TABLE OF CONTENTS

CHAPTER - 1

INTODUCTION

1.1 PROJECT OVERVIEW

This project is aimed at developing a desktop-based application named Inventory Management System for managing the inventory system of any organization.The Inventory Management System (IMS) refers to the system and processes to manage the stock of an organization with the involvement of a Technology system. This system can be used to store the details of the inventory, stock maintenance, update the inventory based on the sales details, and generate sales and inventory reports daily or weekly based.

1.2 PURPOSE

The Inventory Management System is a real time inventory database capable of connecting multiple stores. This can be used to track the inventory of a single store or to manage the delivery of stock between several branches of a larger franchise. However, the system merely records sales and restocking data and provides warning of low stock at any location through email at a specified interval.The goal is to reduce the stress of tracking rather than to holder all store maintenance. Further features may consist of the ability to create reports of sales, but again the explanation is left to the management. In addition, since theft does occasionally occur, the system provides solutions for confirming the store inventory and for correcting stock quantities.Production units use an inventory management system to reduce their transport costs. The system is used to track products and parts as they are transported from a seller to a storeroom, between storerooms, and finally to a retail location or directly to a customer.

CHAPTER - 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

Retail inventory management is the process of ensuring you carry merchandise that shoppers want, with neither too little nor too much on hand. By managing inventory, retailers meet customer demand without running out of stock or carrying excess supply.In practice, effective retail inventory management results in lower costs and a better understanding of sales patterns. Retail inventory management tools and methods give retailers more information on which to run their businesses. Applications have been developed to help retailers track and manage stocks related to their own products.

2.2 REFERENCES

1. A STUDY ON INVENTORY MANAGEMENT OF TATA MOTORS [VIVEK HAMAL PROFESSOR PARUL UNIVERSITY,et al,2022]

Inventory control is associated with planning, procuring, storing and offering suitable material of proper quality, and proper amount in the proper vicinity to coordinate and agenda the manufacturing pastime in an integrative manner for a commercial undertaking. Inventory Management truly is the technique by which a company is provided with the products. The fundamental goal of the have a look at is to figure out the elements that affect the stock of material, the statistics and reviews of stock control and management on the age of production enterprise might be analyzed and to create a powerful usage of stock at enterprise, to triumph over the problems via means of giving viable recommendations.

2. CASE STUDY ON AN ANDROID APP FOR INVENTORY MANAGEMENT SYSTEM WITH SALES PREDICTION FOR LOCAL SHOPKEEPERS IN INDIA [6TH INTERNATIONAL CONFERENCE ON ADVANCED COMPUTING AND COMMUNICATION SYSTEMS (ICACCS),et al, 2020]

The retail sector has widely adopted different inventory management applications and some retail. However, a lot of day-to-day shopping in India happens through local shops. The owners of such mom-and-pop shops do not necessarily have the capital to invest in proprietary applications for setting up an inventory management system. Needless to say the same is the case for any sales prediction software.Many of the shopkeepers end up hoarding a lot of irrelevant and nonprofitable products that lead to financial losses. A very cost-effective and accessible solution for this problem is a mobile application that provides all the features of a point-of-sale system as well as gives future sales insights. It will enable shopkeepers to manage their current product purchases and invoicing. The predictive sales analysis will help them to modify their investments in products and supplies thereby ensuring maximum profits.

3. A STUDY ON INVENTORY MANAGEMENT CONCERNING COCA-COLA [ALUKA MAHESH GOUD, N. LAKSHMI DEEPTHI, et al,2019]

Inventories are often outlined because of the ad of the worth of raw materials, fuels and lubricants, spare elements, maintenance materials, semi-finished merchandise and finished product that square measure offered at a given time. In massive corporations, inventory accounts for a considerable portion of current assets. The corporation has fifteen to half-hour inventories of total assets. Inventories carry with them assets that square measure oversubscribed as special things within the normal course of business. The assets that corporations store as inventory square measure raw materials add progress and finished merchandise. Inventory is often outlined because of the worth of raw materials, fuels and lubricants, spare elements, maintenance materials, semi-finished merchandise and

finished product that square measure offered at a given time.

4. A STUDY ON INVENTORY MANAGEMENT [B. ARCHANA, K.DIVYA, et al,2019]

An inventory Management System is important to ensure quality control in businesses that handle transactions revolving around consumer goods. Without proper inventory control, a large retail store may run out of stock on an important item. A good Inventory Management System will alert the retailer when it is time to reorder. An inventory Management System is also an important means of automatically tracking large shipments. For example, if a business orders ten pairs of socks for retail resale, but only receives nine pairs, this will be obvious upon inspecting the contents of the package, and error is not likely. On the other hand, say a wholesaler orders 100,000 pairs of socks and 10,000 are missing. Manually counting each pair of socks is likely to result in an error. An automated Inventory Management System helps to minimize the risk of error. In retail stores, an Inventory Management System also helps track theft of retail merchandise, providing valuable information about store profits and the need for theft-prevention systems. The automated Inventory Management System works by scanning a barcode either on the item. A barcode scanner is used to read the barcode, and the information encoded by the barcode is read by the machine. This information is then tracked by a central computer system.

5. RESEARCH PAPER ON INVENTORY MANAGEMENT SYSTEM [PROF.MANJUSHA TAMALE, et al,2018]

Inventory Management System is software which is helpful for businesses that operate hardware stores, where the store owner keeps the records of sales and purchases. Mismanaged inventory means disappointed customers, too much cash tied up in warehouses and slower sales. This project eliminates the paperwork, human faults, and manual delays and speeds up the process. The inventory Management System will can track sales and available inventory, telling a store

owner 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 on the area of Inventory control and generated the various required reports.

6. A STUDY OF THE INVENTORY MANAGEMENT SYSTEM OF LINAMAR INDIA PVT. LTD, PUNE [AMITY JOURNAL OF OPERATIONS MANAGEMENT,et al,2018]

An important area of the manufacturing industry. If a company fails to manage inventory, it will face failure. It is a challenge for the company to maintain fair inventory. There are various inventory management techniques available for maintaining fair inventory levels in the company. The basic objective of this paper. The aim of the study is to examine the inventory management process. The significance of this research is based on the benefits that can be obtained by identifying the issues of inventory control. The methodology used is unstructured interviews, on-site study, and annual report analysis. Inventory management is an is to study inventory management techniques used in Lina-mar India Pvt. Ltd. and find out some measures for improvement on the inventory management process of the concerned company. The present system of inventory management of the company is good. For improvement of the present inventory management system, companies should adopt other inventory management techniques.

2.3 PROBLEM STATEMENT DEFINITION

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Retailer	find the products count in the stock	it's complicated	it takes more time	tired
PS-2	Retailer	to calculate the bill for transportation purposes	it's hard	it takes a long time	uninterested
PS-3	Retailer	find the customer's review	it's hard to gather the information	I don't have enough contacts	disappointed
PS-4	Retailer	maintain the ledger	it's difficult to secure	it may be lost	afraid
PS-5	Retailer	find the high-demand products	it's difficult to calculate	it takes more time	challenging

CHAPTER - 3

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

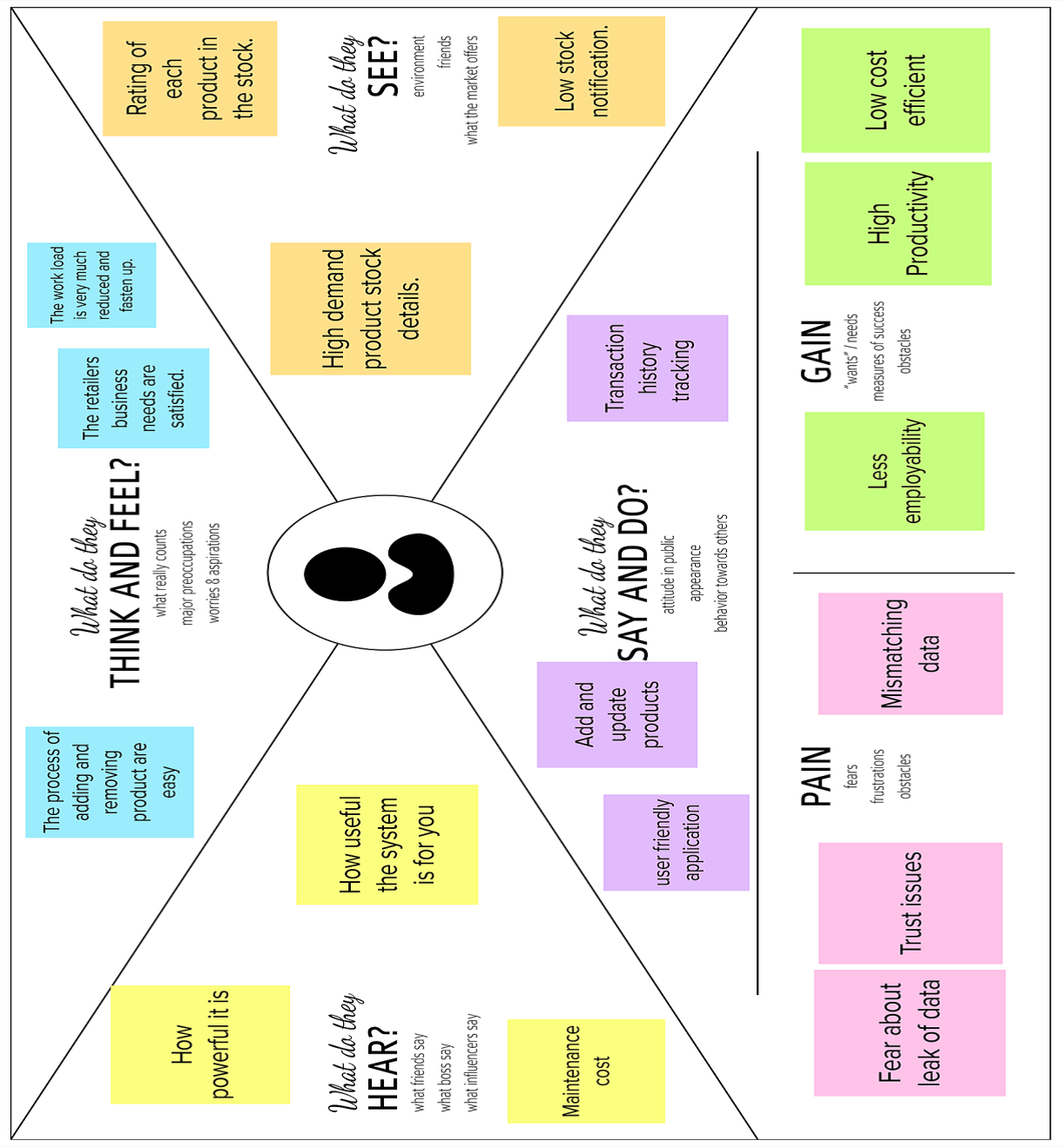


Fig 3.1 Empathy Map Canvas

3.2 IDEATION & BRAINSTORMING

Fig 3.2.1 Ideation & Brainstorming

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

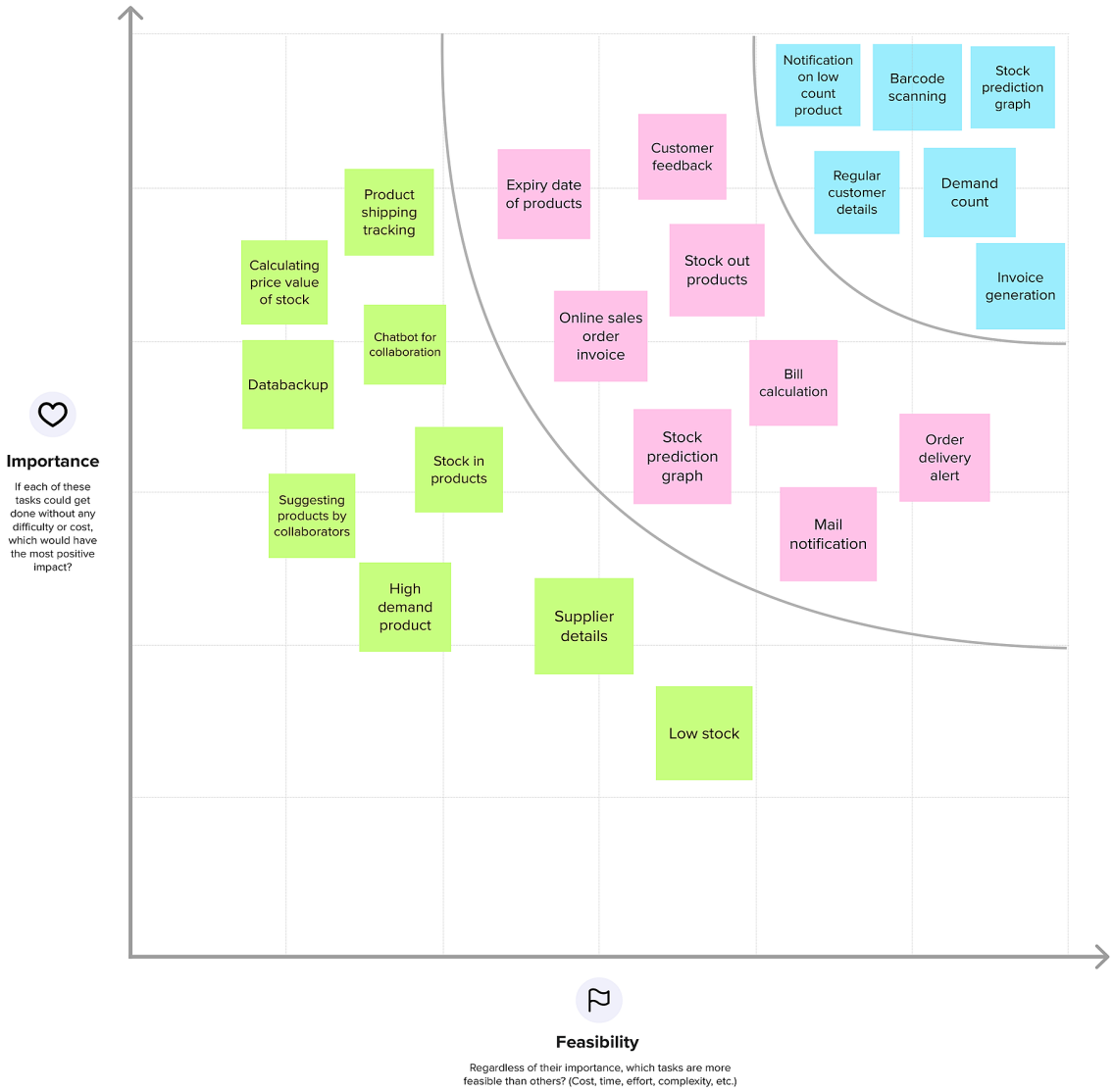


Fig 3.2.2 Ideation & Brainstorming

3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	For Retailers maintaining and securing the Ledger is difficult. Calculating and updating the product count every time needs Manpower. There is a need to count low stock and High Demand products. If any product is under low stock retailer is not aware of that information.
2.	Idea / Solution description	Instead of maintaining a ledger we create software that displays the count of all product and their information. It calculates the price value of available products in the stock. If any product is under low stock the Retailer will get an email alert.
3.	Novelty / Uniqueness	The High sold product along with customer details and their product feedback will be maintained. While selling the product with the help of the Barcode scanning option the product count will be updated.
4.	Social Impact / Customer Satisfaction	Customer details and feedback will be collected and maintained so that the retailers view the details of the product in the stock. Without any Manpower the stock details will be tracked and maintained easily.
5.	Business Model (Revenue Model)	With the help of high-demand product details, retailers can order more supply. We provide data backup using cloud storage

3.4 PROBLEM SOLUTION FIT

Fig 3.4 Problem Solution Fit

CHAPTER 4

RECUIREMENT ANALYSIS

4.1 FUNCTIONAL RECUIREMENTS

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through User Form Registration through Gmail/ Phone Number Registration through Facebook
FR-2	User Confirmation	Confirmation via Email, Phone Number
FR-3	Adding,updating,deleting the products	Add or update or delete products using the options given in dashboard
FR-4	High demand products	User can view the high demand products by using the previous sales details
FR-5	Barcode scanning	User can check or update products by scanning the Barcode
FR-6	Invoice generation	User can generate invoice by their email for future reference

4.2 NON-FUNCTIONAL RECUIREMENTS

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Stocks are managed and tracked easily
NFR-2	**Security**	Two step verification
NFR-3	**Reliability**	Huge number of stock products can be easily calculated
NFR-4	**Performance**	The website's loading time should be less than 5 seconds
NFR-5	**Availability**	All kind of retailers and wholesale business man can use .
NFR-6	**Scalability**	Starting from retailer shop owners to department store owners can use our product

CHAPTER 5

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

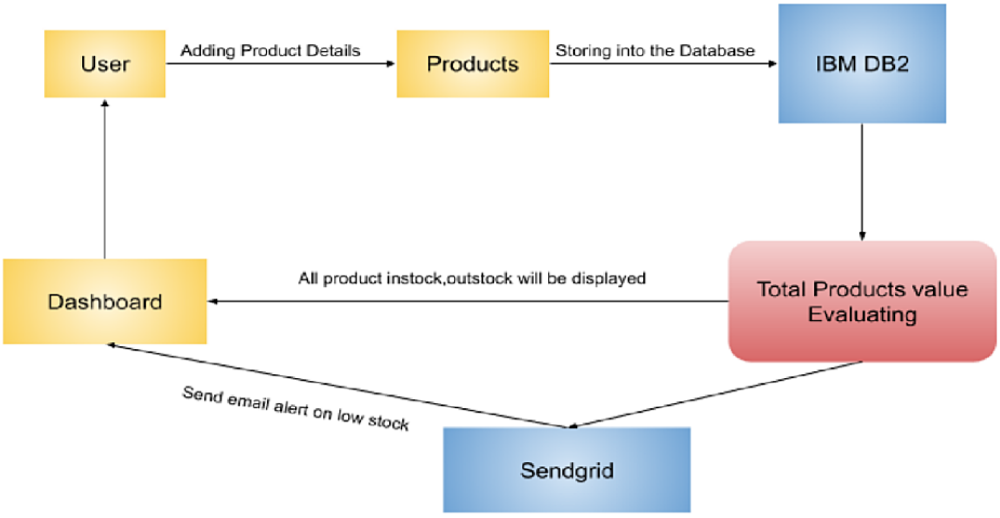


Fig 5.1 Data Flow Diagrams

5.2.1 SOLUTION ARCHITECTURE

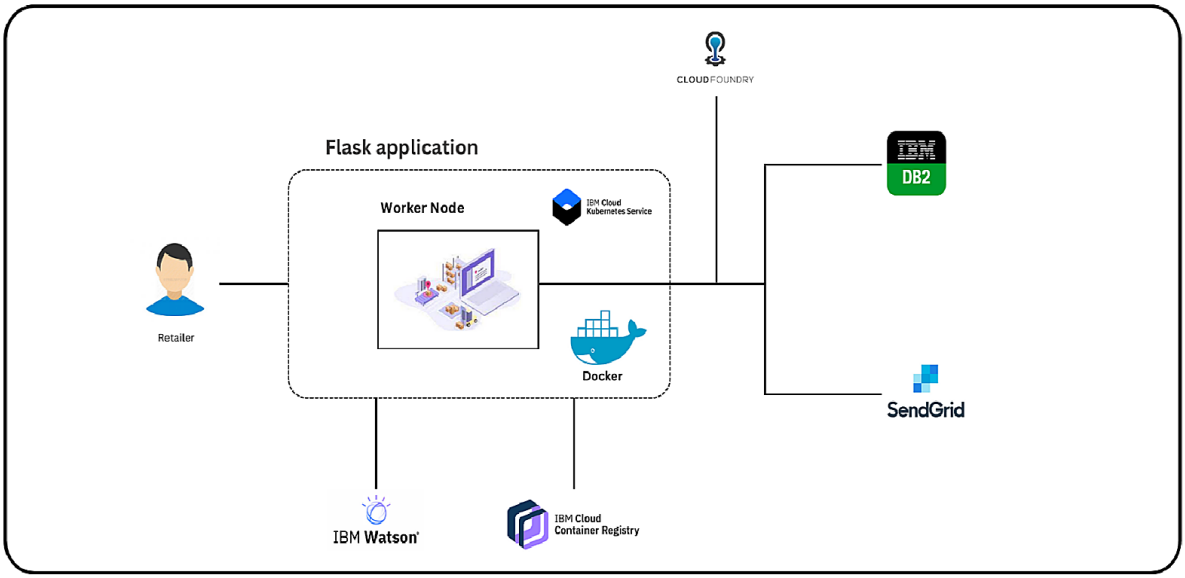


Fig 5.2.1 Solution Architecture

5.2.2 TECHNICAL ARCHITECTURE

Technical Architecture:

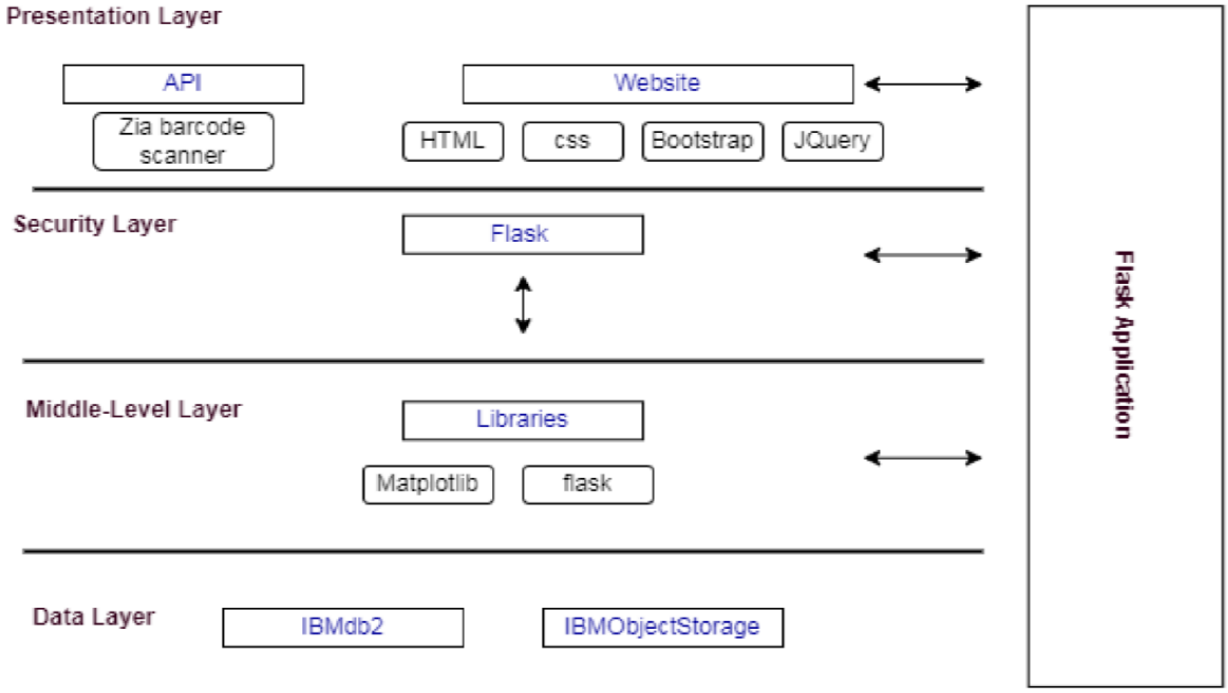


Table-1 : Components & Technologies:

S.No	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 Matplot Library
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 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

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Retailer(Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I will be redirected to login page	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can verify the OTP number	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my account / dashboard	High	Sprint-1
	Dashboard	USN-6	As a user,I can update stock in & out count details	Updation can be made through barcode scanning	High	Sprint -2
	Dashboard	USN-7	As a user,I can check the low stock details through alert message	Alert message can be received by registered mail	High	Sprint -1
		USN-8	As a user,I can check the total product details	I can view the value of total products in the stock	Medium	Sprint -2
		USN-9	As a user,I can check the high demand product details	I can update sales details of the products	High	Sprint -2
		USN-10	As a user,I can generate the invoice details	I can add incoming stock details	High	Sprint -1

CHAPTER 6

PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

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 by entering my email Id, password, and confirming my password.	5	High	Vimalraj D
Sprint-1		USN-2	As a user, I will receive a confirmation email once I have registered for the application	4	Medium	Praveenkumar S
Sprint-1		USN-3	As a user, I can register for the application through Facebook	3	Low	Veerammal S
Sprint-1		USN-4	As a user, I can register for the application through a Google account	4	Medium	Priyaranjan K
Sprint-1	Login	USN-5	As a user, I can log in into the application by entering email id & password	4	Medium	Venkatesh Kumar B
Sprint-2		USN-6	As a user, I can log in into the application by using google account.	5	High	Priyaranjan K
Sprint-2		USN-7	As a user, I can log in to the application by using Facebook	3	Low	Vimalraj D

Sprint-2		USN-8	As a User, I can view the In stock count details.	5	High	Venkatesh Kumar B
Sprint-2		USN-9	As a User, I can view the low stock count details.	5	High	Veerammal S
Sprint-2		USN-10	As a User, I can view the high-demand product count details.	4	Medium	Praveenkumar S
Sprint-3	Dashboard	USN-11	As a User, I can view recently added product detail.	4	Medium	Priyaranjan K
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-3		USN-12	As a User, I can view recently sold product information.	3	Medium	Vimalraj D
Sprint-3		USN-13	As a User, I can view recently purchased customer detail.	5	High	Venkatesh Kumar B
Sprint-3	Products	USN-14	As a User , I can able to see all product name, price and quantity	4	Medium	Praveenkumar S
Sprint-3	Peoples	USN-15	As a User , I can able to see all users and customer details	4	Medium	Veerammal S
Sprint-4	Reports	USN-16	As a User, I can see all product count in form of graph	5	High	Praveenkumar S
Sprint-4	Sendgrid Integration	USN-17	As a User, I can send the email alert on low product	5	High	Priyaranjan K
Sprint-4	IBM Watson Integration	USN-18	As a User, I can ask any queries to developer of application	3	Low	Veerammal S
Sprint-4	Documentation	USN-19	As a user I can refer the documentation for features of application	4	Medium	Venkatesh Kumar B
Sprint-4		USN-20	As a user I can view a demo video of website	3	Low	Vimalraj D

6.2 SPRINT DELIVERY SCHEDULE

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	20	6 Days	24 Oct 2022	29 Oct 2022		
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		

VELOCITY:

Sprint duration : 6 days

Velocity of the team : 20

Team's average velocity

AV = velocity / sprint duration= 3.33 points per day

6.3 REPORTS FROM JIRA

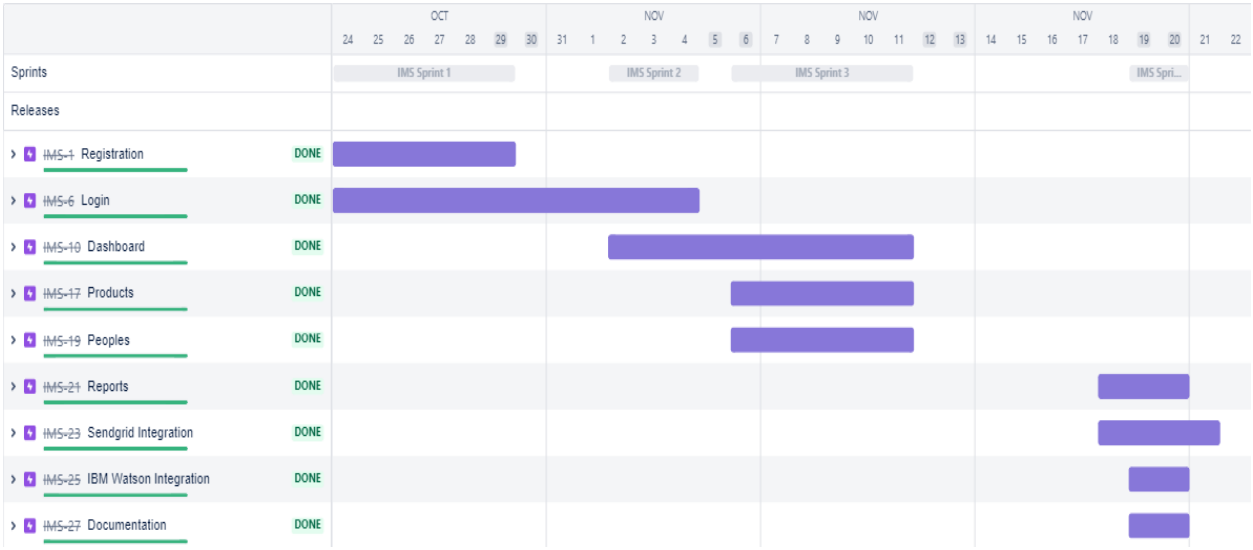


Fig 6.3.1 Reports From JIRA

SPRINT 1

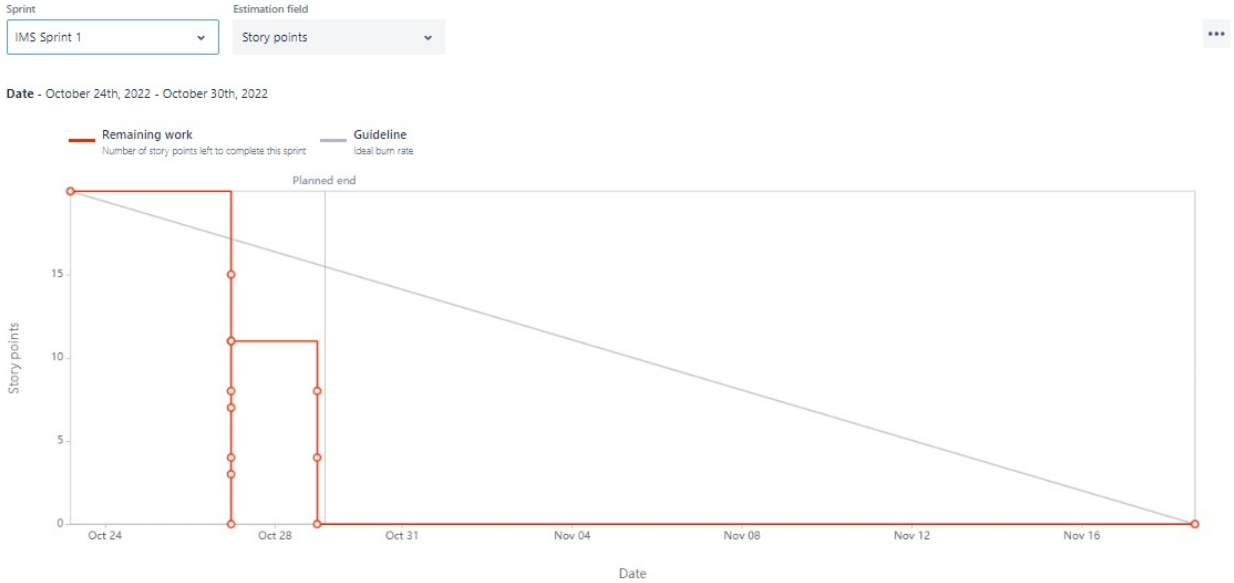


Fig 6.3.2 Reports From JIRA

SPRINT 2

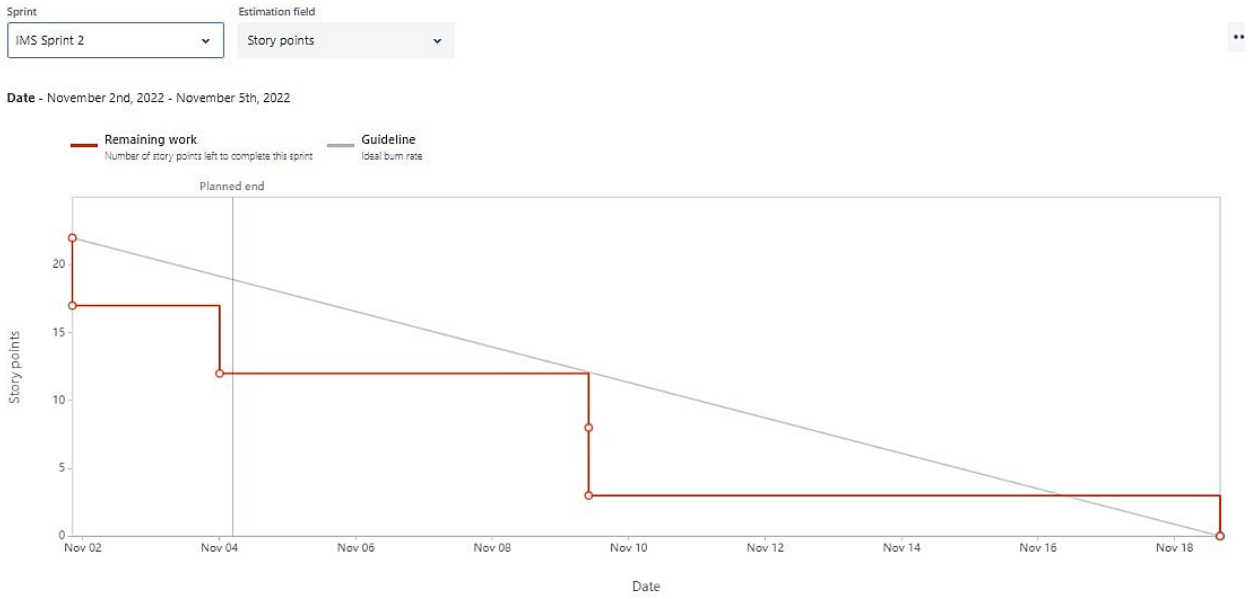


Fig 6.3.3 Reports From JIRA

SPRINT 3

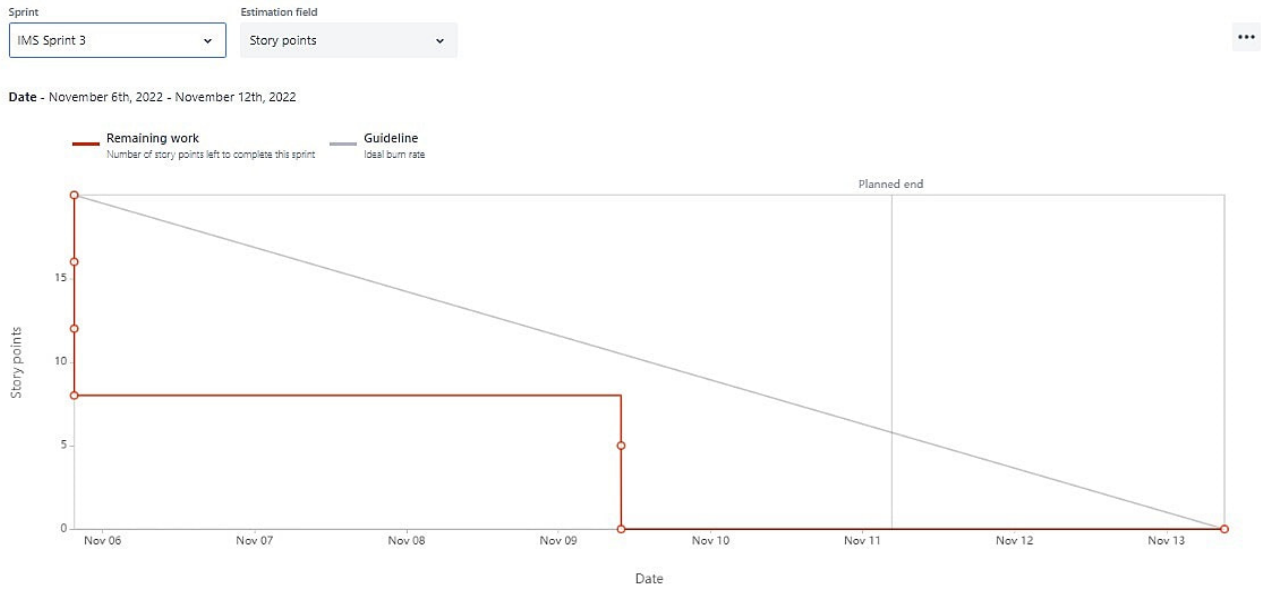


Fig 6.3.5 Reports From JIRA

SPRINT 4

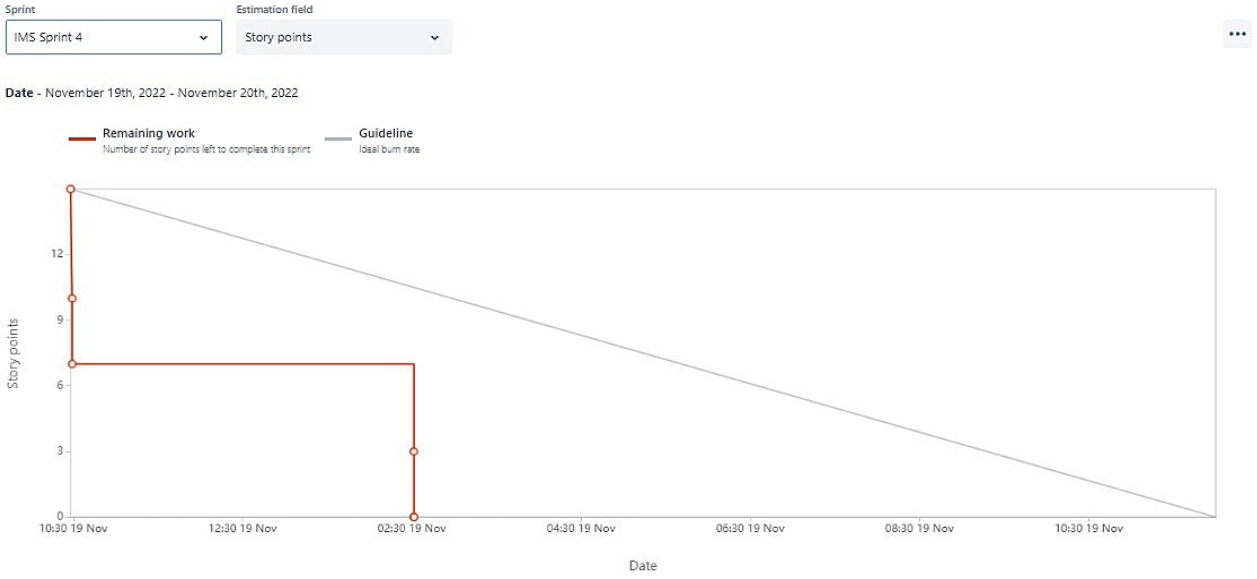


Fig 6.3.6 Reports From JIRA

VELOCITY REPORT

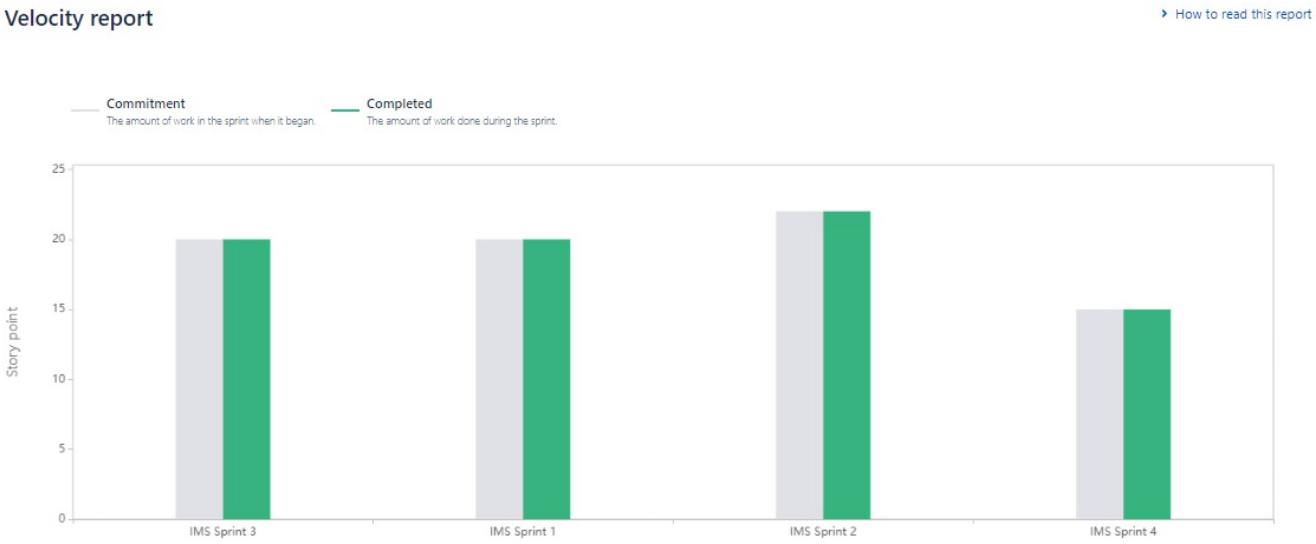


Fig 6.3.7 Reports From JIRA

CHAPTER 7

CODING & SOLUTIONING

7.1 FEATURE 1 - STOCK VIRTUALIZATION

In dashboard, displaying piechart that shows the percentage of each product available in the stock and combo chart that shows daily sales of products with percentage & count

Fig 7.1 Stock Virtualization

7.2 FEATURE 2 - LOW STOCK ALERT

If the product count is low, the customer will get the low stock alert by the email with product id, name & count

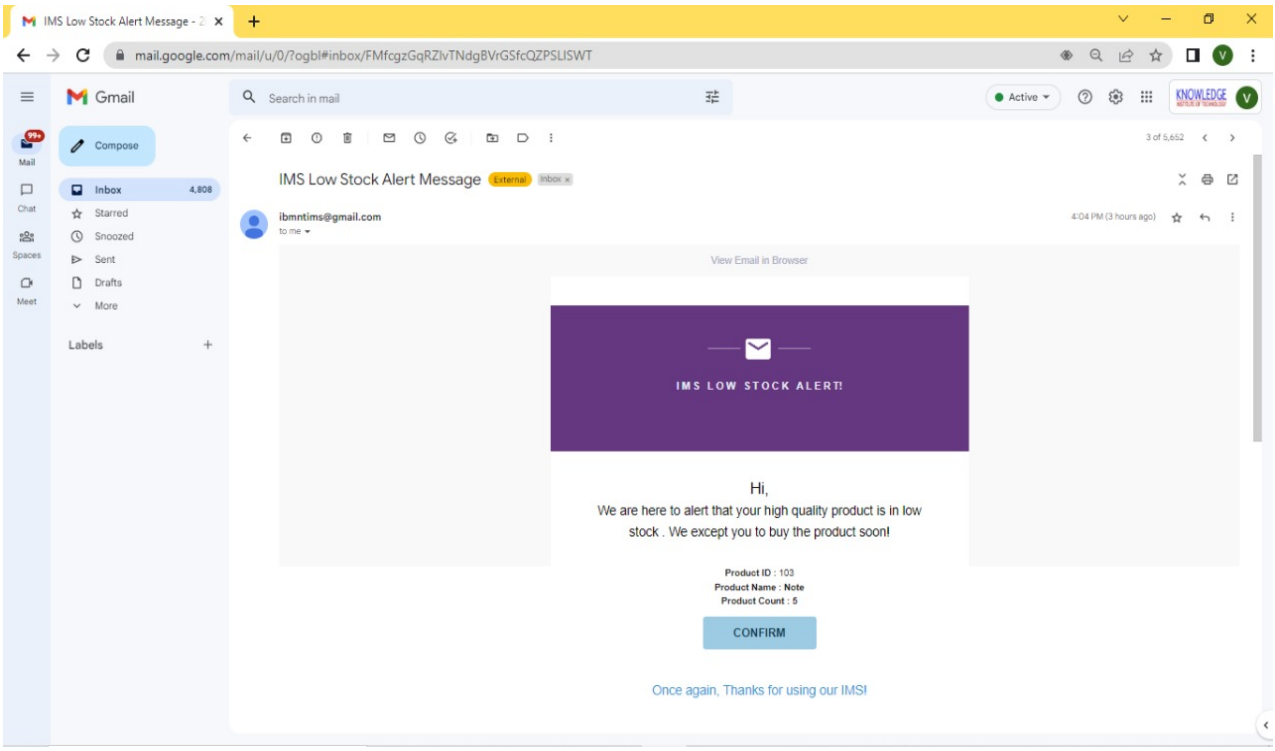


Fig 7.2 Low Stock alert

CHAPTER 8

TESTING

8.1 TEST CASES

			NFT - Risk Assessment						
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Voluem Changes	Risk Score	Justification
1	Inventory Manageme	New	Moderate	No Changes	Low		>5 to 10%	GREEN	As we have seen the changes
2	Inventory Management System For Retailers		Low	No Changes	Low		>5 to 10%	GREEN	As we have seen the changes
3	Inventory Management System For Retailers		High	No Changes	High		>50 to 70%	RED	As we have seen the changes
4	Inventory Management System For Retailers		Moderate	No Changes	Moderate		>30 to 50%	ORANGE	As we have seen the changes

NFT - Detailed Test Plan				
S.No	Project Overview	NFT Test approach	umptions/Dependencies/R	Approvals/SignOff
1	Inventory Management System For Retailers	Scalability	Moderate	Praveen Kumar Senthilkumar

End Of Test Report								
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
	Inventory Management System For Retailers		Yes	Good		Increase the number of pods	Closed	Praveen Kumar Senthilkumar

8.2 USER ACCEPTANCE TESTING

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
By Design	8	3	2	2	15
Duplicate	1	0	1	0	2
External	1	2	0	1	4
Fixed	11	2	3	3	19
Not Reproduced	0	3	0	0	3
Skipped	0	1	0	0	1
Won't Fix	0	1	0	0	1
Totals	21	12	6	6	45

Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	23	0	0	23
Security	1	0	0	1
Outsource Shipping	2	0	0	2
Exception Reporting	4	0	0	4
Final Report Output	2	0	0	2
Version Control	4	0	0	4

Testcases Report

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
LoginPage_TC_001	Functional	Home Page	Verify user is able to see the		1.Enter URL and click go	169.51.207.132:30133	Login/Signup popup should display	Working as	Pass				Praveenkumar Senthilkumar
LoginPage_TC_002	UI	Home Page	Verify the UI elements in		1.Enter URL and click go	169.51.207.132:30133	Application should show below UI	Working as	Pass				Veerammal S
LoginPage_TC_003	Functional	Home page	Verify user is able to log into		1.Enter URL(169.51.207.132:30133) and click go	Username: 2kf3cse117@kiot.ac.in	User should navigate to user	Working as expecte	Pass				VimalrajD
LoginPage_TC_004	Functional	Login page	Verify user is able to log into		1.Enter URL(169.51.207.132:30133) and click go	Username: 2kf3cse117@kiot.ac.in	Application should show 'Incorrect	Working as	Pass				VimalrajD
LoginPage_TC_004	Functional	Login page	Verify user is able to log into		1.Enter URL(169.51.207.132:30133) and click go	Username: chalam@gmail.com	Application should show 'Incorrect	Working as	Pass				VimalrajD
LoginPage_TC_005	Functional	Login page	Verify user is able to log into		1.Enter URL(169.51.207.132:30133) and click go	Username: chalam	Application should show 'Incorrect	Working as	Pass				Priyaranjan K
Dashboard_TC_006	Functional	Dashboard	Verify if the stock data is visualized correctly		1.Enter URL(169.51.207.132:30133) and click go2. click on Dashboard Page button after logged in		Application should show visualization	Working as expecte	Pass				Priyaranjan K
ProductPage_TC_007	Functional	Product Page	Verify all the stored stock details are displayed		1.Enter URL(169.51.207.132:30133) and click go2. click on Product page		Application should not show irrelavant	Working as expecte	Pass				Venkatesh Kumar B
NewProductPage_TC_00	Functional	New Product Page	verify all the stock data can benewly entered correctly		1. Enter URL(169.51.207.132:30133) and click go 2. click on NewProduct Page icon3. fill the form with		Application should receive all the data	Working as expecte	Pass				Praveenkumar Senthilkumar
NewProductPage_TC_	Functional	New Product	Verify user is able to		Fill the form		Application should allow editing the	Working as	Pass				Veerammal S
PeoplePage_TC_010	Functional	Product Page	Verify all the customer		click on the people page		Application should fetch all the	Working as	Pass				Veerammal S
SalesPage_TC_011	Functional	Sales Page	Verify all the sales details are		click on the sales page		Application should fetch all the sales	Working as	Pass				VimalrajD

CHAPTER 9

RESULTS

9.1 PERFORMANCE METRICS

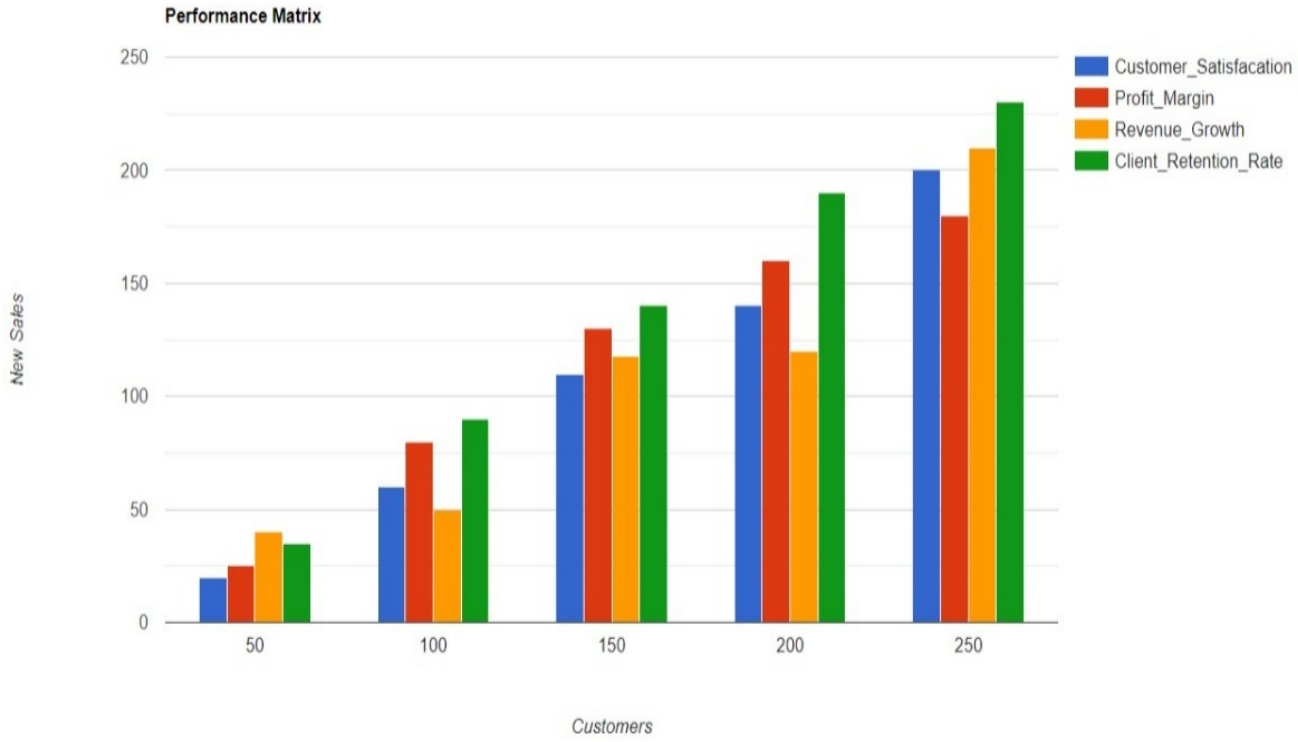


Fig 9.1 Performance Metrics

CHAPTER 10

ADVANTAGES & DISADVANTAGES

10.1 ADVANTAGES

1. Inventory Management System that helps to maintain the right amount of stocks
2. It leads to a more organized warehouse
3. It saves time and money
4. A well-structured inventory management system leads to improved customer retention
5. Improves efficiency and productivity
6. Reduction in holding costs
7. Increased information transparency

10.2 DISADVANTAGES

1. Disadvantage of inventory management is a lack of personal touch.
2. Increased space is need to hold the inventory
3. The control of inventory is complex because of the many functions it performs. It should thus be viewed as a shared responsibility
4. Holding inventory can result to a greater risk of loss to devaluation (changes in price)

CHAPTER 11

CONCLUSION

An effective inventory management system helps to reduce stock-related costs such as warehousing, carrying, and ordering costs.

The system is designed to reduce human labor and efficiently maintaining the stock.It provides flexible and powerful reports regarding items. purchase, sales and ledger.We hope that it will help people to reduce both time and money.

The goal of inventory management is to understand stock levels and stock's location in warehouses. Inventory management software tracks the flow of products from supplier through the production process to the customer.

CHAPTER 12

FUTURE SCOPE

In the future we will improve our application to the next level by adding amazing features.This inventory management system is for retailers & in future large scale industries can manage their stock by using this system.We can automaticaly update the stock through the online web commerce stock by this software in future.

The scope of an inventory system can cover many needs, including valuing the inventory, measuring the change in inventory and planning for future inventory levels.

CHAPTER 13

APPENDIX

13.1 SOURCE CODE

app.py

```
from flask import Flask
from flask_session import Session
from authlib.integrations.flask_client import OAuth
from google_auth_oauthlib.flow import Flow
import ibm_db
import os
import pathlib
from flask import render_template, request, redirect, url_for, session
from pip._vendor import cachecontrol
from google.oauth2 import id_token
from random import randint
import google.auth.transport.requests
import requests
import ibm_db
import json

import smtplib
import ssl
```

```
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText


app = Flask(__name__)
app.secret_key = b'\x84\xda1\x83@DUX\xf29%}Z<v\xdd'

app.config["SESSION_PERMANENT"] = False
app.config['SESSION_TYPE'] = 'filesystem'
Session(app)

os.environ['OAUTHLIB_INSECURE_TRANSPORT'] = '1'

oauth = OAuth(app)
app.config['GOOGLE_CLIENT_ID'] = '91863464450-
chmqoncg99unjfcet63ebab98fjlu8eg.apps.googleusercontent.com'
app.config['GOOGLE_CLIENT_SECRET'] = 'GOCSPX-d9l-
MgZNXSRg1m_RgdBSTxoepKs1'
app.config['REDIRECT_URI'] = '/gentry/auth'
client_secrets_file = os.path.join(pathlib.Path(__file__).parent, 'client_secret.json')

flow = Flow.from_client_secrets_file(
    client_secrets_file=client_secrets_file,
    scopes=['https://www.googleapis.com/auth/userinfo.profile',
    'https://www.googleapis.com/auth/userinfo.email', 'openid'],
```

```
    redirect_uri='http://127.0.0.1:5000/gentry/auth'
)

conn = None

## DB2 Database connectivity
try:
    conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=2d46b6b4-cbf6-
40eb-bbce-
6251e6ba0300.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32328;S
ECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=nwn0828
7;PWD=zQNzuapvAUXeDLhx;PROTOCOL=TCPIP",",")
    print("Successfully connected with db2")
except:
    print("Unable to connect: ", ibm_db.conn_errormsg())



config = app.config
GOOGLE_CLIENT_ID = config.get('GOOGLE_CLIENT_ID')
GOOGLE_CLIENT_SECRET = config.get('GOOGLE_CLIENT_SECRET')
REDIRECT_URI = config.get('REDIRECT_URI')
```

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

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



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

    userid = session['userid']

    products_stock = []
    sql = "SELECT * FROM product WHERE userid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        products_stock.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
```

```
products_count = 0
products_value = 0
product_price = {}

for product in products_stock:
    products_count += product["STOCKCOUNT"]
    products_value += (product["STOCKCOUNT"] * product["PRICE"])
    product_price[product["PRODID"]] = product["PRICE"]



salesdata = []
sql = "SELECT * FROM sales WHERE userid = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, userid)
ibm_db.execute(stmt)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    salesdata.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

sales_count = 0
sales_value = 0
```

```
    for sales in salesdata:
        if sales["PRODID"] in product_price.keys():
            sales_count += sales["UNIT"]
            sales_value += (sales["UNIT"] * product_price[sales["PRODID"]])


    return render_template("dashboard.html",
products_stock=json.dumps(products_stock), salesdata=json.dumps(salesdata,
default=str), products_count=products_count, products_value=products_value,
sales_count=sales_count, sales_value=sales_value)


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

    userid = session['userid']

    productlist = []
    sql = "SELECT * FROM product WHERE userid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
```

```
        productlist.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if productlist:
        return render_template("products.html", productlist=productlist)
    else:
        return render_template("products.html", productlist=[])

@app.route('/newproduct')
def newproduct():
    product = {
        'USERID': '',
        'PRODID': '',
        'PRODNAME': '',
        'CATEGORY': '',
        'BRAND': '',
        'PRICE': '',
        'STOCKCOUNT': ''
    }
    return render_template('productform.html', product= product)

@app.route('/people')
def peoples():
```

```
    userid = session['userid']


    peoplelist = []
    sql = "SELECT * FROM people WHERE userid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        peoplelist.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    return render_template("people.html", peoplelist=peoplelist)



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

    userid = session['userid']


    salelist = []
    sql = "SELECT * FROM sales WHERE userid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm_db.execute(stmt)
```

```
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        salelist.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    return render_template("sales.html", salelist=salelist)



@app.errorhandler(404)
def page_not_found(error):
    # status code of that response
    return render_template('page_not_found.html'), 404



@app.route("/adduser", methods=["POST"])
def adduser():
    username = request.form.get("username")
    userid = request.form.get("userid")
    password = request.form.get("password")

    sql = "SELECT * FROM user WHERE userid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm_db.execute(stmt)
```

```
    account = ibm_db.fetch_assoc(stmt)


    if account:
        return render_template('entry.html', imsg="You are already a member, please login using your details")
    else:
        insert_sql = "INSERT INTO user VALUES (?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, username)
        ibm_db.bind_param(prep_stmt, 2, userid)
        ibm_db.bind_param(prep_stmt, 3, password)
        ibm_db.execute(prep_stmt)
        return render_template('entry.html', smsg="You are Successfully Registered with IMS, please login using your details")



@app.route("/login", methods=["POST"])
def login():
    userid = request.form.get("userid")
    password = request.form.get("password")
    sql = "SELECT * FROM user WHERE userid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm_db.execute(stmt)
```

```
    account = ibm_db.fetch_assoc(stmt)
    if not account:
        return render_template('entry.html', wmsg="You are not yet registered, please sign up using your details")
    else:
        if (password == account['PASSWORD']):
            session['username'] = account['USERNAME']
            session['userid'] = userid
            return redirect(url_for('dashboard'))
        else:
            return render_template('entry.html', msg="Please enter the correct password")

# New Method for Google Authentication
@app.route("/gentry")
def gentry():
    authorization_url, state = flow.authorization_url()
    session["state"] = state
    return redirect(authorization_url)


@app.route("/gentry/auth")
def gentry_auth():
    flow.fetch_token(authorization_response=request.url)
```

```
    credentials = flow.credentials
    request_session = requests.session()
    cached_session = cachecontrol.CacheControl(request_session)
    token_request = google.auth.transport.requests.Request(
        session=cached_session)

    id_info = id_token.verify_oauth2_token(
        id_token=credentials._id_token,
        request=token_request,
        audience=GOOGLE_CLIENT_ID
    )

    userid = id_info.get('email')
    username = id_info.get('given_name')
    session['userid'] = userid
    session['username'] = username
    sql = "SELECT * FROM user WHERE userid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    if account:
        return redirect(url_for('dashboard'))
    else:
```

```
        password = "No_Password"
        insert_sql = "INSERT INTO user VALUES (?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, username)
        ibm_db.bind_param(prep_stmt, 2, userid)
        ibm_db.bind_param(prep_stmt, 3, password)
        ibm_db.execute(prep_stmt)
        return redirect(url_for('dashboard'))
# New Method end
@app.route('/sendotp', methods=['POST'])
def sendotp():
    if request.method == 'POST':
        userid = request.form.get("userid")
        sql = "SELECT * FROM user WHERE userid = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, userid)
        ibm_db.execute(stmt)
        user = ibm_db.fetch_assoc(stmt)
        if not user:
            return render_template('entry.html', wmsg="You are not Signed up IMS")
        else:
            email_from = 'ibmntims@gmail.com'
            epassword = 'hobseglbddzxypst'
            email_to = userid
```

```
            otp = randint(000000, 999999)
            email_message = MIMEMultipart()
            email_message['From'] = email_from
            email_message['To'] = email_to
            email_message['Subject'] = f'IMS OTP email'
            email_string = f'Report email - {otp}'
            # Connect to the Gmail SMTP server and Send Email
            context = ssl.create_default_context()
            with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as
server:
                server.login(email_from, epassword)
                server.sendmail(email_from, email_to, email_string)
            return render_template('verification.html', userid=userid, otp=otp)
@app.route('/verifyotp', methods=['POST'])
def verifyotp():
    if request.method == 'POST':
        otp = request.form.get('otp')
        cotp = request.form.get('cotp')
        userid = request.form.get('userid')
        if (otp == cotp):
            return render_template('changepswd.html', userid=userid)
        else:
            return redirect(url_for('recoverymail'))
```

```
@app.route('/updatepassword', methods=['POST'])
def update_password():
    if request.method == 'POST':
        userid = request.form.get('userid')
        password = request.form.get('pswd')
        sql = "UPDATE user SET password = ? WHERE userid =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, password)
        ibm_db.bind_param(stmt, 2, userid)
        ibm_db.execute(stmt)
        return render_template('entry.html', smsg="Password updated successfully")


@app.route('/addproducts', methods=["POST"])
def addproducts():
    if request.method == 'POST':
        userid = session['userid']
        prodid = request.form.get('prodid')
        prodname = request.form.get('prodname')
        category = request.form.get('category')
        brand = request.form.get('brand')
        price = request.form.get('price')
        stockcount = request.form.get('stockcount')
        sql = "SELECT * FROM product WHERE prodid =?"
        stmt = ibm_db.prepare(conn, sql)
```

```
        ibm_db.bind_param(stmt, 1, prodid)
        ibm_db.execute(stmt)
        product = ibm_db.fetch_assoc(stmt)

        if product:
            update_sql = "UPDATE product SET USERID = ?, PRODID = ?,
PRODNAME = ?, CATEGORY = ?, BRAND = ?,  PRICE = ?, STOCKCOUNT =
? WHERE prodid = ?"
            prep_stmt = ibm_db.prepare(conn, update_sql)
            ibm_db.bind_param(prep_stmt, 1, userid)
            ibm_db.bind_param(prep_stmt, 2, prodid)
            ibm_db.bind_param(prep_stmt, 3, prodname)
            ibm_db.bind_param(prep_stmt, 4, category)
            ibm_db.bind_param(prep_stmt, 5, brand)
            ibm_db.bind_param(prep_stmt, 6, price)
            ibm_db.bind_param(prep_stmt, 7, stockcount)
            ibm_db.bind_param(prep_stmt, 8, prodid)
            ibm_db.execute(prep_stmt)
        else:
            insert_sql = "INSERT INTO product VALUES (?,?,?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, userid)
            ibm_db.bind_param(prep_stmt, 2, prodid)
            ibm_db.bind_param(prep_stmt, 3, prodname)
```

```
            ibm_db.bind_param(prep_stmt, 4, category)
            ibm_db.bind_param(prep_stmt, 5, brand)
            ibm_db.bind_param(prep_stmt, 6, price)
            ibm_db.bind_param(prep_stmt, 7, stockcount)
            ibm_db.execute(prep_stmt)
    return redirect(url_for('products'))


@app.route('/addsales', methods=["POST"])
def addsales():
    if request.method == 'POST':
        userid = session["userid"]
        customername = request.form.get('customername')
        customer_email = request.form.get('customeremail')
        address = request.form.get('address')
        prodid = request.form.get('prodid')
        unit = request.form.get('unit')
        date = request.form.get('date')
        accept = request.form.get('accept')
        sql = "SELECT * FROM people WHERE customer_email =? AND userid =
?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,customer_email)
        ibm_db.bind_param(stmt,2,session['userid'])
        ibm_db.execute(stmt)
```

```
    people = ibm_db.fetch_assoc(stmt)


    if people :
        pass
    else :
        insert_sql = "INSERT INTO people VALUES (?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, userid)
        ibm_db.bind_param(prep_stmt, 2, customername)
        ibm_db.bind_param(prep_stmt, 3, customer_email)
        ibm_db.bind_param(prep_stmt, 4, address)
        ibm_db.execute(prep_stmt)


    sql = "SELECT stockcount, prodname from product WHERE prodid = ?
AND userid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,prodid)
    ibm_db.bind_param(stmt,2,session['userid'])
    ibm_db.execute(stmt)
    stockcount = ibm_db.fetch_both(stmt)
    if(int(stockcount[0]) >= int(unit)):
        insert_sql = "INSERT INTO sales VALUES (?,?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, userid)
```

```
        ibm_db.bind_param(prep_stmt, 2, prodid)
        ibm_db.bind_param(prep_stmt, 3, customer_email)
        ibm_db.bind_param(prep_stmt, 4, unit)
        ibm_db.bind_param(prep_stmt, 5, date)
        ibm_db.execute(prep_stmt)

        stockcount[0] = int(stockcount[0]) - int(unit)

        sql = "UPDATE product SET stockcount = ? WHERE prodid = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,stockcount[0])
        ibm_db.bind_param(stmt,2,prodid)
        ibm_db.execute(stmt)
        if(stockcount[0] <= 10):
            email1 = requests.get('https://raw.githubusercontent.com/Praveenkumar-
S2805/privacy/main/email.html').text
            email2 = requests.get('https://raw.githubusercontent.com/Praveenkumar-
S2805/privacy/main/email2.html').text

            email_from = 'ibmntims@gmail.com'
            epassword = 'hobseglbddzxypst'
            email_to = session['userid']
            # Create a MIMEMultipart class, and set up the From, To, Subject fields
            email_message = MIMEMultipart()
```

```
            email_message['From'] = email_from
            email_message['To'] = email_to
            email_message['Subject'] = f'IMS Low Stock Alert Message'
            alert = "<div style = \"text-align: center\"> <strong>Product ID :
</strong>" + str(prodid) + "<br/> <strong>Product Name : <strong>" +
str(stockcount[1]) + "<br/> <strong>Product Count : </strong>" +
str(stockcount[0]) + "</div>"
            # Attach the html doc defined earlier, as a MIMEText html content type
to the MIME message
            email_message.attach(MIMEText(email1, "html"))
            email_message.attach(MIMEText(alert, "html"))
            email_message.attach(MIMEText(email2, "html"))
            # Convert it as a string
            email_string = email_message.as_string()
            # Connect to the Gmail SMTP server and Send Email
            context = ssl.create_default_context()
            with smtplib.SMTP_SSL("smtp.gmail.com", 465, context=context) as
server:
                server.login(email_from, epassword)
                server.sendmail(email_from, email_to, email_string)
        return redirect(url_for('sales'))
    else:
        return redirect(url_for('sales'))
```

```
@app.route('/delete/<prodid>/<userid>')
def delete(prodid, userid):
    sql = "SELECT * FROM product WHERE prodid=? AND userid=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, prodid)
    ibm_db.bind_param(stmt, 2, userid)
    ibm_db.execute(stmt)
    product = ibm_db.fetch_row(stmt)
    if product:
        sql = "DELETE FROM product WHERE prodid=? AND userid=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, prodid)
        ibm_db.bind_param(stmt, 2, userid)
        ibm_db.execute(stmt)

    product = []
    sql = "SELECT * FROM product WHERE userid = ?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, userid)
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        product.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
```

```
    if product:
        return render_template("products.html", product=product, msg="Delete
successfully")


@app.route('/edit/<prodid>/<userid>', methods=['GET', 'POST'])
def edit(prodid, userid):
    product = []
    sql = "SELECT * FROM product WHERE prodid=? AND userid=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, prodid)
    ibm_db.bind_param(stmt, 2, userid)
    ibm_db.execute(stmt)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        product.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
    if product:
        return render_template("productform.html", product= product[0])
    return redirect(url_for('products'))


@app.route('/exit')
def exit():
    session.clear()
    session.pop('name', default=None)
```

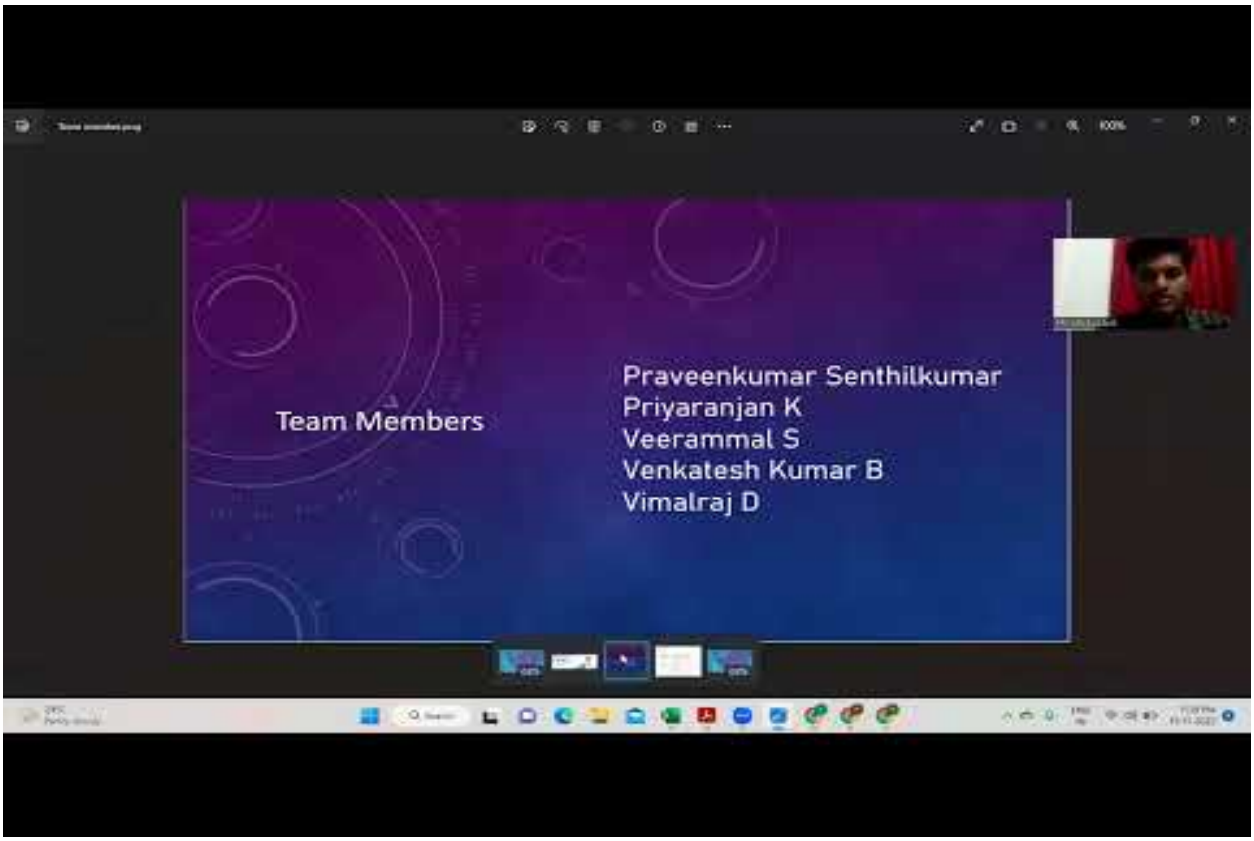
```
    session.pop('email', default=None)
    return redirect("/entry")


if __name__ == '__main__':
  app.run(host='0.0.0.0', port=5000)
```

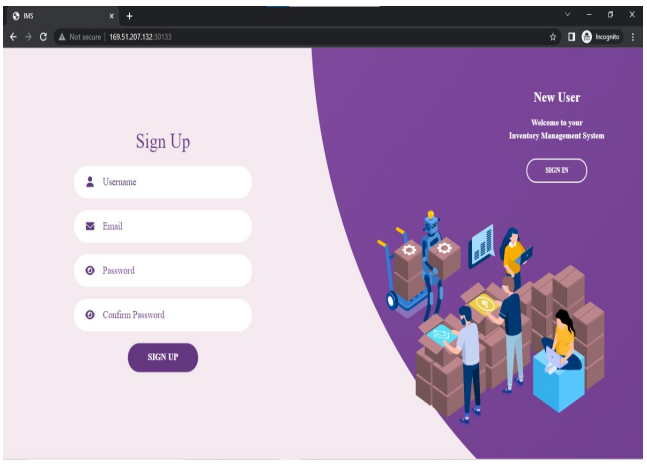
13.2 GitHub & Project Demo Link

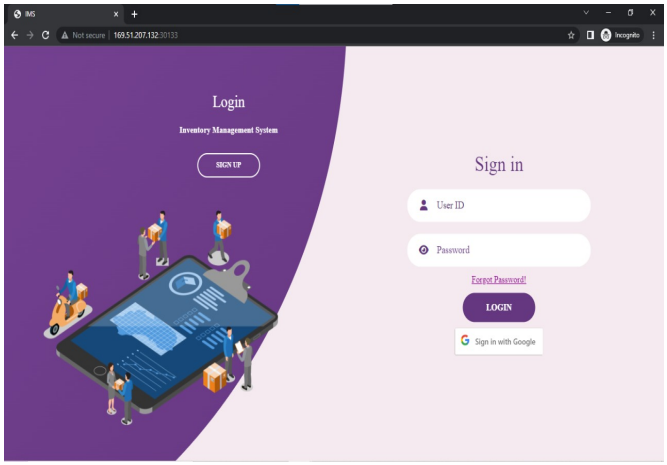
GitHub Repository Link : https://github.com/IBM-EPBL/IBM-Project-18747-1659689186

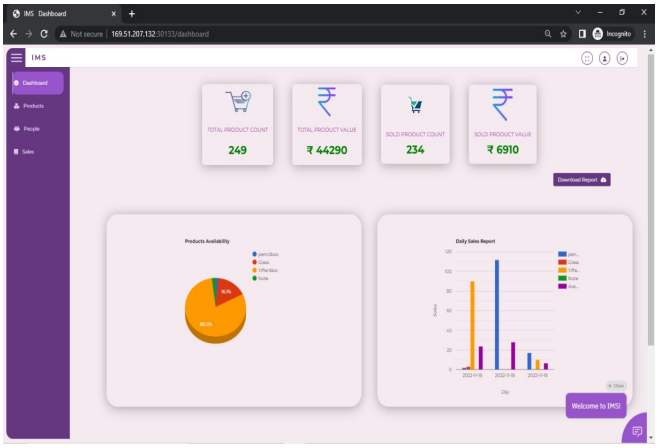
Demonstartion Video Link :

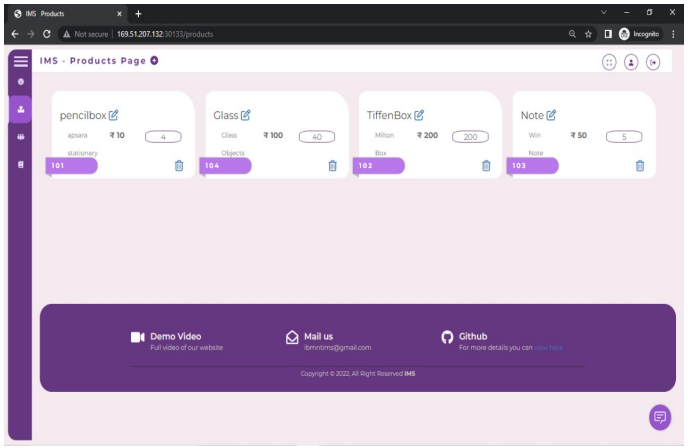


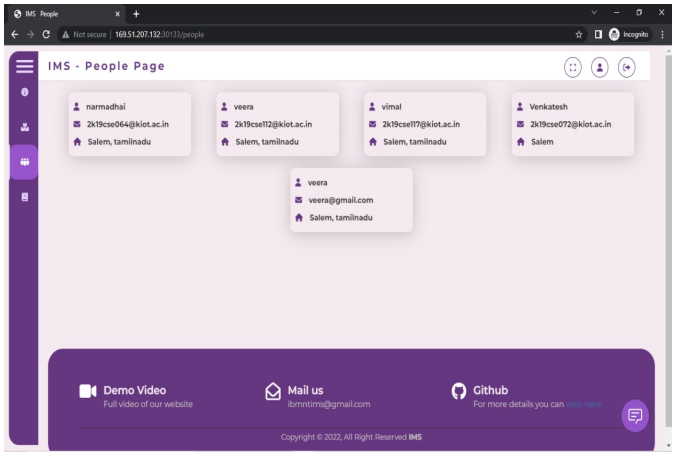
13.2 SCREENSHOTS

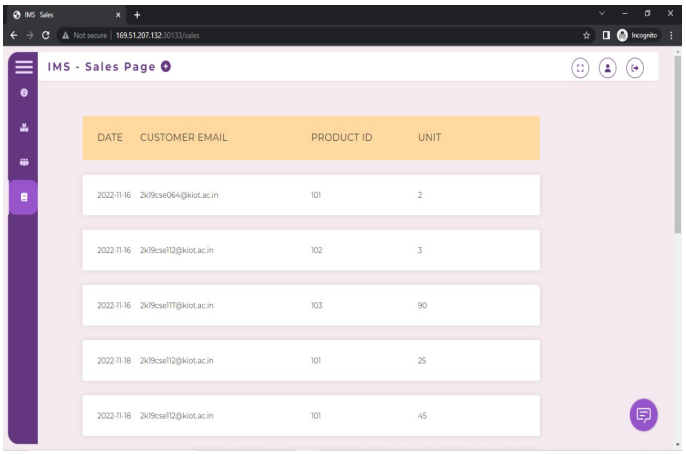


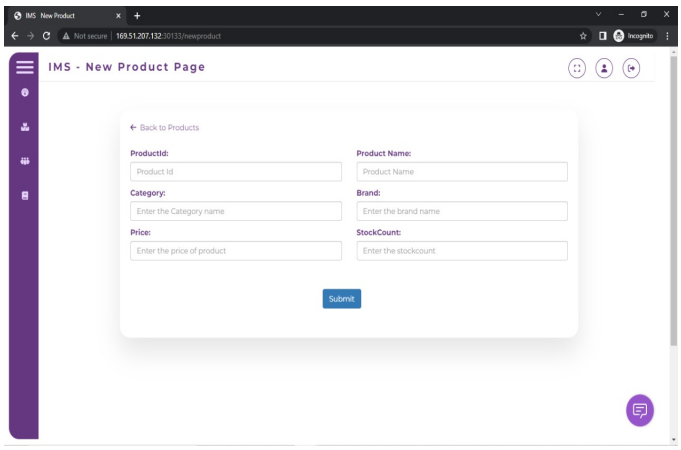


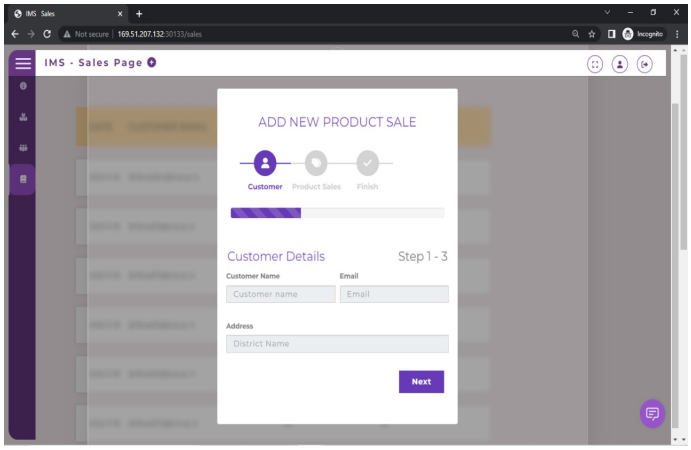


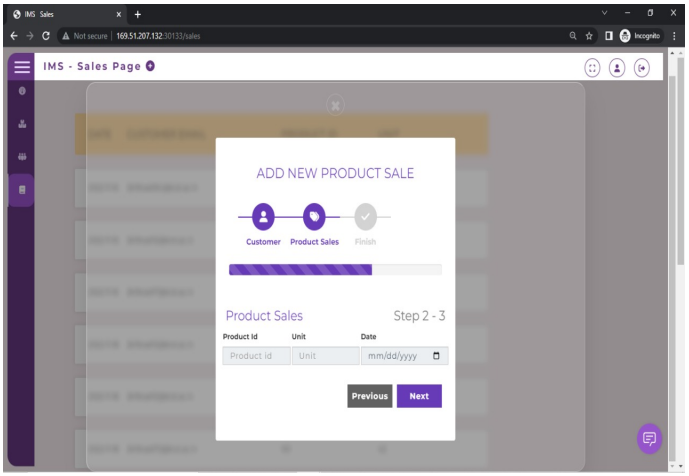


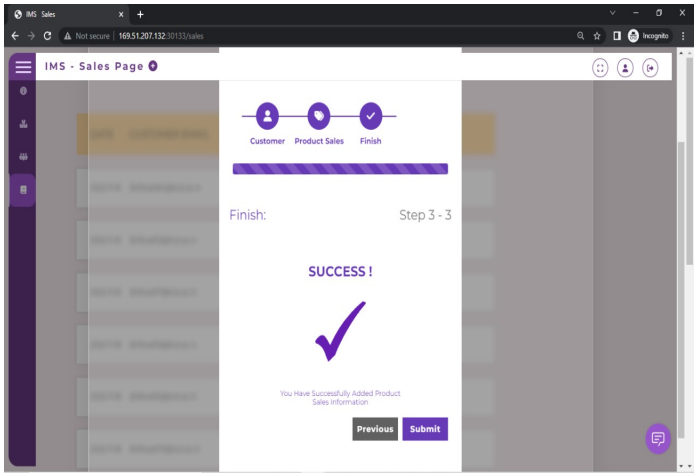


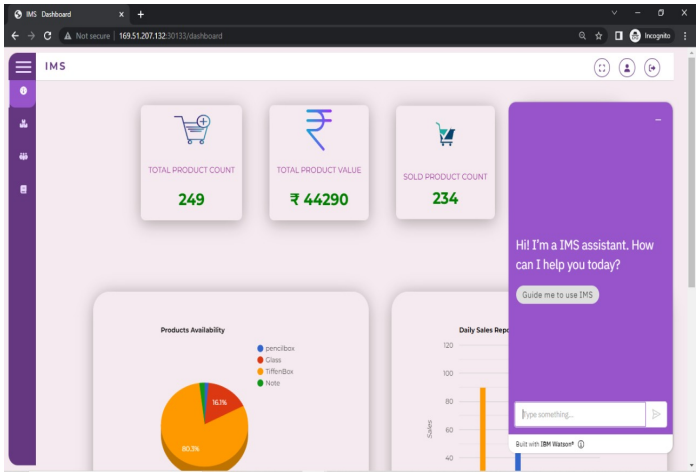


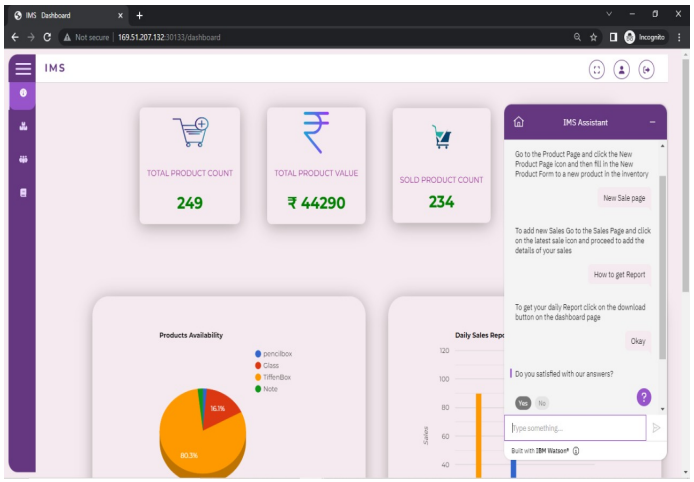

IMS - New Product Page
Back to Products
ProductId:
Product Id
Product Name:
Product Name
Category:
Enter the Category name
Brand:
Enter the brand name
Price:
Enter the price of product
StockCount:
Enter the stockcount
Submit


IMS - Sales Page
ADD NEW PRODUCT SALE
Customer
Product Sales
Finish
Customer Details
Step 1 - 3
Customer Name
Customer name
Email
Email
Address
District Name
Next


IMS - Sales Page
ADD NEW PRODUCT SALE
Customer
Product Sales
Finish
Product Sales
Step 2 - 3
Product Id
Unit
Date
Product id
Unit
mm/dd/yyyy
Previous
Next


IMS - Sales Page
Customer
Product Sales
Finish
Finish:
Step 3 - 3
SUCCESS !
You Have Successfully Added Product Sales Information
Previous
Submit


IMS
TOTAL PRODUCT COUNT
249
TOTAL PRODUCT VALUE
₹ 44290
SOLD PRODUCT COUNT
234
Hi! I'm a IMS assistant. How can I help you today?
Guide me to use IMS
Products Availability
pencilbox
Glass
TiffenBox
Note
Daily Sales Report
Sales
Type something...
Built with IBM Watson®


IMS
TOTAL PRODUCT COUNT
249
TOTAL PRODUCT VALUE
₹ 44290
SOLD PRODUCT COUNT
234
IMS Assistant
Go to the Product Page and click the New Product Page icon and then fill in the New Product Form to a new product in the Inventory
New Sale page
To add new Sales Go to the Sales Page and click on the latest sale icon and proceed to add the details of your sales
How to get Report
To get your daily Report click on the download button on the dashboard page
Okay
Do you satisfied with our answers?
Yes
No
Products Availability
pencilbox
Glass
TiffenBox
Note
Daily Sales Report
Sales
Type something...
Built with IBM Watson®

CHAPTER 14

REFERENCES

1. INVENTORY MANAGEMENT OF TATA MOTORS
[VIVEK HAMAL PROFESSOR PARUL UNIVERSITY,et al,2022]

2. INVENTORY MANAGEMENT SYSTEM WITH SALES PREDICTION LOCAL SHOPKEEPERS IN INDIA
[6TH INTERNATIONAL CONFERENCE ON ADVANCED COMPUTING AND COMMUNICATION SYSTEMS (ICACCS),et,al, 2020]

3. INVENTORY MANAGEMENT CONCERNING COCA-COLA
[ALUKA MAHESH GOUD, N. LAKSHMI DEEPTHI, et al,2019]

4. A STUDY ON INVENTORY MANAGEMENT
[B. ARCHANA, K.DIVYA, et al,2019]

5. RESEARCH PAPER ON INVENTORY MANAGEMENT SYSTEM
[PROF.MANJUSHA TAMALE,et al, 2018]

6. THE INVENTORY MANAGEMENT SYSTEM OF LINAMAR INDIA PVT. LTD, PUNE
[AMITY JOURNAL OF OPERATIONS MANAGEMENT,et al,2018]