**HX8001- PROFESSIONAL READINESS FOR**

**INNOVATION,EMPLOYABILITY AND ENTREPRENEURSHIP**

**SUBMITTED BY**

**TEAM ID : PNT2022TMID31378**

**SRIVARSHAN S- 710719104096**

**SABARISHWARAN G- 710719104080**

**SARAVANAPRAKASH S - 710719104085**

**PRADEESH R - 710719104072**

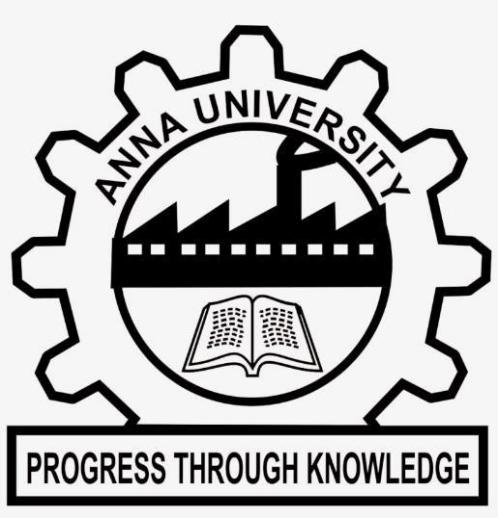
**Pursuing final year of the degree BACHELOR OF ENGINEERING**

**In**

**DR.N.G.P. INSTITUTE OF TECHNOLOGY**

**NOV 2022**

**COIMBATORE - 641048**



**ANNA UNIVERSITY: CHENNAI 600 025**

**ACKNOWLEDGEMENT**

**INTRODUCTION**

* 1. Project Overview
  2. Purpose

1. **LITERATURE SURVEY**
   1. Existing problem
   2. References
   3. Problem Statement Definition
2. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit
3. **REQUIREMENT ANALYSIS**
   1. Functional requirement
   2. Non-Functional requirements
4. **PROJECT DESIGN**
   1. Data Flow Diagrams
   2. Solution & Technical Architecture
   3. User Stories
5. **PROJECT PLANNING & SCHEDULING**
   1. Sprint Planning & Estimation
   2. Sprint Delivery Schedule
   3. Reports from JIRA
6. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
   1. Feature 1
   2. Feature 2
7. **TESTING**
   1. Test Cases
8. **RESULTS**
   1. Performance Metrics
9. **ADVANTAGES & DISADVANTAGES**
10. **CONCLUSION**
11. **FUTURE SCOPE**
12. **APPENDIX**

Source Code

GitHub & Project Demo Link

1. **INTRODUCTION**
   1. **PROJECT OVERVIEW**

Inventory management helps companies identify which and how much stock to order at what time. It tracks inventory from purchase to the sale of goods. The practice identifies and responds to trends to ensure there's always enough stock to fulfill customer orders and proper warning of a shortage.

* 1. **PURPOSE**

Retail inventory management tools and methods give retailers more information with which to run their businesses, including: Product locations. Quantities of each product type. Which stock sells well and which doesn't, by location and sales channel.

1. **LITERATURE**
   1. **EXISTING PROBLEMS**
      * Lack of Inventory Visibility.
      * Inefficient Inventory Management Process or Software.
      * Tracking Obsolete Material.
      * Identifying Incorrectly Located Materials.
      * Keeping up with Overstocks.
      * Managing Inventory Waste & Defects.
      * Lack of Centralized Inventory Hub.
      * Changing Demand
   2. **REFERENCES**
      * Digital Reference by tranquilbs website
      * Reference from tranquil content writer.
      * Reference by [https://www.tranquilbs.](http://www.tranquilbs.com/inventory-management-problems/)com/inve[ntory-management-problems/](http://www.tranquilbs.com/inventory-management-problems/)
   3. **PROBLEM STATEMENT DEFINITION**

In inventory systems, demand is usually uncertain, and the lead-time can also vary. To avoid shortages, managers often maintain a safety stock. In such situations, it

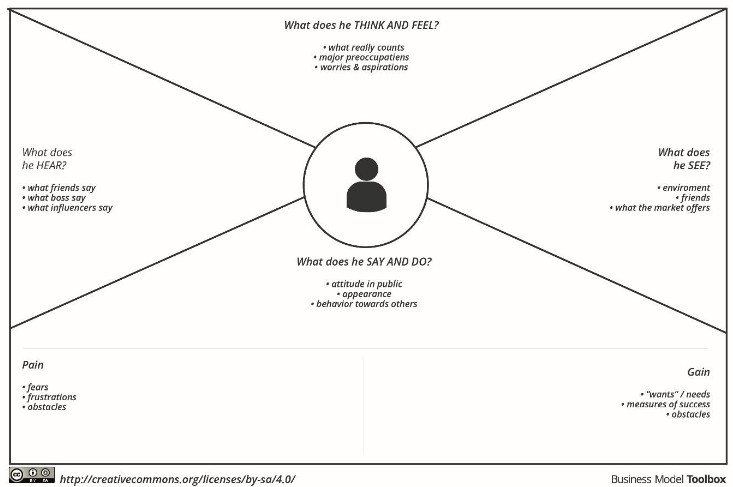
is not clear what order quantities and reorder points will minimize expected total inventory cost. Simulation models can address this question.

1. **IDEATION & PROPOSED SOLUTION**
   1. **EMPATHY MAP CANVAS**

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviours and attitudes.

It is a useful tool to helps teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user’s perspective along with his or her goals and challenges.



* 1. **IDEATION AND BRAINSTORMING**

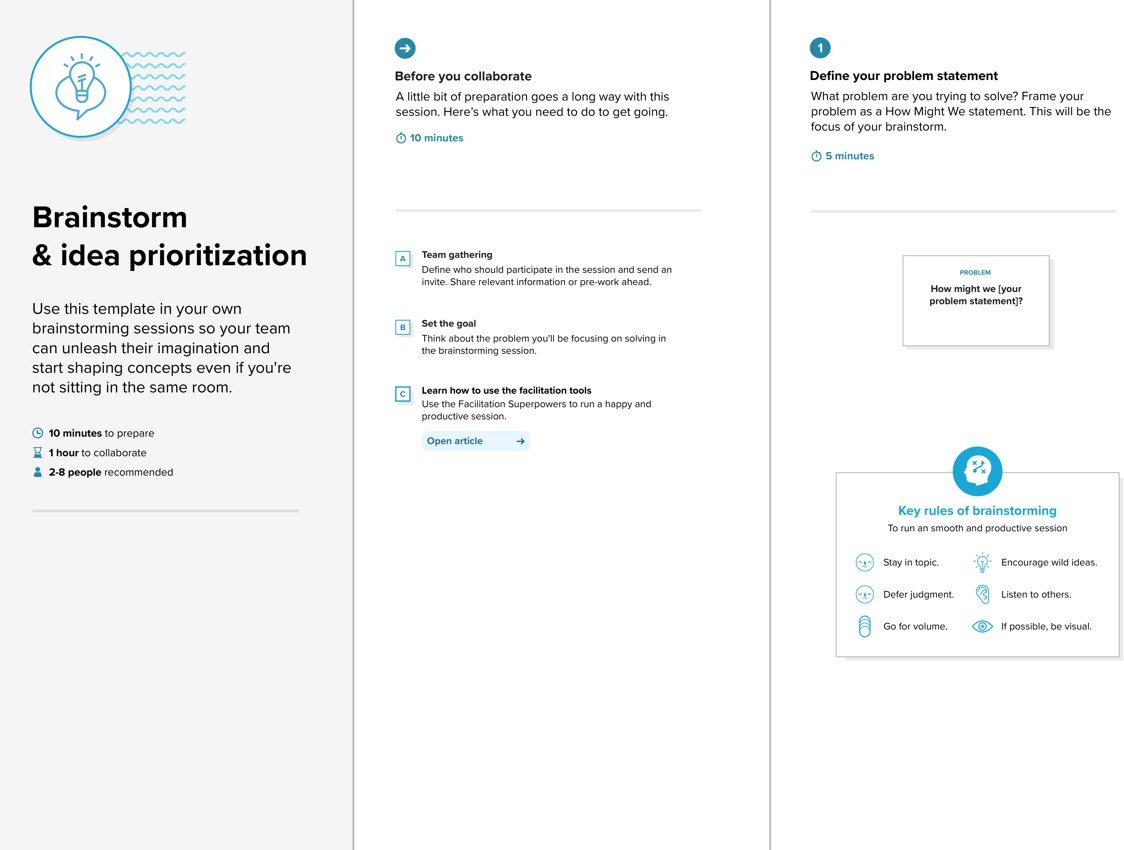
Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and

all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

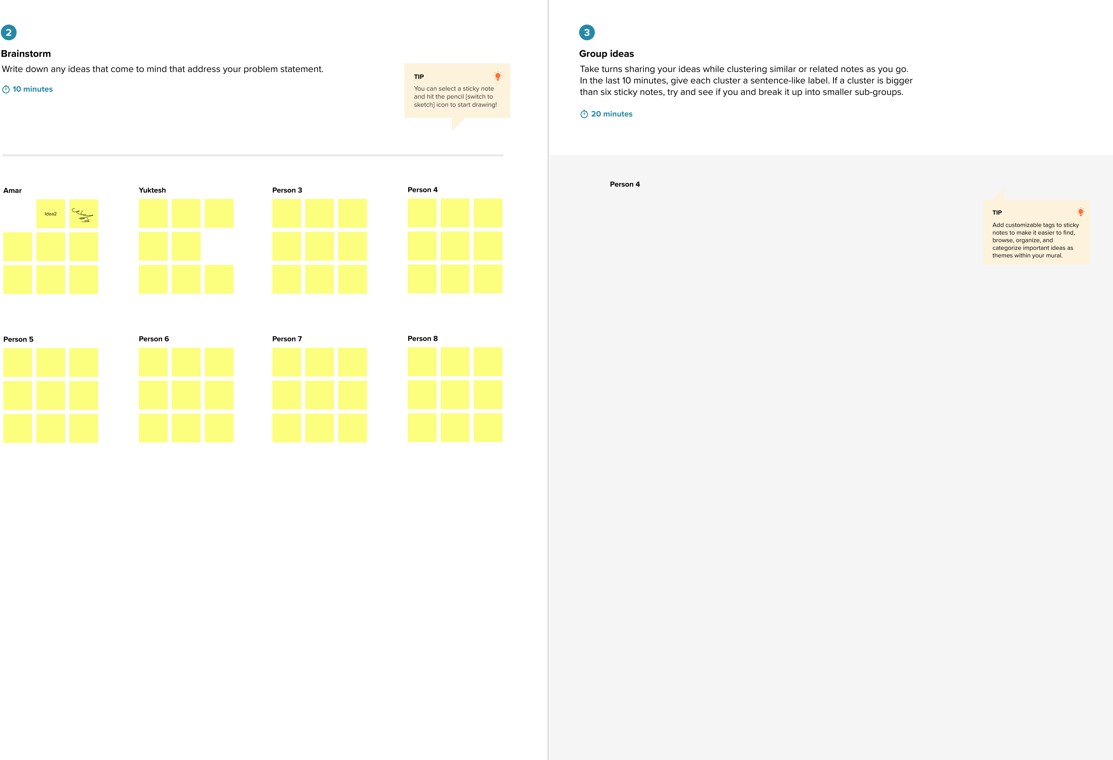
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Reference: <https://www.mural.co/templates/empathy-map-canvas>

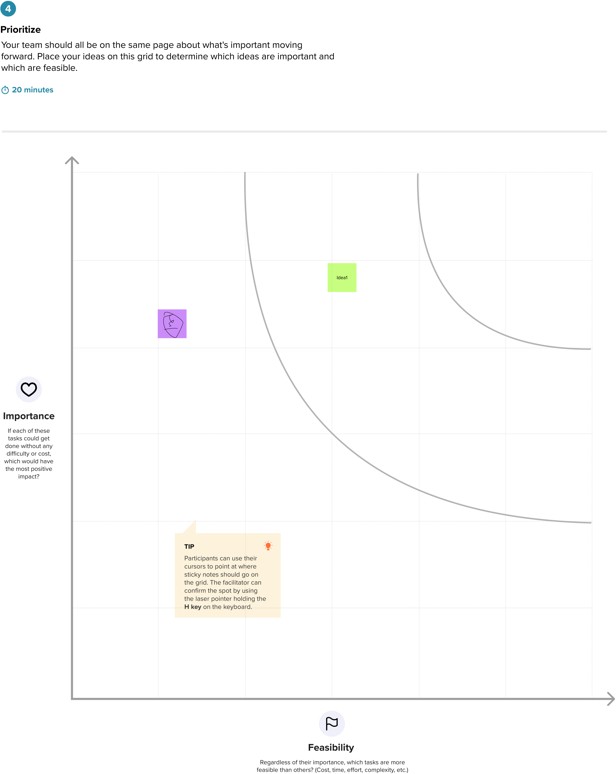
**Step-1: Team Gathering, Collaboration and Select the Problem Statement**



**Step-2: Brainstorm, Idea Listing and Grouping**



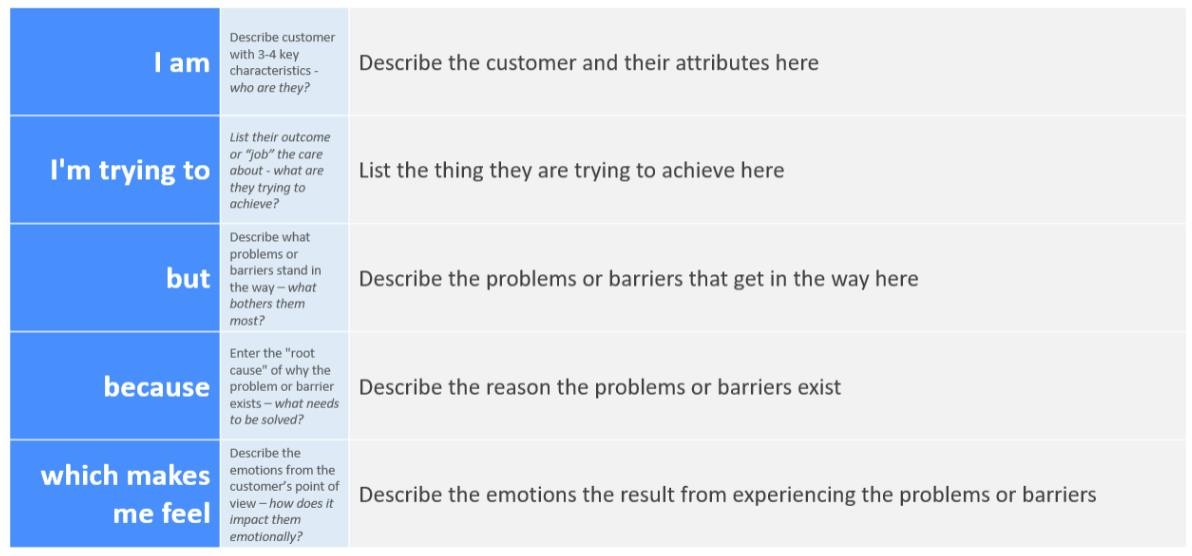
**Step-3: Idea Prioritization**



* 1. **PROPOSED SOLUTION**
     + Centralized Tracking: Consider upgrading to tracking software that provides automated features for re-ordering and procurement.
     + Transparent Performance.
     + Stock Auditing.
     + Demand Forecasting.
     + Add Imagery.
     + Go Paperless.
     + Preventive Control.
     + Measure Control.
     + Safety stock
  2. **PROBLEM SOLUTION FIT**

Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love.

A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you’ll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.



Reference: <https://miro.com/templates/customer-problem-statement/>

**Example:**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Statement (PS)** | **I am (Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 |  |  |  |  |  |
| PS-2 |  |  |  |  |  |

1. **REQUIREMENT ANALYSIS**
   1. **FUNCTIONAL REQUIREMENTS**

Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR**  **No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through Form Registration through Gmail Registration through LinkedIN |
| FR-2 | User Confirmation | Confirmation via Email Confirmation via OTP |
| FR-3 | User login | Login with username Login with password |
| FR-4 | Centralized Record of all product | Product name, Stock keep unit, brand, retail price, product category, lot number , expire date, vendor details, wholesale cost, minimum reorder amount, case quantity amount, reorder lead time |
| FR-5 | Stock location identification | Provide number label for- Shelf, Rack and Boxes |
| FR-6 | Periodical stock checking | Physical counting and Cycle counting |
| FR-7 | Integration of sales and inventory data | sales administration and database upkeep FIFO,LILO according to the goods |
| FR-8 | Purchase management and Forecasting | Order review and placement, Avoid risk stock, review product, priorities purchases based on an item's  profitability, popularity, and lead time, ABC,FSC,XYZ,JIT techniques |
| FR-9 | Markdown and promotion | Show product discount,  Maintain enough stock on hand to meet demand. |
| FR-10 | Management of Receiving Stock | Accurately recording goods on an inventory |
| FR-11 | Returns Management System | Check for damage or defects and return to vendor as needed  If sellable add it to inventory counts |
| FR-12 | Determination of death stock | Return to the vendor for credits |
| FR-13 | Inventory KPIs(Key  Performance Indicator) | Sale KPIs, Receive KPIs, Operational KPIs, Employee KPIs |

* 1. **NON-FUNCTIONAL REQUIREMENTS**

Following are the non-functional requirements of the proposed solution.

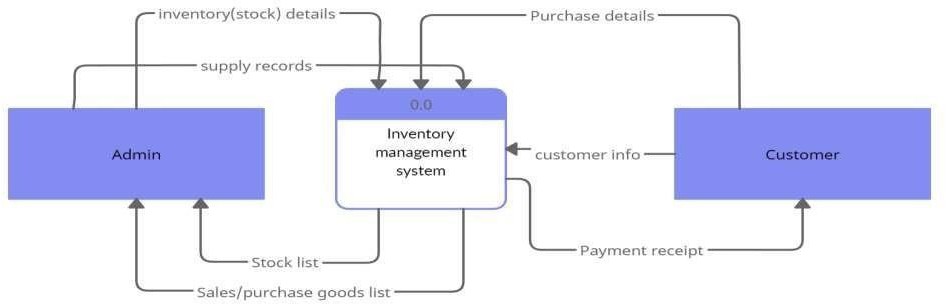
|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |

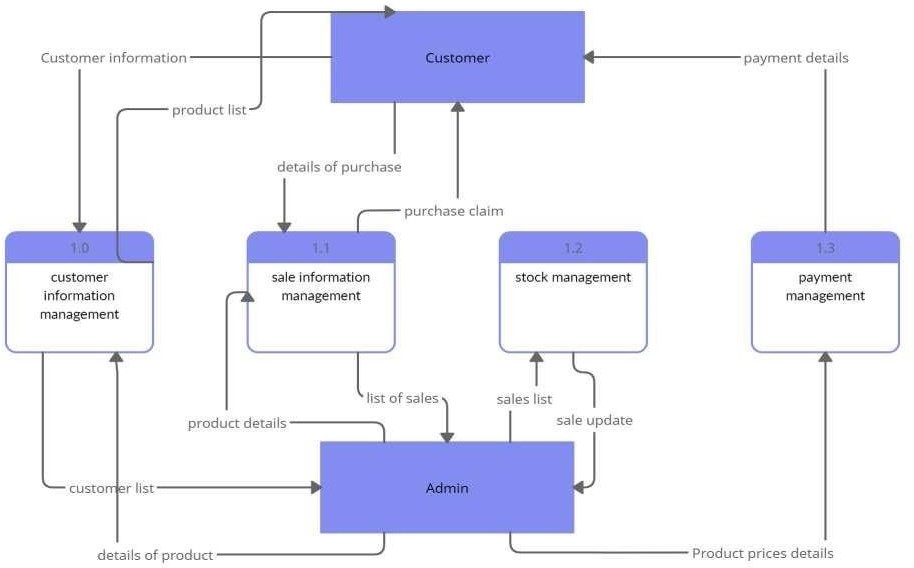
|  |  |  |
| --- | --- | --- |
| NFR-1 | **Usability** | This system must be easy to use by both managers and chefs, such that they do not need to read an extensive number of manuals; it must be quickly accessible by both managers and chefs; it must be intuitive and simple in the way it displays all relevant data and relationships; and the menus of the system are easily navigable by  the users with buttons that are easy to understand. |
| NFR-2 | **Security** | The security requirements deal with the primary security. Only authorized users can access the system with user name and password of administrator. |
| NFR-3 | **Reliability** | The system must give accurate inventory status to the user continuously. Any inaccuracies are corrected by regularly comparing the actual levels to the levels displayed in the system.  The system must successfully add any recipe, ingredients, vendors, or special occasions given by the user and provide estimations and  inventory status in relevance to the newly updated entities. |
| NFR-4 | **Performance** | The system must not lag, because the workers using it don't have downtime to wait for it to complete an action.  The system must successfully complete updating the databases, adding new recipes, ingredients, vendors, and occasions every time the user requests such a process.  All the functions of the system must be available to the user every time the system is turned on. The calculations performed by the system must comply with the norms set by the user and should not vary unless explicitly  changed by the user. |
| NFR-5 | **Availability** | The software will be available only to administrator of the organization and the product as well as customer details will be recorded by him. He can add customers, Update and delete them as well as add new products and manage  them |
| NFR-6 | **Scalability** | The ability of a system to handle a growing amount of work. |

1. **PROJECT DESIGN**
   1. **DATA FLOW DIAGRAMS**

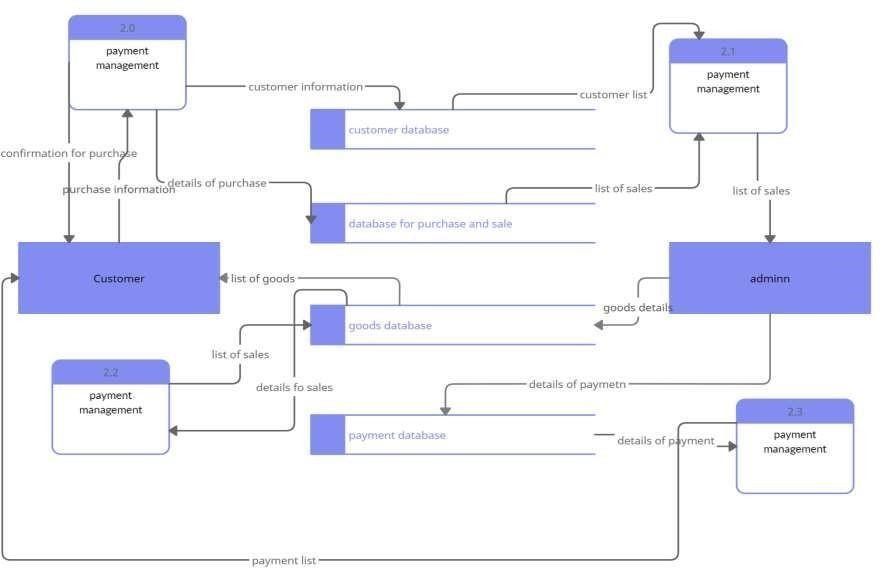
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

**Level 0 Data Flow Diagram for inventory management system for retailers:**

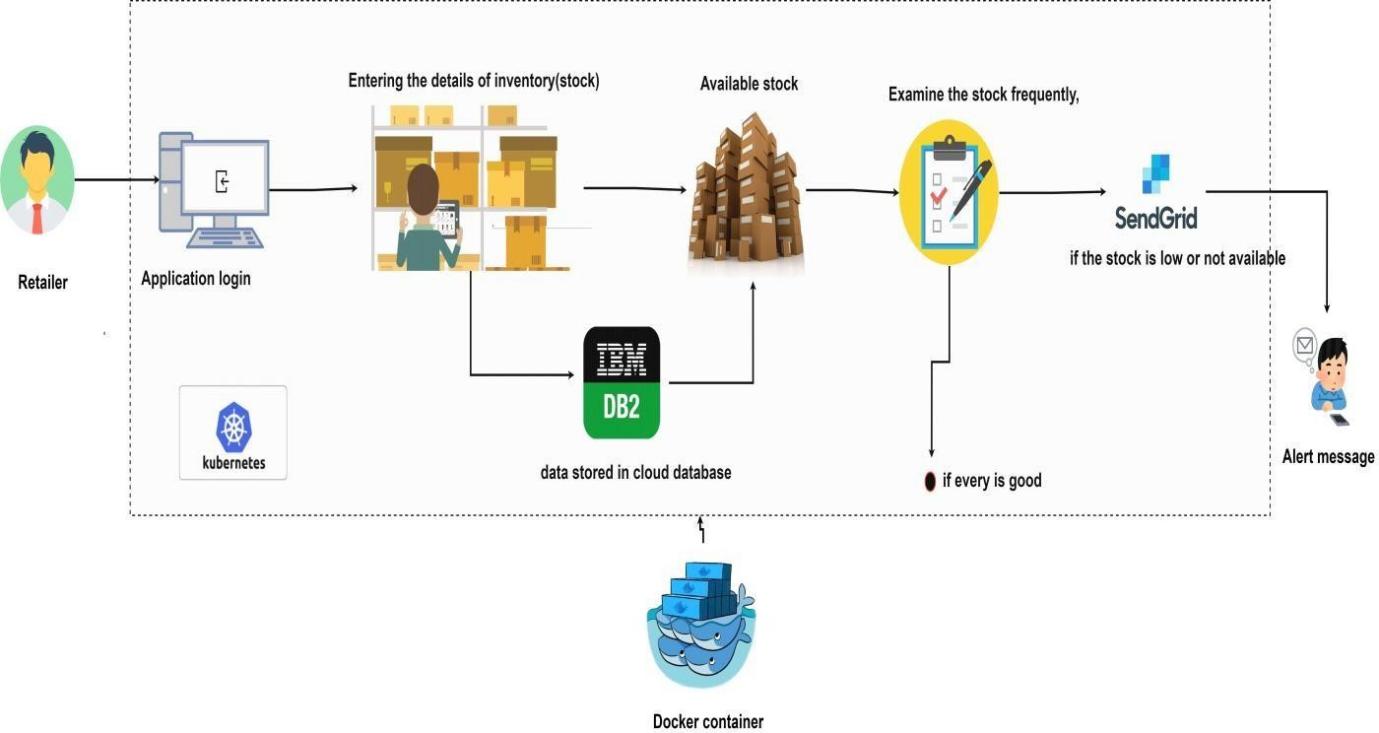


**Level 1 Data Flow Diagram for inventory management system for retailers:**

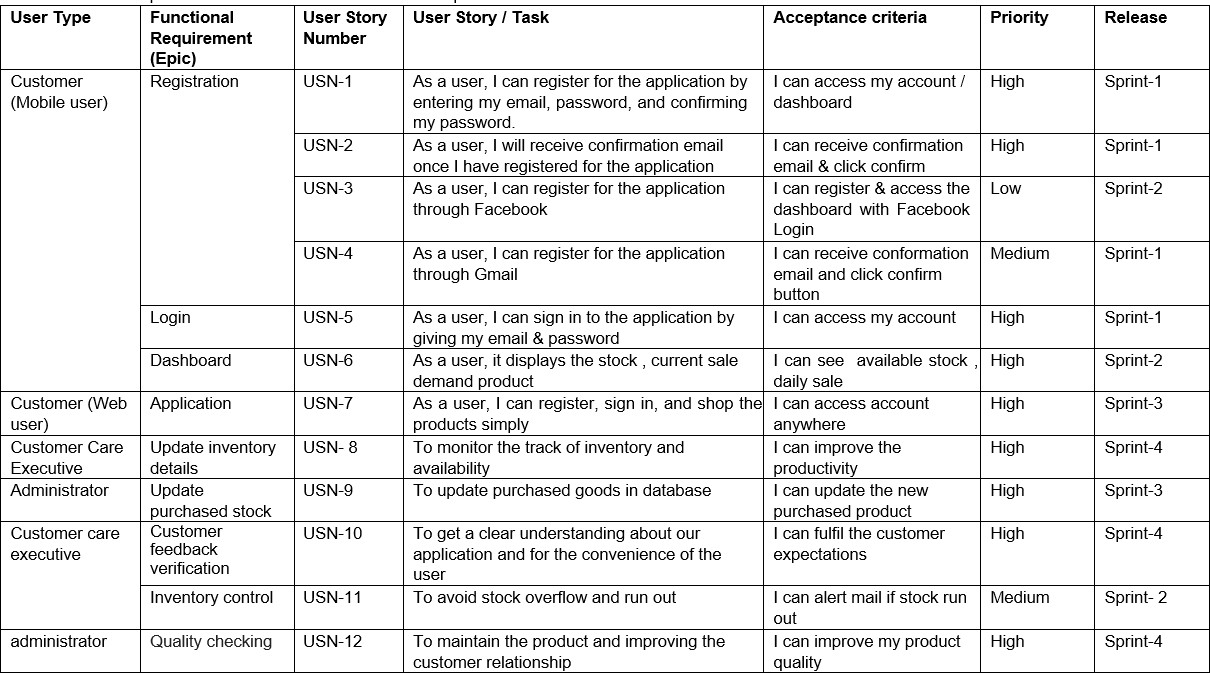
**Level 2 Data Flow Diagram for inventory management system for retailers:**



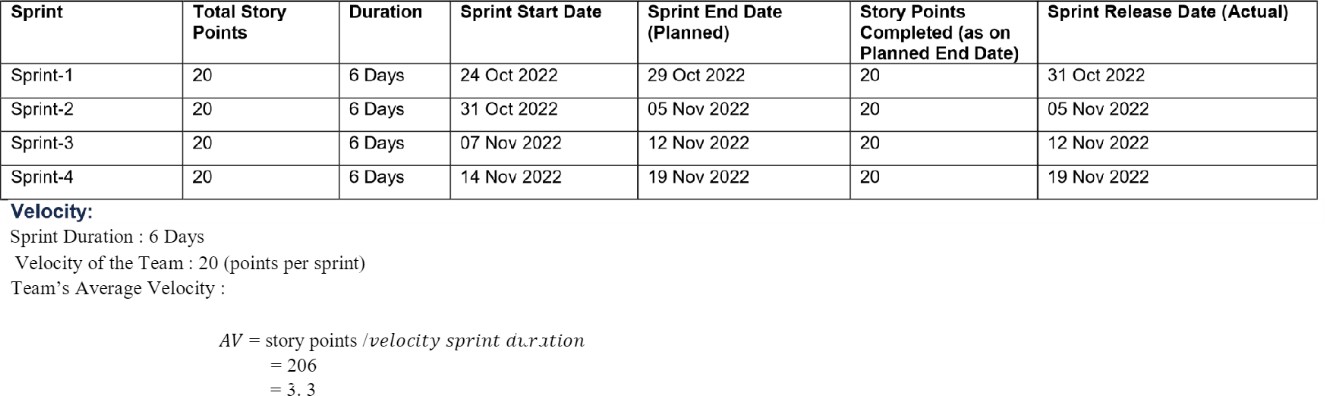
* 1. **SOLUTION & TECHNICAL ARCHITECTURE**



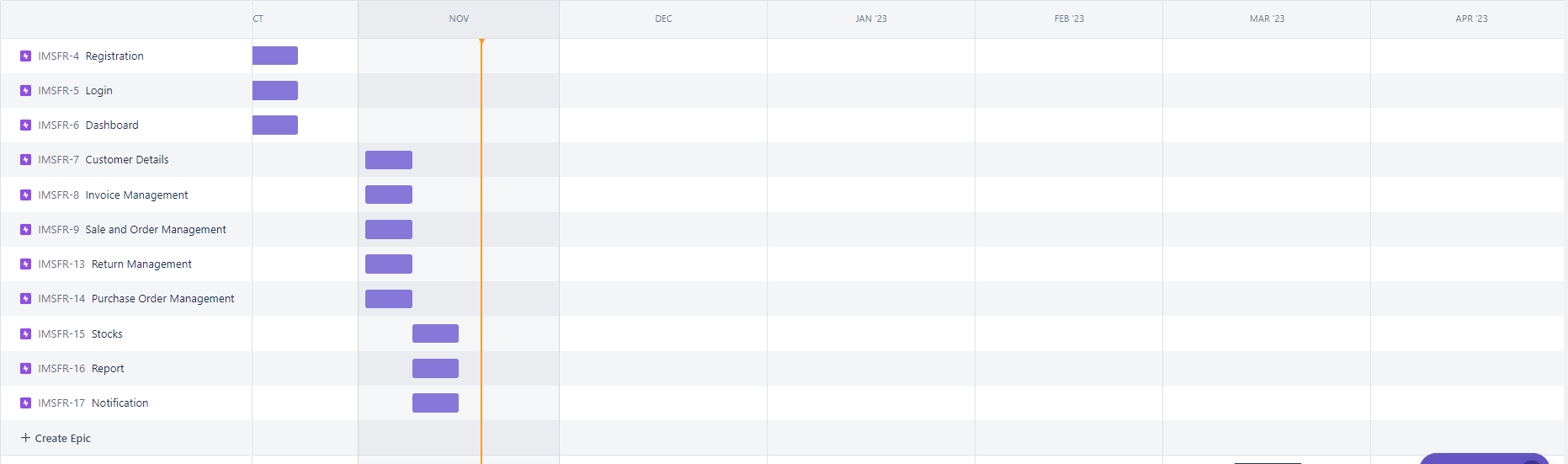
* 1. **USER STORIES**



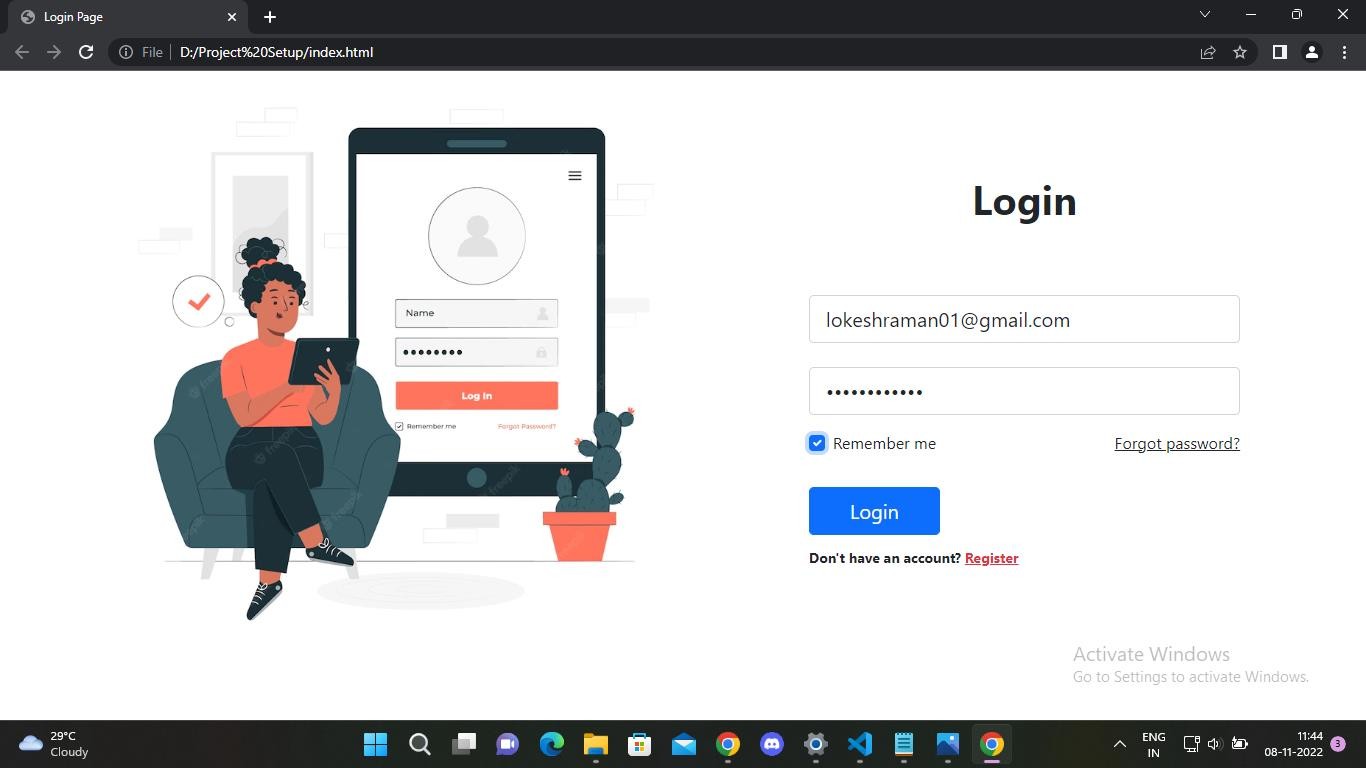
1. **PROJECT PLANNING & SCHEDULING**
   1. **SPRINT PLANNING & ESTIMATION**



**6.2 REPORTS FROM JIRA**



1. **CODING & SOLUTION**
   1. **FEATURE 1 Log-In**



**Login.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@5.0.2/dist/css/bootstrap.min.css](https://cdn.jsdelivr.net/npm/bootstrap%405.0.2/dist/css/bootstrap.min.css)" rel="stylesheet" integrity="sha384- EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

<script src="[https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js](https://cdn.jsdelivr.net/npm/bootstrap%405.0.2/dist/js/bootstrap.bundle.min.js)" integrity="sha384- MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

<style>

.divider:after,

.divider:before { content: ""; flex: 1; height: 1px;

background: #eee;

}

.h-custom {

height: calc(100% - 73px);

}

@media (max-width: 450px) {

.h-custom { height: 100%;

}

}

</style>

<title>Login Page</title>

</head>

<body>

<section class="vh-100">

<div class="container-fluid h-custom">

<div class="row d-flex justify-content-center align-items-center h-100">

<div class="col-md-9 col-lg-6 col-xl-5">

<img src="https://img.freepik.com/free-vector/tablet-login-concept-illustration\_114360- 7863.jpg?w=740&t=st=1667711849~exp=1667712449~hmac=d4ef0be91c59f7ea273e94343f3799a0 bda80f45635cb1d6187e575f2d5b5fd9"

class="img-fluid" alt="Sample image">

</div>

<div class="col-md-8 col-lg-6 col-xl-4 offset-xl-1">

<p class="text-center h1 fw-bold mb-5 mx-1 mx-md-4 mt-4">Login</p>

<form>

<!-- Email input -->

<br>

<div class="form-outline mb-4">

<input type="email" id="form3Example3" class="form-control form-control-lg" placeholder="Enter a valid email address" />

<!-- <label class="form-label" for="form3Example3">Email address</label> -->

</div>

<!-- Password input -->

<div class="form-outline mb-3">

<input type="password" id="form3Example4" class="form-control form-control-lg" placeholder="Enter password" />

<!-- <label class="form-label" for="form3Example4">Password</label> -->

</div>

<div class="d-flex justify-content-between align-items-center">

<!-- Checkbox -->

<div class="form-check mb-0">

<input class="form-check-input me-2" type="checkbox" value="" id="form2Example3"

/>

<label class="form-check-label" for="form2Example3"> Remember me

</label>

</div>

<a href="#!" class="text-body">Forgot password?</a>

</div>

<div class="text-center text-lg-start mt-4 pt-2">

<button type="button" class="btn btn-primary btn-lg"

style="padding-left: 2.5rem; padding-right: 2.5rem;">Login</button>

<p class="small fw-bold mt-2 pt-1 mb-0">Don't have an account? <a href="signup.html" class="link-danger">Register</a></p>

</div>

</form>

</div>

</div>

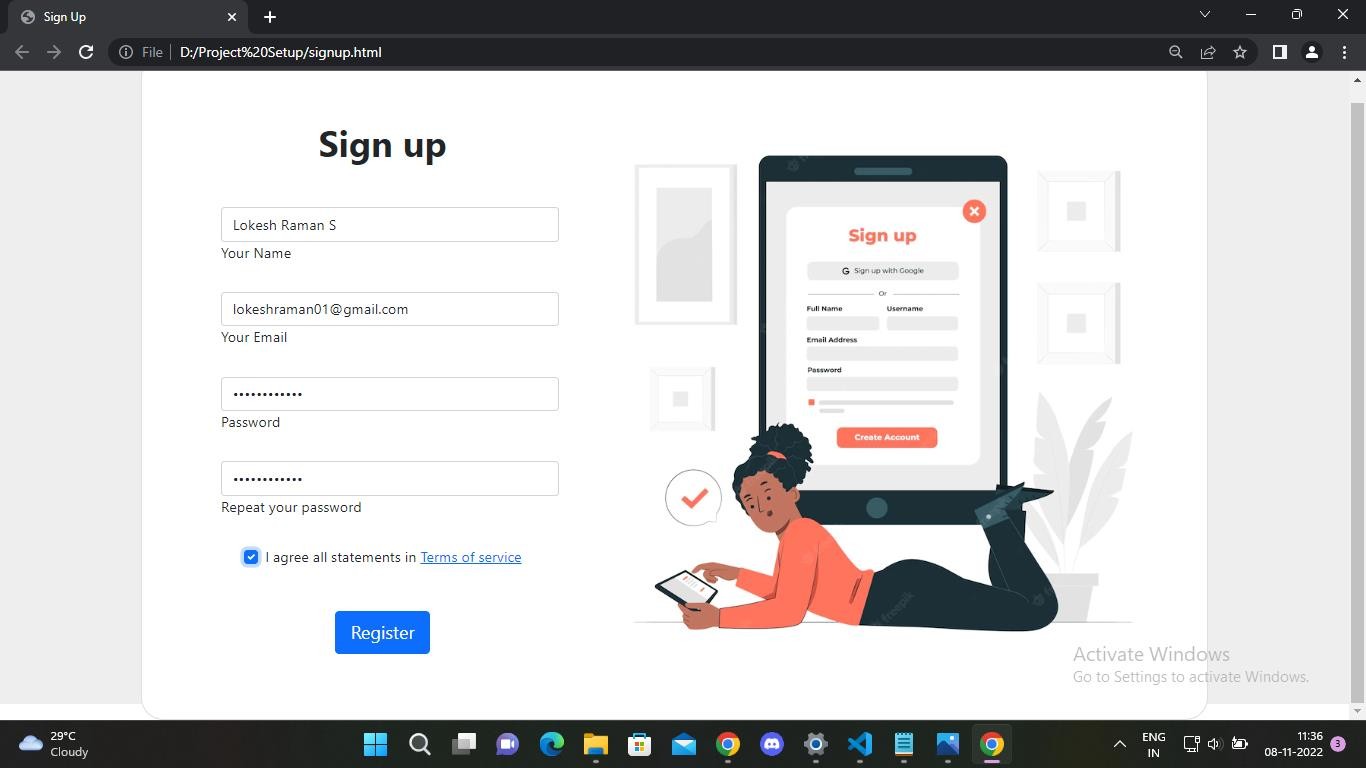
</div>

</section>

</body>

</html>

* 1. **FEATURE 2**



**Signup.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@5.0.2/dist/css/bootstrap.min.css](https://cdn.jsdelivr.net/npm/bootstrap%405.0.2/dist/css/bootstrap.min.css)" rel="stylesheet" integrity="sha384- EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

<script src="[https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js](https://cdn.jsdelivr.net/npm/bootstrap%405.0.2/dist/js/bootstrap.bundle.min.js)" integrity="sha384- MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>

<title>Sign Up</title>

</head>

<body>

<section class="vh-100" style="background-color: #eee;">

<div class="container h-100">

<div class="row d-flex justify-content-center align-items-center h-100">

<div class="col-lg-12 col-xl-11">

<div class="card text-black" style="border-radius: 25px;">

<div class="card-body p-md-5">

<div class="row justify-content-center">

<div class="col-md-10 col-lg-6 col-xl-5 order-2 order-lg-1">

<p class="text-center h1 fw-bold mb-5 mx-1 mx-md-4 mt-4">Sign up</p>

<form class="mx-1 mx-md-4">

<div class="d-flex flex-row align-items-center mb-4">

<i class="fas fa-user fa-lg me-3 fa-fw"></i>

<div class="form-outline flex-fill mb-0">

<input type="text" id="form3Example1c" class="form-control" />

<label class="form-label" for="form3Example1c">Your Name</label>

</div>

</div>

<div class="d-flex flex-row align-items-center mb-4">

<i class="fas fa-envelope fa-lg me-3 fa-fw"></i>

<div class="form-outline flex-fill mb-0">

<input type="email" id="form3Example3c" class="form-control" />

<label class="form-label" for="form3Example3c">Your Email</label>

</div>

</div>

<div class="d-flex flex-row align-items-center mb-4">

<i class="fas fa-lock fa-lg me-3 fa-fw"></i>

<div class="form-outline flex-fill mb-0">

<input type="password" id="form3Example4c" class="form-control" />

<label class="form-label" for="form3Example4c">Password</label>

</div>

</div>

<div class="d-flex flex-row align-items-center mb-4">

<i class="fas fa-key fa-lg me-3 fa-fw"></i>

<div class="form-outline flex-fill mb-0">

<input type="password" id="form3Example4cd" class="form-control" />

<label class="form-label" for="form3Example4cd">Repeat your password</label>

</div>

</div>

<div class="form-check d-flex justify-content-center mb-5">

<input class="form-check-input me-2" type="checkbox" value="" id="form2Example3c" />

<label class="form-check-label" for="form2Example3">

I agree all statements in <a href="#!">Terms of service</a>

</label>

</div>

<div class="d-flex justify-content-center mx-4 mb-3 mb-lg-4">

<button type="button" class="btn btn-primary btn-lg">Register</button>

</div>

</form>

</div>

<div class="col-md-10 col-lg-6 col-xl-7 d-flex align-items-center order-1 order-lg-2">

<img src="https://img.freepik.com/free-vector/sign-up-concept-illustration\_114360- 7865.jpg?w=740&t=st=1667712441~exp=1667713041~hmac=4224cf9893b8b9c5e20fa396de1b4c00 1b8f2f9241d21dc07e260e9918c34018"

class="img-fluid" alt="Sample image">

</div>

</div>

</div>

</div>

</div>

</div>

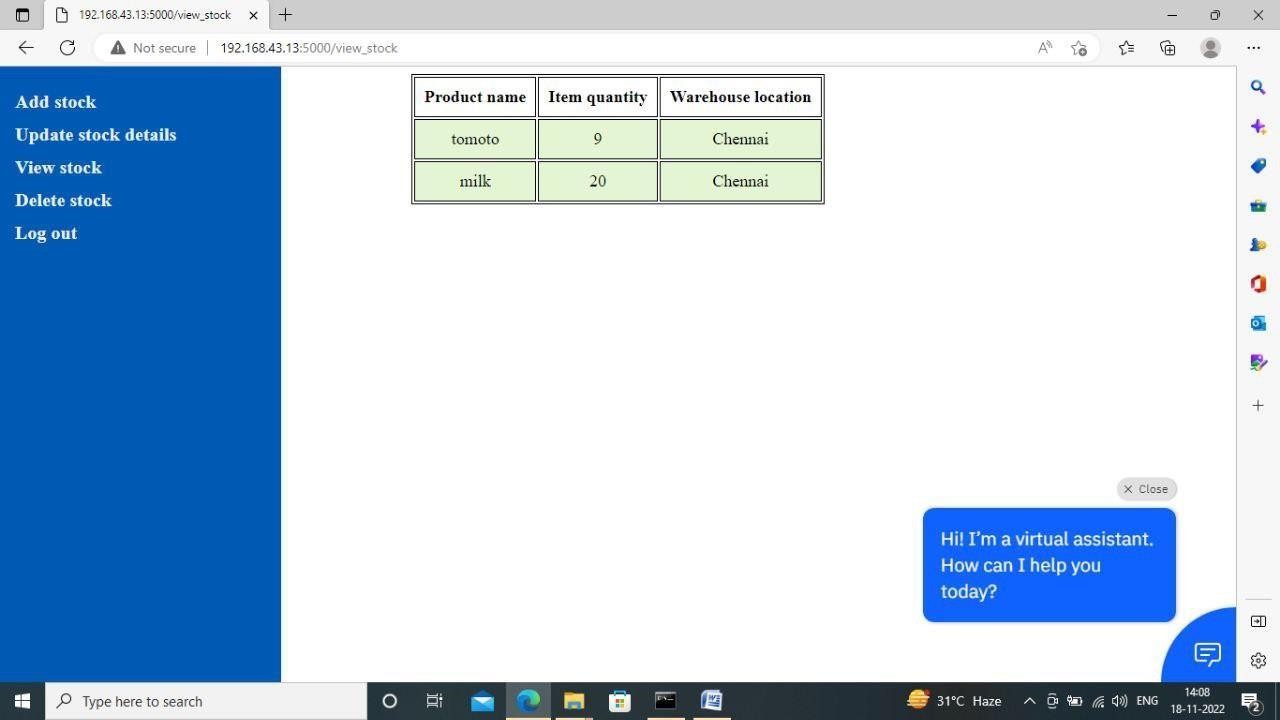
</div>

</section>

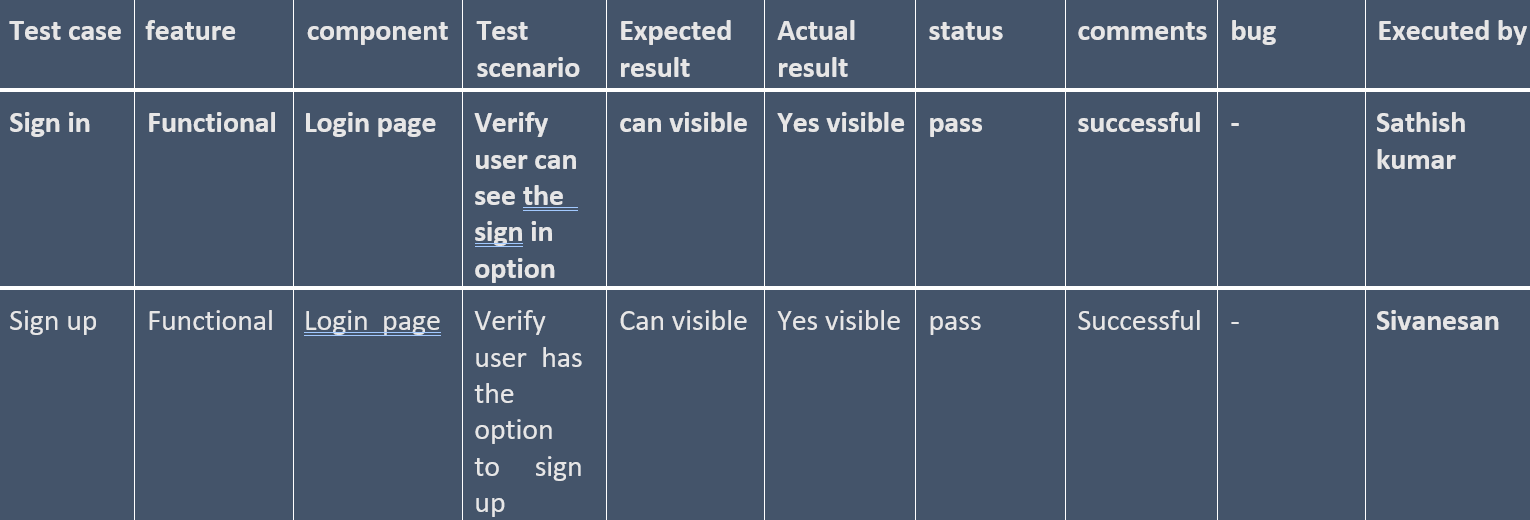
</body>

</html>

**7.3. DATABASE SCHEMA**



1. **TESTING**
   1. **TEST CASES**



**8.2.USER ACCEPTANCE TESTING**

# Test case : Testing the Add Recipe Interface and its functioning

Case 1: Testing the Quantity input field. Case 2: Testing the Recipe Name field.

Case 3: Testing the Ingredients in recipe list and Quantity of ingredient list. Case 4: Testing the available ingredients list.

Case 5: Testing the all the above cases together and checking if the entries are updated to the tables in database.

# Test Case : Check Threshold Interface

Case 1: Check if the Ingredients under the threshold values are shown in the Ingredients below threshold list.

Case 2: Check if the Create order button asks the user to enter values for all the ingredients listed under the ingredients below threshold list.

Case 3: Check if pressing the Process Order button creates a file with the order details in it.

# Test Case : Testing the Update after sales interface

Case 1: Test the Recipe list box. Case 2: Test the quantity text field..

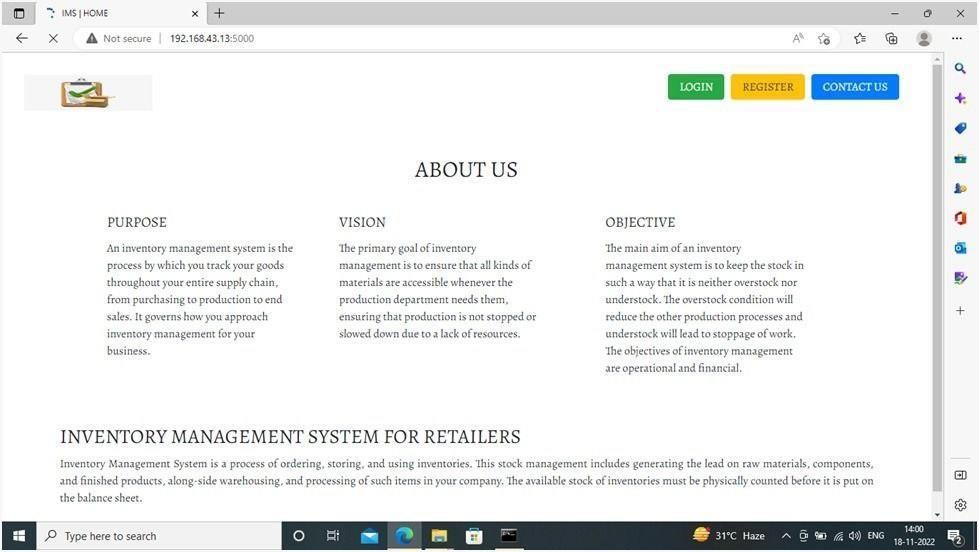
Case 3: Test the recipe sold list box quantity sold list box.

Case 4: Test if the details are updated to the database when requested.

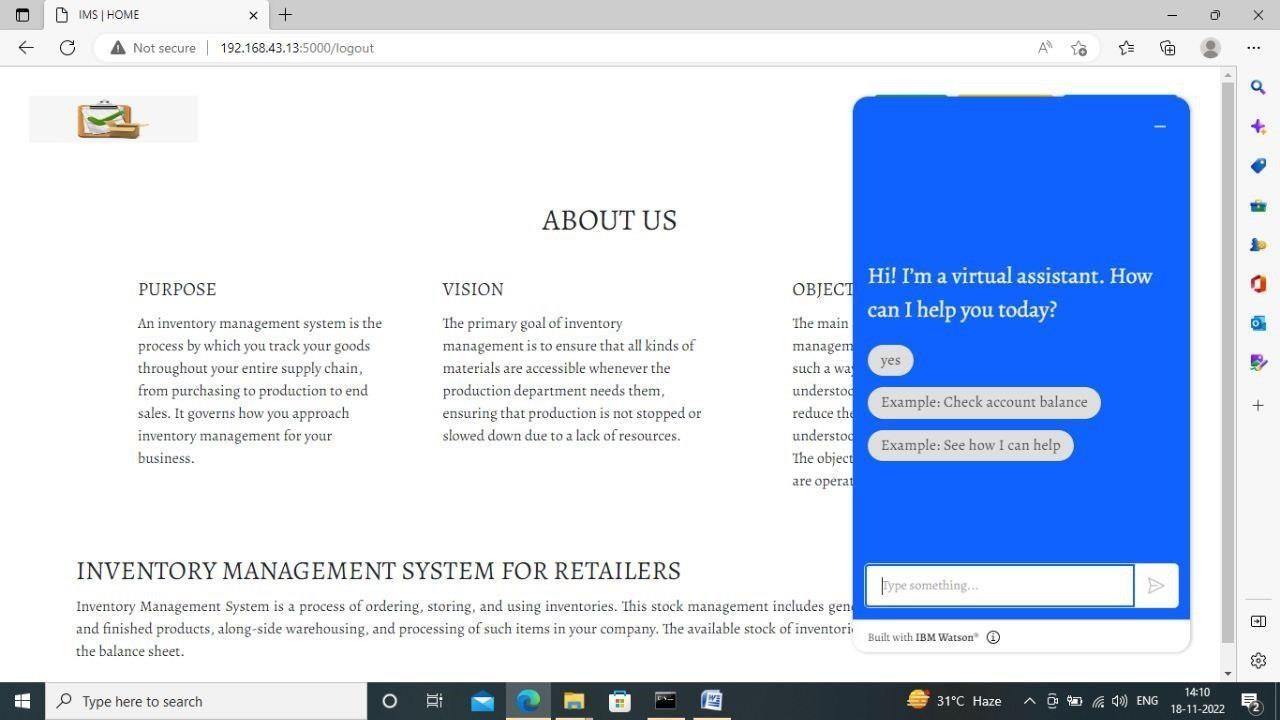
1. **RESULT**

Inventory Management System for Retailers is developed using Cloud and executed at the level of completed progress.

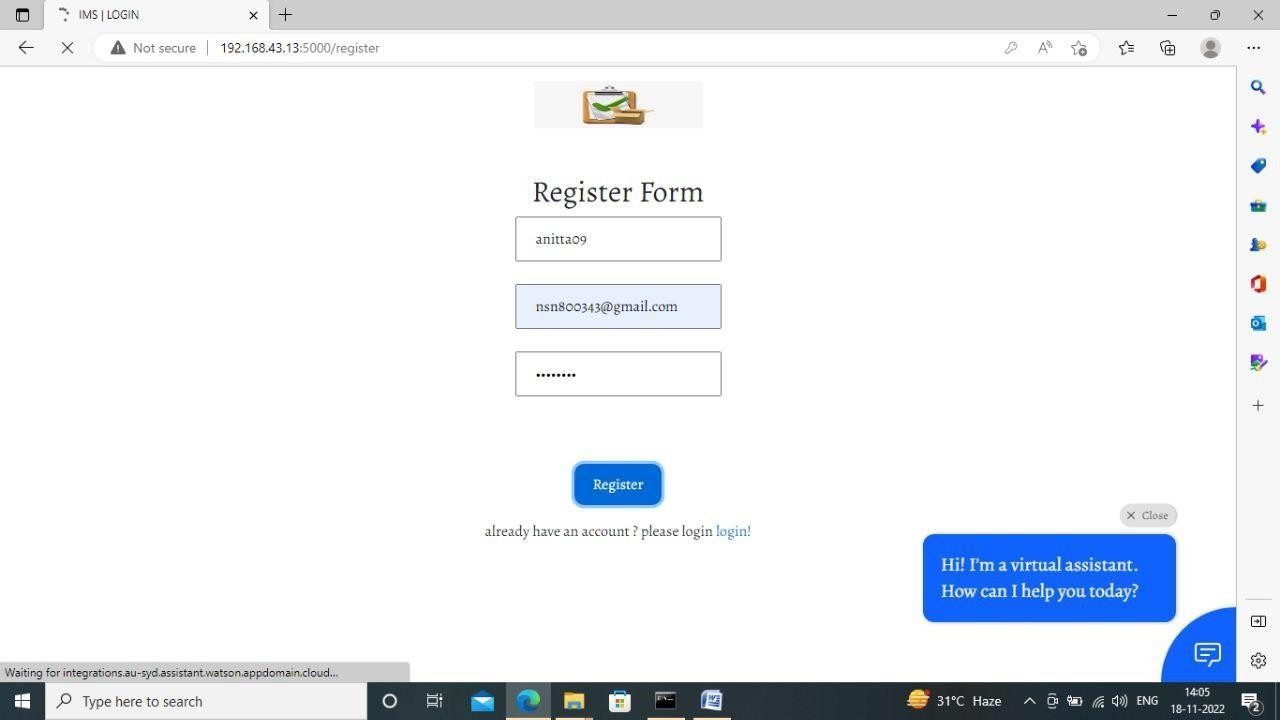
**HOME PAGE**



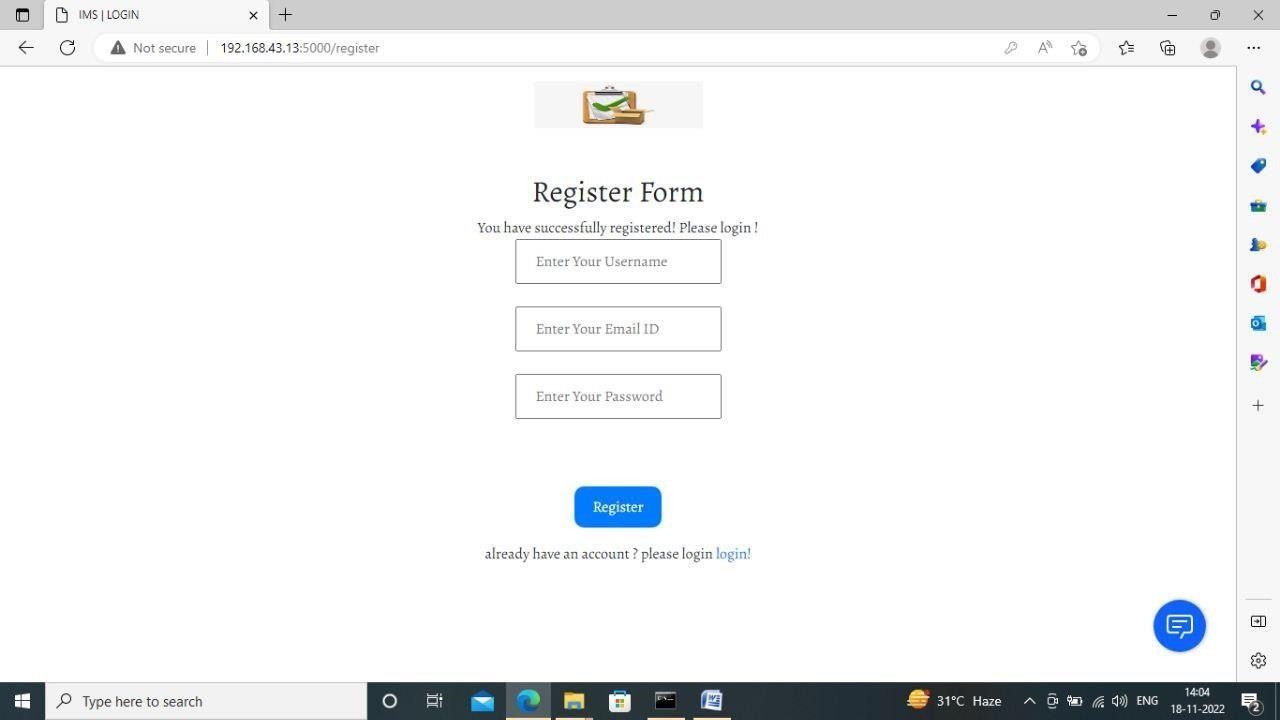
**HOMEPAGE WITH VIRTUAL ASSISTANT**



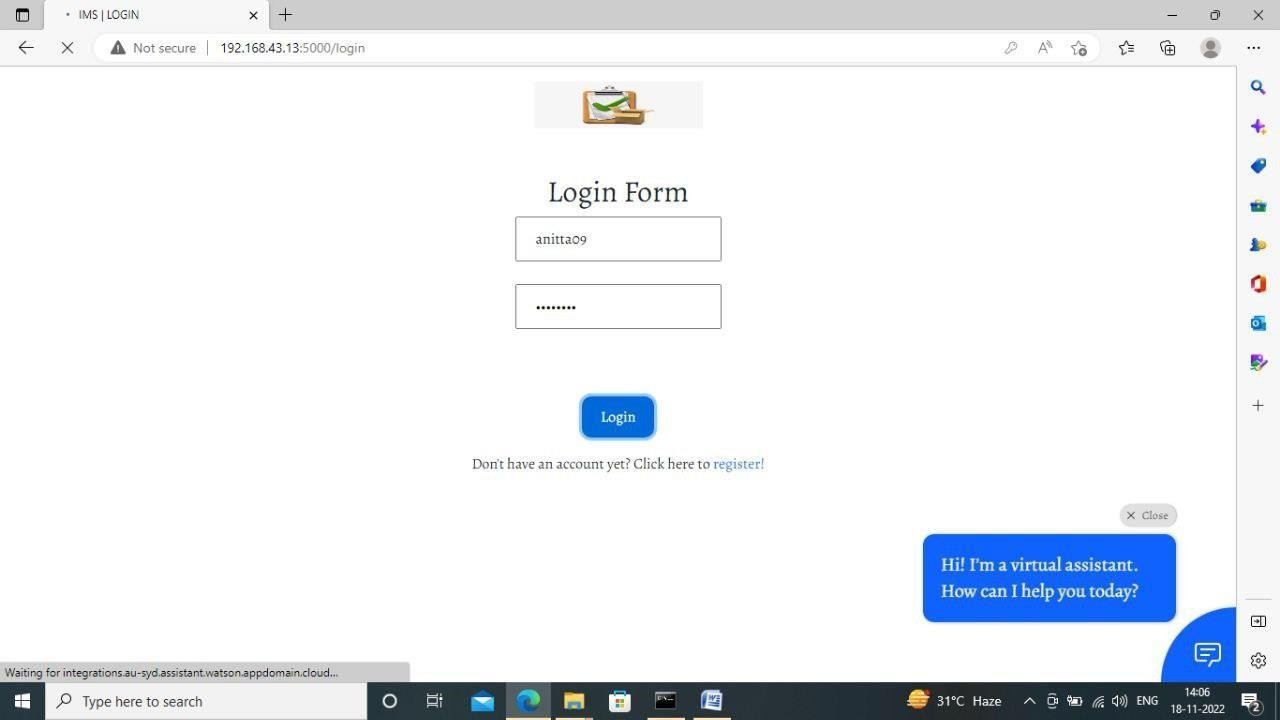
**REGISTRATION FORM**

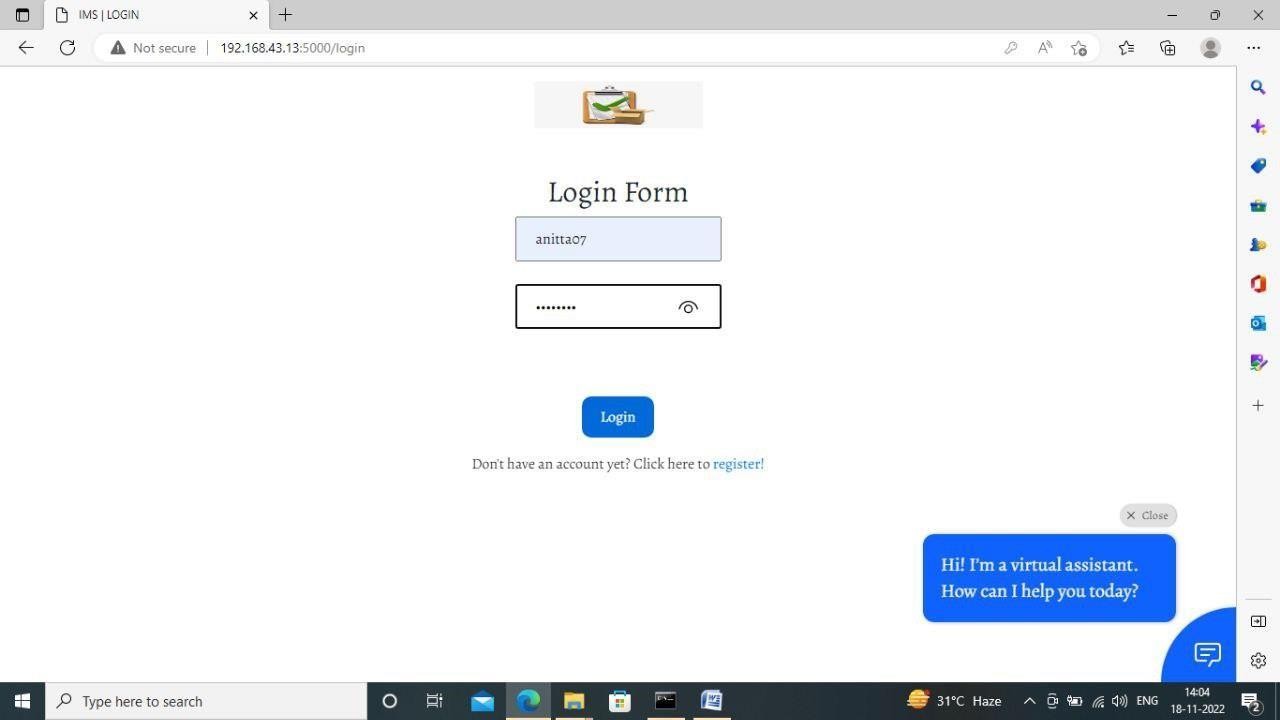


**REGISTRATION SUCCESSFULLY**

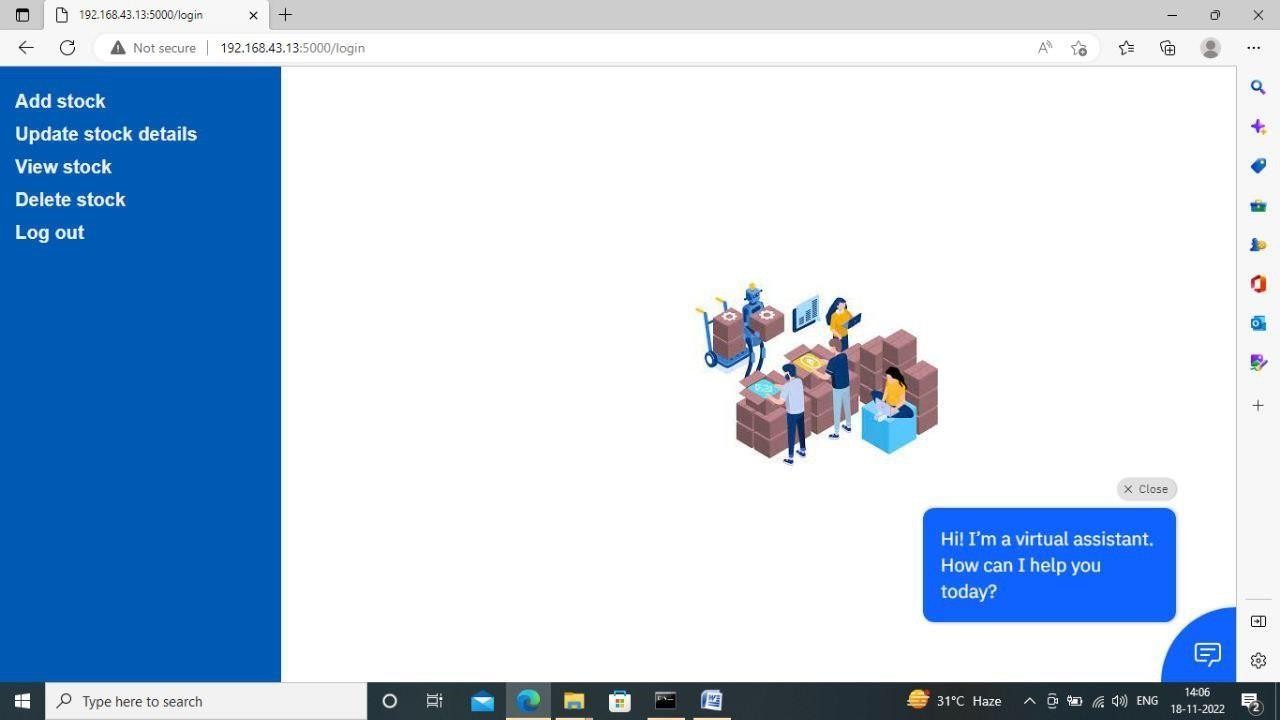


**LOGIN FORM**

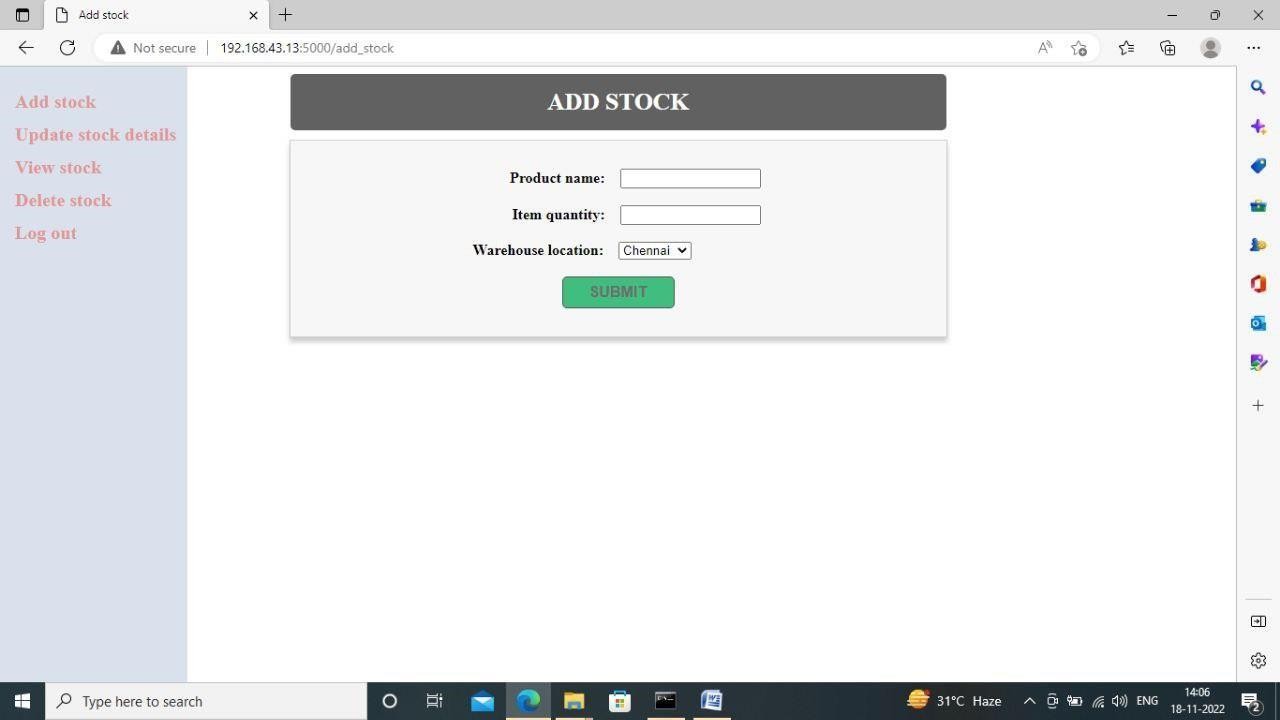




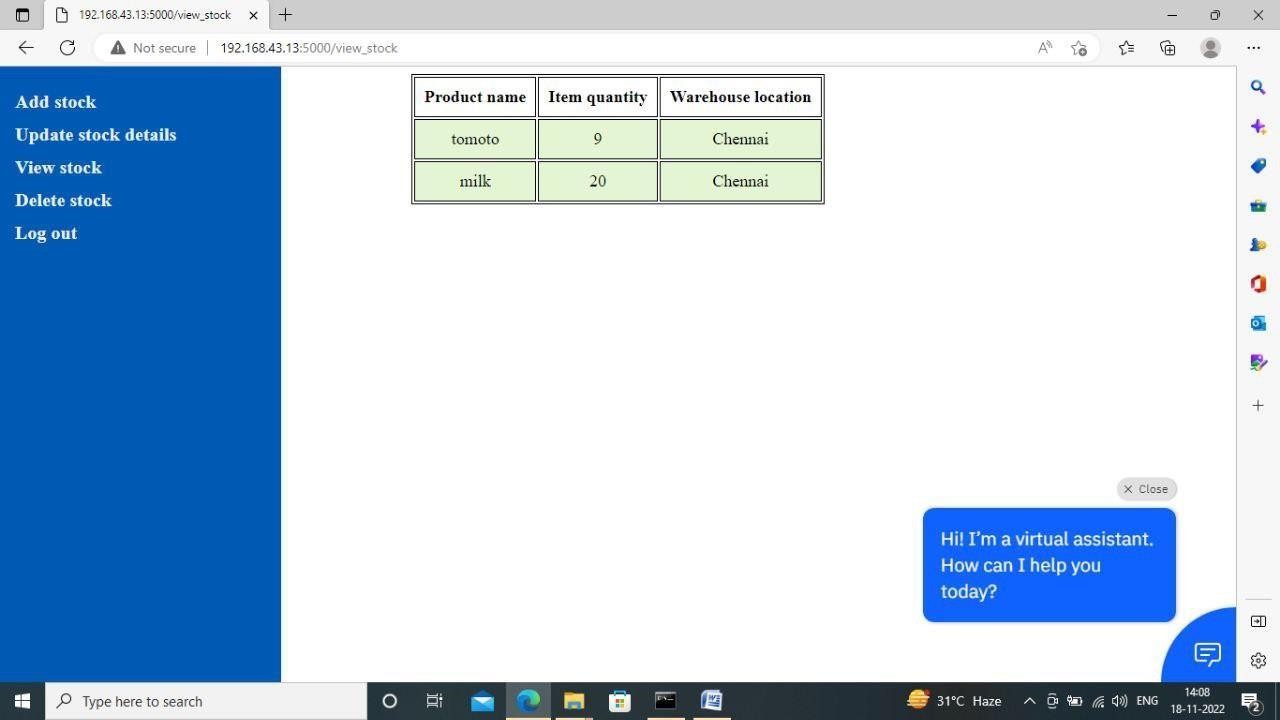
**DASHBOARD**



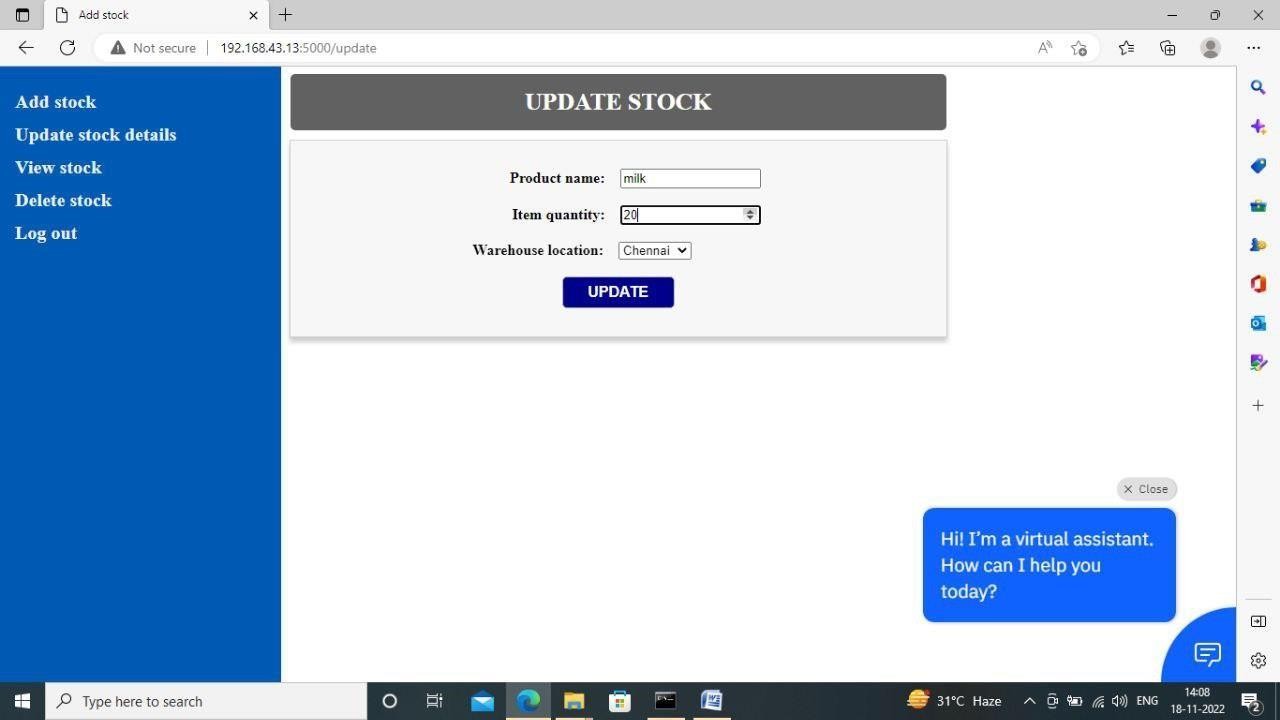
**ADDSTOCK**



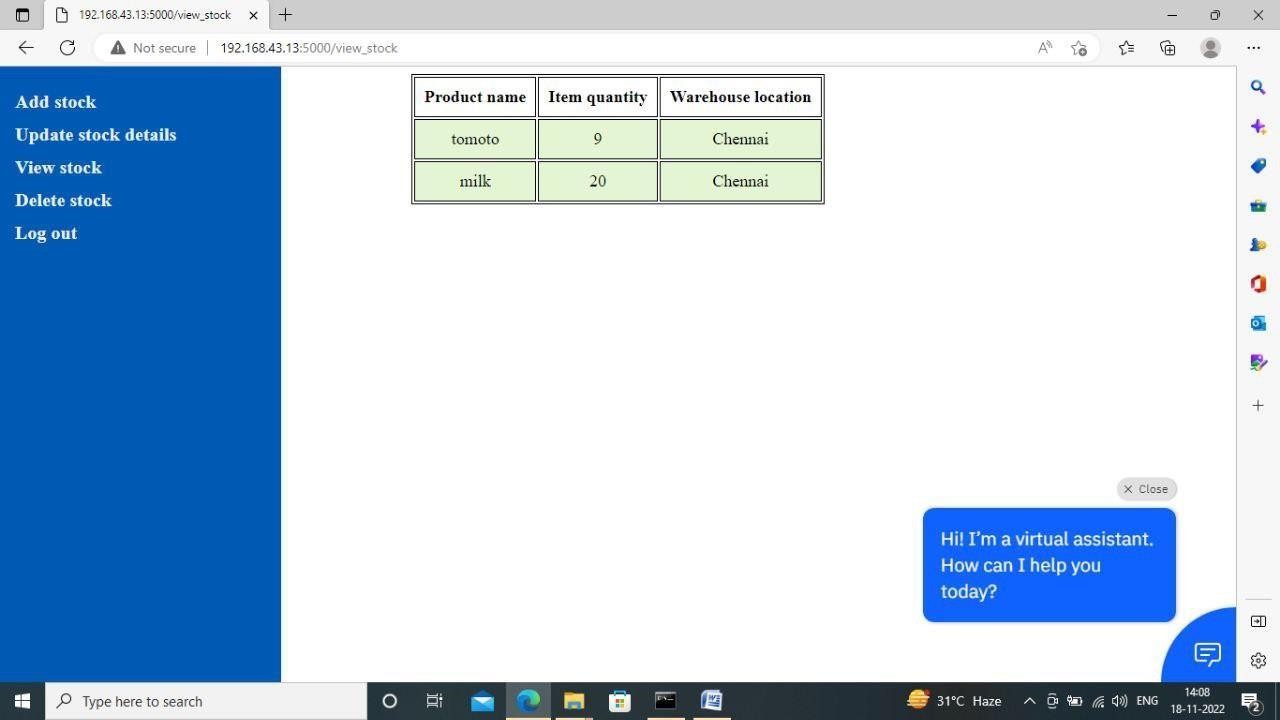
**VIEW STOCK**



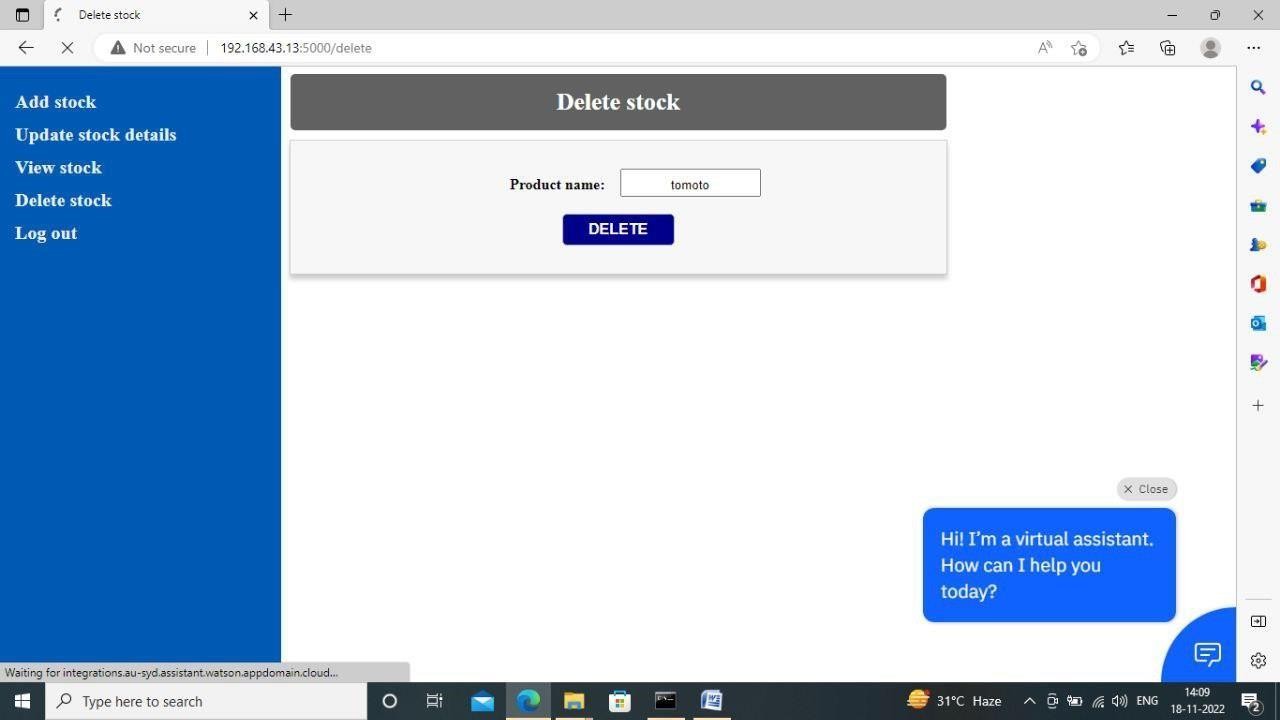
**UPDATE STOCK**

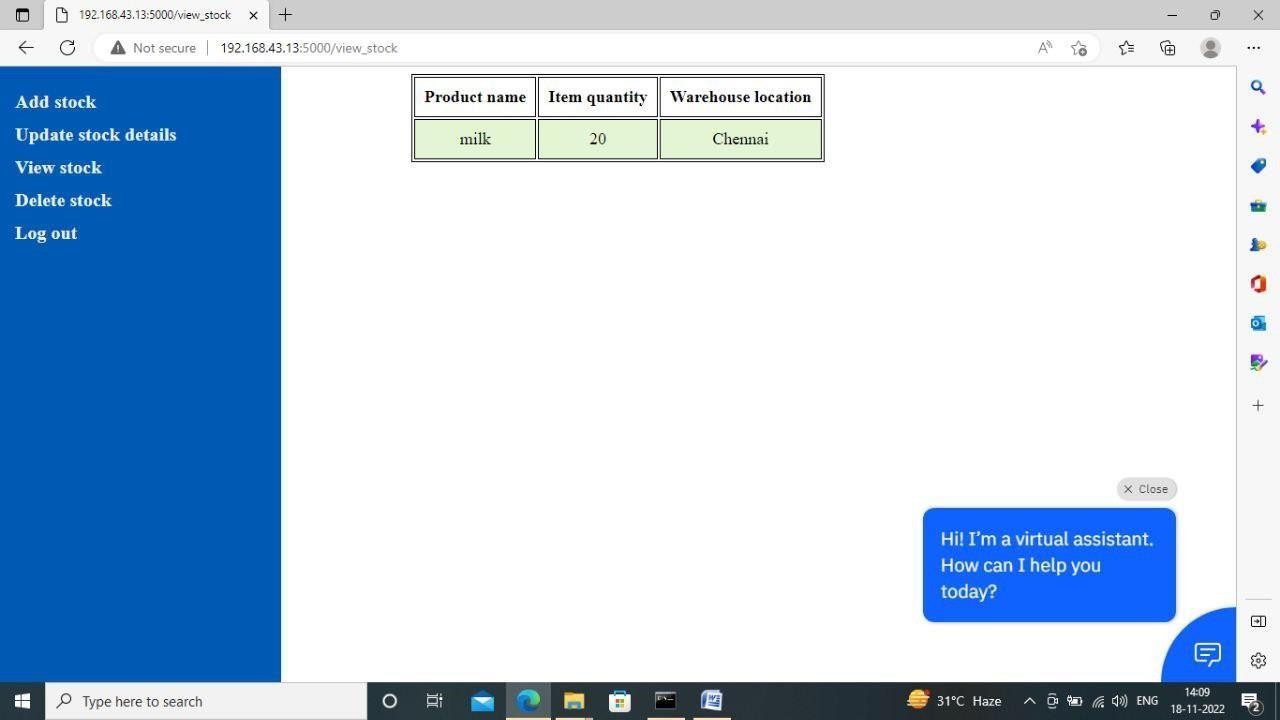


**UPDATESTOCKDETAILS**

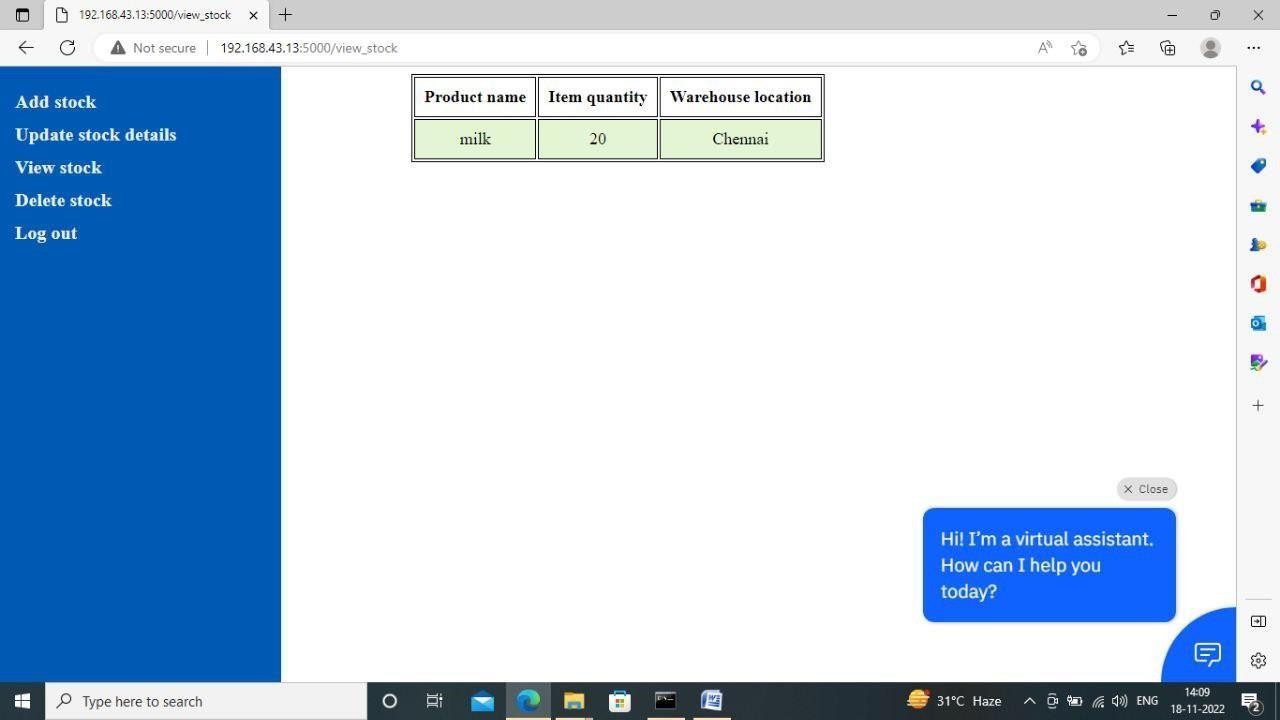


**DELETE STOCK**





**LOGOUT**



# PERFORMANCE METRICS

Inventory Performance is a measure of how effectively and efficiently inventory is used and replenished. The goal of inventory performance metrics is to compare actual on-hand dollars versus forecasted cost of goods sold. Many Lean practitioners claim that inventory performance is the single best indicator of the overall operational performance of a facility. Inventory performance looks at and is measured using either Inventory Days OnHand (DOH) or Inventory Turns.

* + - **Inventory Days On-Hand:** The number of days it would take to consume current on- hand inventory. Always measure multiple inventory item numbers in terms of currency (i.e. COGS).
    - **Inventory Turns**: The number of times inventory is replaced in a year.

1. **ADVANTAGES & DISADVANTAGES**
   1. **ADVANTAGES**
      * It helps to maintain the right amount of stocks
      * It leads to a more organized warehouse
      * It saves time and money
      * Improves efficiency and productivity
      * A well-structured inventory management system leads to improved customer retention
      * Avoid lawsuits and regulatory fines
      * Schedule maintenance
   2. **DISADVANTAGES**
      * Bureaucracy
      * Impersonal touch
      * Production problem
      * Increased space is need to hold the inventory
      * Complexity
      * High implementation costs
2. **CONCLUSION**

Inventory management is a very complex but essential part of the supply chain. An effective inventory management system helps to reduce stock-related costs such as warehousing, carrying, and ordering costs.

1. **FUTURE SCOPE**

According to Easy Post, 'Companies can reap a 25% increase in productivity, a 20% gain in space usage, and a 30% improvement in stock use efficiency if they use integrated order processing for their inventory system. Advanced mobile applications allow companies to manage their inventory and supply chains effectively.

**14.APPENDIX**

# App.py

from flask import Flask, render\_template, request, redirect, url\_for, session, flash import ibm\_db import sqlite3 as sql import re

app = Flask(name) app.secret\_key = 'a'

conn = ibm\_db.connect("DATABASE=bludb;HOSTNAME=1bbf73c5- d84a-

4bb0- 85b9ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain. cloud;PORT=322 86;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID

=w jy24066;PWD=3w6H3sui635KMvWX",'','') print(conn) print("Connecting Successful!!!!!!!!")

@app.route('/') def homer() return

render\_template('home.html')

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

global userid msg = ''

if request.method == 'POST' :

username = request.form['username'] password = request.form['password'] sql = "SELECT \* FROM users WHERE username =? AND password=?" stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,username)

ibm\_db.bind\_param(stmt,2,password) ibm\_db.execute(stmt) account = ibm\_db.fetch\_assoc(stmt) print (account) if account:

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

= account['USERNAME'] userid= account['USERNAME'] session['username'] = account['USERNAME'] msg = 'Logged in successfully !' return render\_template('dashboard.html', msg = msg) else:

msg = 'Incorrect username / password

!' return render\_template('login.html', msg

= msg)

@app.route('/register', methods =['GET', 'POST']) def registet(): msg = '' if request.method == 'POST' :

username = request.form['username'] email = request.form['email'] password = request.form['password'] sql = "SELECT \* FROM users WHERE username =?" stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,username)

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

msg = 'Account already exists !' elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

msg = 'Invalid email address !' elif not re.match(r'[A-Za-z0-9]+', username):

msg = 'name must contain only characters and numbers !' else:

insert\_sql = "INSERT INTO users VALUES (?, ?,

?)" prep\_stmt = ibm\_db.prepare(conn, insert\_sql) ibm\_db.bind\_param(prep\_stmt, 1,

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

email) ibm\_db.bind\_param(prep\_stmt, 3, password) ibm\_db.execute(prep\_stmt) msg = 'Please fill out the form !' if request.method == 'POST':

msg = 'You have successfully registered! Please login

!' return render\_template('register.html', msg = msg)

@app.route('/add\_stock',methods=['GET','POST'])

def add\_stock(): msg='' if

request.method == "POST":

prodname=request.form['prodname'] quantity=request.form['quantity'] warehouse\_location=request.form['warehouse\_loc ation'] sql='SELECT \* FROM product WHERE prodname =?' stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,prodname) ibm\_db.execute(stmt) acnt=ibm\_db.fetch\_assoc(stmt) print(acnt)

if acnt:

msg='Product already exits!!' else:

insert\_sql='INSERT INTO product VALUES (?,?,?)' pstmt=ibm\_db.prepare(conn, insert\_sql) ibm\_db.bind\_param(pstmt,1,prodname) ibm\_db.bind\_param(pstmt,2,quantity) ibm\_db.bind\_param(pstmt,3,warehouse\_location) ibm\_db.execute(pstmt) msg='You have

successfully added the products!!' return render\_template("dashboard.html")

else:

msg="fill out the form first!" return render\_template('add\_stock.html',meg=msg) @app.route('/delete\_stock',methods=['GET','P OST']) def delete\_stock(): if(request.method=="POST"):

prodname=request.form['prodname'] sql2="DELETE FROM product WHERE prodname=?" stmt2 = ibm\_db.prepare(conn, sql2) ibm\_db.bind\_param(stmt2,1,prodname) ibm\_db.execute(stmt2)

flash("Product Deleted", "success")

return render\_template("dashboard.html")

@app.route('/update\_stock',methods=['GET','PO ST']) def update\_stock():

mg='' if request.method == "POST":

prodname=request.form['prodname'] quantity=request.form['quantity'] quantity=int(quantity) print(quantity) print(type(quantity)) warehouse\_location=request.form['warehouse\_loc ation'] sql='SELECT \* FROM product WHERE prodname =?' stmt = ibm\_db.prepare(conn, sql) ibm\_db.bind\_param(stmt,1,prodname) ibm\_db.execute(stmt) acnt=ibm\_db.fetch\_assoc(stmt)

print(acnt)

if acnt: insert\_sql='UPDATE product SET quantity=?,warehouse\_location=? WHERE prodname=? '

pstmt=ibm\_db.prepare(conn, insert\_sql) ibm\_db.bind\_param(pstmt,1,quantity) ibm\_db.bind\_param(pstmt,2,warehouse\_location

) ibm\_db.bind\_param(pstmt,3,prodname) ibm\_db.execute(pstmt) mg='You have successfully updated the products!!'

limit=5 print(type(limit)) if(quantity<=limit):

("Please update the quantity of the product {}, Atleast {} number of pieces must be added!".format(prodname,10)) return render\_template("dashboard.html",meg=mg)

else:

mg='Product not found!!'

else:

msg="fill out the form first!" return render\_template('update\_stock.html',meg=msg)

@app.route('/view\_stoc k') def view\_stock():

sql = "SELECT \* FROM product" stmt = ibm\_db.prepare(conn, sql)

result=ibm\_db.execute(stmt) print(result)

products=[] row = ibm\_db.fetch\_assoc(stmt) print(row) while(row):

products.append(row) row

= ibm\_db.fetch\_assoc(stmt) print(row) products=tuple(products) print(products)

if result>0:

return render\_template('view.html', products = products) else:

msg='No products found' return render\_template('view.html', msg=msg)

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

return render\_template('delete\_stock.html')

@app.route('/updat e') def update():

return render\_template('update\_stock.html')

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

session.pop('loggedin', None) session.pop('id', None) session.pop('username', None) return render\_template('home.html')

if name == 'main': app.run(host='0.0.0.0')

# home.html

<!DOCTYPE html>

<html>

<head>

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

href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/4.7.0/css

/font-awesome.min.css">

<style> body { font- family: "Lato", sans-serif;

}

/\* Fixed sidenav, full height \*/

.sidenav { height:

100%;

width:

300px; position:

fixed; z- index: 1;

top: 0; left:

0;

background-color: #0059b3; overflow-x: hidden; padding-top: 20px;

}

/\* Style the sidenav links and the dropdown button \*/

.sidenav a{ padding: 6px 8px 6px 16px; text- decoration: none; font- size: 20px; color: rgb(239, 239, 239);

display: block; border: none; background: none; width: 100%;

text-align:

left; cursor: pointer; outline:

none;

}

/\* On mouse-over \*/

.sidenav a:hover{ color: #111;

}

/\* Some media queries for responsiveness \*/ @media screen and (max-height: 450px) {

.sidenav {padding-top: 15px;}

.sidenav a {font-size: 18px;}

}

</style>

</head>

<body>

> Deeps:

<div class="sidenav">

<a href="{{url\_for('add\_stock') }}"><strong>Add stock<strong></a>

<a href="{{url\_for('update') }}"><strong>Update stock details<strong></a>

<a href="{{url\_for('view\_stock') }}"><strong>View stock<strong></a>

<a href="{{url\_for('delete')}}"><strong>Delete stock<strong></a>

<a href="{{url\_for('logout') }}"><strong>Log out<strong></a>

</div>

<nav>

<script> window.watsonAssistantChatOptions = { integrationID: "4bd6f313-33d4-4e87-8825-22b90b8e3c2c", // The ID of this integration. region: "au-syd", // The region your

integration is hosted in. serviceInstanceID: "60e1396a-421f- 4091-b39a-a23a546843e8", // 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(

});

</script>

</nav>

</body>

</html>

**GITHUB:**

**https://github.com/IBM-EPBL/IBM-Project-38526-1660382075**

**DEMOLINK:**



**https://1drv.ms/v/s!Au3985NsWb5CgQIUPkrXBiC7qaLV?e=62Fjjp**