## **SPRINT-4**

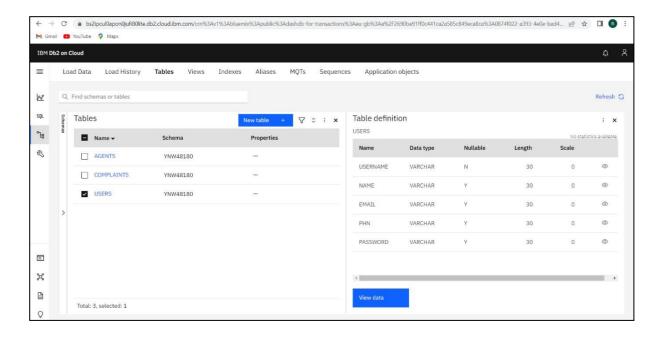
TEAM ID: PNT2022TMID50436

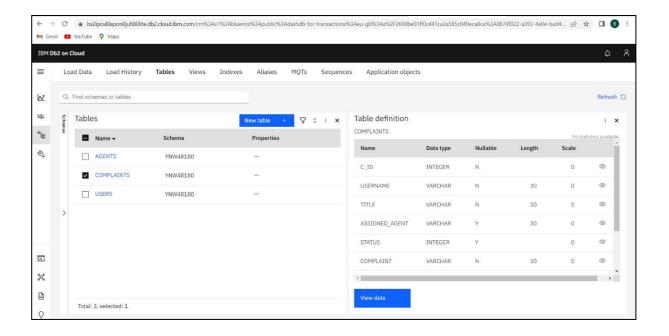
## **TEAM MEMBERS:**

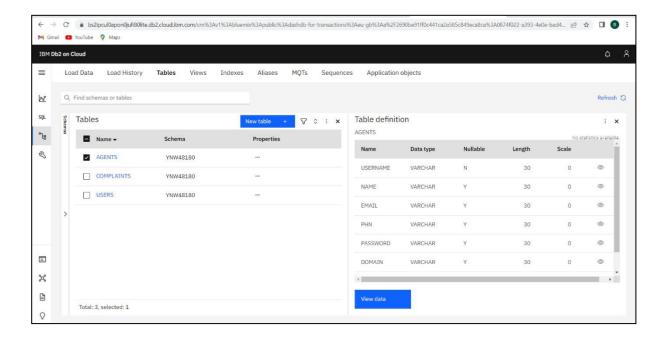
- 1. K.THANGA ESAKKIAMMAL (Team Leader)
- 2. S.SAKTHESWARI
- 3. T.DHANUSHIYA
- 4. G.MAHESWARI

## **SPRINT - 4**

## **DB CONNECTION AND TABLE CREATION**







```
App.py
from flask import Flask, render_template, request, redirect, session, url_for
import ibm_db
import re
app = Flask(__name___)
# for connection
# conn= ""
app.secret_key = 'a'
print("Trying to connect...")
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=55fbc997-9266-4331-afd3-
888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31929;SECURITY=
SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=ynw48180;PWD=I1EGQPDz745BoBj
p;", ", ")
print("connected..")
@app.route('/signup', methods=['GET', 'POST'])
def signup():
 global userid
 msg = "
 if request.method == 'POST':
    username = request.form['username']
    name = request.form['name']
    email = request.form['email']
    phn = request.form['phn']
```

CODE:

```
password = request.form['pass']
    repass = request.form['repass']
    print("inside checking")
    print(name)
    if len(username) == 0 or len(name) == 0 or len(email) == 0 or len(phn) == 0 or
len(password) == 0 or len(repass) == 0:
      msg = "Form is not filled completely!!"
      print(msg)
      return render_template('signup.html', msg=msg)
    elif password != repass:
      msg = "Password is not matched"
      print(msg)
      return render template('signup.html', msg=msg)
    elif not re.match(r'[a-z]+', username):
      msg = 'Username can contain only small letters and numbers'
      print(msg)
      return render template('signup.html', msg=msg)
    elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
      msg = 'Invalid email'
      print(msg)
      return render_template('signup.html', msg=msg)
    elif not re.match(r'[A-Za-z]+', name):
      msg = "Enter valid name"
      print(msg)
      return render template('signup.html', msg=msg)
    elif not re.match(r'[0-9]+', phn):
      msg = "Enter valid phone number"
      print(msg)
      return render_template('signup.html', msg=msg)
```

```
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 = 'Acccount already exists'
    else:
      userid = username
      insert_sql = "insert into users values(?,?,?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, username)
      ibm db.bind param(prep stmt, 2, name)
      ibm_db.bind_param(prep_stmt, 3, email)
      ibm_db.bind_param(prep_stmt, 4, phn)
      ibm_db.bind_param(prep_stmt, 5, password)
      ibm_db.execute(prep_stmt)
      print("successs")
      msg = "succesfully signed up"
    return render template('dashboard.html', msg=msg, name=name)
  else:
    return render template('signup.html')
@app.route('/dashboard')
def dashboard():
  return render_template('dashboard.html')
```

```
@app.route('/')
def base():
 return redirect(url_for('login'))
@app.route('/login', methods=["GET", "POST"])
def login():
 global userid
  msg = "
 if request.method == 'POST':
    username = request.form['username']
    userid = username
    password = request.form['pass']
    if userid == 'admin' and password == 'admin':
      print("its admin")
      return render template('admin.html')
    else:
      sql = "select * from agents 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'
```

```
sql = "select * from complaints where assigned agent = ?"
        complaints = []
        stmt = ibm db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        dictionary = ibm_db.fetch_assoc(stmt)
        while dictionary != False:
          complaints.append(dictionary)
          dictionary = ibm db.fetch assoc(stmt)
        print(complaints)
        return render_template('agentdash.html', name=account['USERNAME'],
complaints=complaints)
    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'
      # for getting complaints details
```

# for getting complaints details

```
sql = "select * from complaints where username = ?"
      complaints = []
      stmt = ibm_db.prepare(conn, sql)
      ibm db.bind param(stmt, 1, username)
      ibm db.execute(stmt)
      dictionary = ibm_db.fetch_assoc(stmt)
      while dictionary != False:
        # print "The ID is : ", dictionary["EMPNO"]
        # print "The Name is : ", dictionary[1]
        complaints.append(dictionary)
        dictionary = ibm_db.fetch_assoc(stmt)
      print(complaints)
      return render_template('dashboard.html', name=account['USERNAME'],
complaints=complaints)
    else:
      msg = 'Incorrect user credentials'
      return render template('dashboard.html', msg=msg)
  else:
    return render template('login.html')
@app.route('/addnew', methods=["GET", "POST"])
def add():
  if request.method == 'POST':
    title = request.form['title']
    des = request.form['des']
    try:
      sql = "insert into complaints(username,title,complaint) values(?,?,?)"
      stmt = ibm_db.prepare(conn, sql)
```

```
ibm_db.bind_param(stmt, 1, userid)
      ibm_db.bind_param(stmt, 2, title)
      ibm_db.bind_param(stmt, 3, des)
      ibm db.execute(stmt)
    except:
      print(userid)
      print(title)
      print(des)
      print("cant insert")
    sql = "select * from complaints where username = ?"
    complaints = []
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm db.execute(stmt)
    dictionary = ibm_db.fetch_assoc(stmt)
    while dictionary != False:
      # print "The ID is : ", dictionary["EMPNO"]
      # print "The Name is: ", dictionary[1]
      complaints.append(dictionary)
      dictionary = ibm_db.fetch_assoc(stmt)
    print(complaints)
    return render_template('dashboard.html', name=userid, complaints=complaints)
@app.route('/agents')
def agents():
 sql = "select * from agents"
  agents = []
  stmt = ibm_db.prepare(conn, sql)
```

```
ibm_db.execute(stmt)
  dictionary = ibm_db.fetch_assoc(stmt)
  while dictionary != False:
    agents.append(dictionary)
    dictionary = ibm_db.fetch_assoc(stmt)
  return render_template('agents.html', agents=agents)
@app.route('/addnewagent', methods=["GET", "POST"])
def addagent():
  if request.method == 'POST':
    username = request.form['username']
    name = request.form['name']
    email = request.form['email']
    phone = request.form['phone']
    domain = request.form['domain']
    password = request.form['password']
    try:
      sql = "insert into agents values(?,?,?,?,?,2)"
      stmt = ibm db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, username)
      ibm_db.bind_param(stmt, 2, name)
      ibm_db.bind_param(stmt, 3, email)
      ibm_db.bind_param(stmt, 4, phone)
      ibm_db.bind_param(stmt, 5, password)
      ibm db.bind param(stmt, 6, domain)
      ibm_db.execute(stmt)
    except:
      print("cant insert")
```

```
sql = "select * from agents"
    agents = []
    stmt = ibm db.prepare(conn, sql)
    ibm db.execute(stmt)
    dictionary = ibm db.fetch assoc(stmt)
    while dictionary != False:
      agents.append(dictionary)
      dictionary = ibm_db.fetch_assoc(stmt)
    return render_template('agents.html', agents=agents)
@app.route('/updatecomplaint', methods=["GET", "POST"])
def updatecomplaint():
  if request.method == 'POST':
    cid = request.form['cid']
    solution = request.form['solution']
    try:
      sql = "update complaints set solution =?,status=1 where c_id = ? and
assigned agent=?"
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, solution)
      ibm_db.bind_param(stmt, 2, cid)
      ibm db.bind param(stmt, 3, userid)
      ibm_db.execute(stmt)
      sql = "update agents set status = 3 where username=?"
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, userid)
      ibm_db.execute(stmt)
    except:
```

```
print("cant insert")
    sql = "select * from complaints where assigned_agent = ?"
    complaints = []
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_assoc(stmt)
    while dictionary != False:
      complaints.append(dictionary)
      dictionary = ibm_db.fetch_assoc(stmt)
    # print(complaints)
    return render_template('agentdash.html', name=userid, complaints=complaints)
@app.route('/tickets')
def tickets():
  sql = "select * from complaints"
 complaints = []
  stmt = ibm_db.prepare(conn, sql)
  ibm_db.execute(stmt)
  dictionary = ibm_db.fetch_assoc(stmt)
  while dictionary != False:
    complaints.append(dictionary)
    dictionary = ibm_db.fetch_assoc(stmt)
 sql = "select username from agents where status <> 1"
 freeagents = []
 stmt = ibm_db.prepare(conn, sql)
  ibm_db.execute(stmt)
```

```
dictionary = ibm_db.fetch_assoc(stmt)
  while dictionary != False:
    freeagents.append(dictionary)
    dictionary = ibm db.fetch assoc(stmt)
  print(freeagents)
  return render_template('tickets.html', complaints=complaints, freeagents=freeagents)
@app.route('/assignagent', methods=['GET', 'POST'])
def assignagent():
  if request.method == "POST":
    ccid = request.form['ccid']
    agent = request.form['agent']
    print(ccid)
    print(agent)
    try:
      sql = "update complaints set assigned_agent =? where c_id = ?"
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, agent)
      ibm db.bind param(stmt, 2, ccid)
      ibm_db.execute(stmt)
      sql = "update agents set status =1 where username = ?"
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, userid)
      ibm_db.execute(stmt)
    except:
      print("cant update")
    return redirect(url_for('tickets'))
```

