DEPARTMENT OF COMPUTER SCIENCE AND

(*NBA Accredited*)

GOVERNMENT COLLEGE OF ENGINEERING , SALEM – 11

(An Autonomous Institution Affiliated to Anna University, Chennai * NAAC Accredited)

IBM PROJECT

DOMAIN: Cloud App Development

TITLE : Retailer Management System for Retailers

TEAM ID: PNT2022TMID06765

DATE : 19.11.2022

BATCH MEMBERS

1. ANANDH J 61772021T301
2. GOKULA KRISHNAN G K 61772021T302
3. GOWTHAM S 61772021T303 – TEAMLEADER
4. HARIHARAN K 61772121T501

Faculty Advisor **HOD / CSE**

IDEATION PHASE

Submitted Date: 06.10.2022

The Problem statement of the project, Literature survey, Empathy map and Brainstorming for Idea prioritization were done.

PROJECT DESIGN PHASE-1

Submitted Date: 22.10.2022

The Problem solution fit, Proposed solution for the problem statement and Solution architecture were made.

PROJECT DESIGN PHASE-2

Submitted Date: 22.10.2022

The Solution requirements, Architecture of Technologies used, Data Flow diagrams and Customer journey were prepared.

PROJECT PLANNING PHASE

Submitted Date: 25.10.2022

The activity list is prepared and all four Sprint planning were made.

PROJECT DEVELOPMENT PHASE

Submitted Date: 10.11.2022

The Codes and Test cases are performed and uploaded for all four Sprint.

ASSIGNMENTS AND QUIZ

Completed all four Assignments and all four Quizzes.

COMMENTS

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Project Overview

Our project is **INVENTORY MANAGEMENT SYSTEM FOR RETAILERS** The project is cloud application development. The aim is to 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.

This issue must be solved so continuous supply of materials and stock so that production should not suffer at the time of customers demand. To avoid both overstocking and under-stocking of inventory.

1.2 Purpose

The project aims to help

- To avoid both overstocking and under-stocking of inventory.
- To eliminate duplication in ordering stocks.
- To ensure the quality of goods at reasonable prices.
- To minimize loss through deterioration, pilferage, wastages, and damages.

Users may have the facility to register to log in to the app and access the account by purchasing the product as their wish.

CHAPTER 2
LITERATURE SURVEY

A literature survey is "A survey of related literature refers to a study done before or after selecting a research problem to know about the previous research work, ideas, theories, procedures, techniques, problems occurring during the research, etc. is done for"

As purchasing of the product is increasingly accessed on smartphones and tablets, the need for personalizing app interactions is apparent. We report studies addressing key issues in the development of the adaptive app with good interfaces.

Title: **ZOHO**

Technologies:

Web Technologies: HTML, CSS, and javascript; Java.

Authors:

Mira Natarajan.

Description:

Zoho Inventory API is built using REST principles which ensures predictable URLs that make writing applications easy. This API follows HTTP rules, enabling a wide range of HTTP clients which can be used to interact with the API. Every resource is exposed as a URL. The URL of each resource can be obtained by accessing the API Root Endpoint.

Title: **Handifox:** Enhanced Inventory Management For Small Businesses

Technoliges:

Barcode systems and radio frequency identification (RFID) systems

Description:

With HandiFox, automating inventory management has never been easier. Utilizing mobile computers and barcodes, our inventory management software for small businesses simplifies your business process, making transparency and control easier than ever before - even across multiple sites.

Title: **Vend.**

Technologies :

• Warehouse Management System (WMS)

• Radio Frequency Identification (RFID) Technology.

Authors:

Novak V, Krzykovi M.

Description:

Manage inventory across multiple outlets with a centralized product catalog, accessible from your POS, back-office, or on the road. Edit products in bulk. Automatically reorder stock. Create variants and composites. Print labels and barcodes. We've even simplified stock-taking with our free app.

2.1 EXISTING PROBLEM:

- In the existing system categorization is not in detail.
- The features provided are clumsy and not user-friendly and adaptive.

- Irregular patterns of products are fed which are not location-based and of top priority.
- Irrelevant product details are fed which are either inappropriate or no longer interesting.

2.2 REFERENCES:

- https://www.zoho.com/in/inventory/
- https://www.vendhq.com/asia/
- https://www.handifox.com/

2.3 PROBLEM STATEMENT DEFINITION

A problem statement is a concise description of the problem or issues a project seeks to address. The problem statement identifies the current state, the desired future state, and any gaps between the two.

- ❖ Can manage the stocks using mobile devices.
- ❖ Users do not need to pay for using the application.
- ❖ Reduce time-consuming frustration.
- ❖ Users can access accounts of their own choice.
- ❖ Feed the user with relevant products instantly based on their interests.
- ❖ Prefer their regular purchasing product based on their activity.

CHAPTER 3

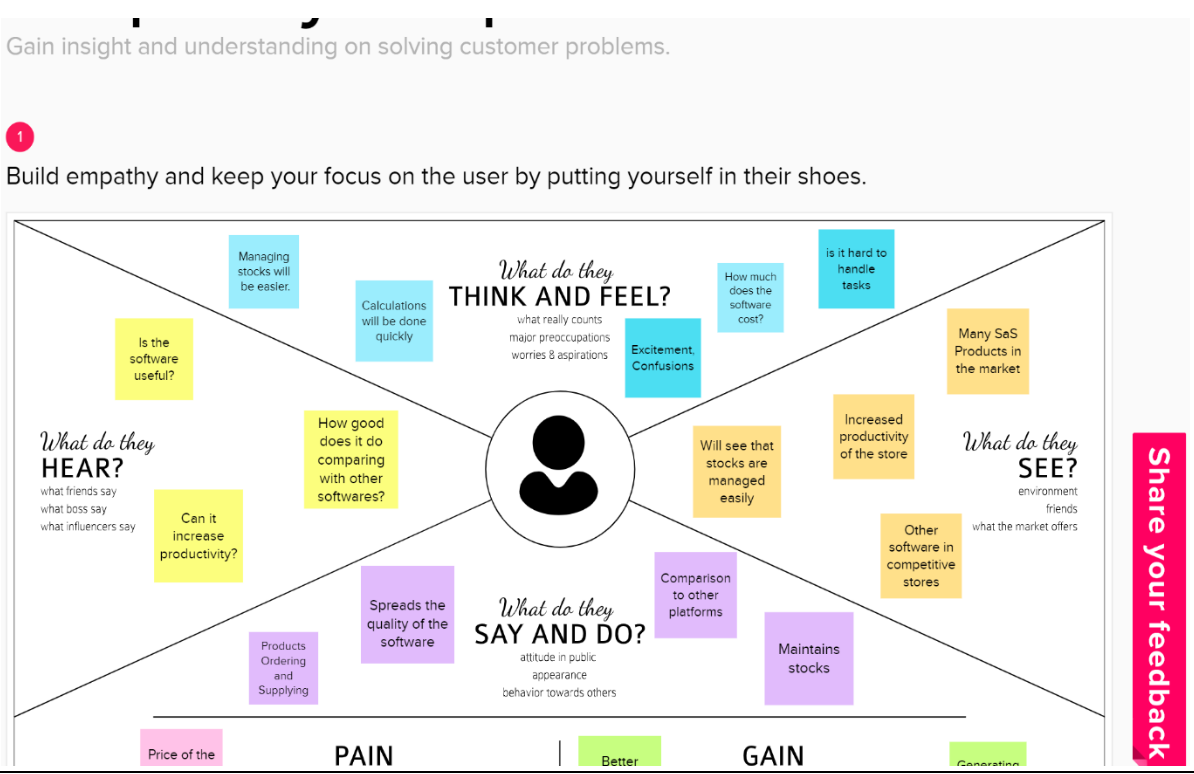
IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

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

The empathy map is used to analyze the problem from the user's perspective such as,

- What do they think and feel?
- What do they see?
- What do they hear?
- What do they say and do?
- The pains and gains of Inventory Management System for Retailers


Gain insight and understanding on solving customer problems.

1

Build empathy and keep your focus on the user by putting yourself in their shoes.

What do they THINK AND FEEL?
what really counts
major preoccupations
worries 8 aspirations
Managing stocks will be easier.
Calculations will be done quickly
Excitement, Confusions
How much does the software cost?
is it hard to handle tasks
What do they HEAR?
what friends say
what boss say
what influencers say
Is the software useful?
How good does it do comparing with other softwares?
Can it increase productivity?
What do they SEE?
environment
friends
what the market offers
Many SaS Products in the market
Increased productivity of the store
Will see that stocks are managed easily
Other software in competitive stores
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Spreads the quality of the software
Products Ordering and Supplying
Comparison to other platforms
Maintains stocks
PAIN
Price of the
GAIN
Better
Generating
Share your feedback

3.2 Ideation & Brainstorming

Brainstorming is a group problem-solving method that involves the spontaneous contribution of creative ideas and solutions.

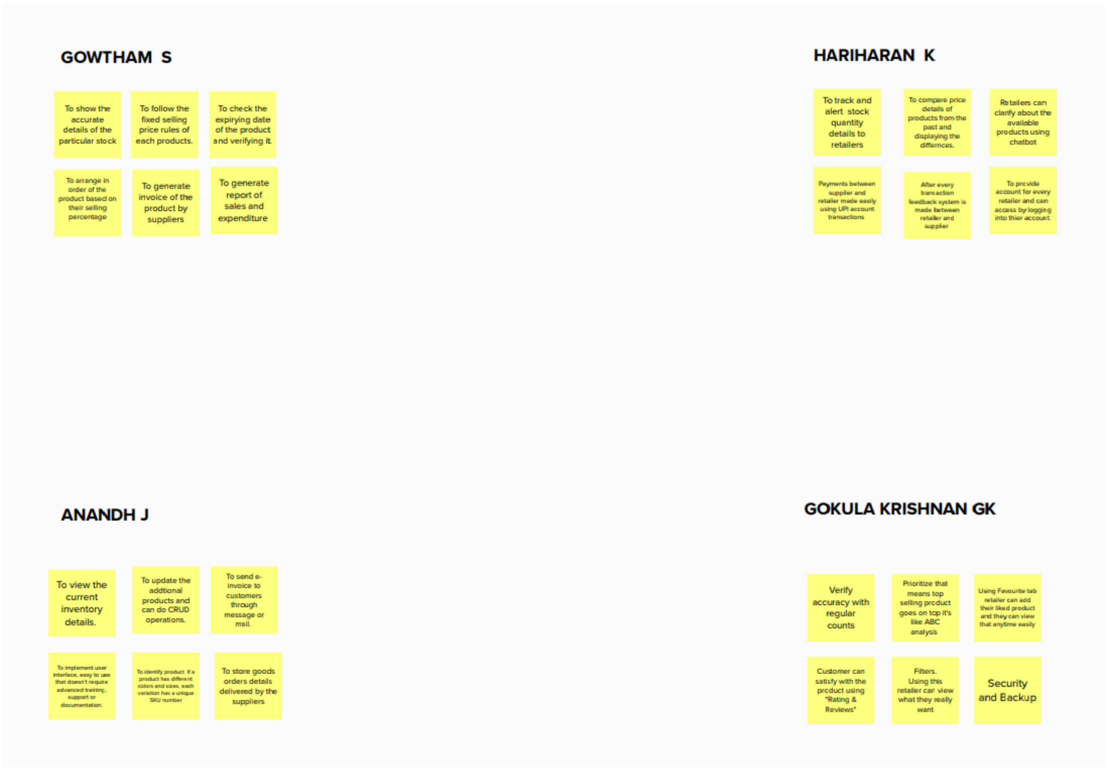
Brainstorming allows people to think more freely, without fear of judgment. Brainstorming encourages open and ongoing collaboration to solve problems and generate innovative ideas. Brainstorming helps teams generate a large number of ideas quickly, which can be refined and merged to create the ideal solution.

Idea prioritization has been performed in the following ways

1. Define problem statement :

To avoid inconsistency, irrelevant and irregular patterns of purchasing products over the websites and provide a one-stop solution.

2. Brainstorm
   Write down any ideas that come to mind that address your problem statement.



GOWTHAM S

- To show the accurate details of the particular stock
- To follow the fixed selling price rules of each products.
- To check the expirying date of the product and verifying it.
- To arrange in order of the product based on their selling percentage
- To generate invoice of the product by suppliers
- To generate report of sales and expenditure

HARIHARAN K

- To track and alert stock quantity details to retailers
- To compare price details of products from the past and displaying the differnces.
- Retailers can clarify about the available products using chatbot
- Payments between suppiler and retailer made easily using UPI account transactions
- After every transaction feedback system is made between retailer and supplier
- To provide account for every retailer and can access by logging into thier account.

ANANDH J

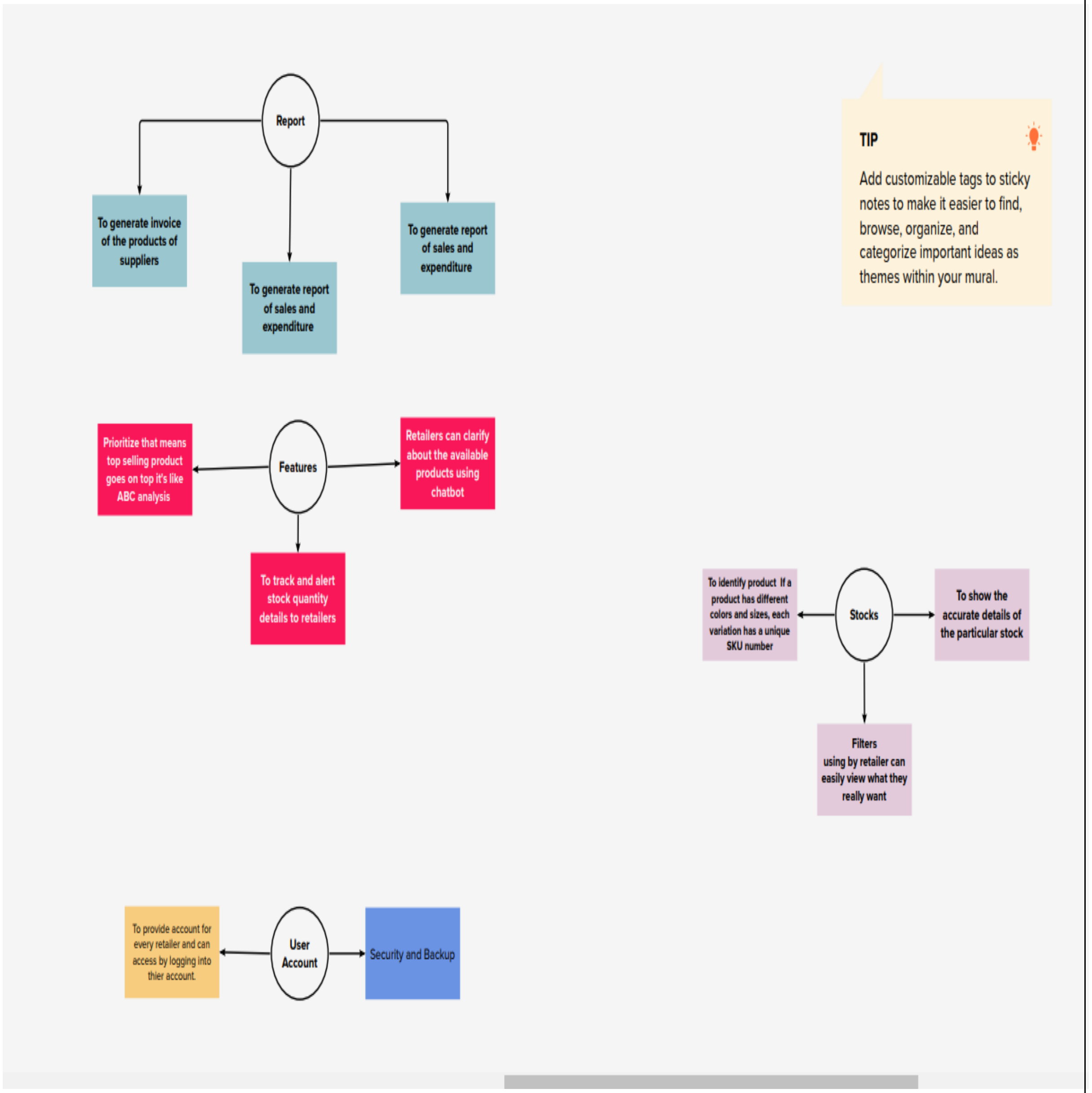
- To view the current inventory details.
- To update the addtional products and can do CRUD operations.
- To send e-invoice to customers through message or mail.
- To implement user interface, easy to use that doesn't require advanced training, support or documentation.
- To identify product If a product has different colors and sizes, each variation has a unique SKU number
- To store goods orders details delivered by the suppliers

GOKULA KRISHNAN GK

- Verify accuracy with regular counts
- Prioritize that means top selling product goes on top it's like ABC analysis
- Using Favourite tab retailer can add their liked product and they can view that anytime easily
- Customer can satisfy with the product using "Rating & Reviews"
- Filters. Using this retailer can view what they really want
- Security and Backup

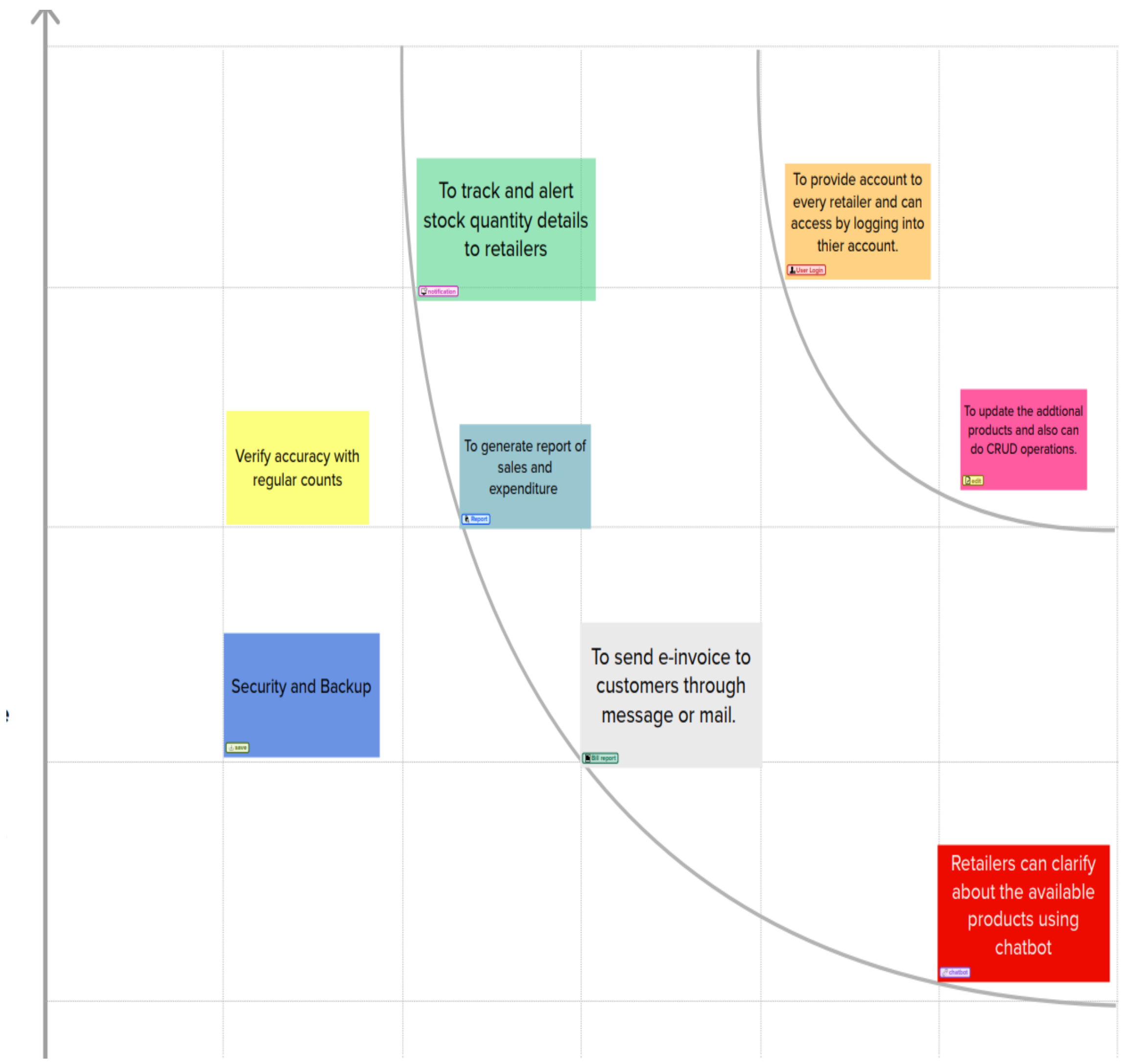
3. Group Ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, 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.


Report
To generate invoice of the products of suppliers
To generate report of sales and expenditure
To generate report of sales and expenditure
TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.
Features
Prioritize that means top selling product goes on top it's like ABC analysis
Retailers can clarify about the available products using chatbot
To track and alert stock quantity details to retailers
Stocks
To identify product If a product has different colors and sizes, each variation has a unique SKU number
To show the accurate details of the particular stock
Filters using by retailer can easily view what they really want
User Account
To provide account for every retailer and can access by logging into thier account.
Security and Backup

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.


To track and alert stock quantity details to retailers
notification
To provide account to every retailer and can access by logging into thier account.
User Login
Verify accuracy with regular counts
To generate report of sales and expenditure
Report
To update the addtional products and also can do CRUD operations.
edit
Security and Backup
save
To send e-invoice to customers through message or mail.
Bill report
Retailers can clarify about the available products using chatbot
chatbot

3.3 Proposed Solution

1. Problem Statement (Problem to be solved)

   Retailers are generally facing problems with maintaining the availability of stocks. Not possible to know which product is sold more and which is not.

2. Idea / Solution description

   The proposed software will help retailers track the stock's availability very often and efficiently. The proposed system can give filter options on how much each product is sold and can give insights accordingly. Novelty / Uniqueness

3. Novelty / Uniqueness

Emailing the monthly report of every product and product reaching its threshold limits. The products are arranged in a manner such that most sold products are at the top of the list.

4. Social Impact / Customer Satisfaction

Retailers will save so much time by using this software. Customers can give their feedback and ratings about particular products.

5. Business Model (Revenue Model)

Discounts can be offered to customers based on their amount of purchase. Can give discounts on the least sold products.

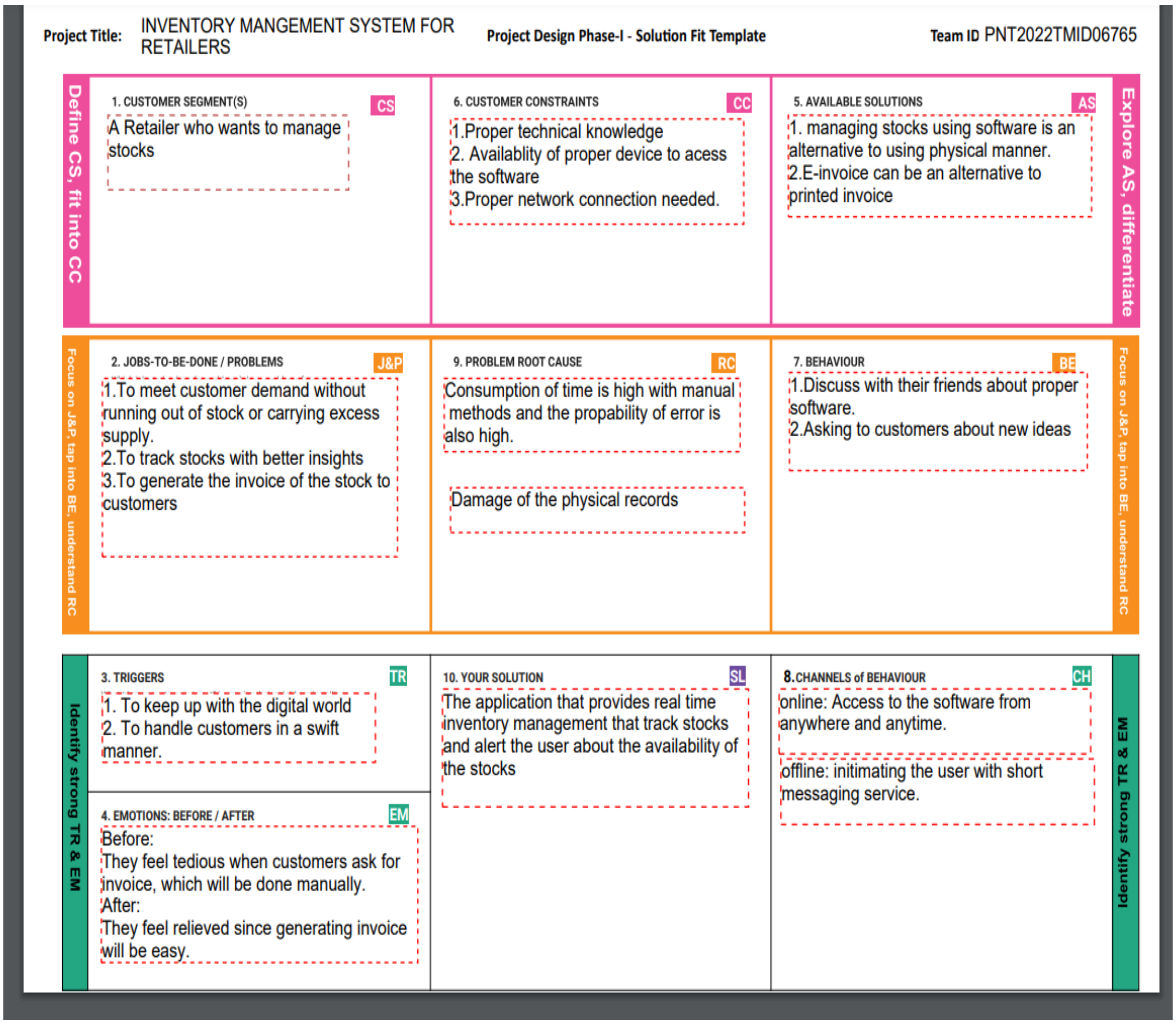
6. Scalability of the Solution

Customers can be in the comfort of their homes while shopping. Retailers can manage large volumes of data in the future with this software.

3.3 Problem Solution Fit

Problem solution fit defines the following,

- Define Customer Segment fit into customer constraints Explore Available resources and differentiate
- Focus on Job-to-be-done & problems, tap into behavior, understand the root cause
- Identify strong Triggers and emotions, channel of behaviour and your solutions



Project Title: INVENTORY MANGEMENT SYSTEM FOR RETAILERS

Project Design Phase-I - Solution Fit Template

Team ID PNT2022TMID06765

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

A Retailer who wants to manage stocks

6. CUSTOMER CONSTRAINTS CC

1.Proper technical knowledge
2. Availablity of proper device to acess the software
3.Proper network connection needed.

5. AVAILABLE SOLUTIONS AS

1. managing stocks using software is an alternative to using physical manner.
2.E-invoice can be an alternative to printed invoice

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

1.To meet customer demand without running out of stock or carrying excess supply.
2.To track stocks with better insights
3.To generate the invoice of the stock to customers

9. PROBLEM ROOT CAUSE RC

Consumption of time is high with manual methods and the propability of error is also high.

Damage of the physical records

7. BEHAVIOUR BE

1.Discuss with their friends about proper software.
2.Asking to customers about new ideas

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

1. To keep up with the digital world
2. To handle customers in a swift manner.

4. EMOTIONS: BEFORE / AFTER EM

Before:
They feel tedious when customers ask for invoice, which will be done manually.
After:
They feel relieved since generating invoice will be easy.

10. YOUR SOLUTION SL

The application that provides real time inventory management that track stocks and alert the user about the availability of the stocks

8.CHANNELS of BEHAVIOUR CH

online: Access to the software from anywhere and anytime.

offline: initimating the user with short messaging service.

Identify strong TR & EM

CHAPTER 4

REQUIREMENT ANALYSIS

Requirement analysis is “the process of determining user expectations for a new or modified product. These requirements must be quantifiable, relevant, and detailed”.

4.1 Functional Requirements:

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks.

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through Username and Password
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User login	Login through the browser directly by entering your username and password
FR-4	User interaction	Can view the product details The order required products by putting them in a cart first.

4.1 Non-Functional Requirements:

A Non-functional requirement is a requirement that does not relate to functionality, but to attributes such as reliability, efficiency, usability, maintainability and portability.

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

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	Designing or developing the site to be having a learning curve. Having simple and easy user interface to navigate website for users. Attractive looking web-page. Making the website to be responsive for both desktops and mobile users
NFR-2	**Security**	The security should be strong as to the attackers won’t be penetrating to the authorized users account or data. Log in system is used to prove authentication and authorization. Security can be increased by using OTP. Cookies based security system for authentication and improved visiting experience on the site for clients.
NFR-3	**Reliability**	Should be having the capacity to handle enough users and not be lagging or experiencing any discomfort when browsing when the web-page is busy. Should have minimum errors when executing the programs. Should be available even at the times of calamity.

NFR-5	**Availability**	This uses IBM DB2 to ensure the high availability of database servers and performances.
NFR-6	**Scalability**	As DB2 is highly scalable, the coding can be produced and developed efficiently and new features can be introduced easily. Reusing the code can be done to add any new features. IBM Container in Docker registry is used which is highly scalable.
NFR-4	**Performance**	The convenience of this is it reduces the time period of searching for desired product, etc. It reduces costs, saves time, restocking period, and predicts the bestselling products. This makes the business more productive and profitable by having an organized management system.

CHAPTER 5
PROJECT DESIGN

5.1 Data flow diagram

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

The DFD belongs to structured-analysis modelling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

Advantages of DFD

- It helps us to understand the functioning and the limits of a system.
- It is a graphical representation which is very easy to understand as it helps visualize contents.
- Data Flow Diagram represent detailed and well explained diagram of system components.
- It is used as the part of system documentation file.
- Data Flow Diagrams can be understood by both technical or nontechnical person because they are very easy to understand.

Disadvantages of DFD

- At times DFD can confuse the programmers regarding the system.
- Data Flow Diagram takes long time to be generated, and many times due to this reasons analysts are denied permission to work on it.

Rules for creating DFD

- The name of the entity should be easy and understandable without any extra assistance(like comments).
- The processes should be numbered or put in ordered list to be referred easily.
- The DFD should maintain consistency across all the DFD levels.
- A single DFD can have maximum processes upto 9 and minimum 3 processes.

Data Flow Diagram

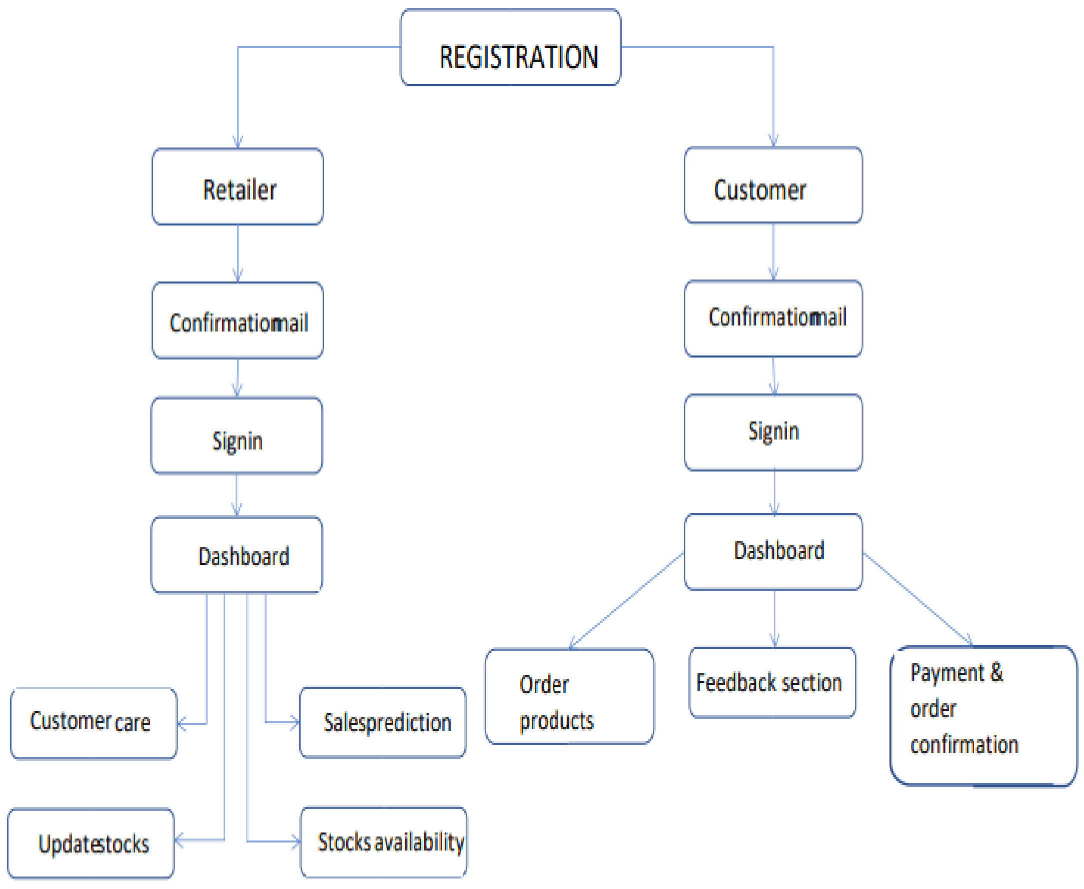


Fig 5.1: Data flow diagram

5.2 Solution & Technical Architecture

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behaviour-, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

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.

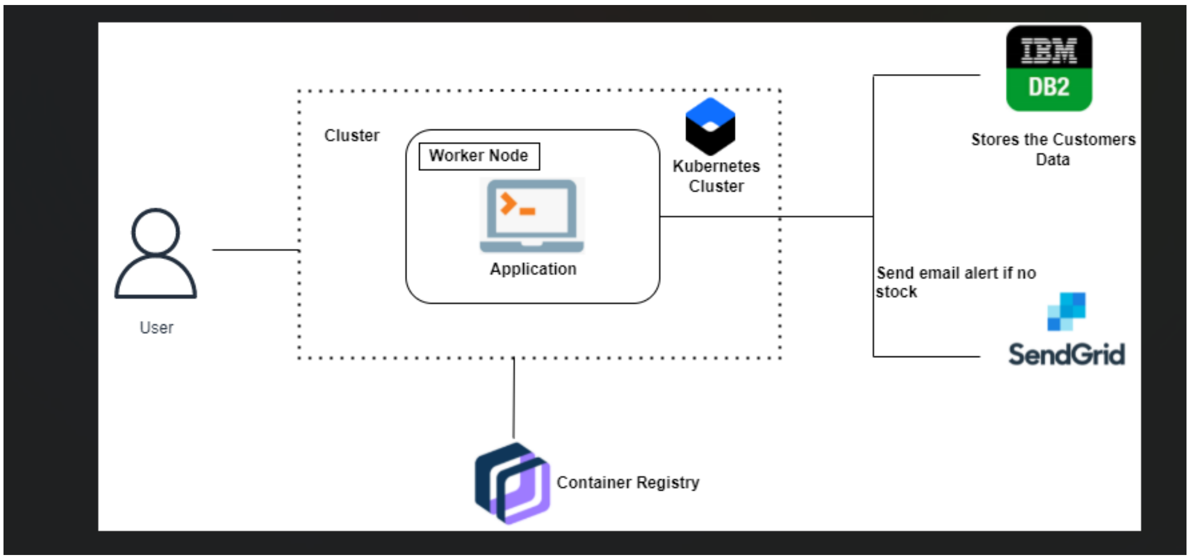


Fig 5.2: Solution Architecture

Project Business Model

All users will be able to use the app for free. providing fundamental solutions enabling a single view of the business from finance to the customer experience. In short, solutions focus on helping retailers manage intelligent processes in a digital world. cloud is opening the doors for a new level of co-innovation with customers and partners, enabling next practices and new business models that can help you capture opportunities and take the next step toward becoming an Intelligent Enterprise. Launched in 2020 and building momentum ever since industry cloud is essentially an innovation space where customers and partners are building and integrating cloud solutions to address specific industry needs.

Characteristics and behaviors

- It will be simple for us to scale the application to a bigger set of users because it requires the same set of input from all users and does not carry out a lot of complicated computations.
- As users can add their own interests as well, networks get more complicated, and suggestions get better.
- Users can make friends, and suggestions can be adjusted as a result. Users are asked about their preferences when creating an account, and users are given the option to rate the relevancy of articles in order to enhance suggestions.

Specifications of solution

The APIs are doing great work on this application. They gather the product from all the mentioned websites. Later all data are going under the cleaner phase. Here these unorganized data are organized by the methods

- Validating (The Originality of the product checked overall data)
- Filtering (eliminating duplicates)
- Categorizing (using keywords e.g.: -Grocery, Electronics)

This organized data will be stored in the IBM cloud. In this populous world, every second is data, so the above process will be continuously working for a certain time period to fetch product details.

When the user/customer log-in to the application, they have to give their priority.

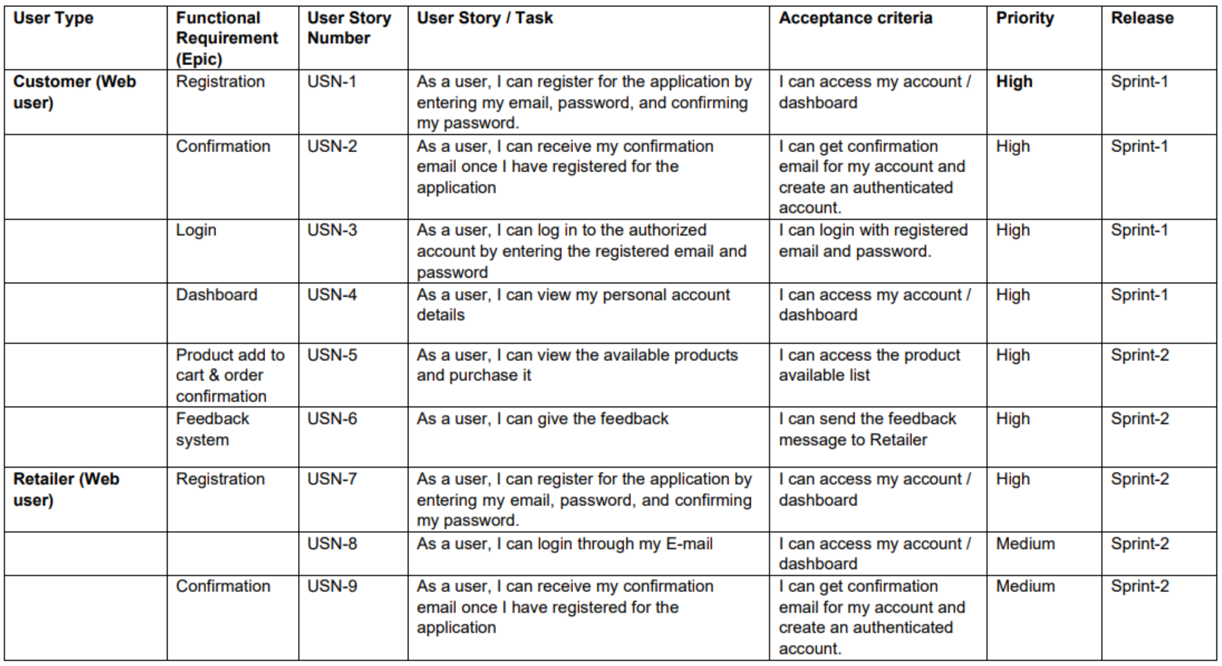
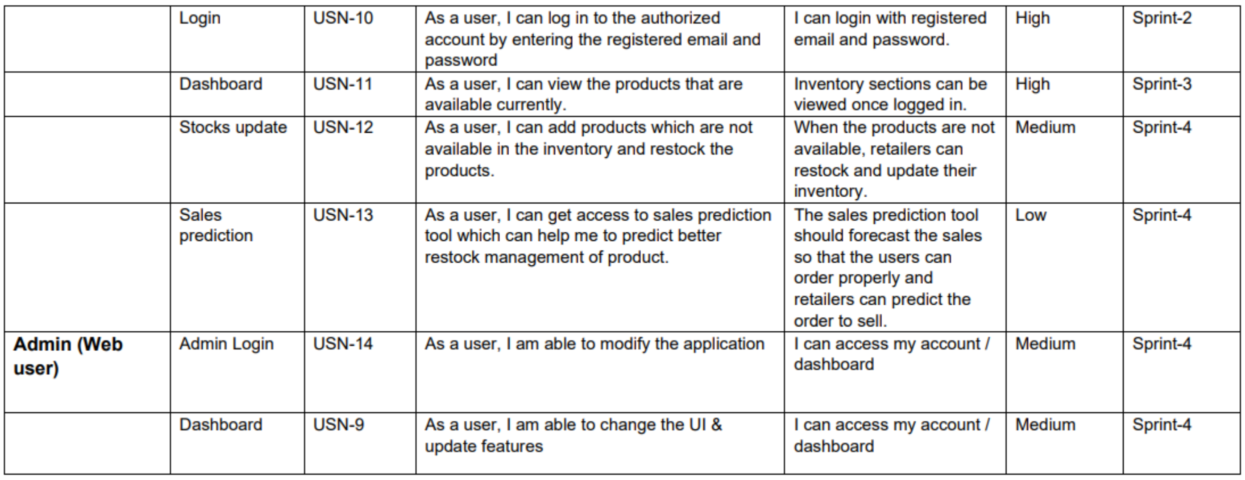
These priorities will be also stored in the cloud with that user’s info.

These priorities are the core to portrait only preferred news as per the user to avoid user inconvenience and have an improved interaction towards this application.

The user can also gather other preferred products by exploring the search bar in the application. The user’s request will be sent to the database then the response (The user’s searched product) will be published on the page.

5.3 User Stories

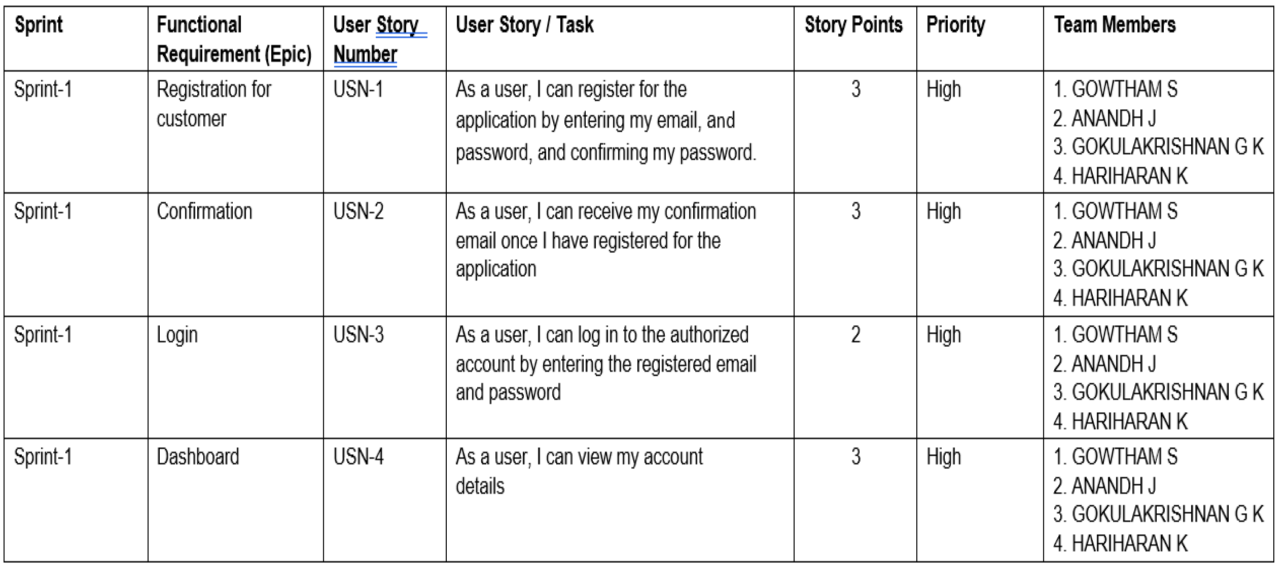
Use the below template to list all the user stories for the product.

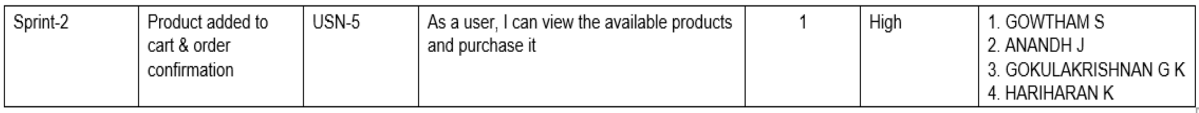
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	**High**	Sprint-1
	Confirmation	USN-2	As a user, I can receive my confirmation email once I have registered for the application	I can get confirmation email for my account and create an authenticated account.	High	Sprint-1
	Login	USN-3	As a user, I can log in to the authorized account by entering the registered email and password	I can login with registered email and password.	High	Sprint-1
	Dashboard	USN-4	As a user, I can view my personal account details	I can access my account / dashboard	High	Sprint-1
	Product add to cart & order confirmation	USN-5	As a user, I can view the available products and purchase it	I can access the product available list	High	Sprint-2
	Feedback system	USN-6	As a user, I can give the feedback	I can send the feedback message to Retailer	High	Sprint-2
Retailer (Web user)	Registration	USN-7	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-2
		USN-8	As a user, I can login through my E-mail	I can access my account / dashboard	Medium	Sprint-2
	Confirmation	USN-9	As a user, I can receive my confirmation email once I have registered for the application	I can get confirmation email for my account and create an authenticated account.	Medium	Sprint-2
	Login	USN-10	As a user, I can log in to the authorized account by entering the registered email and password	I can login with registered email and password.	High	Sprint-2
	Dashboard	USN-11	As a user, I can view the products that are available currently.	Inventory sections can be viewed once logged in.	High	Sprint-3
	Stocks update	USN-12	As a user, I can add products which are not available in the inventory and restock the products.	When the products are not available, retailers can restock and update their inventory.	Medium	Sprint-4
	Sales prediction	USN-13	As a user, I can get access to sales prediction tool which can help me to predict better restock management of product.	The sales prediction tool should forecast the sales so that the users can order properly and retailers can predict the order to sell.	Low	Sprint-4
Admin (Web user)	Admin Login	USN-14	As a user, I am able to modify the application	I can access my account / dashboard	Medium	Sprint-4
	Dashboard	USN-9	As a user, I am able to change the UI & update features	I can access my account / dashboard	Medium	Sprint-4

CHAPTER-6

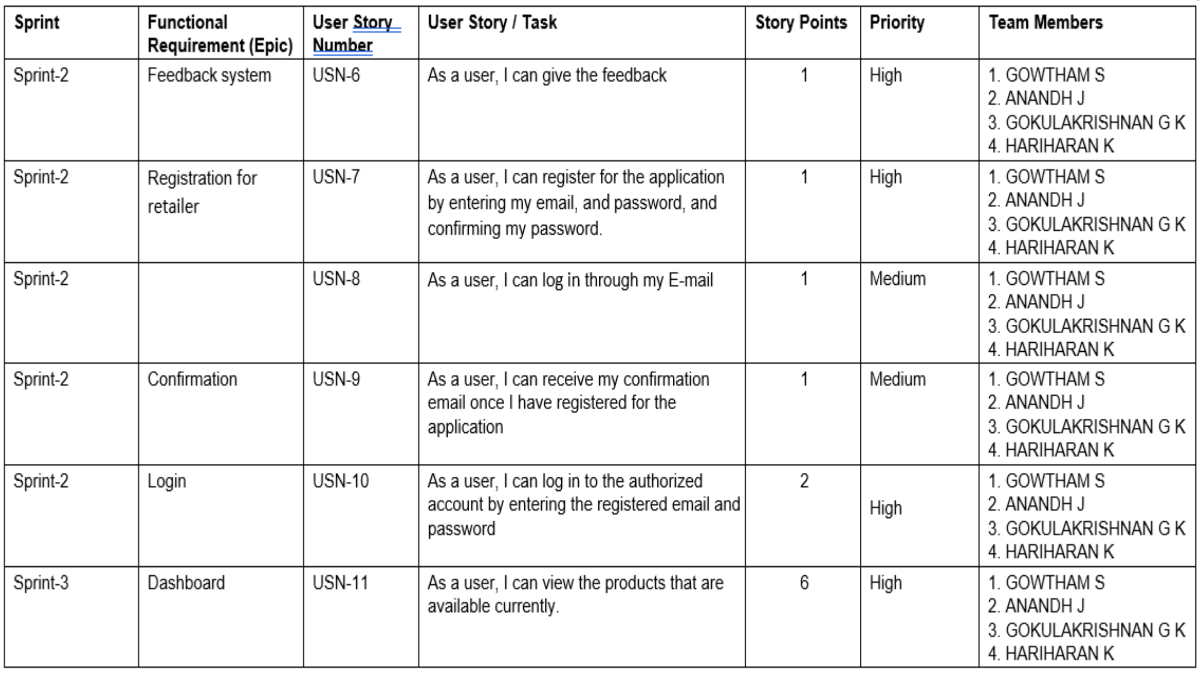
6.1 Sprint Planning & Estimation



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration for customer	USN-1	As a user, I can register for the application by entering my email, and password, and confirming my password.	3	High	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K
Sprint-1	Confirmation	USN-2	As a user, I can receive my confirmation email once I have registered for the application	3	High	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K
Sprint-1	Login	USN-3	As a user, I can log in to the authorized account by entering the registered email and password	2	High	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K
Sprint-1	Dashboard	USN-4	As a user, I can view my account details	3	High	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K



Sprint-2	Product added to cart & order confirmation	USN-5	As a user, I can view the available products and purchase it	1	High	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K



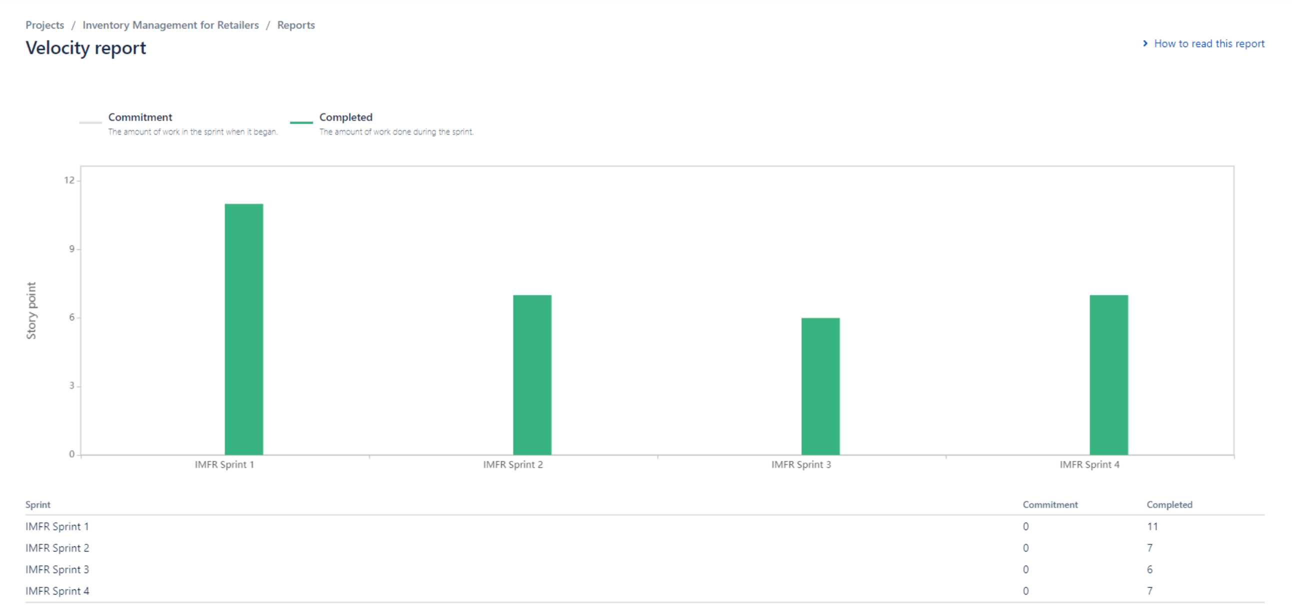
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Feedback system	USN-6	As a user, I can give the feedback	1	High	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K
Sprint-2	Registration for retailer	USN-7	As a user, I can register for the application by entering my email, and password, and confirming my password.	1	High	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K
Sprint-2		USN-8	As a user, I can log in through my E-mail	1	Medium	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K
Sprint-2	Confirmation	USN-9	As a user, I can receive my confirmation email once I have registered for the application	1	Medium	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K
Sprint-2	Login	USN-10	As a user, I can log in to the authorized account by entering the registered email and password	2	High	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K
Sprint-3	Dashboard	USN-11	As a user, I can view the products that are available currently.	6	High	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K

Sprint-4	Stocks update	USN-12	As a user, I can add products that are not available in the inventory and restock the products.	2	Medium	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K
Sprint-4	Stock Alert	USN-13	Alerting the retailer when the stock is low on quantity by using SendGrid	2	High	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K
Sprint-4	Dashboard	USN-14	As a user, I can change the UI & update features	3	Medium	1. GOWTHAM S 2. ANANDH J 3. GOKULAKRISHNAN G K 4. HARIHARAN K

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	11	6 Days	24 Oct 2022	29 Oct 2022	11	29 Oct 2022
Sprint-2	7	6 Days	31 Oct 2022	05 Nov 2022	7	05 Nov 2022
Sprint-3	6	6 Days	07 Nov 2022	12 Nov 2022	6	12 Nov 2022
Sprint-4	7	6 Days	14 Nov 2022	19 Nov 2022	7	19 Nov 2022

6.3 Reports from JIRA

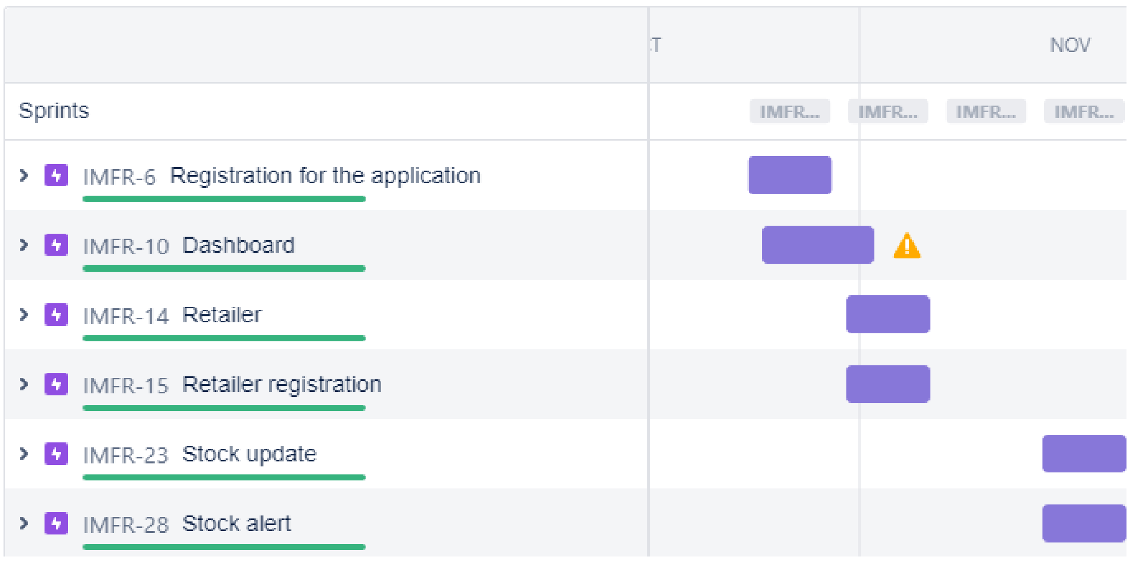

Projects / Inventory Management for Retailers / Reports

Velocity report

› How to read this report

Commitment
The amount of work in the sprint when it began.
Completed
The amount of work done during the sprint.
Story point
12
9
6
3
0
IMFR Sprint 1
IMFR Sprint 2
IMFR Sprint 3
IMFR Sprint 4

Sprint	Commitment	Completed
IMFR Sprint 1	0	11
IMFR Sprint 2	0	7
IMFR Sprint 3	0	6
IMFR Sprint 4	0	7



Sprints

IMFR-6 Registration for the application
IMFR-10 Dashboard
IMFR-14 Retailer
IMFR-15 Retailer registration
IMFR-23 Stock update
IMFR-28 Stock alert

CHAPTER-7

CODING & SOLUTIONING

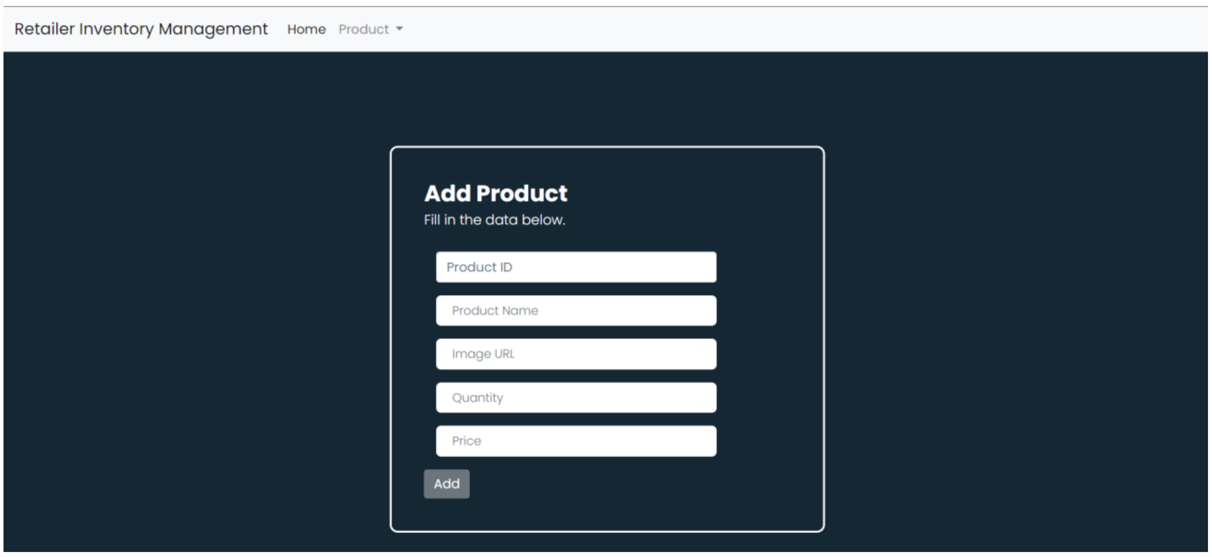
7.1 Feature 1

Stock Update:

```
{% extends "basetemplate.html" %}
{% block content %}
{% with messages = get_flashed_messages() %}
   {% if messages %}
      {% for message in messages %}
         <h1 id="msg" display="none" style="background-color:
white;">{{message}}</h1>
      {% endfor %}
   {% endif %}
{% endwith %}
<body>


    <div class="form-body">
    <div class="row">
        <div class="form-holder">
            <div class="form-content">
                <div class="form-items">
                    <h3>Add Product</h3>
                    <p>Fill in the data below.</p>
                    <form class="requires-validation" action="/addproduct"
method="post">
                        <div class="col-md-10">
```

```
                             <input class="form-control" type="number"
name="productid" placeholder="Product ID" required>
                          </div>
                         <div class="col-md-10">
                            <input class="form-control" type="text"
name="productname" placeholder="Product Name" required>
                         </div>

                         <div class="col-md-10">
                             <input class="form-control" type="text" name="imageurl"
placeholder="Image URL" required>
                          </div>

                         <div class="col-md-10">
                             <input class="form-control" type="text" name="quantity"
placeholder="Quantity" required>
                         </div>

                        <div class="col-md-10">
                           <input class="form-control" type="text" name="price"
placeholder="Price" required>
                        </div>









                         <div  class="form-button mt-3">
                             <button id="submit" type="submit" class="btn btn-
primary">Add</button>
                         </div>
                     </form>
                 </div>
             </div>
         </div>
    </div>
</div>
<script type="text/javascript">

   $("#submit").click(function(){
  $("#msg").fadeOut(2000);
});
  </script>
</body>

{% endblock %}
```

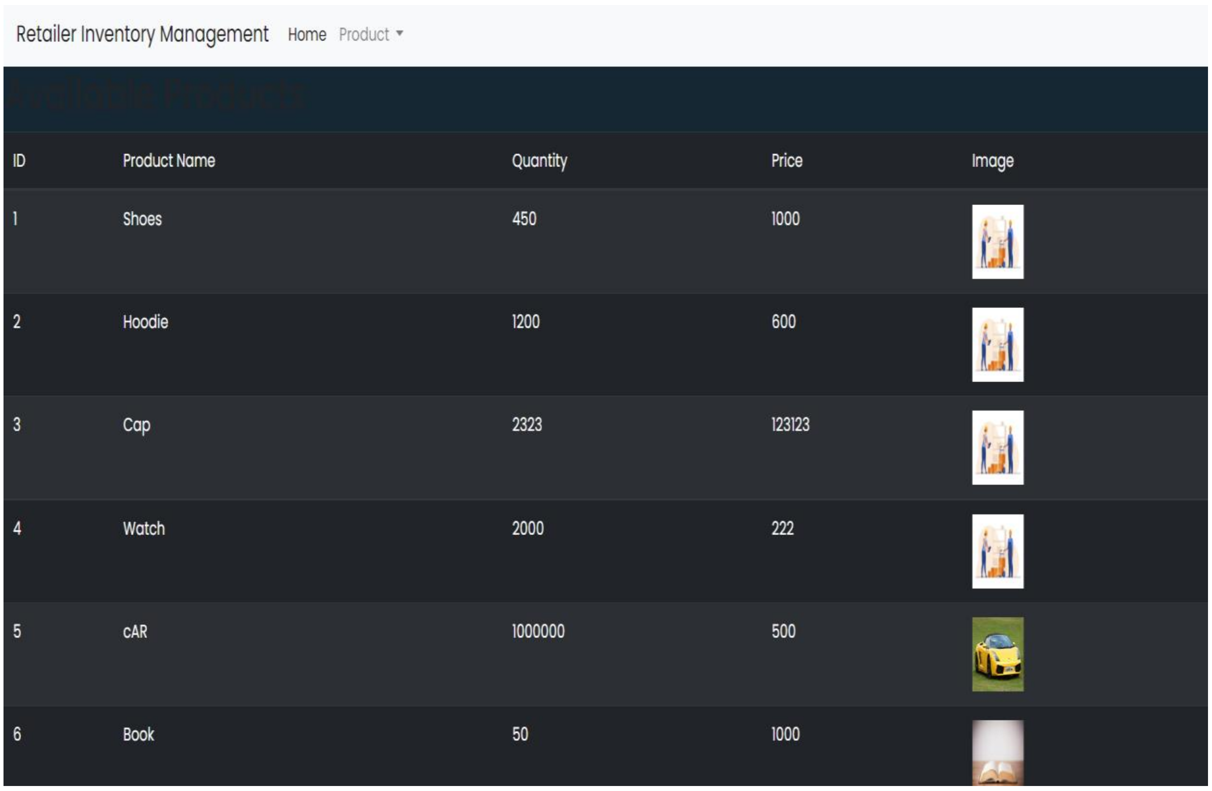
Output:



Stocks Display

```
{% extends "basetemplate.html" %}
{% block content %}
<h1>Available Products</h1>
<table class="table table-striped table-dark">
    <thead>
      <tr>
        <th scope="col">ID</th>
        <th scope="col">Product Name</th>
        <th scope="col">Quantity</th>
        <th scope="col">Price</th>
        <th scope="col">Image</th>
      </tr>
    </thead>
    <tbody>
      {%for item in product%}
      <tr>
        <th scope="row">{{item['PRODUCTID']}}</th>
        <td>{{item['PRODUCTNAME']}}</td>
        <td>{{item['PRICE']}}</td>
        <td>{{item['QUANTITY']}}</td>
        <td><img src='{{item["IMGURL"]}}' alt=""></td>
      </tr>
      {% endfor %}

    </tbody>
  </table>
{% endblock %}
```

Output:



Available Products

ID	Product Name	Quantity	Price	Image
1	Shoes	450	1000	
2	Hoodie	1200	600	
3	Cap	2323	123123	
4	Watch	2000	222	
5	cAR	1000000	500	
6	Book	50	1000	

7.2 Feature 2

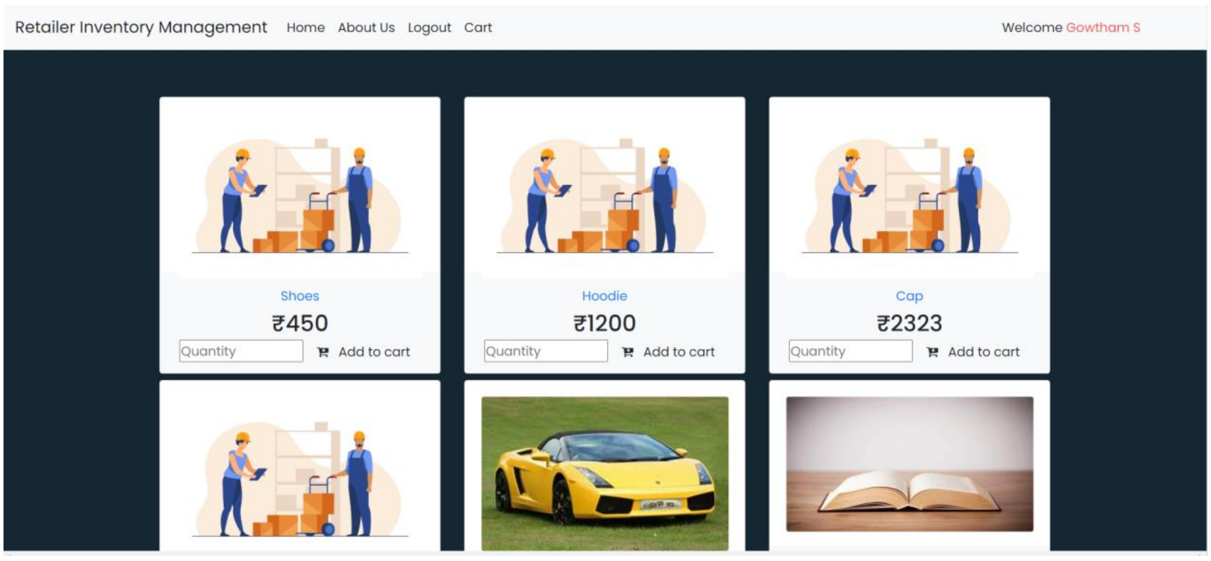
Add to cart

```
{% extends "basetemplate.html" %}
{% block content %}
{% with messages = get_flashed_messages() %}
   {% if messages %}
      {% for message in messages %}
         <h1 id="msg" display="none" style="background-color:
white;">{{message}}</h1>
      {% endfor %}
   {% endif %}
{% endwith %}
<body>


    <div class="form-body">
    <div class="row">
        <div class="form-holder">
            <div class="form-content">
                <div class="form-items">
                    <h3>Add Product</h3>
                    <p>Fill in the data below.</p>
```

```html
                    <form class="requires-validation" action="/addproduct"
method="post">
                        <div class="col-md-10">
                            <input class="form-control" type="number"
name="productid" placeholder="Product ID" required>
                         </div>
                        <div class="col-md-10">
                           <input class="form-control" type="text"
name="productname" placeholder="Product Name" required>
                        </div>

                        <div class="col-md-10">
                            <input class="form-control" type="text" name="imageurl"
placeholder="Image URL" required>
                         </div>

                        <div class="col-md-10">
                            <input class="form-control" type="text" name="quantity"
placeholder="Quantity" required>
                        </div>

                       <div class="col-md-10">
                          <input class="form-control" type="text" name="price"
placeholder="Price" required>
                       </div>









                        <div  class="form-button mt-3">
                            <button id="submit" type="submit" class="btn btn-
primary">Add</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
<script type="text/javascript">

   $("#submit").click(function(){
  $("#msg").fadeOut(2000);
});
  </script>
</body>
```

```
{% endblock %}
```

Output:

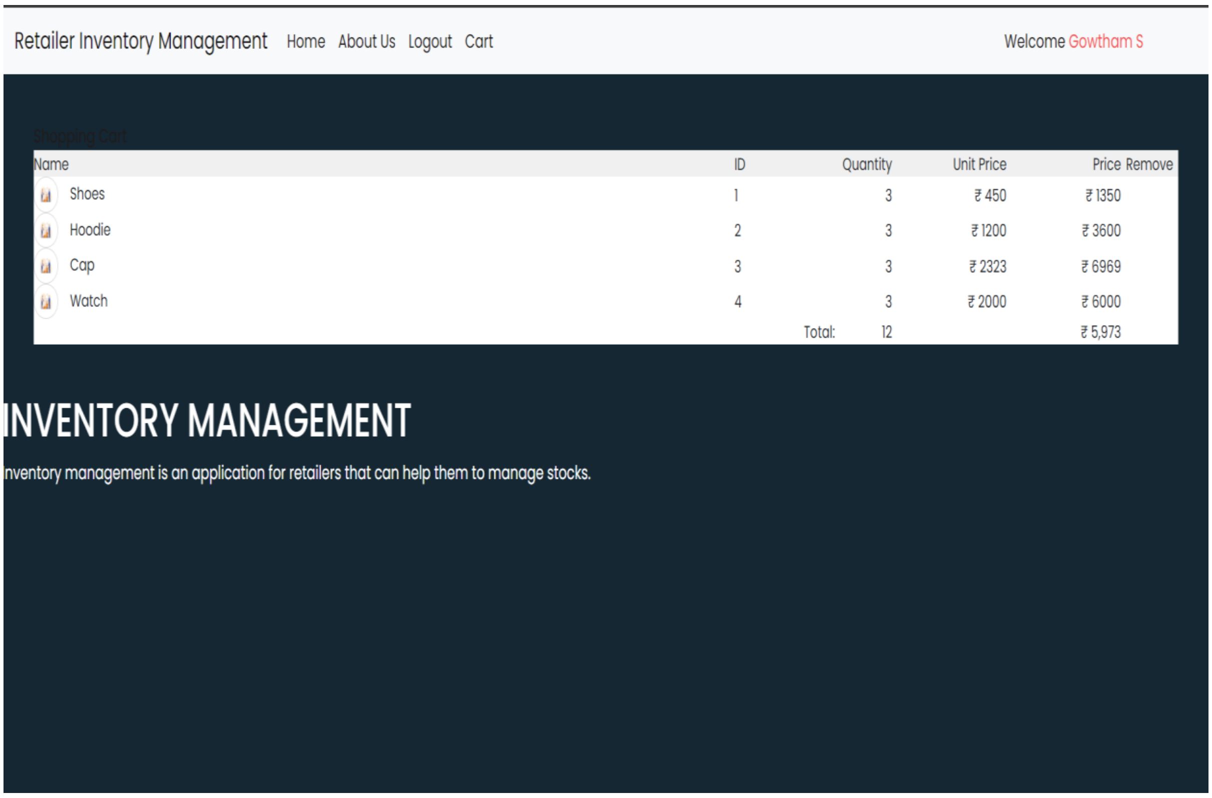


Retailer Inventory Management Home About Us Logout Cart Welcome Gowtham S

Shoes
₹450
Quantity Add to cart

Hoodie
₹1200
Quantity Add to cart

Cap
₹2323
Quantity Add to cart



Retailer Inventory Management Home About Us Logout Cart Welcome Gowtham S

Shopping Cart

Name	ID	Quantity	Unit Price	Price Remove
Shoes	1	3	₹ 450	₹ 1350
Hoodie	2	3	₹ 1200	₹ 3600
Cap	3	3	₹ 2323	₹ 6969
Watch	4	3	₹ 2000	₹ 6000
	Total:	12		₹ 5,973

INVENTORY MANAGEMENT

nventory management is an application for retailers that can help them to manage stocks.

CHAPTER 10
ADVANTAGES & DISADVANTAGES

Advantages

- This project helps to ensure a continuous supply of materials and stock so that production should not suffer at the time of customer demand.
- To avoid both overstocking and under-stocking of inventory.
- To maintain the availability of materials whenever and wherever required in enough quantity.
- To maintain minimum working capital as required for operational and sales activities.
- To optimize various costs indulged with inventories like purchase cost, carrying a cost, storage cost, etc.
- To keep material costs under control as they contribute to reducing the cost of production.
- Ensure everlasting inventory control so that materials shown in stock ledgers are physically lying in the warehouse
- To facilitate the furnishing of data for short and long-term planning with a controlled inventory.

Disadvantages

- Sales prediction
- Unclear communication
- Stocks damage

CHAPTER 11
CONCLUSION

Inventory Management System for Retailers can be used by the business organization to manage their stocks and records easily. Achieving this objective is difficult using the manual system as the information is scattered, can be redundant, and collecting relevant information may be very time-consuming. All these problems are solved by this project. This system helps in maintaining the information of pupils of the organization. It can be easily accessed by persons who and kept safe for a long period of time without any changes.

Inventory management systems make more accessible to Retailers by giving them an easy place to find and sort information. This system allows customers and Retailers to easily communicate and do their requirements.

This project is intended d to serve as the easiest way to handle the Stock

database in a centralized manner which can be accessed by any person with the required credentials.

To conclude, this project works like a component that can access all the databases. It overcomes the many limitations in stock management.

- Easy implementation environment
- Generate reports flexibly.

CHAPTER 12

FUTURE SCOPE

The project has a very vast scope in the future. The project can be implemented on

The intranet in the future. The project can be updated near future as and when hen requirement for the same arises, as it is very flexible in terms of expansion. With the proposed software of database Space Manager ready and fully functional, the client is now able to manage and hence run the entire work in a much better, more accurate, and error-free manner.

CHAPTER 13

APPENDIX

Source code:

app.py

```
from flask import Flask, render_template,request,flash,redirect,url_for,session
import ibm_db
import sendgrid
import os
from dotenv import load_dotenv
from sendgrid.helpers.mail import Mail, Email, To, Content


load_dotenv()
app = Flask(__name__)
app.secret_key="123"
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=6667d8e9-9d4d-4ccb-ba32-
21da3bb5aafc.c1ogj3sd0tgtu01qde00.databases.appdomain.cloud;PORT=30376;SECURITY=SSL
;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=rwc61888;PWD=kjMZoqMxzEKqkAAL",'
','')



@app.route('/')
def index():
```

```python
    stmt = "SELECT * FROM PRODUCT"
    sql = ibm_db.prepare(conn,stmt)
    ibm_db.execute(sql)
    dictionary = ibm_db.fetch_assoc(sql)
    # print(dictionary)
    a=[]
    while dictionary != False:
        a.append(dictionary)
        # print(dictionary)
        dictionary = ibm_db.fetch_assoc(sql)
    return render_template('index.html', product=a)

@app.route('/login',methods=["GET","POST"])
def login():
    if request.method=='POST':
        email=request.form['email']
        password=request.form['password']
        sql = "SELECT * FROM CUSTOMER WHERE email =? AND password = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        # print(account)
        if account:
            session["name"]=account["NAME"]
            session["mail"]=account["EMAIL"]
            session["cart_item"]=True
            # print(session)
            return redirect(url_for('index'))
        else:
            flash("Username and Password Mismatch","danger")
    return render_template('login.html')


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

            password2= request.form['password2']
            name=request.form['name']
            password = request.form['password']
            mail=request.form['email']
            print(name)
            if(len(password2)<1 or len(name)<1 or len(mail)<1):
                flash("Error in Insert Operation")
                return redirect(url_for('register'))
            if(password!=password2):
                flash("Password mismatch")
                return redirect(url_for('register'))
            sql = "SELECT * FROM CUSTOMER WHERE name =?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt,1,name)
            ibm_db.execute(stmt)
            account = ibm_db.fetch_assoc(stmt)
            if account:
                return render_template('login.html',msg="Account already exists 
please login")
            insert_sql = "INSERT INTO CUSTOMER VALUES (?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, mail)
            ibm_db.bind_param(prep_stmt, 3, password2)
            ibm_db.execute(prep_stmt)

            flash("Record Added  Successfully","success")

            sg = 
sendgrid.SendGridAPIClient(api_key=os.environ.get('SENDGRID_API_KEY'))
            from_email = Email("gowthamjpm534@gmail.com")  # Change to your 
verified sender
            to_email = To(mail)  # Change to your recipient
            subject = "Registration successful  "
            content = Content("text/plain", "You have successfully registered to 
the inventory management system for retailers as a customer")
            mail = Mail(from_email, to_email, subject, content)

            # Get a JSON-ready representation of the Mail object
            mail_json = mail.get()

            # Send an HTTP POST request to /mail/send
            response = sg.client.mail.send.post(request_body=mail_json)
            print(response.status_code)
            print(response.headers)
            return redirect(url_for('login'))
        except:
            flash("Error in Insert Operation","danger")



    return render_template('register.html')

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for("index"))

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

    stmt = "SELECT * FROM CUSTOMER"
    sql = ibm_db.prepare(conn,stmt)
    ibm_db.execute(sql)
    dictionary = ibm_db.fetch_assoc(sql)
    print(dictionary)
    a=[]
    while dictionary != False:
        a.append(dictionary)
        print(dictionary)
        dictionary = ibm_db.fetch_assoc(sql)


    return render_template('test.html')

@app.route('/addproduct', methods=['GET','POST'])
def addproduct():
    if request.method=='POST':
        try:
            productname = request.form['productname']
            productid = request.form['productid']
            quantity = int(request.form['quantity'])
            price = int(request.form['price'])
            imageurl = request.form['imageurl']
            insert_sql = "INSERT INTO PRODUCT VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, productid)
            ibm_db.bind_param(prep_stmt, 2, productname)
            ibm_db.bind_param(prep_stmt, 3, quantity)
            ibm_db.bind_param(prep_stmt, 4, price)
            ibm_db.bind_param(prep_stmt, 5,imageurl)
            ibm_db.execute(prep_stmt)
            flash('Product added successfully')

        except:
            flash('An error occured while adding the product')
        finally:
            return redirect(url_for('addproduct'))

    return render_template('addproduct.html')

@app.route('/home')
def index1():
    stmt = "SELECT * FROM PRODUCT"
    sql = ibm_db.prepare(conn,stmt)
    ibm_db.execute(sql)
    dictionary = ibm_db.fetch_assoc(sql)
    print(dictionary)
    a=[]
    while dictionary != False:
        a.append(dictionary)
        # print(dictionary)
        dictionary = ibm_db.fetch_assoc(sql)
```

```python
    return render_template("product.html",product=a)


# @app.route('/cartadd',methods=['POST'])
# def cartadd():
#     try:

#        id= request.form['id']
#        quan = int(request.form['quantity'])
#        print(id,quan)
#        if id and quan:
#         stmt = "SELECT * FROM PRODUCT WHERE PRODUCTID =?"
#         sql = ibm_db.prepare(conn,stmt)
#         ibm_db.bind_param(stmt,1,id)
#         ibm_db.execute(sql)
#         dictionary = ibm_db.fetch_assoc(sql)
#         print(dictionary)
#        cart =list()
#        session["cart"]=[]



#     except:
#         return redirect(url_for("index"))
#     finally:
#         # print(cart)
#         return redirect(url_for('index'))


@app.route('/cart')
def cart():
    stmt = "SELECT * FROM PRODUCT"
    sql = ibm_db.prepare(conn,stmt)
    ibm_db.execute(sql)
    dictionary = ibm_db.fetch_assoc(sql)
    print(dictionary)
    a=[]
    while dictionary != False:
        if dictionary['PRODUCTID'] in [1,2,3,4]:
            a.append(dictionary)
        # print(dictionary)
        dictionary = ibm_db.fetch_assoc(sql)
    print(a)
    return render_template("cart.html",products=a)

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

GitHub link:

https://github.com/IBM-EPBL/IBM-Project-23478-1659883673

Video Demo Link:

https://drive.google.com/file/d/1tnt5waXL_sTfgLvRMWPnYj-jnJ73uSAA/view?usp=sharing