SOURCE CODE

```
from flask import Flask, render_template, flash, request, session,send_file
from flask import render_template, redirect, url_for, request

import ibm_db
import pandas
import ibm_db_dbi
from sqlalchemy import create_engine

engine = create_engine('sqlite://',
                       echo = False)

dsn_hostname = "b70af05b-76e4-4bca-a1f5-
23dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_uid = "crc83247"
dsn_pwd = "eHGBftxhodLDnNpM"

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"
dsn_port = "32716"
dsn_protocol = "TCPIP"
dsn_security = "SSL"

dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"
    "SECURITY={7};").format(dsn_driver, dsn_database, dsn_hostname, dsn_port,
dsn_protocol, dsn_uid, dsn_pwd,dsn_security)
```

```
try:
    conn = ibm_db.connect(dsn, "", "")
    print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ",
dsn_hostname)

except:
    print ("Unable to connect: ", ibm_db.conn_errormsg() )



app = Flask(__name__)
app.config['DEBUG']
app.config['SECRET_KEY'] = '7d441f27d441f27567d441f2b6176a'

@app.route("/")
def homepage():

    return render_template('index.html')

@app.route("/AdminLogin")
def AdminLogin():

    return render_template('AdminLogin.html')

@app.route("/UserLogin")
def UserLogin():
    return render_template('UserLogin.html')



@app.route("/NewUser")
def NewUser():
    return render_template('NewUser.html')
```

```
@app.route("/NewComplaint")
def NewComplaint():
    user = session['uname']
    return render_template('NewComplaint.html',uname=user)

@app.route("/NewAgent")
def NewAgent():



    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM agenttb where  "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()

    return render_template('NewAgent.html',data=data)




@app.route("/AdminHome")
def AdminHome():
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * from regtb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('Employee_Data',
                     con=engine,
                     if_exists='append')

    # run a sql query
```

```
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()
    return render_template('AdminHome.html',data=data)


@app.route("/UserHome")
def UserHome():
    user = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM regtb where  UserName= '" + user + "' "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()
    return render_template('UserHome.html',data=data)


@app.route("/UserComplaint")
def UserComplaint():
    user = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM booktb where  UserName= '" + user + "' "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()

    return render_template('UserComplaint.html',data=data)

@app.route("/AdminComplaintInfo")
def AdminComplaintInfo():
```

```
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM booktb    "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()

    return render_template('AdminComplaintInfo.html',data=data)

@app.route("/adminlogin", methods=['GET', 'POST'])
def adminlogin():
    error = None
    if request.method == 'POST':
      if request.form['uname'] == 'admin' or request.form['password'] == 'admin':

          conn = ibm_db.connect(dsn, "", "")
          pd_conn = ibm_db_dbi.Connection(conn)

          selectQuery = "SELECT * from regtb "
          dataframe = pandas.read_sql(selectQuery, pd_conn)

          dataframe.to_sql('Employee_Data',
                           con=engine,
                           if_exists='append')

          # run a sql query
          data = engine.execute("SELECT * FROM Employee_Data").fetchall()
          return render_template('AdminHome.html' , data=data)

      else:
       return render_template('index.html', error=error)

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

```
    if request.method == 'POST':
        username = request.form['uname']
        password = request.form['password']
        session['uname'] = request.form['uname']

        conn = ibm_db.connect(dsn, "", "")
        pd_conn = ibm_db_dbi.Connection(conn)

        selectQuery = "SELECT * from regtb where UserName='" + username + "' and
password='" + password + "'"
        dataframe = pandas.read_sql(selectQuery, pd_conn)

        if dataframe.empty:
            data1 = 'Username or Password is wrong'
            return render_template('goback.html', data=data1)
        else:
            print("Login")
            selectQuery = "SELECT * from regtb where UserName='" + username + "' and
password='" + password + "'"
            dataframe = pandas.read_sql(selectQuery, pd_conn)

            dataframe.to_sql('Employee_Data',
                             con=engine,
                             if_exists='append')

            # run a sql query
            data = engine.execute("SELECT * FROM Employee_Data").fetchall()

            return render_template('UserHome.html', data=data )

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

```
        name1 = request.form['name']
        gender1 = request.form['gender']
        Age = request.form['age']
        email = request.form['email']
        pnumber = request.form['phone']
        address = request.form['address']

        uname = request.form['uname']
        password = request.form['psw']



        conn = ibm_db.connect(dsn, "", "")

        insertQuery = "INSERT INTO regtb VALUES ('" + name1 + "','" + gender1 + "','" +
Age + "','" + email + "','" + pnumber + "','" + address + "','" + uname + "','" + password + "')"
        insert_table = ibm_db.exec_immediate(conn, insertQuery)
        print(insert_table)



    return render_template('UserLogin.html')



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

        name1 = request.form['name']
        gender1 = request.form['gender']
        Age = request.form['age']
        email = request.form['email']
        pnumber = request.form['phone']
        address = request.form['address']
```

```
        uname = request.form['uname']



        conn = ibm_db.connect(dsn, "", "")
        pd_conn = ibm_db_dbi.Connection(conn)

        insertQuery = "INSERT INTO agenttb VALUES ('" + name1 + "','" + gender1 + "','" +
Age + "','" + email + "','" + pnumber + "','" + address + "','" + uname + "')"
        insert_table = ibm_db.exec_immediate(conn, insertQuery)
        print(insert_table)

        selectQuery = "SELECT * FROM agenttb  "
        dataframe = pandas.read_sql(selectQuery, pd_conn)
        dataframe.to_sql('booktb1', con=engine, if_exists='append')
        data = engine.execute("SELECT * FROM booktb1").fetchall()




    return render_template('NewAgent.html',data=data)




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

        name = request.form['name']
        com = request.form['com']
        uname = session['uname']
```

```
        conn = ibm_db.connect(dsn, "", "")
        pd_conn = ibm_db_dbi.Connection(conn)

        selectQuery = "SELECT  *  FROM  booktb"
        dataframe = pandas.read_sql(selectQuery, pd_conn)

        dataframe.to_sql('booktb', con=engine, if_exists='append')
        data2 = engine.execute("SELECT * FROM booktb").fetchall()
        count = 0

        for item in data2:
            count += 1

        Bookingid = "COMID00" + str(count)

        insertQuery = "INSERT INTO booktb VALUES ('" + Bookingid + "','" + uname + "','" +
com + "','")"
        insert_table = ibm_db.exec_immediate(conn, insertQuery)
        print(insert_table)



        selectQuery = "SELECT * FROM booktb where  UserName= '" + uname + "' "
        dataframe = pandas.read_sql(selectQuery, pd_conn)

        dataframe.to_sql('booktb1', con=engine, if_exists='append')
        data = engine.execute("SELECT * FROM booktb1").fetchall()
        return render_template('UserComplaint.html', data=data)

@app.route("/AgentAssign", methods=['GET'])
def AgentAssign():
    cid = request.args.get('id')
    session['cid'] = cid

    conn = ibm_db.connect(dsn, "", "")
```

```
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery = "SELECT * FROM agenttb  "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

    dataframe.to_sql('booktb1', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM booktb1").fetchall()

    return render_template('AgentAssign.html',data=data)



@app.route("/Action", methods=['GET'])
def Action():
    cid = request.args.get('id')
    session['cid'] = cid

    return render_template('Action.html')

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

    agid = request.form['agid']

    cid = session['cid']

    uname = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery1 = "SELECT  *  FROM  regtb where  UserName='" + uname + "'"
    dataframe = pandas.read_sql(selectQuery1, pd_conn)

    dataframe.to_sql('regtb', con=engine, if_exists='append')
```

```
    data1 = engine.execute("SELECT * FROM regtb").fetchall()

    for item1 in data1:
        Mobile = item1[5]
        Email = item1[4]
        sendmsg(Email,"Assign Agent id"+agid)

    insertQuery = "update booktb set AgentName='"+ agid +"' where ComplaintId='"+ cid +"'
"
    insert_table = ibm_db.exec_immediate(conn, insertQuery)

    alert = 'Agent Assign Send Notication'

    return render_template('goback.html', data=alert)



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

    com = request.form['com']

    cid = session['cid']

    uname = session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)

    selectQuery1 = "SELECT  *  FROM  regtb where  UserName='" + uname + "'"
    dataframe = pandas.read_sql(selectQuery1, pd_conn)

    dataframe.to_sql('regtb', con=engine, if_exists='append')
    data1 = engine.execute("SELECT * FROM regtb").fetchall()
```

```
    for item1 in data1:
        Mobile = item1[5]
        Email = item1[4]
        sendmsg(Email,"Action Information "+com)

    insertQuery = "update booktb set ACTIONINFO="'+ com +"' where ComplaintId="'+ cid
+"' "
    insert_table = ibm_db.exec_immediate(conn, insertQuery)

    alert = 'Action Info Saved Send Notication'

    return render_template('goback.html', data=alert)

def sendmsg(Mailid,message):
    import smtplib
    from email.mime.multipart import MIMEMultipart
    from email.mime.text import MIMEText
    from email.mime.base import MIMEBase
    from email import encoders

    fromaddr = "sampletest685@gmail.com"
    toaddr = Mailid

    # instance of MIMEMultipart
    msg = MIMEMultipart()

    # storing the senders email address
    msg['From'] = fromaddr

    # storing the receivers email address
    msg['To'] = toaddr

    # storing the subject
    msg['Subject'] = "Alert"
```

```
    # string to store the body of the mail
    body = message

    # attach the body with the msg instance
    msg.attach(MIMEText(body, 'plain'))

    # creates SMTP session
    s = smtplib.SMTP('smtp.gmail.com', 587)

    # start TLS for security
    s.starttls()

    # Authentication
    s.login(fromaddr, "hneucvnontsuwgpj")

    # Converts the Multipart msg into a string
    text = msg.as_string()

    # sending the mail
    s.sendmail(fromaddr, toaddr, text)

    # terminating the session
    s.quit()

if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)
```