INVENTORY MANAGEMENT SYSTEM FOR REATAILERS

A PROJECT REPORT

Submitted By

GANESH L	**210819205008**
GOWRI SANKAR P	**210819205010**
HARI HARAN R S	**910719205008**
KUGAN K	**210819205027**
MUTHESWARAN D	**210819205702**

In partial fulfillment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

KINGS ENGINEERING COLLEGE,IRUNGATTUKOTAI

ANNA UNIVERSITY:CHENNAI 600025

November 2022

ANNAUNIVERSITY:CHENNAI600025BON AFIDECERTIFICATE

Certified that mini project report **" INVENTORY MANAGEMENT SYSTEM FOR RETAILERS "** is bonafide work of **"GANESH L, GOWRI SANKAR P, HARIHARAN R S, KUGAN K, MUTHESHWARAN D"** who carried out this mini project work under my supervision.

SIGNATURE

Dr.G. MANIKANDAN

HEADOFTHEDEPARTMENT

Professor

Deportment of information Technology,

Kings Engineering College,

Irungattukottai,

Chennai-602117

SIGNATURE

Mrs. K. SARANYA

MENTOR

Assistant Professor

Deportment of Information Technology,

Kings Engineering College,

Irungattukottai,

Chennai-602117.

ACKNOWLEDGEMENT

We thank God for his blessings and also forgiving as good knowledge and strength in enabling us to finish our project. Our deep gratitude goes to our founder late **Dr.D.SELVARAJ,M.A.,M.Phil.,** for his patronage in the completion of our project.We like to take this opportunity to thank our honourable chairperson **Dr.S. NALINI SELVARAJ, M.COM., MPhil., Ph.D.**and honourable director, **MR.S. AMIRTHARAJ, M.Tech., M.B.A** for their support given to us to finish our project successfully. We wish to express our sincere thanks to our beloved principal.**Dr.T. JOHN ORAL BASKAR., M.E.,Ph.D** for his kind encouragement and his interest towards us.

We are extremely grateful and thanks to our professor **Dr.G.MANIKANDAN,** head of the Information Technology department, Kings Engineering College, for his valuable suggestion ,guidance and encouragement. We wish to express our sense of gratitude to our project Mentor **Mrs. K. SARANYA,** Assistant Professor of Information Technology Department, Kings Engineering College and our project Evaluator **Mr. B. MUTHAZAGHAN,** Associate Professor of Information Technology Department, Kings Engineering College, We express our sincere thanks to our parents ,friends and staff members, who have helped and encouraged us during the entire course of completing this project work successfully.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The problem faced by the retailers is that they do not have any system to record and keep their inventory data. It is difficult for the owner to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized. Inventory management facilitates the smooth functioning of your business and enhances sales, promotes cost-effectiveness, and improves customer experience. Listed below are some of the reasons why businesses need inventory management:

- Managing Finances
- Tracking Inventory
- Avoiding late deliveries
- Managing time and effort
- Predicting future sales
- Enhancing customer loyalty

1.2 PURPOSE

Retail inventory management is the process of ensuring retailers meet customer demand without running out of stock or carrying excess supply. The objective of the project is to create an application that help retailers to track and manage stocks which results in lower costs and a better understanding of sales pattern. By creating an application, retailers can log in to it and can update

inventory details, also users will be able to add new stock by submitting essential details related to the stock. Retailers can also 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.

CHAPTER 2

LITERATURE SURVEY

Inventory Management System for Retailers

1. Lan Teng, Zhenji Zhang, et al, "Integrated Inventory-Transportation Problem in Vendor-Managed Inventory System", 2019.

The paper presents a two-echelon inventory-transportation problem in Vendor Managed Inventory (VMI) system. We consider a distribution system composedwith single supplier, single distribution center and multiple retailers. Single kindof products are required to deliver from the manufacturer through distribution center to the retailers within soft time window. A mixed algorithm is designed to solve the problem with simulated annealing and ant colony with local search.The solution of upper and lower echelon model are substituted into each other based on the mixed algorithm step by step to get the optimization solutions.

2. Soonkyo Lee, Young Joo Kim, et al, "Effects of Yield and Lead-Time Uncertainty on Retailer-Managed and Vendor-Managed Inventory Management", 2019.

Generally, there are various elements of uncertainty in a supply chain. In particular, uncertainties in lead time, demand, and yield are very important in the semiconductor industry. Higher uncertainty can lead to bullwhip effects that can undermine the performance of the entire supply chain. This study examines the relationship between uncertainty in the supply chain and the outcome of inventory replenishment policies. Specifically, we analyze the effects of well- known uncertainties on manufacturer production quantity and retailer order quantity decisions in a decentralized supply chain. Using numerical experiments, a comparative analysis of the two alternatives is conducted to determine suitable options for improving supply chain performance.

3. Lijun Ma, Can Wang, et al, "The Influence of Supply Chain Finance on Inventory Management Under Supply Uncertainty", 2018.

It's well known that small and medium-sized enterprises(SMEs) occupy a significant position in Chinese economy. However, in credit practices, SMEs are often considered as high-risk

lenders who often need to pay higher capital costs to obtain funds. This paper explores (1) debt financing can distort a retailer's inventory decision when the retailer with limited funds and selling multiple products with different price, cost, revenue, and yield uncertainty parameters; (2) we also explore the role of each parameter in this distortion.

Because of the limited liability, a debt-financed retailer prefers items with high

elling price, high penalty factor(late delivery) and low salvage value. Furthermore, based on the fact that the capital cost of suppliers has always beenhigher than that of banks, we discuss that this distortion can be mitigated when the financing is provided by the supplier who can observe the actual order quantities before determining the credit terms. On the other hand, based on the fact that the capital cost of suppliers has always been higher than that of banks, we studied the combination of bank and supplier financing to enable retailers to achieve the best way of financing.

4. Lin Li, Zhaojun Yang, et al, ""Buffer Inventory + Information Sharing"Strategy for Retailers in Two-Level Fresh Supply Chain", 2020.

In the supply chain of fresh agricultural products in China, there are huge commodity losses during the transportation and storage of agricultural products due to limitations in cold chain logistics technology, which affects the performance of the fresh supply chain. This paper aims to improve the accuracy in forecasting market demand and reduce inventory by studying the impact of information sharing strategies on inventory and revenue at all levels of the supply chain, by establishing a system dynamics model that analyzes the capability of information sharing to reduce the expected inventory of suppliers and retailers, and by examining the impact of information sharing on demand forecasting accuracy and inventory stability. Results show that the strategy of information sharing combined with setting the buffer inventory can better improve the performance of the fresh produce supply chain.

5. Abhijit Barman, Rubi Das, et al, "Pricing and Inventory Decisions of Multi-item Deteriorating Inventory System under Stock, Time and Price Sensitive Demand Policy", 2021.

In most of the inventory like food, fashion, electric materials, building materials, a retailer needs to maintain varieties of different inventory items. On the other hand, racks overflowing with a large number of quantities in an inventory attracts the attention of more customers. Besides, the selling price is acrucial factor in demand based on marketing and economic theory. Assimilatingall these variates, the present paper advocates a multi-item single-period inventory

model that generalizes the pricing and inventory policy for instantaneous deteriorating items. An iterative algorithm has been incorporated to find the optimal procedure. The prime objective of this model is to determine the selling price, time length up to zero inventory, optimal lot size so that the profit of the retailer will be maximized. The model is demonstrated with a numerical example which is followed by a sensitivity analysis.

6. Zhang Zhenmin, Li Lin, “Perishables Inventory Management Model with Backroom Effect”, 2020.

This paper considers two storage locations (shelf and backroom) in supermarket selling perishable products. Due to the backroom effect, the products with higher freshness are periodically replenished from backroom to shelf, where freshness-and-shelf level-sensitive consumers purchase the products according to their "perceived average freshness" of displayed fresh products. Then it develops the decision-making model including shelf replenishment period and reorder point, and the neighborhood search algorithm is designed to solve this model. Finally, it conducts groups of numerical examples revealing the impact of the backroom effect on the retailer's optimal order quantity, shelf replenishment period, and reorder point The main results show that retailers who ignore the backroom effect will miss out on market share and profit margins. When the retailer realizes the existence of the backroom effect, he should increase the order quantity to obtain a higher profit value.

7. Yantong Li, Feng Chu, et al, “Integrated Production Inventory Routing Planning for Intelligent Food Logistics Systems”, 2018.

An intelligent logistics system is an important branch of intelligent transportation systems. It is a great challenge to develop efficient technologies and methodologies to improve its performance in meeting customer requirements while this is highly related to people's life quality. Its high efficiency can reduce food waste, improve food quality and safety, and enhance the competitiveness of food companies. In this paper, we investigate a new integrated planning problem for intelligent food logistics systems. Two objectives are considered: minimizing total production, inventory, and transportation cost and maximizing average food quality. For the problem, a bi-objective mixed integer linear programming model is formulated first.

Computational results on a case study and on 185 randomly generated instanceswith up to 100 retailers and 12 periods show the effectiveness and efficiency ofthe proposed method.

8. Ji Quan, Xiaofeng Wang, et al, “Effects of Consumers’ Strategic Behavior and Psychological Satisfaction on the Retailer’s Pricing and Inventory Decisions”, 2019.

This paper introduces a concept of psychological satisfaction to describe the utility of customers under different psychological perception. The following conclusions are got in this study. (i) Compared to psychological neutral strategicconsumers, the behavior of the psychological elation strategic consumers will

further induce the retailer to lower price and reduce inventory, thereby further damaging the profits of the retailer. And the stronger the emotion of psychological elation, the greater the loss of damage. (ii) The behavior of disappointment aversion strategic consumers will alleviate the adverse effects oftheir strategic behavior on the retailer's profit to a certain extent. And the stronger the disappointment aversion emotion, the more obvious the alleviating effect it has. (iii) Considering the effect of consumers' psychological satisfaction. The profit of the retailer is positively correlated with the valuation of consumers, but the relationship between the retailer's profit and product cost or the salvage price depends on other parameters.

9. Michael V. Basin, Fernando Guerra-Avellaneda, et al, “Stock Management Problem: Adaptive Fixed-Time Convergent Continuous Controller Design”, 2021.

This paper presents an adaptive fixed-time convergent continuous controller designed to solve a stock management problem with the objective to drive stock and supply chain levels at the reference values, subject to loss rate disturbances whose bounds are unknown. The only measurable state of the supply chain is the inventory retailer stock level, whereas the supply line inventory level should be estimated. The designed controller includes a fixed-time convergent differentiator, an adaptive fixed-time convergent disturbance observer, and a fixed-time convergent regulator. The controller design is validated in a case study of stock management. The calculated upper estimate for the total settling (convergence) time and the obtained simulation results confirm the fixed-time convergence and the robustness of the designed controller.

10. Wanying Jia, Zhencai Wu, “Impacts of consumer market search behavior on retailers' decision-making: a CVaR analysis”, 2019.

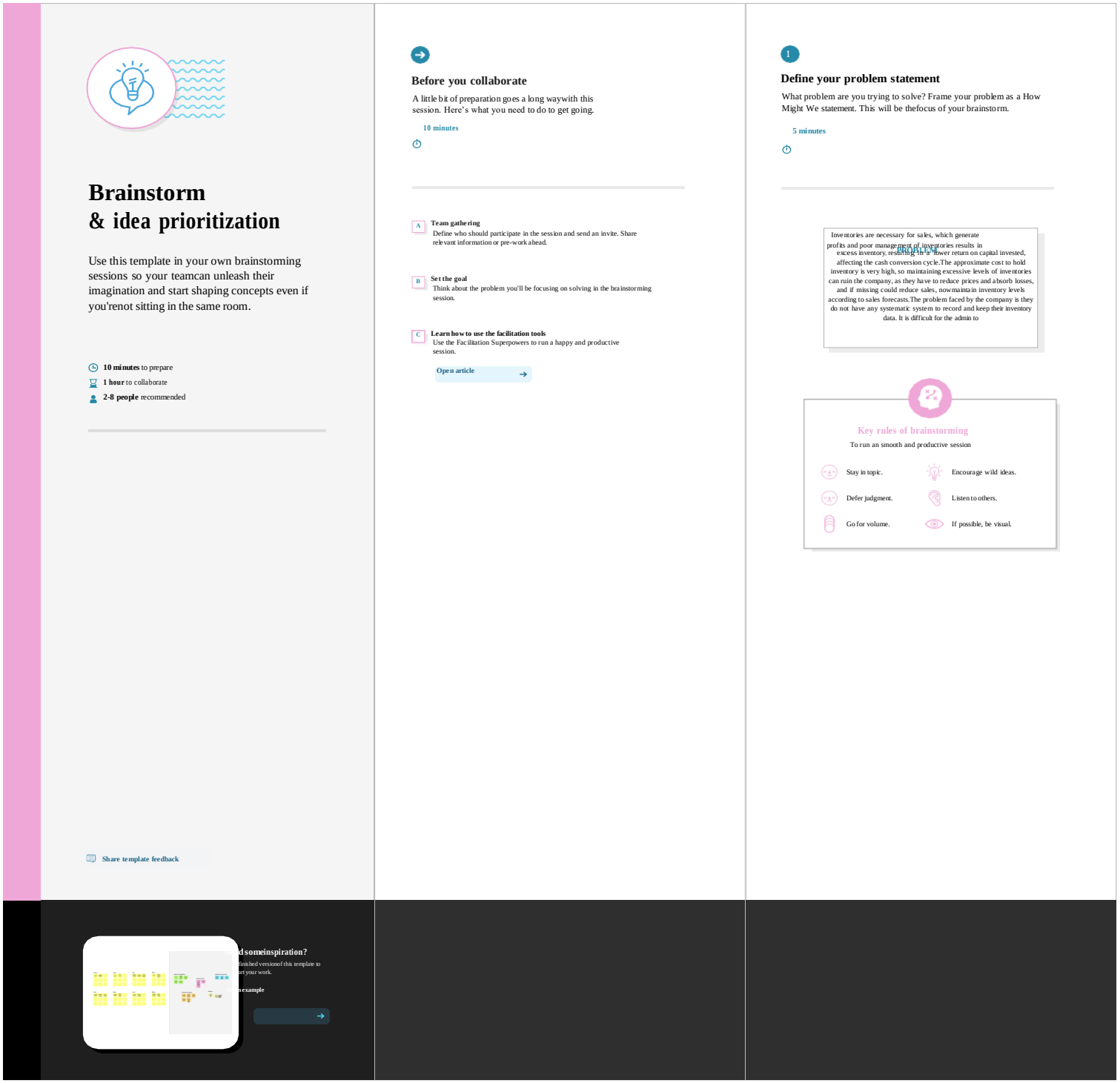
In recent years, the impact of human behavioral issues on the supply chain decision making is to arouse higher attention, such as risk attitude and so on. Behavioral factors, such as risk aversion, can directly influence a manager's procurement decisions. In this paper, we focus on

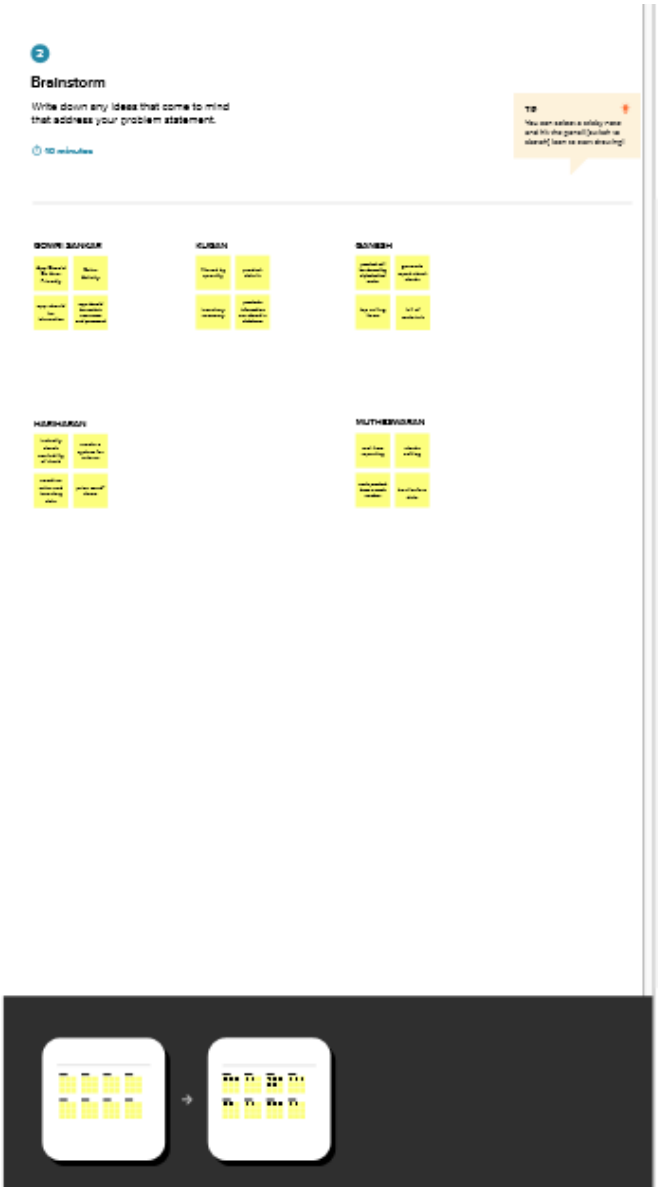
the impact of a retailer's risk aversion on its decision making considering the consumer market search behavior. We use the CVaR to evaluate retailer's risk preference and select a single product supply chain system, which consists of one manufacturer and two retailers. By the theoretical analysis, we find that the customer market search behavior can promote the retailers' order quantity, for there is probability that retailers are in short supply. So the retailer with risk aversion should make a trade-off and we explore the bilateral influence of risk attitude and shortage penalty.

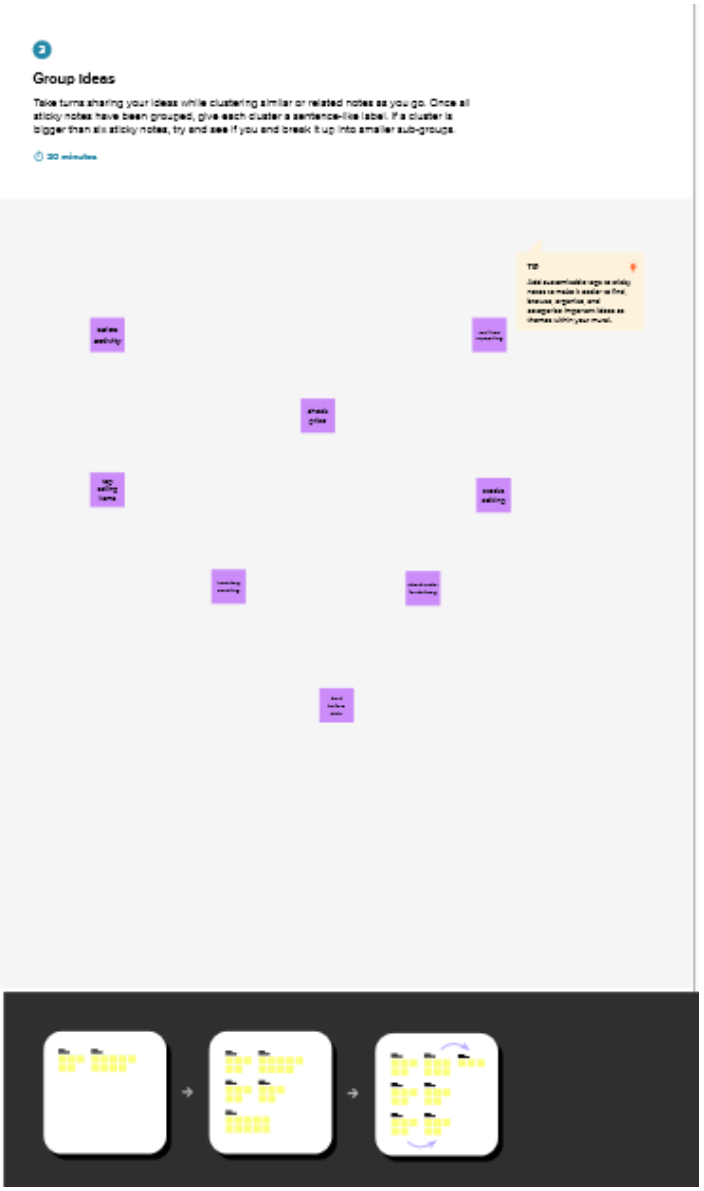
CHAPTER 3

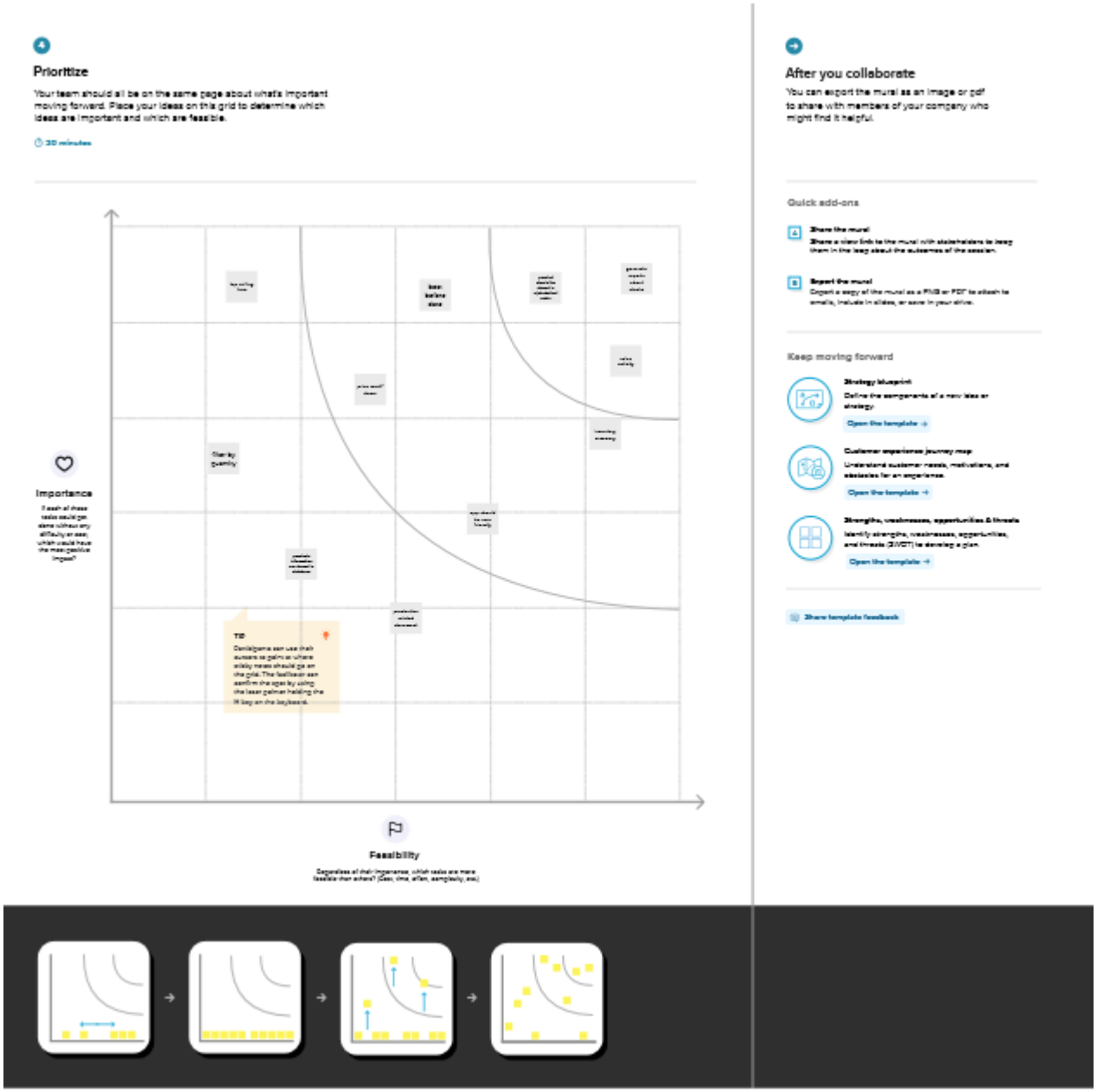
IDEATION & PURPOSE SOLUTION

3.1 IDEATION & BRAINSTOMING



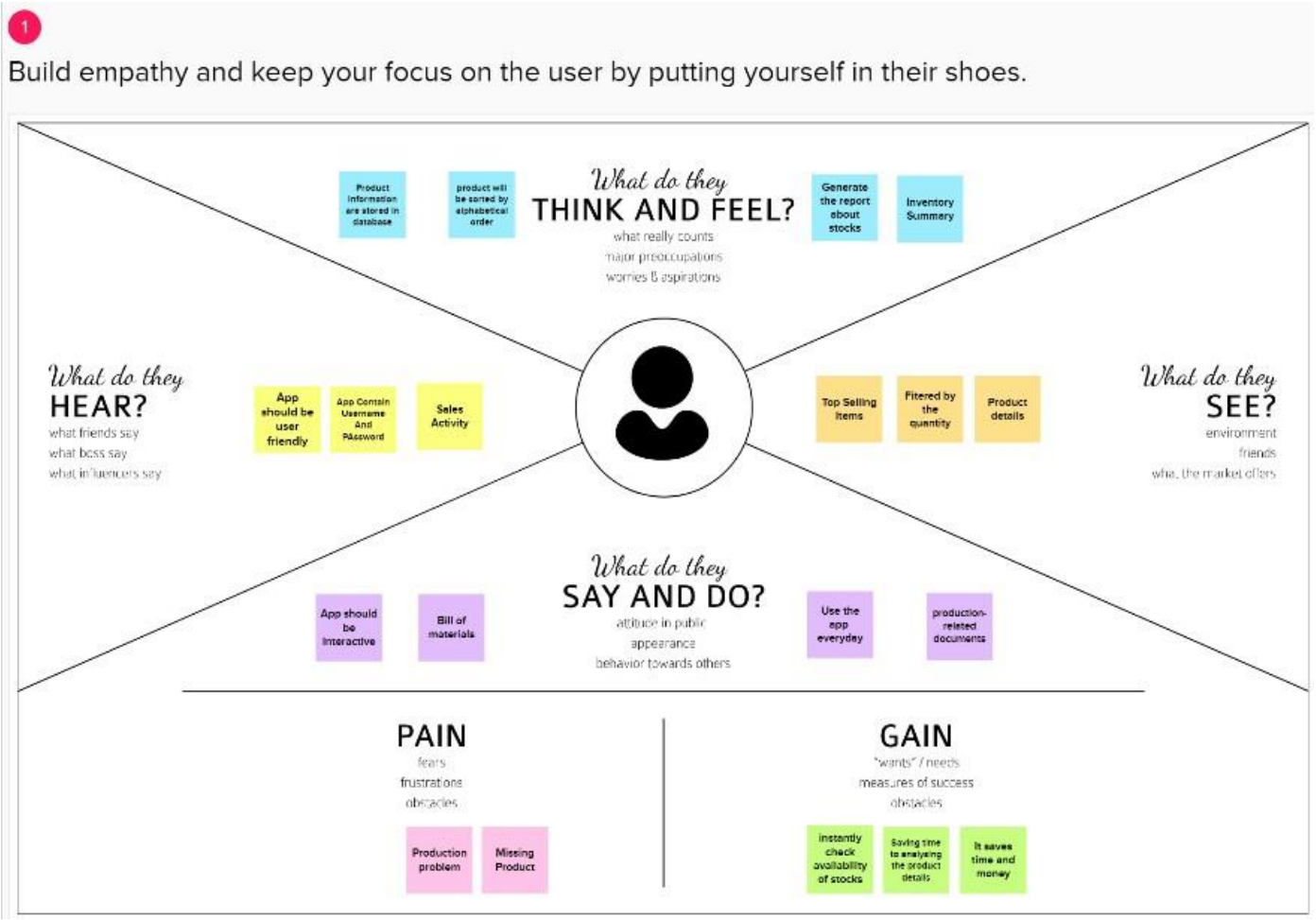

2
Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
TIP
GOWTHAMAN
KUMAN
SANJEEV
MURUGAN
MUTHESWARAN


3
Group Ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes
TIP


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.
30 minutes
Importance
Feasibility
TIP
After you collaborate
You can export the mural as an image or pdf to share with members of your company who might find it helpful.
Quick add-ons
Share the mural
Export the mural
Keep moving forward
Strategy blueprint
Define the components of a new idea or strategy
Open the template
Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience
Open the template
Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan
Open the template
Share template feedback

EMPATHY MAP:

Date	19 September 2022
Team ID	PNT2022TMID25500
Project Name	Project - inventory management system for retailers
Maximum Marks	4 Marks



Proposed Solution

Date	19 September 2022
Team ID	PNT2022TMID25500
Project Name	Project - Inventory Management System For Retailers
Maximum Marks	2 Marks

Proposed Solution Template:

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

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	❖ Stock details and management is a big task ❖ Due to human error, Lack of interest and ❖ consciousness ❖ As there is no proper inventory Managementsolutions known to them
2.	Idea / Solution description	❖ By providing a platform to maintain a proper account of the product details ❖ The application allows the Retailer to know all the present time available stocks ❖ The system will notify or alert the retailer over the expiry date of the products.
3.	Novelty / Uniqueness	❖ Notifications will be sent to the retailers if any product that the customers have been looking for is not available so that the product can be stocked up soon. ❖ Certain machine learning algorithms are used to predict the seasonal high selling products which can be made available during that time.
4.	Social Impact / Customer Satisfaction	● The retailer will be highly satisfied since the wasting of time while searching for an unavailable product is reduced ● The work load of the retailers will be minimized if the system is automated every day and during every purchase.

5.	Business Model (Revenue Model)	❖ Through this solution we can improve customer satisfaction which leads to more profit as well as proper inventory maintenance will reduce the investment cost as well as the wastage of the product will be avoided. Through this we are going to stock only the demanded products.
6.	Scalability of the Solution	❖ This solution is feasible because we have enough technology to implement this solution. ❖ We have cloud computing for remote accessing and various messagesystem to notify.

Problem Solution Fit

Date	19 September 2022
Team ID	PNT2022TMID25500
Project Name	Project - Inventory Management System For Retailers
Maximum Marks	2 Marks

Customer Problem Statement Template:

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

I am	Describe customer with 3-4 key characteristics - *who are they?*	Describe the customer and their attributes here
I'm trying to	*List their outcome or "job" the care about - what are they trying to achieve?*	List the thing they are trying to achieve here
but	Describe what problems or barriers stand in the way – *what bothers them most?*	Describe the problems or barriers that get in the way here
because	Enter the "root cause" of why the problem or barrier exists – *what needs to be solved?*	Describe the reason the problems or barriers exist
which makes me feel	Describe the emotions from the customer's point of view – *how does it impact them emotionally?*	Describe the emotions the result from experiencing the problems or barriers

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

Example:

Problem Statement (PS)	I am (Customer)	I'm trying to	Because	Which makes me feel
PS-1	Retailer	Manage the inventory	Due to human error, Lack of interest and consciousness	Frustrated

CHAPTER 4

REQUIRMENT ANALYSIS

Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	03 October 2022
Team ID	PNT2022TMID25500
Project Name	Project - Inventory management system for retailers
Maximum Marks	4 Marks

4.1Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Installation	User can install the app from google play store or fromthe website.
FR-2	User Registration	Registration through form registration through Gmail.
FR-3	User Confirmation	Confirmation via Email Confirmation via OTP.
FR-4	User Login	User should login the app with the user's name and password.

4.2 Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The Retailer can analyse the stock availability in the store via the app.
NFR-2	**Security**	It is more secured app.
NFR-3	**Reliability**	compare previous stock details and current stock details.
NFR-4	**Performance**	High level Performance.
NFR-5	**Availability**	the stocks availability are shown.
NFR-6	**Scalability**	Keep Inventory Levels as Low as Possible.

CHAPTER 5

PROJECT DESIGN

Project Design Phase-II

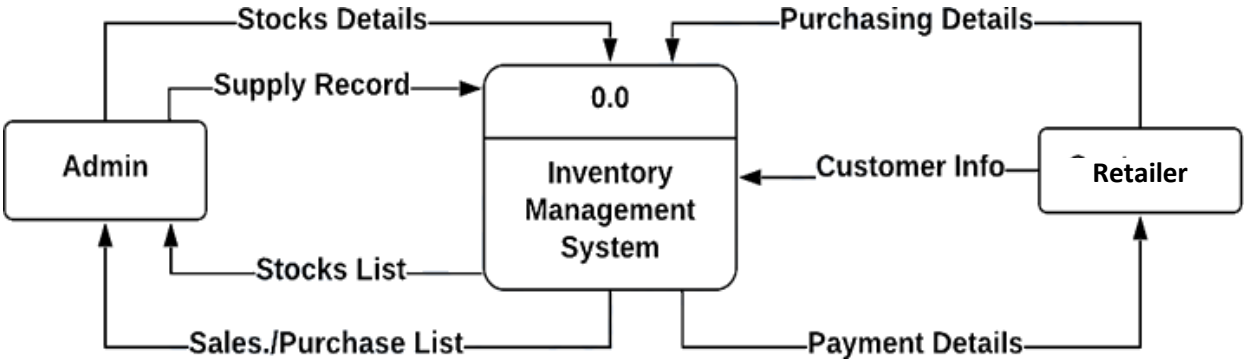
Data Flow Diagram & User Stories

Date	27 October 2022
Team ID	PNT2022TMID25500
Project Name	Project – Inventory Management System forRetailers.
Maximum Marks	4 Marks

Data Flow Diagrams:

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range fromsimple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model anew one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

DATA FLOW DIAGRAM



USER STORIES

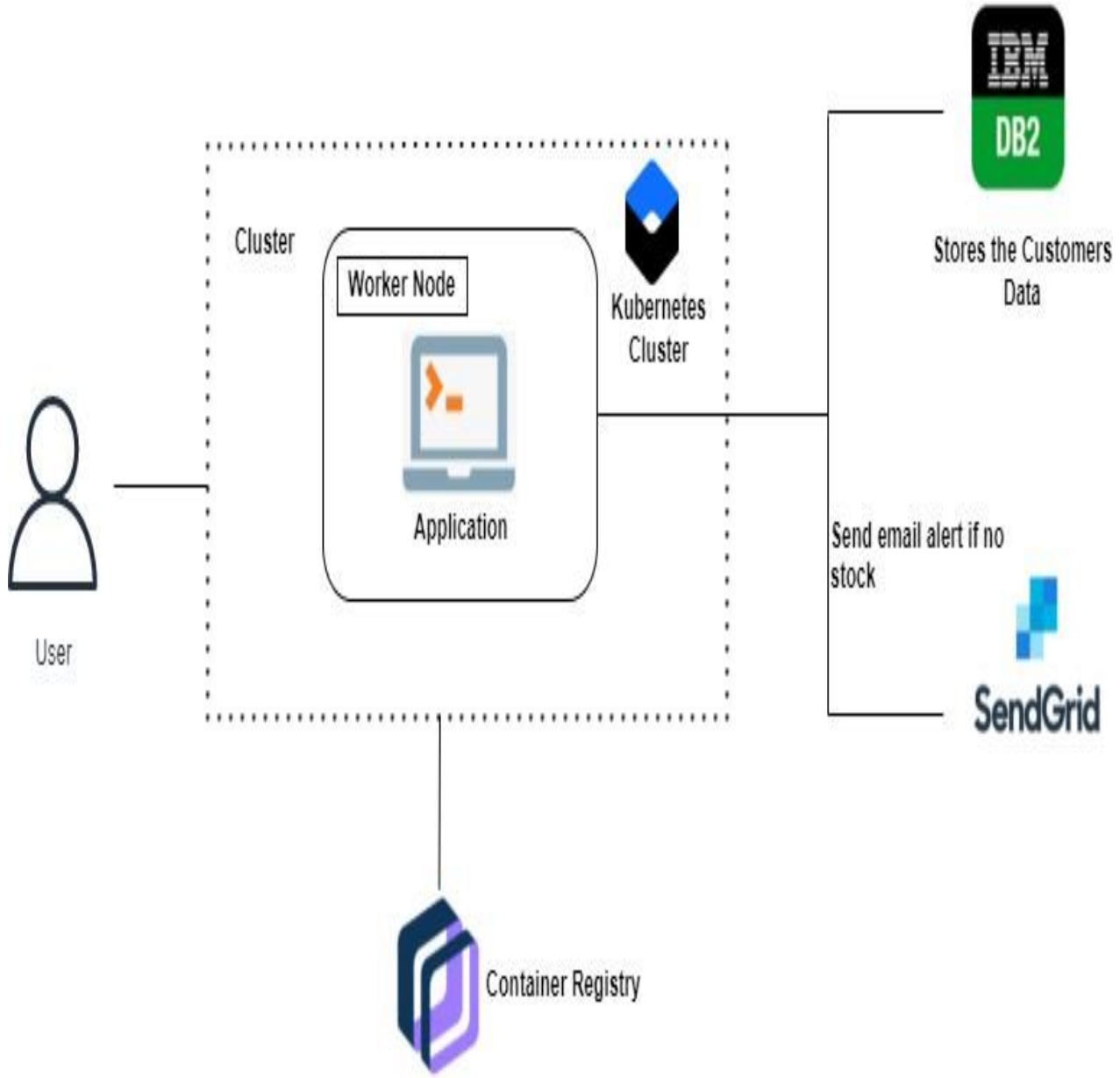
Use the below template to list all the user stories for the product.

User Type	Functional Requirement (EPIC)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Retailer (Mobile user)	Registration	USN-1	As a retailer, I can register for the application by entering my email, password and confirmed password	I can access my account / dashboard	High	Sprint–1
		USN-2	As a retailer, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint–2
		USN-3	As a retailer, I can register for the application through Facebook, Gmail	I can register & access the dashboard with Facebook Login, Gmail Login	Low	Sprint-3
		USN-4	As a retailer, I can register for the application through Gmail, Facebook	I can register for the application through Gmail	Medium	Sprint-2
	Login	USN-5	As a retailer, I can log into the application by entering username & password	I can log in by entering username & password	High	Sprint-1
	Dashboard	USN-6	As a retailer, I can track data of salesof products and inventory levels	I can track data of sales of products and inventory levels	High	Sprint-1
Retailer (Web user)	Registration	USN-7	As a retailer, I can register for the application by entering my email, password and confirmed password	I can access my account / dashboard	High	Sprint–1
		USN-8	As a retailer, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint–2
		USN-9	As a retailer, I can register for the application through Facebook, Gmail	I can register & access the dashboard with Facebook Login, Gmail Login	Low	Sprint-3
		USN-10	As a retailer, I can register for the application through Gmail, Facebook	I can register for the application through Gmail	Medium	Sprint-2
	Login	USN-11	As a retailer, I can log into the application by entering username & password	I can log in by entering username & password	High	Sprint-1
	Dashboard	USN-12	As a retailer, I can track data of salesof products and inventory levels	I can track data of sales of products and inventory levels	High	Sprint-1
Customer Care Extension	Support	USN-13	As a Executive, I provide answersfor the queries asked by retailer	I provide the answers for the queries asked by the users.	High	Sprint-1
Admin	Manage the Stocks	USN-14	As a admin, I manage the stocks y adding, shipping and storing he stocks in the storage units.	I manage the stocks by adding, shipping and storing the stocks in the storage units.	High	Sprint-1

	Control all the users	USN-15	As a admin, I can control all theretails by performing basic CRUD operations.	I can control all the retailers y performing basic CRUD operations.	High	Sprint-1
	Access the database	USN-16	As a admin, I can control and accessthe database	I can control and access the database.	High	Sprint-1

Technical Architecture

Date	12 October 2022
Team ID	PNT2022TMID25500
Project Name	Project – Inventory Management System For Retailers
Maximum Marks	4 Marks



Project Planning Phase

Project Planning Template (Product Backlog, Sprint Planning, Stories, Story points)

Date	18 October 2022
Team ID	PNT2022TMID25500
Project Name	Project - Inventory Management System For Retailes
Maximum Marks	8 Marks

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

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	5	High	Gowri sankar Hari haran Kugan
Sprint-1	Confirmation	USN-2	As a user, I will receive confirmation email once I have registered for the application	2	High	Hari haran Kugan Mutheswaran
Sprint-1	Login	USN-3	As a user, I can register for the applicationthrough Gmail	8	Low	Ganesh Hari haran Mutheswaran
Sprint-1		USN-4	As a user, I can log into the application byentering email & password	4	Medium	Gowri sankar Mutheswaran Ganesh
Sprint-2	Dashboard	USN-5	As a user, I can view the products which are available	10	High	hari haran Kugan Ganesh

Sprint-3	Stock Update	USN-6	As a user, I can add products which are not available in the dashboard to the stock list	6	Medium	Mutheswaran Gowri sankar Ganesh
Sprint-3			As a user, can edit the product price details in the inventory	4	High	Kugan Hari haran

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
sprint-3		USN-7	As a user , can click the submit bitton to updatethe stock inventory	1 0	Medium	Ganesh Gowri sankar
Sprint-4	Contact Customer Care	USN-8	I can be able to report any difficulties I experience as a report	1 5	HIgh	Mutheswaran Gowri sankar Ganesh Kugan Hari haran

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

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	19	6 Days	24 Oct 2022	29 Oct 2022	19	29 Oct 2022
Sprint-2	10	6 Days	31 Oct 2022	05 Nov 2022	10	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	15	6 Days	14 Nov 2022	19 Nov 2022	15	19 Nov 2022

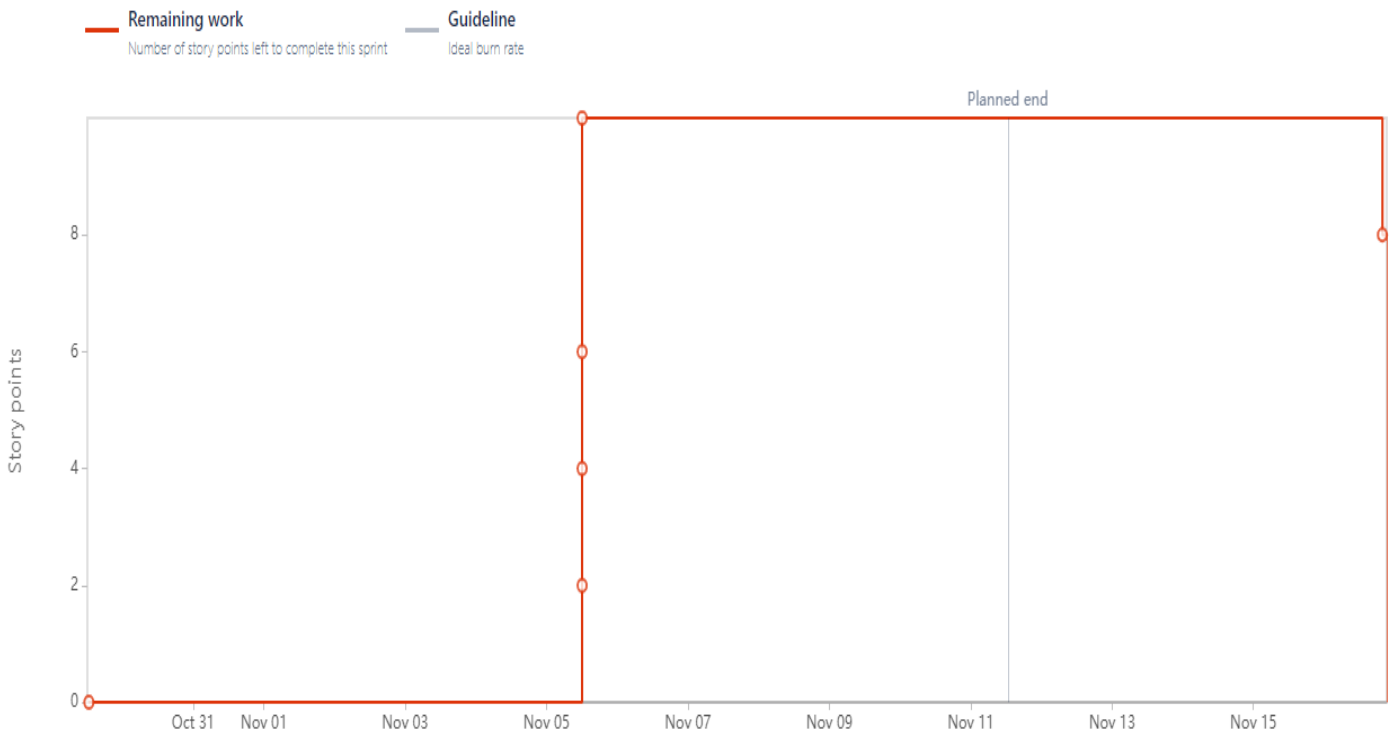
Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{\text{SPRINT DURATION}}{\text{VELOCITY}} = \frac{20}{10} = 2$$

Burndown Chart:

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

Date - October 29th, 2022 - November 11th, 2022



IMSFR Sprint 2 | Story points

Date - November 18th, 2022 - November 25th, 2022

IMSFR Sprint 3 ⌄ Story points ⌄ •••

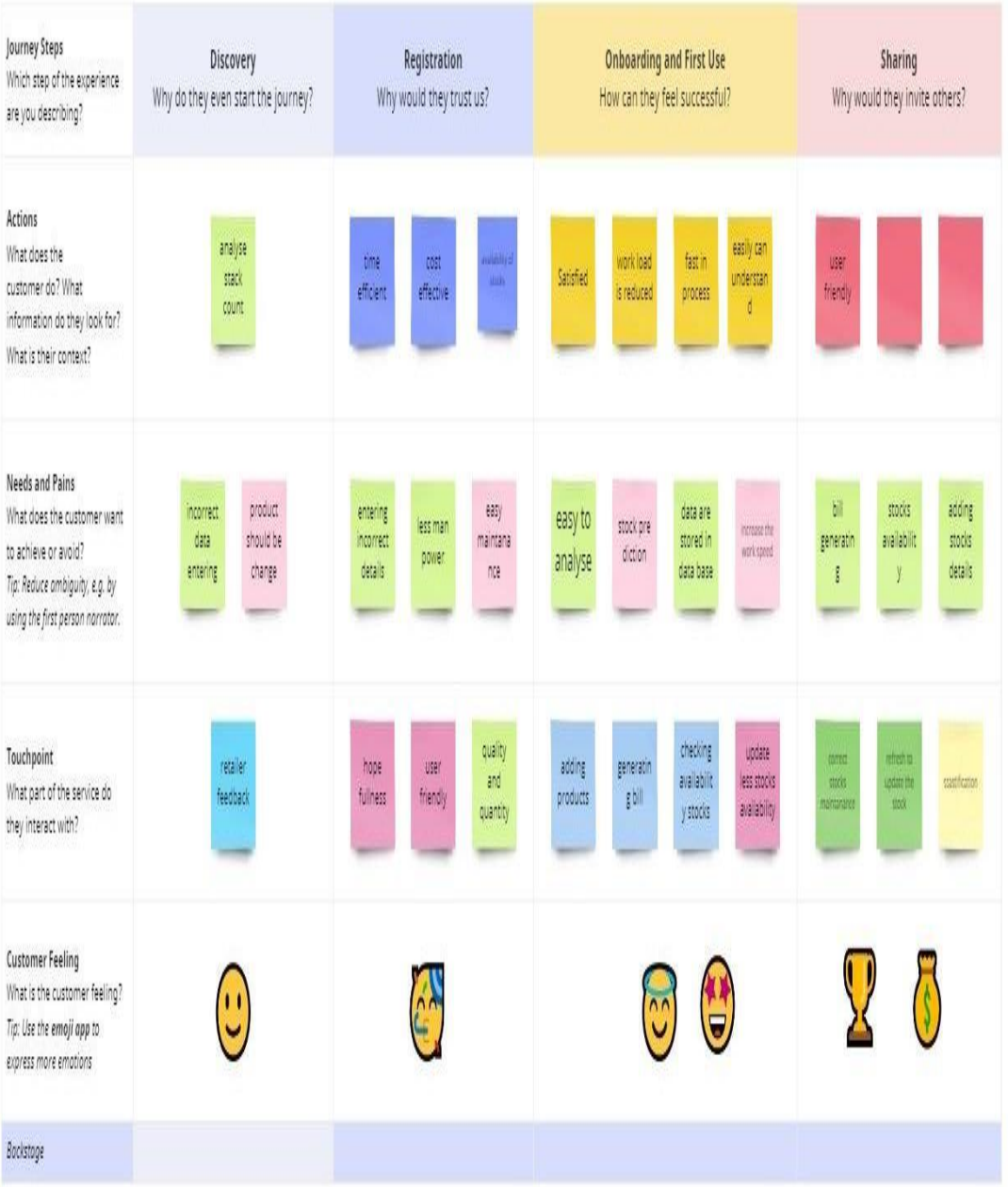
Date - November 18th, 2022 - December 9th, 2022

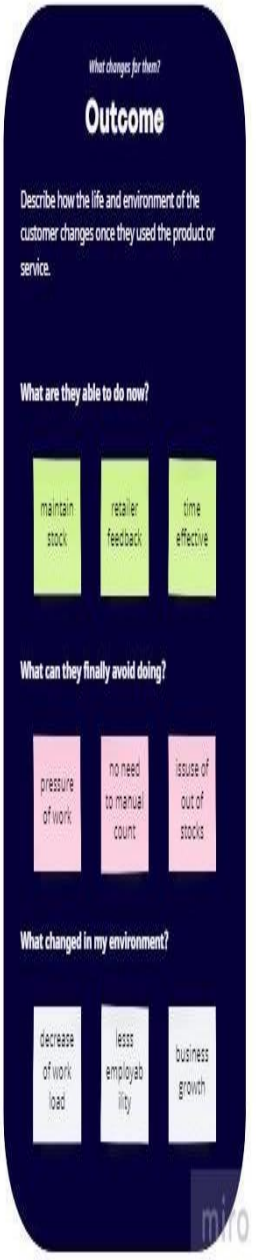
https://www.visual-paradigm.com/scrum/scrum-burndown-chart/
https://www.atlassian.com/agile/tutorials/burndown-charts

Reference:
https://www.atlassian.com/agile/project-management
https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software
https://www.atlassian.com/agile/tutorials/epics
https://www.atlassian.com/agile/tutorials/sprints
https://www.atlassian.com/agile/project-management/estimation
https://www.atlassian.com/agile/tutorials/burndown-charts

Project Design Phase-II
Customer Journey
Map

Date	03 October 2022
Team ID	PNT2022TMID25500
Project Name	Project – Inventory Management System For Retailers
Maximum Marks	4 Marks





Project Planning Phase
Milestone and Activity List

Date	22 October 2022
Team ID	PNT2022TMID52752
Project Name	Inventory Management System for Retailers

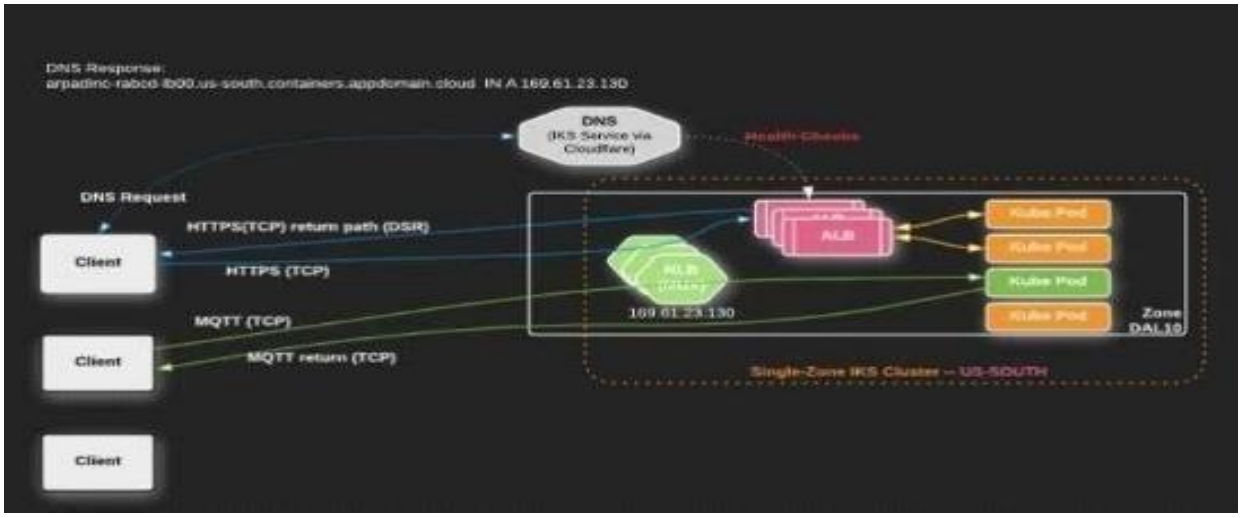
TITLE	DESCRIPTION	DATE
Literature Survey & Information Gathering	Literature survey on selected project and gathering information by referring the project's related technical papers, research publications, etc.	28 SEPTEMBER 2022
Prepare Empathy Map	Prepare empathy map canvasto capture the user's pains & gains and prepare the list of problem statements.	24 SEPTEMBER 2022
Ideation	To list by the organizing brainstorm sessions and prioritize the top three ideas based on the feasibility and importance.	25 SEPTEMBER 2022
Proposed Solution	To prepare the proposed solution documents, which includes the novelty, feasibilityof ideas, business model, social impact, scalability of thesolution, etc.	23 SEPTEMBER 2022
Problem Solution Fit	Preparing the problem solutionfit document.	30 SEPTEMBER 2022

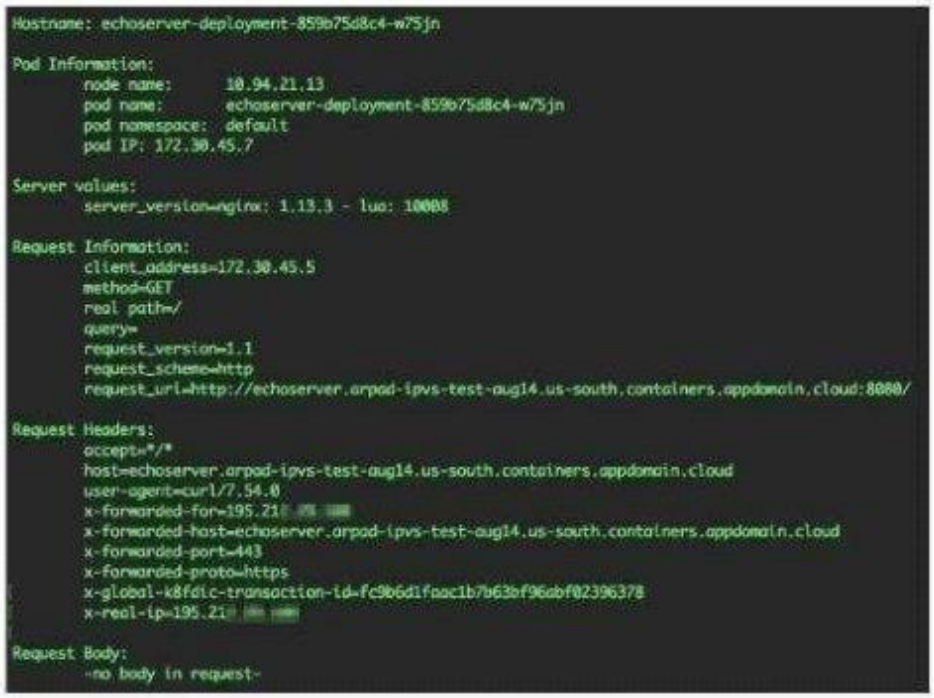
Solution Architecture	To prepare the solution architecture document	28 SEPTEMBER 2022
Customer Journey	Prepare the customers journey map help the customers understand the user interaction and experiences with the application from the beginning to the end.	20 OCTOBER 2022
Functional Requirement	Prepare the functional requirement document.	8 OCTOBER 2022
Data Flow Diagrams	Draw the data flow diagrams and submit for the review.	9 OCTOBER 2022
Technology Architecture	Prepare technical architecture diagram.	10 OCTOBER 2022
Prepare Milestone & Activity List	Prepare the milestones and activity of the project.	22 OCTOBER 2022
Project Development – Delivery of Sprint-1, 2, 3 & 4	Develop and submit the developed code by testing it and having no errors.	IN PROGRESS...

DEPLOYMENT OF APP IN IBM CLOUD DEPLOY IN KUBERNETES CLUSTER

Date	17 November 2022
Team ID	PNT2022TMID25500
Project Name	Inventory Management System For Retailers

DEPLOYMENT OF APP IN IBM CLOUD DEPLOY IN KUBERNETES CLUSTER





```
Hostname: echoserver-deployment-859b75d8c4-w75jn

Pod Information:
        node name:      10.94.21.13
        pod name:       echoserver-deployment-859b75d8c4-w75jn
        pod namespace:  default
        pod IP: 172.30.45.7

Server values:
        server_version=nginx: 1.13.3 - lua: 10008

Request Information:
        client_address=172.30.45.5
        method=GET
        real path=/
        query=
        request_version=1.1
        request_scheme=http
        request_uri=http://echoserver.arpad-ipvs-test-aug14.us-south.containers.appdomain.cloud:8080/

Request Headers:
        accept=*/*
        host=echoserver.arpad-ipvs-test-aug14.us-south.containers.appdomain.cloud
        user-agent=curl/7.54.0
        x-forwarded-for=195.21[redacted]
        x-forwarded-host=echoserver.arpad-ipvs-test-aug14.us-south.containers.appdomain.cloud
        x-forwarded-port=443
        x-forwarded-proto=https
        x-global-k8fdlc-transaction-id=fc9b6d1faac1b7b63bf96abf02396378
        x-real-ip=195.21[redacted]

Request Body:
        -no body in request-
```

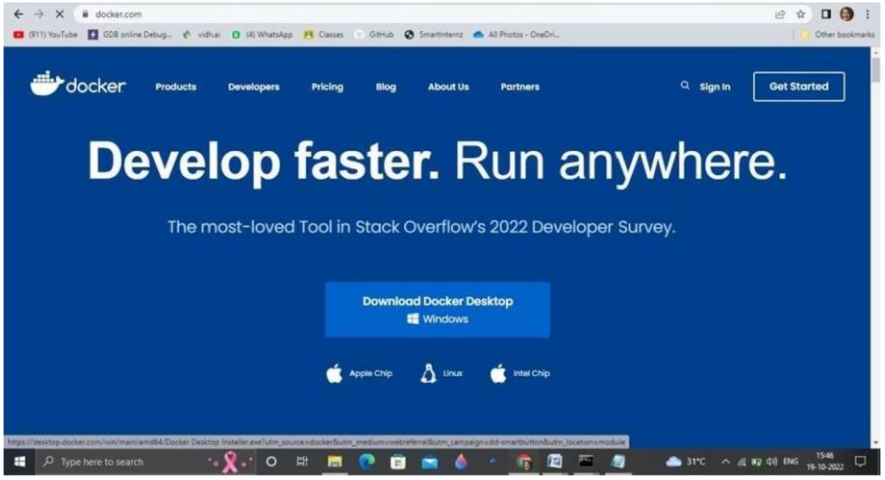
```
$ curl http://169.61.18.4:1884

Hostname: echoserver-deployment-859b75d8c4-r6s62

Pod Information:
        node name:      10.73.115.27
        pod name:       echoserver-deployment-859b75d8c4-r6s62
        pod namespace:  default
        pod IP: 172.30.154.209

Server values:
        server_version=nginx: 1.13.3 - lua: 10008

Request Information:
        client_address=195.21[redacted]
        method=GET
        real path=/
        query=
        request_version=1.1
        request_scheme=http
        request_uri=http://169.61.18.4:8080/

Request Headers:
        accept=*/*
        host=169.61.18.4:1884
        user-agent=curl/7.54.0

Request Body:
        -no body in request-
```

```
$ curl https://echoserver.arpad-ipvs-test-aug14.us-south.containers.appdomain.cloud


Hostname: echoserver-deployment-859b75d8c4-d6fdx

Pod Information:
        node name:      10.73.115.19
        pod name:       echoserver-deployment-859b75d8c4-d6fdx
        pod namespace:  default
        pod IP: 172.30.116.132

Server values:
        server_version=nginx: 1.13.3 - lua: 10008

Request Information:
        client_address=172.30.119.129
        method=GET
        real path=/
        query=
        request_version=1.1
        request_scheme=http
        request_uri=http://echoserver.arpad-ipvs-test-aug14.us-south.containers.appdomain.cloud:8080/

Request Headers:
        accept=*/*
        host=echoserver.arpad-ipvs-test-aug14.us-south.containers.appdomain.cloud
        user-agent=curl/7.54.0
        x-forwarded-for=10.184.100.58
        x-forwarded-host=echoserver.arpad-ipvs-test-aug14.us-south.containers.appdomain.cloud
        x-forwarded-port=443
        x-forwarded-proto=https
        x-global-k8fdic-transaction-id=838e8708691877ea4ac744837036 2e22
        x-real-ip=10.184.100.58

Request Body:
        -no body in request-
```

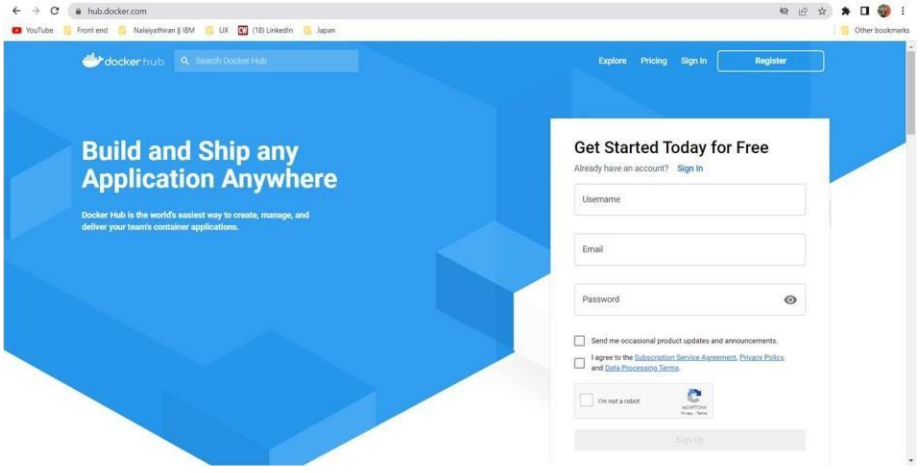
Date	17 November 2022
Team ID	PNT2022TMID25500
Project Name	Inventory Management System For Retailers

Setting up Application Environment Docker CLI Installation

Step 1. Download Docker from dockeí.com and install it by running the Docker Desktop Installe.exe file



Step 2: Move to hub.docker.com register and create an account and login with the same



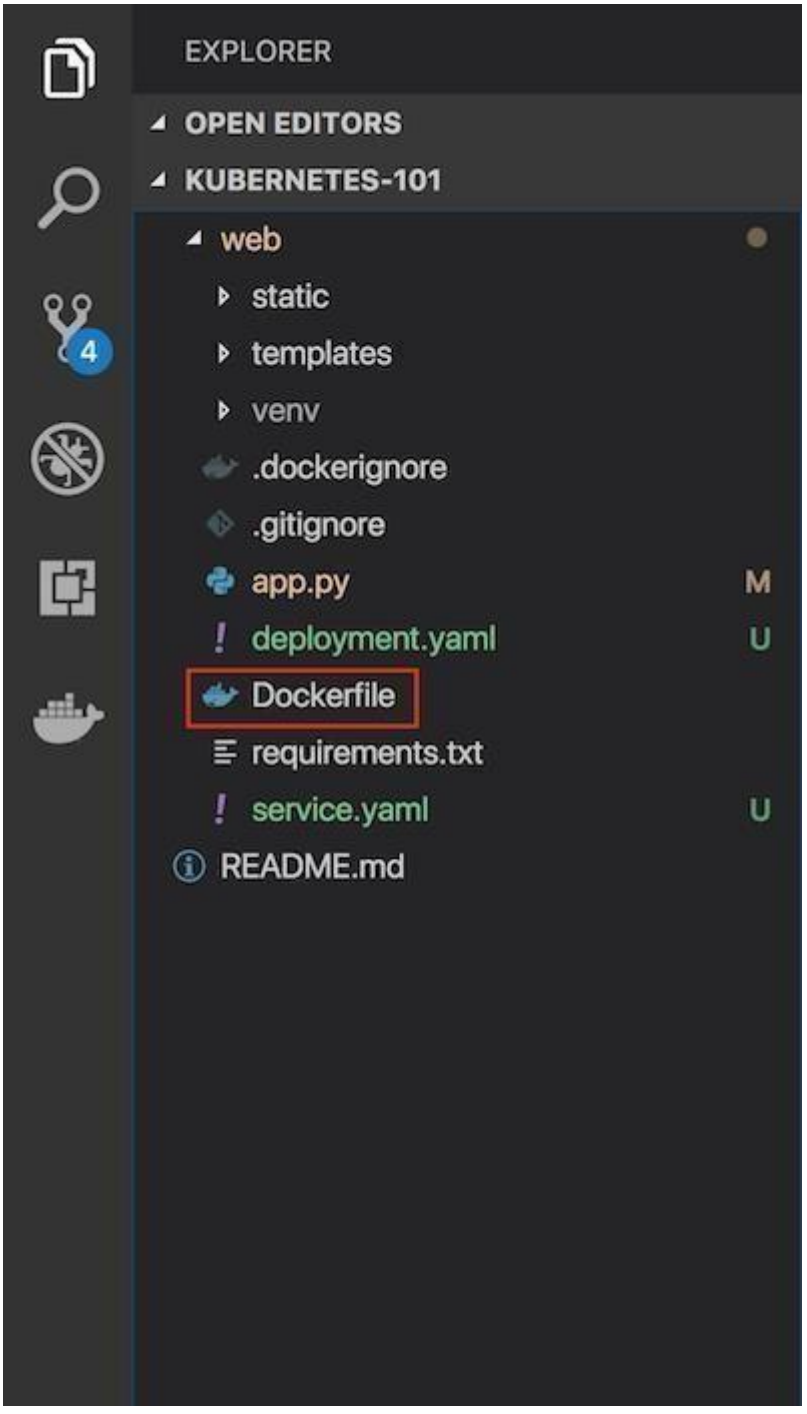
Step 3. Open Docker Desktop and start creating containers and images


Docker Desktop
Upgrade plan
Sign in
Containers
Images
Volumes
Dev Environments
BETA
Extensions
BETA
Add Extensions
Docker Desktop starting...
Type here to search
31°C
ENG
16:00
19-10-2022

Date	17 November 2022
Team ID	PNT2022TMID25500
Project Name	Inventory Management System For Retailers

Containerize your Flask application

- In your project directory, create a file named "Dockerfile." *Suggestion: Name your file exactly "Dockerfile," nothing else.*



A "Dockerfile" is used to indicate to Docker a base image, the Docker settings you need, and a list ofcommands you would like to have executed to prepare and start your new container.

- In the file, paste this code:

```
FROM python:2.7
LABEL maintainer="Kunal Malhotra, kunal.malhotra1@ibm.com"
```

- RUN apt-get update
- RUN mkdir /app WORKDIR /app COPY . /app
- RUN pip install -r requirements.txt
- EXPOSE 5000
- ENTRYPOINT ["python"]
- CMD ["app.py"]

Show more

Explanation and breakdown of the above Dockerfile code

1. The first part of the code above is:
2. FROM python:2.7

Show more

Because this Flask application uses Python 2.7, we want an environment that supports itand
already has it installed. Fortunately, DockerHub has an official image that's installed on top of Ubuntu. In one line, we will have a base Ubuntu image with Python 2.7, virtualenv, and pip. Thereare tons of images on DockerHub, but if you would like to start off with a fresh Ubuntu image andbuild on top of it, you could do that.

3. Let's look at the next part of the code:
4. LABEL maintainer="Kunal Malhotra, kunal.malhotra1@ibm.com"
5. RUN apt-get update

Show more

6. Note the maintainer and update the Ubuntu package index. The command is RUN, which is

7. RUN mkdir /app
8. WORKDIR /app
9. COPY . /app

afunction that runs the command after it.

Show more

10. Now it's time to add the Flask application to the image. For simplicity, copy the application underthe /appdirectory on our Docker Image.

WORKDIR is essentially a **cd** in bash, and COPY copies a certain directory to the provided directory in an image. ADD is another command that does the same thing as COPY, but it also allows you to add a repository from a URL. Thus, if you want to clone your git repository insteadof copying it from your local repository (for staging and production purposes), you can use
that. COPY, however, should be used most of the time unless you have a URL.

11. Now that we have our repository copied to the image, we will install all of our dependencies,which is defined in the requirements.txtpart of the code.
12. RUN pip install --no-cache-dir -r requirements.txt

Show more

13. We want to expose the port(5000) the Flask application runs on, so we use EXPOSE.
14. EXPOSE 5000

Show more

15. ENTRYPOINTspecifies the entrypoint of your application.
16. ENTRYPOINT ["python"]
17. CMD ["app.py"]

Show more

Build an image from the Dockerfile

Open the terminal and type this command to build an image from your Dockerfile: docker build -t <image_name>:<tag> .(note the period to indicate we're in our apps top level directory). For example: docker build -t app:latest .

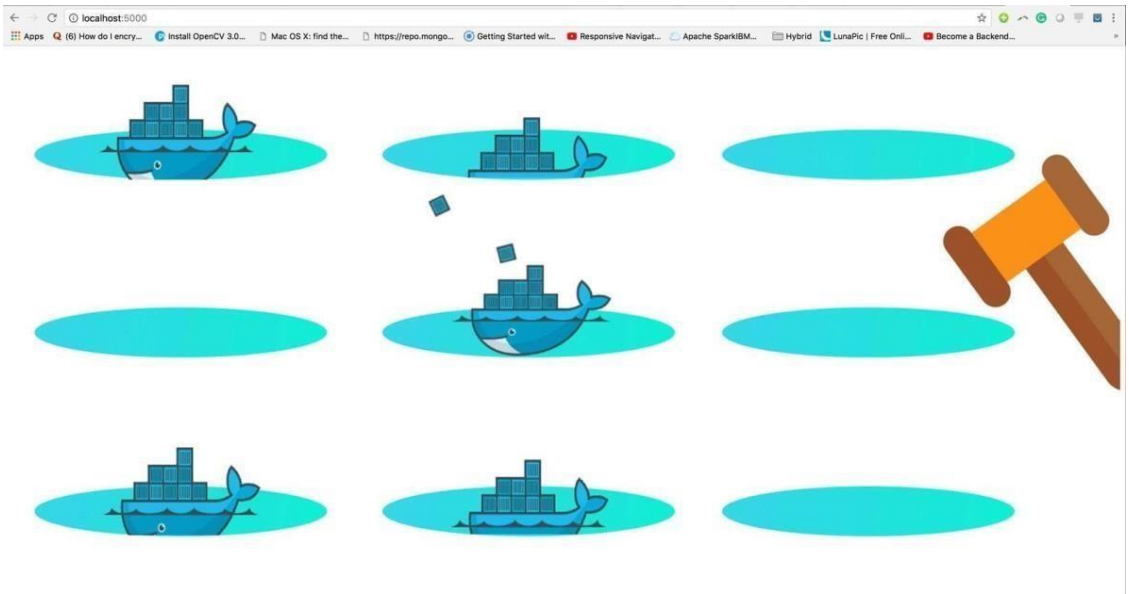
```
kunals-mbp:web kunalmalhotra$ docker build -t app:latest .
Sending build context to Docker daemon  348.2kB
Step 1/8 : FROM python:2.7
 ---> 6c76e39e7cfe
Step 2/8 : LABEL maintainer="Kunal Malhotra, kunal.malhotra1@ibm.com"
 ---> Using cache
 ---> d8b57d41591c
Step 3/8 : RUN apt-get update
 ---> Using cache
 ---> 626za134e40e
Step 4/8 : COPY . /app
 ---> f07f7377099f
Step 5/8 : WORKDIR /app
Removing intermediate container f9010b99d2fe
 ---> 0bcc6af20e3d
Step 6/8 : RUN pip install -r requirements.txt
 ---> Running in 8153040b00b7
Collecting click==6.7 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/34/c1/8806f99713ddb993c5366c362b2f908f18269f8d792aff1abfd700775a77/click-6.7-py2.py3-none-any.whl (71kB)
Collecting Flask==1.0.2 (from -r requirements.txt (line 2))
  Downloading https://files.pythonhosted.org/packages/7f/e7/08578774ed4536d3242b14dacb4696386634607af824ea997202cd0edb4b/Flask-1.0.2-py2.py3-none-any.whl (91kB)
Collecting itsdangerous==0.24 (from -r requirements.txt (line 3))
  Downloading https://files.pythonhosted.org/packages/dc/b4/a60bcdba945c00f6d608d8975131ab3f25b22f2bcfe1dab221165194b2d4/itsdangerous-0.24.tar.gz (46kB)
Collecting Jinja2==2.10 (from -r requirements.txt (line 4))
  Downloading https://files.pythonhosted.org/packages/7f/ff/ae64bacdfc95f27a016a7bed8e8686763ba4d277a78ca76f32659220a731/Jinja2-2.10-py2.py3-none-any.whl (126kB)
Collecting MarkupSafe==1.0 (from -r requirements.txt (line 5))
  Downloading https://files.pythonhosted.org/packages/4d/de/32d741db316d8fdb7680822dd37001ef7a448255de9699ab4bfcbdf4172b/MarkupSafe-1.0.tar.gz
Collecting Werkzeug==0.14.1 (from -r requirements.txt (line 6))
  Downloading https://files.pythonhosted.org/packages/20/c4/12e3e56473e52375aa29c4764e70d1b8f3efa6682bef8d0aae04fe335243/Werkzeug-0.14.1-py2.py3-none-any.whl (322kB)
Building wheels for collected packages: itsdangerous, MarkupSafe
  Running setup.py bdist_wheel for itsdangerous: started
  Running setup.py bdist_wheel for itsdangerous: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/2c/4a/61/5599631c1554768c6290b08c02c72d7317910374ca602ff1e5
  Running setup.py bdist_wheel for MarkupSafe: started
  Running setup.py bdist_wheel for MarkupSafe: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/33/56/20/ebe49a5c612fffe1c5a632146b16596f9e64676768661e4e46
Successfully built itsdangerous MarkupSafe
Installing collected packages: click, itsdangerous, MarkupSafe, Jinja2, Werkzeug, Flask
Successfully installed Flask-1.0.2 Jinja2-2.10 MarkupSafe-1.0 Werkzeug-0.14.1 click-6.7 itsdangerous-0.24
Removing intermediate container 8153040b00b7
 ---> 66d2636b97bc
Step 7/8 : ENTRYPOINT [ "python" ]
 ---> Running in bdc1c83815e1
Removing intermediate container bdc1c83815e1
 ---> 73cefc38ac1c
Step 8/8 : CMD [ "app.py" ]
 ---> Running in a784d430dd6f
Removing intermediate container a784d430dd6f
 ---> d8b6b83763a5
Successfully built d8b6b83763a5
Successfully tagged app:latest
kunals-mbp:web kunalmalhotra$
```

Run your container locally and test

After you build your image succesfully, type: docker run -d -p 5000:5000 app

This command will create a container that contains all the application code anddependencies from the image and runs it locally.

```
kunals-mbp:web kunalmalhotra$ docker run -d -p 5000:5000 app
3c2bbf86f758e9a606006eb52a2ef389ea8400eb88263137ca5543c60c616247
kunals-mbp:web kunalmalhotra$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                    NAMES
3c2bbf86f758        app                 "python app.py"     Less than a second ago   Up 5 seconds        0.0.0.0:5000->5000/tcp   compassionate_keldysh
kunals-mbp:web kunalmalhotra$
```


localhost:5000
Apps
(6) How do I encry...
Install OpenCV 3.0...
Mac OS X: find the...
https://repo.mongo...
Getting Started wit...
Responsive Navigat...
Apache SparkIBM...
Hybrid
LunaPic | Free Onli...
Become a Backend...

CHAPTER 7

CODING & SOLUTIONING

7.1 IBM Cloud

The IBM Cloud platform combines platform as a service (PaaS) with infrastructure as a service (IaaS) to provide an integrated experience. The platform scales and supports both small development teams and organizations, and large enterprise businesses. Globally deployed across data centers around the world, the solution you build on IBM Cloud spins up fast and performs reliably in a tested and supported environment you can trust!

IBM Cloud provides solutions that enable higher levels of compliance, security, and management, with proven architecture patterns and methods for rapid delivery for running mission-critical workloads.

7.2 Flask framework

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

7.3 IBM DB2 Module

Module features allow you to

• Extend schema support by allowing you to group together, in a named set, a collection of related data type definitions, database object definitions and other logic elements including: o SQL procedures o A module initialization procedure for implicit execution upon

module initialization o User-defined data type definitions including: distinct type, array type, associative array type, row type, and cursor type

• Define a namespace such that objects defined within the module can refer to other objects defined in the module without providing an explicit qualifier.

• Add object definitions that are private to the module. These objects can only be referenced by other objects within the module.

• Add object definitions that are published. Published objects can be referenced from within the module or from outside of the module.

• Define published prototypes of routines without routine-bodies in modules and later implement the routine-bodies using the routine prototype.

• Initialize the module by executing the module initialization procedure for the module. This procedure can include SQL statements, SQL PL statements, and can be used to set default values for global variables or to open cursors.

• Reference objects defined in the module from within the module and from outside of the module by using the module name as a qualifier (2-part name support) or a combination of the module name and schema name as qualifiers (3-part name support).

• Drop objects defined within the module.

• Drop the module.

• Manage who can reference objects in a module by allowing you to grant and revoke the EXECUTE privilege for the module.

7.4 Docker CLI

The Docker client enables users to interact with Docker. The Docker client can reside on the same host as the daemon or connect to a daemon on a remote host. A docker client can

communicate with more than one daemon. The Docker client provides a command line interface (CLI) that allows you to issue build, run, and stop application commands to a Docker daemon. The main purpose of the Docker Client is to provide a means to direct the pull of images from a registry and to have it run on a Docker host. Common commands issued by a client are:

- docker build
- docker pull
- docker run

7.5IBM cloud CLI

IBM Cloud CLI provides full management of your IBM Cloud account via command line. Some installation steps described along this guide may need the IBM Cloud Command Line Interface (CLI) available to be performed.

7.6 SendGrid API

SendGrid's web API allows users to pull information about their email program without having to actually log on to SendGrid.com. Users can pull lists, statistics, and even email reports. In addition to this, users can send email via the web API without using traditional SMTP. 7.7 Kubernetes Kubernetes is an open-source Container Management tool which automates container deployment, container scaling, and descaling and container load balancing (also called as container orchestration tool). It is written in Golang and has a huge community because it was first developed by Google and later donated to CNCF (Cloud Native Computing Foundation). Kubernetes can group 'n' number of containers into one logical unit for managing and deploying them easily. It works brilliantly with all cloud vendors i.e. public, hybrid and on-premises. Kubernetes is an open-source platform that manages Docker containers in the form of a cluster. Along with the automated deployment and scaling of containers, it provides healing by automatically restarting failed containers and rescheduling them when their hosts die. This capability improves the application's availability

Feature 1

```
@import 'https://fonts.googleapis.com/css?family=Poppins:300,400,500,600,700';
body {
  font-family: 'Poppins', sans-serif;
  background: #fafafa;
}

p {
  font-family: 'Poppins', sans-serif;
  font-size: 1.1em;
  font-weight: 300;
  line-height: 1.7em;
  color: #999;
}

a,
a:hover,
a:focus {
  color: inherit;
  text-decoration: none;
  transition: all 0.3s;
}

.navbar {
  padding: 15px 10px;
  background: #fff;
  border: none;
  border-radius: 0;
  margin-bottom: 40px;
  box-shadow: 1px 1px 3px rgba(0, 0, 0, 0.1);
```

```
}

.navbar-btn {
  box-shadow: none;
  outline: none !important;
  border: none;
}

.line {
  width: 100%;
  height: 1px;
  border-bottom: 1px dashed #ddd;
  margin: 40px 0;
}

/* ---------------------------------------------------
    SIDEBAR STYLE
--------------------------------------------------- */

.wrapper {
  display: flex;
  width: 100%;
  align-items: stretch;
}

#sidebar {
  min-width: 250px;
  max-width: 250px;
  background: #48494b;
  color: #fff;
  transition: all 0.3s;
}

#sidebar.active {
  margin-left: -250px;
```

```
}

#sidebar .sidebar-header {
  padding: 20px;
  background: #48494b;
}

#sidebar ul.components {
  padding: 20px 0;
  border-bottom: 1px solid #47748b;
}

#sidebar ul p {
  color: #fff;
  padding: 10px;
}
.project-title {
  font-size: 20px;
  padding-left: 10px;
  text-align: center;
}

#sidebar ul li a {
  padding: 10px;
  font-size: 1.1em;
  display: block;
}

#sidebar ul li a:hover {
  color: #7386d5;
  background: #fff;
}

#sidebar ul li.active > a,
a[aria-expanded='true'] {
```

```
  color: #fff;
  background: #48494b;
}

a[data-toggle='collapse'] {
  position: relative;
}

.dropdown-toggle::after {
  display: block;
  position: absolute;
  top: 50%;
  right: 20px;
  transform: translateY(-50%);
}

ul ul a {
  font-size: 0.9em !important;
  padding-left: 30px !important;
  background: #48494b;
}

ul.CTAs {
  padding: 20px;
}

ul.CTAs a {
  text-align: center;
  font-size: 0.9em !important;
  display: block;
  border-radius: 5px;
  margin-bottom: 5px;
}

a.download {
```

```
  background: #fff;
  color: #48494b;
}

a.article,
a.article:hover {
  background: #48494b !important;
  color: #fff !important;
}

.login-card {
  box-shadow: rgba(0, 0, 0, 0.35) 0px 5px 15px;
  border-radius: 10px;
  padding: 10px;
}
.login-card p {
  padding-left: 20px;
}
.login-card a {
  color: rgba(84, 84, 220, 0.888);
}

/* ---------------------------------------------------
    CONTENT STYLE
--------------------------------------------------- */

#content {
  width: 100%;
  padding: 20px;
  min-height: 100vh;
  transition: all 0.3s;
}

/* ---------------------------------------------------
    MEDIAQUERIES
```

```
---------------------------------------------------- */

@media (max-width: 768px) {
  #sidebar {
    margin-left: -250px;
  }
  #sidebar.active {
    margin-left: 0;
  }
  #sidebarCollapse span {
    display: none;
  }
}

/* Table Styles */

.table-wrapper {
  margin: 10px 70px 70px;
  box-shadow: rgba(99, 99, 99, 0.2) 0px 2px 8px 0px;
}

.fl-table {
  border-radius: 5px;
  font-size: 16px;
  font-weight: normal;
  border: none;
  border-collapse: collapse;
  width: 100%;
  max-width: 100%;
  white-space: nowrap;
  background-color: white;
}

.fl-table td,
.fl-table th {
```

```
  text-align: center;
  padding: 8px;
}

.fl-table td {
  border-right: 1px solid #f8f8f8;
  font-size: 16px;
}

.fl-table thead th {
  color: #ffffff;
  background: #68716e !important ;
}

.fl-table thead:nth-child(odd) {
  color: #ffffff;
  background: #324960;
}

.fl-table tr:nth-child(even) {
  background: #f8f8f8;
}
.custom-label {
  font-size: 18px;
  font-weight: 400;
}
.field input[type='text'] {
  /* width: 100%; */
  border: 2px solid #aaa;
  border-radius: 4px;
  /* margin: 8px 0; */
  outline: none;
  padding: 2px 10px;
  box-sizing: border-box;
  transition: 0.3s;
```

```
}
.field input[type='number'] {
  /* width: 100%; */
  border: 2px solid #aaa;
  border-radius: 4px;
  /* margin: 8px 0; */
  outline: none;
  padding: 2px 10px;
  box-sizing: border-box;
  transition: 0.3s;
}

.submit-button {
  padding: 5px 10px;
  color: white;
  background-color: rgb(41, 115, 41);
  border: none;
  border-radius: 8px;
  min-width: 100px;
}
.submit-button a {
  color: white;
}
.mg-20 {
  margin-top: 20px;
}
.user-deatils h4 {
  font-size: 18px;
}
/* .field input[type='text']:focus {
  border-color: rgba(59, 67, 75, 0.687);
  box-shadow: 0 0 8px 0 rgba(80, 94, 108, 0.667);
} */
.field {
  display: flex;
```

```
  align-items: center;
  padding: 10px 0px;
}
.text-inputs {
  margin: 0px 10px;
}
/* Responsive */

@media (max-width: 767px) {
  .fl-table {
    display: block;
    width: 100%;
  }
  .table-wrapper:before {
    content: 'Scroll horizontally >';
    display: block;
    text-align: right;
    font-size: 11px;
    color: white;
    padding: 0 0 10px;
  }
  .fl-table thead,
  .fl-table tbody,
  .fl-table thead th {
    display: block;
  }
  .fl-table thead th:last-child {
    border-bottom: none;
  }
  .fl-table thead {
    float: left;
  }
  .fl-table tbody {
    width: auto;
    position: relative;
```

```
  overflow-x: auto;
}
.fl-table td,
.fl-table th {
  padding: 20px 0.625em 0.625em 0.625em;
  height: 60px;
  vertical-align: middle;
  box-sizing: border-box;
  overflow-x: hidden;
  overflow-y: auto;
  width: 120px;
  font-size: 13px;
  text-overflow: ellipsis;
}
.fl-table thead th {
  text-align: left;
  border-bottom: 1px solid #f7f7f9;
}
.fl-table tbody tr {
  display: table-cell;
}
.fl-table tbody tr:nth-child(odd) {
  background: none;
}
.fl-table tr:nth-child(even) {
  background: transparent;
}
.fl-table tr td:nth-child(odd) {
  background: #f8f8f8;
  border-right: 1px solid #e6e4e4;
}
.fl-table tr td:nth-child(even) {
  border-right: 1px solid #e6e4e4;
}
.fl-table tbody td {
```

```
    display: block;
    text-align: center;
  }
}

.forms-wrapper {
  display: flex;
  /* align-items: center; */
  justify-content: space-around;
}

.red-button {
  background-color: rgb(186, 13, 13);
}
```

Js file

```
const selectedItem = document
  .getElementById('sidebarCollapse')
  .addEventListener('click', (e) => {
    const ele = document.getElementById('sidebar');
    ele.classList.toggle('active');
```

Feature 2

```
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=815fa4db-dc03-4c70-869a-
a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30367;SECUR
ITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=gkx49901;PWD=kvW
CsySl7vApfsy2", ", ")

app = Flask(__name__)
app.secret_key = 'jackiechan'



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('/user/<id>')
@login_required
def user_info(id):
    with sql.connect('inventorymanagement.db') as con:
        con.row_factory = sql.Row
        cur = con.cursor()
        cur.execute(f'SELECT * FROM users WHERE email="{id}"')
        user = cur.fetchall()
    return render_template("user_info.html", user=user[0])


@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 users WHERE email =? 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 users WHERE email =?'
        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,FIRSTNAME,LASTNAME,EMAIL,PASSWORD) VALUES (?,?,?,?,?)'
        pstmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(pstmt, 1, username)
        ibm_db.bind_param(pstmt, 2, "firstname")
        ibm_db.bind_param(pstmt, 3, "lastname")
        # ibm_db.bind_param(pstmt,4,"123456789")
        ibm_db.bind_param(pstmt, 4, email)
        ibm_db.bind_param(pstmt, 5, 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)
        # print(f"The ID is : ", dictionary["NAME"])
        # print(f"The name is : ", dictionary["QUANTITY"])
        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 users 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  users 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:
```

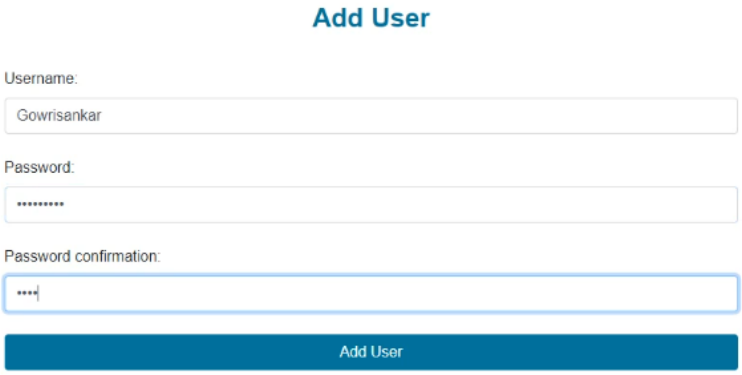
```
        return redirect(url_for('suppliers'))


@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)
```

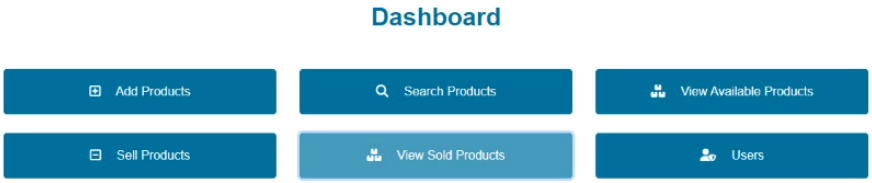
CHAPTER 8

TESTING

8.1TEST CASE



Adding The new user ID



Dashboard UI

All Products

Product ID	Product Name	Product Price	Quantity
286	Mobile	600	65
287	Shoes	565	657
288	Bike	4000	67
289	TV	600	50
290	Camera	400	90
291	Realme Air Buds 2	3300	5

Go Back

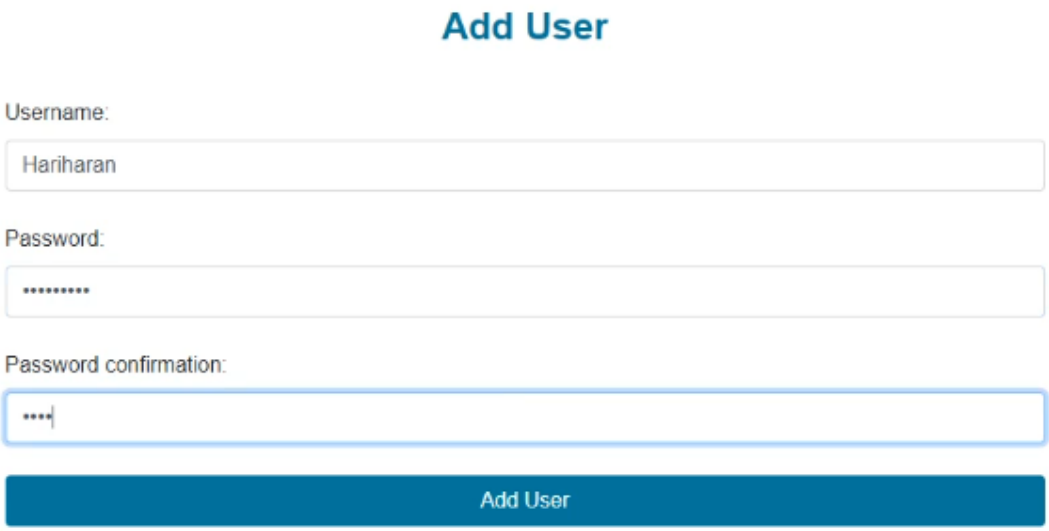
Product Details

Sell Products

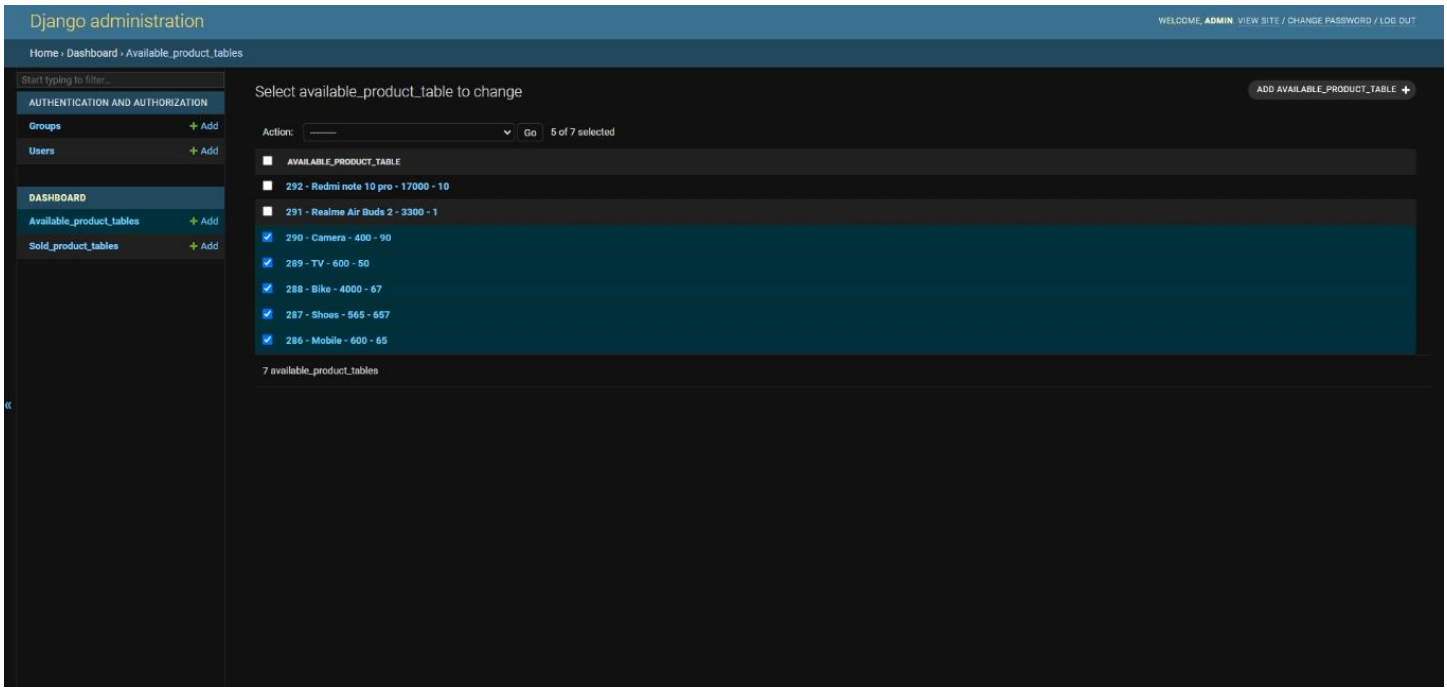
Product sold successfully!

Go Back

Sold product Details



Adding Another user ID


Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Dashboard › Available_product_tables
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
DASHBOARD
Available_product_tables + Add
Sold_product_tables + Add
Select available_product_table to change
ADD AVAILABLE_PRODUCT_TABLE +
Action: ——— Go 5 of 7 selected
AVAILABLE_PRODUCT_TABLE
292 - Redmi note 10 pro - 17000 - 10
291 - Realme Air Buds 2 - 3300 - 1
290 - Camera - 400 - 90
289 - TV - 600 - 50
288 - Bike - 4000 - 67
287 - Shoes - 565 - 657
286 - Mobile - 600 - 65
7 available_product_tables

8.2USER ACCEPTANCE TESTING

Defect Analysis

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

	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	0	0	0	0
Duplicate	0	0	0	0	0
External	0	0	0	0	0
Fixed	0	0	0	0	0
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	0	0	0	0	0

Test Case Analysis

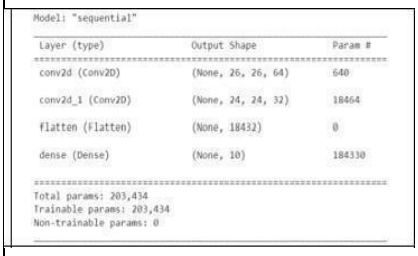
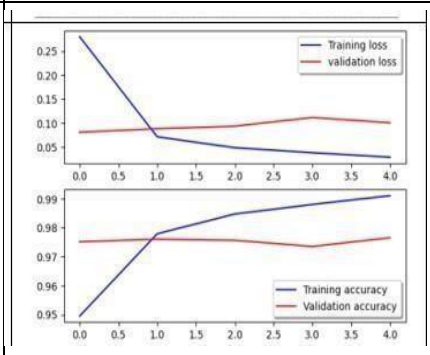
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Client Application	5	0	0	5
Security	5	0	0	5
Final Report Output	5	0	0	5
Version Control	5	0	0	5

CHAPTER 9

RESULTS

9.1Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Model Summary	-	
2.	Accuracy	Training Accuracy – 99% Validation Accuracy – 97%	
3.	Confidence Score (OnlyYolo Projects)	Class Detected - Confidence Score -	

CHAPTER 10

ADVANTAGES AND DISADVANTAGES

10.1 ADVANTAGES

The inventory management system for retailers is a web-based application. A dashboard is given to retailers where they can able to update, manage and create a product and assign the number of quantities for the products. Whenever a product goes out of stock the inventory management system for retailers alerts the retailers through email stating product goes out of stocks. Thus, inventory management system helps retailers to increase the profit and reduce the risk of holding large quantity of a particular stock. The inventory management system helps the retailers to efficiently utilizes the inventory area i.e., where the store all the products

10.2 DISADVANTAGES

The inventory management system for retailers helps them in many ways, but it needs a manual way of updating the quantity of the product. The retailers need to create and update the quantity of stock in the web-based application. The inventory management system necessitates manual updating of stock quantities, which adds to their workload.

CHAPTER 11

CONCLUSION

A web-based application is created to manage inventory stocks. Retailers can able to update, create, and manage products in this web application. The inventory management system gives a mail alert, if a product goes out of stock, stating that a product has gone out of stock. This allows retailers to increase their profit, reduce the risk of having too much stock, and make better use of their inventory space.

CHAPTER 12

FUTURE SCOPE

The future scope of the web-based inventory management application for retailers includes building a charting system into the application that helps them know the sales performance of a product for different time periods like a day, a week, or a year. Automation of updating the quantity of stock for each product using technologies like barcodes, QR codes, etc. Analyze each product's sales performance in relation to key performance indicators to determine when to offer discounts and offers on products with good and bad stock performance.

GitHub link : https://github.com/IBM-EPBL/IBM-Project-26619-1660031270.git

Video link : https://drive.google.com/file/d/1SOUtq82QKhLXRelqrPPDb9M212KX7RKn/view?usp=drivesdk

REFERENCES

1. Y. Fan, 2010, "Development of inventory management system," 2nd IEEE International Conference on Information Management and Engineering, 2010, pp. 207-210, Doi: 10.1109/ICIME.2010.5478077.

2. A. Milella, A. Petitti, R. Marani, G. Cicirelli and T. D'orazio, "Towards Intelligent Retail: Automated on-Shelf Availability Estimation Using a Depth Camera," in IEEE Access, vol. 8, pp. 19353-19363, 2020, doi: 10.1109/ACCESS.2020.2968175.

3. Inventory management for retail companies, "A literature review and current trends", March 2021, DOI:10.1109/ICI2ST51859.2021.00018, Conference: 2021 Second International Conference on Information Systems and Software Technologies (ICI2ST)