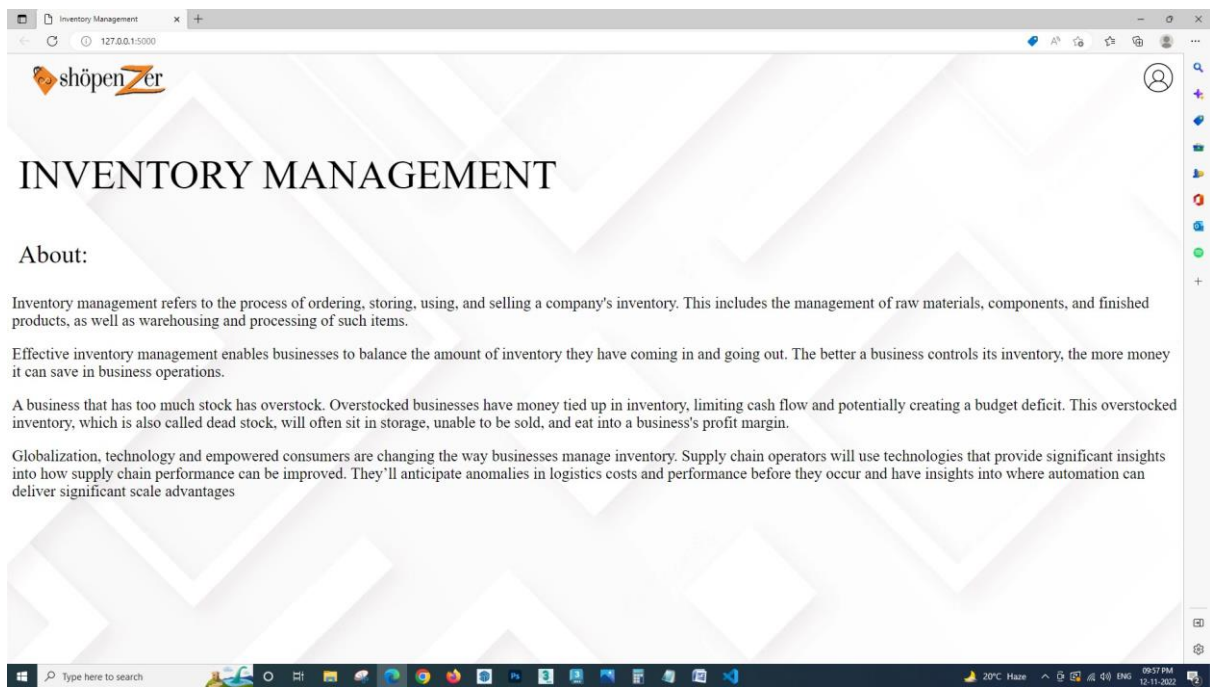# PROJECT DEVELOPMENT PHASE

SPRINT - III:

Team Id   : PNT2022TMID45514
Project Name : Inventory Management For Retailers

Home Page and About Page :

Source Code :

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Inventory Management</title>
        <link rel="stylesheet" href="{{url_for('static', filename='ind.css')}}">
    </head>
    <body>
        <div class="header">
            <img class="margin" src="{{url_for('static', filename='logo.webp')}}">
            <div class="space"></div>
            <div class="account">
                <a href="/login"><img class="acc_icon" src="{{url_for('static', filename='acc_icon.webp')}}"> </a>
            </div>
        </div>
        <br>
        <p class="title">INVENTORY MANAGEMENT</p>
        <p class="title1">About:</p>
        <p class="about">Inventory management refers to the process of ordering, storing, using, and
            selling a company's inventory. This includes the management of raw materials, components,
            and finished products, as well as warehousing and processing of such items.</p>
        <p class="about1">Effective inventory management enables businesses to balance the amount of inventory they have
            coming in and going out. The better a business controls its inventory, the more money it can save in business
            operations.</p>

        <p class="about2">A business that has too much stock has overstock. Overstocked businesses have money tied
            up in inventory, limiting cash flow and potentially creating a budget deficit. This overstocked inventory,
            which is also called dead stock, will often sit in storage, unable to be sold, and eat into a business's profit
            margin.</p>
        <p class="about3">Globalization, technology and empowered consumers are changing the way businesses manage inventory.
            Supply chain operators will use technologies that provide significant insights into how supply chain performance
            can be improved. They'll anticipate anomalies in logistics costs and performance before they occur and have
            insights into where automation can deliver significant scale advantages</p>
    </body>
</html>
```

meet.google.com is sharing your screen.   Stop sharing   Hide

```python
from turtle import st
from flask import Flask, render_template, request, redirect, url_for, session
from markupsafe import escape

import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
import ibm_db
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31198;SECURITY=SSL;SSLServerCertificate=DigiCert
print ("Database connection established", conn)

app = Flask(__name__)


@app.route('/')
def home():
    return render_template('index.html')

@app.route('/2')
def home2():
    return render_template('index1.html')

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

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

@app.route('/adminlogin')
def adminsignin():
    return render_template('adminsignin.html')
```

```python
@app.route('/regsignup',methods = ['POST', 'GET'])
def regsignup():
    if request.method == 'POST':

        name = request.form['username']
        email = request.form['loginUser']
        password = request.form['loginPassword']
        ConfirmPassword = request.form['confirmPassword']

        sql = "SELECT * FROM CUSTOMER_DATA WHERE email = ?"

        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

        if account:
            return render_template('list.html', msg="You are already a user, please login using your details")
        else:
            insert_sql = "INSERT INTO CUSTOMER_DATA VALUES (?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, password)
            ibm_db.bind_param(prep_stmt, 4, ConfirmPassword)


            ibm_db.execute(prep_stmt)


        return render_template('msg.html', )
```

Terminal output:

```
  File "C:\Users\DELL XPS\AppData\Local\Programs\Python\Python310\lib\site-packages\flask\templating.py", line 90, in _get_source_fast
    raise TemplateNotFound(template)
jinja2.exceptions.TemplateNotFound: result.html
PS C:\Users\DELL XPS\Desktop\New website\flask-with-ibm-db2-main>  c:; cd 'c:\Users\DELL XPS\Desktop\New website\flask-with-ibm-db2-main'; & 'C:\Users\DELL XPS\AppData\Local\Programs\Python\Python310\python.e
xe' 'c:\Users\DELL XPS\.vscode\extensions\ms-python.python-2022.18.2\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '51148' '--' '-m' 'flask' 'run' '--no-debugger' '--no-reload'
Database connection established <ibm_db.IBM_DBConnection object at 0x0000026AF1BA9570>
 * Serving Flask app 'app.py'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```