Project Development Phase

Sprint 3

Profile.html:

```
{% extends 'base2.html '%} {% block head %}

<title>Profile</title>

{% endblock%} {%block body%}

<h2>Profile</h2>

<hr />

<div class="user-deatils">

  <h3>User Details</h3>

  <h4>USERNAME : {{data['USERNAME']}}</h4>

  <h4>FIRSTNAME : {{data['FIRSTNAME']}}</h4>

  <h4>LASTNAME : {{data['LASTNAME']}}</h4>

  <h4>EMAIL : {{data['EMAIL']}}</h3>

</div>

<hr>

<div class="forms-wrapper mg-20 " >

  <form action="{{url_for('updateUser')}}" method="post">

    <h3>Update user details</h3>

    <div class="field">

      <label for="input-field">Choose a field :</label>

      <select name="input-field" id="field">

        <option value="USERNAME">USERNAME</option>

        <option value="FIRSTNAME">FIRSTNAME</option>

        <option value="LASTNAME">LASTNAME</option>

      </select>

    </div>

    <div class="field">

      <label class="custom-label" for="input-value"> Enter Value</label>

      <input

        class="text-inputs"

        type="text"

        name="input-value"

        placeholder=" "

      />
  </div>
  <button class="submit-button">Update</button>
</form>
<form action="{{url_for('updatePassword')}}" method="post">
  <h3>Update Password</h3>
  <div class="field">
    <label class="custom-label" for="prev-password">
      Enter Old Password</label
    >
    <input
      class="text-inputs"
      type="password"
      name="prev-password"
      placeholder=" "
    />
  </div>
  <div class="field">
    <label class="custom-label" for="cur-password"> Enter New Password</label>
    <input
      class="text-inputs"
      type="password"
      name="cur-password"
      placeholder=" "
    />
  </div>
  <div class="field">
    <label class="custom-label" for="confirm-password">
      Enter Confirm Password</label
    >
    <input
      class="text-inputs"
      type="password"
      name="confirm-password"
      placeholder=" "
    />
  </div>
  <button class="submit-button">Update</button>
</form>
```

```
</div>

{% endblock%}
```

Result.html:

```
{% extends 'base.html '%}

{% block head %}

<title>Login page</title>

{% endblock%}

{%block body%}

<main class="container ">

  <div class="mx-auto mt-5 border bg-light login-card " style="width:500px;">

    <h2 class='mx-4 mt-2'>Password changed successfully</h2>

    <button class="submit-button">

      <a  href="/profile">Go Back</a>

    </button>

  </div>

</main>

</p>

</main>

{% endblock%}
```

Table.html:

```
<div class="table-wrapper">

 <table class="fl-table">

  <thead>

   {% for header in headings %}

   <th>{{header}}</th>

   {% endfor %}

  </thead>

  <tbody>

   {% for row in data %}

   <tr>

    {% for cell in row %}

    <td>{{row[cell]}}</td>

    {% endfor %}

   </tr>

   {% endfor %}

  </tbody>
  </table>
</div>
```

Base2.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi" crossorigin="anonymous" />
  <link rel="stylesheet" href="static/css/style.css" />
  {% block head%}{% endblock %}
</head>
<body>
  <div class="wrapper">
    <!-- Sidebar  -->
    <nav id="sidebar">
      <div class="sidebar-header">
        <h3>Inventory</h3>
      </div>
      <ul class="list-unstyled components">
        <li>
          <a href="/dashboard">Dashboard</a>
        </li>
        <li>
          <a href="/orders">Orders</a>
        </li>
        <li>
          <a href="/suppliers">Suppliers</a>
        </li>
        <li>
          <a href="/profile">Profile</a>
        </li>
        <li>
          <a href="/logout">logout</a>
        </li>
      </ul>

    </nav>

    <!-- Page Content  -->

    <div id="content">{% block body %} {% endblock %}</div>

  </div>

  <script src="static/js/app.js"></script>

</body>

</html>
```

App.py:

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

import sqlite3 as sql

from functools import wraps

import re

import ibm_db

import os

from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail

from datetime import datetime, timedelta


conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=815fa4db-dc03-4c70-869a-
a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30367;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.
crt;UID=gkx49901;PWD=kvWCsySl7vApfsy2", '', '')


app = Flask(__name__)

app.secret_key = 'jackiechan'



def rewrite(url):

    view_func, view_args = app.create_url_adapter(request).match(url)

    return app.view_functions[view_func](**view_args)



def login_required(f):

    @wraps(f)

    def decorated_function(*args, **kwargs):

        if "id" not in session:

            return redirect(url_for('login'))

        return f(*args, **kwargs)
    return decorated_function



@app.route('/')
def root():
    return render_template('login.html')



@app.route('/user/<id>')
@login_required
def user_info(id):
    with sql.connect('inventorymanagement.db') as con:
        con.row_factory = sql.Row
        cur = con.cursor()
        cur.execute(f'SELECT * FROM users WHERE email="{id}"')
        user = cur.fetchall()
    return render_template("user_info.html", user=user[0])



@app.route('/login', methods=['GET', 'POST'])
def login():
    global userid
    msg = ''

    if request.method == 'POST':
        un = request.form['username']
        pd = request.form['password_1']
        print(un, pd)
        sql = "SELECT * FROM users WHERE email =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, un)
        ibm_db.bind_param(stmt, 2, pd)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['loggedin'] = True
            session['id'] = account['EMAIL']
            userid = account['EMAIL']

            session['username'] = account['USERNAME']

            msg = 'Logged in successfully !'


            return rewrite('/dashboard')

        else:

            msg = 'Incorrect username / password !'

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



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

def signup():

    mg = ''

    if request.method == "POST":

        username = request.form['username']

        email = request.form['email']

        pw = request.form['password']

        sql = 'SELECT * FROM users WHERE email =?'

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, email)

        ibm_db.execute(stmt)

        acnt = ibm_db.fetch_assoc(stmt)

        print(acnt)


        if acnt:

            mg = 'Account already exits!!'


        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

            mg = 'Please enter the avalid email address'

        elif not re.match(r'[A-Za-z0-9]+', username):

            ms = 'name must contain only character and number'

        else:

            insert_sql = 'INSERT INTO users (USERNAME,FIRSTNAME,LASTNAME,EMAIL,PASSWORD) VALUES (?,?,?,?,?)'

            pstmt = ibm_db.prepare(conn, insert_sql)

            ibm_db.bind_param(pstmt, 1, username)

            ibm_db.bind_param(pstmt, 2, "firstname")

            ibm_db.bind_param(pstmt, 3, "lastname")

            # ibm_db.bind_param(pstmt,4,"123456789")
            ibm_db.bind_param(pstmt, 4, email)
            ibm_db.bind_param(pstmt, 5, pw)
            print(pstmt)
            ibm_db.execute(pstmt)
            mg = 'You have successfully registered click login!'
            message = Mail(
                from_email=os.environ.get('MAIL_DEFAULT_SENDER'),
                to_emails=email,
                subject='New SignUp',
                html_content='<p>Hello, Your Registration was successfull. <br><br> Thank you for choosing us.</p>')

            sg = SendGridAPIClient(
                api_key=os.environ.get('SENDGRID_API_KEY'))

            response = sg.send(message)
            print(response.status_code, response.body)
            return render_template("login.html", meg=mg)

    elif request.method == 'POST':
        msg = "fill out the form first!"
    return render_template("signup.html", meg=mg)



@app.route('/dashboard', methods=['POST', 'GET'])
@login_required
def dashBoard():
    sql = "SELECT * FROM stocks"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_assoc(stmt)
    stocks = []
    headings = [*dictionary]
    while dictionary != False:
        stocks.append(dictionary)
        # print(f"The ID is : ", dictionary["NAME"])
        # print(f"The name is : ", dictionary["QUANTITY"])
        dictionary = ibm_db.fetch_assoc(stmt)

    return render_template("dashboard.html", headings=headings, data=stocks)
```

```
@app.route('/addstocks', methods=['POST'])
@login_required
def addStocks():
    if request.method == "POST":
        print(request.form['item'])
        try:
            item = request.form['item']
            quantity = request.form['quantity']
            price = request.form['price']
            total = int(price) * int(quantity)
            insert_sql = 'INSERT INTO stocks (NAME,QUANTITY,PRICE_PER_QUANTITY,TOTAL_PRICE) VALUES (?,?,?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.bind_param(pstmt, 2, quantity)
            ibm_db.bind_param(pstmt, 3, price)
            ibm_db.bind_param(pstmt, 4, total)
            ibm_db.execute(pstmt)

        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('dashBoard'))




@app.route('/updatestocks', methods=['POST'])
@login_required
def UpdateStocks():
    if request.method == "POST":
        try:
            item = request.form['item']
            print("hello")
            field = request.form['input-field']
            value = request.form['input-value']
            print(item, field, value)

            insert_sql = 'UPDATE stocks SET ' + field + "= ?" + " WHERE NAME=?"

            print(insert_sql)

            pstmt = ibm_db.prepare(conn, insert_sql)

            ibm_db.bind_param(pstmt, 1, value)

            ibm_db.bind_param(pstmt, 2, item)

            ibm_db.execute(pstmt)

            if field == 'PRICE_PER_QUANTITY' or field == 'QUANTITY':

                insert_sql = 'SELECT * FROM stocks WHERE NAME= ?'

                pstmt = ibm_db.prepare(conn, insert_sql)

                ibm_db.bind_param(pstmt, 1, item)

                ibm_db.execute(pstmt)

                dictonary = ibm_db.fetch_assoc(pstmt)

                print(dictonary)

                total = dictonary['QUANTITY'] * dictonary['PRICE_PER_QUANTITY']

                insert_sql = 'UPDATE stocks SET TOTAL_PRICE=? WHERE NAME=?'

                pstmt = ibm_db.prepare(conn, insert_sql)

                ibm_db.bind_param(pstmt, 1, total)

                ibm_db.bind_param(pstmt, 2, item)

                ibm_db.execute(pstmt)

        except Exception as e:

            msg = e


        finally:

            # print(msg)

            return redirect(url_for('dashBoard'))



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

@login_required

def deleteStocks():

    if request.method == "POST":

        print(request.form['item'])

        try:

            item = request.form['item']

            insert_sql = 'DELETE FROM stocks WHERE NAME=?'

            pstmt = ibm_db.prepare(conn, insert_sql)

            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('dashBoard'))



@app.route('/update-user', methods=['POST', 'GET'])
@login_required
def updateUser():
    if request.method == "POST":
        try:
            email = session['id']
            field = request.form['input-field']
            value = request.form['input-value']
            insert_sql = 'UPDATE users SET ' + field + '= ? WHERE EMAIL=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, email)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('profile'))



@app.route('/update-password', methods=['POST', 'GET'])
@login_required
def updatePassword():
    if request.method == "POST":
        try:
            email = session['id']
            password = request.form['prev-password']
            curPassword = request.form['cur-password']
            confirmPassword = request.form['confirm-password']
            insert_sql = 'SELECT * FROM  users WHERE EMAIL=? AND PASSWORD=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, email)
            ibm_db.bind_param(pstmt, 2, password)
            ibm_db.execute(pstmt)
            dictionary = ibm_db.fetch_assoc(pstmt)
            print(dictionary)
            if curPassword == confirmPassword:
                insert_sql = 'UPDATE users SET PASSWORD=? WHERE EMAIL=?'
                pstmt = ibm_db.prepare(conn, insert_sql)
                ibm_db.bind_param(pstmt, 1, confirmPassword)
                ibm_db.bind_param(pstmt, 2, email)
                ibm_db.execute(pstmt)
        except Exception as e:
            msg = e
        finally:
            # print(msg)
            return render_template('result.html')


@app.route('/orders', methods=['POST', 'GET'])
@login_required
def orders():
    query = "SELECT * FROM orders"
    stmt = ibm_db.exec_immediate(conn, query)
    dictionary = ibm_db.fetch_assoc(stmt)
    orders = []
    headings = [*dictionary]
    while dictionary != False:
        orders.append(dictionary)
        dictionary = ibm_db.fetch_assoc(stmt)
    return render_template("orders.html", headings=headings, data=orders)


@app.route('/createOrder', methods=['POST'])
@login_required
def createOrder():
    if request.method == "POST":
        try:
            stock_id = request.form['stock_id']
            query = 'SELECT PRICE_PER_QUANTITY FROM stocks WHERE ID= ?'
            stmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(stmt, 1, stock_id)
            ibm_db.execute(stmt)
            dictionary = ibm_db.fetch_assoc(stmt)
            if dictionary:
                quantity = request.form['quantity']
                date = str(datetime.now().year) + "-" + str(
                    datetime.now().month) + "-" + str(datetime.now().day)
                delivery = datetime.now() + timedelta(days=7)
                delivery_date = str(delivery.year) + "-" + str(
                    delivery.month) + "-" + str(delivery.day)
                price = float(quantity) * \
                    float(dictionary['PRICE_PER_QUANTITY'])
                query = 'INSERT INTO orders (STOCKS_ID,QUANTITY,DATE,DELIVERY_DATE,PRICE) VALUES (?,?,?,?,?)'
                pstmt = ibm_db.prepare(conn, query)
                ibm_db.bind_param(pstmt, 1, stock_id)
                ibm_db.bind_param(pstmt, 2, quantity)
                ibm_db.bind_param(pstmt, 3, date)
                ibm_db.bind_param(pstmt, 4, delivery_date)
                ibm_db.bind_param(pstmt, 5, price)
                ibm_db.execute(pstmt)
        except Exception as e:
            print(e)

        finally:
            return redirect(url_for('orders'))


@app.route('/updateOrder', methods=['POST'])
@login_required
def updateOrder():
    if request.method == "POST":
        try:
            item = request.form['item']
            field = request.form['input-field']

            value = request.form['input-value']

            query = 'UPDATE orders SET ' + field + "= ?" + " WHERE ID=?"

            pstmt = ibm_db.prepare(conn, query)

            ibm_db.bind_param(pstmt, 1, value)

            ibm_db.bind_param(pstmt, 2, item)

            ibm_db.execute(pstmt)

        except Exception as e:

            print(e)


        finally:

            return redirect(url_for('orders'))



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

@login_required

def cancelOrder():

    if request.method == "POST":

        try:

            order_id = request.form['order_id']

            query = 'DELETE FROM orders WHERE ID=?'

            pstmt = ibm_db.prepare(conn, query)

            ibm_db.bind_param(pstmt, 1, order_id)

            ibm_db.execute(pstmt)

        except Exception as e:

            print(e)


        finally:

            return redirect(url_for('orders'))



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

@login_required

def suppliers():

    sql = "SELECT * FROM suppliers"

    stmt = ibm_db.exec_immediate(conn, sql)

    dictionary = ibm_db.fetch_assoc(stmt)

    suppliers = []
    orders_assigned = []
    headings = [*dictionary]
    while dictionary != False:
        suppliers.append(dictionary)
        orders_assigned.append(dictionary['ORDER_ID'])
        dictionary = ibm_db.fetch_assoc(stmt)

# get order ids from orders table and identify unassigned order ids
    sql = "SELECT ID FROM orders"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_assoc(stmt)
    order_ids = []
    while dictionary != False:
        order_ids.append(dictionary['ID'])
        dictionary = ibm_db.fetch_assoc(stmt)

    unassigned_order_ids = set(order_ids) - set(orders_assigned)
    return render_template("suppliers.html",headings=headings,data=suppliers,order_ids=unassigned_order_ids)

@app.route('/updatesupplier', methods=['POST'])
@login_required
def UpdateSupplier():
    if request.method == "POST":
        try:
            item = request.form['name']
            field = request.form['input-field']
            value = request.form['input-value']
            print(item, field, value)
            insert_sql = 'UPDATE suppliers SET ' + field + "= ?" + " WHERE NAME=?"
            print(insert_sql)
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            return redirect(url_for('suppliers'))


@app.route('/addsupplier', methods=['POST'])
@login_required
def addSupplier():
    if request.method == "POST":
        try:
            name = request.form['name']
            order_id = request.form.get('order-id-select')
            print(order_id)
            print("Hello world")
            location = request.form['location']
            insert_sql = 'INSERT INTO suppliers (NAME,ORDER_ID,LOCATION) VALUES (?,?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, name)
            ibm_db.bind_param(pstmt, 2, order_id)
            ibm_db.bind_param(pstmt, 3, location)
            ibm_db.execute(pstmt)

        except Exception as e:
            msg = e

        finally:
            return redirect(url_for('suppliers'))


@app.route('/deletesupplier', methods=['POST'])
@login_required
def deleteSupplier():
    if request.method == "POST":
        try:
            item = request.form['name']
            insert_sql = 'DELETE FROM suppliers WHERE NAME=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e
        finally:
            return redirect(url_for('suppliers'))
@app.route('/profile', methods=['POST', 'GET'])
@login_required
def profile():
    if request.method == "GET":
        try:
            email = session['id']
            insert_sql = 'SELECT * FROM users WHERE EMAIL=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, email)
            ibm_db.execute(pstmt)
            dictionary = ibm_db.fetch_assoc(pstmt)
            print(dictionary)
        except Exception as e:
            msg = e
        finally:
            # print(msg)
            return render_template("profile.html", data=dictionary)


@app.route('/logout', methods=['GET'])
@login_required
def logout():
    print(request)
    resp = make_response(render_template("login.html"))
    session.clear()
    return resp


if __name__ == '__main__':
    app.run(debug=True)

# ALTER TABLE stocks ALTER COLUMN ID SET GENERATED BY DEFAULT AS IDENTITY
```