

## Assignment -2

### DB and Flask

Assignment Date	12 October 2022
Student Name	Mr M J M Salman
Student Roll Number	621319104047
Maximum Marks	2 Marks

#### Question-1:

Create user table with email, username, roll number, password

The screenshot shows the IBM Db2 on Cloud interface. On the left, the 'Data objects' pane shows a database named 'LHK92942'. The main editor area is titled 'Assignment-2' and contains the following SQL script:

```
1 create table user(
2     Sl_No int generated by default as identity not null,
3     User_Name varchar(200) not null,
4     User_Email varchar(200) not null,
5     Roll_No int not null,
6     password varchar(200),
7     primary key(Sl_No)
8 );
```

Below the script editor, the 'History' tab is active, showing a table with columns: Script, Date, Status, and Runtime. The table contains two entries:

Script	Date	Status	Runtime
Assignment-2	Oct 18, 2022 3:11:13 PM	✓ 1	0.111 s
create table user( Sl_No int generated by default as identity n...		✓	0.111 s

#### Insert Values:

The screenshot shows the IBM Db2 on Cloud interface with a script titled '\*Untitled - 1'. The script contains the following SQL statements:

```
3 user_name varchar(200) not null,
4 user_email varchar(200) not null,
5 roll_no int not null,
6 password varchar(200),
7 primary key(sl_no)
8 );
9
10 insert into user(user_name,user_email,roll_no,password) values('Karthikeyan','karthi@gmail.com',22,'karthi');
11 insert into user(user_name,user_email,roll_no,password) values('sathiyanth','sathi@gmail.com',51,'sathi');
12 insert into user(user_name,user_email,roll_no,password) values('naveen','naveen@gmail.com',37,'naveen');
13 insert into user(user_name,user_email,roll_no,password) values('salman','salman@gmail.com',47,'salman');
14
15 update user set user_name = 'kiruba' where roll_no = 37;
16
17 delete from user where user_name = 'kiruba';
```

Below the script editor, the 'History' tab is active, showing a table with columns: Script, Date, Status, and Runtime. The table contains four entries:

Script	Date	Status	Runtime
*Untitled - 1	Oct 27, 2022 7:14:48 PM	✓ 1	0.013 s
delete from user where user_name = 'kiruba'		✓	0.013 s
*Untitled - 1	Oct 27, 2022 7:12:20 PM	✓ 1	0.009 s
update user set user_name = 'kiruba' where roll_no = 37		✓	0.009 s
*Untitled - 1	Oct 27, 2022 7:09:32 PM	✓ 4	0.036 s

## User table:

The screenshot shows the IBM Db2 on Cloud interface. The top navigation bar includes 'Load Data', 'Load History', 'Tables', 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. The left sidebar has icons for 'Data objects', 'SQL', and 'My script'. The main area displays the table 'VBD21229.USER'. A 'Back' button is in the top right. An 'Export to CSV' button is in the top right. A table with 5 columns is shown: SL\_NO, USER\_NAME, USER\_EMAIL, ROLL\_NO, and an unnamed column. The table contains 4 rows of data. A 'Game clip recorded' notification is visible on the right side of the table.

SL_NO	USER_NAME	USER_EMAIL	ROLL_NO	
1	Karthikeyan	karthi@gmail.com	22	
2	sathiyanth	sakthi@gmail.com	51	sathi
3	kiruba	naveen@gmail.com	37	naveen
4	salman	salman@gmail.com	47	salman

## Question-2:

Perform update & delete queries with the table.

The screenshot shows the IBM Db2 on Cloud interface with a SQL editor. The editor contains the following SQL queries:

```
3 user_name varchar(200) not null,  
4 user_email varchar(200) not null,  
5 roll_no int not null,  
6 password varchar(200),  
7 primary key(sl_no)  
8 );  
9  
10 insert into user(user_name,user_email,roll_no,password) values('Karthikeyan','karthi@gmail.com',22,'karthi');  
11 insert into user(user_name,user_email,roll_no,password) values('sathiyanth','sakthi@gmail.com',51,'sathi');  
12 insert into user(user_name,user_email,roll_no,password) values('naveen','naveen@gmail.com',37,'naveen');  
13 insert into user(user_name,user_email,roll_no,password) values('salman','salman@gmail.com',47,'salman');  
14  
15 update user set user_name = 'kiruba' where roll_no = 37;  
16  
17 delete from user where user_name = 'kiruba';
```

The 'History' tab shows the execution of these queries:

Script	Date	Status	Runtime
Untitled - 1	Oct 27, 2022 7:14:48 PM	✓ 1	0.013 s
delete from user where user_name = 'kiruba'		✓	0.013 s
Untitled - 1	Oct 27, 2022 7:12:20 PM	✓ 1	0.009 s
update user set user_name = 'kiruba' where roll_no = 37		✓	0.009 s
Untitled - 1	Oct 27, 2022 7:09:32 PM	✓ 4	0.036 s

IBM Db2 on Cloud

00:02:05

Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

VBD21229.USER

Back

Export to CSV

SL_NO	USER_NAME	USER_EMAIL	ROLL_NO	PASSWORD
1	Karthikeyan	karthi@gmail.com	22	karthi
2	sathiyananth	sakthi@gmail.com	51	sathi
4	salman	salman@gmail.com	47	salman

### Question-3:

Connect python code to db2.

```
import ibm_db
```

```
import bcrypt
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b70af05b-76e4-4bca-a1f523dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32716;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=Ihh93934;PWD=6vYNck1qqbU6aoBz","")
```

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

### Solution:

#### app.py

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

```

import ibm_db

import bcrypt

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b70af05b-76e4-4bca-
a1f523dbb4c6a74e.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=
32716;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;PROTOCO
L=TCPIP;UID=lhh93934;PWD=6vYNck1qqbU6aoBz","")

# url_for('static', filename='style.css')

app = Flask(__name__)

app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

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

def home():

    if 'email' not in session:

        return redirect(url_for('login'))

    return render_template('home.html',name='Home')

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

def register():

    if request.method == 'POST':

        email = request.form['email']

        username = request.form['username']

        rollNo = request.form['rollNo']

        password = request.form['password']

        if not email or not username or not rollNo or not password:

            return render_template('register.html',error='Please fill all fields')

```

```

hash=bcrypt.hashpw(password.encode('utf-8'),bcrypt.gensalt())

query = "SELECT * FROM user_details WHERE email=? OR rollNo=?"

stmt = ibm_db.prepare(conn, query)

ibm_db.bind_param(stmt,1,email)

ibm_db.bind_param(stmt,2,rollNo)

ibm_db.execute(stmt)

isUser = ibm_db.fetch_assoc(stmt)


if not isUser:

    insert_sql = "INSERT INTO user_details(EMAIL, USERNAME, ROLLNO,
PASSWORD) VALUES (?, ?, ?, ?)"

    prep_stmt = ibm_db.prepare(conn, insert_sql)

    ibm_db.bind_param(prepare_stmt, 1, email)

    ibm_db.bind_param(prepare_stmt, 2, username)

    ibm_db.bind_param(prepare_stmt, 3, rollNo)

    ibm_db.bind_param(prepare_stmt, 4, hash)

    ibm_db.execute(prepare_stmt)

    return render_template('register.html',success="You can login")

else:

    return render_template('register.html',error='Invalid Credentials')

return render_template('register.html',name='Home')

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

```

```
def login():

    if request.method == 'POST':

        email = request.form['email']

        password = request.form['password']


        if not email or not password:

            return render_template('login.html',error='Please fill all fields')

        query = "SELECT * FROM user_details WHERE email=?"

        stmt = ibm_db.prepare(conn, query)

        ibm_db.bind_param(stmt,1,email)

        ibm_db.execute(stmt)

        isUser = ibm_db.fetch_assoc(stmt)

        print(isUser,password)

        if not isUser:

            return render_template('login.html',error='Invalid Credentials')

        isPasswordMatch = bcrypt.checkpw(password.encode('utf-8'),isUser['PASSWORD'].encode('utf-8'))

        if not isPasswordMatch:

            return render_template('login.html',error='Invalid Credentials')


        session['email'] = isUser['EMAIL']

        return redirect(url_for('home'))
```

```
        return render_template('login.html',name='Home')
```

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

```
def logout():
```

```
    session.pop('email', None)
```

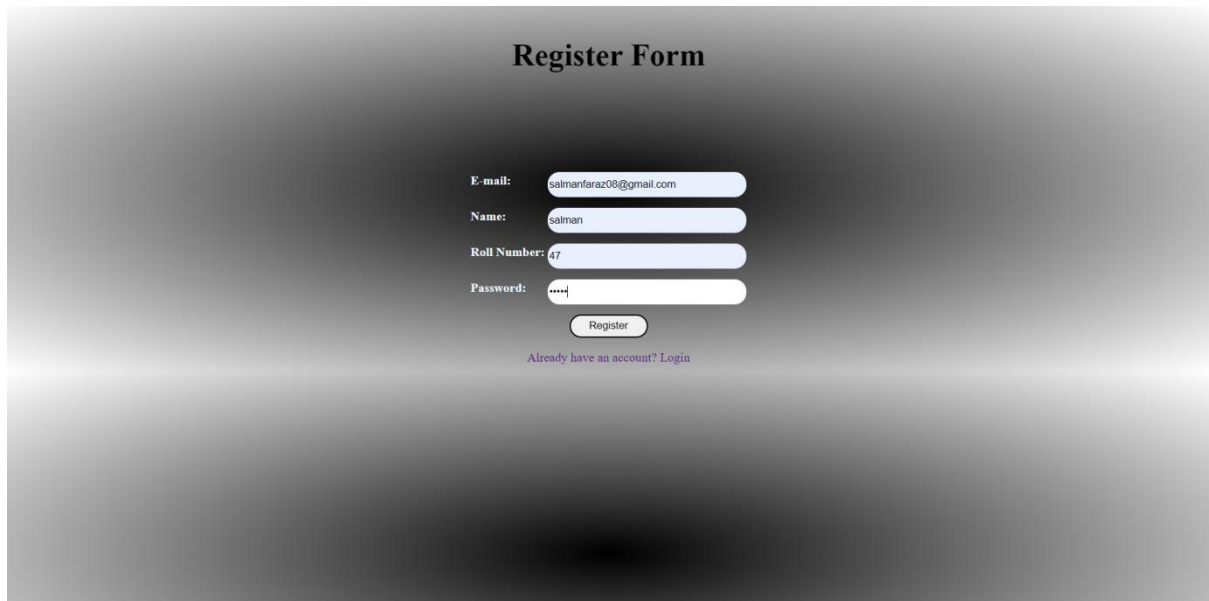
```
    return redirect(url_for('login'))
```

```
if __name__ == "__main__":
```

```
    app.run(debug=True)
```

**Output:**

**Registration Page:**



The registration form is titled "Register Form" in a bold, black, serif font, centered at the top. Below the title, there are four input fields, each with a label to its left: "E-mail:" with the value "salmanfaraz08@gmail.com", "Name:" with the value "salman", "Roll Number:" with the value "47", and "Password:" with masked characters "\*\*\*\*". Each input field has a light blue border and rounded corners. Below the password field is a "Register" button with a light blue border and rounded corners. At the bottom, there is a link "Already have an account? Login" in a purple, italicized font.

**Register Form**

E-mail: salmanfaraz08@gmail.com

Name: salman

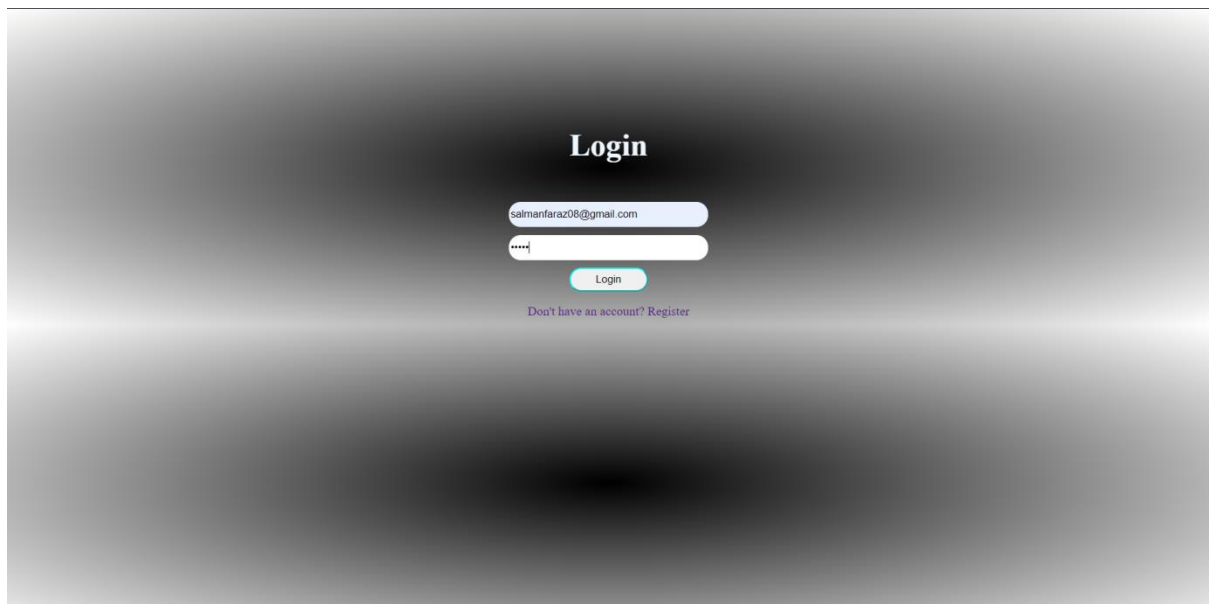
Roll Number: 47

Password: \*\*\*\*

Register

*Already have an account? Login*

**Login Page:**



The login page is titled "Login" in a bold, black, serif font, centered at the top. Below the title, there are two input fields, each with a light blue border and rounded corners: the first contains the email "salmanfaraz08@gmail.com" and the second contains masked characters "\*\*\*\*". Below the password field is a "Login" button with a light blue border and rounded corners. At the bottom, there is a link "Don't have an account? Register" in a purple, italicized font.

**Login**

salmanfaraz08@gmail.com

\*\*\*\*

Login

*Don't have an account? Register*



Home Page:



Database:

