INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

DOMAIN: CLOUD APPLICATION DEVELOPMENT

-

-

TEAM MEMBERS:

1. SARANYA D
2. AKSHITA S
3. DEEKSHANA A
4. ILAVARASI A

INDEX

1.INTRODUCTION

1.1 Project Overview

The inventory management system is developed to overcome the downside of traditional management systems. The application is deployed in cloud. ISTOCK can handle stocks of a business with ease. Furthermore, this system is developed to handle the needs of various companies and not just one. The application has minimal error tolerance and stores all the data securely. The project Inventory Management System is a complete desktop-based application designed on Net technology using Visual Studio Software. The main aim of the project is to develop an Inventory Management System Model software in which all the information regarding the stock of the organization will be presented. It is an intranet-based desktop application that has an admin component to manage the inventory and maintenance of the inventory system. This desktop application is based on the management of the stock of an organization.

The application contains a general organization profile, sales details, Purchase details, and the remaining stock that is presented in the organization. There is a provision for updating the inventory also. This application also provides the remaining balance of the stock as well as the details of the balance of the transaction. Each new stock is created and entitled with the name and the entry date of that stock and it can also be updated any time when required as per the transaction or the sales are returned in case. Here the login page is created in order to protect the management of the stock of the organization in order to prevent it from the threads and misuse of the inventory.

An Inventory Management System also aids in the tracking of retail product theft, providing useful information regarding store revenues and the need for theft-prevention devices. Scanning a barcode on the item or a barcode scanner is how Automated Inventory Management Systems function. The central computer system then keeps track of this data. The purchase order can also include a list of items that need to be pulled for packaging and shipping. In this situation, the Inventory Management System can perform a range of tasks.

This is a system that is capable of executing repetitive tasks with little manual help, once a set of rules have been set up. This not only helps you have real-time visibility on your inventory levels as your stock count automatically updates when a sale is made. This feature is critical not only to accurate forecasting but also to delivering a good customer experience by avoiding overselling. Automated inventory management also gives you real-time visibility on where your stock is, which is essential particularly if you store stock in multiple locations like a warehouse and a physical store, or with more than one selling channel.

1.2 Purpose

Every business small or big needs needs to handle the stock that comes in and out. Handling stock is hard especially if the incoming product amount is large. Traditional methods, without the involvement of computers or even net have been developed by many over the years. However, they aren’t the best solutions to handle a business better. ISTOCK, permits business owners to keep track of their incoming and outgoing products and suppliers efficiently.

One of the most valuable assets of a company is its inventory. In various industries, such as retail, food services, and manufacturing, a lack of inventory can have detrimental effects. Aside from being a liability, inventory can also be considered a risk. It can be prone to theft, damage, and spoilage. Having a large inventory can also lead to a reduction in sales.

Regardless of the size of your company, having a proper inventory management system is very important for any business. It can help you keep track of all your supplies and determine the exact prices. It can also help you manage sudden changes in demand without sacrifice customer experience or product quality. This is especially important for brands looking to become a more customer oriented.

There is a provision for updating the inventory also. This application also provides the remaining balance of the stock as well as the details of the balance of the transaction. Each new stock is created and entitled with the name and the entry date of that stock and it can also be updated any time when required as per the transaction or the sales are returned in case. Here the login page is created in order to protect the management of the stock of the organization in order to prevent it from the threads and misuse of the inventory.

Balancing the risks of overstocks and shortages is an especially challenging process for companies with complex supply chains. A company's inventory is typically a current asset that it plans to sell within a year. It must be measured and counted regularly to be considered a current asset

LITERATURE SURVEY

S. NO	TITLE	AUTHORS	ABSTRACT
1.	Inventory management system	Anish Singh Maharjan, Mandip Humagain	This project is aimed at developing a desktop-based application named Inventory Management System for managing the inventory system of any organisation. The Inventory Management System (IMS) refers to the system and processes to manage the stock of an organisation with the involvement of Technology system. This system can be used to store the details of the inventory, stock maintenance, update the inventory based on the sales details, generate sales and inventory reports daily or weekly based. This project is categorised individual aspects for the sales and inventory management system. Inventory Management System is important to ensure quality control in businesses that handle transactions resolving around consumer goods.

2.	Research paper on Inventory management system	Punam Khobragade, Roshni Selokar , Rina Maraskolhe Prof.Manjusha Talmale	Inventory Management System is software which is helpful for the businesses operate hardware stores, where storeowner keeps the records of sales and purchase. Mismanaged inventory means disappointed customers, too much cash tied up in warehouses and slower sales. This project eliminates the paper work, human faults, manual delay and speed up process. Inventory Management System will have the ability to track sales and available inventory, tells a storeowner when it's time to reorder and how much to purchase.Inventory Management System is a windows application developed for Windows operating systems which focused in the area of Inventory control and generates the various required reports.

3	A Study of Inventory Management System Case Study	Tariq Sheakh	Inventory management is a challenging problem area in supply chain management. Companies need to have inventories in warehouses in order to fulfil customer demand, meanwhile these inventories have holding costs and this is frozen fund that can be lost. Therefore, the task of inventory management is to find the quantity of inventories that will fulfil the demand, avoiding overstocks. This paper presents a case study for the steel manufacturing industry (Small Scale Industry) on inventory management. T. The study also proved that there was a significant relationship between return on asset (ROA) and inventory days. This paper also provides recommendation to the company and for further research.

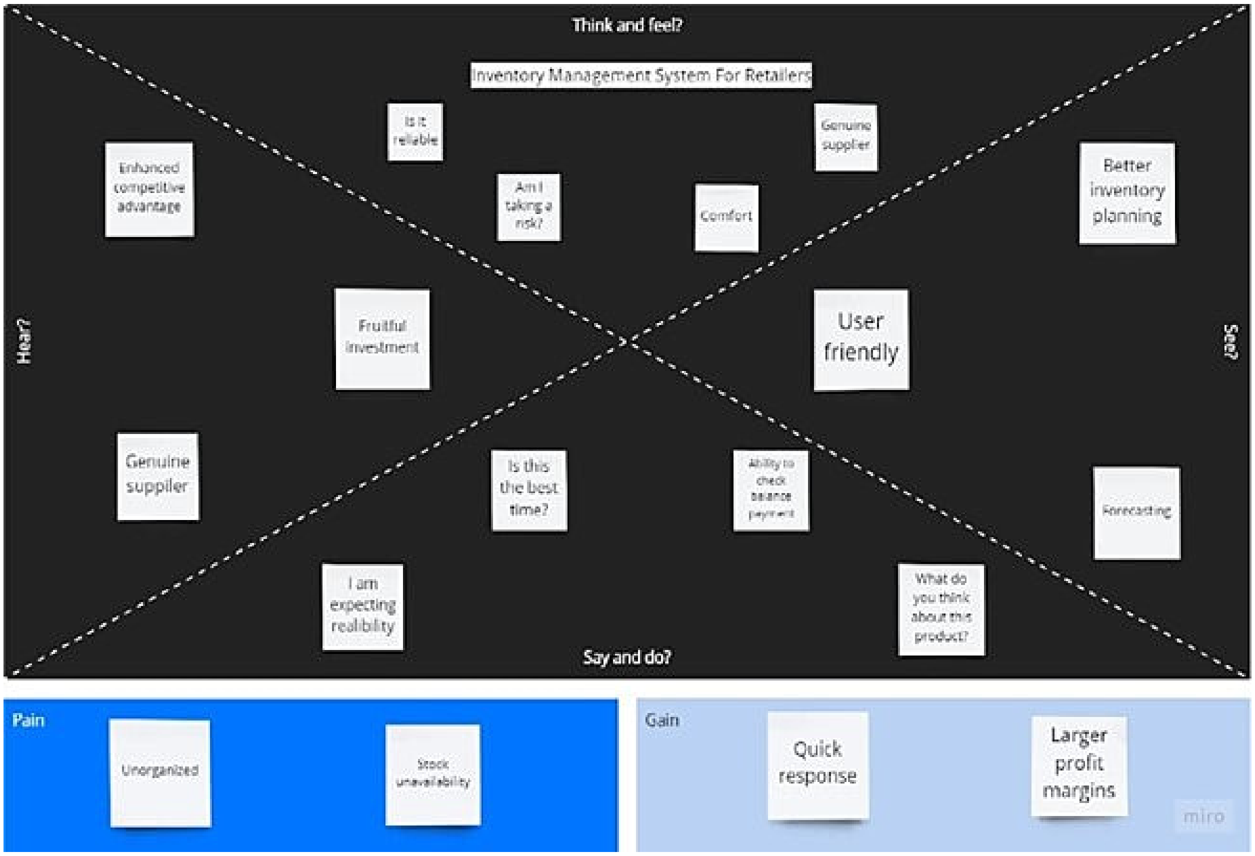
4	Performance Improvement of Inventory Management System	Anas M. Atieh,Hazem Kaylani,Yousef Al-abdallat,Abeer Qaderi,Luma Ghoul, Lina Jaradat,Iman Hdairis	This study investigates the impact of a warehouse management system on supply chain performance that provides less resources effort, more efficient, and reliable inventory management system. The supply chain procedures carried out in the warehouse were reviewed before customizing a software that can handle the necessary transactions. The software was tested for enhancing the work flow and providing a timely and efficient handling.This work can serve both as a practical guide and industrial example for some researchers to compare the software inventory management system with the traditional manual system in the telecommunications sector in Jordan. It also highlights the gap between theory and practice; to motivate researchers to develop and customize new systems for mitigating supply chain disruptions.

5	Study of smart inventory management system	Souvik Paul,Atrayee Chatterjee,Digbijay Guha	n developing enterprises and the constant demands of the product diversity, traditional Inventory Managem heavy workload and low efficiency. This paper presents a new type of intelligent Inventory Management S principles and structure of it. This system has great advantages compared to the traditional mode, and we ex Inventory Management is a key area for customer service and cost optimization in any manufacturing set thousands of components and hundreds of warehouses the inventory becomes a nightmare and a lot of ensuring right shipments. Traditional systems of robotic arms for inventory pick and drop have been bas warehouse and tracking it.

6	Research and Design of the Intelligent Inventory Management System Based on RFID	Xiaojun Jing,Peng Tang	This paper introduces the characteristics and basic application of RFID technology, analyses the data flow of intelligent inventory system from the perspective of business and function, then puts forward the specific framework programs and function modules of intelligent inventory management system based on IOT RFID technology, focuses on elaborating the design and implementation process of the intelligent inventory system. The system realizes full control and management of all products, faster in/out warehouse and dynamic inventory, utilizes warehouse efficiently and improves the capacity of warehouse by effective combining with the ERP system in enterprise.

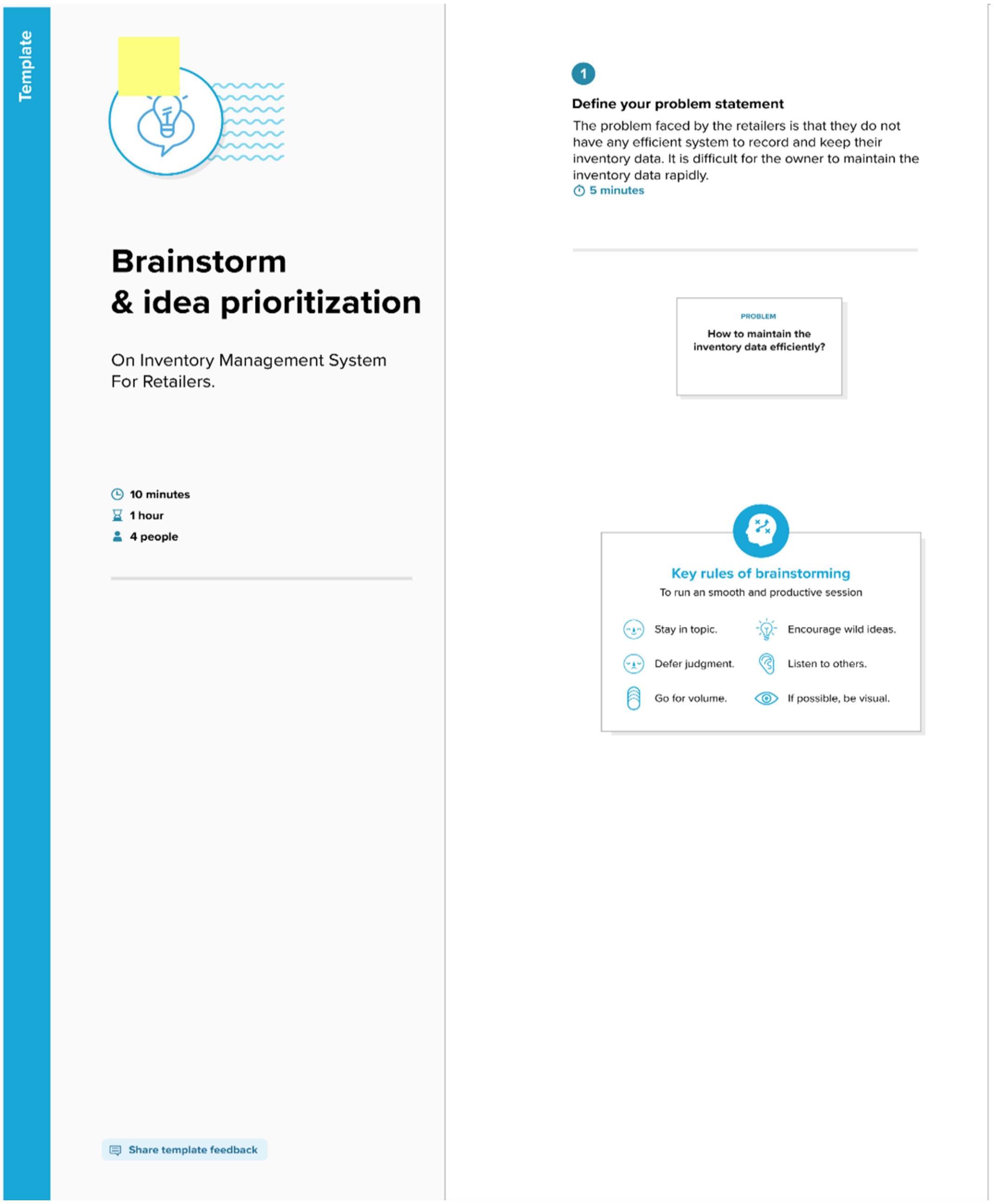
3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Brainstorm & Idea Prioritization:

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



Brainstorm & idea prioritization

On Inventory Management System For Retailers.

- 10 minutes
- 1 hour
- 4 people

1

Define your problem statement

The problem faced by the retailers is that they do not have any efficient system to record and keep their inventory data. It is difficult for the owner to maintain the inventory data rapidly.

5 minutes

PROBLEM

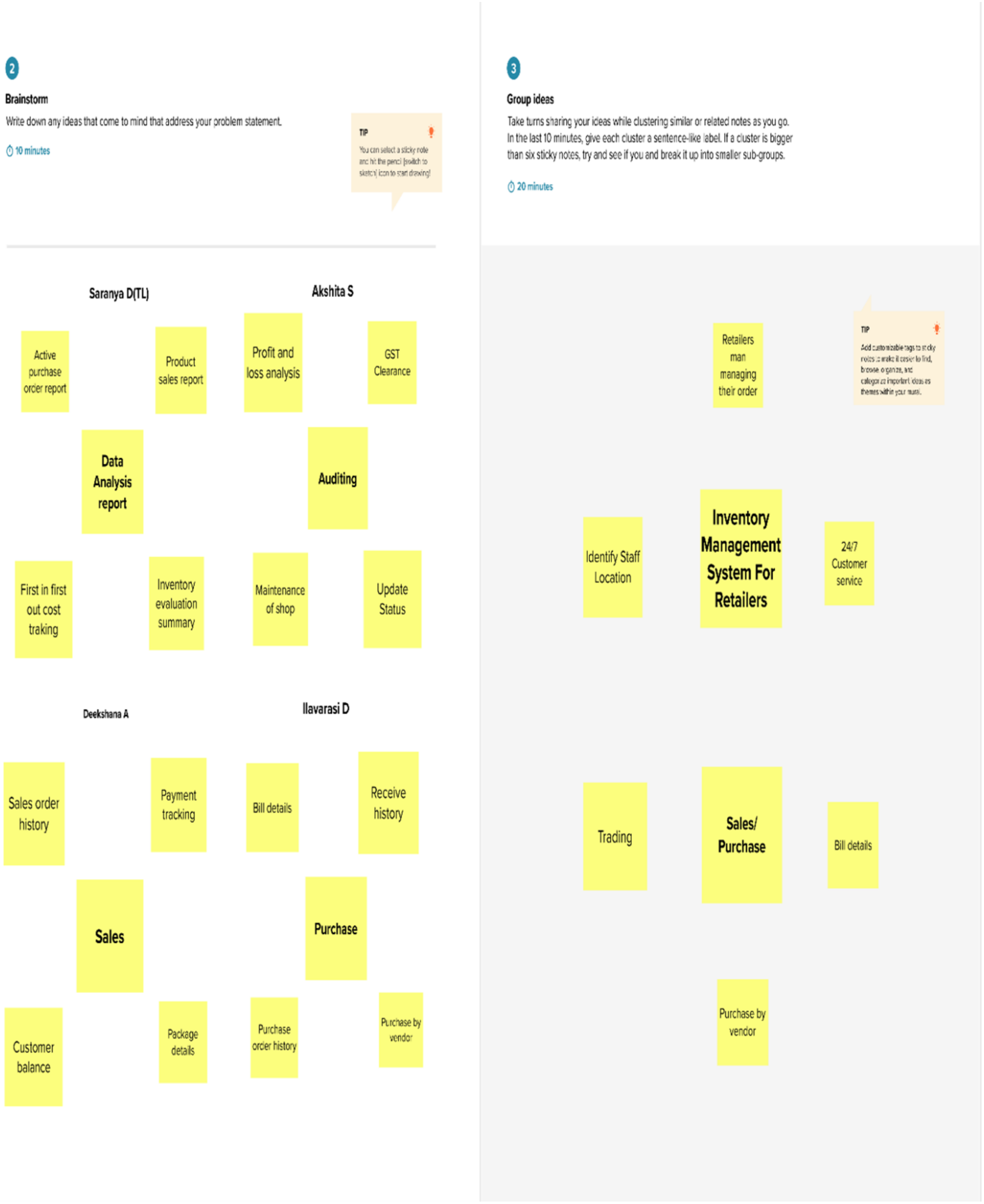
How to maintain the inventory data efficiently?

Key rules of brainstorming

To run an smooth and productive session

- Stay in topic.
- Encourage wild ideas.
- Defer judgment.
- Listen to others.
- Go for volume.
- If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

TIP

You can select a sticky note and hit the pencil (switch to sketch) icon to start drawing!

3

Group ideas

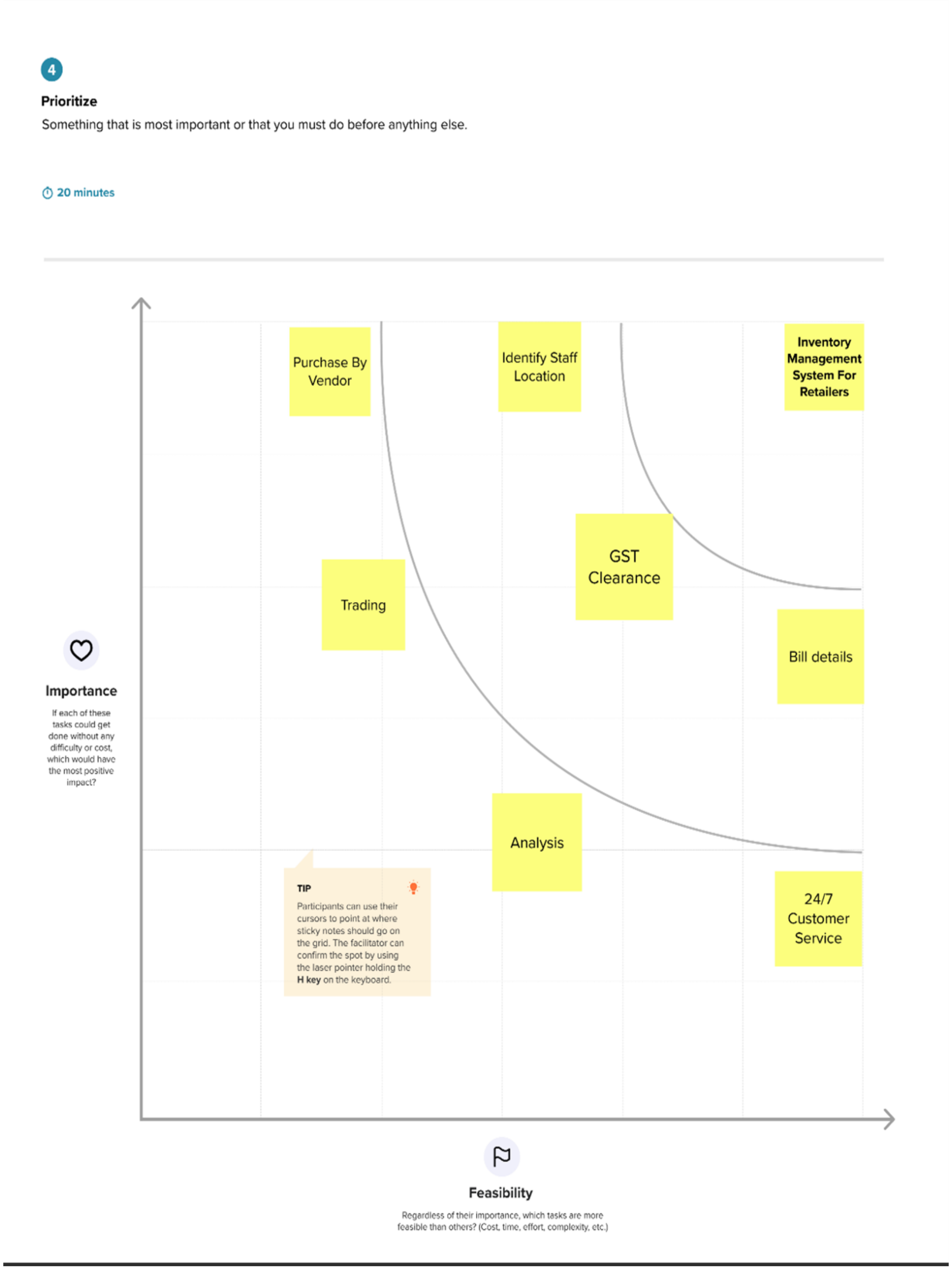
Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Step-3: Idea Prioritization



4

Prioritize

Something that is most important or that you must do before anything else.

⏱ 20 minutes

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

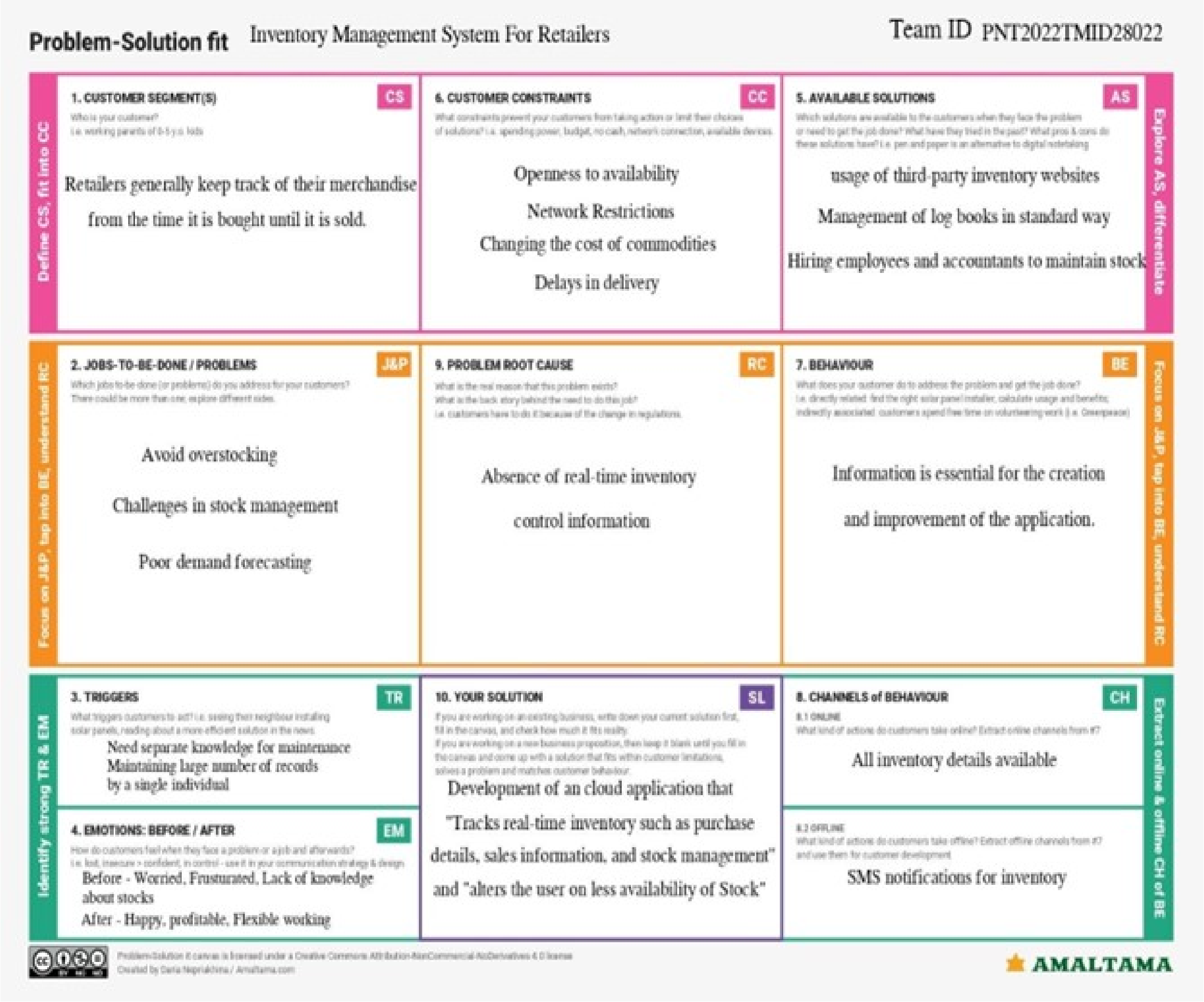
Purchase By Vendor
Identify Staff Location
Inventory Management System For Retailers
GST Clearance
Trading
Bill details
Analysis
24/7 Customer Service
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The problem statement aims to make desktop application forretailersandtotrackall areasof Inventory Management System like purchase details , salesdetails , stockmanagement and other policies .
2.	Idea / Solution description	The application is developed to help retailers track and manage stocksrelated to theirown products. The Systemwill ask the retailers to create their accounts by providing essential details. Retailers can access their accounts by logging into the application. Once retailers successfully log in to the application they can update their inventory details, alsousers will be able toadd new stockby submitting essential details related to thestock. They can viewdetails of thecurrent inventory. The System willautomatically send anemail alert to the retailers if there is no stock found in their accounts. So that they can order new stock.
3.	Novelty / Uniqueness	Reduced costs. Prioritize responsiveness. Sending mail notification in case of low stock availability.
4.	Social Impact / Customer Satisfaction	It makes it easier for retailers to control their whole stock warehouse from a single platform. Additionally, itaidsinthe management oftheir inventories' supply and demand. A good inventory management system lowers the possibility of overstocking and stops retailers fromwasting money andproduct.
5.	Business Model (Revenue Model)	Retailers can orderthe right amount and type of stock at the right time with the aid of an inventory management system. It eliminates the unnecessary expense for the retailers.

6.	Scalability of the Solution	With the helpof these technologies, businesses can automatically refill low-stock inventory before it runs out,preventing lost sales opportunities. This streamlines order management. Similarly, if someitems aren't selling as anticipated, overstocking can be prevented.

3.4 Problem- Solution fit



Problem-Solution fit Inventory Management System For Retailers

Team ID PNT2022TMID28022

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Retailers generally keep track of their merchandise from the time it is bought until it is sold.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

Openness to availability

Network Restrictions

Changing the cost of commodities

Delays in delivery

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

usage of third-party inventory websites

Management of log books in standard way

Hiring employees and accountants to maintain stock

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

Avoid overstocking

Challenges in stock management

Poor demand forecasting

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

Absence of real-time inventory control information

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

Information is essential for the creation and improvement of the application.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

Need separate knowledge for maintenance
Maintaining large number of records
by a single individual

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Before - Worried, Frusturated, Lack of knowledge about stocks

After - Happy, profitable, Flexible working

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

Development of an cloud application that "Tracks real-time inventory such as purchase details, sales information, and stock management" and "alters the user on less availability of Stock"

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7

All inventory details available

8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

SMS notifications for inventory

Extract online & offline CH of BE

AMALTAMA

4.REQUIREMENT ANALYSIS

4.1 Functional Requirements:

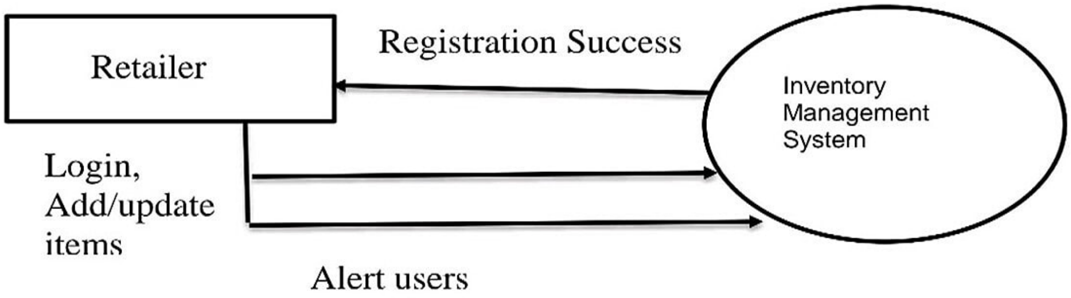
FR No.	**Functional Requirement (Epic)**	**Sub Requirement (Story / Sub-Task)**
FR-1	**User Registration**	Registration through Gmail
FR-2	**Log In**	Log In via email and password.
FR-3	**Reporting Requirements**	Once the stock reaches the minimum alert is sent to the mail.
FR-4	**Product**	Users can edit or delete product in the product tab.
FR-5	**Supplier**	Users can edit or supplier information in the supplier tab.
FR-6	**Reorder**	User can send email to reorder the stock.

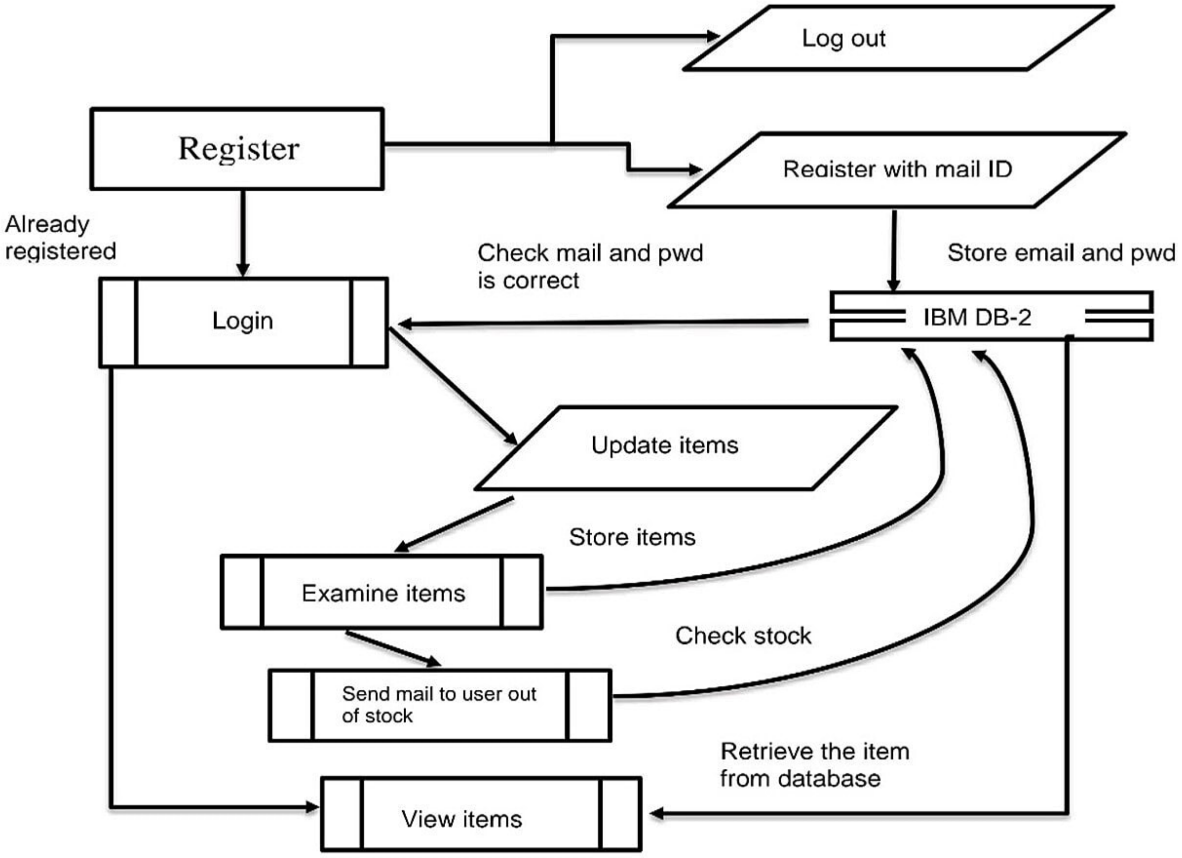
4.2 Non-functional Requirements:

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	It is ease to handle the app, navigate and efficient from the user point of view.
NFR-2	**Security**	The application get only name and Mail Id/Phone Number .It doesn’t get additional personal information from the user.
NFR-3	**Reliability**	The probability of the system getting fail is very less as the code used in the program is minimum and does not utilize more time and cause run time failure during execution.
NFR-4	**Performance**	The launch time and load time is less and the app size is small .
NFR-5	**Availability**	Available free in play store and premium account requires only minimum amount.
NFR-6	**Scalability**	The app is capable to handle more users and evolving concurrently to the user needs.

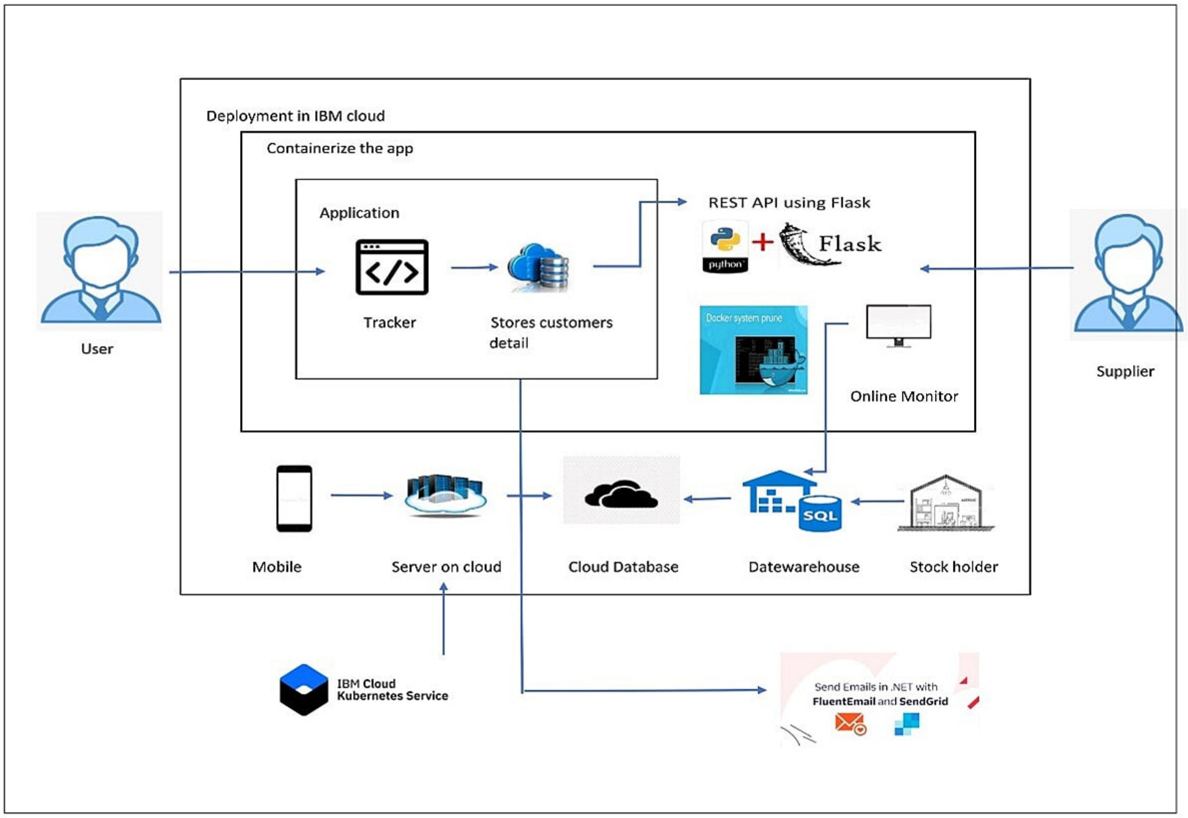
5.PROJECT DESIGN

5.1 Data Flow Diagram





5.2 Solution and Technology Architecture:



5.3 User Stories

User Type	Functional Requirement(Epic)	User Story Number	User Story / Task	Acceptancecriteria	Priority	Release
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	Account created	High	Sprint-1
	Login	USN-2	As a user, I can log in to the application by entering email& password	I can access my account / dashboard	High	Sprint-1
	Dashboard	USN-3	As a user, I can view the available stocks and list of suppliers	Once I log in to theapplication, I can view stocks.	High	Sprint-2

User Type	Functional Requirement(Epic)	User Story Number	User Story / Task	Acceptancecriteria	Priority	Release
	Add items	USN-4	As a user, I can add the stocks to the inventory.	As a user, I can add the stocks required for purchasing.	Medium	Sprint-2
	Add supplier	USN-5	As a user, I can add the suppliers to the inventory.	As a user, I can add the suppliers required for purchasing of stocks.	Medium	Sprint-3
	Stock and Supplier Update	USN-6	As a user, I can update the table.	As a user, I can update the supplier and stock table after adding or removing the stock or supplier.	Medium	Sprint-3
	Re-Order	USN-7	As a user, I can order the products when	As a user, I can contact supplier to deliver products.	Low	Sprint-4

			in need.			
	Notify on less stock	USN-8	As a user, I am notified when the stock is less.	As user, I can givemy support in my possible ways to administrator and the administration.	High	Sprint-4

6.PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

TITLE	**DESCRIPTION**	**DATE**
Literature Survey & Information Gathering	Literature survey on the selected project & gathering information by referring the, technical papers, research publications etc.	3 SEPTEMBER 2022.
Prepare Empathy Map	Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements	10 SEPTEMBER 2022
Ideation	organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.	10 SEPTEMBER 2022
Proposed Solution	Prepare the proposed solution document, which includes the scalability of solution ,idea, novelty business model, social impact, etc.	24 SEPTEMBER 2022
Problem Solution Fit	Prepare problem - solution fit document	01 OCTOBER 2022
Solution Architecture	Prepare solution architecture document.	08 OCTOBER 2022

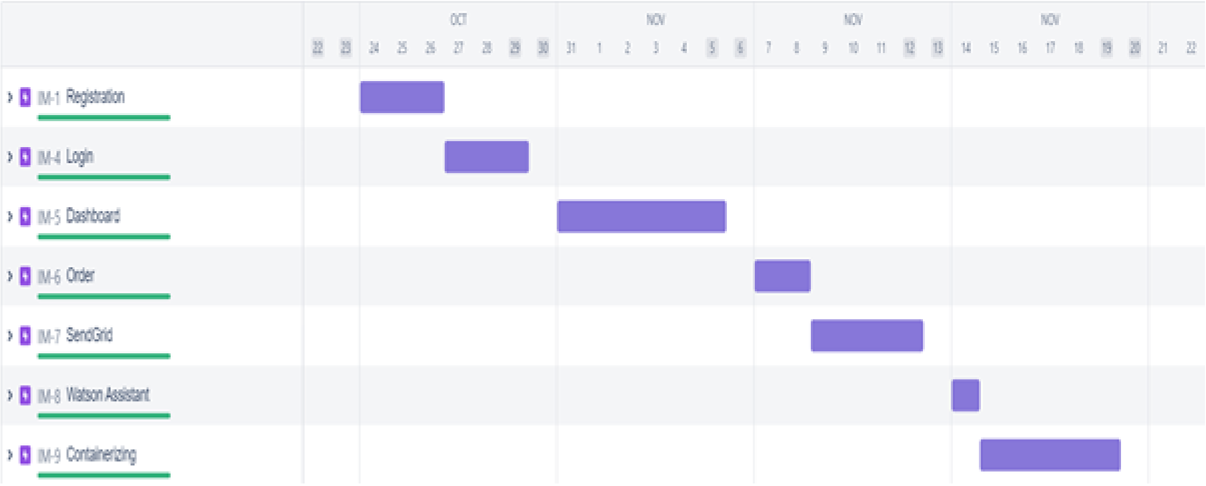
Customer Journey	Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).	08 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	15 OCTOBER 2022
Data Flow Diagrams	Prepare the functional requirement document.	15 OCTOBER 2022
Technology Architecture	Prepare the technology architecture diagram.	15 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones & activity list of the project.	18 OCTOBER 2022
Sprint Delivery Plan	Prepare sprint delivery plan	18 OCTOBER 2022
Project Development - Delivery of Sprint-1, 2, 3 & 4	Develop & submit the developed code by testing it.	18 NOVEMBER 2022

6.2 Sprint delivery schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Saranya, Akshita
Sprint-1	Login	USN-2	As a user, I can log into the application by entering email & password	2	Medium	Ilavarasi
Sprint-1		USN-3	As a user, I can log into the application by entering user name & password	1	Low	Deekshana
Sprint-2	Dashboard	USN-4	As a user, I can enter into the dashboard and view inventory	2	High	Deekshana, Saranya
Sprint-2		USN-5	As a user, I can add and update details of products and supplier	2	Medium	Akshita, Deekshana

Sprint-2		USN-6	As a user, I can delete and search for the details of products and supplier.	2	Medium	Ilavarasi, Saranya
Sprint-3	Order	USN-7	As a user, I can order the products based on needs.	2	High	Ilavarasi, Akshita
Sprint-3	SendGrid	USN-8	As a user, I can receive Alerts and messages via email when the stocks are below the minimum stock quantity	3	high	Saranya, Akshita
		USN-9	As a user, I can request for the inventory report through email.	1	Medium	Deekshana, Ilavarasi
Sprint-4	Watson Assistant	USN-10	As a user, I can clarify the queries using chatbot	2	Medium	Saranya, Akshita
Sprint-4	Containerizing	USN-11	Containerizing the application.	4	High	Saranya, Akshita, Ilavarasi, Deekshana

6.3 Reports from JIRA:



7 CODING AND SOLUTION

7.1 FEATURE 1

HOME:

The home page is the first page in the app and contains the link to register and log in page. It also has the Contact page, which details about the management system.

HomePage.html

```
<!DOCTYPE html>
<!--Code by Divinector (www.divinectorweb.com)-->
<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">
    <title>ISTOCK</title>
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700;900&display=swap" rel="stylesheet">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.6.2/animate.min.css" rel="stylesheet">
    <link rel="stylesheet" href="{{url_for('static',filename='css/HomePage.css')}}">

</head>
<body>

    <header>
        <nav class="navbar navbar-default navbar-fixed-top navbar-inverse">
```

```
      <div class="container">
        <!-- Brand and toggle get grouped for better mobile display -->
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>

        </div>

        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">

          <ul class="nav navbar-nav navbar-right">

            <li><a href="sign">SignUp/SignIn</a></li>

            <li><a href="#">contact</a></li>
          </ul>
        </div><!-- /.navbar-collapse -->
      </div><!-- /.container-fluid -->
    </nav>


    <div id="carousel-example-generic" class="carousel slide" data-ride="carousel">
      <!-- Indicators -->
```

```
            <ol class="carousel-indicators">
                <li data-target="#carousel-example-generic" data-slide-to="0" class="active"></li>
                <li data-target="#carousel-example-generic" data-slide-to="1"></li>
                <li data-target="#carousel-example-generic" data-slide-to="2"></li>
            </ol>

            <!-- Wrapper for slides -->
            <div class="carousel-inner" role="listbox">
                <div class="item active">
                    <div class="banner" style="background-image:
url(https://i.postimg.cc/pTGKnyy0/1.jpg);"></div>
                    <div class="carousel-caption">
                        <h2 class="animated bounceInRight" style="animation-delay: 1s">We Are
<span>ISTOCK</span></h2>
                        <h3 class="animated bounceInLeft" style="animation-delay: 2s">Inventory
Management Agency</h3>
                        <p class="animated bounceInRight" style="animation-delay: 3s"><a
href="#">Contact us</a></p>
                    </div>
                </div>
                <div class="item">
                    <div class="banner" style="background-image:
url(https://i.postimg.cc/k4Bvsxrr/2.jpg);"></div>
                    <div class="carousel-caption">
                        <h2 class="animated slideInDown" style="animation-delay: 1s">We Are
<span>Yours</span></h2>

                    </div>
                </div>
```

```
        <div class="item">
          <div class="banner" style="background-image: url(https://i.postimg.cc/tgg3Rh41/3.jpg);"></div>
          <div class="carousel-caption">
            <h2 class="animated zoomIn" style="animation-delay: 1s">We are<span> Trustworthy</span></h2>
            <h3 class="animated fadeInLeft" style="animation-delay: 2s">Inventory Management Agency</h3>

          </div>
        </div>

      </div>

      <!-- Controls -->
      <a class="left carousel-control" href="#carousel-example-generic" role="button" data-slide="prev">
        <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
        <span class="sr-only">Previous</span>
      </a>
      <a class="right carousel-control" href="#carousel-example-generic" role="button" data-slide="next">
        <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
        <span class="sr-only">Next</span>
      </a>
    </div>

  </header>
```

```
    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>
</body>
</html>
```

REGISTER:

A new user can register themselves by providing user_email and password. By registering as a new user, they can login anytime.

Register.html

```
<!DOCTYPE html>
<!-- Coding by CodingLab | www.codinglabweb.com-->
<html lang="en" dir="ltr">
  <head>
    <meta charset="UTF-8">
    <!--<title> Login and Registration Form in HTML & CSS | CodingLab </title>-->
    <link rel="stylesheet" href="{{url_for('static',filename='css/Reg.css')}}">
    <!-- Fontawesome CDN Link -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">
      <script src="https://code.jquery.com/jquery-3.4.1.js"></script>
   <title>Inventory Managemanet System For Retailers</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
<body>
  <div class="container">
    <input type="checkbox" id="flip">
  <div class="cover">
   <div class="front">
    <img src="C:\Users\deeks\OneDrive\Desktop\frontImg.jpg" alt="">
    <div class="text">
     <span class="text-1">Every new friend is a <br> new adventure</span>
     <span class="text-2">Let's get connected</span>
    </div>
   </div>
   <div class="back">
    <img class="backImg" src="C:\Users\deeks\OneDrive\Desktop\Img.jpg" alt="">
    <div class="text">
     <span class="text-1">Complete miles of journey <br> with one step</span>
     <span class="text-2">Let's get started</span>
    </div>
   </div>
  </div>
  <div class="forms">
   <div class="form-content">

    <div class="signup-form">
     <div class="title">Signup</div>
    <form action="register" method="post">
      <div class="input-boxes">
       <div class="input-box">
        <i class="fas fa-building"></i>
        <input type="text" placeholder="Enter your company name" name="company"
required>
       </div>
       <div class="input-box">
```

```
          <i class="fas fa-map-marker "></i>
          <input type="text" placeholder="Enter your location" required name="location">
        </div>
        <div class="input-box">
          <i class="fas fa-envelope"></i>
          <input type="text" placeholder="Enter your email" required name="email">
        </div>
        <div class="input-box">
          <i class="fas fa-user"></i>
          <input type="text" placeholder="Enter User ID" required name="username">
        </div>
        <div class="input-box">
          <i class="fas fa-lock"></i>
          <input type="password" placeholder="Enter your password" required
name="password">
        </div>
        {% if mesaagemail %}
            <center> <p style="color:red">{{mesaagemail}}</p></center>
        {% endif %}
        {% if registeraccount %}
            <center> <p style="color:red">{{registeraccount}}</p></center>
        {% endif %}
        {% if registermsg %}
            <center> <p style="color:green">{{registermsg}}</p></center>
        {% endif %}
        <div class="button input-box">
          <input type="submit" value="Sumbit">
        </div>
        <div class="text sign-up-text">Already have an account? <label for="flip">Login
```

```
now</label></div>
            </div>
      </form>
     </div>

    <div class="login-form">
    <div class="title">Login</div>
    <form action="login" method="post">
      <div class="input-boxes">
        <div class="input-box">
          <i class="fas fa-envelope"></i>
          <input type="text" placeholder="Enter your email" required name="username">
        </div>
        <div class="input-box">
          <i class="fas fa-lock"></i>
          <input type="password" placeholder="Enter your password" required name="password">
        </div>
        <div class="text"><a href="#onclick">Forgot password?</a></div>
        <div class="button input-box">
          <input type="submit" value="Sumbit">
        </div>
        <div class="text sign-up-text">Don't have an account? <label for="flip">Sigup
now</label></div>

    {% if error %}
    <p><strong style="color:red">Error</strong>: {{error}}</p>
  {% endif %}
  {% with messages = get_flashed_messages() %}
  {% if messages %}
```

```
      {% for message in messages %}
          <center> <p style="color:green">{{ message }}</p>  </center>
      {% endfor %}
  {% endif %}
  {% endwith %}
  {% if msgpass %}
        <center> <p style="color:green">{{msgpass}}</p>  </center>
      {% endif %}
      </div>
  </form>
</div>
    </div>
  </div>
</div>
<div id="onclick" class="overlay">
  <div class="popup">
      <h1 Style="color:#4070f4; font-family:Cambria, Cochin, Georgia, Times, 'Times New Roman', serif">Change Password</h1>
      <a class="close" href="#">&times;</a>
      <div class="content">
          <!--Supplier Count:{{tot_sup}}
          <br>
          {% if tot_sup==0 %}
              <br><h3>You Don't Have Supplier! Add supplier!</h3>
          {% else %}
              <br><h3>Add Product</h3>
          {% endif %}-->
      </div>
      <form action="updatepassword" method="POST">
      <div class="input-field">
        <div class="input-field">
          <label>Enter email:</label>
          <input type="text" placeholder="Enter ID" name="email" required>
      </div>
        <label>Old Password:</label>
        <input type="password" placeholder="Enter OTP" name="oldpass" required>
      </div>
      <div class="input-field">
      <label>New Password</label>
      <input type="password" placeholder="Enter New Password" name="newpass" required>
      </div>

      <div class="buttons">
          <!--<a class="button"href="#">Button</a>-->
          <button class="sumbit">
              <span class="btnText">Change</span>
              <i class="uil uil-navigator"></i>
          </button>
      </div>
    </form>
  </div>
</div>

</body>
</html>
```

app.py

```
@app.route("/register",methods=['GET','POST'])
```

```
def register():
    error = None
    if request.method=='POST':
        registermsg=""
        registeraccount=""
        company=request.form['company'].title()
        location=request.form['location'].title()
        email=request.form['email']
        username=request.form['username']
        password=request.form['password']
        mail_check=checkmail(email)
        pass_check=checkpassword(password)
        if mail_check=="Invalid Email":
            mesaagemail="Email ID Not VAlid"
            return render_template('Reg.html',mesaagemail=mesaagemail)
        elif pass_check!="Valid Password":
            mesaagemail=pass_check
            return render_template('Reg.html',mesaagemail=mesaagemail)
        else:
            sql="SELECT * FROM REGISTER WHERE MAIL_ID=?"
            prep_stmt=ibm_db.prepare(conn,sql)
            ibm_db.bind_param(prep_stmt,1,email)
            ibm_db.execute(prep_stmt)
            account=ibm_db.fetch_assoc(prep_stmt)
            print(account)
            print(company)
            if account:
                registeraccount="Account already exists! Log in to continue !"
            else:
                insert_sql="INSERT INTO REGISTER
(COMPANY,LOCATION,MAIL_ID,USER_ID,PASSWORD)values(?,?,?,?,?)"
                prep_stmt=ibm_db.prepare(conn,insert_sql)
                ibm_db.bind_param(prep_stmt,1,company)
                ibm_db.bind_param(prep_stmt,2,location)
                ibm_db.bind_param(prep_stmt,3,email)
                ibm_db.bind_param(prep_stmt,4,username)
                ibm_db.bind_param(prep_stmt,5,password)
                ibm_db.execute(prep_stmt)
                print("inserted")
                subject="Registration successfull"
                html_content="Manage Your Stock Efficiently"
                sendmail(API,from_email,email,subject,html_content)
                registermsg="Registration successfull. Log in to continue !"
    else:
        print("not post")
        pass
    return
render_template('Reg.html',error=error,registermsg=registermsg,registeraccount=registeraccoun
t)

@app.route('/login',methods=['GET','POST'])
def login():
    error = None
    if request.method=='POST':
        username=request.form['username']
        password=request.form['password']
        sql="SELECT * FROM REGISTER WHERE MAIL_ID=? 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['logged_in']=True
            session['id']=account['INVENTORY_ID']
            session["username"]=account["USER_ID"]
            session['company_name']=account['COMPANY']
            session['company_mail']=account["MAIL_ID"]
            flash("Logged in successfully!")
            return redirect(url_for("DashBoard"))
        else:
            error="Incorrect username / password"
            return render_template('Reg.html',error=error)
    else:
        pass
    return render_template('Reg.html',error=error)
```

7.2 FEATURE 2

DASHBOARD:

In dashboard user can view their products, suppliers and the minimum stock available and can have report through email if needed.

Layout.html

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Inventory</title>
        <title>{% block title%}{% endblock %}</title>
   <!--bootstrap--><link href='https://unpkg.com/boxicons@2.0.9/css/boxicons.min.css'
rel='stylesheet'>
        <!-- My CSS -->
        <link rel="stylesheet" href="{{url_for('static',filename='css/MainPage.css')}}">
        {% block style %}{% endblock %}
        <script>
                window.watsonAssistantChatOptions = {
                  integrationID: "e9f04c7c-d8db-48f7-9947-f80357d44388", // The ID of this
integration.
                  region: "us-south", // The region your integration is hosted in.
                  serviceInstanceID: "2708cb0d-9713-42b3-b1ac-42da255363a8", // 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>
        <!-- SIDEBAR -->
        <section id="sidebar">
```

```
		<a href="#" class="brand">
			<i class='bx bxs-smile'></i>
			<span class="text">IStoCk</span>
		</a>
		<ul class="side-menu top">
			<li>
			<!--<li class="active">-->
				<a href="/DashBoard">
					<i class='bx bxs-dashboard' ></i>
					<span class="text">Dashboard</span>
				</a>
			</li>
			<li>
				<a href="/products">
					<i class='bx bxs-shopping-bag-alt' ></i>
					<span class="text">My Store</span>
				</a>
			</li>
	<li>
				<a href="/Supplier">
					<i class='bx bxs-group' ></i>
					<span class="text">Suppliers</span>
				</a>
			</li>
			<li>
				<a  href="/Reorder">
					<i class='bx bxs-message-dots' ></i>
					<span class="text">Order</span>
				</a>
```

```
            </li>
        </ul>
        <ul class="side-menu">
            <li>
                <a href="/Operation">
                    <i class='bx bxs-cog' ></i>
                    <span class="text">Product/Supplier</span>
                </a>
            </li>
            <li>
                <a href="logout" class="logout">
                    <i class='bx bxs-log-out-circle' ></i>
                    <span class="text">Logout</span>
                </a>
            </li>
        </ul>
    </section>
    <!-- SIDEBAR -->
    <!-- CONTENT -->
    <section id="content">
        <!-- NAVBAR -->
        <nav>
            <i class='bx bx-menu' ></i>
            <a href="#" class="nav-link">Categories</a>
            <form action="search_nav" method="GET">
                <div class="form-input">
                    <input type="search" placeholder="Search..."
name="search_nav">
                    <button type="submit" class="search-btn"><i class='bx bx-
```

```
search' ></i></button>
                </div>
            </form>
            <input type="checkbox" id="switch-mode" hidden>
            <label for="switch-mode" class="switch-mode"></label>
            <a href="#" class="notifications">
                <i class='bx bxs-bell' ></i>
                <span class="num">8</span>
            </a>
            <a href="#onclick" class="profile">
                <img src="img/account.jpg">
            </a>
        </nav>
        <!-- NAVBAR -->
        <div id="onclick" class="overlay">
            <div class="popup">
                <h1 Style="color:#4070f4; font-family:Cambria, Cochin, Georgia,
Times, 'Times New Roman', serif">Change Password</h1>
                <a class="close" href="#">&times;</a>
                <div class="content">
                    <!--Supplier Count:{{tot_sup}}
                    <br>
                    {% if tot_sup==0 %}
                        <br><h3>You Don't Have Supplier! Add
supplier!</h3>
                    {% else %}
                        <br><h3>Add Product</h3>
                    {% endif %}-->
                </div>
```

```
                                        <form action="updatepassword" method="POST">
                                        <div class="input-field">
                                          <div class="input-field">
                                                <label>Enter email:</label>
                                                <input type="text" placeholder="Enter ID" name="email"
required>
                                        </div>
                                          <label>Old Password:</label>
                                          <input type="password" placeholder="Enter OTP"
name="oldpass" required>
                                        </div>
                                        <div class="input-field">
                                        <label>New Password</label>
                                        <input type="password" placeholder="Enter New Password"
name="newpass" required>
                                        </div>

                                        <div class="buttons">
                                                <!--<a class="button"href="#">Button</a>-->
                                                <button class="sumbit">
                                                        <span class="btnText">Change</span>
                                                        <i class="uil uil-navigator"></i>
                                                </button>
                                        </div>
                                   </form>
                                  </div>
                            </div>
        {% block content %}{% endblock %}
        </section>
```

```
            <script src="{{url_for('static',filename='Js/MainPage.js')}}"></script>
</body>
</html>
```

DashBoard.html

```
{% extends "MainPage.html" %}
{% block title%}DashBoard{% endblock %}
{% block style %}
<link rel="stylesheet" href="{{url_for('static',filename='css/DashBoard.css')}}">
{% endblock %}
{% block content %}
    <!-- MAIN -->
            <!--<div id='template' class="template">-->
            <main>
                <div class="head-title">
                    <div class="left">
                        <h1>Dashboard</h1>
                        <ul class="breadcrumb">
                            <li>
                                <a href="#">Dashboard</a>
                            </li>
                            <li><i class='bx bx-chevron-right' ></i></li>
                            <li>
                                <a class="active" href="#">Welcome {{username}},</a>
                            </li>
                        </ul>
                    </div>
                    <a href="/sendreport" class="btn-download">
                        <i class='bx bxs-cloud-download' ></i>
```

```
        <span class="text">Send Report To Mail</span>
    </a>
</div>

<ul class="box-info">
    <li>
        <i class='bx bxs-calendar-check' ></i>
        <span class="text">
            <h3>{{products}}</h3>
            <p>Total Products</p>
        </span>
    </li>
    <li>
        <i class='bx bxs-group' ></i>
        <span class="text">
            <h3>{{supplier}}</h3>
            <p>Suppliers</p>
        </span>
    </li>
    <li>
        <i class='bx bx-trending-down' ></i>
        <span class="text">
            <h3>{{min}}</h3>
            <p>Minimum Stock</p>
        </span>
    </li>
</ul>
```

```
<div class="table-data">
    <div class="order">
        <div class="head">
            <h3>Product Group</h3>
            <i class='bx bx-search' ></i>
            <i class='bx bx-filter' ></i>
        </div>
        <table>
            <thead>
                <tr>
                    <th>Product Group</th>
                    <th>Product Count</th>
                    <th>Supplier</th>
                </tr>
            </thead>
            <tbody>
                {% for inlist in listpy %}
                <tr>
                    <td>{{inlist.PRODUCT_GRP}}</td>
                    <td>{{inlist.COUNT}}</td>
                    <td>{{inlist.SUPPLIER}}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>
    <div class="todo">
        <div class="head">
            <h3>Vendors</h3>
```

```
                            <i class='bx bx-plus' ></i>
                            <i class='bx bx-filter' ></i>
                        </div>
                        <ul class="todo-list">
                            {% for inlist in listsup %}
                            <li class="completed">
                                <p>{{inlist.SUPPLIER_NAME}} - {{inlist.SUP_DASH}}</p>
                                <i class='bx bx-dots-vertical-rounded' ></i>
                                <!--<i class='bx bxs-checkbox-checked' ></i>-->
                            </li>
                            {% endfor %}
                        </ul>
                    </div>
                </div>
            </main>
            <!-- MAIN -->
        <!--</div>-->
{% endblock %}
```

AddProducts.html:

```
<!DOCTYPE html>

<!--=== Coding by CodingLab | www.codinglabweb.com === -->

<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">
<title>Products</title>
<!----======== CSS ======== -->
<link rel="stylesheet" href="{{url_for('static',filename='css/AddProducts.css')}}">
<!----===== Iconscout CSS ===== -->
<link rel="stylesheet" href="https://unicons.iconscout.com/release/v4.0.0/css/line.css">

<!--<title>Responsive Regisration Form </title>-->
</head>
<body>
<div class="container">
<header>Add Products</header>

<form method="POST">
<div class="form first">
<div class="details personal">
<span class="title">Product Details</span>
```

```
<div class="fields">

<div class="input-field">

<label>Product ID</label>

<input type="text" placeholder="Enter ID" name="pro_id" required>

</div>



<div class="input-field">

<label>Product Group</label>

<select name="pro_grp" required>

<option disabled selected>Select Product Group</option>

{% for drop in drop_pro %}

<option value="{{drop}}">{{drop}}</option>

{% endfor %}

</select>

</div>

<div class="input-field">

<label>Product Name</label>

<input type="text" placeholder="Enter ID" name="pro_name" required>
```

```
</div>

<div class="input-field">
<label>Total Quantity</label>
<input type="number" placeholder="Enter Total Quantity" name="tot_quantity" required>
</div>

<div class="input-field">
<label>Available</label>
<input type="number" placeholder="Enter Avaliable" name="available" value="val" required>
</div>
<div class="input-field">
<label>Shipped</label>
<input type="number" placeholder="Enter Shipped Quantity" name="Shipped" required>
</div>
<div class="input-field">
<label>Minimum Stock for this Product</label>
<input type="number" placeholder="Enter Minimum Stock" name="min_stock" required>
```

```
</div>

<div class="input-field">
<label>Supplier</label>
<select name="sup" required>
<option disabled selected>Select Supplier</option>
{% for drop in drop_sup %}
<option value="{{drop}}">{{drop}}</option>
{% endfor %}
</select>
</div>
<div class="input-field">
<label>Date</label>
<input type="date" placeholder="Enter date" name="date" required >
</div>
</div>
</div>
</form>
```

```
<div class="details personal">
{% with messages = get_flashed_messages() %}
{% if messages %}
{% for message in messages %}
<!--<center> <p style="color:green"></p> </center>-->
<span class="title">{{ message }}</span>
{% endfor %}
{% endif %}
{% endwith %}
<div class="buttons">
<button class="submit">
<span class="btnText"><a href="backProSup">Back</a></span>
<i class="uil uil-navigator"></i>
</button>
<button class="sumbit" formaction="addPro">
<span class="btnText">Add</span>
<i class="uil uil-navigator"></i>
</button>
```

```
<button class="sumbit" formaction="updatePro">
<span class="btnText">Update</span>
<i class="uil uil-navigator"></i>
</button>

<button class="sumbit" formaction="deletePro">
<span class="btnText">Delete</span>
<i class="uil uil-navigator"></i>
</button>
</div>
</div>
</div>
</div>
</div>
<script src="{{url_for('static',filename='Js/AddProducts.js')}}"></script>
</body>
</html>
```

Reorder.html

```
{% extends "MainPage.html" %}
{% block title%}Order Products{% endblock %}
{% block style %}
<link rel="stylesheet" href="{{url_for('static',filename='css/Reorder.css')}}">
{% endblock %}
{% block content %}
<div class="wrapper">
  <div class="title">
    <h1>Order Products</h1>
  </div>
  <form action="/sendordermail" method="POST">
  <div class="contact-form">
    <div class="input-fields">
      <input type="text" class="input" placeholder="To" name="tomail">
      <input type="text" class="input" placeholder="Subject" name="subject">
      <!--<input type="text" class="input" placeholder="Phone">--
      <input type="text" class="input" placeholder="Subject">-->
    </div>
    <div class="msg">
      <textarea placeholder="Message" name="text"></textarea>
      {% if ordermail %}
      <div class="title">
        <h3>{{ordermail}}</h3>
      </div>
      {% endif %}
      <div class="buttons">
      <button class="sumbit">
        <span class="btnText">Send</span>
        <i class="uil uil-navigator"></i>
      </button>
      </div>
    </div>
  </div>
</form>
</div>
{% endblock %}
```

app.py

```
def count():
    sql="SELECT count(PRODUCT_ID) as PRO FROM PRODUCT WHERE INVENTORY_ID = ?"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    account=ibm_db.fetch_assoc(stmt)
    print(account)
    session['total_pro']=account['PRO']
    sql="SELECT count(SUPPLIER_ID) as SUP FROM SUPPLIER WHERE INVENTORY_ID = ?"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    account=ibm_db.fetch_assoc(stmt)
    print(account)
    session['total_sup']=account['SUP']

    sql="SELECT count(PRODUCT_ID) as MIN FROM PRODUCT WHERE INVENTORY_ID = ? AND AVAILABLE_PRODUCT<MIN_STOCK"
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    account=ibm_db.fetch_assoc(stmt)
    print(account)
    session['total_min']=account['MIN']

def procount():
    list=[]
    sql="SELECT PRODUCT_GRP,SUPPLIER,COUNT(PRODUCT_ID) AS COUNT FROM
PRODUCT WHERE INVENTORY_ID = ? GROUP BY PRODUCT_GRP,SUPPLIER "
    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    tot_count=ibm_db.fetch_assoc(stmt)
    while tot_count != False:
        list.append(tot_count)
        tot_count = ibm_db.fetch_assoc(stmt)
    return  list
def supcount():
    list_sup=[]
    sql_sup="SELECT SUPPLIER_NAME,COUNT(PRODUCT_GRP) AS SUP_DASH FROM
SUPPLIER WHERE INVENTORY_ID = ? GROUP BY SUPPLIER_NAME"
    stmt=ibm_db.prepare(conn,sql_sup)
    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)
    sup=ibm_db.fetch_assoc(stmt)
    while sup != False:
        list_sup.append(sup)
        sup = ibm_db.fetch_assoc(stmt)
    return list_sup
@app.route('/DashBoard')
@is_logged_in
def DashBoard():
    count()
    list=procount()
    list_sup=supcount()
    return
```

```
render_template('Dashboard.html',username=session["username"],products=session['total_pro'],s
upplier=session['total_sup'],min=session['total_min'],listpy=list,listsup=list_sup)
@app.route('/Reorder')
def Reorder():
    return render_template('Reorder.html')
@app.route("/sendordermail",methods=['GET','POST'])
def sendordermail():
    if request.method=='POST':
        print(session['company_mail'])
        #from_email=session['company_mail']
        to_email=request.form['tomail']
        subject=request.form['subject']
        html_content=request.form['text']
        print(html_content)
        sendmail(API,from_email,to_email,subject,html_content)
        print("Mail sent from sendordermail")
        return redirect(url_for("Reorder"))
    else:
        print("not post from sendordermail")
        return redirect(url_for("Reorder"))
```

DATABASE SCHEMA `

```
CREATE TABLE Register
(
 Inventory_ID int NOT NULL PRIMARY KEY GENERATED ALWAYS AS
IDENTITY(START WITH 1, INCREMENT BY 1) ,
 Company varchar(255),
 Location varchar(255),
 Mail_ID varchar(80),
 User_ID varchar(70),
 Password varchar(30)
);


CREATE TABLE PRODUCT
(
 Inventory_ID int,Product_PK int NOT NULL PRIMARY KEY GENERATED ALWAYS AS
IDENTITY(START WITH 1, INCREMENT BY 1),
 Product_ID varchar(30),
 Product_GRP varchar(50),
 Product_Name varchar(25),
 TOT_Quantity int,
 Available_product int,
 Shipped int,
 Min_stock int,
 Supplier varchar(25),
 Date date,
 FOREIGN KEY Pro_FK (Inventory_ID) REFERENCES Register ON DELETE NO ACTION
)
```

```
CREATE TABLE SUPPLIER
(
Inventory_ID int,Supplier_PK int NOT NULL PRIMARY KEY GENERATED ALWAYS AS
IDENTITY(START WITH 1, INCREMENT BY 1),
Supplier_ID varchar(100),
Supplier_Name varchar(50),
Location varchar(25),
PH_Number varchar(10),
Product_GRP varchar(50),
Product_Name varchar(25),
SupMail_ID varchar(60),
FOREIGN KEY Sup_FK (Inventory_ID) REFERENCES Register ON DELETE NO ACTION
)
```

-

-

-

-

-

-

8.TESTING

8.1 TEST CASES

Test case ID	Component	Test Scenario	Expected Result
Inventory_TC_OO1	Home Page	Verify if the user is able to see the Register and log in button.	Login/Signup button should display
Inventory_TC_OO2	Home Page	Verify if the UI elements in Login/Register button works.	Login and Register page is viewed.
Inventory_TC_OO3	Register page	Verify if the email is a valid email or not	Application should show 'Incorrect email ' validation message
Inventory_TC_OO4	Register page	Verify if the password contains a lower case, upper case and special charecter and the length of the passowrd is between 6-12	Application shows whether the password is Strong.
Inventory_TC_OO4	Register page	Verify if the account is already exist.	Application should show 'Account Already exist!Login to continue ' validation message
Inventory_TC_OO6	Login Page	Verify if the user is able to log into application with Valid credentials	User should navigate to Dashboard page
Inventory_TC_OO7	Login page	Verify if the user is able to log into application with InValid credentials	Application should show 'Incorrect email or password ' validation message.

Inventory_TC_OO8	Change Password	Verify if the user can change password	Application allows the user to change password
Inventory_TC_OO9	Dashboard	Verify if the dashboard works properly	Application shows a set of tabs to go into.
Inventory_TC_O10	Dashboard	Verify if the data in dashboard is correct	Application shows tables of the data entered.
Inventory_TC_O11	My Store	Verify if the data in the table is corrext	Application shows the correct data entered.
Inventory_TC_O12	My Store	Verify if the user can edit or delete the information	Application allows the user to edit or delete the data properly.
Inventory_TC_O13	Supplier	Verify if the user can view the supplier table properly.	Application shows the correct data entered in the supplier tab.
Inventory_TC_O14	Supplier	Verify if the user can edit or delete the information in the table.	Application allows the user to edit or delete the data properly.
Inventory_TC_O15	Order	Verify if user is able to order for more stock through email	Application shows the order has been placed.
Inventory_TC_O16	Product/Supplier tab	Verify if user is able to add a product or supplier.	Applications shows a product or supplier has been added.

8.2 USER ACCEPTANCE TESTING

1. Purpose of Document

The purpose of this document is to manage the stock of various businesses appropriately. Users should be able to add, delete and edit their stock and suppliers and reorder if and when needed.

2. Defect Analysis

This reportshows the numberof resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	1	4	2	3	10
Duplicate	1	0	1	0	2
External	2	0	0	1	3
Fixed	8	2	4	6	20
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	16	6	7	14	43

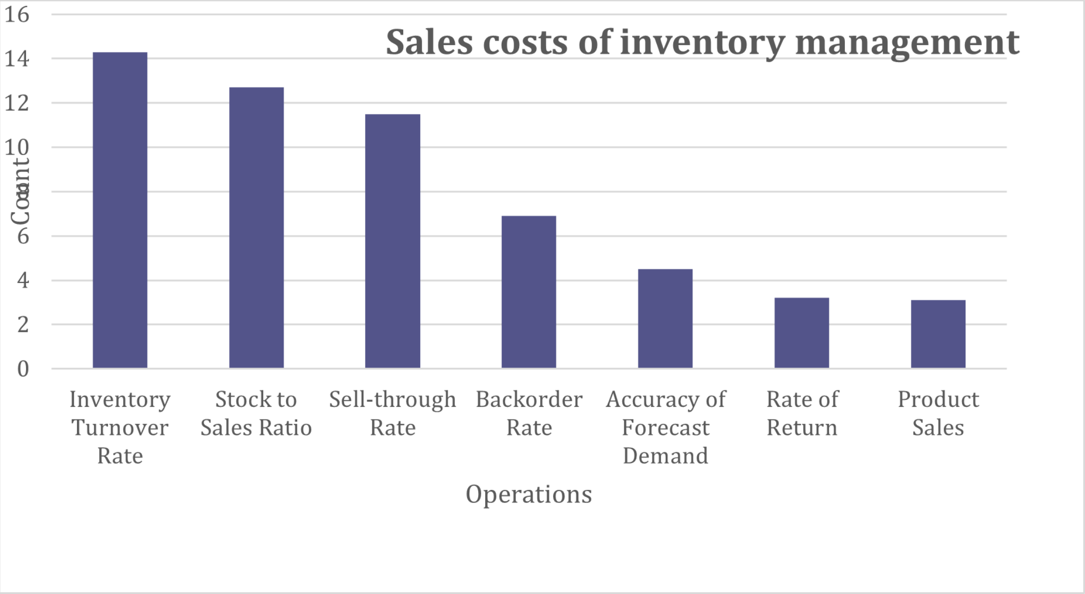
3.Test Case Analysis

This reportshows the numberof test cases that have passed, failed,and untested

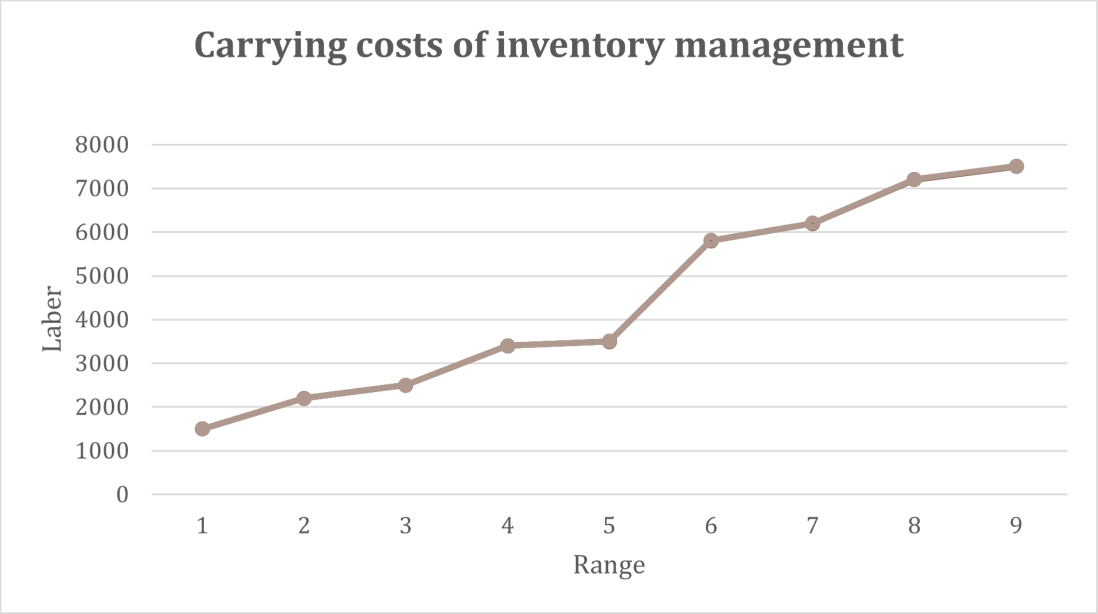
Section	**Total Cases**	**Not Tested**	**Fail**	**Pass**
Login	5	0	0	5
Register	19	0	0	19
Dashboard	5	0	0	5
Log Out	6	0	0	6

9. RESULTS

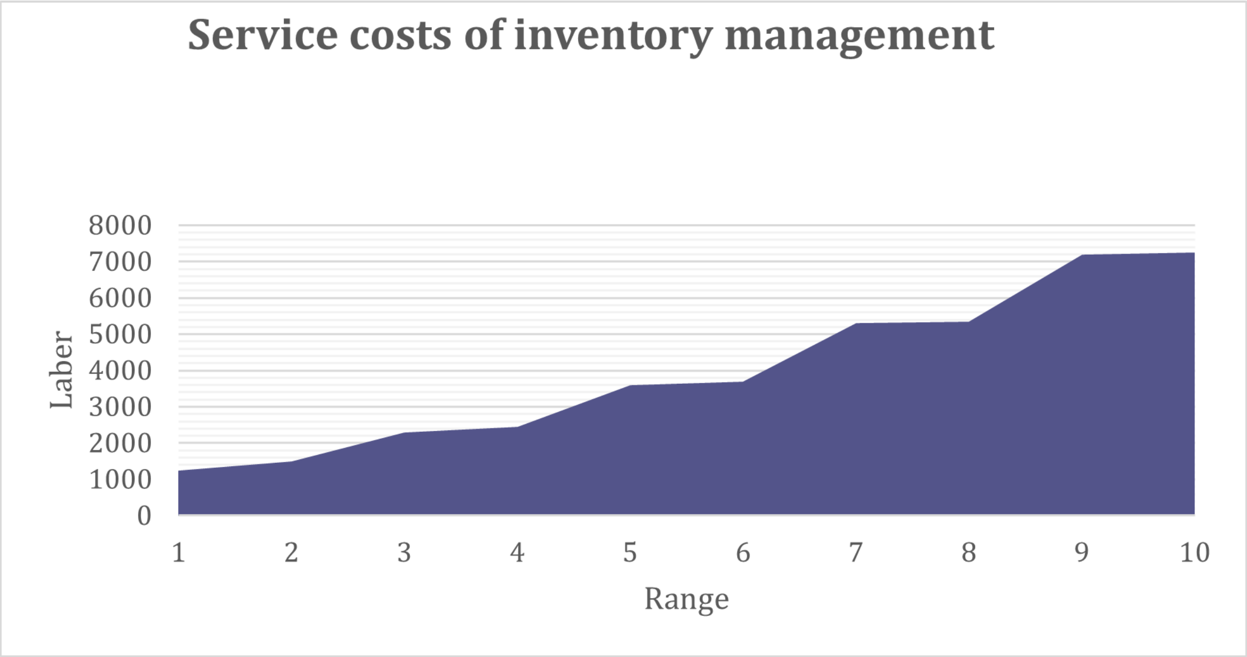
9.1 PERFORMANCE METRICS



-



-



Service costs of inventory management

10. Advantages and Disadvantages

Advantages

- **Understand Inventory Levels Across the Business:** ERP systems can provide an end-to-end view into orders through all departments, from sales to accounting to fulfillment. Centralized purchasing reduces duplication when replenishing stock, and having the ability to purchase in bulk saves money.
- **Automate Manual Tasks:** Barcode and RFID scanning can speed stock-taking, receiving and fulfillment. Using software reduces errors from manual entries and frees staff from repetitive tasks.
- **Greater Visibility with Real-Time Data:** The right inventory management software will give you access to real-time information on all SKUs, in all facilities. It will deliver this data to all devices, no matter where you are.
- **Improve Forecasting:** Software that handles data collection and analytics can provide insights into trends. And when you understand trends, you can improve your stock forecasting.

Disadvantages

- **Expensive for Small Businesses:** The cost of inventory management software can seem daunting to a small business, but the investment often pays for itself in increased profits and improved customer loyalty. Additionally, cloud-based systems have made software that was once the domain of large enterprises available to smaller businesses.
- **Complex to Learn:** Business software is sometimes tricky to learn. However, managers can help by investing in online training to quickly bring users up to speed.
- **Malicious Hacks:** Malicious hacks are a risk to all businesses. The IoT adds even more complexity. Cloud-based software typically has greater security than a single company would offer on its own because of the risk a breach would have on the vendor.

11. CONCLUSION

-

Inventory management is a useful method for simplifying all the warehousing activities of the organization. With this technique, the company can now access and determine its stock and inventory with efficiency to smoothen all the business operations. It has also proved to be a valuable tool for maintaining the working capital requirement.

The barcode system is its automated and simplified version. The management can find out the stock remaining with just one click on a computer device. The scanned barcodes enable the software to maintain a track of all the purchases and the flow of inventory.

An inventory control system is a technology solution that manages and tracks a company's goods through the supply chain. This technology will integrate and manage purchasing, shipping, receiving, warehousing, and returns into a single system.

The best inventory control system will automate a lot of manual processes. It will provide an accurate picture of what inventory you have, where it is, and when you need to reorder to keep your stock at optimal levels.

Inventory is the lifeblood of any ecommerce store. Having the right products in stock will prevent backorders, keeps customers happy, and make your business profitable. A robust inventory control system will do much of this work for you by automating manual inventory control processes, streamlining your logistic workflow, and giving you a real-time view of inventory levels.

12. Future Scope

Safety stock is defined as the difference between the amount stocked to sati.sfy demand during a certain time interval and the mean expected demand for that period. It is for the purpose of providing protection against depletion. If demand remained constant and lead tin-; is invariable, there would be no fear of shortages and no need for safely stocks.

The exact quantity of safety stock of an item depends upon its lead time, usage value, and variability of lead time demand, carrying charges and the importance of its stock out cost. Again, determination of buffer stock reserve stock is included in the management of inventory.

Safety stock is defined as the difference between the amount stocked to sati.sfy demand during a certain time interval and the mean expected demand for that period. It is for the purpose of providing protection against depletion. If demand remained constant and lead tin-; is invariable, there would be no fear of shortages and no need for safely stocks.

The exact quantity of safety stock of an item depends upon its lead time, usage value, and variability of lead time demand, carrying charges and the importance of its stock out cost. Again, determination of buffer stock reserve stock is included in the management of inventory.

The policies of investment procurement, storage, handling, accounting, storages and stock outs, deterioration, obsolescence etc. are to be formulated under the scientific system of inventory control. What, when and how much of purchasing and fixation of minimum and maximum levels is also to be determined for a given period of time.

—

—

—

-

-

-

13.APPENDIX

SOURCE CODE

Register.html

```
<!DOCTYPE html>
<!-- Coding by CodingLab | www.codinglabweb.com-->
<html lang="en" dir="ltr">
  <head>
    <meta charset="UTF-8">
    <!--<title> Login and Registration Form in HTML & CSS | CodingLab </title>-->
    <link rel="stylesheet" href="{{url_for('static',filename='css/Reg.css')}}">
    <!-- Fontawesome CDN Link -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">
      <script src="https://code.jquery.com/jquery-3.4.1.js"></script>
   <title>Inventory Managemanet System For Retailers</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
<body>
  <div class="container">
```

```html
<input type="checkbox" id="flip">
<div class="cover">
  <div class="front">
    <img src="C:\Users\deeks\OneDrive\Desktop\frontImg.jpg" alt="">
    <div class="text">
      <span class="text-1">Every new friend is a <br> new adventure</span>
      <span class="text-2">Let's get connected</span>
    </div>
  </div>
  <div class="back">
    <img class="backImg" src="C:\Users\deeks\OneDrive\Desktop\Img.jpg" alt="">
    <div class="text">
      <span class="text-1">Complete miles of journey <br> with one step</span>
      <span class="text-2">Let's get started</span>
    </div>
  </div>
</div>
<div class="forms">
  <div class="form-content">
```

```
<div class="signup-form">
  <div class="title">Signup</div>
<form action="register" method="post">
    <div class="input-boxes">
      <div class="input-box">
        <i class="fas fa-building"></i>
        <input type="text" placeholder="Enter your company name" name="company" required>
      </div>
      <div class="input-box">
        <i class="fas fa-map-marker "></i>
        <input type="text" placeholder="Enter your location" required name="location">
      </div>
      <div class="input-box">
        <i class="fas fa-envelope"></i>
        <input type="text" placeholder="Enter your email" required name="email">
      </div>
      <div class="input-box">
        <i class="fas fa-user"></i>

        <input type="text" placeholder="Enter User ID" required name="username">

      </div>

      <div class="input-box">

        <i class="fas fa-lock"></i>

        <input type="password" placeholder="Enter your password" required
  name="password"> </div>

      {% if mesaagemail %}

          <center> <p style="color:red">{{mesaagemail}}</p></center>

      {% endif %}

      {% if registeraccount %}

          <center> <p style="color:red">{{registeraccount}}</p></center>

      {% endif %}

      {% if registermsg %}

          <center> <p style="color:green">{{registermsg}}</p></center>

      {% endif %}

      <div class="button input-box">

        <input type="submit" value="Sumbit">

      </div>
```

```
              <div class="text sign-up-text">Already have an account? <label for="flip">Login now</label></div>

            </div>

      </form>

     </div>

    <div class="login-form">

    <div class="title">Login</div>

    <form action="login" method="post">

      <div class="input-boxes">

        <div class="input-box">

          <i class="fas fa-envelope"></i>

          <input type="text" placeholder="Enter your email" required name="username">

        </div>

        <div class="input-box">

          <i class="fas fa-lock"></i>

          <input type="password" placeholder="Enter your password" required name="password">

        </div>

        <div class="text"><a href="#onclick">Forgot password?</a></div>

        <div class="button input-box">
```

```
          <input type="submit" value="Sumbit"></div>

        <div class="text sign-up-text">Don't have an account? <label for="flip">Sigup now</label></div>

    {% if error %}

    <p><strong style="color:red">Error</strong>: {{error}}</p>

  {% endif %}

  {% with messages = get_flashed_messages() %}

  {% if messages %}

      {% for message in messages %}

          <center> <p style="color:green">{{ message }}</p>  </center>

      {% endfor %}

  {% endif %}

  {% endwith %}

  {% if msgpass %}

        <center> <p style="color:green">{{msgpass}}</p>  </center>

      {% endif %}

      </div>

  </form>

</div>
```

```
    </div>
  </div>
</div>
<div id="onclick" class="overlay">
  <div class="popup">
      <h1 Style="color:#4070f4; font-family:Cambria, Cochin, Georgia, Times, 'Times New Roman', serif">Change Password</h1>
      <a class="close" href="#">&times;</a>
      <div class="content">
          <!--Supplier Count:{{tot_sup}}
          <br>
          {% if tot_sup==0 %}
              <br><h3>You Don't Have Supplier! Add supplier!</h3>
          {% else %}
              <br><h3>Add Product</h3>
          {% endif %}-->
      </div>
      <form action="updatepassword" method="POST">
      <div class="input-field">
        <div class="input-field">
          <label>Enter email:</label>
          <input type="text" placeholder="Enter ID" name="email" required>
      </div>
        <label>Old Password:</label>
        <input type="password" placeholder="Enter OTP" name="oldpass" required>
      </div>
      <div class="input-field">
      <label>New Password</label>
      <input type="password" placeholder="Enter New Password" name="newpass" required>
      </div>
      <div class="buttons">
          <!--<a class="button"href="#">Button</a>-->
          <button class="sumbit">
              <span class="btnText">Change</span>
              <i class="uil uil-navigator"></i>
          </button>
      </div>
    </form>
```

```
  </div>

</div>

</body>

</html>
```

MainPage.html

```
<!DOCTYPE html>

<html lang="en">

<head>

        <meta charset="UTF-8">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>Inventory</title>

        <title>{% block title%}{% endblock %}</title>

   <!--bootstrap--><link href='https://unpkg.com/boxicons@2.0.9/css/boxicons.min.css' rel='stylesheet'>

     <link rel="stylesheet" href="{{url_for('static',filename='css/MainPage.css')}}">

        {% block style %}{% endblock %}

        <script>

                window.watsonAssistantChatOptions = {

                  integrationID: "e9f04c7c-d8db-48f7-9947-f80357d44388", // The ID of this
integration.
         region: "us-south", // The region your integration is hosted in.
         serviceInstanceID: "2708cb0d-9713-42b3-b1ac-42da255363a8", // 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>
    <!-- SIDEBAR -->
    <section id="sidebar">
        <a href="#" class="brand">
            <i class='bx bxs-smile'></i>
            <span class="text">IStoCk</span>
        </a>
        <ul class="side-menu top">
            <li>
            <!--<li class="active">-->
                <a href="/DashBoard">
                    <i class='bx bxs-dashboard' ></i>
                    <span class="text">Dashboard</span>
                </a>
            </li>
            <li>
                <a href="/products">
                    <i class='bx bxs-shopping-bag-alt' ></i>
                    <span class="text">My Store</span>
                </a>
            </li>
    <li>
                <a href="/Supplier">
                    <i class='bx bxs-group' ></i>
                    <span class="text">Suppliers</span>
                </a>
            </li>
            <li>
                <a  href="/Reorder">
                    <i class='bx bxs-message-dots' ></i>
                    <span class="text">Order</span>
                </a>
            </li>
        </ul>
        <ul class="side-menu">
            <li>
                <a href="/Operation">
                    <i class='bx bxs-cog' ></i>
                    <span class="text">Product/Supplier</span>
                </a>
            </li>
            <li>
                <a href="logout" class="logout">
```

```
                    <i class='bx bxs-log-out-circle' ></i>
                    <span class="text">Logout</span>
                </a>
            </li>
        </ul>
    </section>
    <!-- SIDEBAR -->
    <!-- CONTENT -->
    <section id="content">
        <!-- NAVBAR -->
        <nav>
            <i class='bx bx-menu' ></i>
            <a href="#" class="nav-link">Categories</a>
            <form action="search_nav" method="GET">
                <div class="form-input">
                    <input type="search" placeholder="Search..." name="search_nav">
                    <button type="submit" class="search-btn"><i class='bx bx-search' ></i></button>
                </div>
            </form>
            <input type="checkbox" id="switch-mode" hidden>
            <label for="switch-mode" class="switch-mode"></label>
            <a href="#" class="notifications">
                <i class='bx bxs-bell' ></i>
                <span class="num">8</span>
            </a>
            <a href="#onclick" class="profile">
                <img src="img/account.jpg">
            </a>
        </nav>
        <!-- NAVBAR -->
        <div id="onclick" class="overlay">
            <div class="popup">
                <h1 Style="color:#4070f4; font-family:Cambria, Cochin, Georgia, Times, 'Times New Roman', serif">Change Password</h1>
                <a class="close" href="#">&times;</a>
                <div class="content">
            <!--Supplier Count:{{tot_sup}}
            <br>
            {% if tot_sup==0 %}
                <br><h3>You Don't Have Supplier! Add supplier!</h3>
            {% else %}
                <br><h3>Add Product</h3>
            {% endif %}-->
        </div>
        <form action="updatepassword" method="POST">
        <div class="input-field">
          <div class="input-field">
            <label>Enter email:</label>
            <input type="text" placeholder="Enter ID" name="email" required>
        </div>
          <label>Old Password:</label>
          <input type="password" placeholder="Enter OTP" name="oldpass" required>
        </div>
```

```
                <div class="input-field">
                <label>New Password</label>
                <input type="password" placeholder="Enter New Password"
name="newpass" required>
                </div>

                <div class="buttons">
                    <!--<a class="button"href="#">Button</a>-->
                    <button class="sumbit">
                        <span class="btnText">Change</span>
                        <i class="uil uil-navigator"></i>
                    </button>
                </div>
             </form>
            </div>
        </div>
    {% block content %}{% endblock %}
    </section>
    <script src="{{url_for('static',filename='Js/MainPage.js')}}"></script>
```

```
</body>
</html>
```

app.py

```
from  flask import Flask,render_template,url_for,request,flash,session,redirect
import ibm_db
import re
from functools import wraps
from tabulate import tabulate
import random
import configparser
import ssl
ssl._create_default_https_context=ssl._create_unverified_context
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
config=configparser.ConfigParser()
config.read("config.ini")
try:
    settings=config["SETTINGS"]
except:
    settings={}
app=Flask(__name__)
app.secret_key='ay'
try:
    conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=31321;SECURITY=SS
L;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=hkl72011;PWD=EWU0l3XLa6KfY4
Kf","","")
except:
    print("Unable to connect: ",ibm_db.conn_error())
API=settings.get("APIKEY",None)
from_email=settings.get("FROM",None)
def sendmail(API,from_email,to_emails,subject,html_content):
   #if API!=None and from_email!=None and len(to_emails)>0:
        message=Mail(from_email,to_emails,subject,html_content)
        print(message)
        try:
            sg = SendGridAPIClient(API)
            response = sg.send(message)
            print(response.status_code)
            print(response.body)
            print(response.headers)
        except Exception as e:
            print(e.message)
def alert(pro_id,pro_grp,pro_name,Tot_Q,Available,Shipped,Minimum_stock,Supplier):
    if Available<Minimum_stock:
        to_email=session['company_mail']
        subject="Your Product Stock Is Low"
        html_content='''Hi '''+session["username"]+''',
        Your product is less than minimum stock for the product '''+pro_name+'''
        Description:
        Product ID:'''+pro_id+'''
        Product Group:'''+pro_grp+'''
        Product Name:'''+pro_name+'''
        Total Quantity:'''+Tot_Q+'''
        Available:'''+Available+'''
        Shipped:'''+Shipped+'''
        Minimum_stock:'''+Minimum_stock+'''
        Supplier:'''+Supplier+'''
        Regards,

        IStock'''

        sendmail(API,from_email,to_email,subject,html_content)

        print("alert sent ")

        return "Mail Sent"

    else:

        return "Mail Not Sent"

def count():

    sql="SELECT count(PRODUCT_ID) as PRO FROM PRODUCT WHERE INVENTORY_ID = ?"

    stmt=ibm_db.prepare(conn,sql)

    ibm_db.bind_param(stmt,1,session['id'])

    ibm_db.execute(stmt)

    account=ibm_db.fetch_assoc(stmt)

    print(account)

    session['total_pro']=account['PRO']

    sql="SELECT count(SUPPLIER_ID) as SUP FROM SUPPLIER WHERE INVENTORY_ID = ?"

    stmt=ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,session['id'])

    ibm_db.execute(stmt)

    account=ibm_db.fetch_assoc(stmt)

    print(account)

    session['total_sup']=account['SUP']

    sql="SELECT count(PRODUCT_ID) as MIN FROM PRODUCT WHERE INVENTORY_ID = ? AND AVAILABLE_PRODUCT<MIN_STOCK"

    stmt=ibm_db.prepare(conn,sql)

    ibm_db.bind_param(stmt,1,session['id'])

    ibm_db.execute(stmt)

    account=ibm_db.fetch_assoc(stmt)

    print(account)

    session['total_min']=account['MIN']

def procount():

    list=[]

    sql="SELECT PRODUCT_GRP,SUPPLIER,COUNT(PRODUCT_ID) AS COUNT FROM PRODUCT WHERE INVENTORY_ID = ? GROUP BY PRODUCT_GRP,SUPPLIER "

    stmt=ibm_db.prepare(conn,sql)

    ibm_db.bind_param(stmt,1,session['id'])
    ibm_db.execute(stmt)

    tot_count=ibm_db.fetch_assoc(stmt)

    while tot_count != False:

        list.append(tot_count)

        tot_count = ibm_db.fetch_assoc(stmt)

    return  list

def supcount():

    list_sup=[]

    sql_sup="SELECT SUPPLIER_NAME,COUNT(PRODUCT_GRP) AS SUP_DASH FROM
SUPPLIER WHERE INVENTORY_ID = ? GROUP BY SUPPLIER_NAME"

    stmt=ibm_db.prepare(conn,sql_sup)

    ibm_db.bind_param(stmt,1,session['id'])

    ibm_db.execute(stmt)

    sup=ibm_db.fetch_assoc(stmt)

    while sup != False:

        list_sup.append(sup)

        sup = ibm_db.fetch_assoc(stmt)

    return list_sup

@app.route('/')
```

```
def index():
    return render_template('HomePage.html')
@app.route("/register",methods=['GET','POST'])
def register():
    error = None
    if request.method=='POST':
        registermsg=""
        registeraccount=""
        company=request.form['company'].title()
        location=request.form['location'].title()
        email=request.form['email']
        username=request.form['username']
        password=request.form['password']
        mail_check=checkmail(email)
        pass_check=checkpassword(password)
        if mail_check=="Invalid Email":
            mesaagemail="Email ID Not VAlid"
            return render_template('Reg.html',mesaagemail=mesaagemail)
        elif pass_check!="Valid Password":
```

```
        mesaagemail=pass_check

        return render_template('Reg.html',mesaagemail=mesaagemail)

    else:

        sql="SELECT * FROM REGISTER WHERE MAIL_ID=?"

        prep_stmt=ibm_db.prepare(conn,sql)

        ibm_db.bind_param(prep_stmt,1,email)

        ibm_db.execute(prep_stmt)

        account=ibm_db.fetch_assoc(prep_stmt)

        print(account)

        print(company)

        if account:

            registeraccount="Account already exists! Log in to continue !"

        else:

            insert_sql="INSERT INTO REGISTER
(COMPANY,LOCATION,MAIL_ID,USER_ID,PASSWORD)values(?,?,?,?,?)"

            prep_stmt=ibm_db.prepare(conn,insert_sql)

            ibm_db.bind_param(prep_stmt,1,company)

            ibm_db.bind_param(prep_stmt,2,location)

            ibm_db.bind_param(prep_stmt,3,email)
```

```
                    ibm_db.bind_param(prep_stmt,4,username)

                    ibm_db.bind_param(prep_stmt,5,password)

                    ibm_db.execute(prep_stmt)

                    print("inserted")

                    subject="Registration successfull"

                    html_content="Manage Your Stock Efficiently"

                    sendmail(API,from_email,email,subject,html_content)

                    registermsg="Registration successfull. Log in to continue !"

    else:

        print("not post")

        pass

    return
render_template('Reg.html',error=error,registermsg=registermsg,registeraccount=registeraccoun
t)

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

def login():

    error = None

    if request.method=='POST':

        username=request.form['username']
```

```
password=request.form['password']

sql="SELECT * FROM REGISTER WHERE MAIL_ID=? 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['logged_in']=True

    session['id']=account['INVENTORY_ID']

    session["username"]=account["USER_ID"]

    session['company_name']=account['COMPANY']

    session['company_mail']=account["MAIL_ID"]

    flash("Logged in successfully!")

    return redirect(url_for("DashBoard"))

else:

    error="Incorrect username / password"

    return render_template('Reg.html',error=error)
    else:

        pass

    return render_template('Reg.html',error=error)

def is_logged_in(f):

    @wraps(f)

    def wrap(*args, **kwargs):

        if 'logged_in' in session:

            return f(*args, **kwargs)

        else:

            flash('Please login', 'info')

            return redirect(url_for('login'))

    return wrap

@app.route('/DashBoard')

@is_logged_in

def DashBoard():

    count()

    list=procount()

    list_sup=supcount()

    return
```

```
render_template('Dashboard.html',username=session["username"],products=session['total_pro'],supplier=session['total_sup'],min=session['total_min'],listpy=list,listsup=list_sup)

@app.route('/Reorder')

def Reorder():

    return render_template('Reorder.html')

@app.route('/AddProducts')

def AddProducts():

        drop_list_pro=[]

        drop_sql="SELECT Product_GRP FROM SUPPLIER WHERE Inventory_ID=?"

        stmt=ibm_db.prepare(conn,drop_sql)

        ibm_db.bind_param(stmt,1,session['id'])

        ibm_db.execute(stmt)

        drop=ibm_db.fetch_assoc(stmt)

        print(drop)

        while drop != False:

            print ("Supplier Name : ", drop["PRODUCT_GRP"])

            drop_list_pro.append(drop["PRODUCT_GRP"])

            drop = ibm_db.fetch_assoc(stmt)

        resultList_pro = list(dict.fromkeys(drop_list_pro)
        drop_list_sup=[]

        drop_sql="SELECT Supplier_Name FROM SUPPLIER WHERE Inventory_ID=?"

        stmt=ibm_db.prepare(conn,drop_sql)

        ibm_db.bind_param(stmt,1,session['id'])

        ibm_db.execute(stmt)

        drop=ibm_db.fetch_assoc(stmt)

        print(drop)

        while drop != False:

            print ("Supplier Name : ", drop["SUPPLIER_NAME"])

            drop_list_sup.append(drop["SUPPLIER_NAME"])

            drop = ibm_db.fetch_assoc(stmt)

        resultList_sup = list(dict.fromkeys(drop_list_sup))

        return render_template('AddProducts.html',drop_sup=resultList_sup,drop_pro=resultList_pro)

@app.route("/addPro",methods=['GET','POST'])

def addPro():

    if request.method=='POST':

        pro_id=request.form['pro_id']

        pro_grp=request.form['pro_grp'].title()
```

```
pro_name=request.form['pro_name'].title()

Tot_Q=request.form['tot_quantity']

Available=request.form['available']

Shipped=request.form['Shipped']

Minimum_stock=request.form['min_stock']

Supplier=request.form.get('sup')

date=request.form['date']

sql="SELECT * FROM PRODUCT WHERE Product_ID=? AND Inventory_ID=? "

prep_stmt=ibm_db.prepare(conn,sql)

ibm_db.bind_param(prep_stmt,1,pro_id)

ibm_db.bind_param(prep_stmt,2,session['id'])

ibm_db.execute(prep_stmt)

account=ibm_db.fetch_assoc(prep_stmt)

print(account)

if account:

    error="Product alread exist!"

    flash("Product alread exist!")

    return redirect(url_for("AddProducts"))

else:
        insert_sql="INSERT INTO PRODUCT (Inventory_ID,Product_ID,Product_GRP,Product_Name,TOT_Quantity,Available_product,Shipped,Min_stock,Supplier,Date) VALUES (?,?,?,?,?,?,?,?,?,?)"

        prep_stmt=ibm_db.prepare(conn,insert_sql)

        ibm_db.bind_param(prep_stmt,1,session["id"])

        ibm_db.bind_param(prep_stmt,2,pro_id)

        ibm_db.bind_param(prep_stmt,3,pro_grp)

        ibm_db.bind_param(prep_stmt,4,pro_name)

        ibm_db.bind_param(prep_stmt,5,Tot_Q)

        ibm_db.bind_param(prep_stmt,6,Available)

        ibm_db.bind_param(prep_stmt,7,Shipped)

        ibm_db.bind_param(prep_stmt,8,Minimum_stock)

        ibm_db.bind_param(prep_stmt,9,Supplier)

        ibm_db.bind_param(prep_stmt,10,date)

        ibm_db.execute(prep_stmt)

        flash(" Product added Successfully !")

alertmsg=alert(pro_id,pro_grp,pro_name,Tot_Q,Available,Shipped,Minimum_stock,Supplier)

        if alertmsg=="Mail Sent":

            flash("Check Mail For Low Stock Details!")
            print("product added")

            return redirect(url_for("AddProducts"))

    else:

        pass

        return redirect(url_for("AddProducts"))

@app.route("/updatePro",methods=['GET','POST'])

def updatePro():

    error = None

    if request.method=='POST':

        pro_id=request.form['pro_id']

        pro_grp=request.form['pro_grp'].title()

        pro_name=request.form['pro_name'].title()

        Tot_Q=request.form['tot_quantity']

        Available=request.form['available']

        Shipped=request.form['Shipped']

        Minimum_stock=request.form['min_stock']

        Supplier=request.form.get('sup')

        date=request.form['date']

        update_sql="UPDATE PRODUCT SET
```

```
(Product_GRP,Product_Name,TOT_Quantity,Available_product,Shipped,Min_stock,Supplier,Date) = (?,?,?,?,?,?,?,?) WHERE Inventory_ID=? AND Product_ID=? "

      prep_stmt=ibm_db.prepare(conn,update_sql)

      ibm_db.bind_param(prep_stmt,1,pro_grp)

      ibm_db.bind_param(prep_stmt,2,pro_name)

      ibm_db.bind_param(prep_stmt,3,Tot_Q)

      ibm_db.bind_param(prep_stmt,4,Available)

      ibm_db.bind_param(prep_stmt,5,Shipped)

      ibm_db.bind_param(prep_stmt,6,Minimum_stock)

      ibm_db.bind_param(prep_stmt,7,Supplier)

      ibm_db.bind_param(prep_stmt,8,date)

      ibm_db.bind_param(prep_stmt,9,session["id"])

      ibm_db.bind_param(prep_stmt,10,pro_id)

      ibm_db.execute(prep_stmt)

      print("product Updated")

      flash("Product updated Successfully !")

      alert(pro_id,pro_grp,pro_name,Tot_Q,Available,Shipped,Minimum_stock,Supplier)

      return render_template('AddProducts.html',error=error)

  else:
   pass

   return render_template('AddProducts.html',error=error)

@app.route("/addSup",methods=['GET','POST'])

def addSup():

  if request.method=='POST':

      sup_id=request.form['supID']

      sup_name=request.form['name'].title()

      location=request.form['location'].title()

      phone=request.form['phone']

      pro_grp=request.form['pro_grp'].title()

      email=request.form['email']

      mail_check=checkmail(email)

      if mail_check=="Invalid Email":

           mesaagemail="Email ID Not Valid"

           flash(mesaagemail)

           return redirect(url_for("AddSupplier"))

      sql="SELECT Supplier_ID FROM SUPPLIER WHERE SUPPLIER_ID=? AND
Inventory_ID=?"

      prep_stmt=ibm_db.prepare(conn,sql)
      ibm_db.bind_param(prep_stmt,1,sup_id)

      ibm_db.bind_param(prep_stmt,2,session["id"])

      ibm_db.execute(prep_stmt)

      account=ibm_db.fetch_assoc(prep_stmt)

      print(account)

      if account:

          flash("Supplier alread exist!")

          return redirect(url_for("AddSupplier"))

      else:

          insert_sql="INSERT INTO SUPPLIER
(Inventory_ID,Supplier_ID,Supplier_Name,Location,PH_Number,Product_GRP,SupMail_ID)
VALUES (?,?,?,?,?,?,?)"

          prep_stmt=ibm_db.prepare(conn,insert_sql)

          ibm_db.bind_param(prep_stmt,1,session["id"])

          ibm_db.bind_param(prep_stmt,2,sup_id)

          ibm_db.bind_param(prep_stmt,3,sup_name)

          ibm_db.bind_param(prep_stmt,4,location)

          ibm_db.bind_param(prep_stmt,5,phone)

          ibm_db.bind_param(prep_stmt,6,pro_grp)
          ibm_db.bind_param(prep_stmt,7,email)

          ibm_db.execute(prep_stmt)

          print("Supplier added")

          flash("Supplier added Successfully !")

          return redirect(url_for("AddSupplier"))

  else:

      pass

      return render_template('AddSupplier.html')

@app.route("/sendordermail",methods=['GET','POST'])

def sendordermail():

  if request.method=='POST':

      print(session['company_mail'])

      #from_email=session['company_mail']

      to_email=request.form['tomail']

      subject=request.form['subject']

      html_content=request.form['text']

      print(html_content)

      sendmail(API,from_email,to_email,subject,html_content)

      print("Mail sent from sendordermail")
      return redirect(url_for("Reorder"))
  else:
      print("not post from sendordermail")
      return redirect(url_for("Reorder"))
@app.route("/sendreport")
def sendreport():
  sendreportmail()
  print("Mail sent from sendreport")
  return redirect(url_for("DashBoard"))
@app.route('/logout',methods=['GET','POST'])
@is_logged_in
def logout():
  #session['_flashes'].clear()
  session.clear()
  flash('You are now logged out', 'success')
  print("Logged Out")
  return redirect(url_for('login'))
if __name__=='__main__':
  app.run(debug=False)
```

GITHUB & PROJECT DEMO LINK

GITHUB LINK

https://github.com/IBM-EPBL/IBM-Project-30516-1660147931

PROJECT DEMO LINK

https://drive.google.com/file/d/1UHPNPr_hZ1kG_TsjIHakp3hVgFLqZqML/view?usp=share_link