Project Development Phase

Sprint 2

Front-End:

Css:

```
@import 'https://fonts.googleapis.com/css?family=Poppins:300,400,500,600,700';
body {
  font-family: 'Poppins', sans-serif;
  background: #fafafa;
}
p {
  font-family: 'Poppins', sans-serif;
  font-size: 1.1em;
  font-weight: 300;
  line-height: 1.7em;
  color: #999;
}
a,
a:hover,
a:focus {
  color: inherit;
  text-decoration: none;
  transition: all 0.3s;
}
.navbar {
  padding: 15px 10px;
  background: #fff;
  border: none;
  border-radius: 0;
  margin-bottom: 40px;
  box-shadow: 1px 1px 3px rgba(0, 0, 0, 0.1);
}
.navbar-btn {
```

```
  box-shadow: none;
  outline: none !important;
  border: none;
}
.line {
  width: 100%;
  height: 1px;
  border-bottom: 1px dashed #ddd;
  margin: 40px 0;
}
.wrapper {
  display: flex;
  width: 100%;
  align-items: stretch;
}
#sidebar {
  min-width: 250px;
  max-width: 250px;
  background: #48494b;
  color: #fff;
  transition: all 0.3s;
}
#sidebar.active {
  margin-left: -250px;
}
#sidebar .sidebar-header {
  padding: 20px;
  background: #48494b;
}
#sidebar ul.components {
  padding: 20px 0;
  border-bottom: 1px solid #47748b;
}
#sidebar ul p {
  color: #fff;
```

```
  padding: 10px;
}
.project-title {
  font-size: 20px;
  padding-left: 10px;
  text-align: center;
}
#sidebar ul li a {
  padding: 10px;
  font-size: 1.1em;
  display: block;
}
#sidebar ul li a:hover {
  color: #7386d5;
  background: #fff;
}
#sidebar ul li.active > a,
a[aria-expanded='true'] {
  color: #fff;
  background: #48494b;
}
a[data-toggle='collapse'] {
  position: relative;
}
.dropdown-toggle::after {
  display: block;
  position: absolute;
  top: 50%;
  right: 20px;
  transform: translateY(-50%);
}
ul ul a {
  font-size: 0.9em !important;
  padding-left: 30px !important;
  background: #48494b;
```

```
}
ul.CTAs {
  padding: 20px;
}
ul.CTAs a {
  text-align: center;
  font-size: 0.9em !important;
  display: block;
  border-radius: 5px;
  margin-bottom: 5px;
}
a.download {
  background: #fff;
  color: #48494b;
}
a.article,
a.article:hover {
  background: #48494b !important;
  color: #fff !important;
}
.login-card {
  box-shadow: rgba(0, 0, 0, 0.35) 0px 5px 15px;
  border-radius: 10px;
  padding: 10px;
}
.login-card p {
  padding-left: 20px;
}
.login-card a {
  color: rgba(84, 84, 220, 0.888);
}
#content {
  width: 100%;
  padding: 20px;
  min-height: 100vh;
```

```
  transition: all 0.3s;
}
@media (max-width: 768px) {
  #sidebar {
    margin-left: -250px;
  }
  #sidebar.active {
    margin-left: 0;
  }
  #sidebarCollapse span {
    display: none;
  }
}
/* Table Styles */
.table-wrapper {
  margin: 10px 70px 70px;
  box-shadow: rgba(99, 99, 99, 0.2) 0px 2px 8px 0px;
}
.fl-table {
  border-radius: 5px;
  font-size: 16px;
  font-weight: normal;
  border: none;
  border-collapse: collapse;
  width: 100%;
  max-width: 100%;
  white-space: nowrap;
  background-color: white;
}
.fl-table td,
.fl-table th {
  text-align: center;
  padding: 8px;
}
.fl-table td {
```

```
  border-right: 1px solid #f8f8f8;
  font-size: 16px;
}
.fl-table thead th {
  color: #ffffff;
  background: #68716e !important ;
}
.fl-table thead:nth-child(odd) {
  color: #ffffff;
  background: #324960;
}
.fl-table tr:nth-child(even) {
  background: #f8f8f8;
}
.custom-label {
  font-size: 18px;
  font-weight: 400;
}
.field input[type='text'] {
  /* width: 100%; */
  border: 2px solid #aaa;
  border-radius: 4px;
  /* margin: 8px 0; */
  outline: none;
  padding: 2px 10px;
  box-sizing: border-box;
  transition: 0.3s;
}
.field input[type='number'] {
  /* width: 100%; */
  border: 2px solid #aaa;
  border-radius: 4px;
  /* margin: 8px 0; */
  outline: none;
  padding: 2px 10px;
```

```
  box-sizing: border-box;
  transition: 0.3s;
}
.submit-button {
  padding: 5px 10px;
  color: white;
  background-color: rgb(41, 115, 41);
  border: none;
  border-radius: 8px;
  min-width: 100px;
}
.submit-button a {
  color: white;
}
.mg-20 {
  margin-top: 20px;
}
.user-deatils h4 {
  font-size: 18px;
}
/* .field input[type='text']:focus {
  border-color: rgba(59, 67, 75, 0.687);
  box-shadow: 0 0 8px 0 rgba(80, 94, 108, 0.667);
} */
.field {
  display: flex;
  align-items: center;
  padding: 10px 0px;
}
.text-inputs {
  margin: 0px 10px;
}
/* Responsive */
@media (max-width: 767px) {
  .fl-table {
```

```
  display: block;
  width: 100%;
}
.table-wrapper:before {
  content: 'Scroll horizontally >';
  display: block;
  text-align: right;
  font-size: 11px;
  color: white;
  padding: 0 0 10px;
}
.fl-table thead,
.fl-table tbody,
.fl-table thead th {
  display: block;
}
.fl-table thead th:last-child {
  border-bottom: none;
}
.fl-table thead {
  float: left;
}
.fl-table tbody {
  width: auto;
  position: relative;
  overflow-x: auto;
}
.fl-table td,
.fl-table th {
  padding: 20px 0.625em 0.625em 0.625em;
  height: 60px;
  vertical-align: middle;
  box-sizing: border-box;
  overflow-x: hidden;
  overflow-y: auto;
```

```
    width: 120px;
    font-size: 13px;
    text-overflow: ellipsis;
  }
  .fl-table thead th {
    text-align: left;
    border-bottom: 1px solid #f7f7f9;
  }
  .fl-table tbody tr {
    display: table-cell;
  }
  .fl-table tbody tr:nth-child(odd) {
    background: none;
  }
  .fl-table tr:nth-child(even) {
    background: transparent;
  }
  .fl-table tr td:nth-child(odd) {
    background: #f8f8f8;
    border-right: 1px solid #e6e4e4;
}
  .fl-table tr td:nth-child(even) {
    border-right: 1px solid #e6e4e4;
  }
  .fl-table tbody td {
    display: block;
    text-align: center;
  }
}
.forms-wrapper {
  display: flex;
  /* align-items: center; */
  justify-content: space-around;
}
.red-button {
```

```
  background-color: rgb(186, 13, 13);
}
```

Javascript:

```
const selectedItem = document
  .getElementById('sidebarCollapse')
  .addEventListener('click', (e) => {
    const ele = document.getElementById('sidebar');
    ele.classList.toggle('active');
  });
```

Templates:

Orders.html:

```
{% extends 'base2.html '%} {% block head %}
<title>Orders</title>
{% endblock%} {%block body%}
<h2>Orders</h2>
<p>
  Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
</p>
{% include 'table.html' %}
<div class="forms-wrapper">
  <form action="{{url_for('createOrder') }}" method="post">
    <h3>Create Order</h3>
    <div class="field">
      <label class="custom-label" for="item">Enter Stock ID</label>
      <input class="text-inputs" type="text" name="stock_id" placeholder="1" />
    </div>
    <div class="field">
      <label class="custom-label" for="item">Enter Quantity</label>
      <input class="text-inputs" type="number" name="quantity" placeholder="10" />
    </div>
    <button class="submit-button">Create</button>
  </form>

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

```
    <h3>Update Order</h3>
    <div class="field">
      <label class="custom-label" for="item">Enter Order ID</label>
      <input class="text-inputs" name="item" type="number" placeholder="1" />
    </div>
    <div class="field">
      <label for="custom-label" for="input-field">Choose a field - </label>
      <select name="input-field" id="field">
        <option value="STOCKS_ID">STOCKS_ID</option>
        <option value="QUANTITY">QUANTITY</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" />
    </div>
    <button class="submit-button">Update</button>
  </form>
  <form action="{{url_for('cancelOrder') }}" method="post">
    <h3>Cancel Order</h3>
    <div class="field">
      <label class="custom-label" for="item">Enter Order ID</label>
      <input class="text-inputs" name="order_id" type="number" placeholder="1" />
    </div>
    <button class="submit-button red-button">Cancel</button>
  </form>
</div>
{% endblock%}
```

Suppliers.html:

```
{% extends 'base2.html '%} {% block head %}
<title>Suppliers</title>
{% endblock%} {%block body%}
<h2>Suppliers</h2>
<p>
  Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
</p>
```

```
{% include 'table.html' %}
<div class="forms-wrapper">
 <form action="{{url_for('UpdateSupplier') }}" method="post">
  <h3>Update Supplier</h3>
  <div class="field">
   <label class="custom-label" for="name"> Enter Name</label>
   <input class="text-inputs" type="text" name="name" placeholder="Supplier name" />
  </div>
  <div class="field">
   <label for="input-field">Choose a field :</label>
   <select name="input-field" id="field">
    <option value="NAME">NAME</option>
    <option value="LOCATION">LOCATION</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('addSupplier') }}" method="post">
  <h3>Add New Supplier</h3>
  <div class="field">
   <label class="custom-label" for="name"> Enter the Supplier</label>
   <input class="text-inputs" name="name" type="text" placeholder="Supplier name" />
  </div>
  <div class="field">
   <label class="custom-label" for="quantity"> Enter Order ID : </label>
   <select name="order-id-select" id="field">
    {% for order_id in order_ids %}
     <option value="{{ order_id }}">{{order_id}}</option>
```

```
      {% endfor %}
    </select>
   </div>
   <div class="field">
    <label class="custom-label" for="location"> Enter Location</label>
    <input class="text-inputs" type="text" name="location" placeholder="Location" />
   </div>
   <button class="submit-button">Add Stock</button>
  </form>
  <form action="{{url_for('deleteSupplier') }}" method="post">
   <h3>Delete Supplier</h3>
   <div class="field">
    <label class="custom-label" for="name"> Enter the name</label>
    <input class="text-inputs" name="name" type="text" placeholder="Supplier Name" />
   </div>
   <button class="submit-button red-button">Delete</button>
  </form>
</div>

{% endblock%}
```

Dashboard.html:

```
{% extends 'base2.html '%} {% block head %}
<title>Dashboard</title>
{% endblock%} {%block body%}
<h2>Dashboard</h2>
<p>
  Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
</p>
{% include 'table.html' %}
<div class="forms-wrapper">
  <form action="{{url_for('UpdateStocks') }}" method="post">
   <h3>Update Stock</h3>
   <div class="field">
    <label class="custom-label" for="item"> Enter Item</label>
    <input class="text-inputs" type="text" name="item" placeholder="milk" />
   </div>
```

```
 <div class="field">
  <label for="input-field">Choose a field :</label>
  <select name="input-field" id="field">
   <option value="NAME">NAME</option>
   <option value="PRICE_PER_QUANTITY">PRICE_PER_QUANTITY</option>
   <option value="QUANTITY">QUANTITY</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('addStocks') }}" method="post">
 <h3>Add New Stock</h3>
 <div class="field">
  <label class="custom-label" for="item"> Enter the item</label>
  <input class="text-inputs" name="item" type="text" placeholder="juice" />
 </div>
 <div class="field">
  <label class="custom-label" for="quantity"> Enter quantity</label>
  <input
   class="text-inputs"
   type="number"
   name="quantity"
   placeholder="200"
  />
 </div>
 <div class="field">
  <label class="custom-label" for="price"> Enter price</label>
  <input class="text-inputs" type="number" name="price" placeholder="25" />
```

```
  </div>
  <button class="submit-button">Add Stock</button>
 </form>
 <form action="{{url_for('deleteStocks') }}" method="post">
  <h3>Remove stocks</h3>
  <div class="field">
   <label class="custom-label" for="item"> Enter the item</label>
   <input class="text-inputs" name="item" type="text" placeholder="juice" />
  </div>
  <button class="submit-button red-button">Remove</button>
 </form>
</div>

{% endblock%}
```

Back-End:

Python File:

```
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=21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud ;PORT=31864;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
```