Assignment-2

1)Table with user with email,username,roll number,password.

TABLE EMPLOYEE CREATED

```
CREATE TABLE Employee
(
Rollno int,
username varchar(255),
password varchar(255),
Email varchar(255)
);
```

Employee

Rollno	username	password	Email
empty			

2) UPDATE,DELETE Queries with user table

INSERT QUERY PERFORMED ON TABLE EMPLOYEE

```
INSERT INTO Employee
(Rollno,username, password, Email)
VALUES (4, "Jack","jac123","jac.gmail.com" ),
(5, "Jay","j123","j.gmail.com" )
```

Employee

Rollno	username	password	Email
4	Jack	jac123	jac.gmail.com
4	Jack	jac123	jac.gmail.com
5	Jay	j123	j.gmail.com

DELETE QUERY PERFORMED ON TABLE EMPLOYEE

```
DELETE FROM Employee WHERE Rollno=4;
```

Employee

Rollno	username	password	Email
5	Jay	j123	j.gmail.com

3)PYTHON CODE TO CONNECT IBM DB2

```
#-----------------------------------------------------------------------------------------#
#  NAME:     ibm_db-connect_SERVER.py                                                     #
#                                                                                         #
#  PURPOSE:  This program is designed to illustrate how to use the ibm_db.connect() API to  #
#            establish a connection to a Db2 server.                                      #
#                                                                                         #
#            Additional APIs used:                                                        #
#               ibm_db.close()                                                            #
#                                                                                         #
#  USAGE:    Log in as a Db2 database instance user (for example, db2inst1) and issue the  #
#            following command from a terminal window:                                    #
#                                                                                         #
#            ./ibm_db-connect_SERVER.py                                                   #
#-----------------------------------------------------------------------------------------#
# Load The Appropriate Python Modules
import sys        # Provides Information About Python Interpreter Constants, Functions, & Methods
```

```
import ibm_db      # Contains The APIs Needed To Work With Db2 Databases
from ipynb_exit import exit
# Define And Initialize The Appropriate Variables
hostName = "197.126.80.22"    # IP Address Of Remote Server
portNum = "50000"            # Port Number That Receives Db2 Connections On The Remote Server
userID = "db2inst2"          # The Instance User ID At The Remote Server
passWord = "ibmdb2"          # The Password For The Instance User ID At The Remote Server
connectionID = None
# Display A Status Message Indicating An Attempt To Establish A Connection To A Db2 Server
# Is About To Be Made
print("\nConnecting to the \'" + hostName + "\' server ... ", end="")
# Construct The String That Will Be Used To Establish A Db2 Server Connection
connString = "DRIVER={IBM DB2 ODBC DRIVER}"
connString += ";ATTACH=TRUE"            # Attach To A Server; Not A Database
connString += ";DATABASE="              # Ignored When Connecting To A Server
connString += ";HOSTNAME=" + hostName    # Required To Connect To A Server
connString += ";PORT=" + portNum         # Required To Connect To A Server
connString += ";PROTOCOL=TCPIP"          # Required To Connect To A Server
connString += ";UID=" + userID
connString += ";PWD=" + passWord
# Attempt To Establish A Connection To The Server Specified
try:
   connectionID = ibm_db.connect(connString, '', '')
except Exception:
   pass
# If A Db2 Server Connection Could Not Be Established, Display An Error Message And Exit
if connectionID is None:
   print("\nERROR: Unable to connect to the \'" + hostName + "\' server.")
   print("Connection string used: " + connString + "\n")
   exit(-1)
```

```
# Otherwise, Complete The Status Message
else:
    print("Done!\n")
# Add Additional Db2 Server-Related Processing Here...
# For Example, ibm_db.createdb(), ibm_db.createdbNX(), ibm_db.recreatedb(), ibm_db.dropdb()
# Attempt To Close The Db2 Server Connection That Was Just Opened
if not connectionID is None:
    print("Disconnecting from the \'" + hostName + "\' server ... ", end="")
    try:
        returnCode = ibm_db.close(connectionID)
    except Exception:
        pass
    # If The Db2 Server Connection Was Not Closed, Display An Error Message And Exit
    if returnCode is False:
        print("\nERROR: Unable to disconnect from the " + hostName + " server.")
        exit(-1)
    # Otherwise, Complete The Status Message
    else:
        print("Done!\n")
# Return Control to the Operating System
exit()
```

4) FLASK APP WITH REGISTRATION PAGE LOGIN PAGE AND WELCOME PAGE

1. Creating Environment :

Step 1 - py -3 -m venv venv

Step 2 - venv\Scripts\activate

Step 3 - pip install Flask

2.MySQL Workbench :

Step-1 - Install MySQL workbench

Step-2 - Install ‘mysqlbd’ module in your venv.

pip install flask-mysqldb

Step 3 - Open My sql workbench and write the code

CREATE DATABASE IF NOT EXISTS 'DEFAULT CHARACTER SET utf8 COLLATE utf8 general_ci; USE 'geeklogin";

CREATE TABLE IF NOT EXISTS `accounts`(

'id' int(11) NOT NULL AUTO_INCREMENT,

username varchar(50) NOT NULL,

password' varchar(255) NOT NULL,

email varchar(100) NOT NULL,

PRIMARY KEY ('id')

)ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;

3.Creating Project Step-1:

Create app.py

Store this code in 'app.py' file

```
from flask import Flask, render_template, request, redirect, url_for, session
from flask_mysqldb import MySQL
import MySQLdb.cursors
import re
app = Flask(__name__)
app.secret_key = 'your secret key'
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'your password'
app.config['MYSQL_DB'] = 'geeklogin'
mysql = MySQL(app)
@app.route('/')
@app.route('/login', methods =['GET', 'POST'])
def login():
```

```
    msg = ''
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM accounts WHERE username = % s AND password = % s', (username, password, ))
        account = cursor.fetchone()
        if account:
            session['loggedin'] = True
            session['id'] = account['id']
            session['username'] = account['username']
            msg = 'Logged in successfully !'
            return render_template('index.html', msg = msg)
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
@app.route('/logout')
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return redirect(url_for('login'))

@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form and 'email' in request.form :
        username = request.form['username']
        password = request.form['password']
```

```
                email = request.form['email']
                cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
                cursor.execute('SELECT * FROM accounts WHERE username = % s', (username, ))
                account = cursor.fetchone()
                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 = 'Username must contain only characters and numbers !'
                elif not username or not password or not email:
                        msg = 'Please fill out the form !'
                else:
                        cursor.execute('INSERT INTO accounts VALUES (NULL, % s, % s, % s)',
(username, password, email, ))
                        mysql.connection.commit()
                        msg = 'You have successfully registered !'
        elif request.method == 'POST':
                msg = 'Please fill out the form !'
        return render_template('register.html', msg = msg)
```

4.Create Templates:

step 1 - login.html

```
    <!-- Store this code in 'login.html' file inside the 'templates' folder -->

<html>
        <head>
                <meta charset="UTF-8">
```

```
        <title> Login </title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    </head>
    <body></br></br></br></br></br>
        <div align="center">
        <div align="center" class="border">
            <div class="header">
                <h1 class="word">Login</h1>
            </div></br></br></br>
            <h2 class="word">
                <form action="{{ url_for('login') }}" method="post">
                <div class="msg">{{ msg }}</div>
                    <input id="username" name="username" type="text"
placeholder="Enter Your Username" class="textbox"/></br></br>
                    <input id="password" name="password" type="password"
placeholder="Enter Your Password" class="textbox"/></br></br></br>
                    <input type="submit" class="btn" value="Sign
In"></br></br>
                </form>
            </h2>
            <p class="bottom">Don't have an account? <a class="bottom"
href="{{url_for('register')}}"> Sign Up here</a></p>
        </div>
        </div>
    </body>
</html>
```

step 2 - register.html

```
<!-- Store this code in 'register.html' file inside the 'templates' folder -->
```

```
<html>
	<head>
		<meta charset="UTF-8">
		<title> Register </title>
		<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

	</head>
	<body></br></br></br></br></br>
		<div align="center">
		<div align="center" class="border">
			<div class="header">
				<h1 class="word">Register</h1>
			</div></br></br></br>
			<h2 class="word">
				<form action="{{ url_for('register') }}" method="post">
				<div class="msg">{{ msg }}</div>
					<input id="username" name="username" type="text"
placeholder="Enter Your Username" class="textbox"/></br></br>
					<input id="password" name="password" type="password"
placeholder="Enter Your Password" class="textbox"/></br></br>
					<input id="email" name="email" type="text"
placeholder="Enter Your Email ID" class="textbox"/></br></br>
					<input type="submit" class="btn" value="Sign Up"></br>
				</form>
			</h2>
			<p class="bottom">Already have an account? <a class="bottom"
href="{{url_for('login')}}"> Sign In here</a></p>
		</div>
		</div>
	</body>
</html>
```

step 3 - index.html

```
<!-- Store this code in 'index.html' file inside the 'templates' folder-->

<html>
    <head>
        <meta charset="UTF-8">
        <title> Index </title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

    </head>
    <body></br></br></br></br></br>
        <div align="center">
        <div align="center" class="border">
            <div class="header">
                <h1 class="word">Index</h1>
            </div></br></br></br>
                <h1 class="bottom">
                    Hi {{session.username}}!!</br></br> Welcome to the index
page...
                </h1></br></br></br>
                <a href="{{ url_for('logout') }}" class="btn">Logout</a>
        </div>
        </div>
    </body>
</html>
```

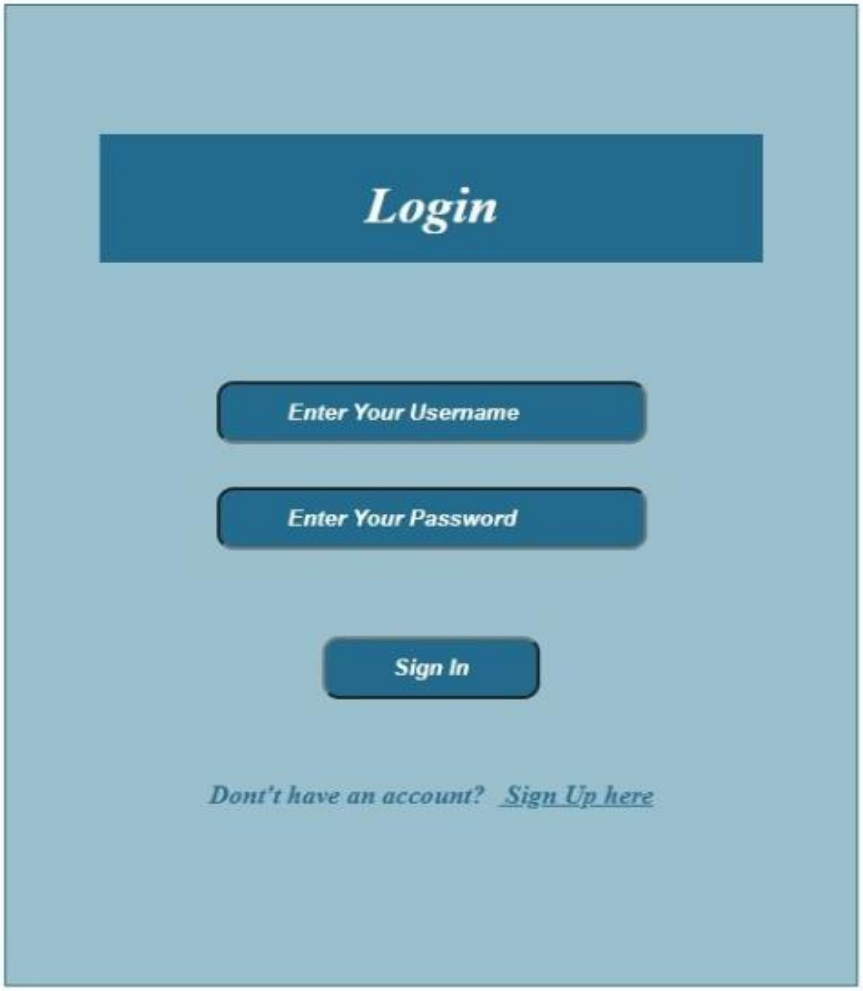
step 4 - create static folder

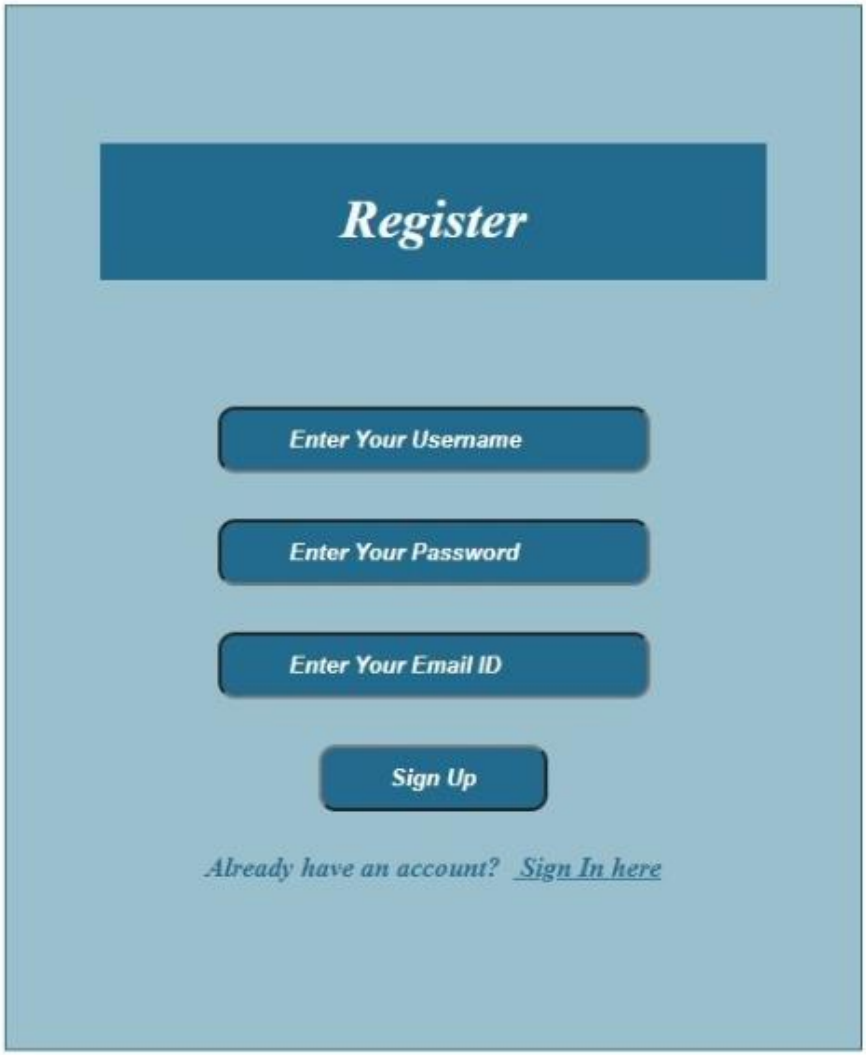
```
/* Store this code in 'style.css' file inside the 'static' folder*/
```

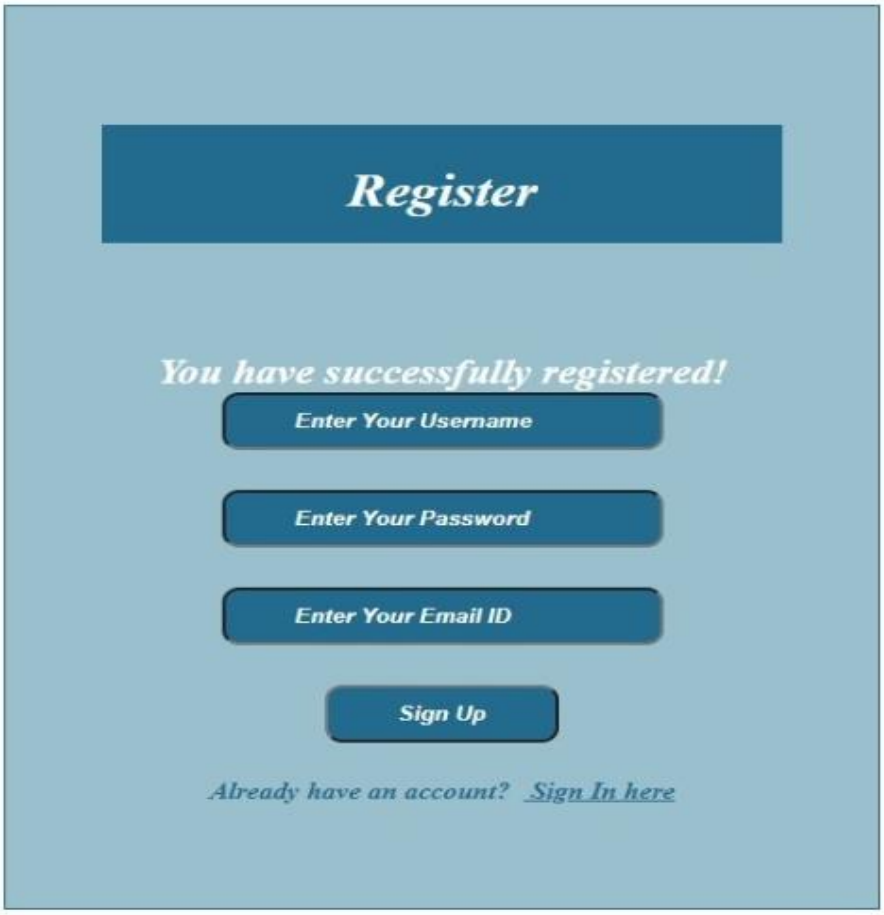
```
.header{
        padding: 5px 120px;
        width: 150px;
        height: 70px;
        background-color: #236B8E;
}

.border{
        padding: 80px 50px;
        width: 400px;
        height: 450px;
        border: 1px solid #236B8E;
        border-radius: 0px;
        background-color: #9AC0CD;
}

.btn {
        padding: 10px 40px;
        background-color: #236B8E;
        color: #FFFFFF;
        font-style: oblique;
        font-weight: bold;
        border-radius: 10px;
}

.textbox{
        padding: 10px 40px;
        background-color: #236B8E;
        text-color: #FFFFFF;
        border-radius: 10px;
```

```
}

::placeholder {
        color: #FFFFFF;
        opacity: 1;
        font-style: oblique;
        font-weight: bold;
}

.word{
        color: #FFFFFF;
        font-style: oblique;
        font-weight: bold;
}

.bottom{
        color: #236B8E;
        font-style: oblique;
        font-weight: bold;
}
```

OUTPUTS


Login
Enter Your Username
Enter Your Password
Sign In
Dont't have an account? Sign Up here


Register
Enter Your Username
Enter Your Password
Enter Your Email ID
Sign Up
Already have an account? Sign In here

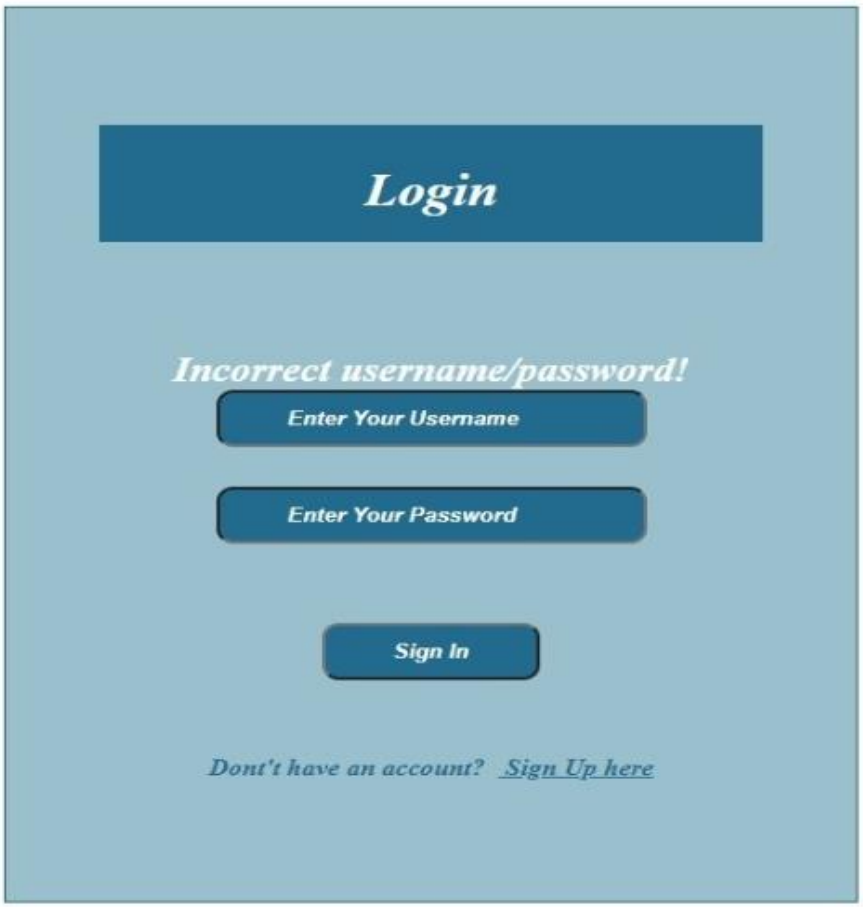

Register
You have successfully registered!
Enter Your Username
Enter Your Password
Enter Your Email ID
Sign Up
Already have an account? Sign In here

Index

Hi user!!

Welcome to the index page...

Logout


Login
Incorrect username/password!
Enter Your Username
Enter Your Password
Sign In
Dont't have an account? Sign Up here