

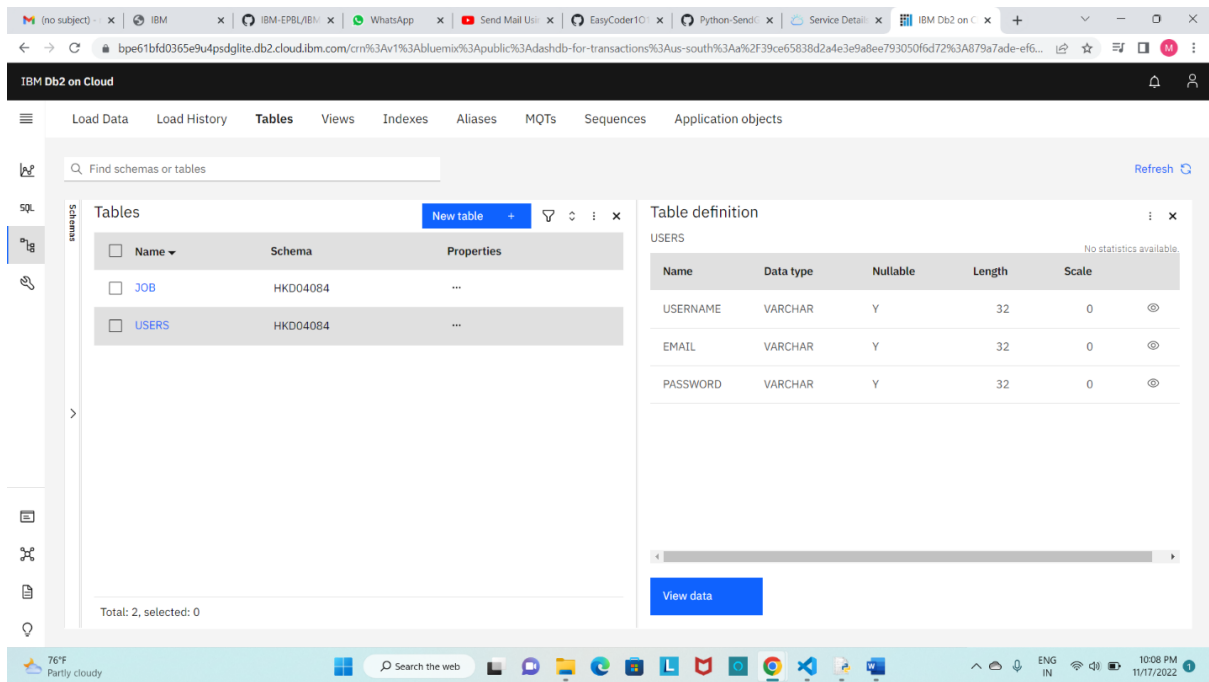
## PROJECT DEVELOPMENT PHASE – SPRINT 4

Date	19 November 2022
Team ID	PNT2022TMID34565
Project Name	Project – Skill/Job Recommender

### DATABASE:

The screenshot shows the IBM Db2 on Cloud console interface. The top navigation bar includes tabs for Load Data, Load History, Tables, Views, Indexes, Aliases, MQTs, Sequences, and Application objects. The main content area is divided into two panels. The left panel, titled 'Schemas', displays a table with columns 'Name', 'Type', and 'Tables'. It shows one schema named 'HKD04084' of type 'User' with 2 tables. The right panel, titled 'Tables', displays a table with columns 'Name', 'Schema', and 'Properties'. It shows two tables: 'JOB' and 'USERS', both belonging to the 'HKD04084' schema. A 'New table' button is visible in the top right of the Tables panel. The bottom status bar shows the temperature as 76°F and the time as 10:08 PM on 11/17/2022.

The screenshot shows the IBM Db2 on Cloud console interface, specifically the 'Table definition' view for the 'JOB' table. The left panel, titled 'Tables', displays a table with columns 'Name', 'Schema', and 'Properties'. It shows two tables: 'JOB' and 'USERS', both belonging to the 'HKD04084' schema. The right panel, titled 'Table definition', displays the structure of the 'JOB' table. It includes a table with columns 'Name', 'Data type', 'Nullable', 'Length', and 'Scale'. The table definition shows the following columns: 'USERNAME' (VARCHAR, Y, 32, 0), 'EMAIL' (VARCHAR, Y, 32, 0), 'QUALIFICATION' (VARCHAR, Y, 32, 0), 'SKILLS' (VARCHAR, Y, 32, 0), and 'JOBS' (VARCHAR, Y, 32, 0). A 'View data' button is visible at the bottom of the Table definition panel. The bottom status bar shows the temperature as 76°F and the time as 10:08 PM on 11/17/2022.



## App.py:

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

```
import ibm_db
```

```
import re
```

```
app = Flask(__name__)
```

```
app.secret_key = 'a'
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=2f3279a5-73d1-4859-88f0-
a6c3e6b4b907.c3n41cmd0nqnkrk39u98g.databases.appdomain.cloud;PORT=30756;SECURITY=SSL;SS
LServerCertificate=DigiCertGlobalRootCA.crt;UID=hkd04084;PWD=Bwjcq4p5AL1sL0FQ", "")
```

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

```
def homer():
```

```
return render_template('login.html')
```

```
@app.route('/login', methods=['GET', 'POST'])
```

```
def login():
```

```
    global userid
```

```
    msg = "
```

```
    if request.method == 'POST' :
```

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

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

```
        sql = "SELECT * FROM users WHERE username=? AND password=?"
```

```

stmt = ibm_db.prepare(conn, sql)

ibm_db.bind_param(stmt,1,username)

ibm_db.bind_param(stmt,2,password)

ibm_db.execute(stmt)

account = ibm_db.fetch_assoc(stmt)

print (account)

if account:

    session['loggedin'] = True

    session['id'] = account['USERNAME']

    userid= account['USERNAME']

    session['username'] = account['USERNAME']

    msg = 'Logged in successfully !'

    msg = 'Logged in successfully !'

    return render_template('dashboard.html', msg = msg)

else:

    msg = 'Incorrect username / password !'

return render_template('login.html', msg = msg)

@app.route('/register', methods=['GET', 'POST'])

def registet():

    msg = ""

    if request.method == 'POST' :

        username = request.form['username']

        email = request.form['email']

        password = request.form['password']

        sql = "SELECT * FROM users WHERE username =?"

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt,1,username)

        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        print(account)

        if account:

```

```

        msg = 'Account already exists !'
elif not re.match(r'^[@]+\.[^@]+', email):
    msg = 'Invalid email address !'
elif not re.match(r'[A-Za-z0-9]+', username):
    msg = 'name must contain only characters and numbers !'
else:
    insert_sql = "INSERT INTO users VALUES (?, ?, ?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, username)
    ibm_db.bind_param(prepare_stmt, 2, email)
    ibm_db.bind_param(prepare_stmt, 3, password)
    ibm_db.execute(prepare_stmt)

    msg = 'You have successfully registered !'

elif request.method == 'POST':
    msg = 'Please fill out the form !'

return render_template('register.html', msg = msg)

@app.route('/dashboard')
def dash():
    return render_template('dashboard.html')

@app.route('/apply', methods = ['GET', 'POST'])
def apply():
    msg = ""
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        qualification= request.form['qualification']
        skills = request.form['skills']
        jobs = request.form['s']
        sql = "SELECT * FROM users WHERE username =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)

```

```

ibm_db.execute(stmt)

account = ibm_db.fetch_assoc(stmt)

print(account)

if account:

    msg = 'there is only 1 job position! for you'

    return render_template('apply.html', msg = msg)

insert_sql = "INSERT INTO job VALUES (?, ?, ?, ?, ?)"

prep_stmt = ibm_db.prepare(conn, insert_sql)

ibm_db.bind_param(prepare_stmt, 1, username)

ibm_db.bind_param(prepare_stmt, 2, email)

ibm_db.bind_param(prepare_stmt, 3, qualification)

ibm_db.bind_param(prepare_stmt, 4, skills)

ibm_db.bind_param(prepare_stmt, 5, jobs)

ibm_db.execute(prepare_stmt)

msg = 'You have successfully applied for job !'

session['loggedin'] = True

TEXT = "Hello sandeep,a new appliaction for job position" +jobs+"is requested"

#sendmail(TEXT,"sandeep@thesmartbridge.com")

sendgridmail("sandeep@thesmartbridge.com",TEXT)

elif request.method == 'POST':

    msg = 'Please fill out the form !'

    return render_template('apply.html', msg = msg)

@app.route('/display')

def display():

    print(session["username"],session['id'])


    cursor = mysql.connection.cursor()

    cursor.execute('SELECT * FROM job WHERE userid = % s', (session['id'],))

    account = cursor.fetchone()

    print("accountdislay",account)

    return render_template('display.html',account = account)

```

```

@app.route('/logout')

def logout():

    session.pop('loggedin', None)

    session.pop('id', None)

    session.pop('username', None)

    return render_template('home.html')

if __name__ == '__main__':

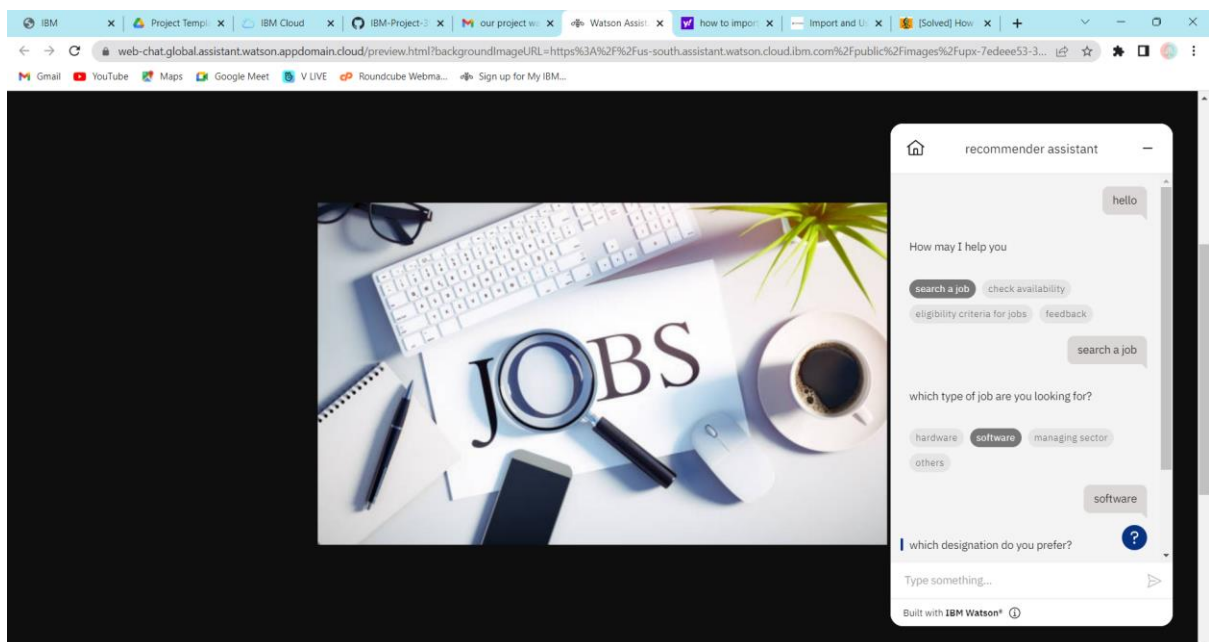
    app.run(host='0.0.0.0')

```

## **CHATBOT:**

Chatbot url:

<https://web-chat.global.assistant.watson.appdomain.cloud/preview.html?backgroundImageUrl=https%3A%2F%2Fus-south.assistant.watson.cloud.ibm.com%2Fpublic%2Fimages%2Fupx-7edeee53-3201-4ab0-afc5-e8482dcaab3f%3A%3Ab279a216-7162-411f-9210-ee1ce0532bd9&integrationID=b9577b7a-aa8a-4fb6-bc1e-d760ebef295&region=us-south&serviceInstanceID=7edeee53-3201-4ab0-afc5-e8482dcaab3f>





recommender assistant

search a job check availability  
eligibility criteria for jobs feedback

search a job

which type of job are you looking for?

hardware software managing sector  
others

software

which designation do you prefer?

developer testing web designer  
others

Type something...

Built with IBM Watson®