Date	11-November 2022
Team ID	PNT2022TMID47816
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" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/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: "6b77ddfe-
1472-4ab5-a4a3-
a3edd9fceccc ", // The ID of this integration.         region: " au-syd", // The region your
integration is hosted in.
       serviceInstanceID: " aa508c3a-c190-4d5e-a720-8c8a6df7e3d2 ", // 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-
bstarget="#offcanvasNavbar" aria-controls="offcanvasNavbar">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="offcanvas offcanvas-end" tabindex="-1" id="offcanvasNavbar"
arialabelledby="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"
arialabel="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" databs-target="#offcanvasNavbar" aria-controls="offcanvasNavbar">

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

</button>

<div class="offcanvas offcanvas-end" tabindex="-1" id="offcanvasNavbar" arialabelledby="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" arialabel="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"
arialabel="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"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
      <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/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');backgroundposition: 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=98538591-7217-4024-b027-
8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=30875;SECURI
TY
=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=
cxz16087;PWD=0ZMLpfUqnDBTiHzz ","") 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..."
```