FINAL DELIVERABLE

DATE	18 November 2022
TEAM ID	PNT2022TMID29412
PROJECT NAME	NUTRITION ASSISTANT APPLICATION

SCORCE CODE :

```
import ibm_db as db

from flask import Flask, render_template, request, redirect, session, abort

import os

import pathlib

import requests

from dotenv import load_dotenv

from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail

from google.oauth2 import id_token

from google_auth_oauthlib.flow import Flow

from pip._vendor import cachecontrol

import google.auth.transport.requests


# Configure Flask app

app = Flask(_name_)

SECRET_KEY = os.urandom(32)

app.config['SECRET_KEY'] = SECRET_KEY


# Load .env file

load_dotenv()
```

```
connection_string = "DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=SS
L;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=wpq72644;PWD=CTsaqSf91OIzLniF",'','
'.format(DATABASE_NAME, HOSTNAME, PORT_NUMBER, USERNAME, PASSWORD)

conn = db.connect(connection_string, "", "")


# Frequently used variables

SIGN_UP_PAGE_URL = '/'

LOG_IN_PAGE_URL = '/login'

HOME_PAGE_URL = '/home'

GOOGLE_LOGIN_PAGE_URL = '/google_login'

PROFILE_PAGE_URL = '/profile'

CHANGE_PASSWORD_URL = '/changepwd'


# Google Auth Configuration

os.environ["OAUTHLIB_INSECURE_TRANSPORT"] = "1"


client_secrets_file = os.path.join(pathlib.Path(_file_).parent, "client_secret.json")


flow = Flow.from_client_secrets_file(

    client_secrets_file=client_secrets_file,

    scopes=["https://www.googleapis.com/auth/userinfo.profile",
"https://www.googleapis.com/auth/userinfo.email", "openid"],

    redirect_uri="http://127.0.0.1:5000/callback"

)


# Helper Function to execute SQL queries

def execute_sql(statement, **params):

    global conn
    stmt = db.prepare(conn, statement)


    param_id = 1
    for key, val in params.items():
        db.bind_param(stmt, param_id, val)
        param_id += 1


    result = ''
    try:
        db.execute(stmt)
        result = db.fetch_assoc(stmt)
    except:
        pass


    return result


# Creates user table if not exists
create_table = "CREATE TABLE IF NOT EXISTS user(email varchar(30), username varchar(30),
password varchar(30))"
execute_sql(statement=create_table)


# Helper function to send confirmation mail on sign in
def send_confirmation_mail(user, email):
    message = Mail(
        from_email="nutrition@gmail.com",
        to_emails=email,
        subject="YAYY!! Your Account was created successfully!",
        html_content= "<strong>Account Created with username
{0}</strong>".format(user)
```

```
    )

    try:
        sg = SendGridAPIClient(os.environ.get('xxxxxxxxxxxxxxxxx'))
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e)

# Sign up page
@app.route(SIGN_UP_PAGE_URL, methods=['GET', 'POST'])
def signup():
    msg = ''

    if session.get('user'):
        return redirect(HOME_PAGE_URL)

    if request.method == 'POST':
        user = request.form['user']
        email = request.form['email']
        password = request.form['password']

        duplicate_check = "SELECT * FROM user WHERE username=?"
        account = execute_sql(statement=duplicate_check, user=user)

        if account:
                msg = "There is already an account with this username!"
            else:
                insert_query = "INSERT INTO user values(?, ?, ?)"
                execute_sql(statement=insert_query, email=email, user=user,
password=password)


                send_confirmation_mail(user, email)
                return redirect(LOG_IN_PAGE_URL)
    return render_template('Registrationform.html', msg=msg)


# Login page
@app.route(LOG_IN_PAGE_URL, methods=['GET', 'POST'])
def login():
    msg = ''


    if session.get('user'):
        return redirect(HOME_PAGE_URL)


    if request.method == "POST":


        user = request.form['user']
        password = request.form['password']


        duplicate_check = "SELECT * FROM user WHERE username=?"
        account = execute_sql(statement=duplicate_check, user=user)


        print(account)
        if account and account['PASSWORD'] == password:
                session['user'] = user
                return redirect(HOME_PAGE_URL)
            elif account and account['PASSWORD'] != password:
                msg = 'Invalid Password!'
            else:
                msg = "Invalid Username!"

        return render_template('Login.html', msg=msg)

# Login using Gmail
@app.route(GOOGLE_LOGIN_PAGE_URL , methods=['GET','POST'])
def google_login():
        authorization_url, state = flow.authorization_url()
        session["state"] = state
        return redirect(authorization_url)

# Configuring user credentials after gmail login
@app.route("/callback")
def callback():
        flow.fetch_token(authorization_response=request.url)

        if session["state"] != request.args["state"]:
            abort(500)    # State does not match!

        credentials = flow.credentials
        request_session = requests.session()
        cached_session = cachecontrol.CacheControl(request_session)
        token_request = google.auth.transport.requests.Request(session=cached_session)
```

```
    id_info = id_token.verify_oauth2_token(
        id_token=credentials._id_token,
        request=token_request,
        audience=GOOGLE_CLIENT_ID,
        clock_skew_in_seconds=10
    )

    session["user"] = id_info.get("email")
    session["google_id"] = id_info.get("sub")
    session["name"] = id_info.get("name")
    return redirect(HOME_PAGE_URL)

# Home page
@app.route(HOME_PAGE_URL, methods=['GET', 'POST'])
def homepage():
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)

    msg = ''
    if request.method == 'POST':
        if request.form['food']:
            msg = 'Image Uploaded Successfully!'
        else:
            msg = "Image wasn't uploaded, Try again!"

    return render_template('front.html', user=session.get('user'), msg=msg)
```

```
# Profile page
@app.route(PROFILE_PAGE_URL, methods=['GET', 'POST'])
def profile():
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)

    sqlst = "select email from user where username=?"
    user = session.get('user')
    email = execute_sql(statement=sqlst, user=user)

    return render_template('front.html', user=user, email=email['EMAIL'])

#change password
@app.route(CHANGE_PASSWORD_URL, methods=['GET', 'POST'])
def changepwd():
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)

    msg = ''
    user = ''
    email = ''
    if request.method == 'POST':
        user = session.get('user')
        oldpass = request.form['oldpass']
        newpass = request.form['newpass']

        sqlst = 'SELECT password from user where username = ?'
        dbpass = execute_sql(statement = sqlst , username = user)['PASSWORD']
        sqlst = 'SELECT email from user where username = ?'
        email = execute_sql(statement = sqlst ,username = user)['EMAIL']

        if dbpass == oldpass:
            sqlst = 'UPDATE user SET password = ? where username = ?'
            execute_sql(statement = sqlst , password = newpass , username = user)
            msg = 'Updated Successfully!'
        else:
            msg = 'Old Password Incorrect!'

        return render_template('front.html', user=user, email=email, msg=msg)

    return render_template('passwordChange.html')



# Logout user
@app.route('/logout')
def logout():
    session['user'] = ''
    return redirect(LOG_IN_PAGE_URL)

# Delete user account
@app.route('/delete')
def delete():
    if not session.get('user'):
        return redirect(LOG_IN_PAGE_URL)

    user = session['user']
    delete_query = "DELETE FROM user WHERE username=?"
    execute_sql(statement=delete_query, user=user)


    session.clear()
    return redirect(SIGN_UP_PAGE_URL)


# Run the application
if _name_ == '_main_':
    app.run(debug=True)
```