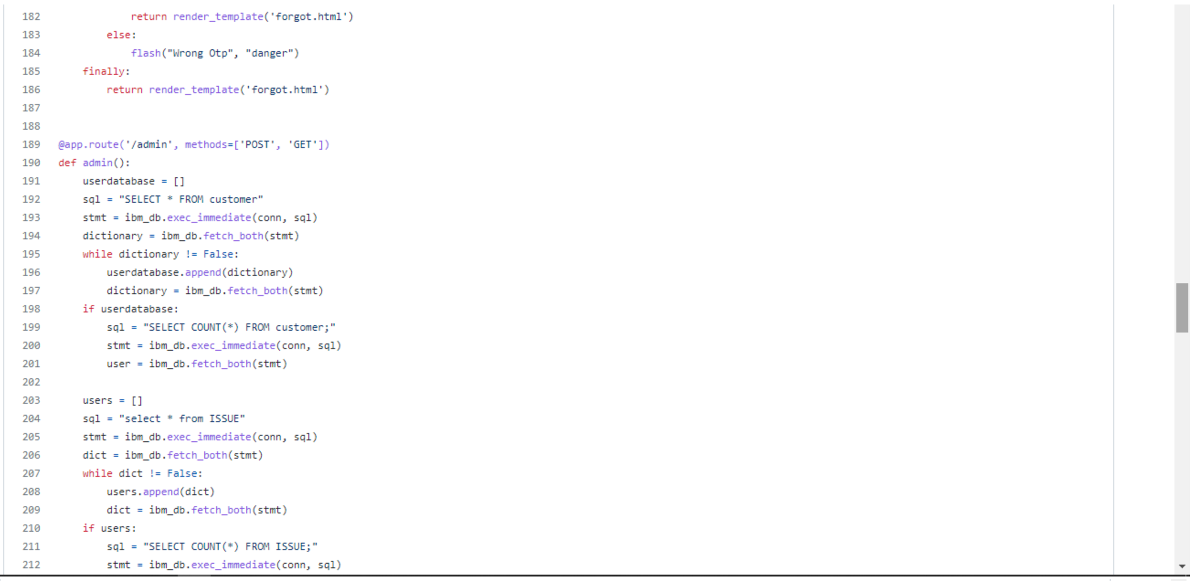

```python
from __future__ import print_function
from audioop import add
import datetime
from unicodedata import name
from sib_api_v3_sdk.rest import ApiException
from pprint import pprint
from flask import Flask, render_template, request, redirect, url_for, session, flash
from markupsafe import escape
from flask import *
import ibm_db
import sib_api_v3_sdk
from init import randomnumber
from init import id
from init import hello

import datetime



conn = ibm_db.connect("DATABASE=bludb;HOSTNAME="";PORT="";SECURITY=SSL;SSLServerCertificate="";UID="";PWD=""", '', '')
print(conn)
print("connection successful...")

app = Flask(__name__)
app.secret_key = 'your secret key'


@app.route('/')
```



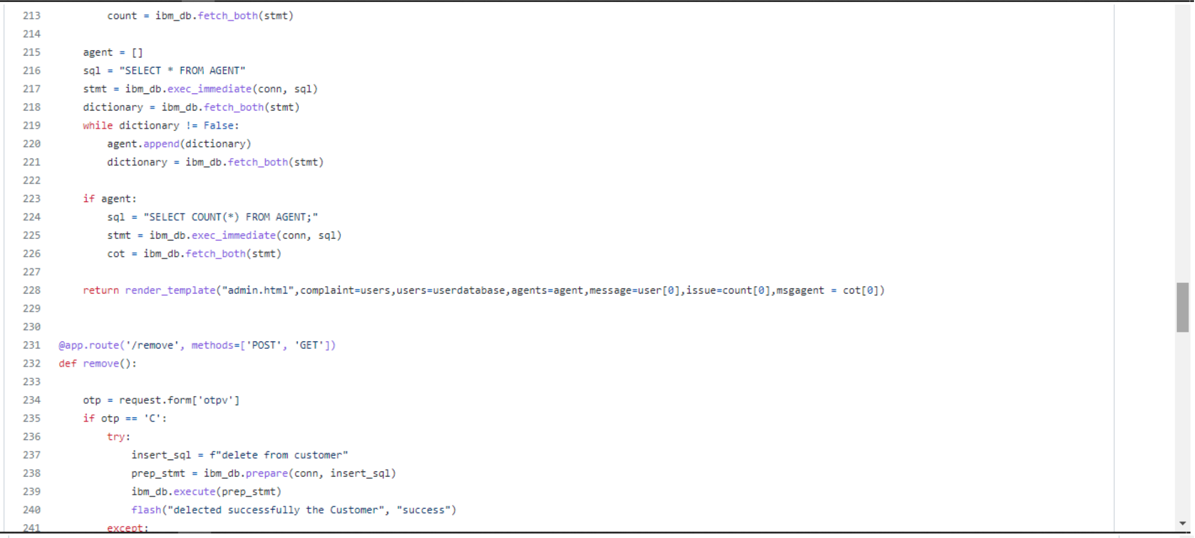
```python
def home():
    message = "TEAM ID : PNT2022TMID37544" +" "+ "BATCH ID : B1-1M3E "
    return render_template('index.html',mes=message)


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


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

    return render_template('signinpageagent.html')


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


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



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


@app.route('/newissue/<name>', methods=['POST', 'GET'])
```



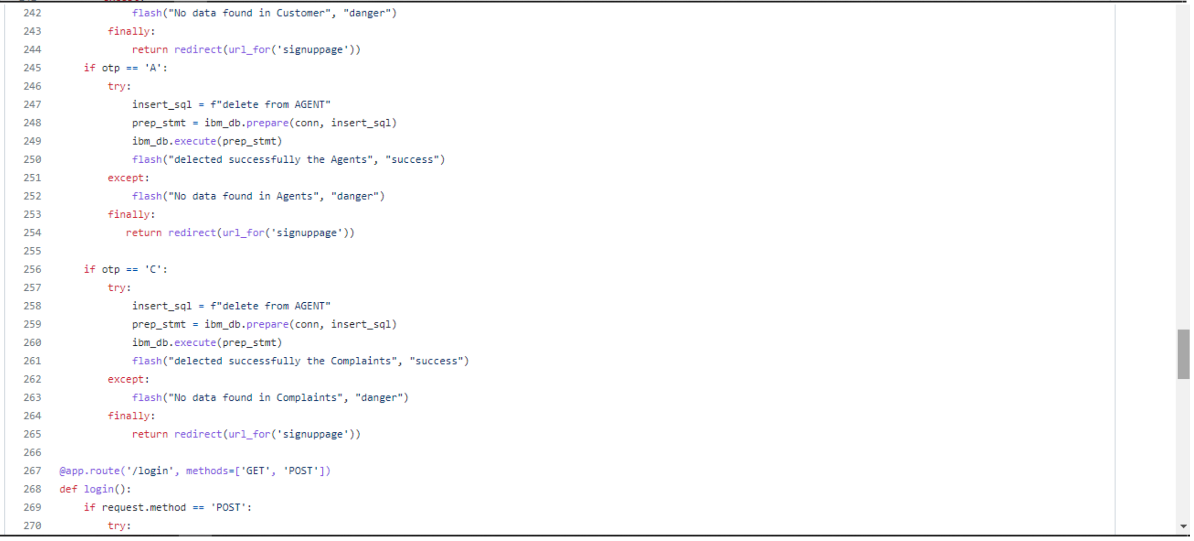
```python
def newissue(name):
    name = name
    return render_template('complaint.html',msg=name)


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

    try:
        global randomnumber
        ida = request.form['custid']
        print(ida)
        global id
        id = ida
        sql = "SELECT EMAIL,NAME FROM Customer WHERE id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, ida)
        ibm_db.execute(stmt)
        emailf = ibm_db.fetch_both(stmt)
        while emailf != False:
            e = emailf[0]
            n = emailf[1]
            break

        configuration = sib_api_v3_sdk.Configuration()
        configuration.api_key['api-key'] = ""

        api_instance = sib_api_v3_sdk.TransactionalEmailsApi(
            sib_api_v3_sdk.ApiClient(configuration))
        subject = "Verification for Password"
        html_content = "<html><body><h1>Your verification Code is : <h2>" + \
            str(randomnumber)+"</h2> </h1> </body></html>"
        sender = {"name": "IBM CUSTOMER CARE REGISTRY",
                  "email": "ibmdemo6@yahoo.com"}
        to = [{"email": e, "name": n}]
        reply_to = {"email": "ibmdemo6@yahoo.com", "name": "IBM"}
        headers = {"Some-Custom-Name": "unique-id-1234"}
        params = {"parameter": "My param value",
                  "subject": "Email Verification"}
        send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(
            to=to, reply_to=reply_to, headers=headers, html_content=html_content, params=params, sender=sender, subject=subject)

        api_response = api_instance.send_transac_email(send_smtp_email)

        pprint(api_response)
        message = "Email send to:"+e+" for password"
        flash(message, "success")

    except ApiException as e:
        print("Exception when calling SMTPApi->send_transac_email: %s\n" % e)
        flash("Error in sending mail")
    except:
        flash("Your didn't Signin with this account")
    finally:
        return render_template('forgot.html')


@app.route('/verifyemail', methods=['POST', 'GET'])
def verifyemail():
    try:
        email = request.form['verifyemail']
        sql = "SELECT ID,NAME FROM Customer WHERE email=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        emailf = ibm_db.fetch_both(stmt)
        while emailf != False:
            id = emailf[0]
            name = emailf[1]
            break
        configuration = sib_api_v3_sdk.Configuration()
        configuration.api_key['api-key'] = ""

        api_instance = sib_api_v3_sdk.TransactionalEmailsApi(
            sib_api_v3_sdk.ApiClient(configuration))
        subject = "Regarding of your Customer Id"
        html_content = "<html><body><h1>Your Customer Id  is : <h2>" + \
            str(id)+"</h2> </h1> </body></html>"
        sender = {"name": "IBM CUSTOMER CARE REGISTRY",
                  "email": "ibmdemo6@yahoo.com"}
        to = [{"email": email, "name": name}]
        reply_to = {"email": "ibmdemo6@yahoo.com", "name": "IBM"}
        headers = {"Some-Custom-Name": "unique-id-1234"}
        params = {"parameter": "My param value",
                  "subject": "Email Verification"}
        send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(
            to=to, reply_to=reply_to, headers=headers, html_content=html_content, params=params, sender=sender, subject=subject)

        api_response = api_instance.send_transac_email(send_smtp_email)

        pprint(api_response)
        message = "Email send to:"+email+" for password"
        flash(message, "success")

    except ApiException as e:
        print("Exception when calling SMTPApi->send_transac_email: %s\n" % e)
        flash("Error in sending mail.")
    except:
        flash("Database not found in mail! Please Register Your account.", "danger")
    finally:
        return render_template('signinpage.html')


@app.route('/otp', methods=['POST', 'GET'])
def otp():
    try:
        otp = request.form['otp']
        cusid = id
        print(id)
        sql = "SELECT PASSWORD FROM Customer WHERE id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, cusid)
        ibm_db.execute(stmt)
        otpf = ibm_db.fetch_both(stmt)
        while otpf != False:
            verify = otpf[0]
            break
        if otp == str(randomnumber):
            msg = "Your Password is "+verify+""
            flash(msg, "success")
```



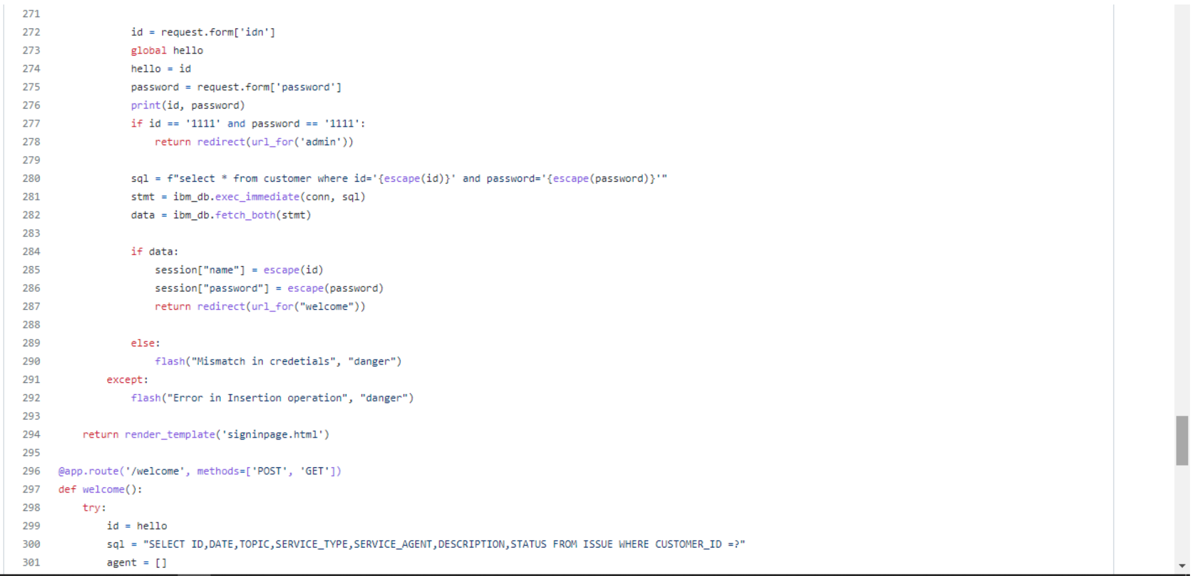
```
            return render_template('forgot.html')
        else:
            flash("Wrong Otp", "danger")
    finally:
        return render_template('forgot.html')


@app.route('/admin', methods=['POST', 'GET'])
def admin():
    userdatabase = []
    sql = "SELECT * FROM customer"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        userdatabase.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    if userdatabase:
        sql = "SELECT COUNT(*) FROM customer;"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    users = []
    sql = "select * from ISSUE"
    stmt = ibm_db.exec_immediate(conn, sql)
    dict = ibm_db.fetch_both(stmt)
    while dict != False:
        users.append(dict)
        dict = ibm_db.fetch_both(stmt)
    if users:
        sql = "SELECT COUNT(*) FROM ISSUE;"
        stmt = ibm_db.exec_immediate(conn, sql)
```



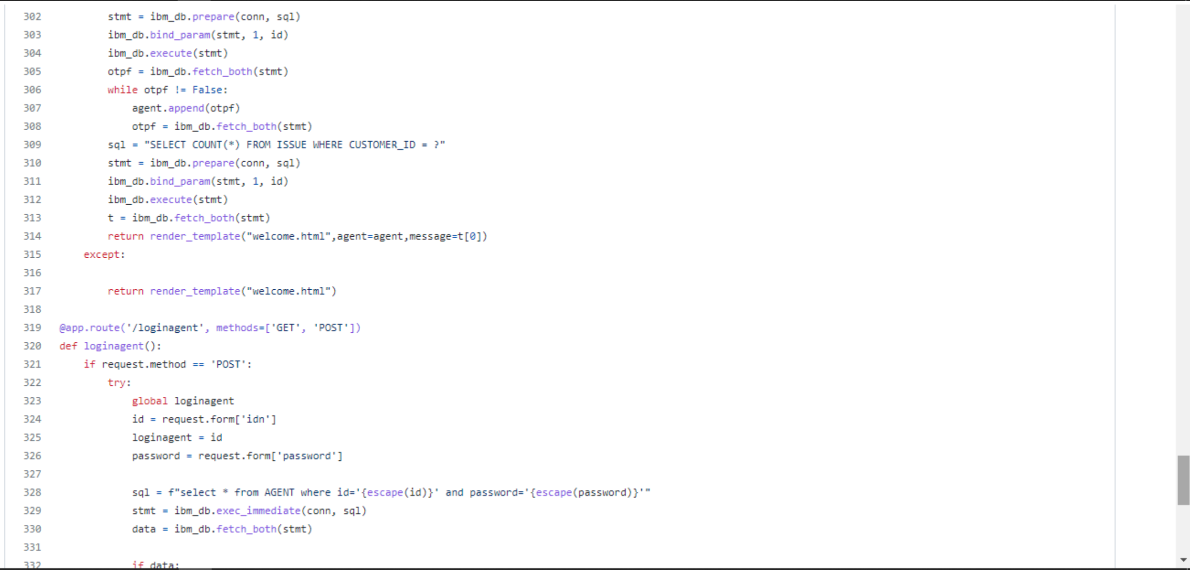
```
        count = ibm_db.fetch_both(stmt)

    agent = []
    sql = "SELECT * FROM AGENT"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        agent.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if agent:
        sql = "SELECT COUNT(*) FROM AGENT;"
        stmt = ibm_db.exec_immediate(conn, sql)
        cot = ibm_db.fetch_both(stmt)

    return render_template("admin.html",complaint=users,users=userdatabase,agents=agent,message=user[0],issue=count[0],msgagent = cot[0])


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

    otp = request.form['otpv']
    if otp == 'C':
        try:
            insert_sql = f"delete from customer"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.execute(prep_stmt)
            flash("delected successfully the Customer", "success")
        except:
```



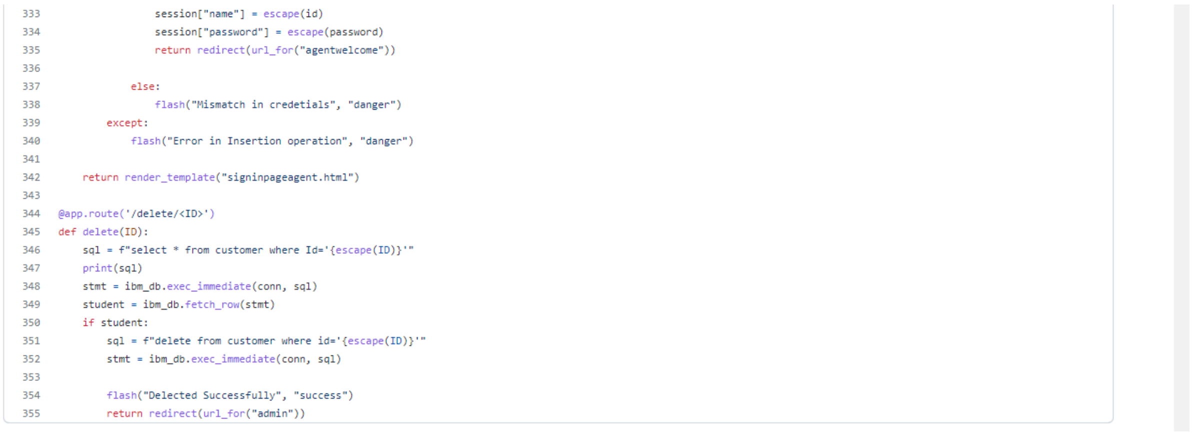
```
            flash("No data found in Customer", "danger")
        finally:
            return redirect(url_for('signuppage'))
    if otp == 'A':
        try:
            insert_sql = f"delete from AGENT"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.execute(prep_stmt)
            flash("delected successfully the Agents", "success")
        except:
            flash("No data found in Agents", "danger")
        finally:
           return redirect(url_for('signuppage'))

    if otp == 'C':
        try:
            insert_sql = f"delete from AGENT"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.execute(prep_stmt)
            flash("delected successfully the Complaints", "success")
        except:
            flash("No data found in Complaints", "danger")
        finally:
            return redirect(url_for('signuppage'))

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        try:
```



```
            id = request.form['idn']
            global hello
            hello = id
            password = request.form['password']
            print(id, password)
            if id == '1111' and password == '1111':
                return redirect(url_for('admin'))

            sql = f"select * from customer where id='{escape(id)}' and password='{escape(password)}'"
            stmt = ibm_db.exec_immediate(conn, sql)
            data = ibm_db.fetch_both(stmt)

            if data:
                session["name"] = escape(id)
                session["password"] = escape(password)
                return redirect(url_for("welcome"))

            else:
                flash("Mismatch in credetials", "danger")
        except:
            flash("Error in Insertion operation", "danger")

    return render_template('signinpage.html')

@app.route('/welcome', methods=['POST', 'GET'])
def welcome():
    try:
        id = hello
        sql = "SELECT ID,DATE,TOPIC,SERVICE_TYPE,SERVICE_AGENT,DESCRIPTION,STATUS FROM ISSUE WHERE CUSTOMER_ID =?"
        agent = []
```



```
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)
        otpf = ibm_db.fetch_both(stmt)
        while otpf != False:
            agent.append(otpf)
            otpf = ibm_db.fetch_both(stmt)
        sql = "SELECT COUNT(*) FROM ISSUE WHERE CUSTOMER_ID = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)
        t = ibm_db.fetch_both(stmt)
        return render_template("welcome.html",agent=agent,message=t[0])
    except:

        return render_template("welcome.html")

@app.route('/loginagent', methods=['GET', 'POST'])
def loginagent():
    if request.method == 'POST':
        try:
            global loginagent
            id = request.form['idn']
            loginagent = id
            password = request.form['password']

            sql = f"select * from AGENT where id='{escape(id)}' and password='{escape(password)}'"
            stmt = ibm_db.exec_immediate(conn, sql)
            data = ibm_db.fetch_both(stmt)

            if data:
```



```
                session["name"] = escape(id)
                session["password"] = escape(password)
                return redirect(url_for("agentwelcome"))

            else:
                flash("Mismatch in credetials", "danger")
        except:
            flash("Error in Insertion operation", "danger")

    return render_template("signinpageagent.html")

@app.route('/delete/<ID>')
def delete(ID):
    sql = f"select * from customer where Id='{escape(ID)}'"
    print(sql)
    stmt = ibm_db.exec_immediate(conn, sql)
    student = ibm_db.fetch_row(stmt)
    if student:
        sql = f"delete from customer where id='{escape(ID)}'"
        stmt = ibm_db.exec_immediate(conn, sql)

        flash("Delected Successfully", "success")
        return redirect(url_for("admin"))
```

css:

```
@import url('https://fonts.googleapis.com/css2?family=Playfair+Display:wght@400;500;600;700&display=swap');


*{
   box-sizing: border-box;
   padding: 0;
   margin: 0;
}
body{
   font-family: 'Playfair Display', serif;
   display: grid;
   background-color: #4158D0;
background-image: linear-gradient (43deg, #4158D0 0%, #C850C0 46%, #FFCC70 100%);


   align-content: center;
   min-height: 100vh;
}
section{
   display: grid;
   grid-template-columns: 1fr 1fr;
   min-height: 70vh;
   width: 75vw;
   margin: 0 auto;
   box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
   border-radius: 12px
}
.image{
   background-color: #12192c;
   display: flex;
   border-radius: 12px 0 0 12px;
}
.image img{
height:50vh;
margin:50px auto
}
```

```css
.content{
   background-color: #12192c;
   display: flex;
   justify-content: center;
   flex-direction: column;
   align-items: center;
   border-radius: 0  12px 12px 0;
   color: #fff;
}
.content h2{
   text-transform: uppercase;
   font-size: 36px;
   letter-spacing: 6px;
   opacity: 0.9;
}
.content span{
   height: 0.5px;
   width: 80px;
   background: #777;
   margin: 30px 0;
}
.content p{
   padding-bottom: 15px;
   font-weight: 300;
   opacity: 0.7;
   width: 60%;
   text-align: center;
   margin: 0 auto;
   line-height: 1.7;
   color:#ffffff
}
.links{
   margin: 15px 0;
}
.links li{
   border: 2px solid #4158D0;
```

```
    list-style: none;
    border-radius: 5px;
    padding: 10px 15px;
    width: 160px;
    text-align: center;
}
.links li a{
    text-transform: uppercase;
    color: #fff;
    text-decoration: none;
}
.links li:hover{
    border-color: #C850C0;
}

.vertical-line{
    height: 30px;
    width: 3px;
    background: #C850C0;
    margin: 0 auto;
}
.icons{
    display: flex;
    padding: 15px 0;
}
.icons li{
    display: block;
    padding: 5px;
    margin: 5px;
}
.icons li i{
    font-size: 26px;
    opacity: 0.8;
}
.icons li i:hover{
    color: #C850C0;
    cursor: pointer;
}



/*****************/

@media(max-width: 900px){
    section{
        grid-template-columns: 1fr;
        width: 100%;
        border-radius: none;
    }
    .image{
        height: 100vh;
        border-radius: none;
    }
    .content{
        height: 100vh;
        border-radius: none;
    }
    .content  h2{
        font-size: 20px;
        margin-top: 50px;
    }
    .content span{
        margin: 20px 0;
    }
    .content p{
        font-size: 14px;
    }
    .links li a{
        font-size: 14px;
    }
    .links{
        margin: 5px 0;
    }
```

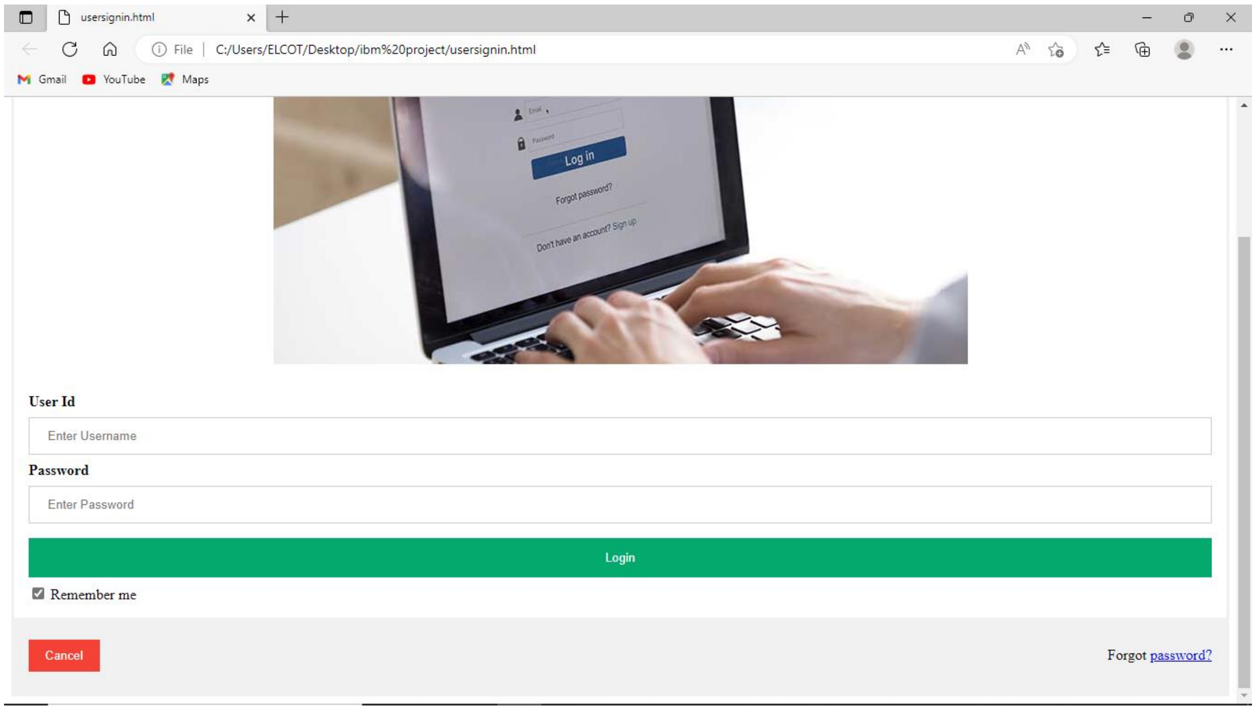
```
  .links li{
      padding: 6px 10px;
  }
  .icons li i{
      font-size: 15px;
  }
}
.credit{
  text-align: center;
  color: #000;
  font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;
 }

 .credit a{
  text-decoration: none;
  color:#000;
  font-weight: bold;
 }
```

CUSTOMER LOGIN.HTML:

```
 <!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://static86.s3.jp-tok.cloud-object-storage.appdomain.cloud/mystyle1.css">
</head>
<body>

<form action="action_page.php" method="post">
  <div class="imgcontainer">
    <img src="https://static86.s3.jp-tok.cloud-object-storage.appdomain.cloud/admin.jpg">
```

```html
  </div>


  <div class="container">
    <label for="uname"><b>User Id</b></label>
    <input type="text" placeholder="Enter Username" name="uname" required>


    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw" required>


    <button type="submit">Login</button>
    <label>
      <input type="checkbox" checked="checked" name="remember"> Remember me
    </label>
  </div>


  <div class="container" style="background-color:#f1f1f1">
    <button type="button" class="cancelbtn">Cancel</button>
    <span class="psw">Forgot <a href="#">password?</a></span>
  </div>
</form>
</body>
</html>
```

CUSTOMER LOGIN.PNG:


usersignin.html
File | C:/Users/ELCOT/Desktop/ibm%20project/usersignin.html
Gmail
YouTube
Maps
Log in
Forgot password?
Don't have an account? Sign up

User Id

Enter Username

Password

Enter Password

Login

☑ Remember me

Cancel

Forgot password?