Inventory Management System for Retailers

Team ID	PNT2022TMID45035
Project Name	Inventory Management Systemfor Retailers
Batch number	B1-1M3E

TEAM MEMBERS: S. SHWETHA,P. BANUPRIYA,S.DHANALAKSHMI, M. NANTHINI, S. VAISHNAVI

1.INTRODUCTION

1.1 Project Overview

The retail industry is one of the industries that is growing in fast pace where the number of retail business keep on increasing from time to time in order to meet the demand from consumers of specified areas. There are different types of retail shops available for consumer to choose ranging from hypermarket to mini market according to their convenience. Most of the shops can be found in residential areas, streets, or in a shopping mall. Basically, retail store sells wide range of goods and services from wholesaler or supplier to the end-user. Thus, the nature of retail

business required a good management of inventory level in order to meet the demand of the customers.

This is because more items will be made available in a larger quantity, thus tracking the sales made with inventory level in the shop would be complicated and time consuming for the retailer. Besides, the situation gets worst when the retailer does not have proper method to determine items purchased by their customers.

Thus, this project will provide solution for retailers that are still using traditional way in keeping their inventory data. Sales and Inventory Management System is a computer- based system that provides the shop structure for maintaining and controlling goods to be stocked. The approach of Inventory Management System is commonly used to avoid product overstock or outrages by integrating daily ‘Point of Sales’ with store’s inventory level.

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 the inventory, retailers always know what items are in stock, how many of them there are, and where they are located.

Inventory Management system provides information to efficiently manage the flow of materials, effectively utilize people and equipment, coordinate internal activities and communicate with customers. Inventory Management does not make decisions or

manage operations, they provide the information to retailers who make more accurate and timely decisions to manage their operations.

The goal of any good inventory management system is to help retailers keep track of the inventory levels of their products. This means allowing them full transparency into their chain to monitor the flow of goods from their supplier.

The benefits are both operational and financial. Not only will it serve to improve performance, but it’s also useful for preventing theft with the help of product tracking and security.

Retailers can also aim to use their inventory management plan to monitor sales procedures which leads to better service. Inventory management is especially useful for businesses that want to effectively manage seasonal items or new bestsellers throughout the year without disrupting the rest of their chain.

2. LITERATURE SURVEY

2.1 Existing problem

Soonkyolee et al. (2019) proposed a model the possible relationship between retailer and salvage retailer. Zero ending inventory is also boost the sale and profit based on the demand formulation. Using numerical experiments, a comparative analysis of the two alternatives is conducted to determine suitable options for improving supply chain performance. In general, the performance of vendor-managed inventory is better than that of retailer-managed inventory, but observed from the numerical experiments that there exist circumstances under which retailer-managed inventory shows better supply chain performance. Thus, the study can be extended to a decentralized supply chain where information is not fully available to the retailer. In addition, consider a supply chain consisting of a single manufacturer and a single retailer.

Cinthya Vanessa Munoz Macas et al. (2021) conducted a research for five years, between 2015 and 2019, focusing specifically on the retail sector. Nowadays, organizations in the retail sector face multiple challenges in the planning and management of their resources. It is important to mention that all retailers may not be able to employ these technologies due to their high cost of implementation and

maintenance. To all those retailers with limited resources, cheaper software is accessible that could help with the management of the inventory like bar codes or policies as EOQ, AUD, and IQD, which will allow optimizing the stock without making considerable investments.

Lakshmi et al.(2021) describes an Inventory Management System that stores sales data for a certain desktop application. A simple desktop application that links to the actual distribution center, allowing information to be refreshed and confirmed in the store. A secure application that prevents data from being spoiled in the stores.System provides sales information on a daily, weekly, and monthly basis. The system makes inventory management a breeze. Increased income and profitability, a better employee climate, and an overall boost in customer satisfaction will be noticed as a result of the inventory management system.

Nazar Sohail et al.(2018) proposed Inventory management has to do with keeping precise records of finished goods that are ready for shipment. This often means posting the production of newly completed goods to the inventory totals as well as subtracting the most recent shipments of finished goods to buyers. When the company has a return policy in place, there is usually a subcategory contained in the finished goods inventory to account for any returned goods that are reclassified or second grade quality. Accurately maintaining figures on the finished goods inventory makes it possible to quickly convey information to sales personnel as to what is available and ready for shipment at any given time. The ROI of Inventory management will be seen in the forms of increased revenue and profits, positive employee atmosphere, and on overall increase of customer satisfaction.

Punam Khobragade et al (2018) proposed an alarm about the information

section of the bill which in view of desktop application. The system is straight forward desktop application in which the network to the immediate distribution center with the goal that information ought to be refreshed in store for the confirmation. A secure application in which the no information spillage from the stockroom. And further more gives the one table organization retailers know about what was sold.

2.2 References

1. “Punam Khobragade, Roshni Selokar, Rina Maraskolhe, Prof.Manjusha Talmale”,” Research paper on Inventory management system”,2018.

2. “Nazar Sohail, Tariq Hussain Sheikh”,” A Study of Inventory Management System Case Study”,2018

3. .“Soonkyolee, youngjoo Kim, taesucheong, seung ho yoo”,” Effects of yield and leadtime uncertainity on retailer-managed and vendor-managed inventory management”, 2019

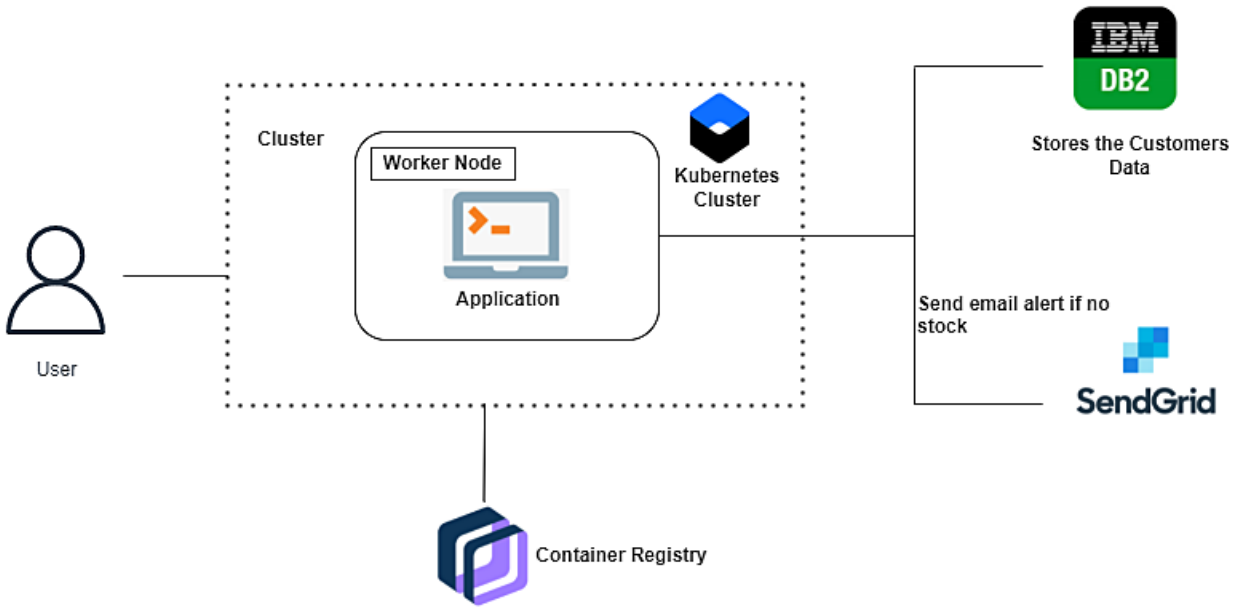
4. “Vara Lakshmi G S , Shivaleela S”,” A Review of Inventory Management System”, 2021.

5. “ Cinthya Vanessa Munoz Macas, Jorge Andrés Espinoza Aguirre ,Rodrigo Carrion, Mario Pena”,” Inventory management for retail companies -A literature review and current trends”,2021.

2.3 Problem Statement Definition

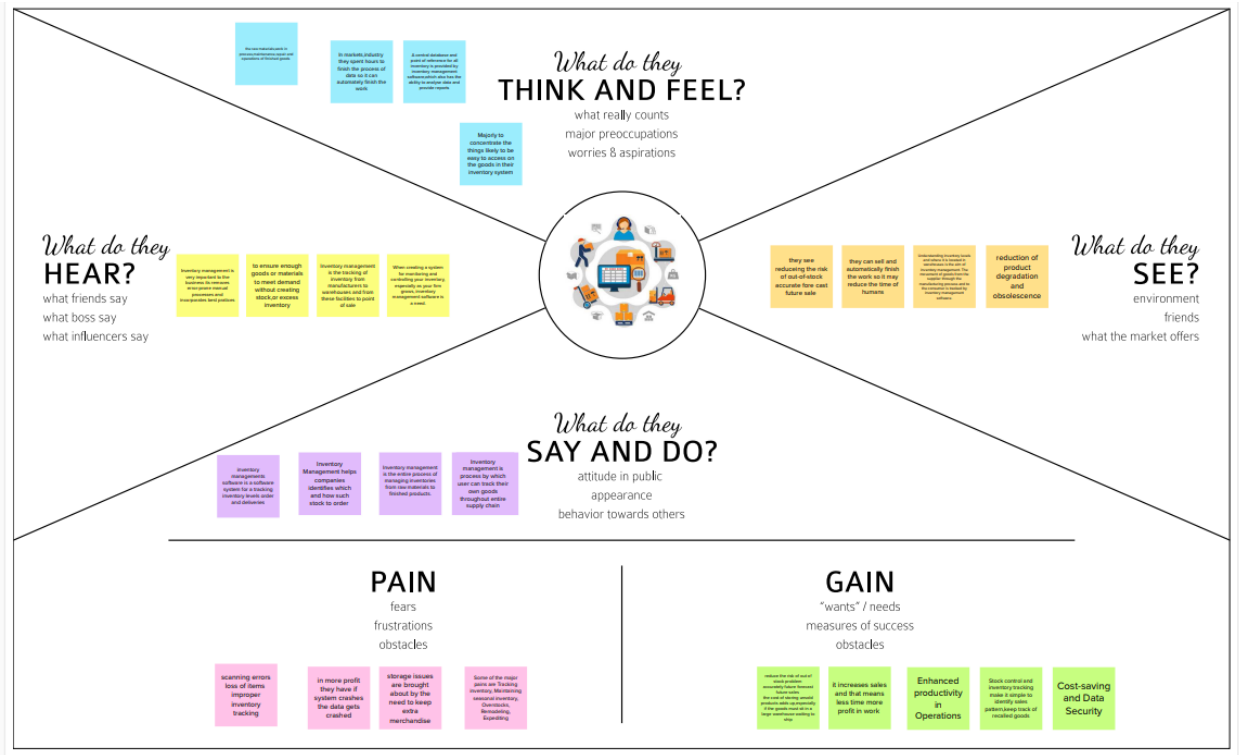
Retail inventory management is the process of ensuring you carry merchandise that

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.



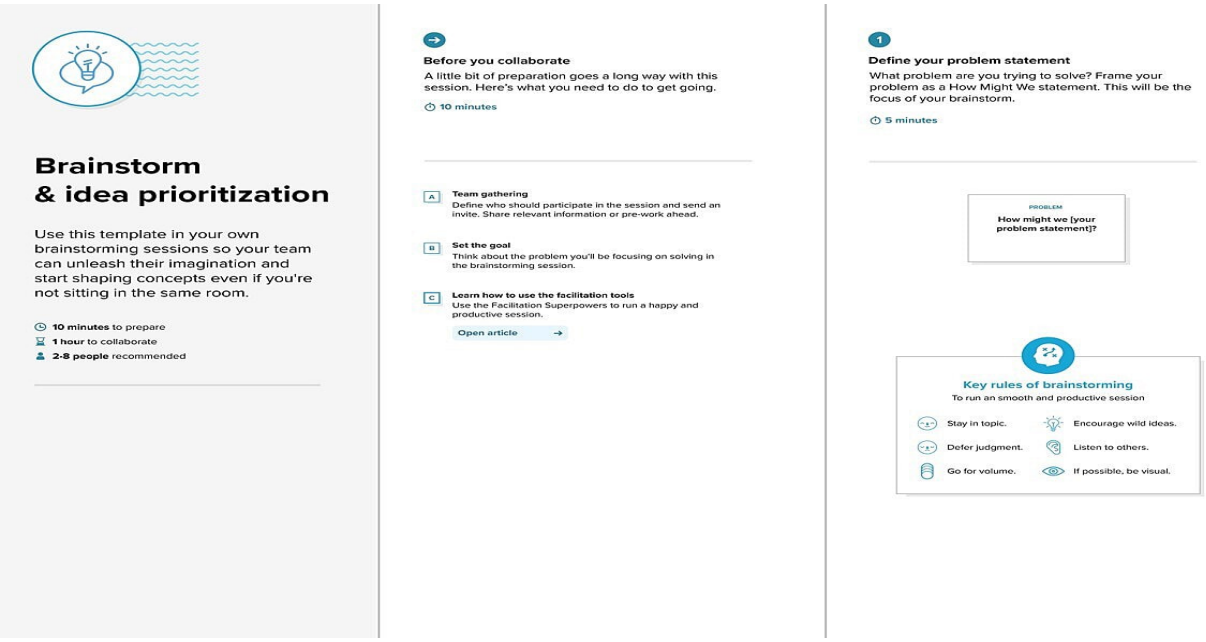
3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

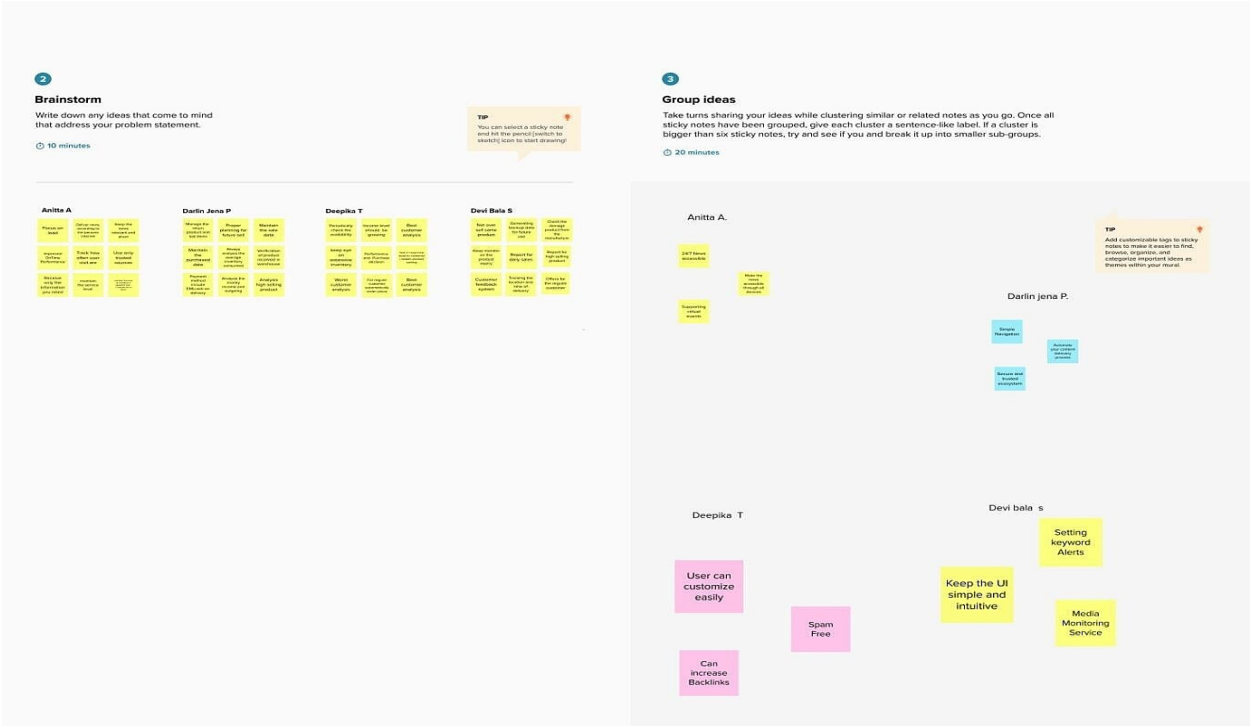


3.2 Ideation & Brainstorming

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



Step-2: Brainstorm, Idea Listing and Grouping



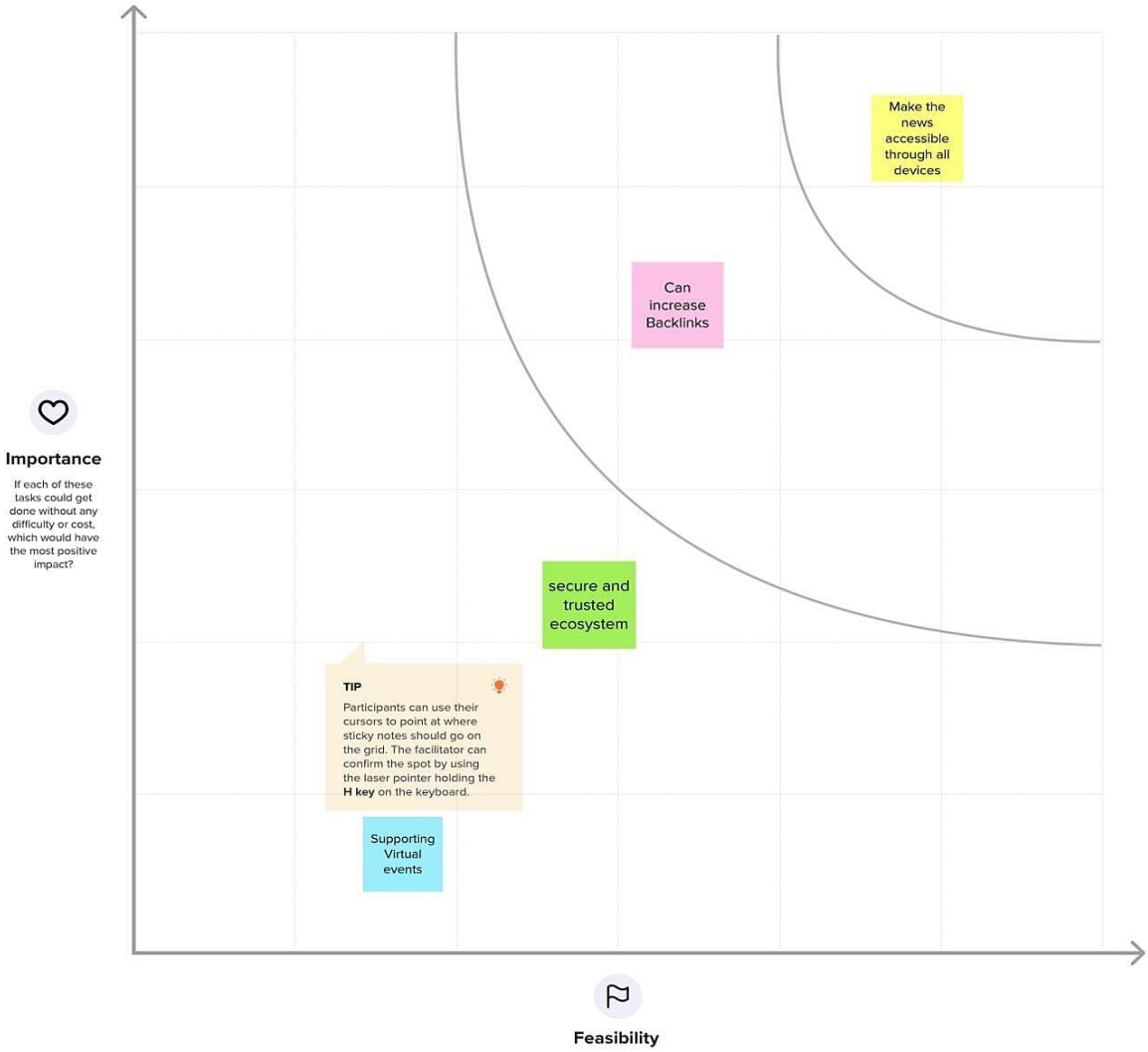
Step-3: Idea Prioritization

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
01	Problem Statement (Problem to be solved)	To solve customer issues using Cloud Application Development.
02	Idea / Solution description	All products that you have should be tagged with RFID, Barcodes, or QR codes etched withalaser.
03	Novelty / Uniqueness	Well managed stocks leads to more efficient and organized warehouses.
04	Social Impact / Customer Satisfaction	User friendly interface. Accurate and easy product search. Deals with customer queries.
05	Business Model (Revenue Model)	Reliable database Secure database Easy-to-use interface

4. REQUIREMENT ANALYSIS

The process of determining user expectations for a system under consideration. These should be quantifiable and detailed.

Requirement Analysis:

- Serves as a foundation for test plans and project plan
- Serves as an agreement between developer and customer
- Process to make stated and unstated requirements clear
- Process to validate requirement for completeness, ambiguity and feasibility.

4.1 Functional requirement

Functional requirements specify what a system should be able to do through computations, technical details, data manipulation and processing, and other specialised functions. Use cases, which are used to represent behavioural requirements, explain all the instances in which the system makes use of the functional requirements.

Non-functional requirements, commonly referred to as "quality requirements," which place restrictions on the design or execution, support functional requirements (such as performance requirements, security, or reliability). Non-functional requirements often take the form "system shall be," while functional needs are typically articulated in the form "system must do." While non-functional needs are defined in the system architecture, the plan for accomplishing functional requirements is detailed in the system design. Functional requirements, as used in requirements engineering, outline specified outcomes of a system.

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it's important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behaviour under specific conditions. For example: The system sends an approval request after the user enters personal information. A search feature allows a user to hunt among various invoices if they want to credit an issued invoice. The system sends a confirmation email when a new user account is created.

SR No	**Functional Requirement(Epic)**	**Sub Requirement(Story/ Sub-Task)**
1	User Registration	Registration through Form Registration through Gmail Registration through Google
2	User Confirmation	Confirmation via Email Confirmation via OTP
3	User Login	Login via Google Login with Email id and Password
4	Admin Login	Login via Google Login with Email id and Password
5	Query Form	Description of the issues Contact information
6	E-mail	Login alertness
7	Feedback	Customer feedback

4.2 Non-Functional requirements

In general, non-functional requirements outline what a system is supposed to be rather than what it should be able to perform. Functional requirements are typically expressed as "system shall do," an individual action or component of the system, maybe explicitly in terms of a mathematical function, or as a black box description of an input, output, process, and control functional model, also known as an IPO Model. Non-functional requirements, on the other hand, have the form of "system shall be," which refers to a general characteristic of the system as a whole or of a particular aspect rather than a specific function. The overall characteristics of the system frequently determine whether a development project is a success or a failure. Non-functional requirements are frequently referred to as a product's "quality traits" in error.

Non-functional requirements, not related to the system functionality, rather define how the system should perform. Some examples are: The website pages should load in 3 seconds with the total number of simultaneous users.

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs. Also known as system qualities, nonfunctional requirements are just as critical as functional Epics, Capabilities, Features, and Stories. They ensure the usability and effectiveness of the entire system.

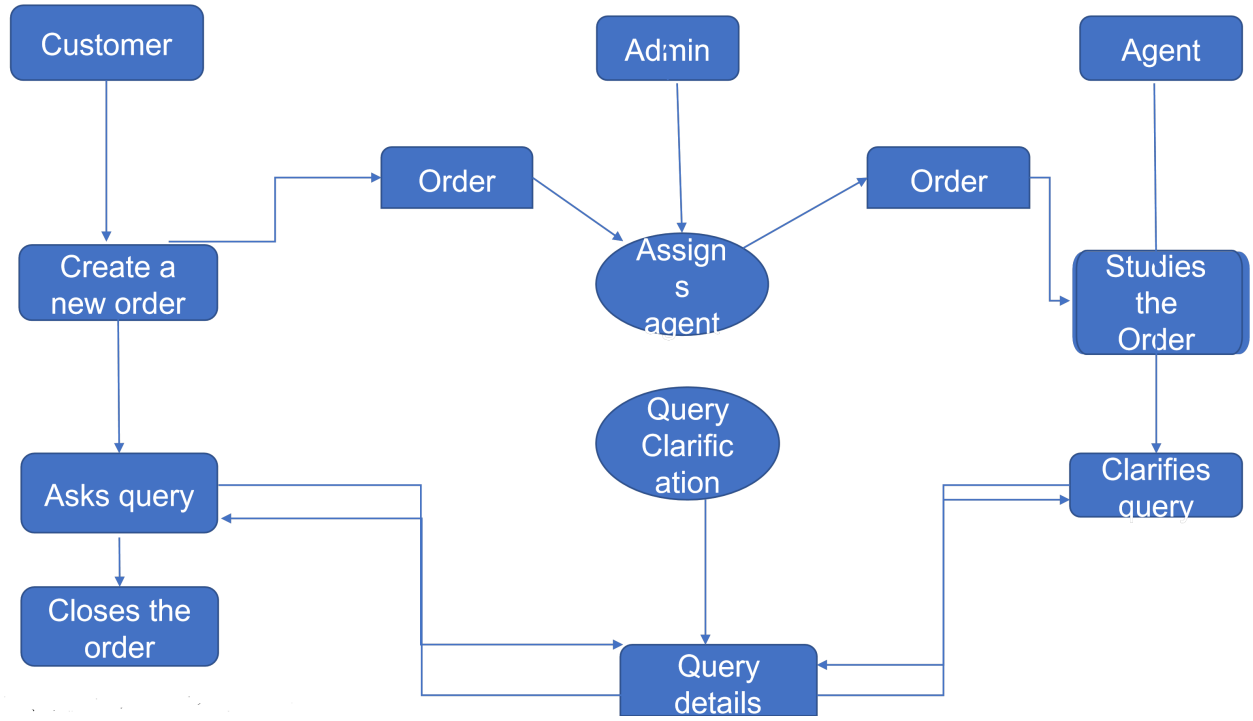
Non-functional requirements as requirements that “do not relate directly to the behaviour of functionality of the solution, but rather describe conditions under

which a solution must remain effective or qualities that a solution must have”

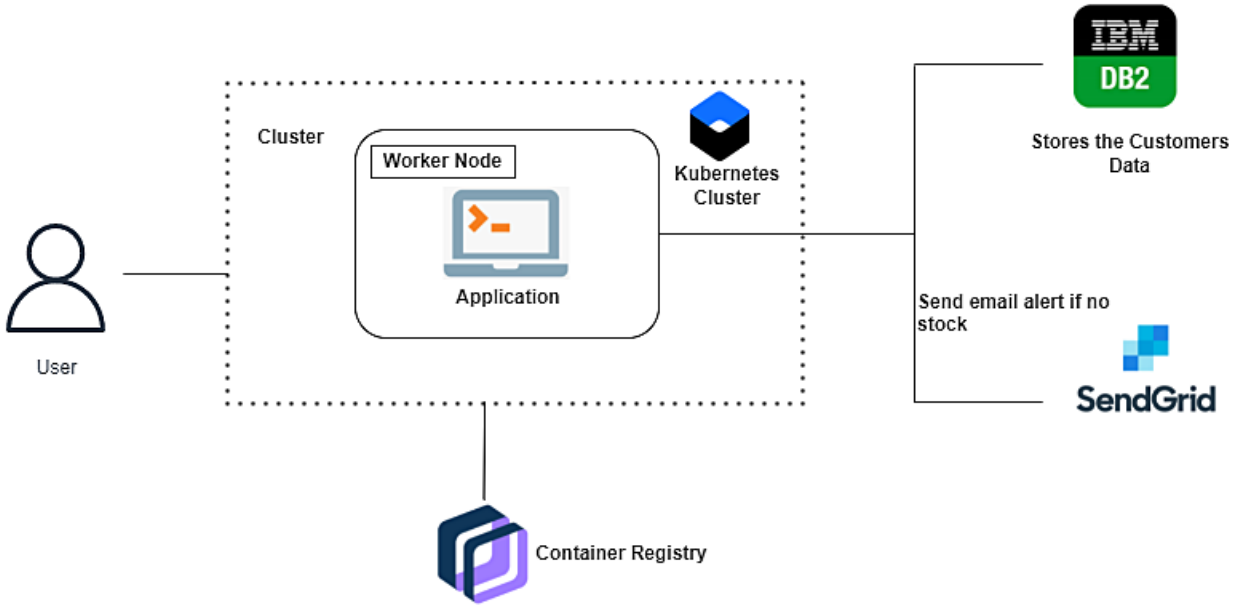
SR No	Non-Functional Requirement	Description
1	Usability	To provide the solution to the problem
2	Security	Track of login authentication
3	Reliability	Tracking of decade status through email
4	Performance	Effective development of web application
5	Availability	24/7 service
6	Scalability	Agents scalability as per the number of customers

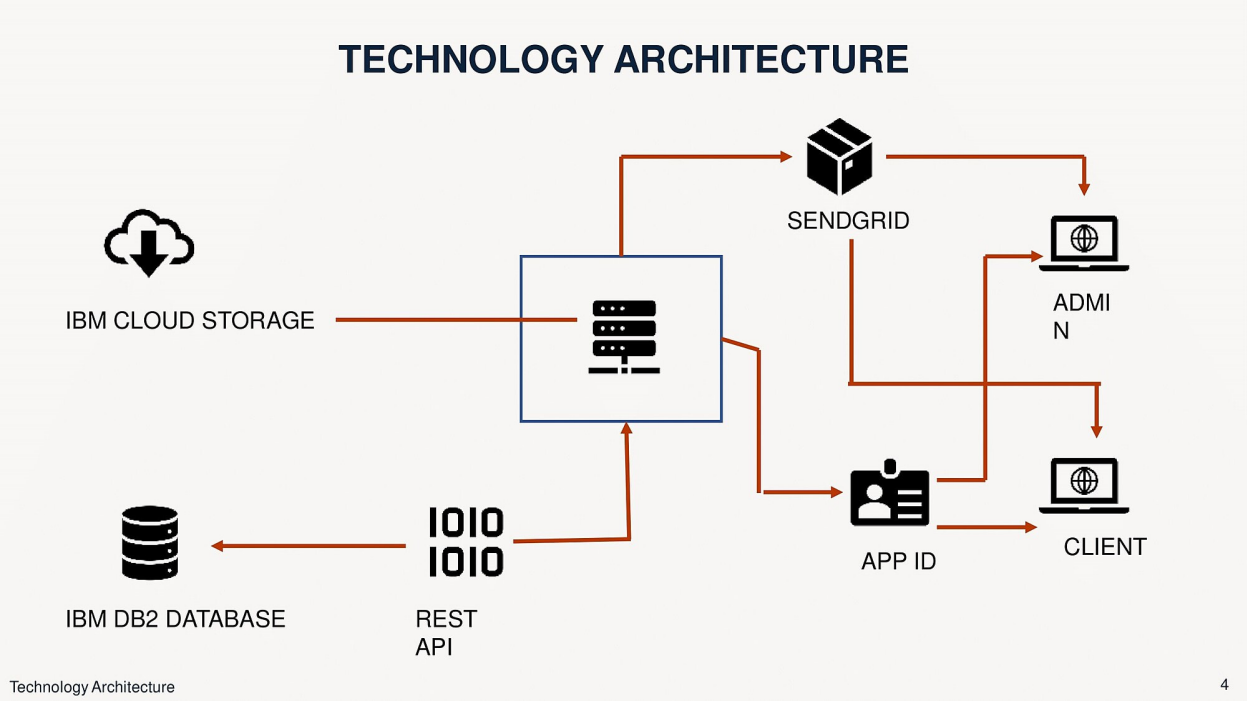
5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture





5.3 User Stories

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
Customer (Mobile user)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	login	USN-2	As a customer, I can login to the application by entering correct email and password.	I can access my account/dashboard.	High	Sprint-1
	Dashboard	USN-3	As a customer, I can see all the orders raised by me.	I get all the info needed in my dashboard.	Low	Sprint-2
	Order creation	USN-4	As a customer, I can place my order with the detailed description of my query	I can ask my query	Medium	Sprint-2
	Address Column	USN-5	As a customer, I can have conversations with the assigned agent and get my queries clarified	My queries are clarified.	High	Sprint-3
	Forgot password	USN-6	As a customer, I can reset my password by this option incase I forgot my old password.	I get access to my account again	Medium	Sprint-4
	Order details	USN-7	As a Customer ,I can see the current stats of order.	I get abetter understanding	Medium	Sprint-4
Agent (web user)	Login	USN-1	As an agent I can login to the application by entering Correct email and password.	I can access my account / dashboard.	High	Sprint-3
	Dashboard	USN-2	As an agent, I can see the order details assigned to me by admin.	I can see the tickets to which I could answer.	High	Sprint-3
	Address column	USN-3	As an agent, I get to have conversations with the customer and clear his/er dobuts	I can clarify the issues.	High	Sprint-3
	Forgot password	USN-4	As an agent I can reset my password by this option in case I forgot my old password.	I get access to my account again.	Medium	Sprint-4

Admin (Mobile user)	Login	USN-1	As a admin, I can login to the appliaction by entering Correct email and password	I can access my account/dashboard	High	Sprint-1
	Dashboard	USN-2	As an admin I can see all the orders raised in the entire system and lot more	I can assign agents by seeing those order.	High	Sprint-1
	Agent creation	USN-3	As an admin I can create an agent for clarifying the customers queries	I can create agents.	High	Sprint-2
	Assignment agent	USN-4	As an admin I can assign an agent for each order created by the customer.	Enable agent to clarify the queries.	High	Sprint-1
	Forgot password	USN-5	As an admin I can reset my password by this option in case I forgot my old password.	I get access to my account.	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

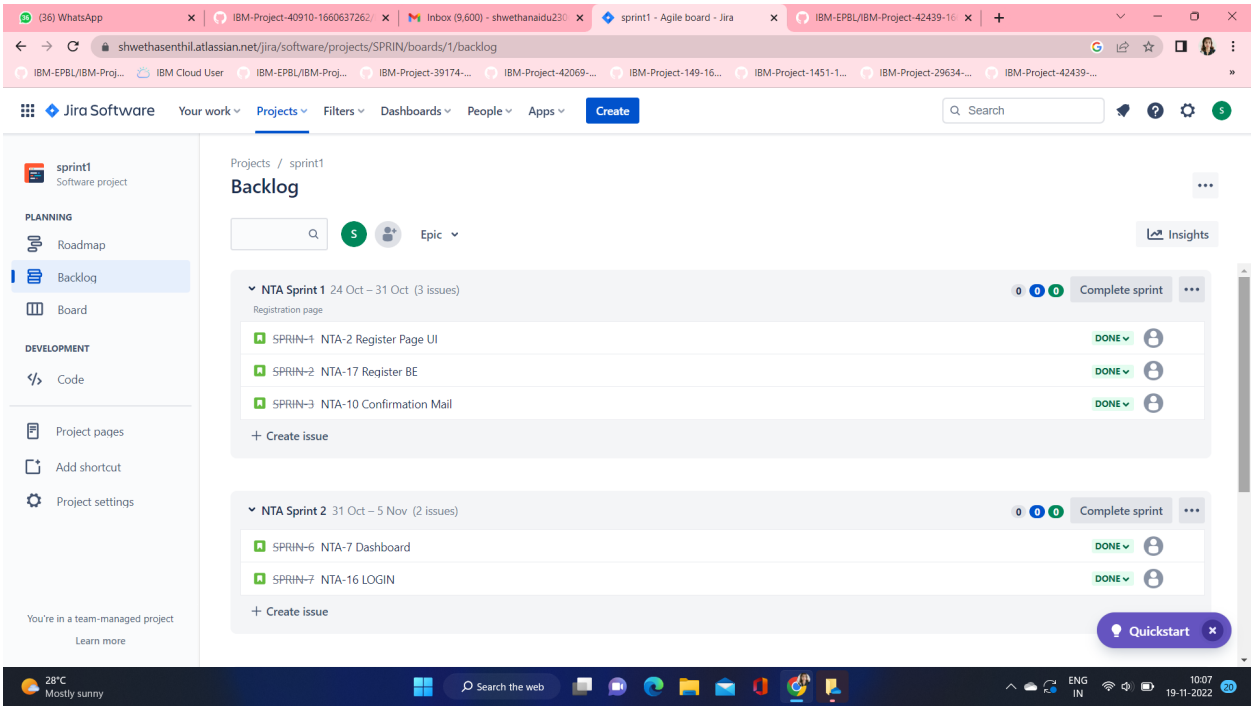
Sprint No.	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel	USN-1	The user will login into the website and gothrough the services available on the webpage	20	High	S.Shwetha P.Banu priya S.Vaishnavi M.Nanthini S.Dhanalakshmi
Sprint-2	Admin panel	USN-2	The role of the admin is to check out the database about the availability and have a trackof all the things that the users are going to service	20	High	S.Shwetha P.BanuPriya
Sprint-3	Chat Bot	USN-3	The user can directly talk to Chatbot regarding the services. Get the recommendations based on information provided by the user.	20	High	S.Vaishnavi M.Nanthini

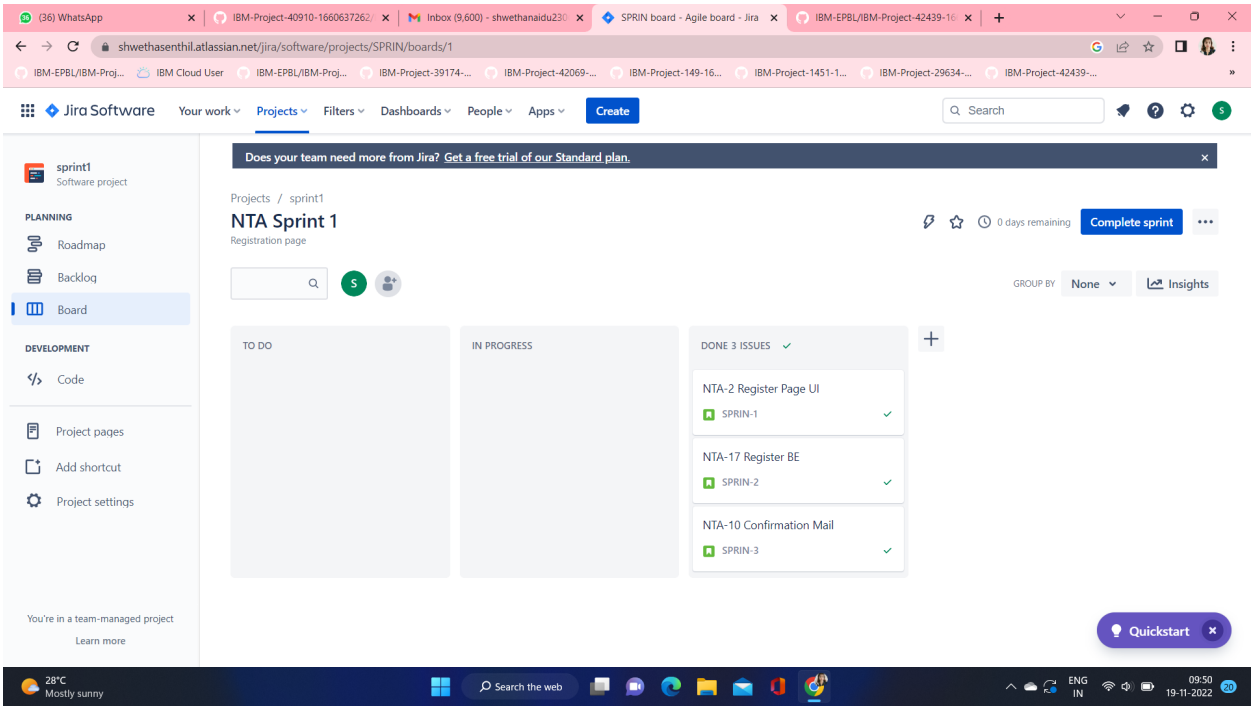
Sprint-4	final delivery	USN-4	Container of applications using docker kubernetes and deployment the application.Create the documentation and final submit the application	20	High	M.Nanthini P.Banu priya S.Vaishnavi S.Dhanalakshmi

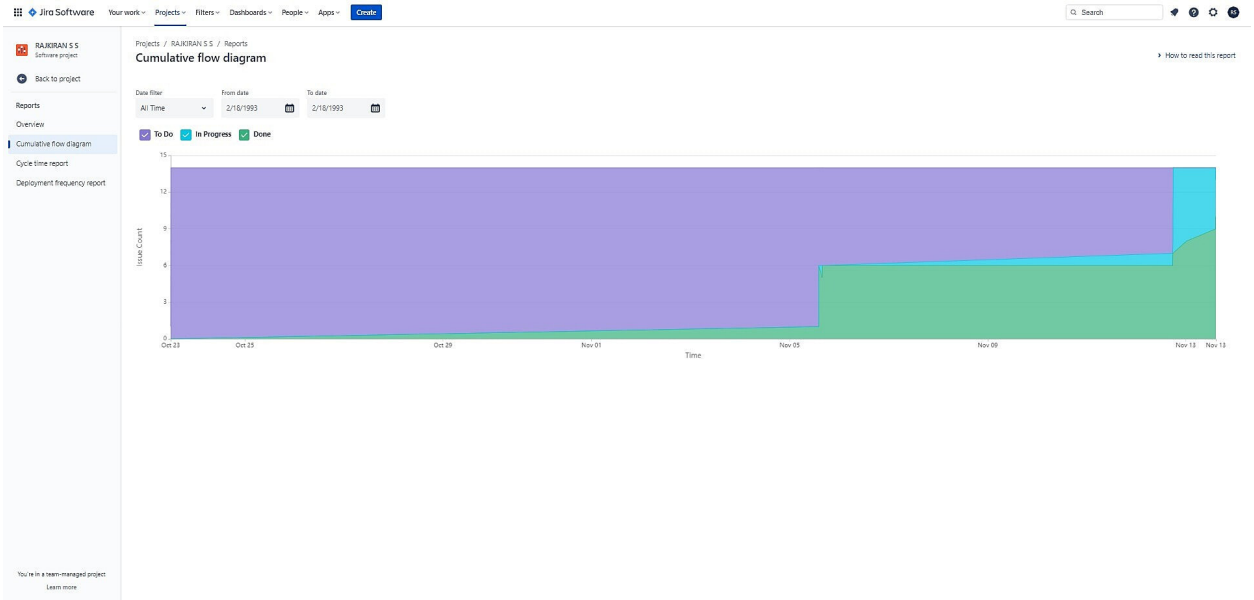
6.2 Sprint Delivery Schedule

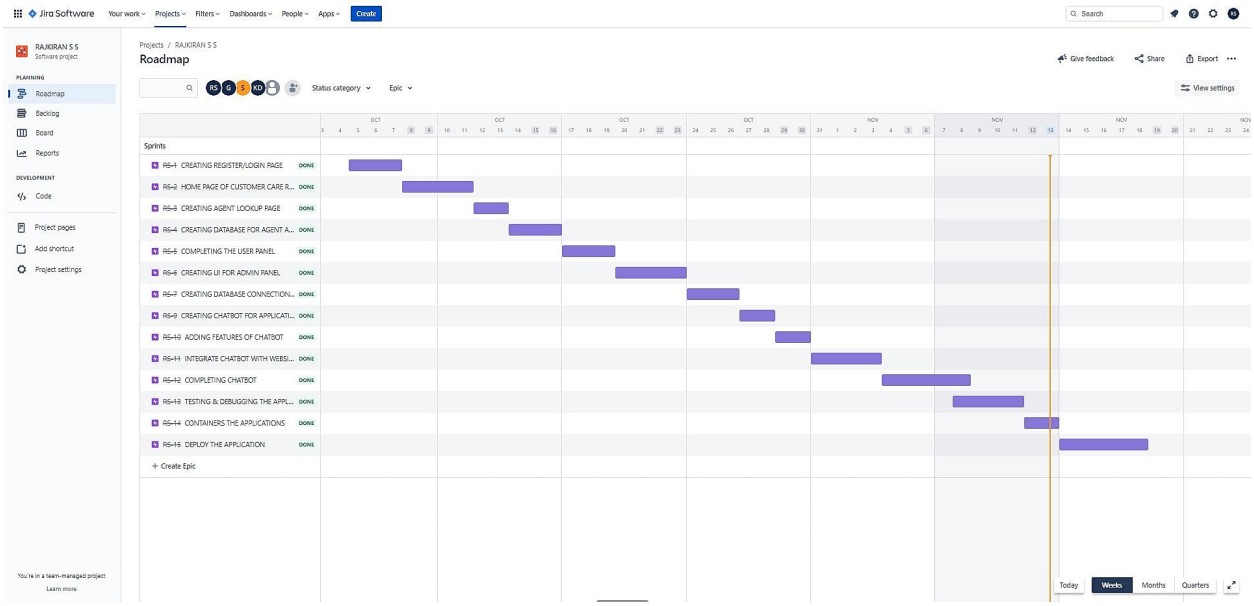
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022		29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		19 Nov 2022

6.3 Reports from JIRA






Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
RAJKIRAN S S
Software project
Back to project
Reports
Overview
Cumulative flow diagram
Cycle time report
Deployment frequency report
You're in a team-managed project
Learn more
Projects / RAJKIRAN S S / Reports
Cumulative flow diagram
How to read this report
Date filter
All Time
From date
2/18/1993
To date
2/18/1993
To Do
In Progress
Done
Issue Count
Time
Oct 23
Oct 25
Oct 29
Nov 01
Nov 05
Nov 09
Nov 13


Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
RAJKIRAN S S
Software project
PLANNING
Roadmap
Backlog
Board
Reports
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Projects / RAJKIRAN S S
Roadmap
Give feedback
Share
Export
Status category
Epic
View settings
Sprints
RS-1 CREATING REGISTER/LOGIN PAGE DONE
RS-2 HOME PAGE OF CUSTOMER CARE R... DONE
RS-3 CREATING AGENT LOOKUP PAGE DONE
RS-4 CREATING DATABASE FOR AGENT A... DONE
RS-5 COMPLETING THE USER PANEL DONE
RS-6 CREATING UI FOR ADMIN PANEL DONE
RS-7 CREATING DATABASE CONNECTION... DONE
RS-9 CREATING CHATBOT FOR APPLICATI... DONE
RS-10 ADDING FEATURES OF CHATBOT DONE
RS-11 INTEGRATE CHATBOT WITH WEBSI... DONE
RS-12 COMPLETING CHATBOT DONE
RS-13 TESTING & DEBUGGING THE APPL... DONE
RS-14 CONTAINERS THE APPLICATIONS DONE
RS-15 DEPLOY THE APPLICATION DONE
Create Epic
Today
Weeks
Months
Quarters

7. CODING & SOLUTIONING

7.1 Feature 1

1. Friendliness

This is the most basic customer need that's associated with things like courtesy and politeness. Friendly agents are a top indicator of a good customer experience, according to the customers surveyed in our 2021 Trends Report.

2. Empathy Customers need to know the organization understands and appreciates their needs and circumstances. In fact, 49% surveyed in our 2021 Trends Report said they want agents to be empathetic.

3. Fairness

Customers must feel that they're getting adequate attention and fair and reasonable answers.

4. Control

Customers want to feel like they have an influence on the outcome. You can empower your customers by listening to their feedback and using it to improve.

5. Alternatives

Customers want choice and flexibility from customer service; they want to know there is a range of options available to satisfy them. In fact, high-performing companies are more likely to provide customers with a choice of customer service channels. 50% of high performers ,compared to 18% of their lower-performing peers.

6. Information

Customers want to know about products and services in a pertinent and time-sensitive manner and selling can be off-putting for them. A knowledge base is a great way to provide existing customers with the information they need, when they need it. And highperforming CX teams are more likely to offer a knowledge base, according to our research.

7. Time

Customers' time is valuable, and organizations need to treat it as such. 73% of customers said resolving their issues quickly is the top component of a good customer experience. To deliver on that expectation, CX teams need customer service software that arms them with tools to respond to customers quickly and effectively.

7.2 Database Schema

Although the term "schema" is used in a wide variety of contexts, it most frequently refers to three distinct types of schema: conceptual database schemas, logical database schemas, and physical database schemas.Conceptual schemas provide a broad overview of the system's contents, organisational structure, and business rules. Typically, conceptual models are developed as a part of obtaining the initial project requirements. Comparatively speaking, logical database schemas are less abstract than conceptual schemas. Table names, field names, entity relationships any regulations governing the database are all well defined schema objects with information. The technical details that the logical database schema lacks are provided by physical database schemas.

8. TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	TC for Automat	Executed By
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	https://fbof7bov2d6crndrk8rora.on.drv.tw/www.shwetha.com/	Login/Signup popup should display	Working as expected	Pass	Y	Shwetha S
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	https://fbof7bov2d6crndrk8rora.on.drv.tw/www.shwetha.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	Working as expected	Pass	Y	S.Shwetha, P. Banupriya
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	Username: shwethanaidu230@gmail.com password: admin	User should navigate to user account homepage	Working as expected	Pass	Y	S.Vaishnavi

Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	TC for Automat	Executed By
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	Username: shwethanaidu230@gmail.com password: admin	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass	y	M.Nanthini
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	Username: shwethanaidu230@gmail.com	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass	y	P.Banupriya
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	password: admin	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass	y	S.Shwetha, P. Banupriya

8.2 User Acceptance Testing

Purpose of Document

The purpose of this document is to briefly explain the test coverageand open issuesof the inventory management system for retailer's project at the time of the release to User Acceptance Testing (UAT).

Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

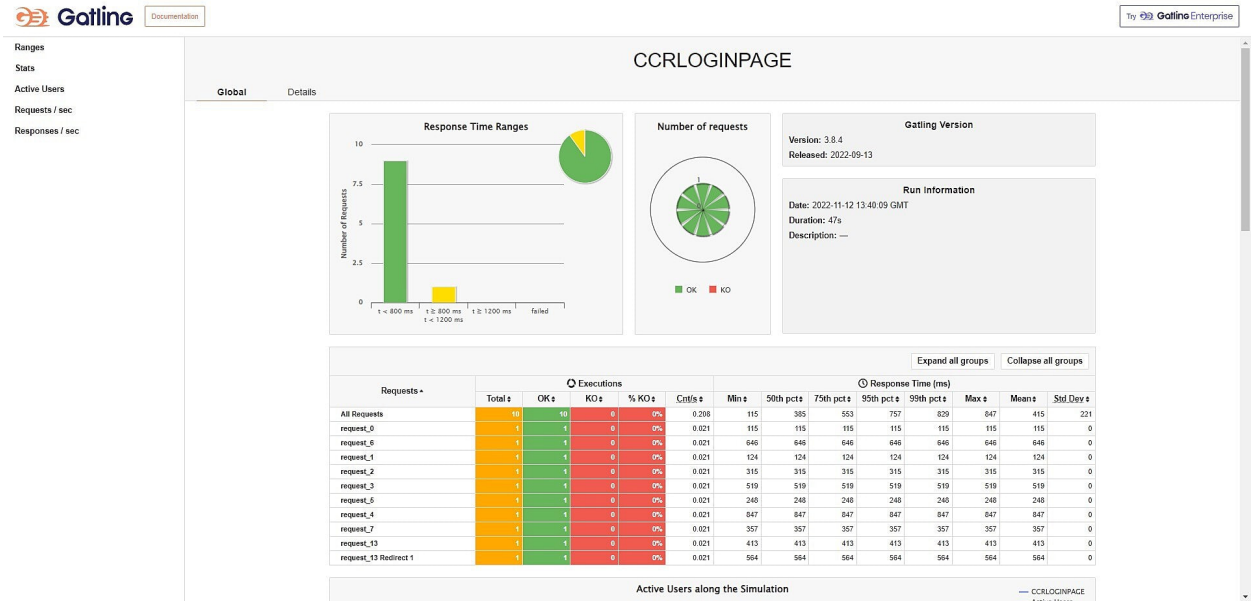
Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	8	0	0	8
Client Application	20	0	0	20
Security	2	0	0	2
OutsourceShipping	3	0	0	3
ExceptionReporting	12	0	0	12
Final Report Output	2	0	0	2
VersionControl	1	0	0	1

9. RESULTS

9.1 Performance Metrics



Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Cnt/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
All Requests	10	10	0	0%	0.208	115	385	553	757	829	847	415	221
request_0	1	1	0	0%	0.021	115	115	115	115	115	115	115	0
request_6	1	1	0	0%	0.021	646	646	646	646	646	646	646	0
request_1	1	1	0	0%	0.021	124	124	124	124	124	124	124	0
request_2	1	1	0	0%	0.021	315	315	315	315	315	315	315	0
request_3	1	1	0	0%	0.021	519	519	519	519	519	519	519	0
request_5	1	1	0	0%	0.021	248	248	248	248	248	248	248	0
request_4	1	1	0	0%	0.021	847	847	847	847	847	847	847	0
request_7	1	1	0	0%	0.021	357	357	357	357	357	357	357	0
request_13	1	1	0	0%	0.021	413	413	413	413	413	413	413	0
request_13 Redirect 1	1	1	0	0%	0.021	564	564	564	564	564	564	564	0

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

1. Improves Accuracy

Real-time inventory tracking helps you improve inventory management and ensures that have optimal stock available to fulfill orders. However, for most retail businesses, the inventory accuracy is merely 63%.

2. Reduces costs

Improving inventory management efficiency avoids chances of errors, and fewer errors eventually require fewer resources spent on fixing errors. Moreover, organized inventory management avoids overstocking and reduces the money spent on holding costs.

3. Saves Time

You can automate your inventory management process to save time in inventory forecasting and optimize the pick-pack process by leveraging robotics and AI. As you automate these tasks, you provide employees with ample time to work on more important tasks and devise strategies for business growth.

4. Improves Business Planning

Using an in-house inventory management strategy can help you get the business insights required to scale your business or improve operations further.Implementing features like barcode scanning and using a central data warehouse enables you to easily transfer data and monitor the happenings of your business.

5. Improves Customer Service

To sustain in today's competitive eCommerce space it is vital to provide your customers with a good shopping experience. Happy customers not only increase

the chances of repeat purchases but can help you drive more conversions with good reviews and word-of-mouth publicity.

DISADVANTAGES

1.System Crash

One of the biggest problems with any computerized system is the potential for a system crash**.** A corrupt hard drive, power outages and other technical issues can result in the loss of needed data. At the least, businesses are interrupted when they are unable to access data they need

2.Malicious Hacks

Hackers look for any way to get company or consumer information. An inventory system connected to point-of-sale devices and accounting is a valuable resource to hack into in search of potential financial information or personal details of owners, vendors or clients.

3.Reduced Physical Audits

When everything is automated, it is easy to forego time-consuming physical inventory audits. They may no longer seem necessary when the computers are doing their work. However, it is important to continue to do regular audits to identify loss such as spoilage or breakage.

4.Incorrect Information

Bookkeeping records are only as good as the data put into the system. Business owners that don't take the time to establish account categories properly may enter

data and generate reports that are not accurate.

5.Technical Issues

When dealing with computers, issues can arise. You may be completing year-end data for your accountant and experience a power outage. Computers might acquire a virus and fail. There is also the potential of users incorrectly performing software tasks that they are not familiar with.

11. CONCLUSION

Inventory management has to do with keeping accurate records of goods that are ready for shipment. This often means having enough stock of goods to the inventory totals as well as subtracting the most recent shipments of finished goods to buyers. When the company has a return policy in place, there is usually a sub-category contained in the finished goods inventory to account for any returned goods that are reclassified or second grade quality. Accurately maintaining figures on the finished goods inventory makes it possible to quickly convey information to sales personnel as to what is available and ready for shipment at any given time by buyer.

Inventory management is important for keeping costs down, while meeting regulation. Supply and demand is a delicate balance, and inventory management hopes to ensure that the balance is undisturbed. Highly trained Inventory management and high-quality software will help make Inventory management a success. The ROI of Inventory management will be seen in the forms of increased revenue and profits, positive employee atmosphere, and on overall increase of customer satisfaction.

12. FUTURE SCOPE

1. The Fourth Industrial Revolution will continue to drive technological change that will impact the way that we manage inventories.

2. Successful companies will view inventory as a strategic asset, rather than an aggravating expense or an evil to be tolerated.

3. Collaboration with supply chain partners, coupled with a holistic approach to supply chain management, will be key to effective inventory management.

4. The nature of globalization will change, impacting inventory deployment decisions dramatically.

5. Increased focus on supply chain security, and concerns about the quality of inventory itself, will be primary motivators to changing supply chain and inventory strategy.

13. APPENDIX

Source Code

ManageSales.html

```
<html>
    <head>
        <meta charset="utf-8">
        <title>MyFlaskApp</title>
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css">
    </head>
    <body>
        {% include 'includes/_navbar.html' %}
        <div class="container mt-4">
            {% include 'includes/_messages.html' %}
            {% block body %}{% endblock %}
        </div>
        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"></script>
</body>
</html>
```

Addsales.html

```
<html>
    <head>
```

```
        <meta charset="utf-8">
        <title>MyFlaskApp</title>
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css">
    </head>
    <body>
        {% include 'includes/_navbar.html' %}
        <div class="container mt-4">
            {% include 'includes/_messages.html' %}
            {% block body %}{% endblock %}
        </div>
        <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"></script>
</body>
</html>
```

edit_product.html

```
{% extends 'layout.html' %}
{% block body %}
<h1>Edit Product</h1>
{% from "includes/_formhelpers.html" import render_field %}
<form action="" method="POST">
    <div class="form-group">
        {{ render_field(form.product_id, class_="form-control") }}
    </div>
    <div class="form-group">
```

```
        {{ render_field(form.product_cost, class_="form-control") }}
    </div>
    <div class="form-group">
        {{ render_field(form.product_num, class_="form-control") }}
    </div>
    <p><input type="submit" value="Update" class="btn btn-primary"></p>
</form>
{% endblock %}
```

product_movement.html

```
{% extends 'layout.html' %}
{% block body %}
    <h1>Product Movements</h1>
    <a class="btn btn-success" href="/add_product_movements">Add Product
Movements</a>
    <hr>
    <table class="table table-striped">
        <thead>
            <tr>
                <th>Movement ID</th>
                <th>Time</th>
                <th>From Location</th>
                <th>To Location</th>
                <th>Product ID</th>
                <th>Quantity</th>
            </tr>
```

```
        </thead>
        <tbody>
            {% for movement in movements %}
            <tr>
            <td>{{movement.MOVEMENT_ID}}</td>
            <td>{{movement.TIME}}</td>
            <td>{{movement.FROM_LOCATION}}</td>
<td>{{movement.TO_LOCATION}}</td>
            <td>{{movement.PRODUCT_ID}}</td>
<td>{{movement.QTY}}</td>
            <!--<td><a href="edit_product_movement/{{movement.MOVEMENT_ID}}"
class="btn btn-primary pull-right">Edit</a></td>-->
            <td>
                <form action="{{url_for('delete_product_movements',
id=movement.MOVEMENT_ID)}}" method="POST">
                    <input type="hidden" name="method" value="DELETE">
                    <input type="submit" value="Delete" class="btn btn-danger">
                </form>
            </td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
{% endblock %}
```

app.py

```
from flask import Flask, render_template, flash, redirect, url_for, session, request, logging

from flask_mysqldb import MySQL

from wtforms import Form, StringField, TextAreaField, PasswordField,

validators, SelectField, IntegerField import ibm_db from passlib.hash import

sha256_crypt from functools import wraps import win32api from sendgrid import *

#creating an app instance app = Flask(__name__)

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=;PORT=;SECURITY=SSL;S SL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=;PWD=;","","")

#Index @app.route('/') def index():     return

render_template('home.html')

#Products @app.route('/products')

def products():

    sql = "SELECT * FROM products"     stmt

= ibm_db.prepare(conn, sql)

result=ibm_db.execute(stmt)     products=[]

row = ibm_db.fetch_assoc(stmt)

while(row):

            products.append(row)              row = ibm_db.fetch_assoc(stmt)

products=tuple(products)      #print(products)      if result>0:          return

render_template('products.html', products = products)      else:

        msg='No products found'         return render_template('products.html',

msg=msg)

#Locations @app.route('/locations')

def locations():

    sql = "SELECT * FROM locations"     stmt

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

```
result=ibm_db.execute(stmt)     locations=[]
row = ibm_db.fetch_assoc(stmt)     while(row):
        locations.append(row)
row = ibm_db.fetch_assoc(stmt)     locations=tuple(locations)
#print(locations)     if result>0:          return
render_template('locations.html', locations = locations)      else:
        msg='No locations found'         return
render_template('locations.html', msg=msg)
#Product Movements
@app.route('/product_movements') def
product_movements():      sql = "SELECT *
FROM productmovements"     stmt =
ibm_db.prepare(conn, sql)
result=ibm_db.execute(stmt)     movements=[]     row =
ibm_db.fetch_assoc(stmt)     while(row):
        movements.append(row)         row =
ibm_db.fetch_assoc(stmt)
movements=tuple(movements)
#print(movements)      if
result>0:
        return render_template('product_movements.html', movements = movements)      else:
        msg='No product movements found'         return
render_template('product_movements.html', msg=msg) #Register Form
Class class RegisterForm(Form):      name = StringField('Name',
[validators.Length(min=1, max=50)])
username = StringField('Username', [validators.Length(min=1, max=25)])     email =
```

```
StringField('Email', [validators.length(min=6, max=50)])     password =
PasswordField('Password', [         validators.DataRequired(),
validators.EqualTo('confirm', message='Passwords do not match')
    ])
    confirm = PasswordField('Confirm Password')
#user register
@app.route('/register', methods=['GET','POST']) def
register():
    form = RegisterForm(request.form)     if
request.method == 'POST' and form.validate():
        name = form.name.data         email = form.email.data         username =
form.username.data         password =
sha256_crypt.encrypt(str(form.password.data))      sql1="INSERT INTO users(name, email,
username, password) VALUES(?,?,?,?)"         stmt1 = ibm_db.prepare(conn, sql1)
ibm_db.bind_param(stmt1,1,name)         ibm_db.bind_param(stmt1,2,email)
ibm_db.bind_param(stmt1,3,username)         ibm_db.bind_param(stmt1,4,password)
ibm_db.execute(stmt1)
            #for flash messages taking parameter and the category of message to be flashed
flash("You are now registered and can log in", "success")                #when registration is
successful redirect to home          return redirect(url_for('login'))     return
render_template('register.html', form = form)
#User login
@app.route('/login', methods = ['GET', 'POST']) def login():
if request.method == 'POST':
#Get form fields         username =
request.form['username']         password_candidate =
```

```
request.form['password']     sql1="Select * from users where
username = ?"         stmt1 = ibm_db.prepare(conn, sql1)
ibm_db.bind_param(stmt1,1,username)
result=ibm_db.execute(stmt1)
d=ibm_db.fetch_assoc(stmt1)          if result > 0:             #Get
the stored hash             data = d
            password = data['PASSWORD']          #compare passwords             if
sha256_crypt.verify(password_candidate, password):
                #Passed                 session['logged_in'] =
True                 session['username'] = username
flash("you are now logged in","success")
return redirect(url_for('dashboard'))              else:
                error = 'Invalid Login'                 return render_template('login.html',
error=error)
            #Close connection              cur.close()
else:
                    error  =  'Username  not  found'                    return
render_template('login.html',     error=error)                    return
render_template('login.html')    #check  if  user  logged  in  def
is_logged_in(f):       @wraps(f)      def wrap(*args, **kwargs):
if  'logged_in'  in  session:                    return  f(*args, **kwargs)
else:
            flash('Unauthorized, Please
login','danger')             return redirect(url_for('login'))
return wrap
#Logout
```

```
@app.route('/logout') @is_logged_in def logout():
session.clear()     flash("You are now logged out",
"success")     return redirect(url_for('login'))
#Dashboard
@app.route('/dashboard' )
@is_logged_in def dashboard():
    sql2="SELECT product_id, location_id, qty FROM product_balance"
sql3="SELECT location_id FROM locations"     stmt2 =
ibm_db.prepare(conn, sql2)      stmt3 = ibm_db.prepare(conn, sql3)
result=ibm_db.execute(stmt2)
ibm_db.execute(stmt3)     products=[]     row =
ibm_db.fetch_assoc(stmt2)     while(row):
        products.append(row)         row =
ibm_db.fetch_assoc(stmt2)
products=tuple(products)     locations=[]
row2 = ibm_db.fetch_assoc(stmt3)
while(row2):          locations.append(row2)

row2 = ibm_db.fetch_assoc(stmt3)
locations=tuple(locations)     locs = []     for i in
locations:
        locs.append(list(i.values())[0])
if result>0:
        return render_template('dashboard.html', products = products, locations = locs)      else:
        msg='No products found'         return
render_template('dashboard.html', msg=msg)
```

```
#Product Form Class class
ProductForm(Form):
    product_id = StringField('Product ID', [validators.Length(min=1, max=200)])     product_cost
= StringField('Product Cost', [validators.Length(min=1, max=200)])     product_num =
StringField('Product Num', [validators.Length(min=1, max=200)])
#Add Product
@app.route('/add_product', methods=['GET', 'POST'])
@is_logged_in def
add_product():
    form = ProductForm(request.form)     if
request.method == 'POST' and form.validate():
        product_id = form.product_id.data         product_cost = form.product_cost.data
product_num = form.product_num.data       sql1="INSERT INTO products(product_id,
product_cost, product_num) VALUES(?,?,?)"         stmt1 = ibm_db.prepare(conn, sql1)
ibm_db.bind_param(stmt1,1,product_id)         ibm_db.bind_param(stmt1,2,product_cost)
ibm_db.bind_param(stmt1,3,product_num)            ibm_db.execute(stmt1)      flash("Product
Added", "success")    return redirect(url_for('products'))     return
render_template('add_product.html', form=form)

#Edit Product
@app.route('/edit_product/<string:id>', methods=['GET', 'POST'])
@is_logged_in def edit_product(id):
    sql1="Select * from products where product_id = ?"
stmt1 = ibm_db.prepare(conn, sql1)
ibm_db.bind_param(stmt1,1,id)
result=ibm_db.execute(stmt1)
product=ibm_db.fetch_assoc(stmt1)        print(product)     #Get
```

```
form     form =
ProductForm(request.form)

 #populate product form fields     form.product_id.data = product['PRODUCT_ID']
form.product_cost.data = str(product['PRODUCT_COST'])     form.product_num.data
= str(product['PRODUCT_NUM'])     if request.method ==
'POST' and form.validate():         product_id = request.form['product_id']
product_cost = request.form['product_cost']         product_num =
request.form['product_num']
        sql2="UPDATE products SET product_id=?,product_cost=?,product_num=? WHERE
product_id=?"         stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,product_id)         ibm_db.bind_param(stmt2,2,product_cost)
ibm_db.bind_param(stmt2,3,product_num)         ibm_db.bind_param(stmt2,4,id)
ibm_db.execute(stmt2)         flash("Product Updated", "success")         return
redirect(url_for('products'))     return render_template('edit_product.html', form=form)
#Delete Product
@app.route('/delete_product/<string:id>', methods=['POST'])
@is_logged_in def delete_product(id):
sql2="DELETE FROM products WHERE
product_id=?"

    stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,id)
ibm_db.execute(stmt2)     flash("Product
Deleted", "success")     return
redirect(url_for('products'))
#Location     Form      Class      class
LocationForm(Form):
```

```
    location_id = StringField('Location ID', [validators.Length(min=1, max=200)])
#Add Location
@app.route('/add_location', methods=['GET', 'POST'])
@is_logged_in def
add_location():
    form = LocationForm(request.form)     if
request.method == 'POST' and form.validate():         location_id =
form.location_id.data         sql2="INSERT into locations
VALUES(?)"         stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,location_id)
ibm_db.execute(stmt2)         flash("Location Added", "success")
return redirect(url_for('locations'))     return
render_template('add_location.html', form=form)
#Edit Location
@app.route('/edit_location/<string:id>', methods=['GET', 'POST'])
@is_logged_in def edit_location(id):
        sql2="SELECT * FROM locations where location_id = ?"
stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,id)
result=ibm_db.execute(stmt2)
location=ibm_db.fetch_assoc(stmt2)      #Get form
form = LocationForm(request.form)
print(location)

@app.route('/delete_product_movements/<string:id>', methods=['POST'])
@is_logged_in def delete_product_movements(id):
    sql2="DELETE FROM productmovements WHERE movement_id=?"
```

```
    stmt2 = ibm_db.prepare(conn, sql2)
ibm_db.bind_param(stmt2,1,id)     ibm_db.execute(stmt2)


    flash("Product Movement Deleted",
"success")     return
redirect(url_for('product_movements')) if
__name__ == '__main__':     app.secret_key =
"secret123"
    #when the debug mode is on, we do not need to restart the server again and again
app.run(debug=True)
```

GitHub link

https://github.com/IBM-EPBL/IBM-Project-40910-1660637262

Project Demo Link

https://youtu.be/PV_yM_ig2gM

Interactive web page – Watson enabled

https://fbof7bov2d6crndrk8rora.on.drv.tw/www.shwetha.com/