TEAM ID:PNT2022TMID30600

TITLE: SKILL AND JOB TRECOMMENDER

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

import json
from json2html import *
import requests

import ibm_db
import pandas
import ibm_db_dbi
from sqlalchemy import create_engine

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

dsn_hostname = "125f9f61-9715-46f9-9399-
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"
dsn_uid = "sxr79922"
dsn_pwd = "C06C0zBL1IYQLze2"

dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "BLUDB"
dsn_port = "30426"
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'
```

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

    return render_template('index.html')




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

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

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

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

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

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


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



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




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

```
    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * from companytb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

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


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



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


    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * from jobtb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

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

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


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


    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * from jobtb "
    dataframe = pandas.read_sql(selectQuery, pd_conn)

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

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

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


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

    uname=  session['uname']

    conn = ibm_db.connect(dsn, "", "")
    pd_conn = ibm_db_dbi.Connection(conn)
    selectQuery = "SELECT * FROM regtb where Username='"+ uname +"' "
    dataframe = pandas.read_sql(selectQuery, pd_conn)
    dataframe.to_sql('Employee_Data', con=engine, if_exists='append')
    data = engine.execute("SELECT * FROM Employee_Data").fetchall()
```

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




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


    cname=  session['cname']


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



    return render_template('CJobInfo.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')
           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():
    error = None
    if request.method == 'POST':

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


        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
            print(engine.execute("SELECT * FROM Employee_Data").fetchall())

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



@app.route("/companylogin", methods=['GET', 'POST'])
def companylogin():
    error = None
    if request.method == 'POST':

        uname = request.form['uname']
        password = request.form['password']
        session['cname'] = uname


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

        pd_conn = ibm_db_dbi.Connection(conn)

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

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

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

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

@app.route("/NewStudent1", methods=['GET', 'POST'])
def NewStudent1():
     if request.method == 'POST':

          name = request.form['name']
          gender = request.form['gender']
          Age = request.form['Age']
          email = request.form['email']
          pnumber = request.form['pnumber']
          address = request.form['address']
          Degree = request.form['Degree']
          depat = request.form['depat']
          uname = request.form['uname']
          passw = request.form['passw']

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

     insertQuery = "insert into regtb values('" + name + "','" + gender + "','" +
Age + "','" + email + "','" + pnumber + "','" + address + "','" + Degree + "','" +
depat + "','" + uname + "','" + passw + "')"
     insert_table = ibm_db.exec_immediate(conn, insertQuery)

     sendmsg(email, "Successfully registered this website")

     data1 = 'Record Saved!'
     return render_template('goback.html', data=data1)




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

          cname = request.form['cname']
          regno = request.form['regno']
          mobile = request.form['mobile']

          email = request.form['email']
          Website = request.form['Website']
          address = request.form['address']
          uname = request.form['uname']
          passw = request.form['passw']

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

          insertQuery = "insert into companytb
values('"+cname+"','"+regno+"','"+mobile+"','"+email+"','"+Website+"','"+address+"
','"+uname +"','"+passw+"')"
          insert_table = ibm_db.exec_immediate(conn, insertQuery)

          data1 = 'Record Saved!'
          return render_template('goback.html', data=data1)
```

```
@app.route("/newjob", methods=['GET', 'POST'])
def newjob():
     if request.method == 'POST':
         cnn = session['cname']
         cname = request.form['cname']
         cno = request.form['cno']
         Address = request.form['Address']
         JobLocation = request.form['JobLocation']
         Vacancy = request.form['Vacancy']
         Job = request.form['Job']
         Department = request.form['depat']
         website = request.form['website']

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

         insertQuery = "insert into jobtb values('" + cname + "','" + cno + "','"
+ Address + "','" + JobLocation + "','" + Vacancy + "','" + Job + "','" +
Department + "','" + website + "','"+cnn+"')"
         insert_table = ibm_db.exec_immediate(conn, insertQuery)

         conn = ibm_db.connect(dsn, "", "")
         pd_conn = ibm_db_dbi.Connection(conn)
         selectQuery1 = "SELECT  *  FROM  regtb where  Department='" + Department
+ "'"
         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,"Jop Title"+Job + " More Info Visit Website")




     data = 'Record Saved!'
     return render_template("goback.html", data=data)




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


        url = "https://linkedin-jobs-search.p.rapidapi.com/"

        payload = {
```

```
            "search_terms": jobname,
            "location": "india",
            "page": "1"
        }
        headers = {
            "content-type": "application/json",
            "X-RapidAPI-Key":
"b045b9af95msha8d7c3160785729p1674cdjsnbdf4adbf9868",
            "X-RapidAPI-Host": "linkedin-jobs-search.p.rapidapi.com"
        }

        response = requests.request("POST", url, json=payload, headers=headers)

        print(response.text)

        infoFromJson = json.loads(response.text)

        df = pandas.json_normalize(infoFromJson)

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



        return render_template('Search.html',data=data1)









#send grid

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




if __name__ == '__main__':
    app.run(host='0.0.0.0', debug='TRUE')
```

DEPLOYMENT:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ibmjob
  labels:
    app: ibmjob
spec:
  replicas: 3
  selector:
    matchLabels:
      app: ibmjob
  template:
    metadata:
      labels:
        app: ibmjob
    spec:
      containers:
      - name: ibmjob
        image: de.icr.io/ibmjob/app
        ports:
        - containerPort: 5000
```

DATA:

```
CREATE TABLE companytb (
  CompanyName varchar(250) NOT NULL,
```

```
  Regno varchar(250) NOT NULL,
  Mobile varchar(250) NOT NULL,
  Email varchar(250) NOT NULL,
  Website varchar(250) NOT NULL,
  Address varchar(500) NOT NULL,
  Username varchar(250) NOT NULL,
  Password varchar(250) NOT NULL
  )



  CREATE TABLE jobtb (
  CompanyName varchar(250) NOT NULL,
  ContactNo varchar(250) NOT NULL,
  Address varchar(250) NOT NULL,
  Location varchar(250) NOT NULL,
  Vacancy varchar(250) NOT NULL,
  Job varchar(250) NOT NULL,
  Department varchar(250) NOT NULL,
  website varchar(250) NOT NULL,
  Cname varchar(250) NOT NULL
)



CREATE TABLE regtb (
  Name varchar(250) NOT NULL,
  Gender varchar(250) NOT NULL,
  Age varchar(250) NOT NULL,
  Email varchar(250) NOT NULL,
  Phone varchar(250) NOT NULL,
  Address varchar(250) NOT NULL,
  Degree varchar(250) NOT NULL,
  Department varchar(250) NOT NULL,
  UserName varchar(250) NOT NULL,
  Password varchar(250) NOT NULL
)
```

SERVICE:

```
apiVersion: v1
kind: Service
metadata:
  name: ibmjob
spec:
  selector:
    app.kubernetes.io/name: ibmjob
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
```

DOCKER:

```
FROM python:3.7
RUN apt-get update
RUN mkdir /app
```

```
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
EXPOSE 5000
ENTRYPOINT [ "python" ]
CMD [ "app.py" ]
```