

# **Project report**

**Team ID: PNT2022TMID27088**

**Project Name Project - Inventory  
Management System for Retailers**

# **INTRODUCTION**

1.1 Project Overview

1.2 Purpose

## **2. LITERATURE SURVEY**

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

## **3. IDEATION & PROPOSED SOLUTION**

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

## **4. REQUIREMENT ANALYSIS**

4.1 Functional requirement

4.2 Non-Functional requirements

## **5. PROJECT DESIGN**

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

## **6. PROJECT PLANNING & SCHEDULING**

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

## **7. CODING & SOLUTIONING (Explain the features added in the project along with code)**

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

## **8. TESTING**

8.1 Test Cases

8.2 User Acceptance Testing

## **9. RESULTS**

9.1 Performance Metrics

## **10. ADVANTAGES & DISADVANTAGES**

## **11. CONCLUSION**

## **12. FUTURE SCOPE**

## **13. APPENDIX**

Source Code

GitHub & Project Demo Link

# **1.INTRODUCTION**

## **1.1 OVERVIEW**

### **What is Inventory management software?**

Inventory management software streamlines all the inventory operations from procurement to sales with real-time tracking and helps gain control across outlets. An effective and reliable inventory management system eliminates error-prone manual tasks with automation for better tracking and increased visibility to run business with high customer satisfaction having optimum inventory available, always.

### **Who uses Inventory management software?**

In order to enjoy complete inventory control, businesses ranging from a single store to multiple-stores rely upon good retail inventory software. From using inventory software to keep up well with the changing demand and make more sales for small businesses, to managing the supply chain across warehouses and outlets from a centralized digital platform, stock inventory management software serves as an essential tool to run a business hassle-free.

## **1.2PURPOSE**

### **Why do business need an Inventory management software?**

Inventory management system facilitates businesses to boost their growth, serve customers well, and increases sales by automating the key inventory operations. These inventory management tools give a complete view of what's happening in the inventory from fast moving stocks to dead stocks and make data-driven timely pricing decisions without affecting the margin to run business with increased profits.

### **What is the main purpose of inventory management?**

The main purpose of inventory management is to help businesses easily and efficiently manage the ordering, stocking, storing, and using of inventory. By effectively managing your inventory, you'll always know what items are in stock, how many of them there are, and where they are located.

Plus, practicing strong inventory management allows you to understand how you use your inventory—and how demand changes for it—over time. You can zero in on

exactly what you need, what's not so important, and what's just a waste of money. That's using inventory management to practice inventory control. By the way, inventory control is the balancing act of always having enough stock to meet demand, while spending as little as possible on ordering and carrying inventory.

## **2.LITERATURE SURVEY**

### **2.1EXISTING PROBLEM**

- i. Lack of Inventory Visibility
- ii. Identifying Incorrectly Located Materials
- iii. Keeping up with Overstocks
- iv. Managing Inventory Waste & Defects
- v. Lack of Centralized Inventory Hub
- vi. Supply Chain Complexity
- vii. Insufficient Order Management and Poor Production Planning:
- viii. Lack of Expertise and Poor communication
- ix. Managing Warehouse Space and Efficiency
- x. Supply Chain Complexity

### **2.2REFERENCES**

- i. <https://www.tranquilbs.com/blog/>
- ii. <https://www.kapturecrm.com/blog/20-top-challenges-solutions-of-inventory-management/>

### **2.3 PROBLEM STATEMENT DEFINITION**

#### **i. Lack of Inventory Visibility**

If you're unable to locate or identify stocks in your inventory, shipping products on time becomes very difficult, and this can dent your business reputation.

Inventory that is incomplete, difficult to find, or erroneous, is sure to hamper your bottom line.

In fact, the most common reason for delayed, wrong, or partial shipments is the difficulty of locating or identifying inventory in the warehouse.

Receiving in finding the correct stock is critical for ensuring warehouse efficiency, as well as good experiences for the customer.

## **ii. Identifying incorrectly located Materials**

When there is no proper system to track products, materials, or equipment in the store, it can be cumbersome and time-consuming to find them when you have sales orders.

After all, a warehouse may typically store thousands of products. This can delay sales and make customers unhappy.

## **iii. Keeping up with Overstocks**

When you purchase new materials with a few unsold products lying in your warehouse, it can affect your profitability.

This situation mostly arises due to the inefficiencies of manual processes, which causes poor control of stock.

Storing too much stock is as bad as storing too little, as overstocking hampers your cash flow and creates problems related to inventory, like storage, or loss.

## **iv. Managing Inventory Waste & Defects**

Though it may seem small, it is one of the most common and repetitive *inventory management problems* that can cause huge losses eventually.

To be able to fulfil orders in time, it is essential that you maintain optimal inventory.

Without standard procedures and untrained operators, you could end up with inventory that gets damaged or wasted – and this can not only prove to be very expensive but also lead to dissatisfied customers.

v. **Lack of Centralized Inventory Hub**

Stocktaking becomes very challenging when you have inventories in multiple locations.

Discrete stock data from various locations makes shipping complex, resulting in delays.

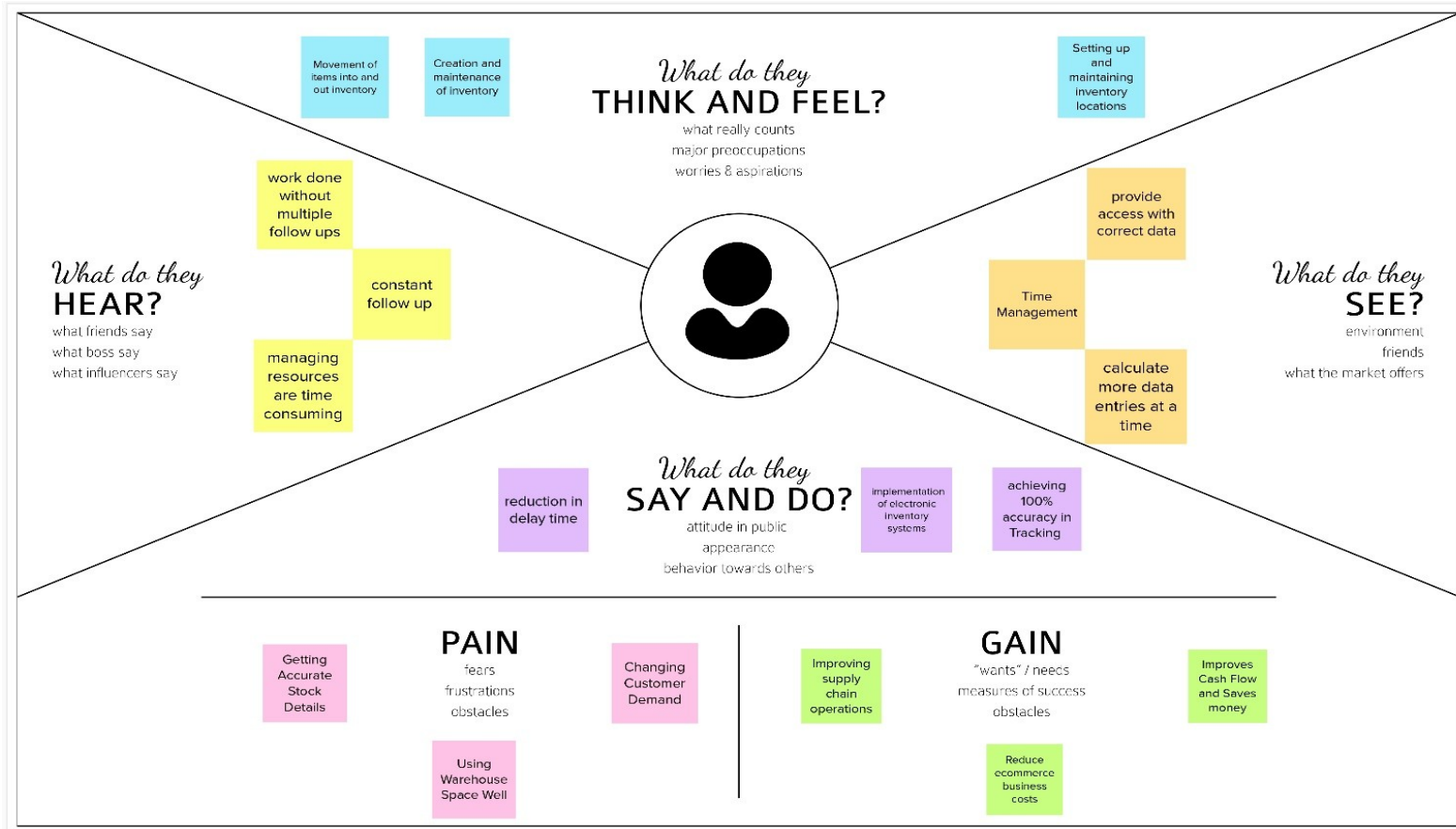
It's one of the biggest and continual challenges faced by most businesses today.



### 3.IDEATION AND PROPOSED SOLUTIONS

#### 3.1EMPATHY MAP CANVAS

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community.



#### 3.2 IDEATION AND BRAINSTROMING

**Praveen M**

- Take forecasting seriously
- Sell older inventory first

- Give each variant a dedicated warehouse bin

### **Rohan Sri M**

- Calculate your total costs.
- Track both inventory and sales data
- Finding the right balance between the costs of inventory

### **Praveen Kumar R**

- Economic order quantity (EOQ)
- Prioritize with ABC analysis
- Give each variant a dedicated warehouse bin

### **Prasanth M**

- Inventory trends inform marketing plans
- Automate tracking with barcode labels
- Don't be afraid to re-evaluate your company's inventory control needs

## **IDEAS**

### **MULTIPLE BRANCH**

- Products can be exchanged between the branches by the same company
- Managing all products using cloud
- Using spreadsheet for the stock management
- Multiple branches cause same spreadsheet with location

### 3.4 Proposed Solution Template:

Project team shall fill the following information in the proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The main aim of this project is to keep the stock in such a way that the number of product is having a good quantity
2.	Idea / Solution description	This research demonstrates a cloud based inventory management project for maintaining the items in the supermarket.
3.	Novelty / Uniqueness	Mail To the Retailers: The chat box will immediately sent the mail to the retailers saying that this particular item is reached its lower limit
4.	Social Impact / Customer Satisfaction	Improvement in fast deliverable to the customers
5.	Business Model (Revenue Model)	Hosting the cloud application to the web host like Hostinger, GoDaddy, Miles Web

6.	Scalability of the Solution	This system can be further enhanced by creating a web app for customers, so that they can have an insightful view of stocks of a store. It is a best advantage for customers, because they can be in their house and see the stocks.
----	-----------------------------	--

### 3.4 Problem Solution Fit:

- Customer segments
- Jobs to be Done/Problems
- Triggers
- Emotions: Before/After
- Available Solutions
- Behaviour
- Channels of Behaviour
- Customer Constraints
- Problem root cause
- Your Solution

## 4.REQUIREMENT ANALYSIS

### Functional Requirements:

Following are the functional requirements of the proposed solution.

<b>FR No.</b>	<b>Functional Requirement (Epic)</b>	<b>Sub Requirement (Story / Sub-Task)</b>
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	Login	Sign into the application by entering the Email and Password
FR-4	Dashboard	Check the stock availability
FR-5	Add items to cart	Consumers can add their product to cart for multiple orders.
FR-6	Stock Update	If there is no stock then they can add it to wishlist to get notified when the stock is available.

## Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	<b>Usability</b>	While usability determines how effective implementing an inventory tracking system is in your business. If it takes hours for your staff to learn the ins and outs of the software, then it's probably not worth buying.
NFR-2	<b>Security</b>	The process of ensuring the safety and optimum management control of stored goods. It is of central importance for optimum warehouse management because the performance of a company stands or falls with the safety and efficiency of a

		warehouse.
NFR-3	<b>Reliability</b>	Relying on manual inventory counts to know what you have will only guarantee high inefficiencies and a loss of customers.
NFR-4	<b>Performance</b>	Creating systems for logging products, receive them into inventory, track changes when sales occur, manage the flow of goods from purchasing to final sale and check number of stocks.
NFR-5	<b>Availability</b>	Whether a specific item is available for customer orders. Additional information provided by retailers may include the quantity available.
NFR-6	<b>Scalability</b>	They should use an automated inventory management system for inventory tracking. This will make your business much more scalable so that you can

		continue building consistent growth and take advantage of increased sales.
--	--	--

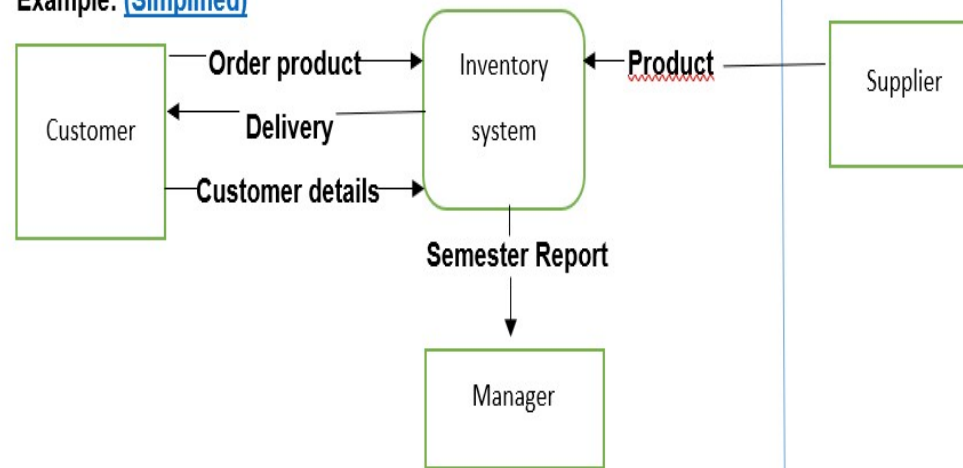
## **5.PROJECT DESIGN**

### **5.1 Data Flow Diagrams**

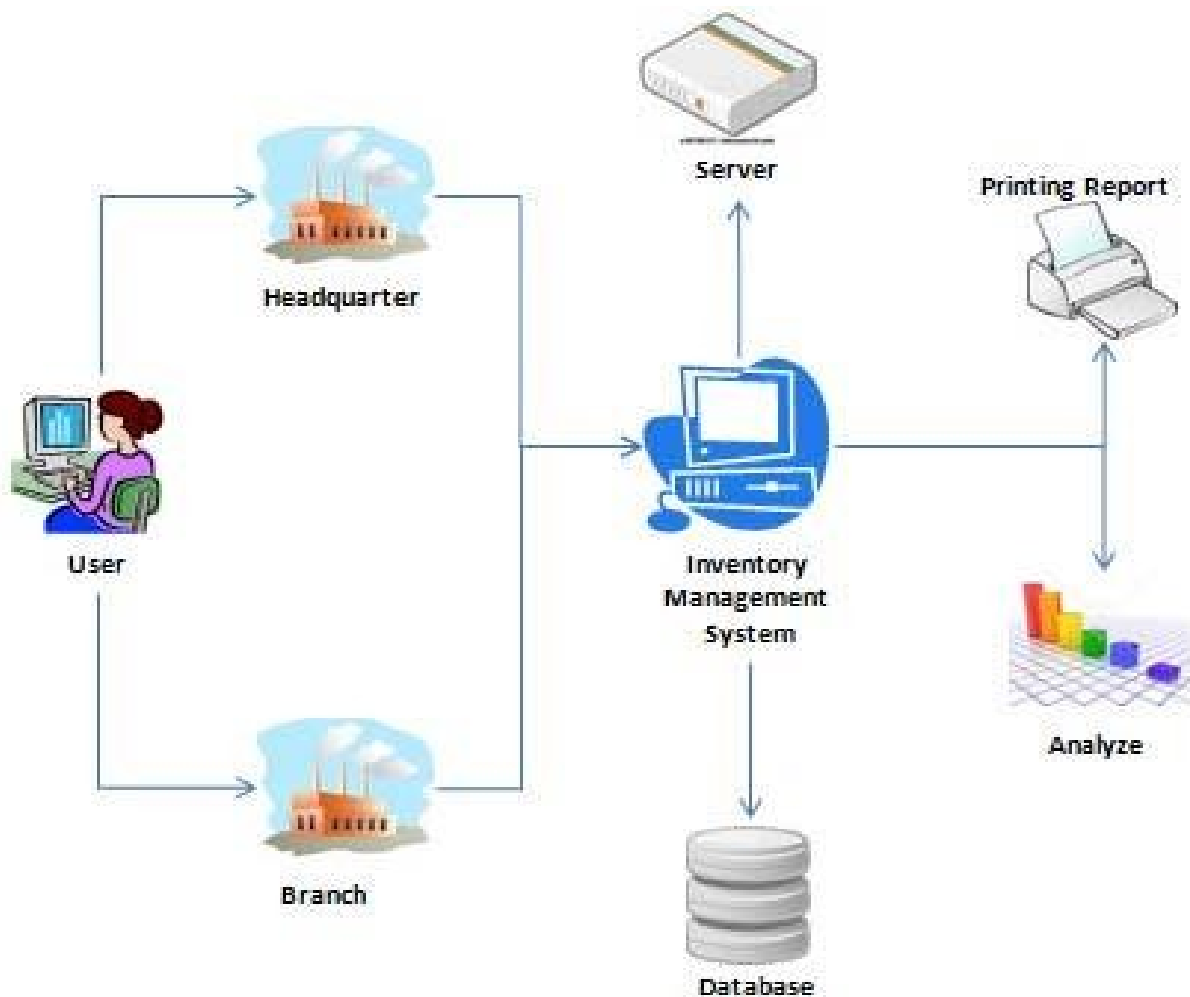
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



Example: (Simplified)



## 5.2 Solution and Technical Architecture



**Table-1 : Components & Technologies:**

S.N o	Component	Description	Technology
1	User Interface	Web UI with Chatbot	HTML, CSS, Bootstrap, JQuery
2	Calculating Products Count	By entering barcode details into the application	Zia Barcode Scanner

3	Showing high demand product	By the products data in IBMdb2	Data Visualization using Python Bar plot by MatplotlibLibrary
4.	Alert and Notification	Alerting the retailers regarding the low stock count of the product	SendGrid
5	Chat	Chat with watson assistant	IBM Watson Assistant
6	Cloud Database	Database Service on Cloud	IBM DB2
7	File Storage	File storage requirements	IBM Object Storage
8	External API-1 Barcode	To Scan the product barcode	Zia Barcode Scanner
9	Infrastructure (Server / Cloud)	Cloud Server Configuration	Cloud Foundry, Kubernetes

**Table-2: Application Characteristics:**

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Styling our page, Python flask microframework	Python Flask, Bootstrap
2.	Security Implementations	For securing g our cloud data	SSL Certificates

3.	Scalable Architecture	Three - tier architecture (MVC)	Web server - HTML, CSS, Javascript Application server - Python Flask, Docker, Container Registry Database server - IBM DB2
4.	Availability	availability of application	IBM Load Balancer
5.	Performance	5 requests per seconds, Use of Local Machine Cache Memory	IBM Cloud, CDN

## 6.PROJECT PLANNING AND SCHELUDING

### 6.1 Sprint Planning and Execution

## Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	PDA-1	As a user, I can register for the application by entering my Name, email, password, confirming my password and Age.	3	High	Prasanth M
		PDA-2	As a user, I will receive confirmation email once I have registered for the application	3	Medium	Praveen M
		PDA-3	As a user, I can register for the application through Gmail	5	Medium	Praveen Kumar R
	Login	PDA-4	As a user, I can Sign in to the application by entering email and password	1	High	Rohan Sri M, Praveen M
		PDA-5	As a user, I can reset my password using Forgot Password option	4	Medium	Prasanth M
		PDA-6	As a user, I can add their product to cart for multiple orders.	2	Low	Praveen Kumar R, Rohan Sri M
		PDA-8	As a user, I can Check the stock availability	2	Low	Prasanth M
Sprint-2	Home Page	PDA-10	As a user, I can view the homepage of the website	2	Medium	Praveen M
	About Page	PDA-12	As a user, I can view the about page on the website and get information related to the Inventory management system.	2	Medium	Praveen Kumar R
	Register as User	PDA-13	As a user, I will get notified when the stock is available by registering as user.	3	High	Rohan Sri M

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3	Send Request	PDA-14	As a user, I can raise a request for stock availability.	2	High	Praveen M
	View Requests	PDA-15	As a user, I can view the stocks in my cart.	4	Medium	Praveen Kumar R
	Maintenance	PDA-16	As an admin, I can maintain the databases involved	2	Medium	Prasanth M, Rohan Sri M
	Handle Requests	PDA-17	As an admin, I can view all requests for stocks.	1	High	Praveen M
Sprint-4		PDA-18	As an admin, I can delete requests that are past some time period or have been closed	3	Low	Praveen Kumar R
	Solving User Queries	PDA-19	Creating an ChatBot that helps to solve the queries of the user.	2	High	Rohan Sri M

## Project Tracker, Velocity & Burndown Chart: (4 Marks)

### Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)		Sprint Release Date (Actual)
Sprint-1	8	5 Days	27 Oct 2022	31 Oct 2022			
Sprint-2	13	6 Days	1 Nov 2022	06 Nov 2022			
Sprint-3	11	6 Days	07 Nov 2022	12 Nov 2022			
Sprint-4	9	6 Days	14 Nov 2022	19 Nov 2022			

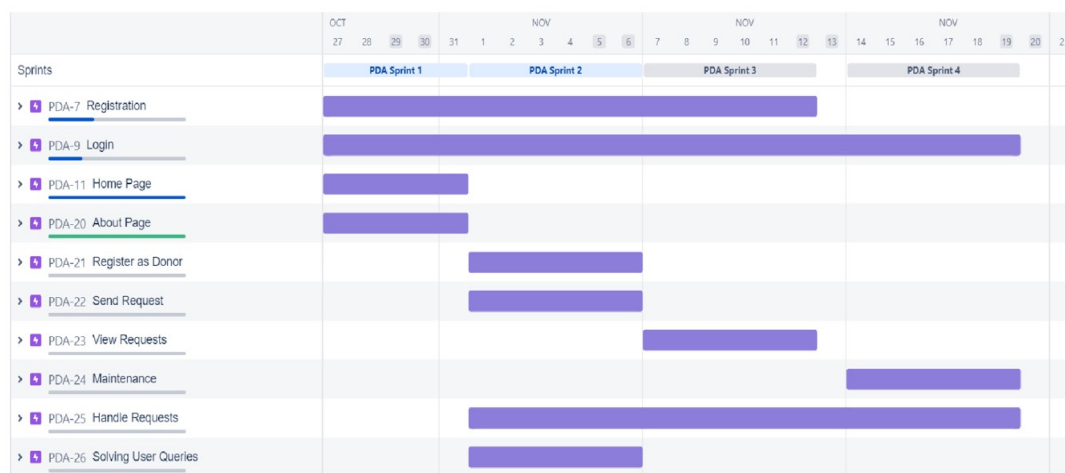
## Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

## Burndown Chart:

A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

<https://www.visual-paradigm.com/scrum/scrum-burndown-chart/>

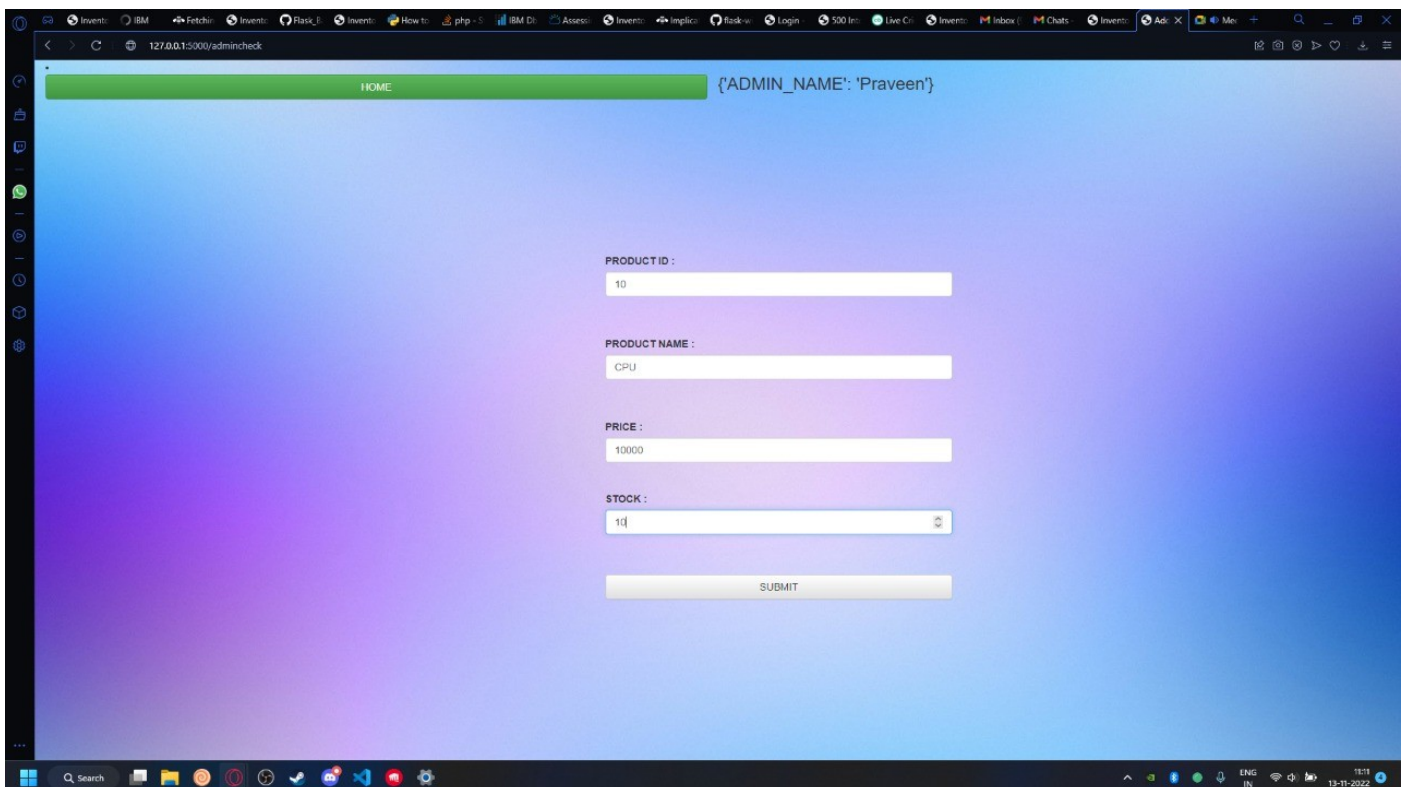
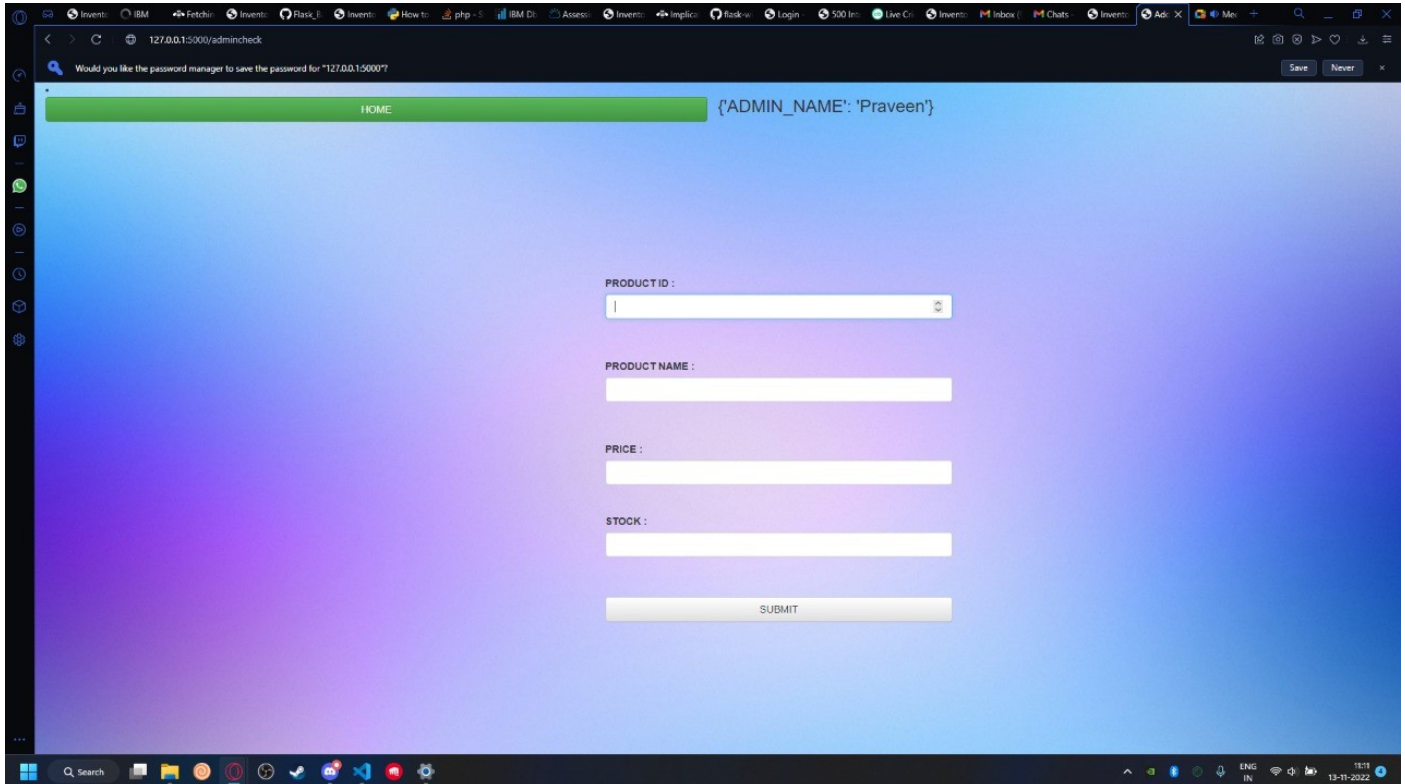


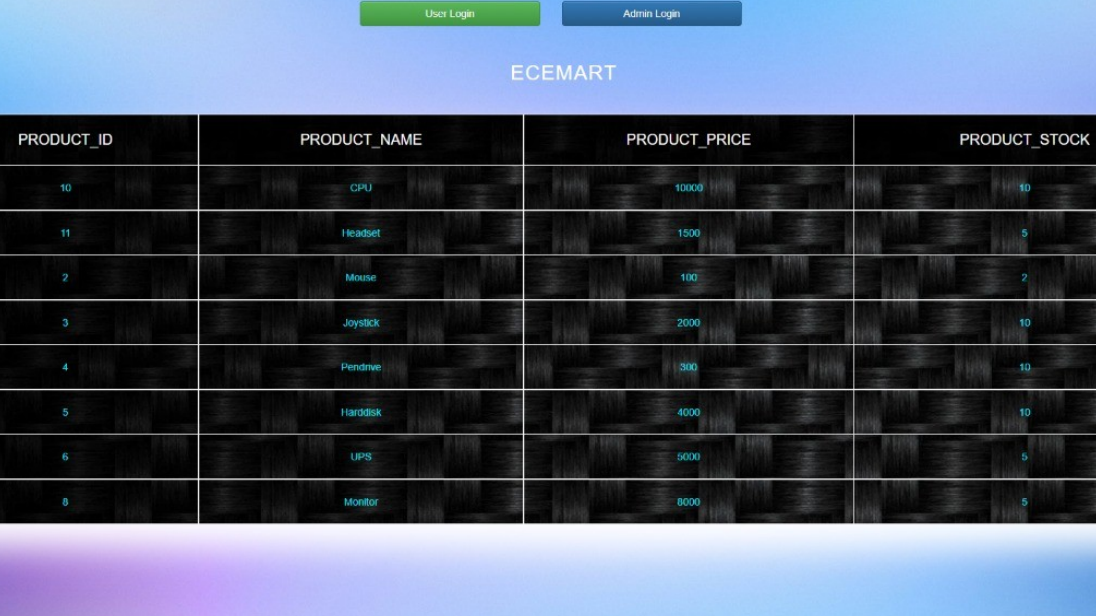


## 7. CODING AND SOLUTIONS

### 7.1 Feature 1

#### Adding the items





PRODUCT_ID	PRODUCT_NAME	PRODUCT_PRICE	PRODUCT_STOCK
10	CPU	10000	10
11	Headset	1500	5
2	Mouse	100	2
3	Joystick	2000	10
4	Pendrive	300	10
5	Harddisk	4000	10
6	UPS	5000	5
8	Monitor	8000	5

## 7.2 Feature 2

### Notify low stock

The screenshot shows the ECEMART admin dashboard. At the top, a message states "Admin have been notified through mail". Below this, there are two tabs: "HOME" and "Notify Out of Stock". The "Notify Out of Stock" tab is active, displaying a table of products with low stock levels.

PRODUCT_ID	PRODUCT_NAME	PRODUCT_PRICE	PRODUCT_STOCK
10	CPU	10000	10
11	Headset	1500	5
2	Mouse	100	2
3	Joystick	2000	10
4	Pendrive	300	10
5	Harddisk	4000	10
6	UPS	5000	5
8	Monitor	8000	5

The screenshot shows a Gmail inbox with four emails from rohansri2002@gmail.com. The subject of all emails is "Out of stock". The emails are dated 11:03 AM (26 minutes ago), 11:10 AM (19 minutes ago), 11:14 AM (15 minutes ago), and 11:26 AM (3 minutes ago). The first two emails have the body text "Some products are out of stock please check it out". The last two emails are marked as "to me" and include a link to sendgrid.net.

Search in mail

Active

1 of 376

Out of stock

rohansri2002@gmail.com

Some products are out of stock please check it out

11:03 AM (26 minutes ago)

rohansri2002@gmail.com

Some products are out of stock please check it out

11:10 AM (19 minutes ago)

rohansri2002@gmail.com via sendgrid.net

to me

11:14 AM (15 minutes ago)

rohansri2002@gmail.com via sendgrid.net

to me

11:26 AM (3 minutes ago)

Reply Forward



## 8.1 Test Cases

## 8.1 Test Cases

1					Date	12-Nov-22									
2					Team ID	PNT2022TMD27088									
3					Project Name	Inventory Management System for Retailers									
4					Maximum Marks	4 marks									
5	Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By	
6	LoginPage_TC_OO 1	Functional	Home Page	Verify user is able to see the Login/Signup	HTML	1.Enter URL and click go 3.Signup/Login that displayed.	<a href="http://159.122.181.203:321/53/">http://159.122.181.203:321/53/</a>	Login/Signup popup should display	Working as expected	Pass	Executed Successfully	Y	Nil	Good on you	
7	LoginPage_TC_OO 2	UI	Login Page	Verify the UI elements in Login/Signup popup	IBM database	1.Verify login/Signup popup with below UI elements: a.email text box b.password text box c.Login button	<a href="http://159.122.181.203:321/53/adminsignup">http://159.122.181.203:321/53/adminsignup</a>	Application should show below UI elements: a.email text box b.password text box c.Login button	Working as expected	Pass	Executed Successfully	Y	Nil	Good on you	
8	LoginPage_TC_OO 3	Functional	Login Page	Verify user is able to log into application with Valid credentials	IBM database	1.Click on Login 2.Enter Valid email in Email text box 3.Enter valid password in password text box	<a href="http://159.122.181.203:321/53/adminsignup">http://159.122.181.203:321/53/adminsignup</a>	User should navigate to user account homepage	Working as expected	Pass	Executed Successfully	Y	Nil	Good on you	
9	LoginPage_TC_OO 4	Functional	Login page	Verify user is able to log into application with Invalid credentials	IBM database	1.Enter Invalid username/email in Email text box 2.Enter valid password in password text box 3.Click on login button	Username: mah@gmail password: mah123	Application should show 'incorrect email or password' validation message.	Working as expected	Pass	Executed Successfully	Y	Nil	Good on you	
10	LoginPage_TC_OO 4	Functional	Home page	Verify user is able to use the add to cart efficiently	IBM database	1.Click on the add to cart. 2.add items or product to the cart and add more items to the list.	adding product to the list	The product is successfully added to the list	Working as expected	Pass	Executed Successfully	Y	Nil	Good on you	
11	LoginPage_TC_OO 5	Functional	Home page	Verify user is able to use the Low stock mail efficiently	sendgrid	1.Click on the notify out of stock. 2.verify the mail has received by the Admin.	Some products are out of stock please check it out	Some products are out of stock please check it out	Working as expected	pass	Executed Successfully	y	Nil	Good on you	
12															

## 8.2 User Acceptance Testing

### Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [Smart fashion recommender] project at the time of the release to User Acceptance Testing (UAT).

### Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
IBM DB	10	4	6	3	23
Docker	4	7	3	0	14
Sendgrid	6	3	0	1	10
Python code	9	2	4	5	20
HDML CSS code	3	0	2	0	5
Cost/Time saving	10	5	4	3	22
Totals	42	21	19	12	94

### Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Home Page	7	0	0	7
About page	4	0	0	4
Log in	11	0	0	11
Sign in	5	0	0	5
Main page	9	0	0	9
Cost/Time saving	56	0	0	56
Totals	92	0	0	92

## 9. ADVANTAGES AND DISADVANTAGES

### 9.1 Advantages

- **Better Inventory Accuracy:** With solid inventory management, you know what's in stock and order only the amount of inventory you need to meet demand.
- **Reduced Risk of Overselling:** Inventory management helps track what's in stock and what's on backorder, so you don't oversell products.
- **Cost Savings:** Stock costs money until it sells. Carrying costs include storage handling and transportation fees, insurance and employee salaries. Inventory is also at risk of theft, loss from natural disasters or obsolescence.
- **Avoiding Stockouts and Excess Stock:** Better planning and management helps a business minimize the number of days, if any, that an item is out of stock *and* avoid carrying too much inventory. Learn more about solving for stockouts in our "[Essential Guide to Inventory Control](#)."
- **Greater Insights:** With inventory tracking and stock control, you can also easily spot sales trends or track recalled products or expiry dates.
- **Better Terms With Vendors and Suppliers:** Inventory management also provides insights about which products sell and in what volume. Use that knowledge as leverage to negotiate better prices and terms with suppliers.
- **More Productivity:** Good inventory management solutions save time that could be spent on other activities.
- **Increased Profits:** A better understanding of both availability and demand leads to higher inventory turnover, which leads to greater profits.
- **A More Organized Warehouse:** An efficient warehouse with items organized based on demand, which items are often sold together and other factors reduces labor costs and speeds order fulfillment.
- **Better Customer Experience:** Customers that receive what they order on time are more loyal.

## 9.2 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.
- **Risk of System Crashes:** Software does crash. However, you can remove the risk of data and productivity loss by using cloud-based platforms.
- **Malicious Hacks:** Malicious hacks are a risk to all businesses. The Internet of Things (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.
- **Reduced Physical Audits:** When you automate some warehouse operations, it's easy to skip a physical inventory check. Solve this by instituting regular audits.



## **10. CONCLUSION**

As you can see the importance of inventory management is very serious, it is one of the most important aspects of any business. The aspect of this part of the business is whether or not you can satisfy the demand of your customers if you aren't sure if you have all the materials available to make the final product (Thibodeaux, 2014). Without Wheeled Coach having the proper inventory management they would not be able to supply their customers with their ordered ambulance. And this product is what their entire business is based on, so it is of great importance. When they are choosing from the different types of programs or automated systems to help with keeping records accurate, Wheeled Coach needs to keep in mind that the customer is not concerned with which materials are needed to complete the finished product, but the product I operating as promised based on the contract. This is why they need to make sure that any processes or programs that they do decide to use are going to be beneficial to their needs as well as the needs to service their customers (Warren, 2012). In addition, the plans for the maintenance of having proper inventory levels need to be in place and also adjusted when the company grows and as the business dictates (Thibodeaux, 2014). If Wheeled Coach implements the new suggestions they will be on the right track to having a well established business.

## 12.FUTURE SCOPE

### Scope of Inventory Management System

- **Manage Inventory:** Inventory management helps to manage the stock of the company. it provides proper details of the products what kind of raw material, what are the sizes we require and etc. to the purchasing department.
- **Less Storage:** When the inventory management provides proper information to management, they buy according to them which helps the company to store fewer products.
- **Improve Productivity:** Inventory management helps to improve the productivity of the machines and manpower. Employees are aware of stocks and the quantity that require to produce.
- **Increase Profits:** Inventory management helps to improve the profits of the company. it helps to provide proper information about stocks, that saves the unnecessary expenses on stocks.

## 13.APPENDIX

### Source Code

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

import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
```

```
import ibm_db
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31929;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=bxv32119;PWD=pddAoeX0uA0i3RZv",
'', '')
print ("Database connection established", conn)
```

```
app = Flask(__name__)
```

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

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

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

```
@app.route('/userlogin')
def userlogin():
    return render_template('usersignin.html')
```

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

```
def adminlogin():  
    return render_template('adminsignin.html')
```

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

```
def adminnotify():  
    message =  
Mail(from_email="rohansri2002@gmail.com",to_emails="praveenmahen2001@gmail.com",subject="Out of stock",html_content="<p>Some products are out of stock please check it out</p>")  
  
    try:  
        sg = SendGridAPIClient("SG.WSZyePDcTA0-fe6-gmqHrA.as0fIRaZhgYuZVqFw3yi29TKS03pffffT_avk-rMs1DM")  
        response = sg.send(message)  
        return render_template('userpanel.html',msg="Admin have been notified through mail")  
    except Exception as e:  
        print(e)  
        return render_template('userpanel.html',msg="Error")
```

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

```
        name = request.form['name']  
        email = request.form['email']  
        password = request.form['password']  
        confirmpassword = request.form['confirmpassword']
```

```
sql = "SELECT * FROM USER_DATA WHERE USER_EMAIL=? "
```

```
stmt = ibm_db.prepare(conn, sql)
```

```
ibm_db.bind_param(stmt,1,email)
```

```
ibm_db.execute(stmt)
```

```
account = ibm_db.fetch_assoc(stmt)
```

```
if account:
```

```
    return render_template('index.html', msg="You are already a user, please  
login using your details")
```

```
else:
```

```
    insert_sql = "INSERT INTO USER_DATA VALUES (?,?,?,?)"
```

```
    prep_stmt = ibm_db.prepare(conn, insert_sql)
```

```
    ibm_db.bind_param(prepare_stmt, 1, name)
```

```
    ibm_db.bind_param(prepare_stmt, 2, email)
```

```
    ibm_db.bind_param(prepare_stmt, 3, password)
```

```
    ibm_db.bind_param(prepare_stmt, 4, confirmpassword)
```

```
    ibm_db.execute(prepare_stmt)
```

```
    message =
```

```
Mail(from_email="rohansri2002@gmail.com",to_emails=email,subject="User  
Registration",html_content="<p>You have been successfully registered as a  
user.</p>")
```

```
try:
```

```
    sg = SendGridAPIClient("SG.WSZyePDcTA0-fe6-  
gmqHrA.as0fIRaZhgyuZVqFw3yi29TKS03pfffT_avk-rMs1DM")
```

```
    response = sg.send(message)
```

```
    return render_template('usersignin.html',msg="Registration successfull.  
Please login using your credentials")
```

```
except Exception as e:
```

```
    print(e)
```

```
        return render_template('usersignin.html',msg="unable to send message to your  
mail" )
```

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

```
def usercheck():
```

```
    if request.method == 'POST':
```

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

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

```
sql = "SELECT * FROM USER_DATA WHERE USER_EMAIL=? and USER_PASSWORD= ?"
```

```
stmt = ibm_db.prepare(conn, sql)
```

```
ibm_db.bind_param(stmt,1,email)
```

```
ibm_db.bind_param(stmt,2,password)
```

```
ibm_db.execute(stmt)
```

```
account = ibm_db.fetch_assoc(stmt)
```

```
if account:
```

```
    sql = "SELECT USER_NAME FROM USER_DATA WHERE USER_EMAIL=?"
```

```
    stmt = ibm_db.prepare(conn, sql)
```

```
    ibm_db.bind_param(stmt,1,email)
```

```
    ibm_db.execute(stmt)
```

```
    account = ibm_db.fetch_assoc(stmt)
```

```
    message =
```

```
Mail(from_email="rohansri2002@gmail.com",to_emails=email,subject="User  
registration",html_content="<p>You have been successfully registered as a  
user.</p>")
```

```
    try:
```

```
        sg = SendGridAPIClient("SG.WSZyePDcTA0-fe6-  
gmqHrA.as0fIRaZhgyuZVqFw3yi29TKS03pfffT_avk-rMs1DM")
```

```
        response = sg.send(message)
```

```
        return render_template('userpanel.html',msg=account)
```

```
    except Exception as e:
```

```

        print(e)

    else:
        return render_template('index.html', msg="Your login credentials are not
mached with our records.Please login correctly or signup")

```

```

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

```

```

        name = request.form['name']
        email = request.form['email']
        password = request.form['password']
        confirmpassword = request.form['confirmpassword']
        ece = request.form['secretkey']

        if ece:

```

```

        sql = "SELECT * FROM ADMIN_DATA WHERE ADMIN_EMAIL=? "

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

        if account:
            return render_template('adminsignin.html', msg="You are already an admin,
please login using your details")
        else:
            insert_sql = "INSERT INTO ADMIN_DATA VALUES (?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prepare_stmt, 1, name)
            ibm_db.bind_param(prepare_stmt, 2, email)

```

```

        ibm_db.bind_param(prepare_stmt, 3, password)
        ibm_db.bind_param(prepare_stmt, 4, confirmpassword)

        ibm_db.execute(prepare_stmt)

        return render_template('adminsignin.html',msg="Admin registration
successfull.Please login with your credentials." )
    else:
        return render_template('adminsignin.html',msg="Your secret key is invalid
provide it correctly" )

```

```

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

    if request.method == 'POST':

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

```

```

sql = "SELECT * FROM ADMIN_DATA WHERE ADMIN_EMAIL=? and ADMIN_PASSWORD= ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,email)
ibm_db.bind_param(stmt,2,password)

ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
if account:
    sql = "SELECT ADMIN_NAME FROM ADMIN_DATA WHERE ADMIN_EMAIL=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,email)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)

    return render_template('adminpanel.html',msg=account)

```



```
    else:
        return render_template('adminsignin.html',msg="Your login credentials are not matched with our records.Please login correctly or signup" )
```

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

```
        id = request.form['id']
        name = request.form['name']
        price = request.form['price']
        stock = request.form['stock']
```

```
sql = "SELECT * FROM PRODUCT_DATA WHERE PRODUCT_NAME=? and PRODUCT_ID=? "
```

```
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,name)
ibm_db.bind_param(stmt,2,id)
ibm_db.execute(stmt)
account = ibm_db.fetch_assoc(stmt)
```

```
if account:
```

```
    return render_template('adminpanel.html', msg="Product already added")
else:
```

```
    insert_sql = "INSERT INTO PRODUCT_DATA VALUES (?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, id)
    ibm_db.bind_param(prepare_stmt, 2, name)
    ibm_db.bind_param(prepare_stmt, 3, price)
    ibm_db.bind_param(prepare_stmt, 4, stock)
```

```
ibm_db.execute(prepare_stmt)
```

```
return render_template('adminpanel.html',msg="Product successfully added" )
```

```
if __name__ == '__main__':
```

```
    app.run(host='0.0.0.0', port=5000, debug=True)
```

## Index

```
<!DOCTYPE html>
```

```
<html>
```

```
    <head>
```

```
        <title>Inventory Management</title>
```

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

```
    </head>
```

```
    <body>
```

```
        <div class="header">
```

```
            
```

```
            <div class="space"></div>
```

```
            <div class="account">
```

```
                <a href="/usersignup">User </a>
```

```
                <a href="/adminsingup">Admin </a>
```

```
            </div>
```

```
        </div>
```

```
        <br>
```

```
        <p class="title">INVENTORY MANAGEMENT</p>
```

```
        <p class="title1">About:</p>
```

```
        <p class="about">Inventory management refers to the process of ordering,
storing, using, and
```

```
        selling a company's inventory. This includes the management of raw
materials, components,

        and finished products, as well as warehousing and processing of such
items.</p>

    <p class="about1">Effective inventory management enables businesses to
balance the amount of inventory they have

        coming in and going out. The better a business controls its
inventory, the more money it can save in business

        operations.</p>

    <p class="about2">A business that has too much stock has overstock.
Overstocked businesses have money tied

        up in inventory, limiting cash flow and potentially creating a budget
deficit. This overstocked inventory,

        which is also called dead stock, will often sit in storage, unable to
be sold, and eat into a business's profit

        margin.</p>

    <p class="about3">Globalization, technology and empowered consumers are
changing the way businesses manage inventory.

        Supply chain operators will use technologies that provide significant
insights into how supply chain performance

        can be improved. They'll anticipate anomalies in logistics costs and
performance before they occur and have

        insights into where automation can deliver significant scale
advantages</p>
</body>
</html>
```

## Admin page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/meyer-
reset/2.0/reset.min.css">
```

```

<title>Add item</title>

<!-- Latest compiled and minified CSS -->

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiISIFeK1dGmJRAkycuHAHRg320mUcww7on3RYdg4Va+PmSTsz/
K68vbdEjh4u" crossorigin="anonymous">


<!-- Optional theme -->

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/
Sp" crossorigin="anonymous">

<style>

    .space{

        padding-left: 520px;

    }

    .row{

        margin-left: 140px;

        margin-top: 200px;

    }

```

```

</style>
</head>
<body background= "{{url_for('static', filename='bg.png')}}" >

    <li class="col-sm-6"><a href="/" class="btn form-control btn-
success">HOME</a></li>

    <div class="space">

        <div class="container">

            <h3>{{ msg }}</h3>

            <div class="row" >

                <div class="col-sm-6">

                    <form action = "{{ url_for('adminpanel') }}" method = "POST">

                        <br> <div class="form-group">

                            <label for="id"> PRODUCT ID : </label>

```

```

        <input type = "number" name='id' class="form-control"
id="password">

    </div>

    <br>

    <br>

    <div class="form-group">
        <label for="name"> PRODUCT NAME : </label>

        <input type = "text" name='name' class="form-control"
id="password">

    </div>

    <br>

    <br>

    <div class="form-group">
        <label for="price">PRICE : </label>

        <input type = "number" name='price' class="form-control"
id="password">

    <br>

    <br>

    <div class="form-group">
        <label for="stock">STOCK : </label>

        <input type = "number" name='stock' class="form-control"
id="password">

    </div><br>

    <br>

    <button type="submit" class="btn form-control btn-
default">SUBMIT</button>

    </form>

</div>

</div>

</div>

```

**User**

```

<!DOCTYPE html>

<html>

  <head>

<title>Inventory Management</title>

  </head>

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

  <meta charset="UTF-8">

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

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/meyer-
reset/2.0/reset.min.css">

    <title>Register</title>

    <!-- Latest compiled and minified CSS -->

    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
integrity="sha384-BVYiisIFeKldGmJRAkycuHAHRg320mUcww7on3RYdg4Va+PmSTsz/
K68vbdEjh4u" crossorigin="anonymous">


    <!-- Optional theme -->

    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css"
integrity="sha384-rHyOnLiRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/
Sp" crossorigin="anonymous">

  <body>

<h2>ECEMART</h2>

<h3>{{ msg }}</h3>

<background="bg.png">

    <li class="col-sm-6"><a href="/" class="btn form-control btn-
success">HOME</a></li>

    <li class="col-sm-6"><a href="/adminnotify" class="btn form-control btn-
primary">Notify Out of Stock</a></li>

<div class="table-wrapper">

```

```


```

```

<table class="fl-table">

  <form action = "{{ url_for('adminpanel') }}" method = "POST"></form>

```

```
<thead>
<tr>
  <th>PRODUCT_ID</th>
  <th>PRODUCT_NAME</th>
  <th>PRODUCT_PRICE</th>
  <th>PRODUCT_STOCK</th>
</tr>
</thead>
<tbody>
```

```
  <tr>
    <td>10</td>
    <td>CPU</td>
    <td>10000</td>
    <td>10</td>

  </tr>
  <tr>
    <td>11</td>
    <td>Headset</td>
    <td>1500</td>
    <td>5</td>

  </tr>
  <tr>
    <td>2</td>
    <td>Mouse</td>
    <td>100</td>
    <td>2</td>

  </tr>
  <tr>
    <td>3</td>
```

```
        <td>Joystick</td>
        <td>2000</td>
        <td>10</td>

</tr>
<tr>
    <td>4</td>
    <td>Pendrive</td>
    <td>300</td>
    <td>10</td>

</tr>
<tr>
    <td>5</td>
    <td>Harddisk</td>
    <td>4000</td>
    <td>10</td>

</tr>
<tr>
    <td>6</td>
    <td>UPS</td>
    <td>5000</td>
    <td>5</td>
</tr>
<tr>
    <td>8</td>
    <td>Monitor</td>
    <td>8000</td>
    <td>5</td>
</tr>
<tbody>
```



```
        </form>

    </table>
</div>
</body>
</html>
```

GitHub & Project Demo Link

<https://github.com/IBM-EPBL/IBM-Project-1026-1658335157>

<http://159.122.181.203:32153/>