

## Assignment-2

<b>Project Domain</b>	Cloud Application Development
<b>Project Title</b>	News Tracker Application
<b>Team ID</b>	PNT2022TMID44401
<b>Name</b>	KISHORE S
<b>Roll No</b>	731119205016
<b>Date</b>	03rd Oct 2022

### Questions:

1. Create registration page in html with username, email, and phone number and by using POST method display it in next html page.
2. Develop a flask program which should contain at least 5 packages used from pypi.org.
3. Create User table with user with email, username, roll number, password.
4. Perform UPDATE, DELETE Queries with user table
5. Connect python code to db2.
6. Create a flask app with registration page, login page and welcome page. By default, load the registration page once the user enters all the fields store the data in database and navigate to login page authenticate user username and password. If the user is valid show the welcome page.

---

### Answers:

1. Create registration page in html with username, email, and phone number and by using POST method display it in next html page.

#### Login.html:

```
<html>
<body>
  <center>
    <form action = "http://localhost:3890/login" method = "post">
      <h1>
        Enter user Name:<input type = "text" name = "userName"/><br><br>
        Enter Email-id:<input type = "text" name = "emailId"/><br><br>
        Enter Phone Number:<input type = "text" name =
"phoneNumber"/><br><br>
        <input type = "submit" value = "SUBMIT"/>
      </h1>
    </form>
```

```
</center>
</body>
</html>
```

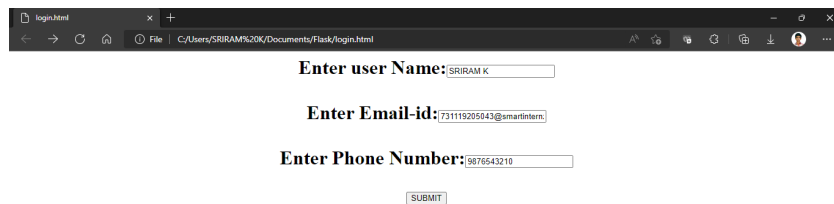
### Sample.py:

```
from flask import Flask, redirect, url_for, request
app = Flask(__name__)
```

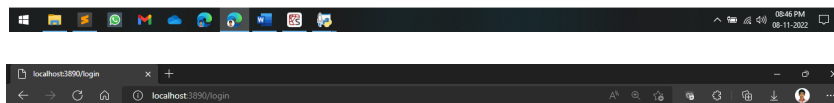
```
@app.route('/login', methods=['POST'])
def login():
```

```
    if request.method == 'POST':
        user_name = request.form['userName']
        email_id = request.form['emailId']
        phone_number = request.form['phoneNumber']
        return ' {} {} {} {} {} {}'.format("<center><h1>Your user name is: ",user_name,"</h1><br><br><h2>Your email-id is: ",email_id,"</h2><br><br><h3>Your phone number is: ",phone_number,"</h3></center>")
```

```
if __name__ == '__main__':
    app.run('127.0.0.1',3890)
```



A screenshot of a web browser window displaying a login form. The browser's address bar shows the file path 'C:\Users\SRIRAM\Documents\Flask\login.html'. The form contains three input fields: 'Enter user Name:' with the value 'SRIRAM K', 'Enter Email-id:' with the value '731119205043@smartinternz', and 'Enter Phone Number:' with the value '9876543210'. Below these fields is a 'SUBMIT' button.



**Your user name is: SRIRAM K**

**Your email-id is: 731119205043@smartinternz.com**

**Your phone number is: 9876543210**



### 3. Create User table with user with email, username, roll number, password.

The screenshot shows the IBM Db2 on Cloud SQL editor interface. The main editor window displays the following SQL code:

```
1 CREATE TABLE ASSIGNMENT2
2 (
3   EMAIL VARCHAR(50) NOT NULL PRIMARY KEY,
4   USER_NAME VARCHAR(20) NOT NULL,
5   ROLL_NUMBER VARCHAR(10) NOT NULL,
6   PASSWORD VARCHAR(20) NOT NULL
7 )
```

The code is highlighted in blue. Below the editor, the 'History' tab is active, showing a table with the following data:

Script	Date	Status	Runtime
Untitled - 1	Nov 8, 2022 2:09:44 PM	1	0.173 s
CREATE TABLE ASSIGNMENT2 ( EMAIL VARCHAR(50) NOT NULL PRIMARY K...			0.173 s

The screenshot shows the IBM Db2 on Cloud Tables view. The 'Tables' tab is active, displaying a table with the following data:

Name	Schema	Properties
ASSIGNMENT2	RKK02660	...

The 'Table definition' tab is also active, showing the table definition for ASSIGNMENT2. The table has the following columns:

Name	Data type	Nullable	Length	Scale
EMAIL	VARCHAR	N	50	0
USER_NAME	VARCHAR	N	20	0
ROLL_NUMBER	VARCHAR	N	10	0
PASSWORD	VARCHAR	N	20	0

The 'View data' button is visible at the bottom of the table definition panel.

The screenshot shows the IBM Db2 on Cloud SQL editor interface. On the left, the 'Data objects' pane shows a database named 'RKK02660'. The main editor area shows a SQL script titled 'Untitled - 1' with the following content:

```
1 INSERT INTO ASSIGNMENT2 VALUES
2 (
3 '731119205043@smartinternz.com',
4 'smartsriram',
5 '19IMT43',
6 'Sriram@123'
7 )
```

Below the editor, the 'History' tab is active, displaying a table of executed scripts:

Script	Date	Status	Runtime
Untitled - 1	Nov 8, 2022 2:13:18 PM	✓ 1	0.007 s
INSERT INTO ASSIGNMENT2 VALUES ( '731119205043@smartinternz.com...		✓	0.007 s

The screenshot shows the IBM Db2 on Cloud SQL editor interface with the 'Tables' tab selected. The table 'RKK02660.ASSIGNMENT2' is displayed. The table structure is as follows:

EMAIL	USER_NAME	ROLL_NUMBER	PASSWORD
731119205043@smartinternz.com	smartsriram	19IMT43	Sriram@123

Buttons for 'Back' and 'Export to CSV' are visible in the top right corner of the table view.

4.Perform UPDATE, DELETE Queries with user table



The screenshot shows the IBM Db2 on Cloud SQL editor interface. On the left, the 'Data objects' pane shows a database named 'RKK02660'. The main editor area shows a SQL script titled 'Untitled - 1' with the following statement:

```
1 DELETE FROM ASSIGNMENT2 WHERE USER_NAME='smartsriram'
```

Below the editor, the 'History' tab is active, displaying a table of executed scripts:

Script	Date	Status	Runtime
Untitled - 1	Nov 8, 2022 2:29:58 PM	✓ 1	0.009 s
DELETE FROM ASSIGNMENT2 WHERE USER_NAME='smartsriram'		✓	0.009 s
Untitled - 1	Nov 8, 2022 2:27:25 PM	✓ 1	0.009 s
UPDATE ASSIGNMENT2 SET ROLL_NUMBER='19INT40',PASSWORD='HeroSanthosh_'		✓	0.009 s

The screenshot shows the IBM Db2 on Cloud 'Tables' view. The table 'RKK02660.ASSIGNMENT2' is selected. The table structure is displayed as follows:

EMAIL	USER_NAME	ROLL_NUMBER	PASSWORD
There is no data here yet			

***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=ba99a9e6-d59e-4
883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PO
RT=31321;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.c
rt;UID=rkk02660;PWD=MjT4FnHjLsN6rpbn",'','')
```

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

```
def home():
```

```
    return render_template('home.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!'
```

```
            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 register():
```

```
    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 exist!'
elif not re.match(r'^@]+@[^@]+\.[^@]+',email):
    msg="Invalid email address"
else:
    insert_sql="INSERT INTO users VALUES(?, ?, ?)"
    prep_stmt=ibm_db.prepare(conn,inser_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',methods=['GET','POST'])
def dash():
    return render_template('dashboard.html')

```

```

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

```



```

insert_sql="INSERT INTO job VALUES(?, ?, ?, ?)"
perp_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(stmt)
msg="You have successfully applied for the job"
session['Loggedin']
TEXT="Hello user, a new application for job position" +
jobs+isrequested

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("accountdisplay",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")

```