

IBM ASSIGNMENT II

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

The screenshot shows the IBM Db2 on Cloud SQL editor interface. The left sidebar displays the 'Data objects' tab with a search filter 'MBC32363'. The main editor area shows a script titled '*Untitled - 1' with the following SQL code:

```
1 insert into User values('sabrina','abc@gmail.com','1234','abc');
2 insert into User values('rayna','abcd@gmail.com','12345','abcd');
3 insert into User values('neil','abcde@gmail.com','123456','abcde');
4 insert into User values('stacey','abcdef@gmail.com','1234567','abcdef');
5
```

The 'History' tab is open, showing a table of script execution history:

Script	Date	Status	Runtime
Untitled - 1	Oct 19, 2022 3:52:07 PM	✓ 4	0.052 s
insert into User values('sabrina','abc@gmail.com','1234','abc')		✓	0.007 s
insert into User values('rayna','abcd@gmail.com','12345','abcd')		✓	0.005 s
insert into User values('neil','abcde@gmail.com','123456','abcd')		✓	0.004 s
insert into User values('stacey','abcdef@gmail.com','1234567','')		✓	0.036 s

The screenshot shows the IBM Db2 on Cloud SQL editor interface with the 'Tables' tab selected. The table 'MBC32363.USER' is displayed with the following columns: USERNAME, EMAIL, reg.no, and PASSWORD. The table contains four rows of data:

USERNAME	EMAIL	reg.no	PASSWORD
neil	abcde@gmail.com	123456	abcde
rayna	abcd@gmail.com	12345	abcd
sabrina	abc@gmail.com	1234	abc
stacey	abcdef@gmail.com	1234567	abcdef

2. Perform UPDATE,DELETE Queries with user table.

The screenshot shows the IBM Db2 on Cloud web interface. The left sidebar contains navigation icons for Data objects, My script, and SQL. The main area is divided into a script editor and a history panel. The script editor contains the following SQL code:

```
1 insert into User values('sabrina','abc@gmail.com','1234','abc');
2 insert into User values('rayna','abcd@gmail.com','12345','abcd');
3 insert into User values('neil','abcde@gmail.com','123456','abcde');
4 insert into User values('stacey','abcde@gmail.com','1234567','abcdef');
5 UPDATE User SET USERNAME = 'swapna' WHERE USERNAME = 'sabrina';
```

The history panel shows the execution of the UPDATE query:

Script	Date	Status	Runtime
Untitled - 1	Oct 19, 2022 4:02:15 PM	1	0.007 s
UPDATE User SET USERNAME = 'swapna' WHERE USERNAME = 'sabrina'			0.007 s

The screenshot shows the IBM Db2 on Cloud web interface. The left sidebar contains navigation icons for Data objects, My script, and SQL. The main area is divided into a script editor and a history panel. The script editor contains the following SQL code:

```
1 insert into User values('sabrina','abc@gmail.com','1234','abc');
2 insert into User values('rayna','abcd@gmail.com','12345','abcd');
3 insert into User values('neil','abcde@gmail.com','123456','abcde');
4 insert into User values('stacey','abcde@gmail.com','1234567','abcdef');
5 UPDATE User SET username = sarah, WHERE username='sabrina';
6 DELETE FROM User WHERE PASSWORD = 'abcd';
7
```

The history panel shows the execution of the DELETE query:

Script	Date	Status	Runtime
Untitled - 1	Oct 19, 2022 4:07:26 PM	1	0.008 s
DELETE FROM User WHERE PASSWORD = 'abcd'			0.008 s

3. Connect python code to db2.

```
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=6667d8e9-9d4d-4ccb-ba32-
21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30376;SECURITY
=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=tbl66129;PWD=LPzyHjRGcm XprQoN",",")

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

def register():

    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)

if __name__ == '__main__':

    app.run(debug = True)

```

register.html

```

<!DOCTYPE html>

<html>

<body>

<center><h1>ASSIGNMENT 2</H1></CENTER>

<center><h2>REGISTRATION</h2></center>

<form action="http://localhost:5000"method="POST">

<P>Enter username:</p>

<p><input type="text" name="username"/></p>

<p>Enter mail:</p>

<p><input type="email" name="email"/></p>

<p>Enter password:</p>

<p><input type="password" name="password"/></p>

```

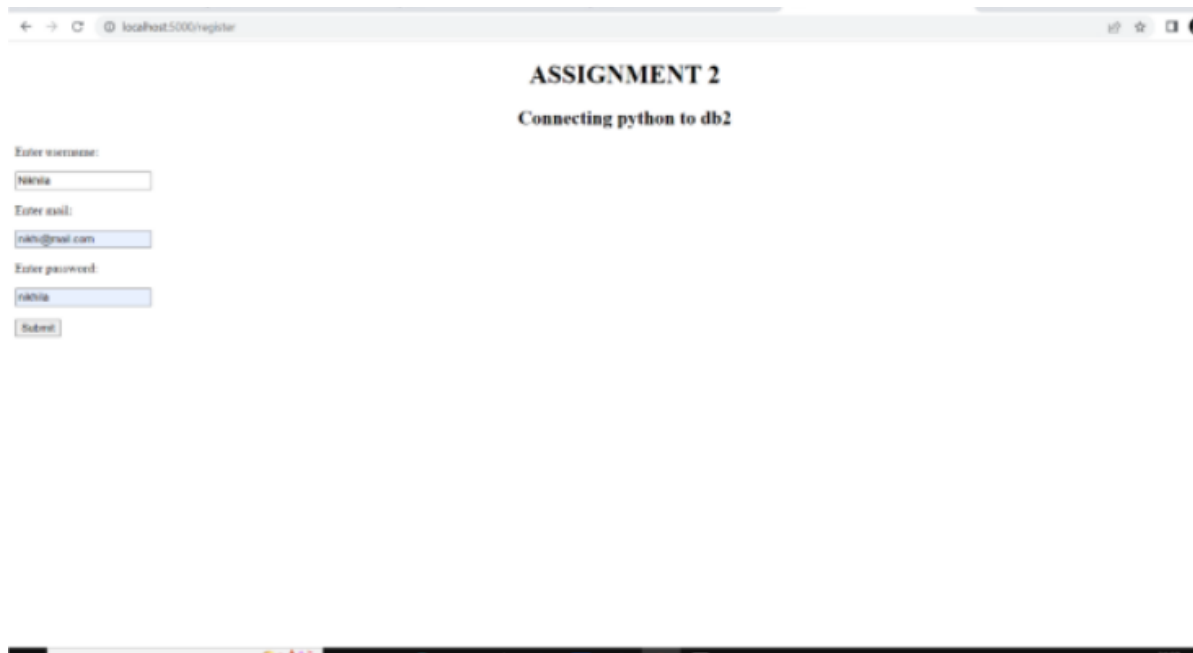
<p><input type="submit" value ="Submit"/></p>

{{msg}}

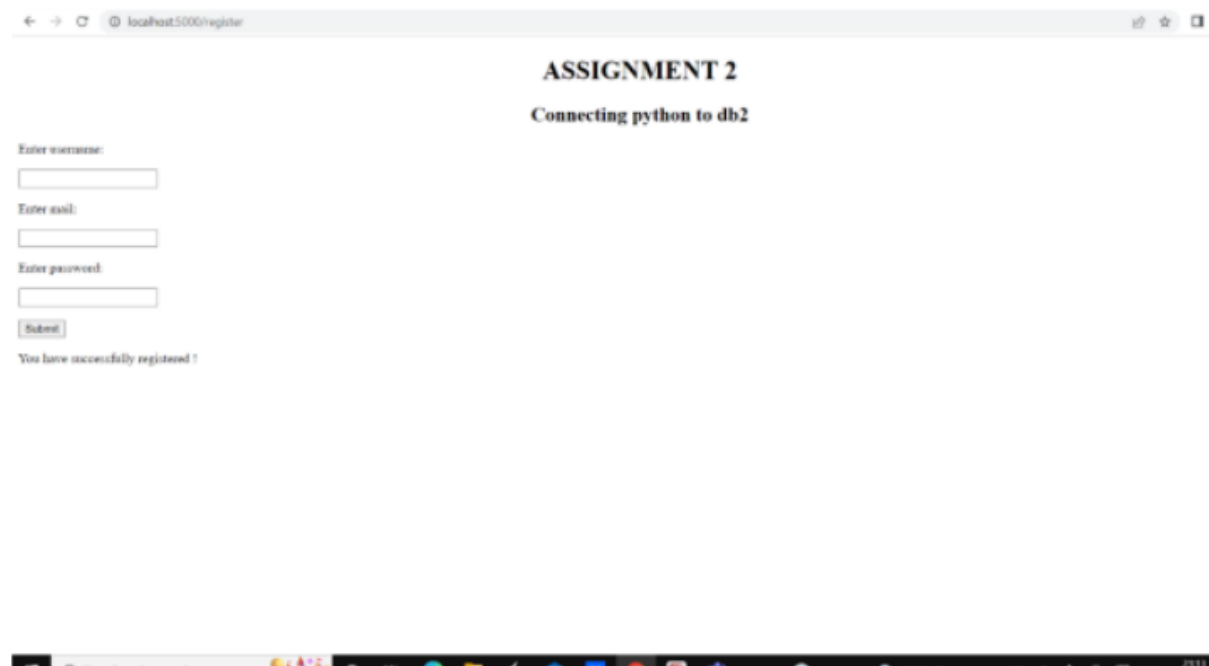
</form>

</body>

</html>



A screenshot of a web browser window. The address bar shows 'localhost:5000/register'. The page title is 'ASSIGNMENT 2' and the subtitle is 'Connecting python to db2'. The form contains three input fields: 'Enter username:' with the value 'nishi', 'Enter email:' with the value 'nishi@gmail.com', and 'Enter password:' with the value 'nishi'. A 'Submit' button is at the bottom of the form.



A screenshot of a web browser window showing the same registration form as above, but with a success message. The address bar shows 'localhost:5000/register'. The page title is 'ASSIGNMENT 2' and the subtitle is 'Connecting python to db2'. The form fields are empty. Below the form, a message says 'You have successfully registered !'. The browser's taskbar is visible at the bottom.

4. 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 the database and navigate to the login page to authenticate user username and password. If the user is valid show the welcome page.

```
<!DOCTYPE html>

<html>

<body>

<center><h1>ASSIGNMENT 2</H1></CENTER>

<center><h2>REGISTRATION</h2></center>

<form action="http://localhost:5000"method="POST">

<P>Enter username:</p>

<p><input type="text" name="username"/></p>

<p>Enter mail:</p>

<p><input type="email" name="email"/></p>

<p>Enter password:</p>

<p><input type="password" name="password"/></p>

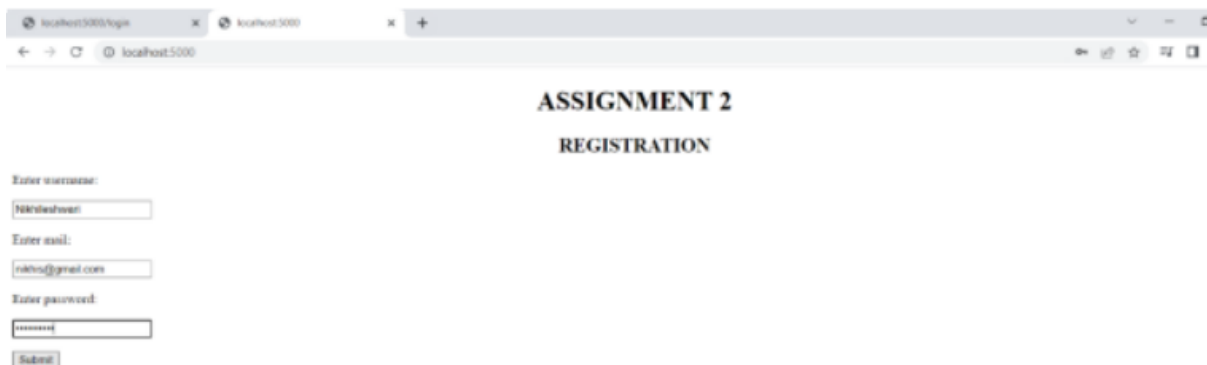
<p><input type="submit" value ="Submit"/></p>

{{msg}}

</form>

</body>

</html>
```



localhost:5000/login x localhost:5000 x +

localhost:5000

ASSIGNMENT 2

REGISTRATION

Enter username:

nikhleshwar

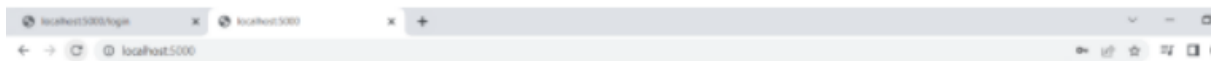
Enter mail:

nikesh@gmail.com

Enter password:

password

Submit



ASSIGNMENT 2

REGISTRATION

Enter username:

Enter email:

Enter password:

You have successfully registered !

TBL66129.USERS Back

Report to CSV

USERNAME	EMAIL	PASSWORD
Donna	donna@abc.com	donna
Harry	harry@gmail.com	harry
Nicki	nik@abc.com	nikabc
Nikhila	nikhi@email.com	nikhila
Nikhleshwari	nikhi@gmail.com	nikhila234
abc	abc@abc.com	abc

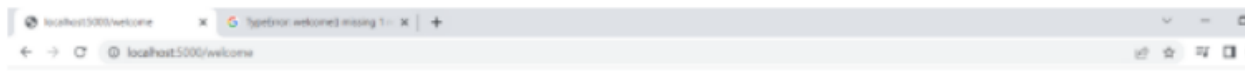


ASSIGNMENT 2

LOGIN

Enter username:

Enter password:



WELCOME

You have successfully logged in!