Project Report - Inventory Management System for Retailers

Team ID: PNT2022TMID07295

INTRODUCTION

1. Project Overview
2. Purpose

2. LITERATURE SURVEY

1. Existing problem
2. References
3. Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

1. Empathy Map Canvas
2. Ideation & Brainstorming
3. Proposed Solution
4. Problem Solution fit

4. REQUIREMENT ANALYSIS

1. Functional requirement
2. Non-Functional requirements

5. PROJECT DESIGN

1. Data Flow Diagrams
2. Solution & Technical Architecture
3. User Stories

6. PROJECT PLANNING & SCHEDULING

1. Sprint Planning & Estimation
2. Sprint Delivery Schedule
3. Reports from JIRA

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

1. Feature 1
2. Feature 2
3. Database Schema (if Applicable)

8. TESTING

1. Test Cases
2. User Acceptance Testing

9. RESULTS

1. Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

1.1 Project Overview

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. The System will ask retailers to create their accounts by providing essential details. Retailers can access their accounts by logging into the application.

Once retailers successfully log in to the application they can update their inventory details, also users will be able to add new stock by submitting essential details related to the stock. They can view details of the current inventory. The System will automatically send an email alert to the retailers if there is no stock found in their accounts. So that they can order new stock.

1.2 Purpose

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

2.1 Existing problem

Inconsistent Tracking:

Using manual inventory tracking procedures across different software and spreadsheets is time-consuming, redundant and vulnerable to errors. Even small businesses can benefit from a centralized inventory tracking system that includes accounting features.

Warehouse Efficiency:

Inventory management controls at the warehouse is labor-intensive and involves several steps, including receiving and, picking, packing and shipping. The challenge is to perform all these tasks in the most efficient way possible.

Inaccurate Data:

You need to know, at any given moment, exactly what inventory you have. Gone are the days when inventory could be counted once a year with an all-hands-on-deck approach.

Changing Demand:

Customer demand is constantly shifting. Keeping too much could result in obsolete inventory you're unable to sell, while keeping too little could leave you unable to fulfill customer orders. Orderstrategies for core items, as well as technology to create and execute an inventory plan, can help compensate for changing demand.

Limited Visibility:

When your inventory is hard to identify or locate in the warehouse, it leads to incomplete, inaccurate or delayed shipments. Receiving and finding the right stock is vital to efficient warehouse operations and positive customer experiences.

Manual Documentation:

Managing inventory with paperwork and manual processes is tedious and not secure. And it doesn't easily scale across multiple warehouses with lots of stock.

Problem Stock:

Perishable and fragile stock need specialized plans for care and storage. And high-value inventory needs specific loss-prevention strategies and inventory controls.

Supply Chain Complexity:

Global supply chains shift daily, placing a burden on your inventory planning and management operations. The manufacturers and wholesale distributors that dictate when, where and how your inventory ships require flexibility and offer unpredictable lead times.

Managing Warehouse Space:

Efficiently managing space is an intimidating task. Planning and designing warehouse spaces with inventory management platforms helps you better control the timing of new stock deliveries. It can account for important factors, such as available space. Read more about the differences between warehouse management and inventory management.

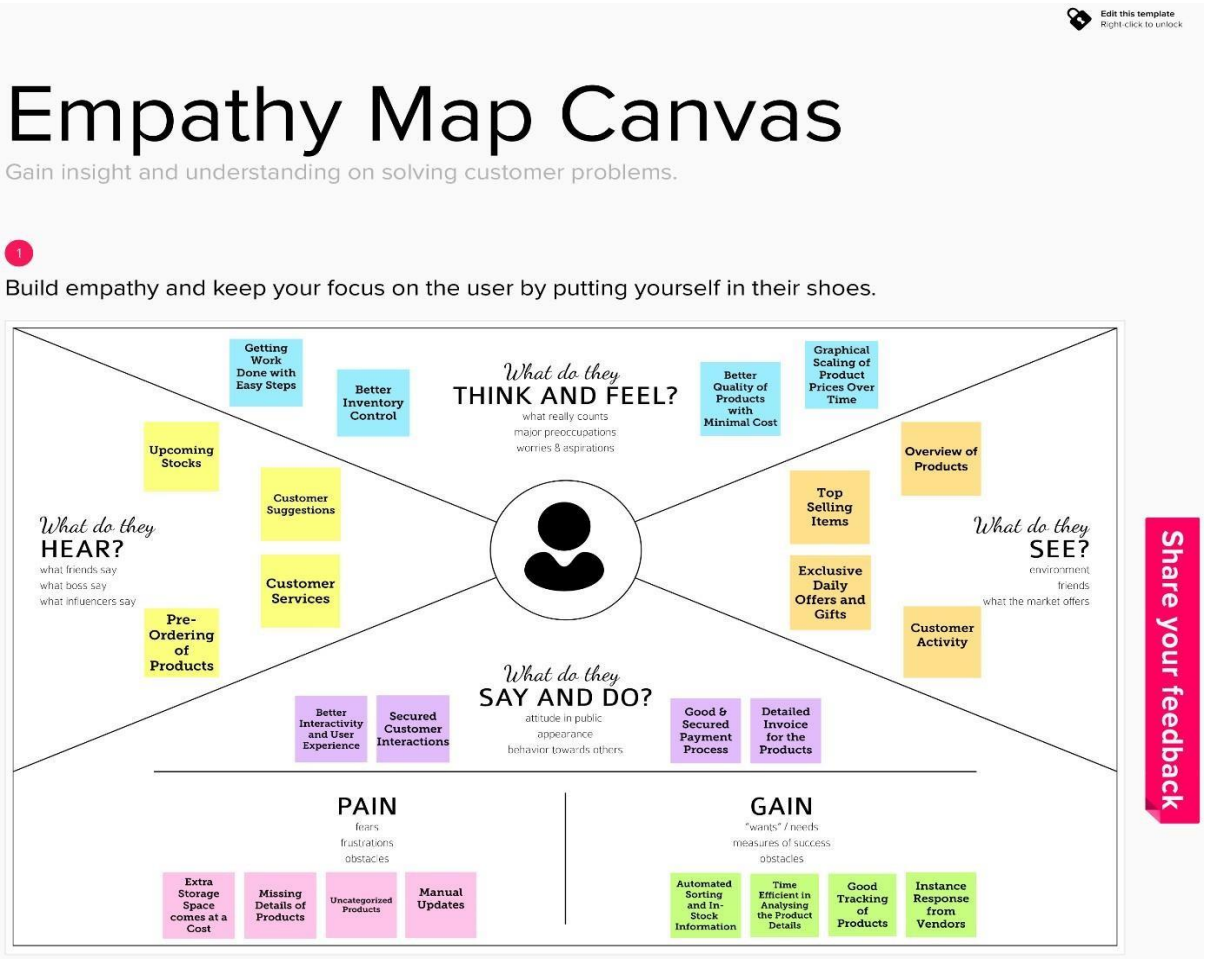
2.2 References

S. No.	Title	How does it related to my study	Used methodologies	limitation	Recommendation from this paper	Year
1.	Agricultural Inventory Management System	Support national commerce policies by annually, semi-annually, quarterly and monthly periods	Assets and their details are recorded at village and farmer level.	opportunity cost and handle the investment in inventory are more the funds are blocks up with inventory.	To make proper agricultural inventory management system to make the proper yield of products to be sold at good price.	2015 Fourth International Conference on Agro-Geoinformatics (Agro-geoinformatics)
2.	Online inventory management of packaged gases	Using advanced gadgets to manage the product or Items to be managed	By sensors the gas are monitored and the total amount is measured managed.	The proper functioning of sensors are monitored and uninterrupted power supply should be provided	Making using of available modern gadgets for the proper maintenance of the inventories	2010 IEEE Sensors Applications Symposium (SAS)
3.	A Case Study of Inventory Management System for an International Lifestyle Product Retailer in Bolivia	It is to help introducing the new type of innovations should be carried out using modern techniques, Finally the products recommended by users.	Survey from user in different categories, considering the data collected.	Traditional way of placing the items.	To include modern techniques according to the latest trend.	April 2021
4.	*A Study of Inventory Management System Case Study*	To make use of holding costs and this is frozen fund that can be lost ,Is move as a basic investment for the products	Analyzing of proper financial demands	To change only the frozen funds	To make use of available funds and not by getting any debts.	Sep 2018
5.	A Study of Inventory Management System of Linamar India	The ultimate aim of the study is to examine the inventory management process.	Use various alternative methodologies to maintain the inventory management system	Inaccurate reordering of products and order Tracking.	To increase the rate of accuracy and proper management trying alternative ways.	June 2018

2.3 Problem Statement Definition

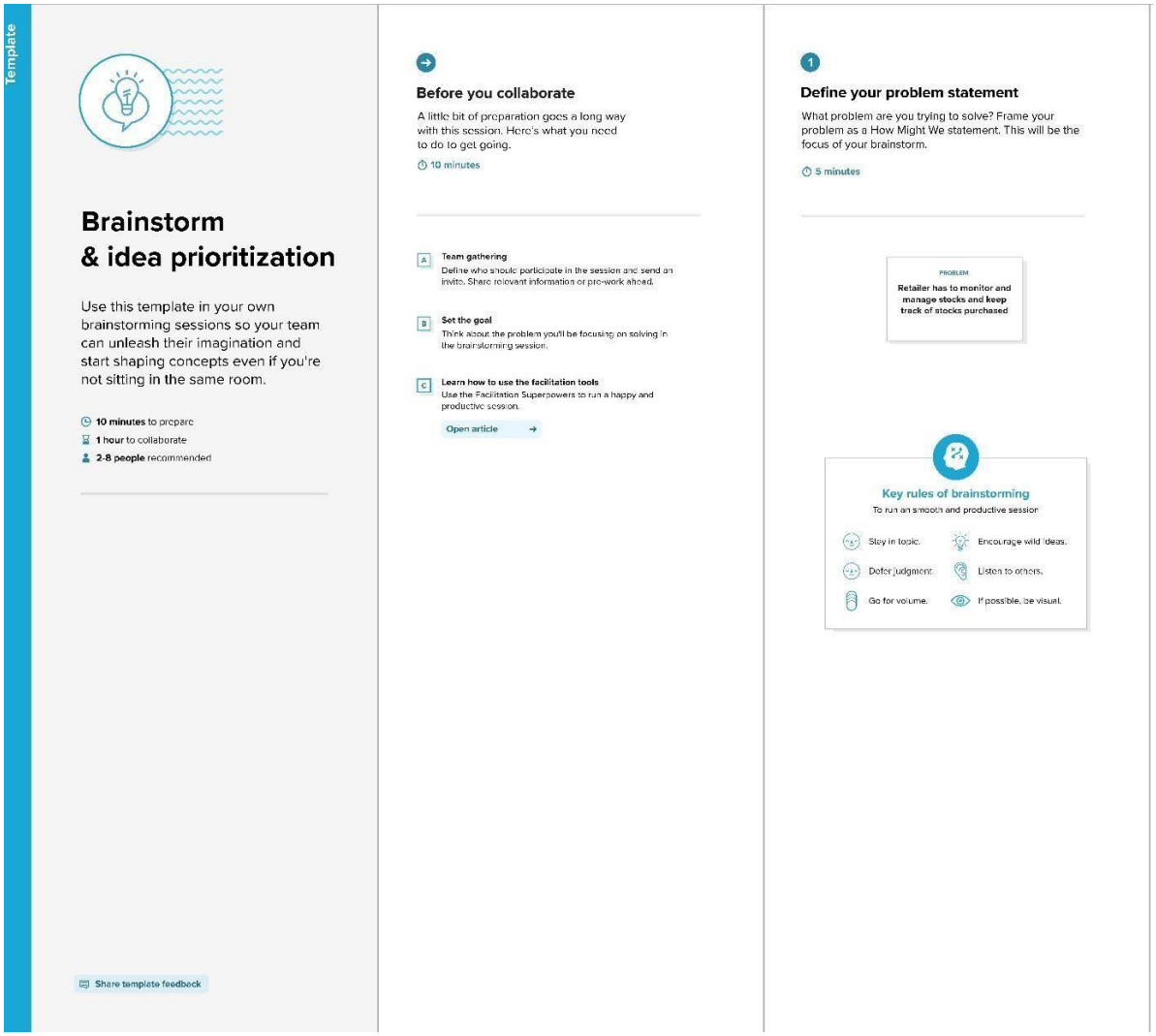
3 IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



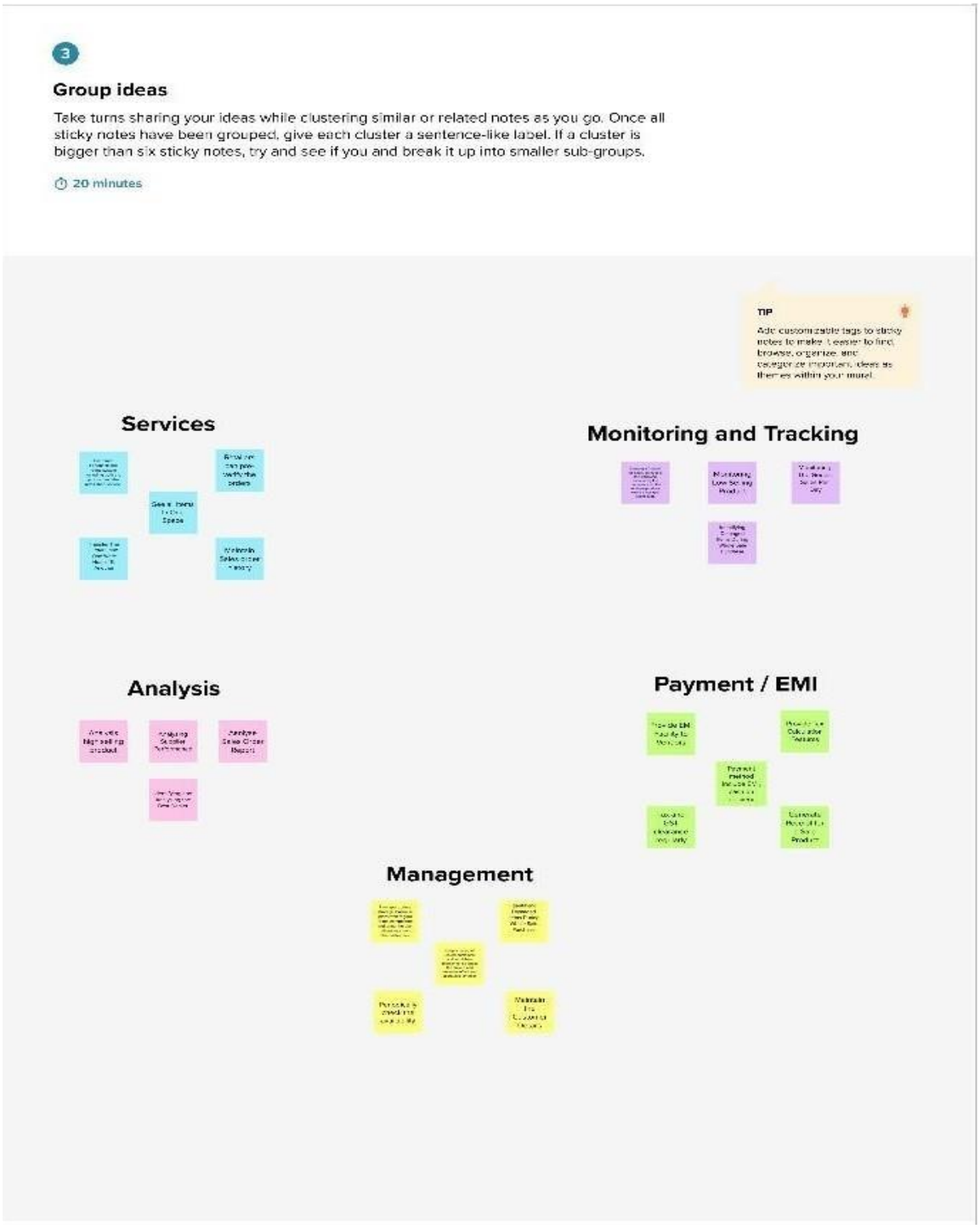
3.2 Ideation & Brainstorming

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



Step-2: Brainstorm, Idea Listing and Grouping

2 write down the ideas that comes to your mind to the problem statement



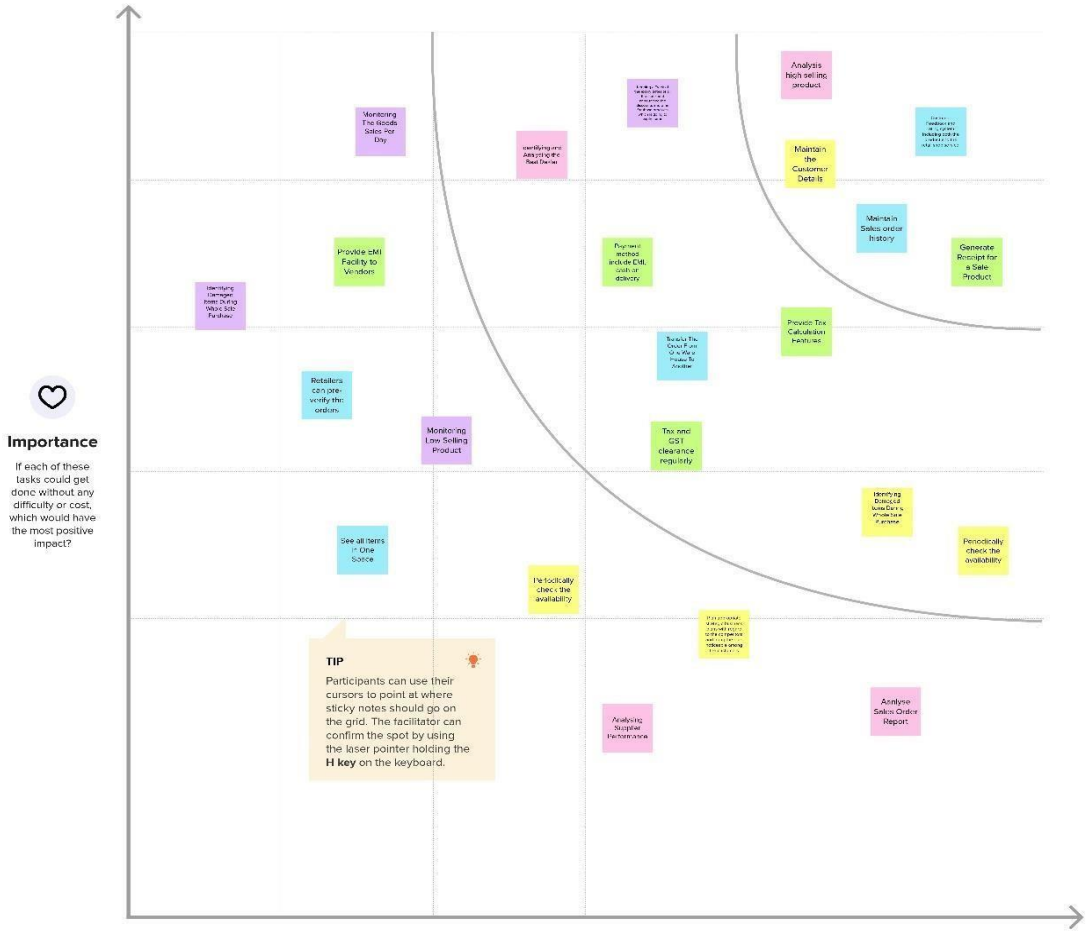
Step-3: Idea Prioritization

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.

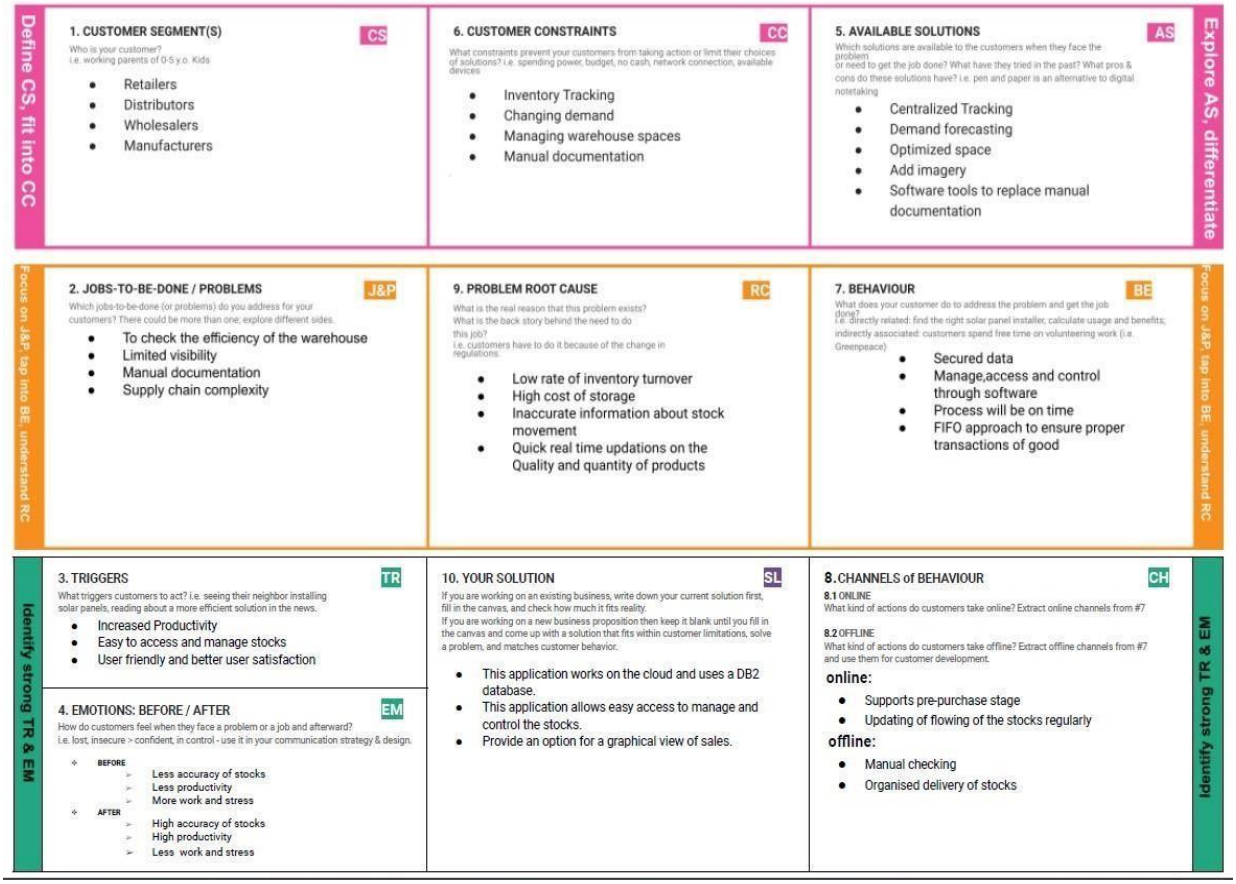
20 minutes



3.3 Proposed Solution

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Retailers do not have any systematic system to record and keep their inventory data.
2.	Idea / Solution description	To develop a cloud web application that will help retailers to manage, track, and control their stocks.
3.	Novelty / Uniqueness	Real time inventory tracking and secured user interactions.
4.	Social Impact / Customer Satisfaction	Better interface to understand the tracking of stocks and better reliability over stock management.
5.	Business Model (Revenue Model)	This cloud web application will get higher usage and acceptance in market and among people of this generation.
6.	Scalability of the Solution	9 out of 10

3.4 Problem Solution fit



4 REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Registration**	Registration through Form Registration through Gmail
FR-2	**User Confirmation**	Confirmation via Email Confirmation via OTP
FR-3	**User Login**	Username/Email-ID Login with Password
FR-4	**Admin Login**	Login with Username/Email-ID Login with Password
FR-5	**Inventory Management**	Track quantity of products present in inventory at any instant
FR-6	**Tracking of stock**	Notifications through Email

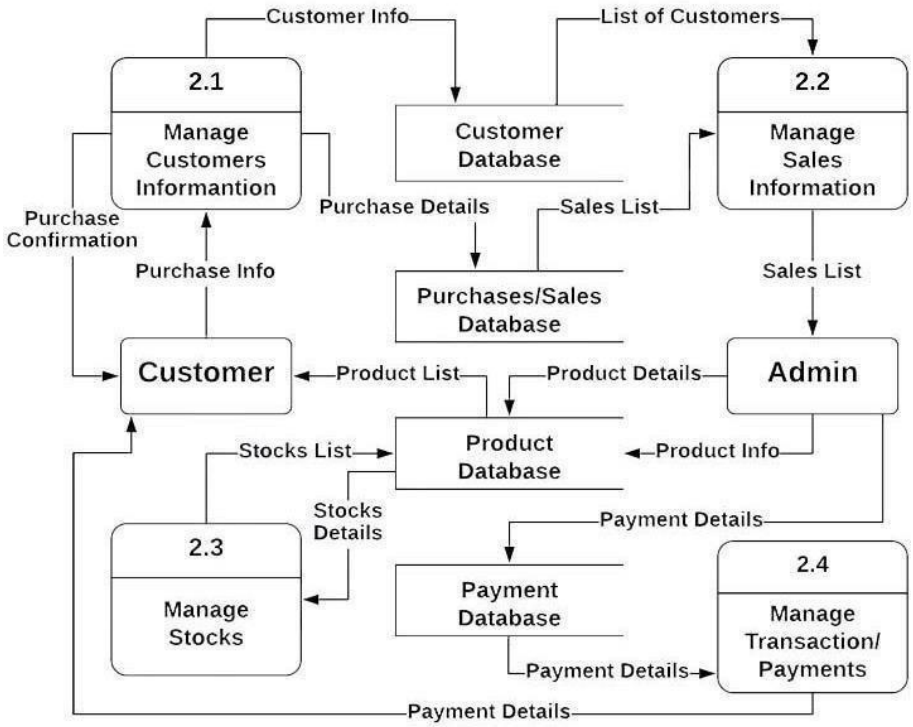
4.2 Non-Functional requirements

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	This cloud web application makes the process of inventory management a lot easier which saves money and time both. This system is highly responsive to both desktop and mobile users.
NFR-2	**Security**	Inventory security aims to prevent inventory losses – for example, due to incorrect storage, theft, or incorrect incoming goods inspection – so that the correct stock is always available.

NFR-3	**Reliability**	The availability of products should be properly updated for customer satisfaction. The out-of-stock information should be notified. The system must give accurate inventory status to the user continuously.
NFR-4	**Performance**	The companies have to design and operate materials management and product distribution functions effectively. Inventory control systems enable a business to determine and maintain an optimum level of investment in inventory in order to achieve the required operational performance.
NFR-5	**Availability**	The software will be available only to the administrator of the organization and the product, as well as customer details, will be recorded by him. He can manage the inventory.
NFR-6	**Scalability**	The System can manage large inventory and provides quick access to the inventory in no time.

5 PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture

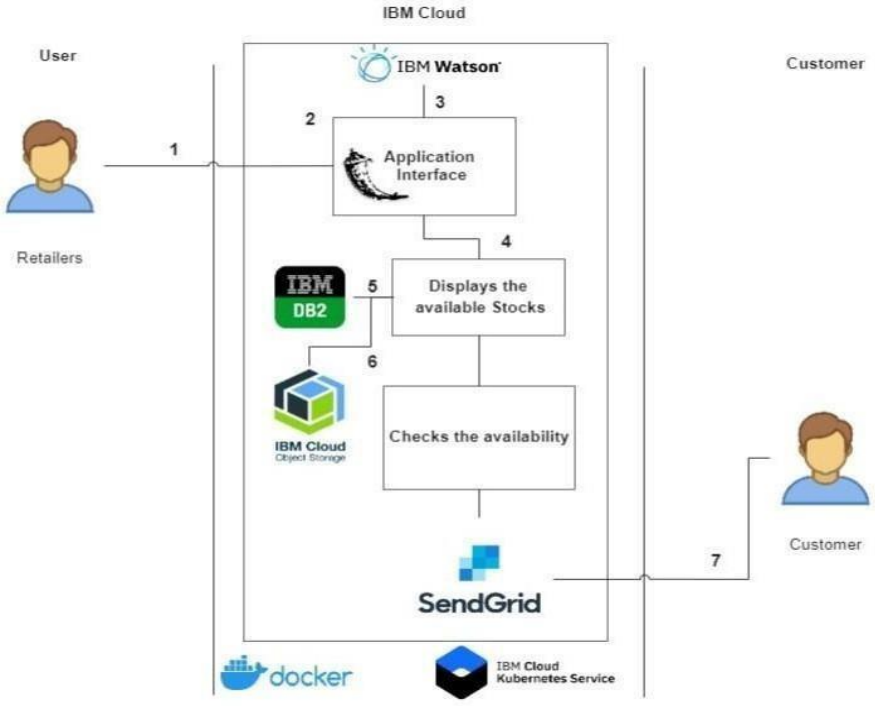


Table-1: Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	User interacts with the cloud web application.	HTML, CSS, JavaScript
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson Assistant
4.	Database	Data Type, Configurations	MySQL
5.	Cloud Database	Database Service on Cloud	IBM DB2
6.	File Storage	File storage requirements	IBM Object Storage
7.	Infrastructure (Server / Cloud)	Kubernetes is an open-source container orchestration system for automating softwaredeployment, scaling, and management	Kubernetes

Table-2: Application Characteristics:

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Micro web framework, written in Python	Flask
2.	Security Implementations	List all the security/access controls implemented,use of firewalls, etc.	SHA-256
3.	Scalable Architecture	Kubernetes is an open-source container orchestration system for automating softwaredeployment, scaling, and management.	Kubernetes
4.	Availability	Docker CLI stores its configuration files in adirectory called .docker within your $HOME directory.	Docker CLI
5.	Performance	To send alerts to users based on their stock	Sendgrid

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application byentering my email, password, and confirmingmy password.	I can access my account/dashboard	High	Sprint-1
		USN-2	As a user, I will receive a confirmation emailonce I have registered for the application	I can receive a confirmation email & click confirm	Medium	Sprint-1
	Confirmation	USN-3	As a user, I will confirm the registration once Ihave received the email from the application	I can get a confirmation for my email and password and create an authenticated account.	Medium	Sprint-1
	Login	USN-4	As a user, I can log in to the applicationthrough Gmail & Password	I can log onto the application with a verifiedemail and password	High	Sprint-1
	Dashboard	USN-5	As a user, I can view the dashboard of the application by entering my email & password	Once I log on to the application, I can view products to buy.	High	Sprint-2
	Add items to thecart	USN-6	As a user, I can add the products I wish to buyto the carts.	As a user, I can buy any product or add it to my cart for buying later	Medium	Sprint-2
	Stock Update	USN-7	As a user, I can add products that are not available in the dashboard to the stock list.	If any of the products arenot available, as a user Ican update the inventory and send mail to theowner	Medium	Sprint-3
Customer (Webuser)	Registration	USN-8	As a user, I can register for the application byentering my email, password, and confirmingmy password.	I can access my account/dashboard	High	Sprint-1
		USN-9	As a user, I will receive a confirmation emailonce I have registered for the application	I can receive a confirmation email & clickconfirm	Medium	Sprint-1
	Confirmation	USN-10	As a user, I will confirm the registration once Ihave received the email from the application	I can get a confirmation formy email and password	Medium	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
				and create an authenticated account.		
	Login	USN-11	As a user, I can log in to the applicationthrough Gmail & Password	I can log onto the application with a verifiedemail and password	High	Sprint-1
	Dashboard	USN-12	As a user, I can view the dashboard of the application by entering my email & password	Once I log on to the application, I can viewproducts to buy.	High	Sprint-2
	Add items to thecart	USN-13	As a user, I can add the products I wish to buyto the carts.	As a user, I can buy any product or add it to my cartfor buying later	Medium	Sprint-2
	Stock Update	USN-14	As a user, I can add products that are not available in the dashboard to the stock list.	If any of the products arenot available, as a user Ican update the inventoryand send mail to the owner	Medium	Sprint-3
Customer Care Executive	Request to Customer Care	USN-15	As a user, I can contact the Customer CareExecutive and request any services I want from customer care.	As a user, I can contactCustomer Care and get support.	Low	Sprint-4
Administrator	Contact Administrator	USN-16	I can be able to report any difficulties Iexperience as a report	As a user, and I can givemy support in possible ways to the administrator and the administration.	Medium	Sprint-4

6 PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

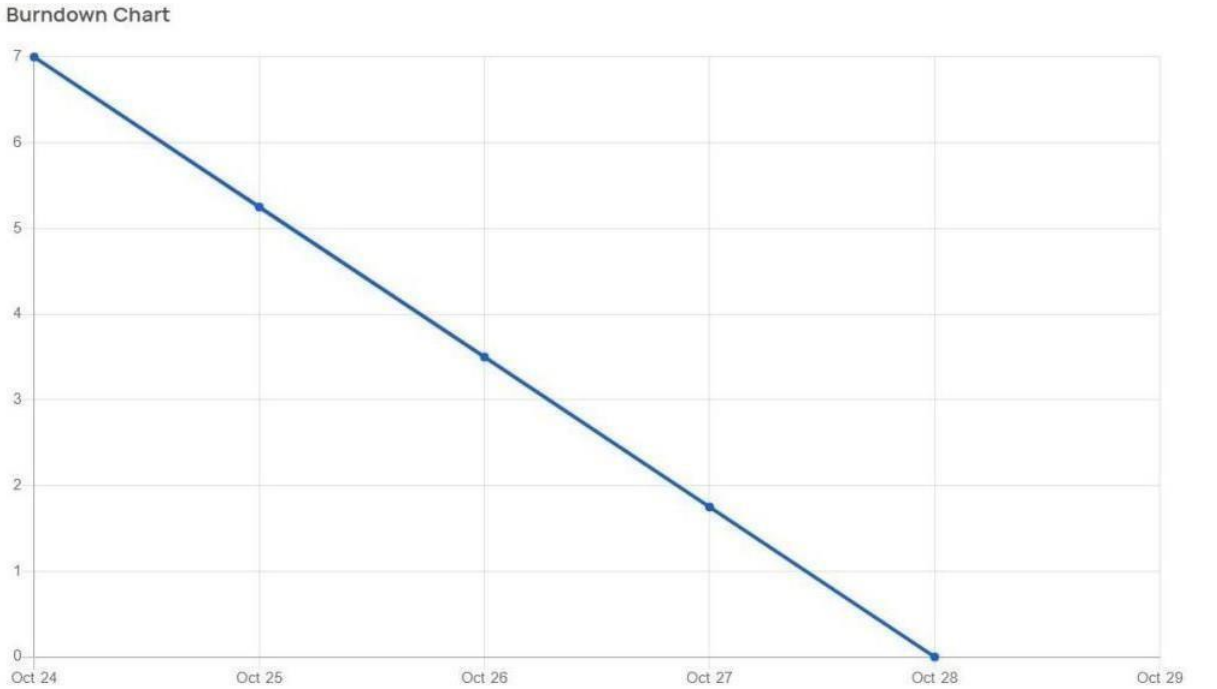
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Bala Yogesh
Sprint-1	Registration	USN-2	As a user, I can register for the application through E-mail	1	High	Vinoth Kumar
Sprint-1	Confirmation	USN-3	As a user, I will confirm the registration once I have received the email from the application	2	Medium	Ajay
Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	2	High	Keerthana
Sprint-2	Dashboard	USN-5	As a user, I can view the dashboard of the application by entering my email & password	3	High	Bala Yogesh
Sprint-2	ChatBot	USN-6	As a user, I can interact and get help from chatbot	3	Medium	Vinoth Kumar
Sprint-2	Add items to cart	USN-7	As a user, I can add the products I wish to buy to the carts..	3	Medium	Ajay M
Sprint-3	Stock Update	USN-8	As a user, I can add products that are not available in the dashboard to the stock list.	5	Medium	Keerthana
Sprint-4	Request to Customer Care	USN-9	As a user, I can post queries through mail	5	Low	Bala Yogesh
Sprint-4	Feedback	USN-10	As a user, I can give my feedback by submitting the form.	5	Low	Vinoth Kumar

Sprints	Sprint Duration	Velocity	Actual velocity
Sprint-1	6	7	0.85
Sprint-2	6	9	0.66
Sprint-3	6	5	1.2
Sprint-4	6	10	0.6

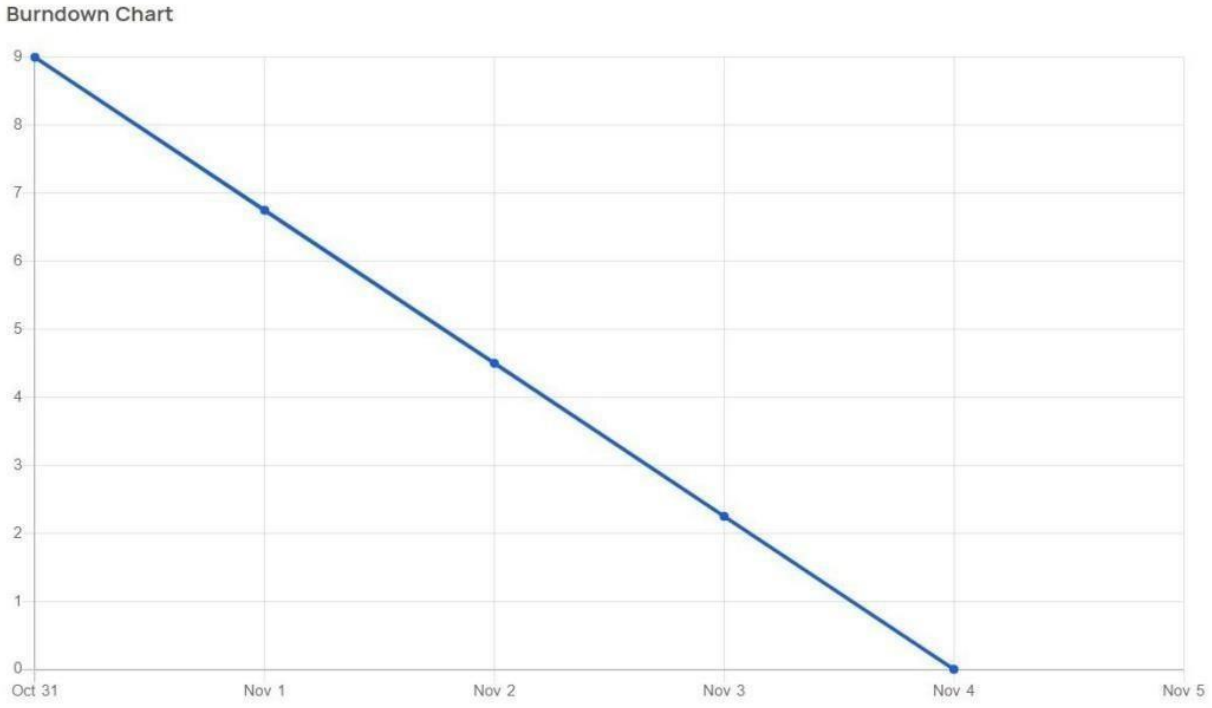
6.2 Sprint Delivery Schedule

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	7	6 Days	24 Oct 2022	29 Oct 2022	7	29 Oct 2022
Sprint-2	9	6 Days	31 Oct 2022	05 Nov 2022	9	29 Oct 2022
Sprint-3	5	6 Days	07 Nov 2022	12 Nov 2022	5	12 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022	10	19 Nov 2022

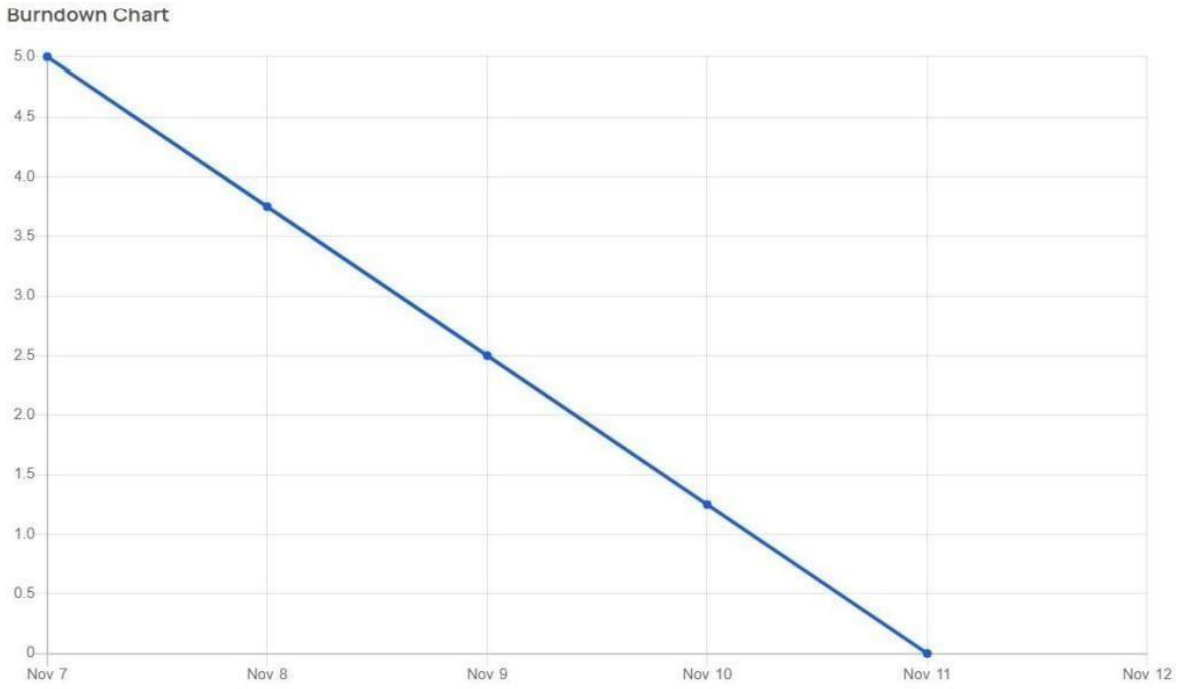
SPRINT-1


Burndown Chart
7
6
5
4
3
2
1
0
Oct 24
Oct 25
Oct 26
Oct 27
Oct 28
Oct 29

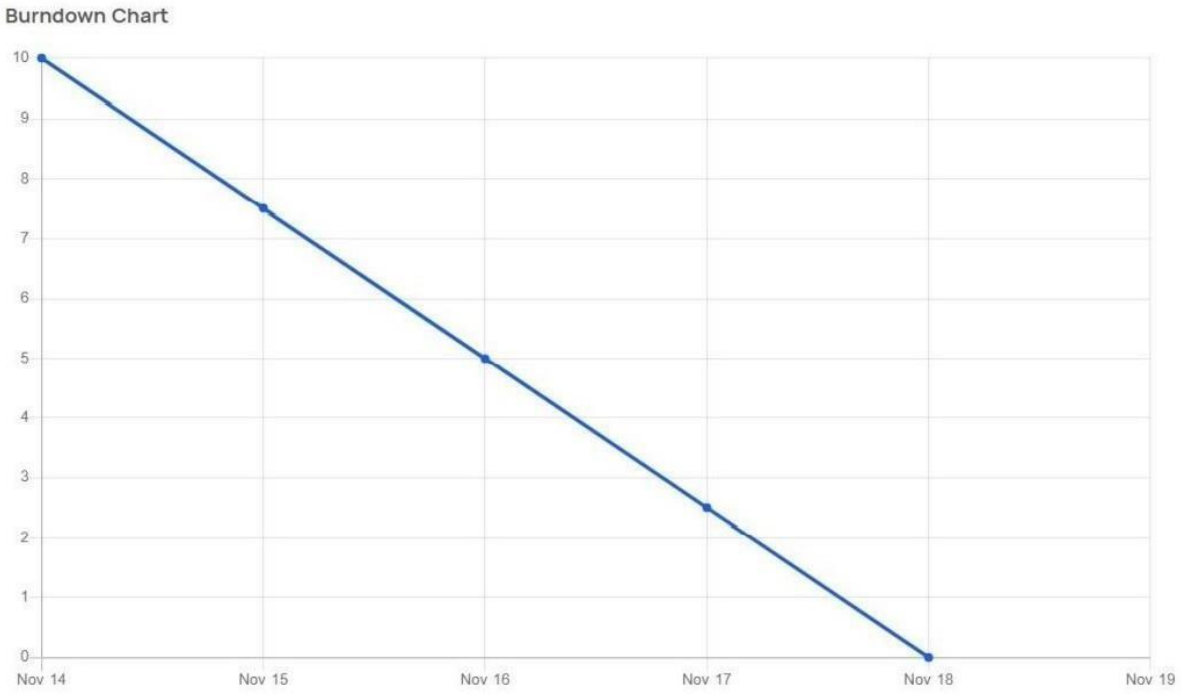
SPRINT-2



SPRINT-3



SPRINT-4

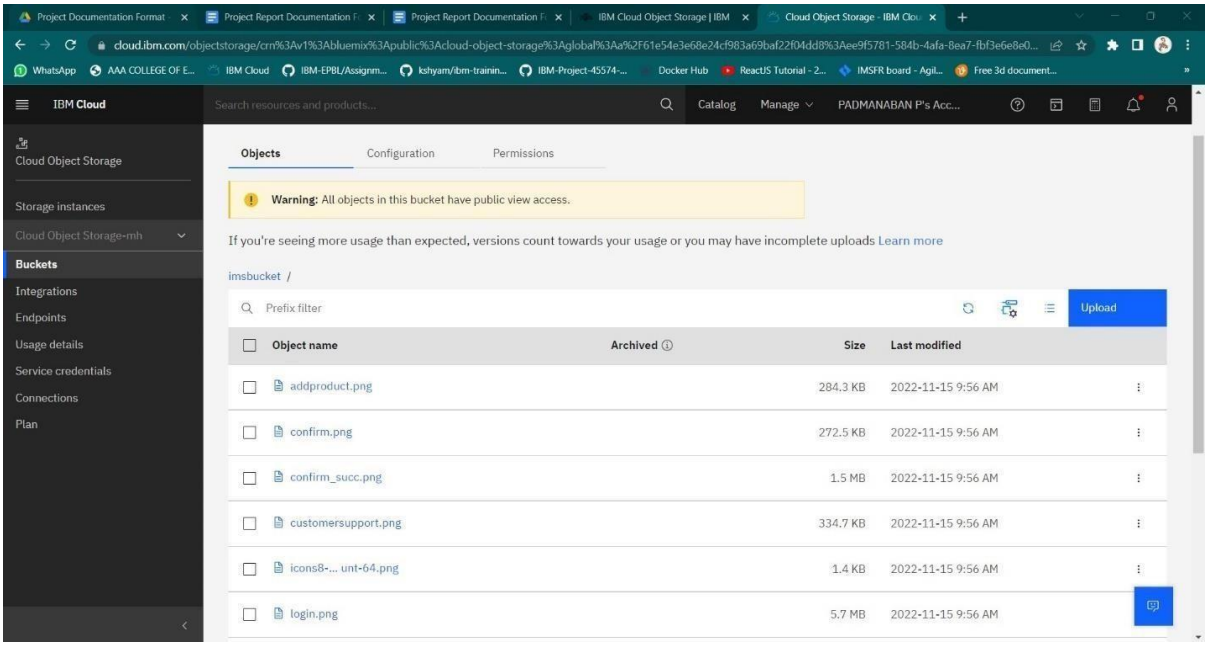

Burndown Chart
10
9
8
7
6
5
4
3
2
1
0
Nov 14
Nov 15
Nov 16
Nov 17
Nov 18
Nov 19

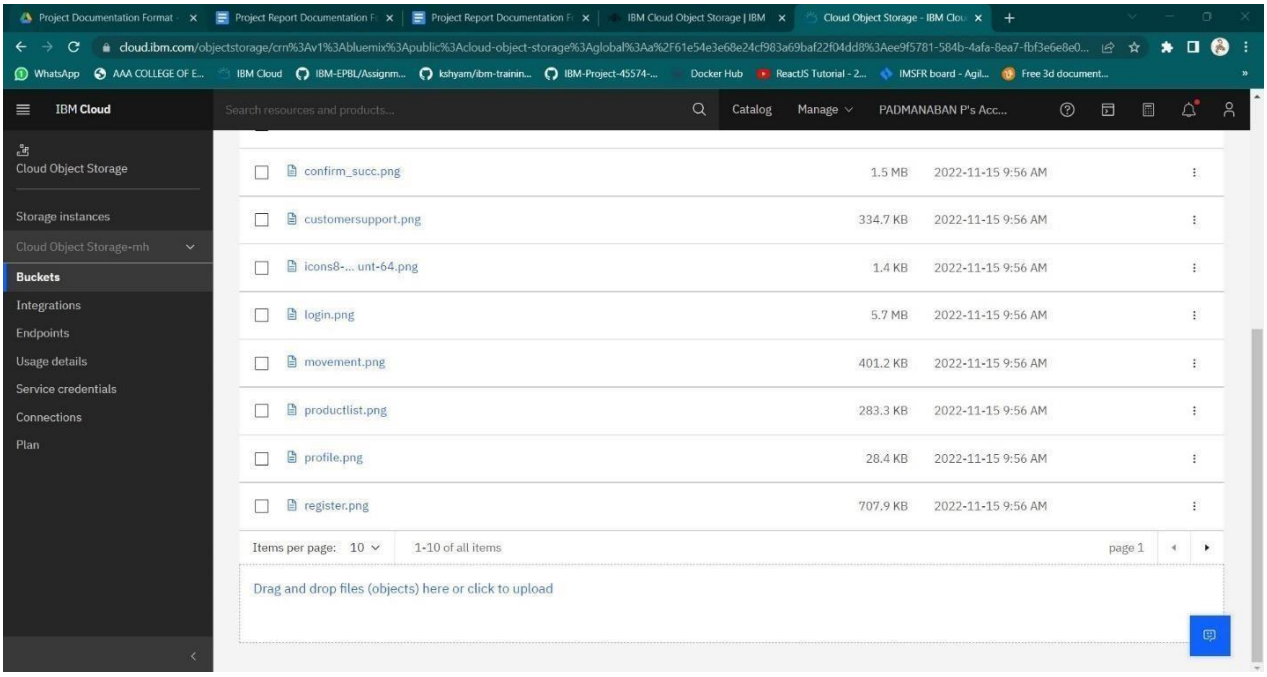
7 CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 IBM Object Storage Service

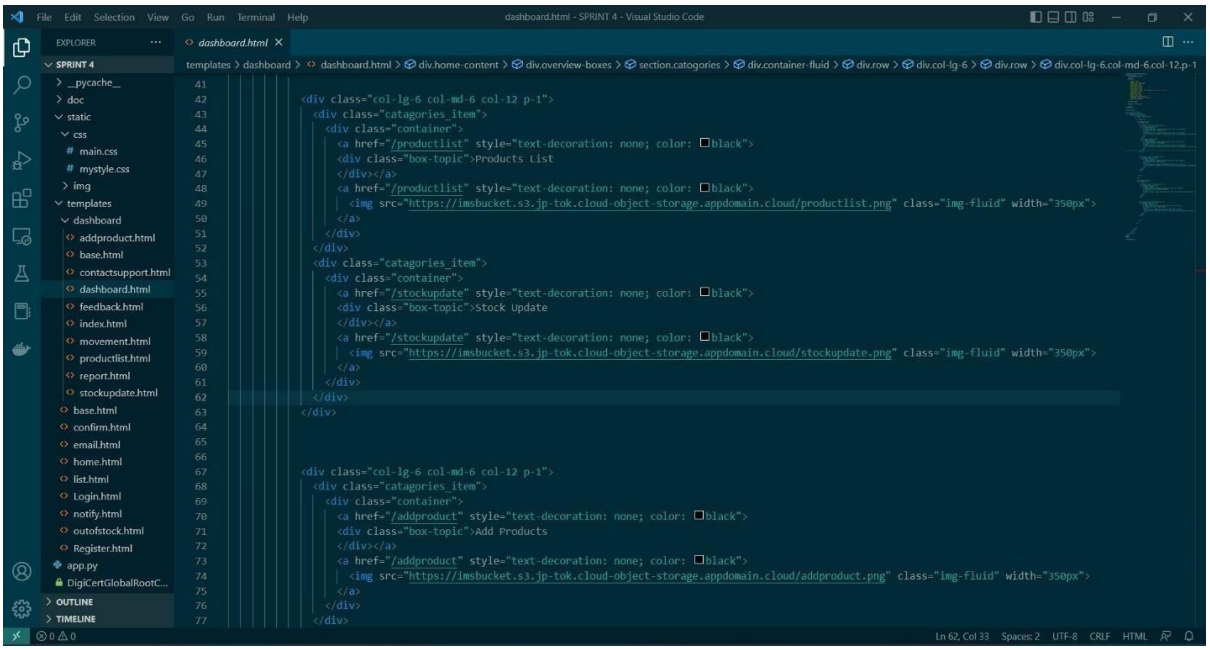
IBM Cloud Object Storage, with its global presence and flexible resiliency options, supports exponential data growth for your cloud-native workloads with best-in-class cost optimization, robust data security, and data governance with ease of use. Built-in data lifecycle operations also make it easy to observe and manage your critical workloads.

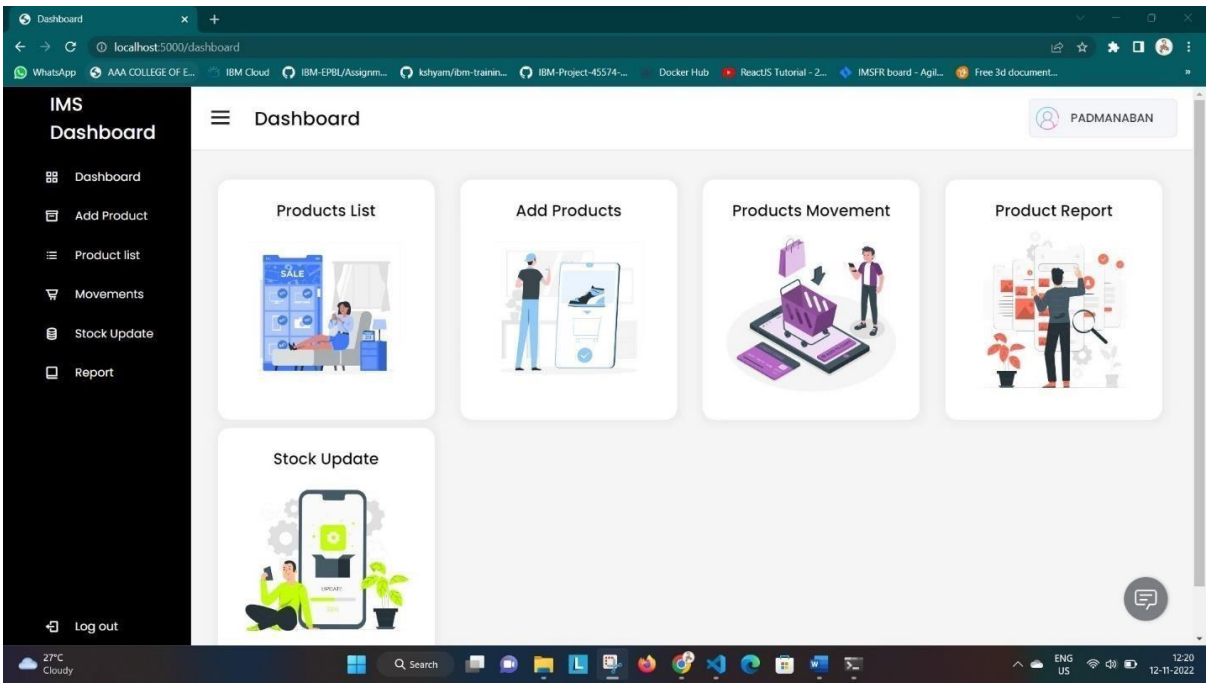
Creating a bucket in IBM Object Storage and uploading the images to the bucket:





Displaying the images in the application by accessing the images publicly:

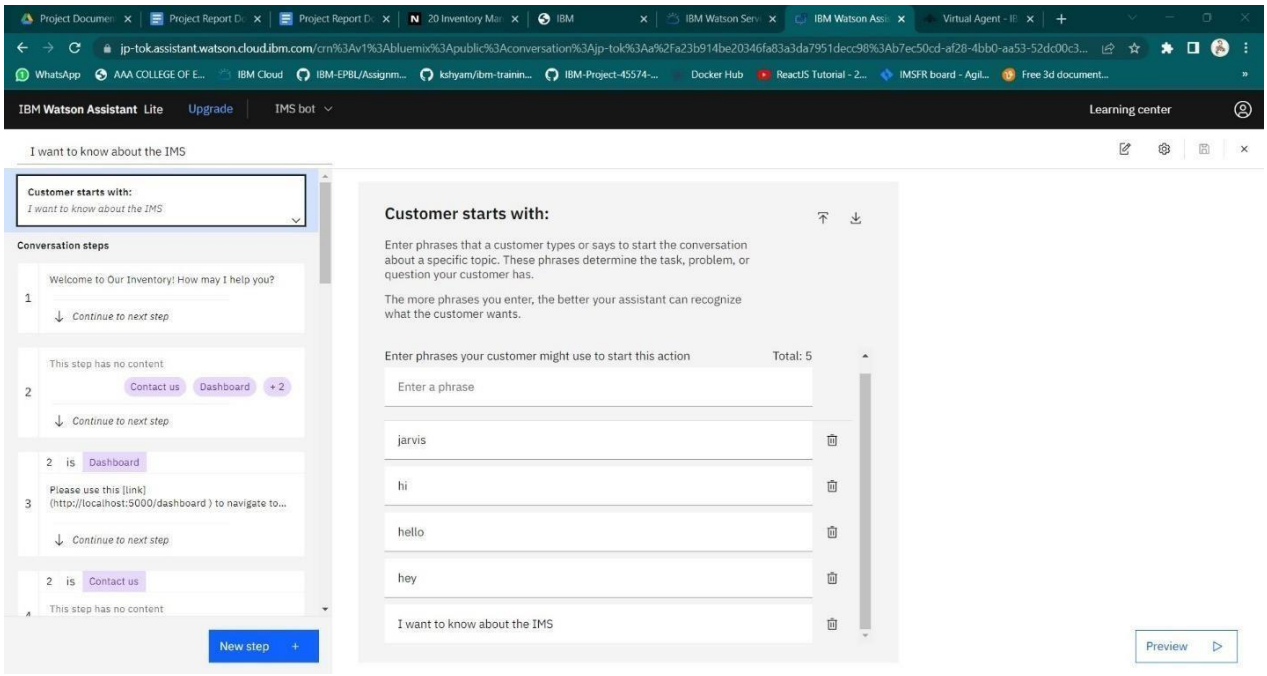


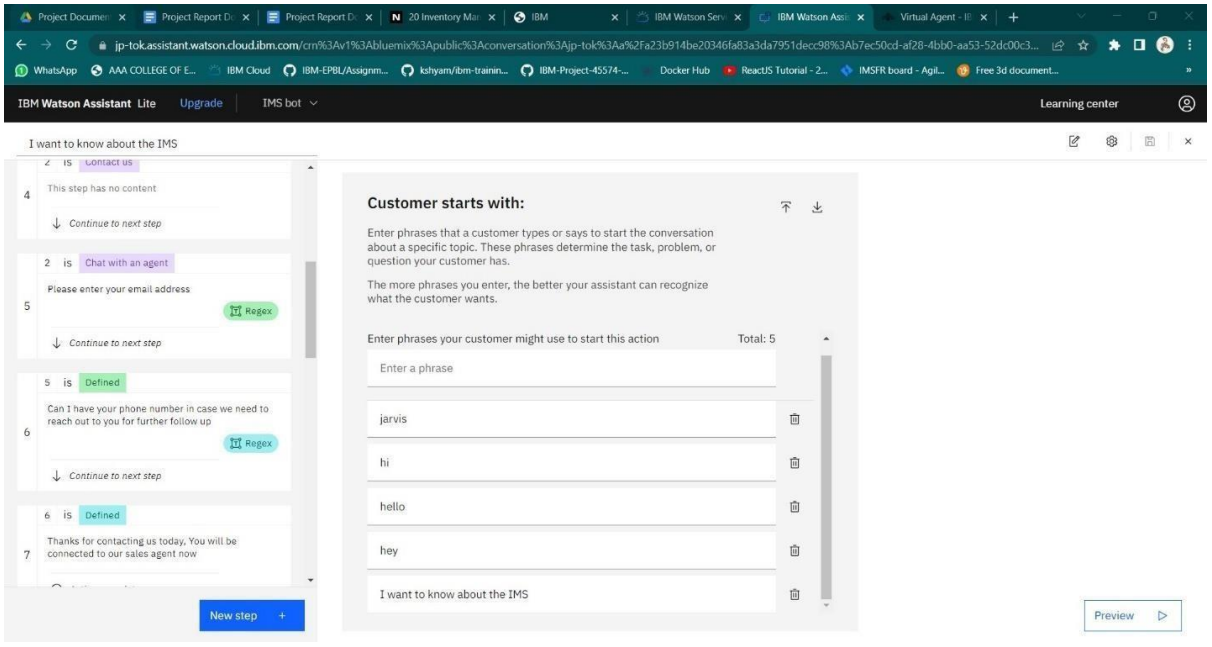

Dashboard
localhost:5000/dashboard
IMS Dashboard
Dashboard
Add Product
Product list
Movements
Stock Update
Report
Log out
Dashboard
PADMANABAN
Products List
Add Products
Products Movement
Product Report
Stock Update

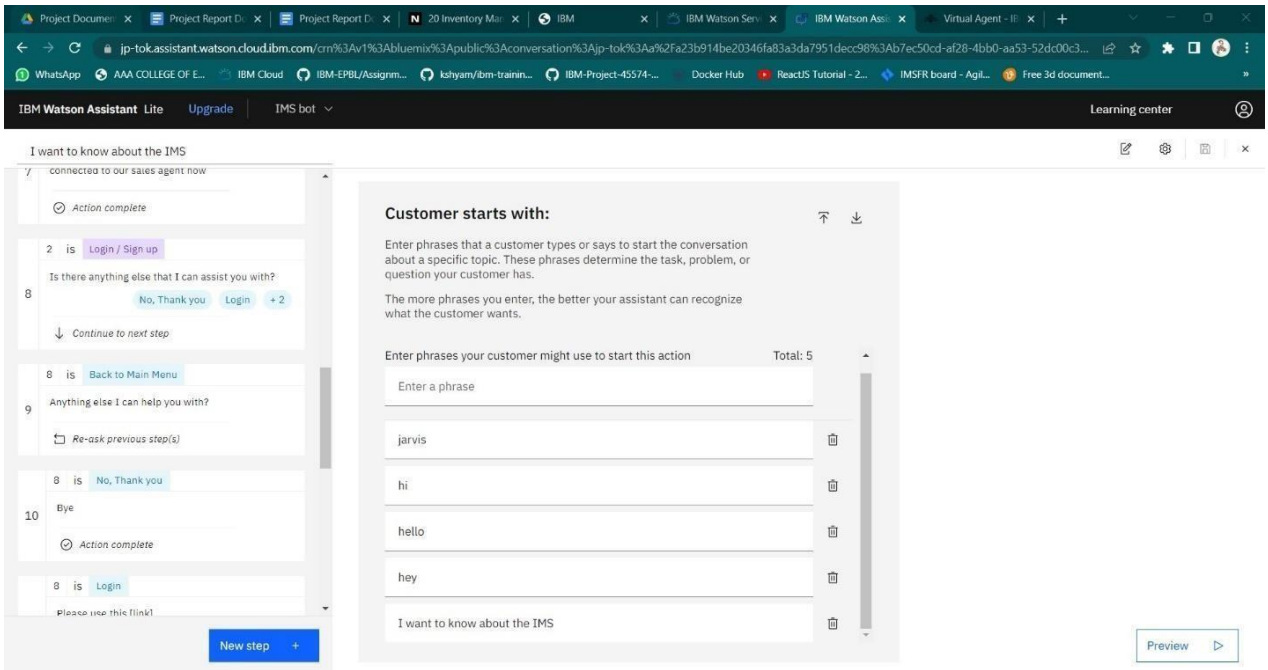
7.2 IBM Watson Assistant Service

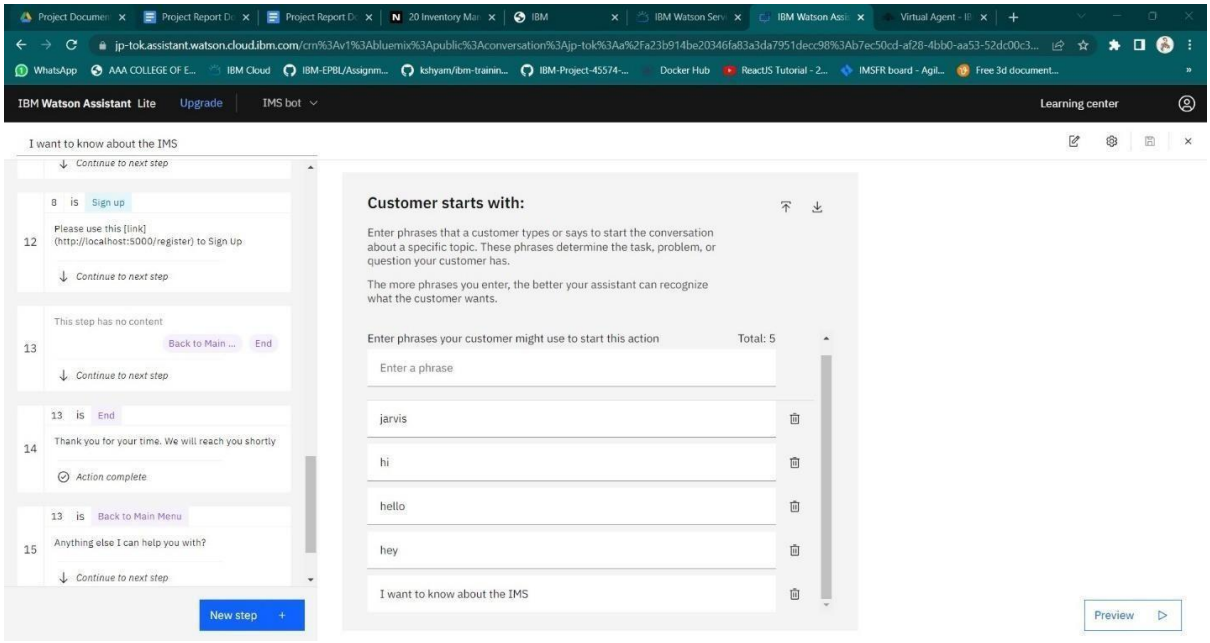
IBM Watson Assistant uses artificial intelligence that understands customers in context to provide fast, consistent, and accurate answers across any application, device, or channel. Remove the frustration of long wait times, tedious searches, and unhelpful chatbots with the leader in trustworthy AI.

Setting Actions to IBM Watson Assistant:





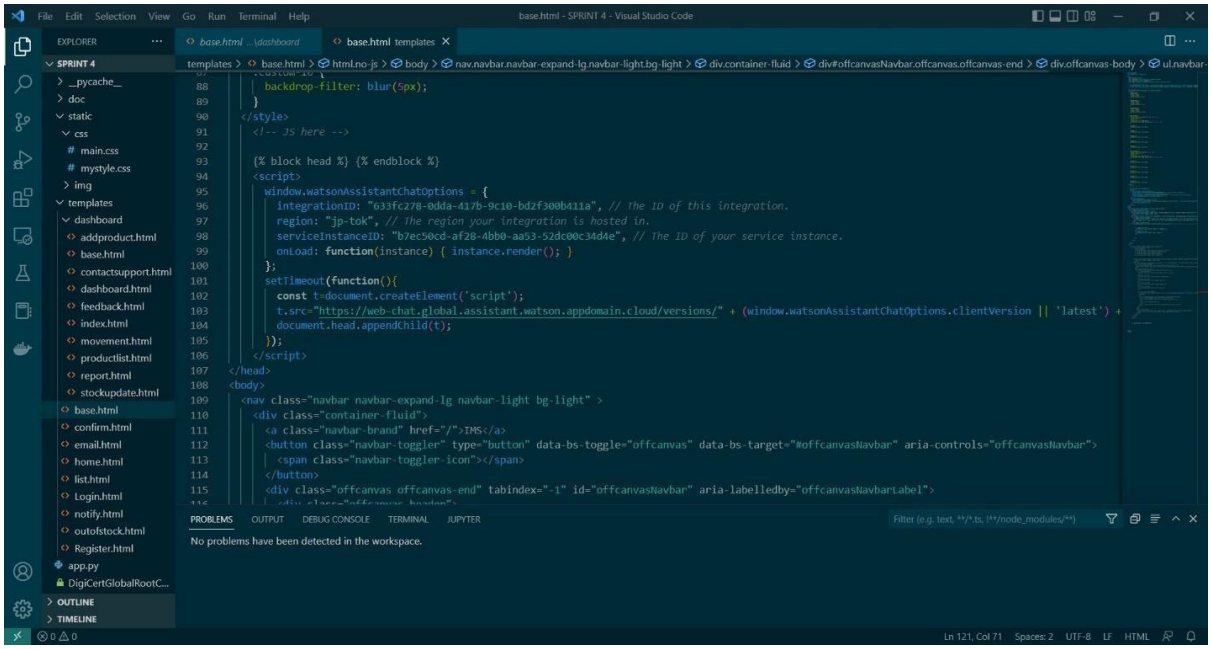

IBM Watson Assistant Lite
Upgrade
IMS bot
Learning center
I want to know about the IMS
Action complete
2 is Login / Sign up
Is there anything else that I can assist you with?
No, Thank you
Login
+ 2
Continue to next step
8 is Back to Main Menu
Anything else I can help you with?
Re-ask previous step(s)
8 is No, Thank you
Bye
Action complete
8 is Login
New step
Customer starts with:
Enter phrases that a customer types or says to start the conversation about a specific topic. These phrases determine the task, problem, or question your customer has.
The more phrases you enter, the better your assistant can recognize what the customer wants.
Enter phrases your customer might use to start this action
Total: 5
Enter a phrase
jarvis
hi
hello
hey
I want to know about the IMS
Preview

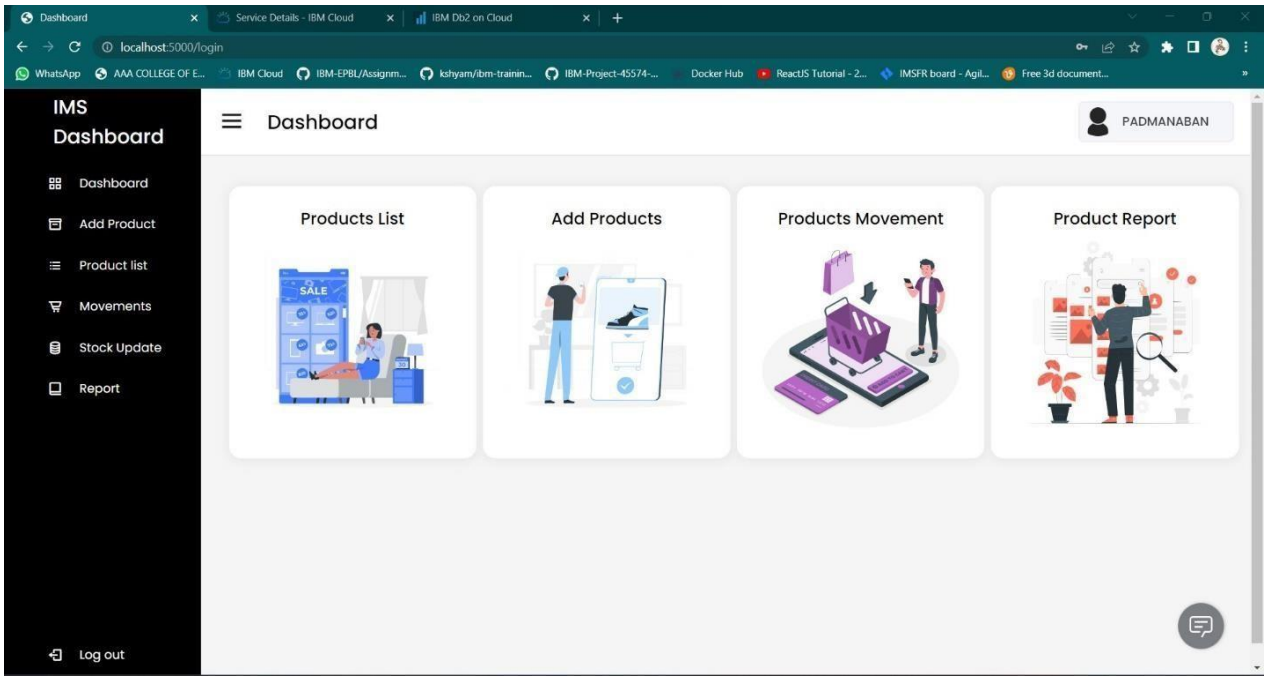

IBM Watson Assistant Lite
Upgrade
IMS bot
Learning center
I want to know about the IMS
Continue to next step
8 is Sign up
Please use this [link] (http://localhost:5000/register) to Sign Up
Continue to next step
This step has no content
Back to Main ...
End
Continue to next step
13 is End
Thank you for your time. We will reach you shortly
Action complete
13 is Back to Main Menu
Anything else I can help you with?
Continue to next step
New step
Customer starts with:
Enter phrases that a customer types or says to start the conversation about a specific topic. These phrases determine the task, problem, or question your customer has.
The more phrases you enter, the better your assistant can recognize what the customer wants.
Enter phrases your customer might use to start this action
Total: 5
Enter a phrase
jarvis
hi
hello
hey
I want to know about the IMS
Preview

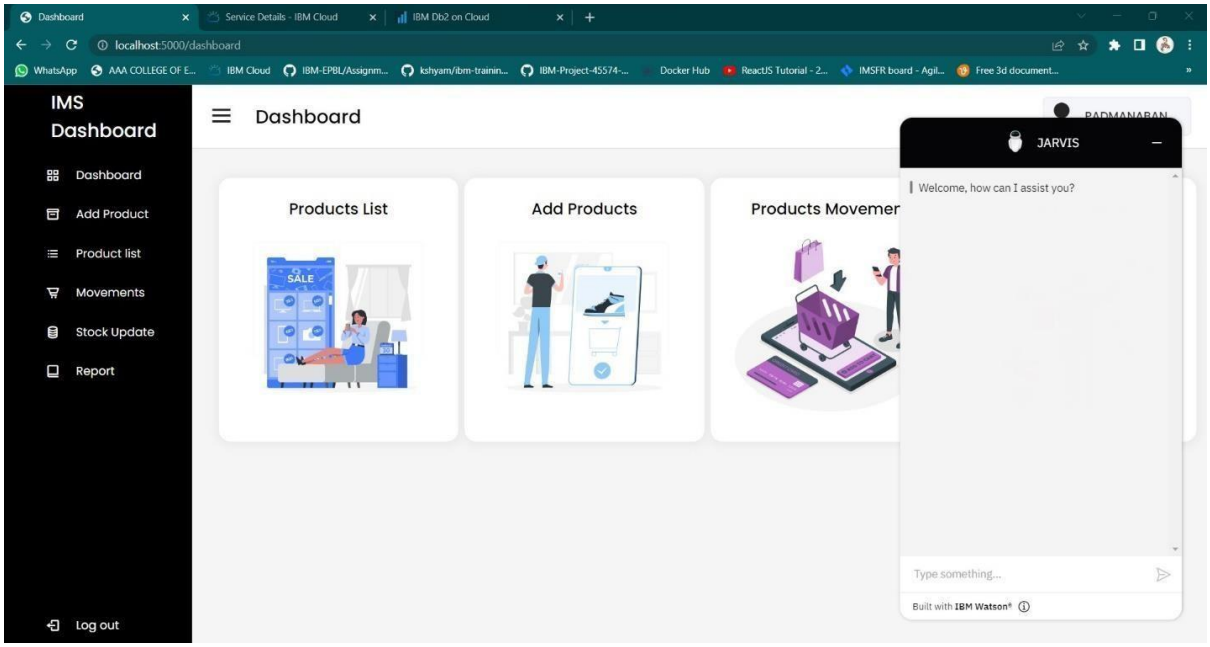
Code:

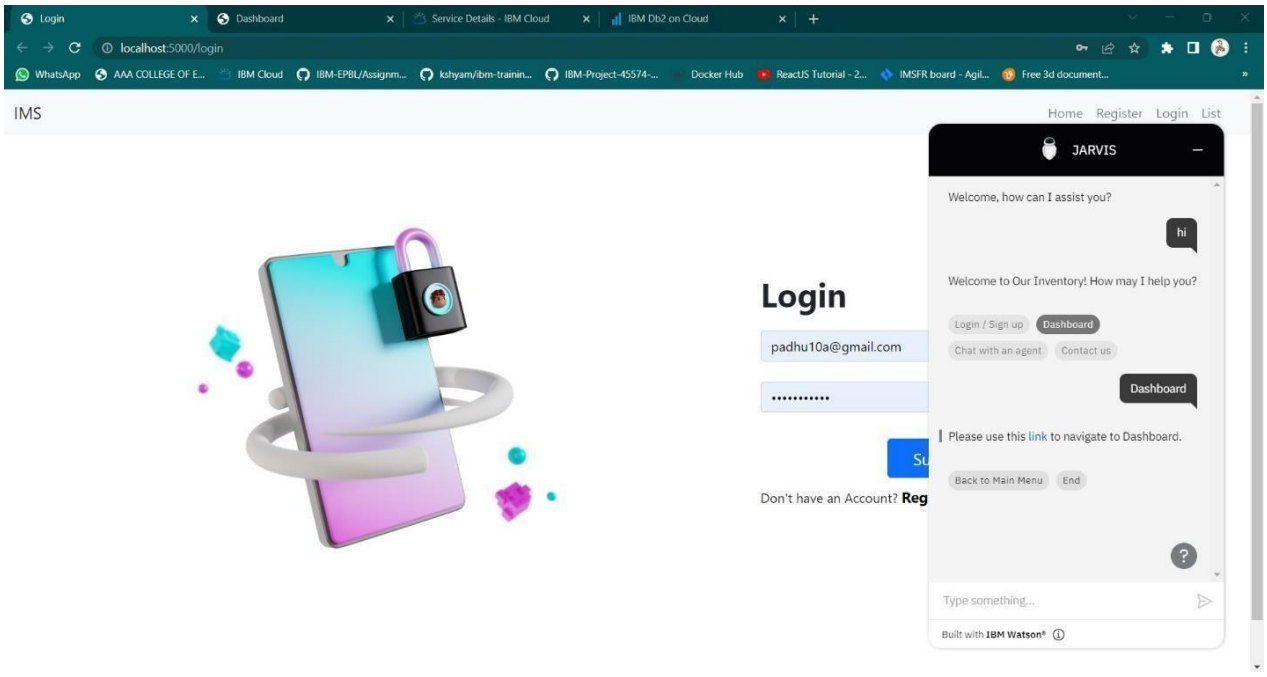
```
<script>
      window.watsonAssistantChatOptions = {
        integrationID: "633fc278-0dda-417b-9c10-bd2f300b411a", // The ID of this integration.
        region: "jp-tok", // The region your integration is hosted in.
        serviceInstanceID: "b7ec50cd-af28-4bb0-aa53-52dc00c34d4e", // The ID of your
service instance.
        onLoad: function(instance) { instance.render(); }
      };
      setTimeout(function(){
        const t=document.createElement('script');
        t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
        document.head.appendChild(t);
      });
    </script>
```

Integrating the IBM Watson Assistant in the base template:




Dashboard
Service Details - IBM Cloud
IBM Db2 on Cloud
localhost:5000/login
IMS Dashboard
Dashboard
Add Product
Product list
Movements
Stock Update
Report
Log out
Dashboard
PADMANABAN
Products List
Add Products
Products Movement
Product Report
SALE


Dashboard
Service Details - IBM Cloud
IBM Db2 on Cloud
localhost:5000/dashboard
IMS Dashboard
Dashboard
Add Product
Product list
Movements
Stock Update
Report
Log out
Dashboard
Products List
Add Products
SALE
JARVIS
Welcome, how can I assist you?
Type something...
Built with IBM Watson®

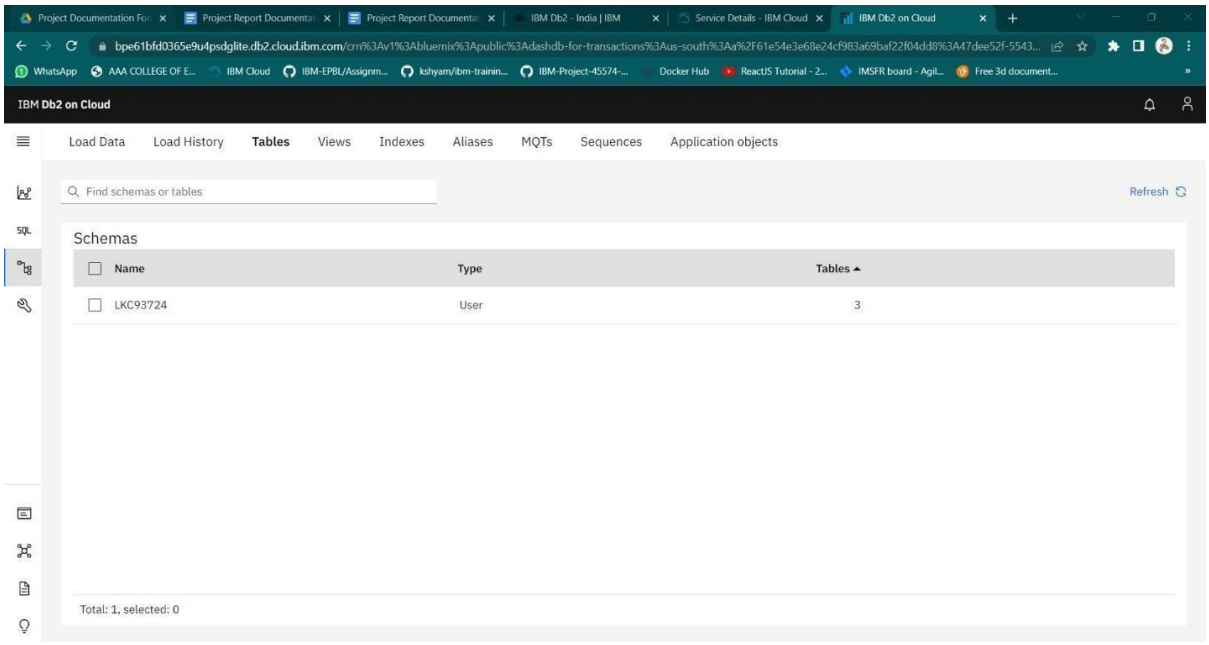

Login
Dashboard
Service Details - IBM Cloud
IBM Db2 on Cloud
localhost:5000/login
WhatsApp
AAA COLLEGE OF E...
IBM Cloud
IBM-EPBL/Assignm...
kshyam/ibm-trainin...
IBM-Project-45574-...
Docker Hub
ReactJS Tutorial - 2...
IMSFR board - Agil...
Free 3d document...
IMS
Home
Register
Login
List
Login
padhu10a@gmail.com
Don't have an Account? Reg
JARVIS
Welcome, how can I assist you?
hi
Welcome to Our Inventory! How may I help you?
Login / Sign up
Dashboard
Chat with an agent
Contact us
Dashboard
Please use this link to navigate to Dashboard.
Back to Main Menu
End
Type something...
Built with IBM Watson®

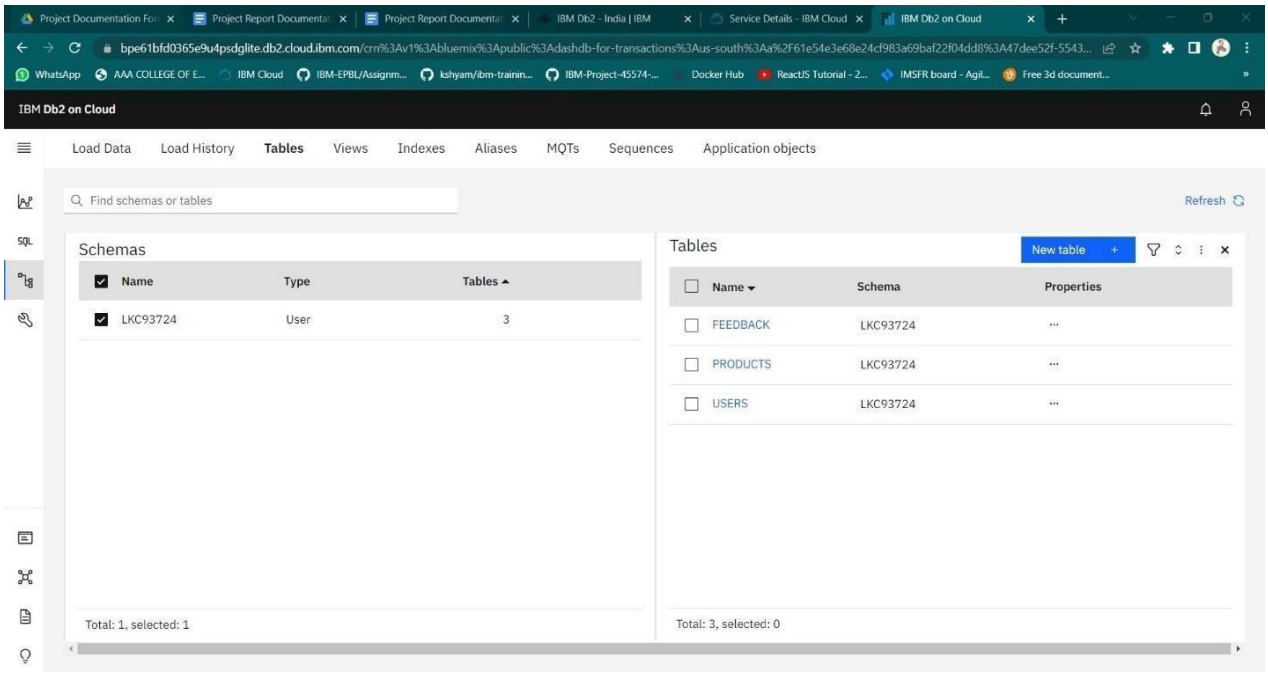
7.3 Database Schema(IBM DB2)

IBM Db2 Database on IBM Cloud combines a proven, AI-infused, enterprise-ready data management system with an integrated data and AI platform built on the security-rich, scalable Red Hat® OpenShift® foundation. Derive insights with machine learning embedded into query processing. Cut costs with the multimodal capability that eliminates the need for data replication and migration. And enhance agility by running Db2 on any cloud vendor.

Code: (Connecting with IBM Db2 in app.py)

```
import ibm_db

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2f15b-5c0f-46df-8954-7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32733;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lkc93724;PWD=zAzNGa6DaNk6xvle","","")
```





8 TESTING

8.1 Test Cases

Date	16-Nov-22
Team ID	PNT2022TMID51170
Project Name	Project - Inventory Management System For Retai
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Singup popup displayed or not	https://shopenzer.com/	Login/Signup popup should display	Working as expected	Pass			
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Singup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link e.Last password? Recovery password link	https://shopenzer.com/	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	Pass	Steps are not clear to follow		BUG-1234
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: padhu10a@gmail.com password: padhu123	User should navigate to user account homepage	Working as expected	Pass			
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: padhu10a@gmail.com password: padhu123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass			
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: padhu10a@gmail.com password: padhu123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass			
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: padhu10a@gmail.com password: padhu123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass			

8.2 User Acceptance Testing

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the INVENTORY MANAGEMENT SYSTEM project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

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

Resolution	Seveíity 1	Seveíity 2	Seveíity 3	Seveíity 4	Subtotal
By Design	10	4	2	3	19
Duplicate	1	0	3	0	4
Exteínal	2	3	0	1	6
Fixed	11	2	4	20	37
Not Repíoduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Ⅰᵓotals	24	14	13	26	77

3. Test Case Analysis

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

Section	Ⅰᵓotal Cases	Not Ⅰᵓested	Ïail	Pass
Píint Engine	6	0	0	6
Client Application	25	0	0	20
Secuíity	2	0	0	2

Outsouíce Shipping	3	0	0	3
Exception Repoíting	7	0	0	7
Final Repoít Output	4	0	0	4
Veísion Contíol	2	0	0	2

9 RESULTS

9.1 Performance Metrics

					Date	16-Nov-22			
					Team ID	PNT2022TMID51170			
					Project Name	Project - Inventory Management System For Retailers			
				NFT - Risk Assessment					
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score	Justification
1	Inventory Management System for Retail	New	Moderate	No Changes	Moderate		>30 to 50 %	ORANGE	As we have seen the changes
2	Inventory Management System for Reta	New	Low	No Changes	Low		>5 to 10%	GREEN	As we have seen the changes
3	Inventory Management System for Reta	New	High	No Changes	High		>50 to 70%	RED	As we have seen the changes
4	Inventory Management System for Reta	New	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	Assumptions/Dependencies/Risks	Approvals/SignOff	
				1	Inventory Management System for Reta	Scalability	moderate	Padmanaban	
				End Of Test Report					
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	(Detected/Closed/Open)	Approvals/SignOff	
1	Inventory Management System for Reta	Scalability	Yes	Good		increase the number of pods	Closed	padmanaban	

10 ADVANTAGES & DISADVANTAGES

10.1 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.

Greater Insights:

With inventory tracking and stock control, you can also easily spot sales trends or track recalled products or expiry dates.

More Productivity:

Good inventory management solutions save time that could be spent on other activities.

10.2 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.

11 CONCLUSION

Inventory management is a very complex but essential part of the supply chain. An effective inventory management system helps to reduce stock-related costs such as warehousing, carrying, and ordering costs. As you have read above, there are different techniques that

businesses can utilize to simplify and optimize stock management processes and control systems.

12 FUTURE SCOPE

Stock control for omnichannel retailing

Stores doing omnichannel retailing are at the top of their game; they attract the 90% of consumers who switch between at least three applications per day to complete specific tasks.

Inventories that power experiential retail

Experiential retail is a trend that's catching fire — especially in the past few months.

In fact, they keep popping up in the news section of Google search results.

Advanced sales forecasting

Inventory management is big on having products on the shelf ready for shoppers when they need it. It curbs stockouts and fosters better personalization.

Season-based product recommendations

Speaking of Artificial Intelligence, AI-powered recommendation engines that adjust inventory based on real-time weather conditions and forecast are making their way to retail stores.

13 APPENDIX

Source Code

App.py:

```
from flask import Flask, render_template, url_for, request, redirect, session, make_response
import sqlite3 as sql
from functools import wraps
import re
import ibm_db
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
from datetime import datetime, timedelta

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=764264db-9824-4b7c-82df-
40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32536;SECURITY=SSL;SSLServerCert
ificate=DigiCertGlobalRootCA.crt;UID=qwq87197;PWD=7TN1X5zgnKSTn9uc",'','')

app = Flask(__name__)
app.secret_key = 'ramcoinstitute'
```

```
def rewrite(url):
    view_func, view_args = app.create_url_adapter(request).match(url)
    return app.view_functions[view_func](**view_args)


def login_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        if "id" not in session:
            return redirect(url_for('login'))
        return f(*args, **kwargs)
    return decorated_function


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



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

    if request.method == 'POST':
        un = request.form['username']
        pd = request.form['password_1']
        print(un, pd)
        sql = "SELECT * FROM Client WHERE username =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, un)
        ibm_db.bind_param(stmt, 2, pd)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['loggedin'] = True
            session['id'] = account['EMAIL']
            userid = account['EMAIL']
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'

            return rewrite('/dashboard')
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg=msg)
```

```
@app.route('/signup', methods=['POST', 'GET'])
def signup():
    mg = ''
    if request.method == "POST":
        username = request.form['username']
        email = request.form['email']
        pw = request.form['password']
        sql = 'SELECT * FROM Client WHERE username =?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        acnt = ibm_db.fetch_assoc(stmt)
        print(acnt)

        if acnt:
            mg = 'Account already exits!!'

        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            mg = 'Please enter the avalid email address'
        elif not re.match(r'[A-Za-z0-9]+', username):
            ms = 'name must contain only character and number'
        else:
            insert_sql = 'INSERT INTO users (USERNAME,EMAIL,PASSWORD) VALUES (?,?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, username)
            ibm_db.bind_param(pstmt, 2, email)
            ibm_db.bind_param(pstmt, 3, pw)
            print(pstmt)
            ibm_db.execute(pstmt)
            mg = 'You have successfully registered click login!'
            message = Mail(
                from_email=os.environ.get('MAIL_DEFAULT_SENDER'),
                to_emails=email,
                subject='New SignUp',
                html_content='<p>Hello, Your Registration was successfull. <br><br> Thank you for
choosing us.</p>')

            sg = SendGridAPIClient(
                api_key=os.environ.get('SENDGRID_API_KEY'))

            response = sg.send(message)
            print(response.status_code, response.body)
            return render_template("login.html", meg=mg)

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

```
        msg = "fill out the form first!"
    return render_template("signup.html", meg=mg)


@app.route('/dashboard', methods=['POST', 'GET'])
@login_required
def dashBoard():
    sql = "SELECT * FROM stocks"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_assoc(stmt)
    stocks = []
    headings = [*dictionary]
    while dictionary != False:
        stocks.append(dictionary)

        dictionary = ibm_db.fetch_assoc(stmt)

    return render_template("dashboard.html", headings=headings, data=stocks)


@app.route('/addstocks', methods=['POST'])
@login_required
def addStocks():
    if request.method == "POST":
        print(request.form['item'])
        try:
            item = request.form['item']
            quantity = request.form['quantity']
            price = request.form['price']
            total = int(price) * int(quantity)
            insert_sql = 'INSERT INTO stocks (NAME,QUANTITY,PRICE_PER_QUANTITY,TOTAL_PRICE)
VALUES (?,?,?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.bind_param(pstmt, 2, quantity)
            ibm_db.bind_param(pstmt, 3, price)
            ibm_db.bind_param(pstmt, 4, total)
            ibm_db.execute(pstmt)

        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('dashBoard'))
```

```
@app.route('/updatestocks', methods=['POST'])
@login_required
def UpdateStocks():
    if request.method == "POST":
        try:
            item = request.form['item']
            print("hello")
            field = request.form['input-field']
            value = request.form['input-value']
            print(item, field, value)
            insert_sql = 'UPDATE stocks SET ' + field + "= ?" + " WHERE NAME=?"
            print(insert_sql)
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, item)
            ibm_db.execute(pstmt)
            if field == 'PRICE_PER_QUANTITY' or field == 'QUANTITY':
                insert_sql = 'SELECT * FROM stocks WHERE NAME= ?'
                pstmt = ibm_db.prepare(conn, insert_sql)
                ibm_db.bind_param(pstmt, 1, item)
                ibm_db.execute(pstmt)
                dictonary = ibm_db.fetch_assoc(pstmt)
                print(dictonary)
                total = dictonary['QUANTITY'] * dictonary['PRICE_PER_QUANTITY']
                insert_sql = 'UPDATE stocks SET TOTAL_PRICE=? WHERE NAME=?'
                pstmt = ibm_db.prepare(conn, insert_sql)
                ibm_db.bind_param(pstmt, 1, total)
                ibm_db.bind_param(pstmt, 2, item)
                ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('dashBoard'))


@app.route('/deletestocks', methods=['POST'])
@login_required
def deleteStocks():
    if request.method == "POST":
        print(request.form['item'])
        try:
            item = request.form['item']
            insert_sql = 'DELETE FROM stocks WHERE NAME=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
```

```
            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('dashBoard'))


@app.route('/update-user', methods=['POST', 'GET'])
@login_required
def updateUser():
    if request.method == "POST":
        try:
            email = session['id']
            field = request.form['input-field']
            value = request.form['input-value']
            insert_sql = 'UPDATE Client SET ' + field + '= ? WHERE EMAIL=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, email)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('profile'))


@app.route('/update-password', methods=['POST', 'GET'])
@login_required
def updatePassword():
    if request.method == "POST":
        try:
            email = session['id']
            password = request.form['prev-password']
            curPassword = request.form['cur-password']
            confirmPassword = request.form['confirm-password']
            insert_sql = 'SELECT * FROM  Client WHERE EMAIL=? AND PASSWORD=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, email)
            ibm_db.bind_param(pstmt, 2, password)
            ibm_db.execute(pstmt)
            dictionary = ibm_db.fetch_assoc(pstmt)
            print(dictionary)
```

```
            if curPassword == confirmPassword:
                insert_sql = 'UPDATE users SET PASSWORD=? WHERE EMAIL=?'
                pstmt = ibm_db.prepare(conn, insert_sql)
                ibm_db.bind_param(pstmt, 1, confirmPassword)
                ibm_db.bind_param(pstmt, 2, email)
                ibm_db.execute(pstmt)
        except Exception as e:
            msg = e
        finally:
            # print(msg)
            return render_template('result.html')


@app.route('/orders', methods=['POST', 'GET'])
@login_required
def orders():
    query = "SELECT * FROM orders"
    stmt = ibm_db.exec_immediate(conn, query)
    dictionary = ibm_db.fetch_assoc(stmt)
    orders = []
    headings = [*dictionary]
    while dictionary != False:
        orders.append(dictionary)
        dictionary = ibm_db.fetch_assoc(stmt)
    return render_template("orders.html", headings=headings, data=orders)


@app.route('/createOrder', methods=['POST'])
@login_required
def createOrder():
    if request.method == "POST":
        try:
            stock_id = request.form['stock_id']
            query = 'SELECT PRICE_PER_QUANTITY FROM stocks WHERE ID= ?'
            stmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(stmt, 1, stock_id)
            ibm_db.execute(stmt)
            dictionary = ibm_db.fetch_assoc(stmt)
            if dictionary:
                quantity = request.form['quantity']
                date = str(datetime.now().year) + "-" + str(
                    datetime.now().month) + "-" + str(datetime.now().day)
                delivery = datetime.now() + timedelta(days=7)
                delivery_date = str(delivery.year) + "-" + str(
                    delivery.month) + "-" + str(delivery.day)
                price = float(quantity) * \
                    float(dictionary['PRICE_PER_QUANTITY'])
```

```
                query = 'INSERT INTO orders (STOCKS_ID,QUANTITY,DATE,DELIVERY_DATE,PRICE) VALUES
(?,?,?,?,?)'
                pstmt = ibm_db.prepare(conn, query)
                ibm_db.bind_param(pstmt, 1, stock_id)
                ibm_db.bind_param(pstmt, 2, quantity)
                ibm_db.bind_param(pstmt, 3, date)
                ibm_db.bind_param(pstmt, 4, delivery_date)
                ibm_db.bind_param(pstmt, 5, price)
                ibm_db.execute(pstmt)
        except Exception as e:
            print(e)

        finally:
            return redirect(url_for('orders'))


@app.route('/updateOrder', methods=['POST'])
@login_required
def updateOrder():
    if request.method == "POST":
        try:
            item = request.form['item']
            field = request.form['input-field']
            value = request.form['input-value']
            query = 'UPDATE orders SET ' + field + "= ?" + " WHERE ID=?"
            pstmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            print(e)

        finally:
            return redirect(url_for('orders'))


@app.route('/cancelOrder', methods=['POST'])
@login_required
def cancelOrder():
    if request.method == "POST":
        try:
            order_id = request.form['order_id']
            query = 'DELETE FROM orders WHERE ID=?'
            pstmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(pstmt, 1, order_id)
            ibm_db.execute(pstmt)
        except Exception as e:
```

```
            print(e)

        finally:
            return redirect(url_for('orders'))


@app.route('/suppliers', methods=['POST', 'GET'])
@login_required
def suppliers():
    sql = "SELECT * FROM suppliers"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_assoc(stmt)
    suppliers = []
    orders_assigned = []
    headings = [*dictionary]
    while dictionary != False:
        suppliers.append(dictionary)
        orders_assigned.append(dictionary['ORDER_ID'])
        dictionary = ibm_db.fetch_assoc(stmt)

# get order ids from orders table and identify unassigned order ids
    sql = "SELECT ID FROM orders"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_assoc(stmt)
    order_ids = []
    while dictionary != False:
        order_ids.append(dictionary['ID'])
        dictionary = ibm_db.fetch_assoc(stmt)

    unassigned_order_ids = set(order_ids) - set(orders_assigned)
    return
render_template("suppliers.html",headings=headings,data=suppliers,order_ids=unassigned_order_ids)

@app.route('/updatesupplier', methods=['POST'])
@login_required
def UpdateSupplier():
    if request.method == "POST":
        try:
            item = request.form['name']
            field = request.form['input-field']
            value = request.form['input-value']
            print(item, field, value)
            insert_sql = 'UPDATE suppliers SET ' + field + "= ?" + " WHERE NAME=?"
            print(insert_sql)
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, item)
```

```
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            return redirect(url_for('suppliers'))

@app.route('/addsupplier', methods=['POST'])
@login_required
def addSupplier():
    if request.method == "POST":
        try:
            name = request.form['name']
            order_id = request.form.get('order-id-select')
            print(order_id)
            print("Hello world")
            location = request.form['location']
            insert_sql = 'INSERT INTO suppliers (NAME,ORDER_ID,LOCATION) VALUES (?,?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, name)
            ibm_db.bind_param(pstmt, 2, order_id)
            ibm_db.bind_param(pstmt, 3, location)
            ibm_db.execute(pstmt)

        except Exception as e:
            msg = e

        finally:
            return redirect(url_for('suppliers'))

@app.route('/deletesupplier', methods=['POST'])
@login_required
def deleteSupplier():
    if request.method == "POST":
        try:
            item = request.form['name']
            insert_sql = 'DELETE FROM suppliers WHERE NAME=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            return redirect(url_for('suppliers'))
@app.route('/profile', methods=['POST', 'GET'])
@login_required
```

```
def profile():
    if request.method == "GET":
        try:
            email = session['id']
            insert_sql = 'SELECT * FROM users WHERE EMAIL=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, email)
            ibm_db.execute(pstmt)
            dictionary = ibm_db.fetch_assoc(pstmt)
            print(dictionary)
        except Exception as e:
            msg = e
        finally:
            # print(msg)
            return render_template("profile.html", data=dictionary)


@app.route('/logout', methods=['GET'])
@login_required
def logout():
    print(request)
    resp = make_response(render_template("login.html"))
    session.clear()
    return resp


if __name__ == '__main__':
    app.run(debug=True)
```

html= '" <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:o="urn:schemas-microsoft-com:office:office">

<head>

<!--[if gte mso 9]>

<xml>

<o:OfficeDocumentSettings>

<o:AllowPNG/>

```
    <o:PixelsPerInch>96</o:PixelsPerInch>
  </o:OfficeDocumentSettings>
</xml>
<![endif]-->
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="x-apple-disable-message-reformatting">
  <!--[if !mso]><!-->
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <!--<![endif]-->
  <title></title>

  <style type="text/css">
    @media only screen and (min-width: 620px) {
      .u-row {
        width: 600px !important;
      }
      .u-row .u-col {
        vertical-align: top;
      }
      .u-row .u-col-100 {
        width: 600px !important;
      }
    }

    @media (max-width: 620px) {
      .u-row-container {
        max-width:  100%  !important;
        padding-left:  0px  !important;
        padding-right: 0px !important;
      }
      .u-row .u-col {
        min-width: 320px !important;
        max-width: 100% !important;
```

```
    display: block !important;
  }
  .u-row {
    width: calc(100% - 40px) !important;
  }
  .u-col {
    width: 100% !important;
  }
  .u-col>div {
    margin: 0 auto;
  }
}

body {
  margin: 0;
  padding: 0;
}

table,
tr,
td {
  vertical-align: top;
  border-collapse: collapse;
}

p {
  margin: 0;
}

.ie-container table,
.mso-container table {
  table-layout: fixed;
}
```

```
        * {
          line-height: inherit;
        }

        a[x-apple-data-detectors='true'] {
          color: inherit !important;
          text-decoration: none !important;
        }

        table,
        td {
          color: #000000;
        }

        @media (max-width: 480px) {
          #u_column_3 .v-col-background-color {
          background-color: #3598db !important;
          }
        }
      </style>



      <!--[if !mso]><!-->
      <link       href="https://fonts.googleapis.com/css?family=Cabin:400,700"       rel="stylesheet"
type="text/css">
      <!--<![endif]-->

    </head>

    <body class="clean-body u_body" style="margin: 0;padding: 0;-webkit-text-size-adjust:
100%;background-color: #f9f9f9;color: #000000">
      <!--[if IE]><div class="ie-container"><![endif]-->
      <!--[if mso]><div class="mso-container"><![endif]-->
```

<table style="border-collapse: collapse;table-layout: fixed;border-spacing: 0;mso-table-lspace: 0pt;mso-table-rspace: 0pt;vertical-align: top;min-width: 320px;Margin: 0 auto;background-color: #f9f9f9;width:100%" cellpadding="0" cellspacing="0">
<tbody>
<tr style="vertical-align: top">
<td style="word-break: break-word;border-collapse: collapse !important;vertical-align: top">
<!--[if (mso)|(IE)]><table width="100%" cellpadding="0" cellspacing="0" border="0"><tr><td align="center" style="background-color: #f9f9f9;"><![endif]-->

<!--[if (mso)|(IE)]><table width="100%" cellpadding="0" cellspacing="0" border="0"><tr><td style="padding: 0px;background-color: transparent;" align="center"><table cellpadding="0" cellspacing="0" border="0" style="width:600px;"><tr style="background-color: #ffffff;"><![endif]-->

<!--[if (mso)|(IE)]><td align="center" width="600" class="v-col-background-color" style="width: 600px;padding: 0px;border-top: 0px solid transparent;border-left: 0px solid transparent;border-right: 0px solid transparent;border-bottom: 0px solid transparent;" valign="top"><![endif]-->

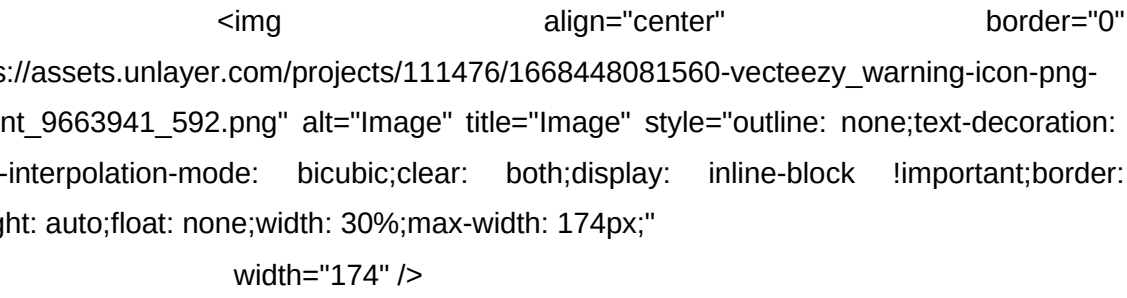
<!--[if (!mso)&(!IE)]><!-->

```
                    <!--<![endif]-->

                    <table style="font-family:'Cabin',sans-serif;" role="presentation"
cellpadding="0" cellspacing="0" width="100%" border="0">
                      <tbody>
                        <tr>
                          <td style="overflow-wrap:break-word;word-break:break-
word;padding:0px;font-family:'Cabin',sans-serif;" align="left">

                            <table width="100%" cellpadding="0" cellspacing="0" border="0">
                              <tr>
                                <td style="padding-right: 0px;padding-left: 0px;" align="right">

                                  <img align="right" border="0"
src="https://assets.unlayer.com/projects/111476/1668447888843-Warning-cuate.png"
alt="Image" title="Image" style="outline: none;text-decoration: none;-ms-interpolation-mode:
bicubic;clear: both;display: inline-block !important;border: none;height: auto;float: none;width:
100%;max-width: 600px;"
                                    width="600" />

                                </td>
                              </tr>
                            </table>

                          </td>
                        </tr>
                      </tbody>
                    </table>

                    <!--[if (!mso)&(!IE)]><!-->
                  </div>
                  <!--<![endif]-->
                </div>
              </div>
```

```
            <!--[if (mso)|(IE)]></td><![endif]-->
            <!--[if (mso)|(IE)]></tr></table></td></tr></table><![endif]-->
          </div>
        </div>
      </div>



      <div class="u-row-container" style="padding: 0px;background-color: transparent">
        <div class="u-row" style="Margin: 0 auto;min-width: 320px;max-width: 600px;overflow-wrap: break-word;word-wrap: break-word;word-break: break-word;background-color: #003399;">
          <div style="border-collapse: collapse;display: table;width: 100%;height: 100%;background-color: transparent;">
            <!--[if (mso)|(IE)]><table width="100%" cellpadding="0" cellspacing="0" border="0"><tr><td style="padding: 0px;background-color: transparent;" align="center"><table cellpadding="0" cellspacing="0" border="0" style="width:600px;"><tr style="background-color: #003399;"><![endif]-->

            <!--[if (mso)|(IE)]><td align="center" width="600" class="v-col-background-color" style="background-color: #3598db;width: 600px;padding: 0px;border-top: 0px solid transparent;border-left: 0px solid transparent;border-right: 0px solid transparent;border-bottom: 0px solid transparent;" valign="top"><![endif]-->
            <div id="u_column_3" class="u-col u-col-100" style="max-width: 320px;min-width: 600px;display: table-cell;vertical-align: top;">
              <div class="v-col-background-color" style="background-color: #3598db;height: 100%;width: 100% !important;">
                <!--[if (!mso)&(!IE)]><!-->
                <div style="height: 100%; padding: 0px;border-top: 0px solid transparent;border-left: 0px solid transparent;border-right: 0px solid transparent;border-bottom: 0px solid transparent;">
                  <!--<![endif]-->
```

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%" border="0">
<tbody>
<tr>
<td style="overflow-wrap:break-word;word-break:break-word;padding:40px 10px 10px;font-family:'Cabin',sans-serif;" align="left">

<table width="100%" cellpadding="0" cellspacing="0" border="0">
<tr>
<td style="padding-right: 0px;padding-left: 0px;" align="center">



</td>
</tr>
</table>

</td>
</tr>
</tbody>
</table>

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%" border="0">
<tbody>
<tr>
<td style="overflow-wrap:break-word;word-break:break-word;padding:10px;font-family:'Cabin',sans-serif;" align="left">

WARNING

</td>
</tr>
</tbody>
</table>

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%" border="0">
<tbody>
<tr>
<td style="overflow-wrap:break-word;word-break:break-word;padding:0px 10px 31px;font-family:'Cabin',sans-serif;" align="left">

You are Out of Stock !

</td>
</tr>
</tbody>
</table>

<!--[if (!mso)&(!IE)]><!-->

```
        <!--<![endif]-->
      </div>
    </div>
    <!--[if (mso)|(IE)]></td><![endif]-->
    <!--[if (mso)|(IE)]></tr></table></td></tr></table><![endif]-->
   </div>
  </div>
</div>


<div class="u-row-container" style="padding: 0px;background-color: transparent">
  <div class="u-row" style="Margin: 0 auto;min-width: 320px;max-width: 600px;overflow-wrap: break-word;word-wrap: break-word;word-break: break-word;background-color: #ffffff;">
    <div style="border-collapse: collapse;display: table;width: 100%;height: 100%;background-color: transparent;">
      <!--[if (mso)|(IE)]><table width="100%" cellpadding="0" cellspacing="0" border="0"><tr><td style="padding: 0px;background-color: transparent;" align="center"><table cellpadding="0" cellspacing="0" border="0" style="width:600px;"><tr style="background-color: #ffffff;"><![endif]-->

      <!--[if (mso)|(IE)]><td align="center" width="600" class="v-col-background-color" style="background-color: #000000;width: 600px;padding: 0px;border-top: 0px solid transparent;border-left: 0px solid transparent;border-right: 0px solid transparent;border-bottom: 0px solid transparent;" valign="top"><![endif]-->
      <div class="u-col u-col-100" style="max-width: 320px;min-width: 600px;display: table-cell;vertical-align: top;">
        <div class="v-col-background-color" style="background-color: #000000;height: 100%;width: 100% !important;">
          <!--[if (!mso)&(!IE)]><!-->
          <div style="height: 100%; padding: 0px;border-top: 0px solid transparent;border-left: 0px solid transparent;border-right: 0px solid transparent;border-bottom: 0px solid transparent;">
```

<!--<![endif]-->

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%" border="0">
<tbody>
<tr>
<td style="overflow-wrap:break-word;word-break:break-word;padding:33px 55px;font-family:'Cabin',sans-serif;" align="left">

Please order new stocks to get rid of the out-of-stock.

</td>
</tr>
</tbody>
</table>

<table style="font-family:'Cabin',sans-serif;" role="presentation" cellpadding="0" cellspacing="0" width="100%" border="0">
<tbody>
<tr>
<td style="overflow-wrap:break-word;word-break:break-word;padding:33px 55px 60px;font-family:'Cabin',sans-serif;" align="left">

Post queries in the Contact Support for further clearance!

Thank you!

</td>
</tr>
</tbody>
</table>

<!--[if (!mso)&(!IE)]><!-->

<!--<![endif]-->

<!--[if (mso)|(IE)]></td><![endif]-->
<!--[if (mso)|(IE)]></tr></table></td></tr></table><![endif]-->

<!--[if (mso)|(IE)]></td></tr></table><![endif]-->
</td>
</tr>
</tbody>
</table>
<!--[if mso]> <![endif]-->
<!--[if IE]> <![endif]-->
</body>

```
        </html>

'''
```

Dockerfile

```
FROM python:3.10.6
WORKDIR /app
COPY requirements.txt ./
RUN pip install -r requirements.txt
COPY . .
EXPOSE 5000
CMD ["python","./app.py"]
```

requirements.txt

```
flask
ibm_db
pandas
```

flask_service.yaml:

```
apiVersion: v1
kind: Service
metadata:
  name: flask-app-service
spec:
  type: NodePort
  ports:
    - port: 5000
  selector:
    app: ims-final
```

flask_ingress.yaml:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: flask-app-ingress
  annotations:
  kubernetes.io/ingress.class: nginx
    nginx.ingress.kubernetes.io/ssl-redirect: "false"

spec:
  # ingressClassName: nginx
  rules:
    - http:
        paths:
```

```
        - backend:
            service:
              name: flask-app-service
              port:
                number: 5000
          path: /
          pathType: Prefix
```

ibm_deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ims-final

spec:
  replicas: 3
  selector:
    matchLabels:
      app: ims-final
  template:
    metadata:
      labels:
        app: ims-final

    spec:
      containers:
        - name: job-portal-container
          image: jp.icr.io/padmanaban/ims-final
          imagePullPolicy: Always
          ports:
            - containerPort: 5000
              protocol: TCP
```