|  |  |
| --- | --- |
| Date | 11-November 2022 |
| Team ID | PNT2022TMID44318 |
| Project Name | Inventory Management System for Retailers |
| Batch number | B8-2A4E |

# PROJECT DEVELOPMENT PHASE-SPRINT 3

**Products.html**

<!doctype html>

<html class="no-js" lang="zxx">

<head>

<meta charset="utf-8">

<meta http-equiv="x-ua-compatible" content="ie=edge">

<meta name="description" content="">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap Css & Js -->

<link href="[https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css](https://cdn.jsdelivr.net/npm/bootstrap%405.1.3/dist/css/bootstrap.min.css)" rel="stylesheet" integrity="sha384- 1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

<script src="[https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js](https://cdn.jsdelivr.net/npm/bootstrap%405.1.3/dist/js/bootstrap.bundle.min.js)" integrity="sha384- ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

<!-- CSS here -->

<link href="static/css/mystyle.css" rel="Stylesheet" />

<style>

.shadow-demo { width: 100px; height: 100px;

background-color: #fff;

}

.shadow-demo-1 { width: 100px; height: 100px;

background-color: #ccc;

}

.shadow-demo-2 { width: 100px; height: 100px;

background-color: #000;

}

.mask-custom {

background-color: rgba(255, 255, 255, 0.2);

border-radius: 10;

border: 0;

background-clip: padding-box;

box-shadow: 10px 10px 10px rgba(46, 54, 68, 0.03);

}

.custom-1 {

backdrop-filter: blur(30px);

}

.custom-2 {

backdrop-filter: blur(60px);

}

.custom-3 {

backdrop-filter: blur(40px);

}

.custom-4 {

backdrop-filter: blur(15px);

}

.custom-5 {

backdrop-filter: blur(5px);

}

.mask-custom-1 {

background-color: rgba(0, 0, 0, 0.2);

border-radius: 20;

border: 0;

background-clip: padding-box;

box-shadow: 10px 10px 10px rgba(46, 54, 68, 0.03);

}

.custom-6 {

backdrop-filter: blur(30px);

}

.custom-7 {

backdrop-filter: blur(60px);

}

.custom-8 {

backdrop-filter: blur(40px);

}

.custom-9 {

backdrop-filter: blur(15px);

}

.custom-10 {

backdrop-filter: blur(5px);

}

</style>

<!-- JS here -->

{% block head %} {% endblock %}

<script> window.watsonAssistantChatOptions = {

integrationID: "633fc278-0dda-417b-9c10-bd2f300b411a", // The ID of this integration. region: "jp-tok", // The region your integration is hosted in.

serviceInstanceID: "b7ec50cd-af28-4bb0-aa53-52dc00c34d4e", // The ID of your service instance.

onLoad: function(instance) { instance.render(); }

};

setTimeout(function(){

const t=document.createElement('script');

t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" + (window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";

document.head.appendChild(t);

});

</script>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<div class="container-fluid">

<a class="navbar-brand" href="/">IMS</a>

<button class="navbar-toggler" type="button" data-bs-toggle="offcanvas" data-bs- target="#offcanvasNavbar" aria-controls="offcanvasNavbar">

<span class="navbar-toggler-icon"></span>

</button>

<div class="offcanvas offcanvas-end" tabindex="-1" id="offcanvasNavbar" aria- labelledby="offcanvasNavbarLabel">

<div class="offcanvas-header">

<h5 class="offcanvas-title" id="offcanvasNavbarLabel">Offcanvas</h5>

<button type="button" class="btn-close text-reset" data-bs-dismiss="offcanvas" aria- label="Close"></button>

</div>

<div class="offcanvas-body">

<ul class="navbar-nav justify-content-end flex-grow-1 pe-3">

<li class="nav-item">

<a class="nav-link" aria-current="page" href="/">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/register" >Register</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/login">Login</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/list" >List</a>

</li>

</ul>

</div>

</div>

</nav>

<!--

<nav class="navbar navbar-light bg-light fixed-top">

<div class="container-fluid"> Page navigation

<a class="nav-link" aria-current="page" href="/">Home</a>

<a class="nav-link" href="/register" >Register</a>

<a class="nav-link" href="/login">Login</a>

<a class="nav-link" href="/logout">Logout</a>

<a class="nav-link" href="/list" >List</a>

<button class="navbar-toggler" type="button" data-bs-toggle="offcanvas" data- bs-target="#offcanvasNavbar" aria-controls="offcanvasNavbar">

<span class="navbar-toggler-icon"></span>

</button>

<div class="offcanvas offcanvas-end" tabindex="-1" id="offcanvasNavbar" aria- labelledby="offcanvasNavbarLabel">

<div class="offcanvas-header">

<h5 class="offcanvas-title" id="offcanvasNavbarLabel">Offcanvas</h5>

<button type="button" class="btn-close text-reset" data-bs-dismiss="offcanvas" aria- label="Close"></button>

</div>

<div class="offcanvas-body">

<ul class="navbar-nav justify-content-end flex-grow-1 pe-3">

<li class="nav-item">

<a class="nav-link" aria-current="page" href="/">Home</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/register" >Register</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/login">Login</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/logout">Logout</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/list" >List</a>

</li>

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#" id="offcanvasNavbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">

Dropdown

</a>

<ul class="dropdown-menu" aria-labelledby="offcanvasNavbarDropdown">

<li><a class="dropdown-item" href="#">Action</a></li>

<li><a class="dropdown-item" href="#">Another action</a></li>

<li>

<hr class="dropdown-divider">

</li>

<li><a class="dropdown-item" href="#">Something else here</a></li>

</ul>

</li>

</ul>

<form class="d-flex">

<input class="form-control me-2" type="search" placeholder="Search" aria- label="Search">

<button class="btn btn-outline-success" type="submit">Search</button>

</form>

</div>

</div>

</div>

</nav> -->

{% block body %} {% endblock %}

</body>

</html>

# Add product.html

<!DOCTYPE html >

<head>

<meta charset="utf-8">

<meta http-equiv="x-ua-compatible" content="ie=edge">

<meta name="description" content="">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- Bootstrap Css & Js -->

<link href="[https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css](https://cdn.jsdelivr.net/npm/bootstrap%405.1.3/dist/css/bootstrap.min.css)" rel="stylesheet" integrity="sha384- 1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

<script src="[https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js](https://cdn.jsdelivr.net/npm/bootstrap%405.1.3/dist/js/bootstrap.bundle.min.js)" integrity="sha384- ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p" crossorigin="anonymous"></script>

<style> html,body

{

height: 100%;

margin: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

</style>

<!-- CSS here -->

<link href="static/css/mystyle.css" rel="Stylesheet" />

<body>

<div style="background-image: url('static/img/Secure login-rafiki.png');background- position: center; background-repeat: no-repeat; background-size: contain; background-repeat: no-repeat; height: 100%;">

<h1 class="display-6" style="text-align: center;">We have sent a confirmation mail to your registerd E-mail.</h1>

<h1 class="display-6" style="text-align: center;"> Please confirm the mail to continue Registration.</h1>

</div>

</body>

</html>

# Config.py

import datetime import os

from dotenv import load\_dotenv load\_dotenv()

basedir = os.path.abspath(os.path.dirname(\_\_file ))

APP\_SETTINGS = os.getenv('APP\_SETTINGS', 'config.DevelopmentConfig') class Config():

EMAIL\_CONFIRMATION\_SENDER\_EMAIL = os.getenv( 'EMAIL\_CONFIRMATION\_SENDER\_EMAIL')

EMAIL\_CONFIRMATION\_SALT = 'email-confirmation' EMAIL\_CONFIRMATION\_TOKEN\_MAX\_AGE\_SECONDS = 300 JSON\_SORT\_KEYS = False

JWT\_ACCESS\_TOKEN\_EXPIRES = datetime.timedelta(minutes=60) SECRET\_KEY = os.getenv('SECRET\_KEY', os.urandom(32)) SENDGRID\_API\_KEY = os.getenv('SENDGRID\_API\_KEY') SQLALCHEMY\_TRACK\_MODIFICATIONS = False WTF\_CSRF\_ENABLED = False

class DevelopmentConfig(Config): DEBUG = True JSON\_SORT\_KEYS = True

SQLALCHEMY\_ECHO = True

SQLALCHEMY\_DATABASE\_URI = f'sqlite:///{os.path.join(basedir, "app.db")}' class ProductionConfig(Config):

DEBUG = False

SQLALCHEMY\_DATABASE\_URI = os.getenv('DB\_URL')

# App.py

from turtle import st

from flask import Flask, render\_template, request, redirect, url\_for, session from markupsafe import escape

import ibm\_db

conn = ibm\_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-8954- 7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;SECURITY

=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lkc93724;PWD=zAzNGa6Da Nk6xvle",'','')

import smtplib, ssl

## email.mime subclasses

from email.mime.multipart import MIMEMultipart from email.mime.text import MIMEText

## The pandas library is only for generating the current date, which is not necessary for sending emails

import pandas as pd

from datetime import datetime from flask import Flask

app = Flask( name ) var\_list = []

app.secret\_key = 'your secret key' @app.route('/')

def home():

if not session.get("name"):

return render\_template('home.html')

return render\_template('home.html', session = session) @app.route('/register')

def new\_student():

return render\_template('Register.html') @app.route('/addrec',methods = ['POST', 'GET']) def addrec():

if request.method == 'POST':

fname = request.form['fname'] lname = request.form['lname'] cname = request.form['cname'] state = request.form['state'] city = request.form['city']

mobileno = request.form['mobileno'] emailid = request.form['emailid'] password = request.form['password'] pincode = request.form['pincode']

sql = "SELECT \* FROM Users WHERE EMAILID =?"

stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,emailid) ibm\_db.execute(stmt)

account = ibm\_db.fetch\_assoc(stmt) if account:

users = []

sql = "SELECT \* FROM Users"

stmt = ibm\_db.exec\_immediate(conn, sql) dictionary = ibm\_db.fetch\_both(stmt)

while dictionary != False:

# print ("The Name is : ", dictionary) users.append(dictionary)

dictionary = ibm\_db.fetch\_both(stmt)

return render\_template('list.html', msg="You are already a member, please login using your details", users = users)

else:

var\_list.append(fname) var\_list.append(lname) var\_list.append(cname) var\_list.append(state) var\_list.append(city) var\_list.append(mobileno) var\_list.append(emailid) var\_list.append(password) var\_list.append(pincode)

bodytemp = r"D:\IBM\GUIDED PROJECT\INVENTORY MANAGEMENT SYSTEM

FOR RETAILERS\SPRINT 2\templates\email.html" with open(bodytemp, "r", encoding='utf-8') as f:

html= f.read()

# Set up the email addresses and password. Please replace below with your email address and password

email\_from = 'padhu10a@gmail.com' epassword = 'rbjibzkssszsbrjo' email\_to = emailid

# Generate today's date to be included in the email Subject date\_str = pd.Timestamp.today().strftime('%Y-%m-%d')

# Create a MIMEMultipart class, and set up the From, To, Subject fields email\_message = MIMEMultipart()

email\_message['From'] = email\_from

email\_message['To'] = email\_to email\_message['Subject'] = f'Report email - {date\_str}'

# Attach the html doc defined earlier, as a MIMEText html content type to the MIME message

email\_message.attach(MIMEText(html, "html")) # Convert it as a string

email\_string = email\_message.as\_string()

# Connect to the Gmail SMTP server and Send Email context = ssl.create\_default\_context()

with smtplib.SMTP\_SSL("smtp.gmail.com", 465, context=context) as server: server.login(email\_from, epassword)

server.sendmail(email\_from, email\_to, email\_string) return render\_template('notify.html')

@app.route('/confirm') def confirnation():

insert\_sql = "INSERT INTO Users (FIRSTNAME, LASTNAME, COMPANYNAME, STATE, CITY, MOBILENO, EMAILID, PASSWORD, PINCODE) VALUES (?,?,?,?,?,?,?,?,?)"

prep\_stmt = ibm\_db.prepare(conn, insert\_sql) ibm\_db.bind\_param(prep\_stmt, 1, var\_list[0])

ibm\_db.bind\_param(prep\_stmt, 2, var\_list[1])

ibm\_db.bind\_param(prep\_stmt, 3, var\_list[2])

ibm\_db.bind\_param(prep\_stmt, 4, var\_list[3])

ibm\_db.bind\_param(prep\_stmt, 5, var\_list[4])

ibm\_db.bind\_param(prep\_stmt, 6, var\_list[5])

ibm\_db.bind\_param(prep\_stmt, 7, var\_list[6])

ibm\_db.bind\_param(prep\_stmt, 8, var\_list[7])

ibm\_db.bind\_param(prep\_stmt, 9, var\_list[8]) ibm\_db.execute(prep\_stmt)

return render\_template('confirm.html')

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

msg = ''

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

password = request.form['password']

sql = "SELECT \* FROM Users WHERE EMAILID =? AND PASSWORD =?"

stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,email) ibm\_db.bind\_param(stmt,2,password) ibm\_db.execute(stmt)

account = ibm\_db.fetch\_assoc(stmt) if account:

session['loggedin'] = True session['id'] = account['ID']

session['email'] = account['EMAILID'] session['name'] = account['FIRSTNAME'] msg = 'Logged in successfully !'

return render\_template('dashboard/dashboard.html', msg = msg) else:

msg = 'Incorrect email / password !'

return render\_template('login.html', msg = msg) @app.route('/dashboard')

def dashboard():

return render\_template('dashboard/dashboard.html') @app.route('/addproduct')

def addproduct():

return render\_template('dashboard/addproduct.html') @app.route('/movement')

def movement(): products = []

sql = "SELECT \* FROM Products WHERE HOLDERNAME = ?"

prep\_stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(prep\_stmt, 1, session['name']) ibm\_db.execute(prep\_stmt)

dictionary = ibm\_db.fetch\_both(prep\_stmt) while dictionary != False:

# print ("The Name is : ", dictionary) products.append(dictionary)

dictionary = ibm\_db.fetch\_both(prep\_stmt) if products:

return render\_template("dashboard/movement.html", products = products , session = session)

else:

return render\_template("dashboard/movement.html") @app.route('/moveproc',methods = ['POST', 'GET'])

def moveproc():

if request.method == 'POST': pname = request.form['pname']

quantityout = request.form['quantityout'] tow = request.form['to']

insert\_sql = "UPDATE products SET QUANTITYOUT = ?, TO = ? WHERE PRODUCTNAME = ? AND HOLDERNAME = ?;"

prep\_stmt = ibm\_db.prepare(conn, insert\_sql) ibm\_db.bind\_param(prep\_stmt, 1,quantityout) ibm\_db.bind\_param(prep\_stmt, 2, tow)

ibm\_db.bind\_param(prep\_stmt, 3, pname)

ibm\_db.bind\_param(prep\_stmt, 4, session['name']) ibm\_db.execute(prep\_stmt)

products = []

sql = "SELECT \* FROM Products WHERE HOLDERNAME = ?"

prep\_stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(prep\_stmt, 1, session['name']) ibm\_db.execute(prep\_stmt)

dictionary = ibm\_db.fetch\_both(prep\_stmt) while dictionary != False:

# print ("The Name is : ", dictionary) products.append(dictionary)

dictionary = ibm\_db.fetch\_both(prep\_stmt)

return render\_template('dashboard/movement.html', msg = "Product movement noted!", products = products)

@app.route('/report') def report():

return render\_template('dashboard/report.html') @app.route('/stockupdate')

def stock(): products = []

sql = "SELECT \* FROM Products WHERE HOLDERNAME = ?"

prep\_stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(prep\_stmt, 1, session['name']) ibm\_db.execute(prep\_stmt)

dictionary = ibm\_db.fetch\_both(prep\_stmt) while dictionary != False:

# print ("The Name is : ", dictionary) products.append(dictionary)

dictionary = ibm\_db.fetch\_both(prep\_stmt) if products:

return render\_template("dashboard/stockupdate.html", products = products , session = session)

else:

return render\_template("dashboard/stockupdate.html") @app.route('/proc\_delete', methods = ['POST', 'GET']) def proc\_delete():

id = request.args.get('pid')

delete\_sql = "DELETE FROM products WHERE ID = ? AND HOLDERNAME = ?;" prep\_stmt = ibm\_db.prepare(conn, delete\_sql)

ibm\_db.bind\_param(prep\_stmt, 1, id)

ibm\_db.bind\_param(prep\_stmt, 2, session['name']) ibm\_db.execute(prep\_stmt)

products = []

sql = "SELECT \* FROM Products WHERE HOLDERNAME = ?"

prep\_stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(prep\_stmt, 1, session['name']) ibm\_db.execute(prep\_stmt)

dictionary = ibm\_db.fetch\_both(prep\_stmt) while dictionary != False:

# print ("The Name is : ", dictionary) products.append(dictionary)

dictionary = ibm\_db.fetch\_both(prep\_stmt)

return render\_template('dashboard/stockupdate.html', msg='Product successfully deleted!' , products = products)

@app.route('/proc\_update', methods = ['POST', 'GET']) def proc\_update():

if request.method == 'POST':

pname = request.form['pname'] quantityin = request.form['quantityin'] pid = request.form['pid']

update\_sql = "UPDATE products SET PRODUCTNAME = ?, QUANTITYIN = ?

WHERE ID = ? AND HOLDERNAME = ?;"

prep\_stmt = ibm\_db.prepare(conn, update\_sql) ibm\_db.bind\_param(prep\_stmt, 1, pname)

ibm\_db.bind\_param(prep\_stmt, 2, quantityin)

ibm\_db.bind\_param(prep\_stmt, 3, pid)

ibm\_db.bind\_param(prep\_stmt, 4, session['name']) ibm\_db.execute(prep\_stmt)

products = []

sql = "SELECT \* FROM Products WHERE HOLDERNAME = ?"

prep\_stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(prep\_stmt, 1, session['name']) ibm\_db.execute(prep\_stmt)

dictionary = ibm\_db.fetch\_both(prep\_stmt) while dictionary != False:

# print ("The Name is : ", dictionary) products.append(dictionary)

dictionary = ibm\_db.fetch\_both(prep\_stmt)

return render\_template('dashboard/stockupdate.html', msg='Product successfully updated!' , products = products)

@app.route('/addproc',methods = ['POST', 'GET']) def addproc():

if request.method == 'POST': pname = request.form['pname']

quantity = request.form['quantity'] the\_time = datetime.now()

the\_time = the\_time.replace(second=0, microsecond=0)

sql = "SELECT \* FROM Products WHERE HOLDERNAME =?"

stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,session['name']) ibm\_db.execute(stmt)

product = ibm\_db.fetch\_assoc(stmt) if product:

if product['PRODUCTNAME']==pname:

return render\_template('dashboard/addproduct.html', msg="Product already added!

Add a new product.") else:

sql ="INSERT INTO Products (PRODUCTNAME,QUANTITYIN,QUANTITYOUT,TO,DATE,HOLDERNAME) VALUES (?,?,?,?,?,?);"

prep\_stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(prep\_stmt, 1, pname)

ibm\_db.bind\_param(prep\_stmt, 2, quantity)

ibm\_db.bind\_param(prep\_stmt, 3, '')

ibm\_db.bind\_param(prep\_stmt, 4, '')

ibm\_db.bind\_param(prep\_stmt, 5, str(the\_time))

ibm\_db.bind\_param(prep\_stmt, 6, session['name']) ibm\_db.execute(prep\_stmt)

return render\_template('dashboard/addproduct.html', msg="Product added") else:

sql ="INSERT INTO Products (PRODUCTNAME,QUANTITYIN,QUANTITYOUT,TO,DATE,HOLDERNAME) VALUES (?,?,?,?,?,?);"

prep\_stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(prep\_stmt, 1, pname)

ibm\_db.bind\_param(prep\_stmt, 2, quantity)

ibm\_db.bind\_param(prep\_stmt, 3, '')

ibm\_db.bind\_param(prep\_stmt, 4, '')

ibm\_db.bind\_param(prep\_stmt, 5, str(the\_time))

ibm\_db.bind\_param(prep\_stmt, 6, session['name']) ibm\_db.execute(prep\_stmt)

return render\_template('dashboard/addproduct.html', msg="Product added")

@app.route('/productlist') def productlist():

products = []

sql = "SELECT \* FROM Products WHERE HOLDERNAME = ?"

prep\_stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(prep\_stmt, 1, session['name']) ibm\_db.execute(prep\_stmt)

dictionary = ibm\_db.fetch\_both(prep\_stmt) while dictionary != False:

# print ("The Name is : ", dictionary) products.append(dictionary)

dictionary = ibm\_db.fetch\_both(prep\_stmt) if products:

return render\_template("dashboard/productlist.html", products = products , session = session)

else:

return render\_template("dashboard/productlist.html") @app.route('/logout')

def logout(): session.pop('loggedin', None) session.pop('id', None) session.pop('email', None) session.pop('name', None) return redirect(url\_for('home'))

@app.route('/list') def list():

users = []

sql = "SELECT \* FROM Users"

stmt = ibm\_db.exec\_immediate(conn, sql)

dictionary = ibm\_db.fetch\_both(stmt) while dictionary != False:

# print ("The Name is : ", dictionary) users.append(dictionary)

dictionary = ibm\_db.fetch\_both(stmt) if users:

return render\_template("list.html", users = users , session = session) return "No users..."