

PROJECT REPORT FORMAT

1. INTRODUCTION

- 1.1 Project Overview
- 1.2 Purpose

2. LITERATURE SURVEY

- 2.1 Existing problem
- 2.2 References
- 2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

- 3.1 Empathy Map Canvas
- 3.2 Ideation & Brainstorming
- 3.3 Proposed Solution
- 3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

- 4.1 Functional requirement
- 4.2 Non-Functional requirements

5. PROJECT DESIGN

- 5.1 Data Flow Diagrams
- 5.2 Solution & Technical Architecture
- 5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

- 6.1 Sprint Planning
- 6.2 Sprint Estimation and Delivery Schedule

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

- 7.1 SendGrid
- 7.2 Database Schema

8. TESTING

9. RESULTS

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

GitHub & Project Demo Link

1.INTRODUCTION

1.1 Project Overview: -

An inventory management system (or inventory system) is the process by which we track your goods throughout your entire supply chain, from purchasing to production to end sales. It governs how we approach inventory management for your business.

1.2 Purpose: -

Any venture that handles stock will need a system to accurately track and control it. Without one, we'll be working on an entirely ad-hoc basis — and you'll quickly run into situations where your business is overstocked or understocked.

Inventory systems tell you the number of components or ingredients you need to create or assemble your final product. Without this information you may end up with excess stock, eroding your bottom line, or with insufficient stock to meet customer demand.

But while you will need an inventory management system, which one you choose is entirely up to you. There are countless different systems you can adopt, ranging from simple approaches to comprehensive solutions.

In our opinion we intend to create an application that is user-friendly for retailers who need to oversee the products inventory and manage stocks accordingly.

2. LITERATURE SURVEY

2.1 Existing Problem:

Businesses are quickly realising that inventory control is absolutely necessary to run an efficient business and

make money in the process. Especially in today's competitive marketplace, business owners simply can't

afford to have money go down the drain.

- High cost of inventory
- Consistent stock outs
- Low rate of inventory turnover
- High amount of obsolete inventory
- High amount of working capital
- High cost of storage
- Spreadsheet data-entry errors
- Lost customers

2.2 Literature Survey: -

1) Hybrid algorithm based on reinforcement learning for smart inventory management

Journal: Journal of Intelligent Manufacturing (2022)

Date: 03 August 2022

Authors: Carlos Cuartas & Jose Aguilar

Abstract:

This article proposes a hybrid algorithm based on reinforcement learning and the inventory management methodology called DDMRP(Demand Driven Material Requirement Planning) to determine the optimal time to buy a certain product, and how much quantity should be requested. For this, the inventory management problem is formulated as a Markov Decision Process where the environment with which the system interacts is designed from the concepts raised in the DDMRP methodology, and through the reinforcement learning

algorithm—specifically, Q-Learning. The optimal policy is determined for making decisions about when and how much to buy. To determine the optimal policy, three approaches are proposed for the reward function: the first one is based on inventory levels; the second is an optimization function based on the distance of the inventory to its optimal level, and the third is a shaping function based on levels and distances to the optimal inventory. The results show that the proposed algorithm has promising results in scenarios with different characteristics, performing adequately in difficult case studies, with a diversity of situations such as scenarios with discontinuous or continuous demand, seasonal and non-seasonal behaviour, and with high demand peaks, among others.

2) Research and Design of the Intelligent Inventory Management System Based on RFID

Published in: 2013 Sixth International Symposium on Computational Intelligence and Design

Date: 28-29 October 2013 **Authors:** Xiaojun Jing; Peng Tang

Abstract:

This paper introduces the characteristics and basic application of RFID technology, analyses the data flow of intelligent inventory system from the perspective of business and function, then puts forward the specific framework programs and function modules of intelligent inventory management system based on IOT RFID technology, focuses on elaborating the design and implementation process of the intelligent inventory system. The system realizes full control and management of all products, faster in/out warehouse and dynamic inventory, utilizes warehouse efficiently and improves the capacity of warehouse by effective combining with the ERP system in enterprise.

2.2 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 & Brainstroming:-

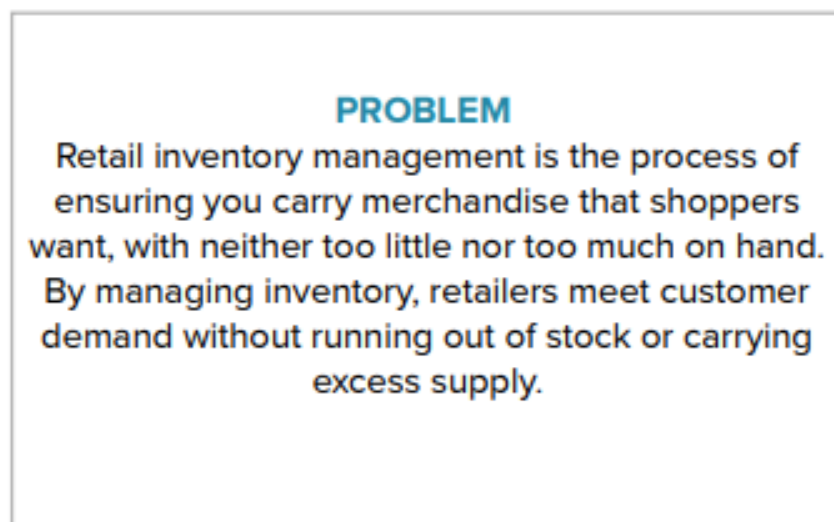


FIG 3.1 PROBLEM DEFINITION

Venkatesha Prabhu P

improve accuracy	improve business planning	reduce cost
limited visibility	decentralized design	lack of system optimization

Sethuram S V

better inventory planning and forecasting	increased efficiency	can manage high demand
saves time	improving supply chain operations	make and hold stocks

Sivaraman M M

better inventory control	supply chain complexity	business should strive to a sweet spot
without ever running out of stock	saves your money and fulfill your customer needs	decr

Dhinesh M N

Set reorder points for each product	Give each variant a dedicated warehouse bin	Use EOQ for optimal order quantities
decreased visibility	Automate as much as possible	saves time and cost

FIG 3.2 Brainstorm

1. improve accuracy
2. improve business planning
3. reduce cost
4. limited visibility
5. decentralized design
6. lack of system optimization
7. better inventory planning and forecasting
8. can manage high demand
9. increased efficiency saves time
10. make and hold stocks
11. business should strive to a sweet spot
12. Set reorder points for each product
13. Automate as much as possible
14. Set reorder points for each product
15. Automate as much as possible

Fig 3.3 Ideas grouped Together.

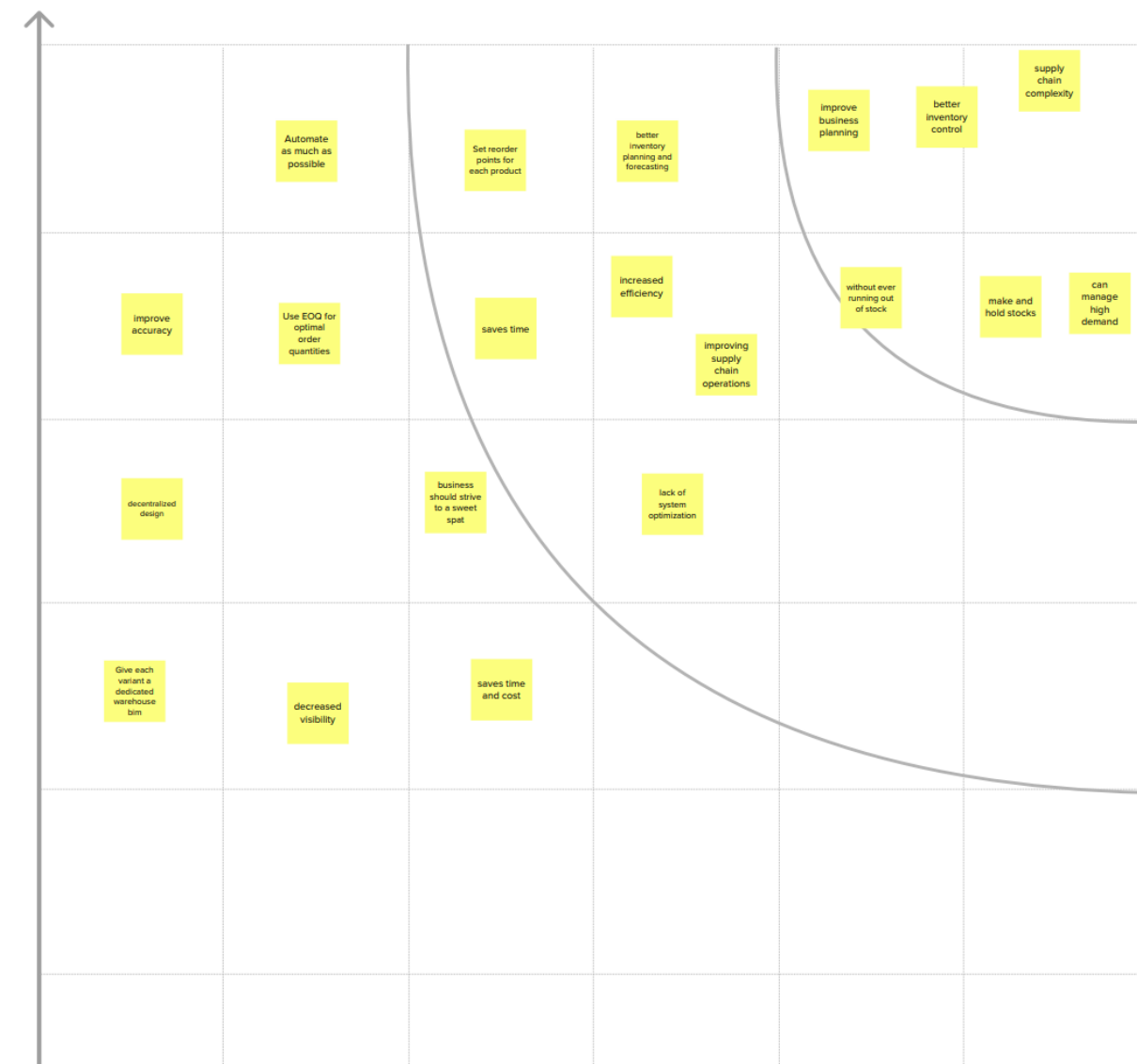


FIG 4.4 PRIORITIZE(CHECKING FEASIBILITY OF IDEAS.)

3.3 - Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Uncertainness in stock availability will affect the business's performance and sales. It will lead to disappointment of clients and thus affects sales and revenues consequently. Also disconnect in syncing of data between availability of stocks and those in inventories might lead to confusion and delayed replacement of stocks in the store. Thus having the products replaced at the right times is essential for the business.
2.	Idea / Solution description	To develop a cloud application that will give complete information on the availability of stocks and send alerts based on the volume of a particular stock that runs out.
3.	Novelty / Uniqueness	Accurate and timely information on the availability of stocks, control over forecasting/projection on volumes of stocks, and improved inventory analysis for the clients.
4.	Social Impact / Customer Satisfaction	The solution will lead to better availability of products as and when needed. This drastically reduces the waiting times of customers and attracts new customers.
5.	Business Model (Revenue Model)	Inventory Management model aids businesses in determining the quantity of products and their future needs. Inventory of the products is tracked from product acquisition phase to sales.
6.	Scalability of the Solution	Inventory Management System is automated for inventory tracking to improve the scalability of the business.

3.4 Problem Solution Fit:-

Project Title: Inventory Management System for Retailers

Project Design Phase-I - Solution Fit Template

Team ID: PNT2022TMD121310

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS Who is your customer? Companies and organisations often use inventory management software applications to properly manage inventory stocks and reduce cost by avoiding manual work for the same task as mentioned. The software application is used to track availability of products in the warehouse and gives alert if it drops down below the set threshold.	6. CUSTOMER CONSTRAINTS CC What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices. Availability of raw materials, workforce/labour, machine's capacity, delivery capacity, inventory space, inventory investment and the total number of orders placed.	5. AVAILABLE SOLUTIONS AS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital retailing Using physical measure to track inventory supplies is an option, but as the application grows a centralized and scalable solution would be a software application based inventory management system.	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS J&P Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one, explore different sides. 1. The major challenge is prevent the situation where the company oversells the product or lose track of the product sales and run into situation of no stocks left in the inventory. 2. By analyzing trends/patterns in the sales can lead to better inventory management.	9. PROBLEM ROOT CAUSE RC What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations. 1. Use of old softwares or traditional methods can lead to eventual increase in complexity of inventory management which also leads to the ill management and productivity in the software. 2. In today's competitive world one must use cutting edge software to keep up with the industry growth.	7. BEHAVIOUR BE What does your customer do to address the problem and get the job done? (X) Directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace) The customer mainly needs to address to cutting of costs and boost efficiency and profitability. These goals play perfectly into an overall strategy of improving customer service. Companies should not lose sight of who keeps them in business. Your customers have certain expectations of you, and it is essential to maintain a level of service that lives up to those expectations. A effective Inventory Management allows you to do that by being flexible and keeping up with seasonal changes in demand and other ups and downs in sales trends.	
Identify strong TR & EM	3. TRIGGERS TR What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news. To help retailers to manage their stocks and order stock based on requirements when it runs out. It tracks products on inventory from purchase till the sale of goods.	10. YOUR SOLUTION SL If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour. After the retailer logins successfully into the system they can update and manage their inventory details and also will be able to add new stocks by providing essential information related to the stock. The user will be able to view details of their current inventory. The system will automatically send an email alert to the retailer if there is no stock (or) very less stock found in their accounts such that they can order new stock.	8.CHANNELS of BEHAVIOUR CH 8.1 ONLINE What kind of actions do customers take online? Extract online channels from #7 Inventory Management System can keep track of important data concerning to related items and give us the option to maintain additional inventory levels that reflect your return rates if required. 8.2 OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. The user will receive constant upgrades through mail even though they are active on the application.	Identify strong TR & EM
	4. EMOTIONS: BEFORE / AFTER EM How do customers feel when they face a problem or a job and afterwards? i.e. lost, insecure -> confident, in control - use it in your communication strategy & design. Having run out of stocks may lead to loss of business for retailers when the product is in high demand. Thus, they can monitor and manage their inventory by themselves easily and quickly.			

4. REQUIREMENT ANALYSIS

4.1. Functional Requirements:-

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 LinkedIN/Google/ any OAuthprovider
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Allocating and updating database	Authentication of the user
FR-4	Creating alert	Through phone number, through email
FR-5	Update inventory status	Update inventory status through inventory dashboard
FR -6	Tracking order status	Tracking order status through, the website of the ordersout for delivery

4.2. *Non- Functional Requirements:-*

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

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	An easy to use interface that doesn't require user instructions, support or documentation.
NFR-2	Security	Security requirements ensure that the software is protected from unauthorized access to the system and its stored data. It considers different levels of authorization and authentication across different user roles.
NFR-3	Reliability	An inventory management system must maintain its adeptness over time while tracking with the uncertainty of the inventory flow.
NFR-4	Performance	The performance of the website must be robust and the updates and alerts must be real-time without any noticeable lags.
NFR-5	Availability	The website must be accessible to the customer at any given point of the time, if the website is down for maintenance an adequate summary of the previous inventory must be displayed or have a cold website backup.
NFR-6	Scalability	Users can scale their online inventory based on their requirements - various branches, franchises.

5.PROJECT DESIGN

5.1 Data Flow Diagrams: -

Data Flow Diagrams:

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.

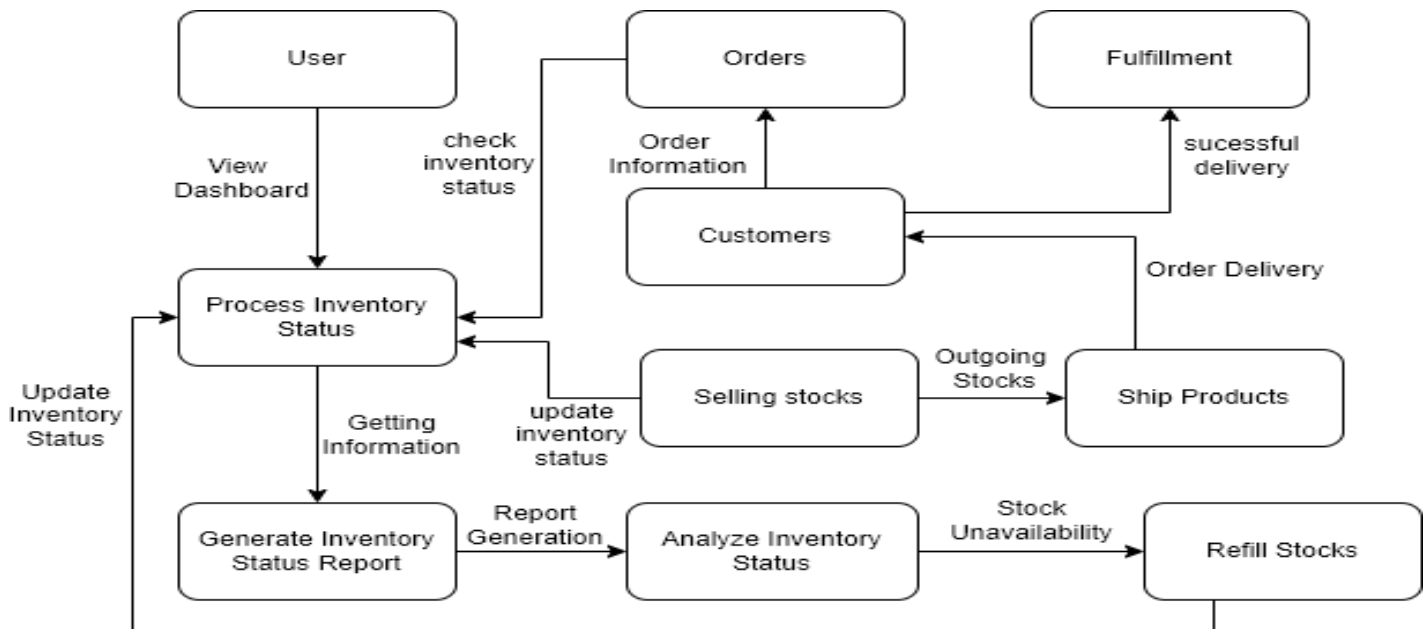
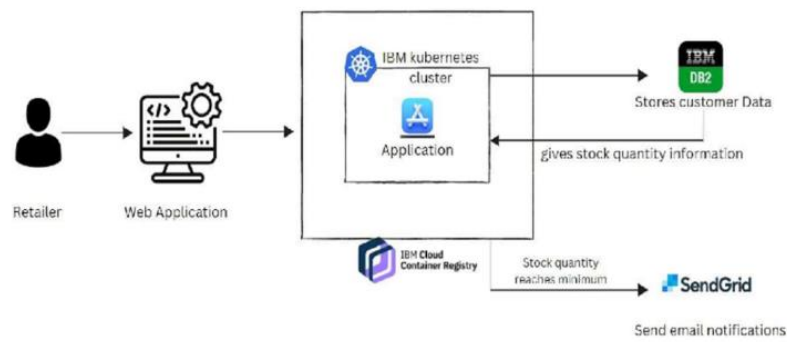


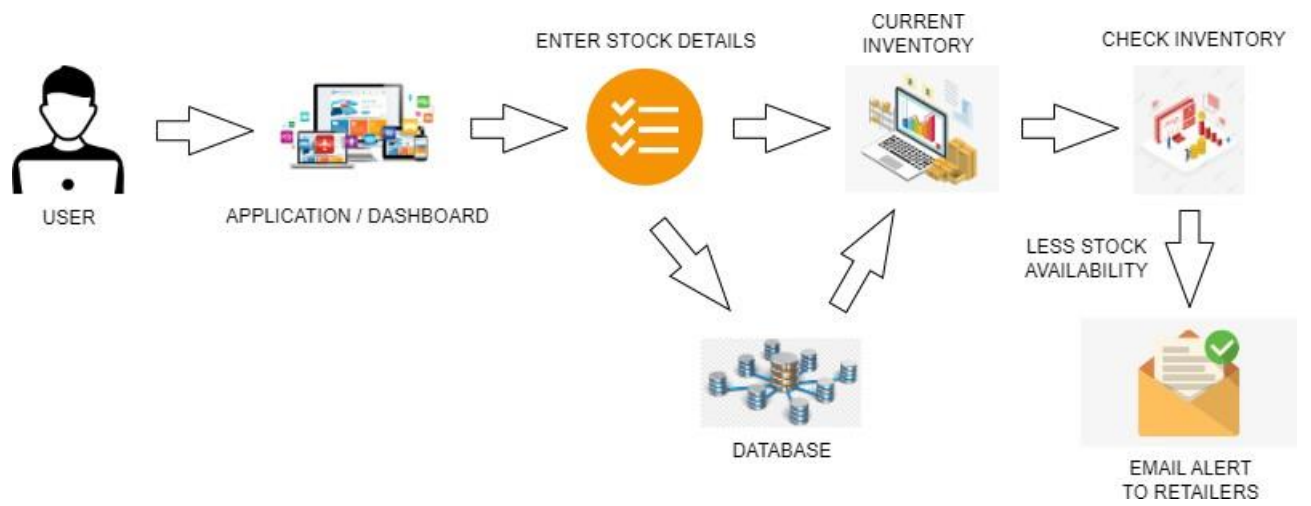
Fig 5.1: Data Flow Diagram Of Inventory Management

5.2 Solution & Technical Architecture: -

Solution Architecture: -



Technical Architecture:-



5.3 User Stories: -

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register and access the dashboard with Gmail login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register and access the dashboard with email and password	High	Sprint-1
	Dashboard	USN-6	As a user, I can view the stock availability status	I can view the stock availability status	High	Sprint-2
		USN-7	As a user, I can view the orders status	I can view the order status	Medium	Sprint-3
	Alerts	USN-8	As a user, I can view the shipping tracking status	I can view the shipping tracking status	Medium	Sprint-4
		USN-9	As a user, I should receive alerts on stock availability if it drops belows the set threshold	I should receive prompt alerts on stock availability if it drops below the set threshold	Medium	Sprint-4
Customer (Web user)	Registration	USN-10	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
		USN-11	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-12	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
		USN-13	As a user, I can register for the application through Gmail	I can register and access the dashboard with Gmail login	Medium	Sprint-1
	Login	USN-14	As a user, I can log into the application by entering email & password	I can register and access the dashboard with email and password	High	Sprint-1
	Dashboard	USN-15	As a user, I can view the stock availability status	I can view the stock availability status	High	Sprint-2
		USN-16	As a user, I can view the orders status	I can view the order status	Medium	Sprint-3
	Alerts	USN-17	As a user, I can view the shipping tracking status	I can view the shipping tracking status	Medium	Sprint-4
		USN-18	As a user, I should receive alerts on stock availability if it drops belows the set threshold	I should receive prompt alerts on stock availability if it drops below the set threshold	Medium	Sprint-4
Customer Care Executive		USN-19	As a customer care executive, I can view the complaints on chat box	I can view the complaints on chat box	Medium	Sprint-4
		USN-20	As a customer, I should be able solve and reply for the customers queries	I can reply to customer queries in the chat thread	Low	Sprint-4
		USN-21	As a customer, I can close the complaint after assisting	I can close the complaint after assisting	Low	Sprint-4
Administrator		USN-22	As a Administrator, I would take care of registrations and maintenance of accounts,	I can take care of registrations and maintenance of accounts	High	Sprint-3
		USN-23	As a Administrator, I Would resolve issues on Access	I can resolve issues in Access	High	Sprint-2
		USN-24	As a Administrator, I Would resolve issues in supply chain /Syncing of Orders	I can resolve issues in supply chain/ Syncing of Orders	High	Sprint-4

6. PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning:

Sprints are the backbone of any good Agile development team. And the better prepared you are before a sprint, the more likely you are to hit your goals. Spring planning helps to refocus attention, minimize surprises, and (hopefully) guarantee better code gets shipped. The main event during agile methodology is the sprint, the stage where ideas turn into innovation and valuable products come to life. On one hand, agile sprints can be highly effective and collaborative. At the same time, they can be chaotic and inefficient if they lack proper planning and guidance. And for this reason, making a sprint schedule is one of the most important things you can do to ensure that your efforts are successful.

We categorized the sprint as 4 phases for creating the application

- Sprint 1 is about creating the login page and the register page.
- Sprint 2 is about sending the confirmation mail to the users during registration.
- Sprint 3 is about as a user, can log into application by entering email and password.
- Sprint 4 is about as user, can register and make request for plasma donation via portal.

Table-1 : Components & Technologies:

S.No	Component	Technology
1.	User Interface	HTML, CSS, JavaScript
2.	Application Logic-1	Python
3.	Application Logic-2	IBM Watson Assistant
4.	Database	MySQL
5.	Cloud Database	IBM DB2, IBM Cloudant etc.
6.	File Storage	IBM Block Storage or Other StorageService or Local Filesystem
7.	External API	SendGrid

8.	Infrastructure (Server / Cloud)	Local, Cloud Foundry, Kubernetes, etc.
----	---------------------------------	--

Table-2: Application Characteristics:

S.No	Characteristics	Technology
1.	Open-Source Frameworks	Flask
2.	Security Implementations	Bcrypt, Encryptions, IAM Controls, etc.
3.	Scalable Architecture	Kubernetes
4.	Availability	Docker CLI
5.	Performance	Browser caching

6.2 Sprint Estimation and Delivery Schedule:

A sprint estimation shows how much effort a series of tasks require. It's based on assumptions, requirements, and dependencies of a project.

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Venkatesha Prabhu
Sprint-1	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	3	High	Sivaraman
Sprint-2	Registration	USN-3	As a user, I can register for the application through Facebook	8	Low	Sethuram
Sprint-1	Registration	USN-4	As a user, I can register for the application through Gmail	3	Medium	Dhinesh
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	5	High	Venkatesha Prabhu
Sprint-2	Dashboard	USN-6	As a user, I can view the stock availability status	3	High	Sivaraman
Sprint-3	Dashboard	USN-7	As a user, I can view the orders status	3	Medium	Sethuram
Sprint-4	Dashboard	USN-8	As a user, I can view the shipping tracking status	2	Medium	Dhinesh
Sprint-4	Alerts	USN-9	As a user, I should receive alerts on stock availability if it drops belows the set threshold	2	Medium	Venkatesha Prabhu
Sprint-1	Registration	USN-10	As a user, I can register for the application by entering my email, password, and confirming my password.	3	High	Sivaraman

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-11	As a user, I will receive confirmation email once I have registered for the application	3	High	Sethuram
Sprint-2	Registration	USN-12	As a user, I can register for the application through Facebook	3	Low	Dhinesh
Sprint-1	Registration	USN-13	As a user, I can register for the application through Gmail	3	Medium	Dhinesh
Sprint-1	Login	USN-14	As a user, I can log into the application by entering email & password	3	High	Sethuram
Sprint-2	Dashboard	USN-15	As a user, I can view the stock availability status	4	High	Sivaraman
Sprint-3	Dashboard	USN-16	As a user, I can view the orders status	8	Medium	Venkatesha Prabhu
Sprint-4	Dashboard	USN-17	As a user, I can view the shipping tracking status	4	Medium	Dhinesh
Sprint-4	Alerts	USN-18	As a user, I should receive alerts on stock availability if it drops belows the set threshold	3	Medium	Sethuram
Sprint-4	Chat box	USN-19	As a customer care executive, I can view the complaints on chat box	4	Medium	Sivaraman
Sprint-4	Chat box	USN-20	As a customer, I should be able solve and reply for the customers queries	3	Low	Venkatesha Prabhu
Sprint-4	Chat box	USN-21	As a customer, I can close the complaint after assisting	3	Low	Dhinesh
Sprint-3	Admin	USN-22	As a Administrator, I would take care of registrations and maintenance of accounts,	8	High	Sethuram
Sprint-2	Admin	USN-23	As a Administrator, I Would resolve issues on Access	3	High	Sivaraman
Sprint-4	Admin	USN-24	As a Administrator, I Would resolve issues in supply chain /Syncing of Orders	4	High	Venkatesha Prabhu

Project Tracker, Velocity & Burndown Chart:

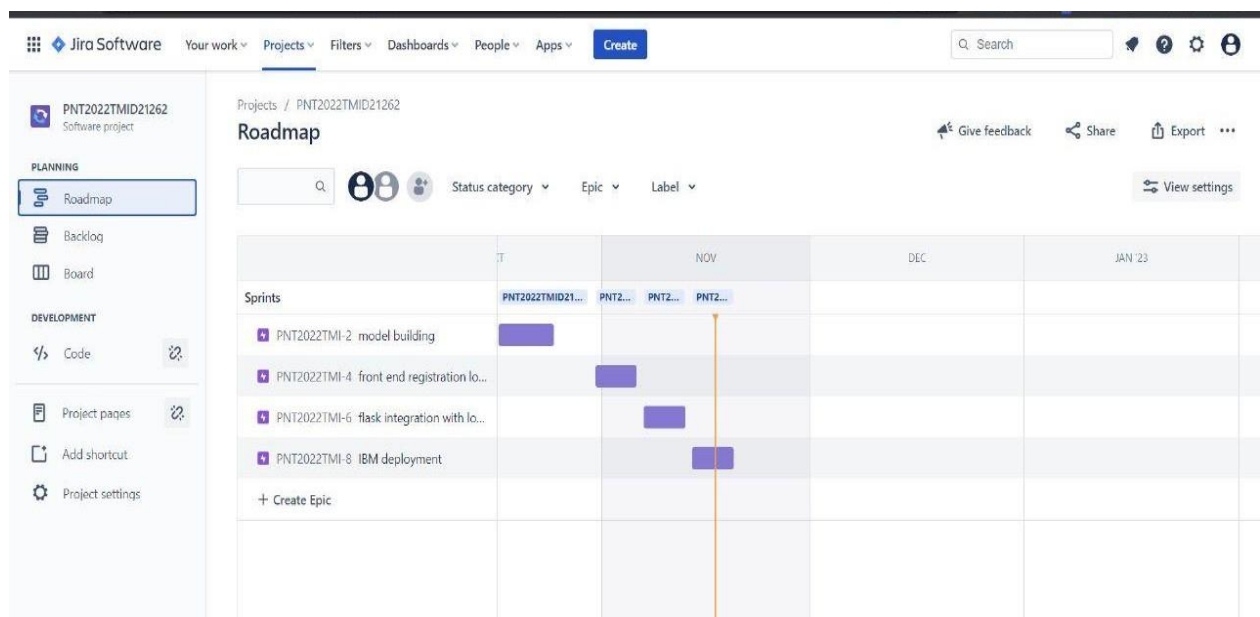
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	22	6 Days	24 Oct 2022	29 Oct 2022	22	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	19	6 Days	07 Nov 2022	12 Nov 2022	19	12 Nov 2022
Sprint-4	22	6 Days	14 Nov 2022	19 Nov 2022	22	19 Nov 2022

Velocity:

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

$$AV = \frac{\text{sprint duration}}{\text{velocity}} = \frac{20}{10} = 2$$

6.3 Reports from JIRA:



Jira Software

Your work

Projects

Filters

Dashboards

People

Apps

Create

Q Search

PNT2022TMID21262

Software project

PLANNING

Roadmap

Backlog

Board

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / PNT2022TMID21262

Backlog

Q

Epic

Insights

PNT2022TMID21262 Sprint 2

31 Oct – 5 Nov (1 issue)

000

Complete sprint

PNT2022TMI-3

PNT2022TMID21262_Front_end_Registration

DONE

+ Create issue

PNT2022TMID21262 Sprint 3

7 Nov – 12 Nov (1 issue)

000

Complete sprint

PNT2022TMI-5

PNT2022TMID21262_Flask_Integration_Local

DONE

+ Create issue

PNT2022TMID21262 Sprint 4

14 Nov – 19 Nov (1 issue)

000

Complete sprint

PNT2022TMI-7

PNT2022TMID21262_IBM_Deployment

IN PROGRESS

Quickstart

X

Jira Software

Your work

Projects

Filters

Dashboards

People

Apps

Create

Q Search

PNT2022TMID21262

Software project

PLANNING

Roadmap

Backlog

Board

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Does your team need more from Jira? Get a free trial of our Standard plan.

Projects / PNT2022TMID21262

All sprints

Q

Epic

Sprint

Complete sprint

GROUP BY: None

Insights

TO DO

IN PROGRESS 1 ISSUE

PNT2022TMID21262_IBM_Deployment

PNT2022TMI-7

DONE 2 ISSUES

PNT2022TMID21262_Front_end_Registration

PNT2022TMI-3

PNT2022TMID21262_Flask_Integration_Local

PNT2022TMI-5

Brave

Quickstart

X

7. CODING & SOLUTIONING

7.1 Features:

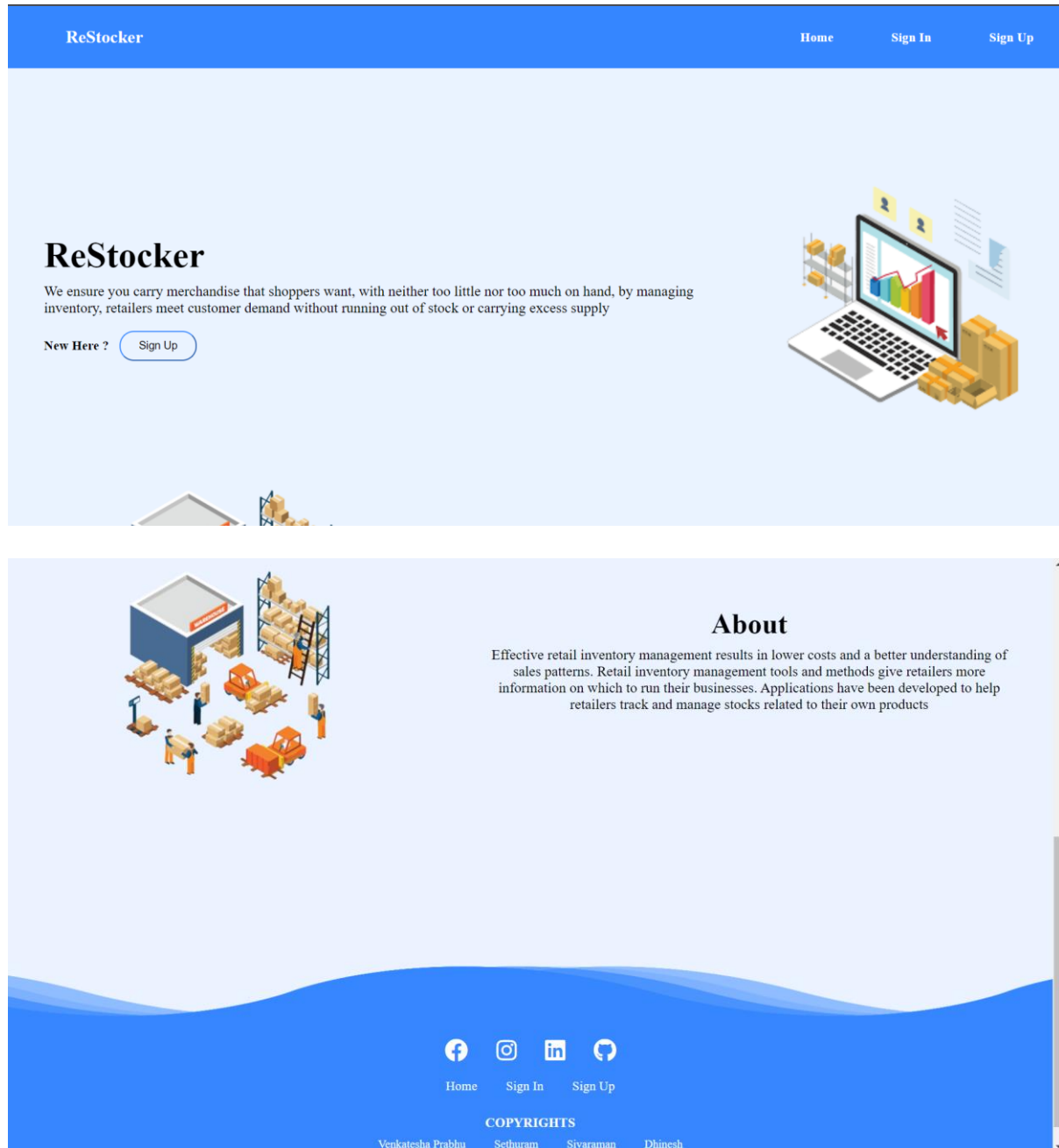


FIG 7.2 Home Page detailing our Product as well as sign up feature.

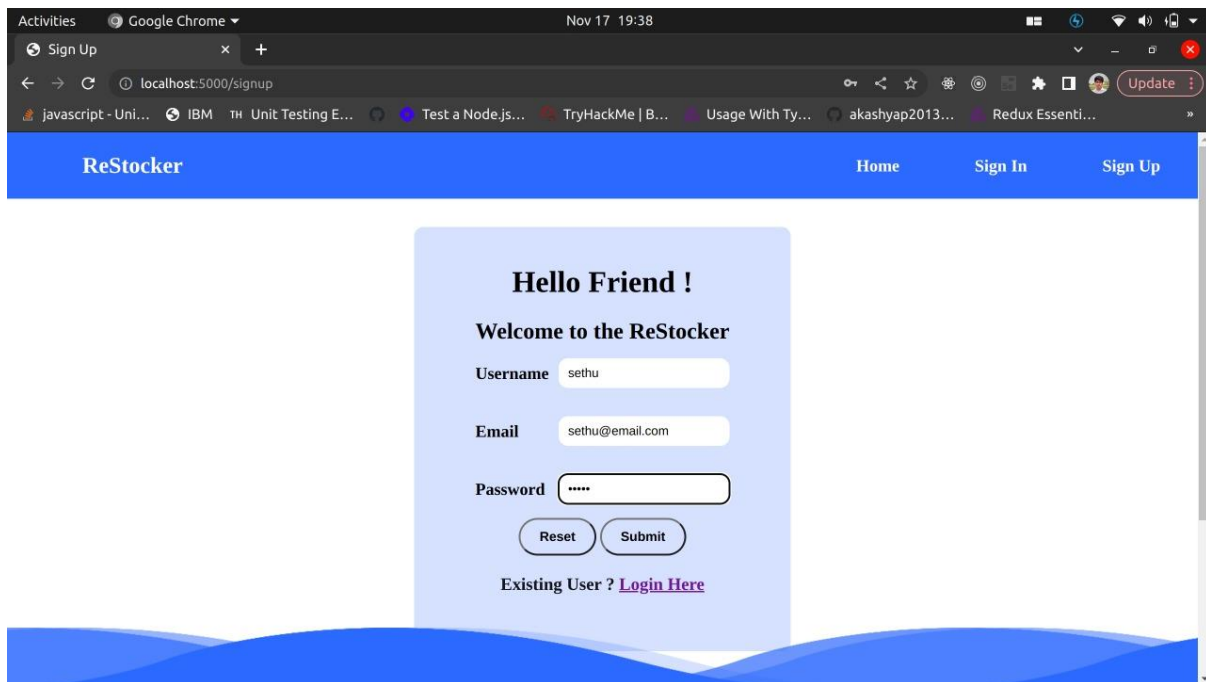
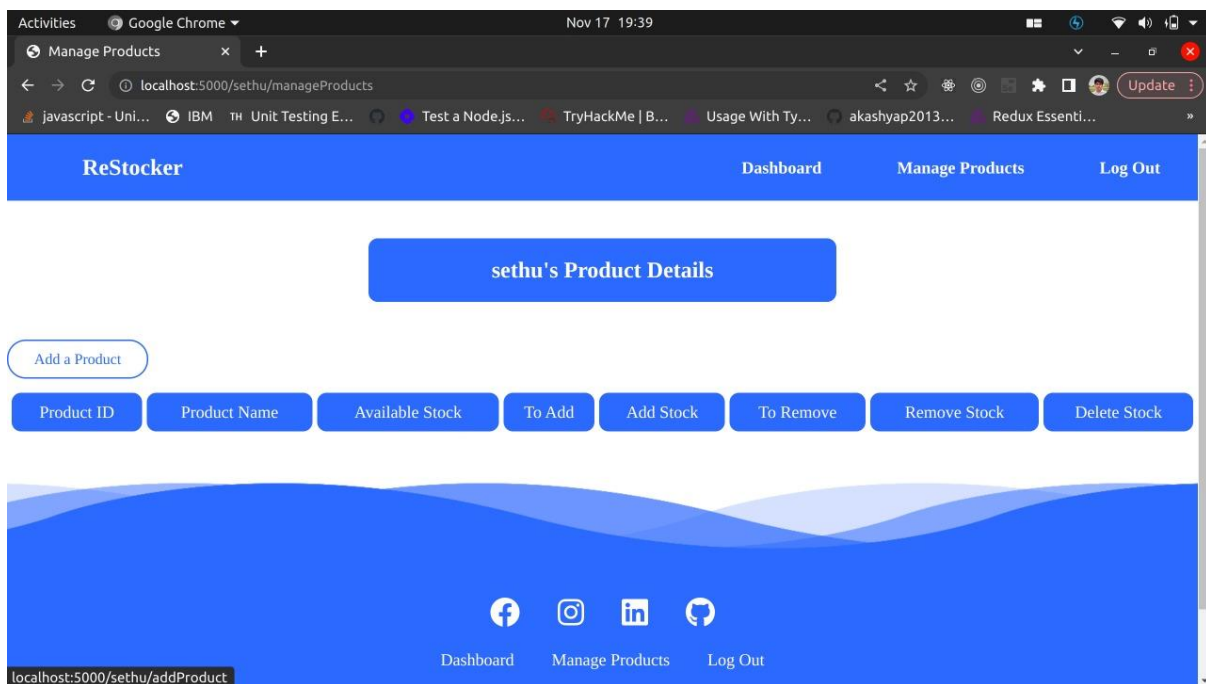


FIG 7.3 Registration Page

STEPS:

Step 1:



Product Details page after logging in

Step 2:

Activities Google Chrome Nov 17 19:39

Add a Product

localhost:5000/sethu/addProduct

ReStocker Dashboard Manage Products Log Out

Add a Product

sethu

Product ID

Product Name

Stock Amount

Price/Unit (in INR)

Activities Google Chrome Nov 17 19:40

Add a Product

localhost:5000/sethu/addProduct

ReStocker Dashboard Manage Products Log Out

Add a Product

sethu

Product ID

Product Name

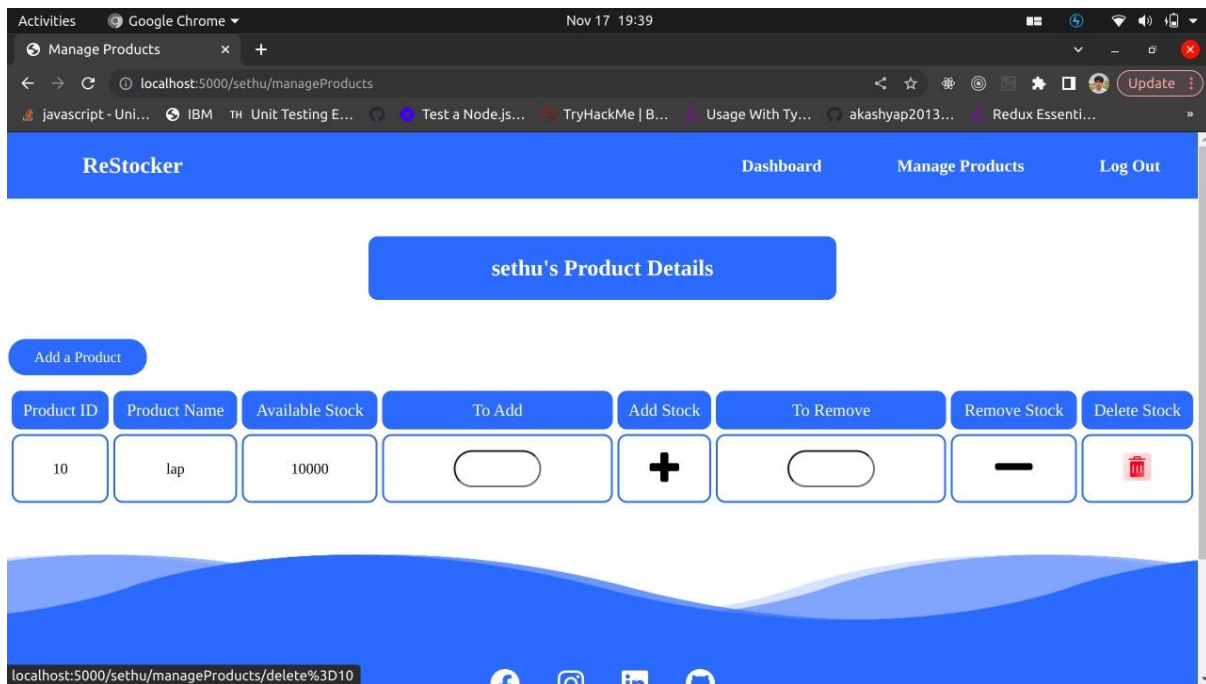
Stock Amount

Price/Unit (in INR)

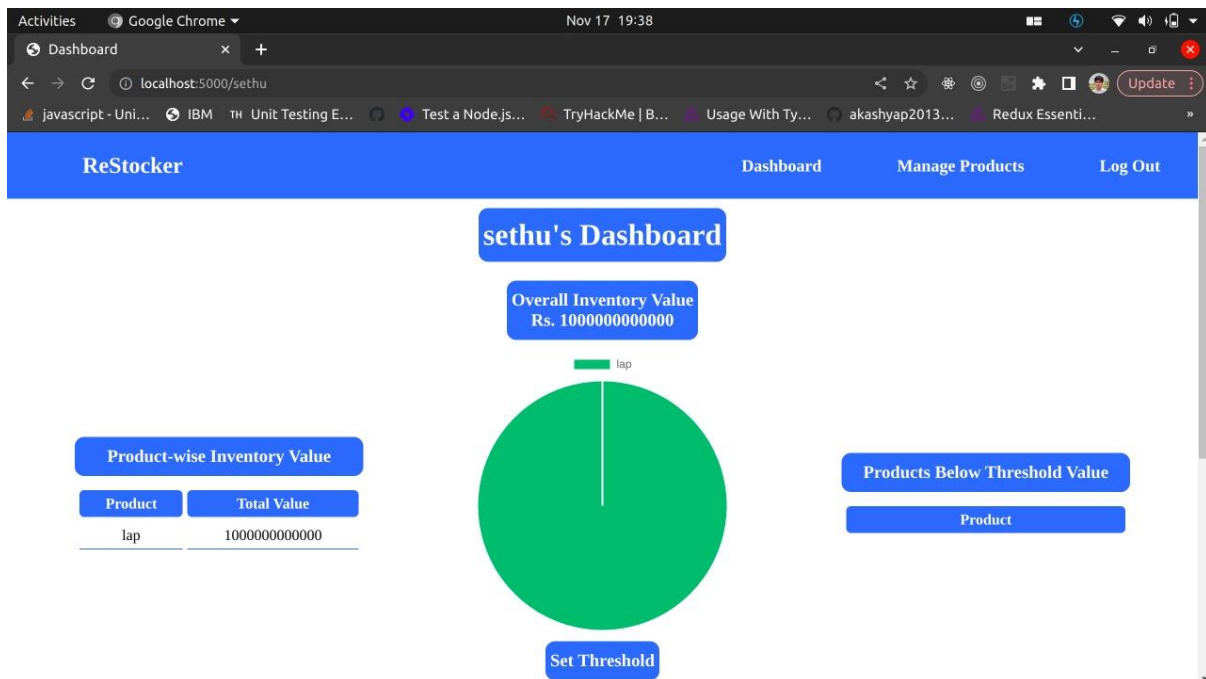
Waiting for localhost...

Add Products

Step 3:



The entered details are displayed.



Details displayed in Dashboard.

Step 4:

ReStocker Dashboard Manage Products Log Out

sethu's Product Details

Add a Product

Product ID	Product Name	Available Stock	To Add	Add Stock	To Remove	Remove Stock	Delete Stock
15	mobile phone	115	<input type="text"/>	+	<input type="text"/>	-	
20	laptop	500	<input type="text"/>	+	<input type="text"/>	-	

Various products are added

ReStocker Dashboard Manage Products Log Out

sethu's Dashboard

Overall Inventory Value
Rs. 26150000

mobile phone laptop

Product-wise Inventory Value

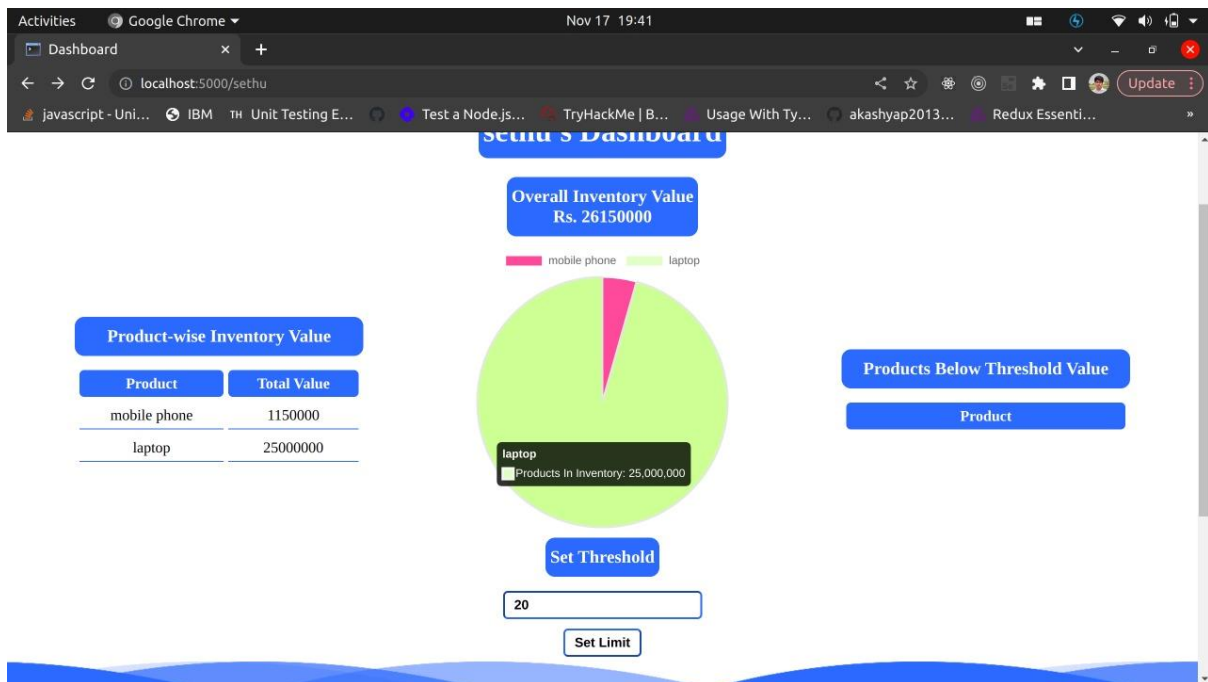
Product	Total Value
mobile phone	1150000
laptop	25000000

Products Below Threshold Value

Product

Set Threshold

Viewed in Dashboard



Threshold values can be set .

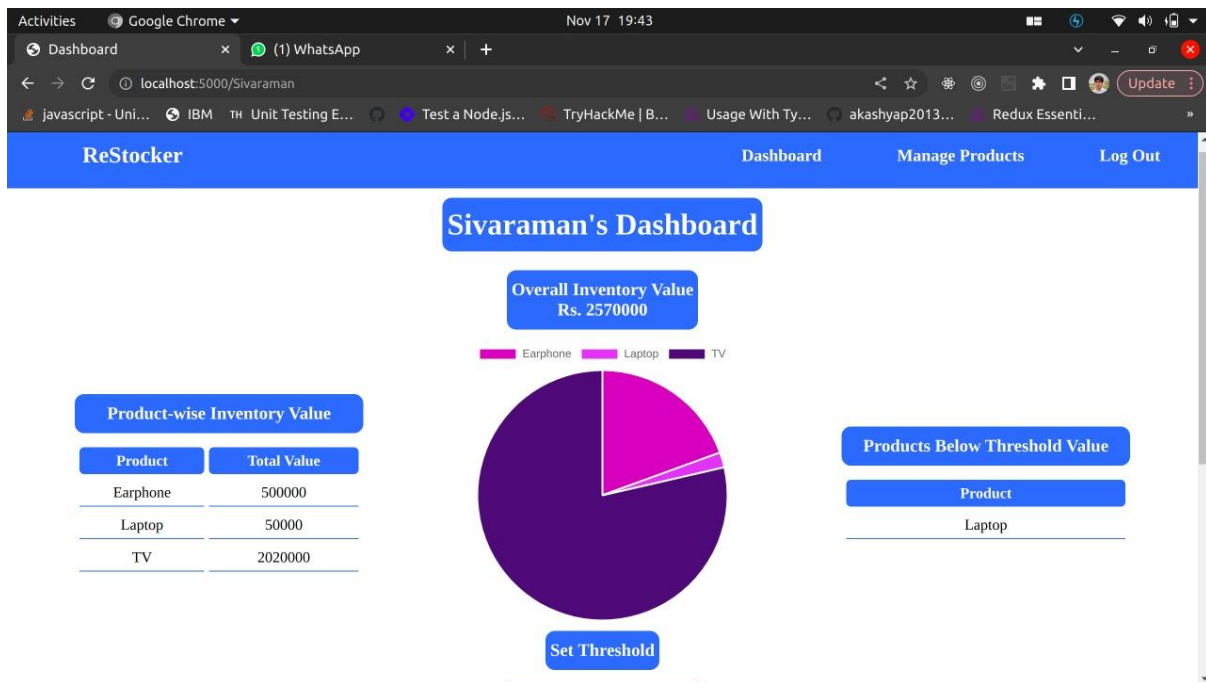
In summary,

Sivaraman's Product Details

Add a Product

Product ID	Product Name	Available Stock	To Add	Add Stock	To Remove	Remove Stock	Delete Stock
1	Earphone	1000	<input type="text"/>	+	<input type="text"/>	-	
2	Laptop	1	<input type="text"/>	+	<input type="text"/>	-	
3	TV	101	<input type="text"/>	+	<input type="text"/>	-	

Products are added



They can be viewed in Dashboard

Deployment in Kubernetes:

```
Activities Terminal Nov 17 21:08
sethuram@sethu: ~
sethuram@sethu: ~
sethuram@sethu:~$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
restocker 1/1     Running   0           3m31s
sethuram@sethu:~$ kubectl logs restocker
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.30.110.75:5000
Press CTRL+C to quit
sethuram@sethu:~$ kubectl logs restocker
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.30.110.75:5000
Press CTRL+C to quit
sethuram@sethu:~$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
restocker 1/1     Running   0           8m3s
sethuram@sethu:~$ ^C
sethuram@sethu:~$
```

Activities Google Chrome Nov 17 21:08

eu-de.containers.cloud.ibm.com/kubeproxy/clusters/cdr35cuf0gl0u9rpd0/service/#/workloads?namespace=default


kubernetes default Search

Workloads

- Workloads
- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Service

- Ingresses
- Ingress Classes
- Services



Running: 1

Pods

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
restocker	Show all	Show all	10.144.51.86	Running	0	1.00m	29.41Mi	9 minutes ago

Activities Google Chrome Nov 17 21:08

eu-de.containers.cloud.ibm.com/kubeproxy/clusters/cdr35cuf0gl0u9rpd0/service/#/pod?namespace=default

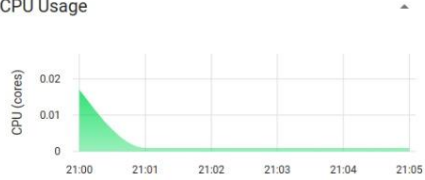
kubernetes default Search

Workloads > Pods

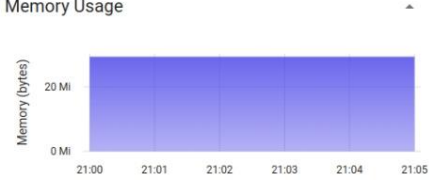
- Workloads
- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers
- Stateful Sets

Service

- Ingresses
- Ingress Classes
- Services



CPU Usage



Memory Usage

Pods

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
restocker	Show all	Show all	10.144.51.86	Running	0	1.00m	29.41Mi	9 minutes ago

7.2 Database Schema

The screenshot shows the IBM Db2 on Cloud console interface. The top navigation bar includes tabs for Load Data, Load History, Tables, Views, Indexes, Aliases, MQTs, Sequences, and Application objects. The main content area is divided into two panels. The left panel, titled 'Schemas', shows a table with columns 'Name', 'Type', and 'Tables'. It lists one schema named 'GLR36049' of type 'User' with 4 tables. The right panel, titled 'Tables', shows a table with columns 'Name', 'Schema', and 'Properties'. It lists four tables: 'PRODUCTS', 'THRESHOLD_VALUE', 'USERPRODUCTS', and 'USERS', all belonging to the 'GLR36049' schema. A 'New table' button is visible in the top right of the Tables panel. The bottom status bar shows the time as 7:25 PM on 11/18/2022.

This screenshot shows the 'Table definition' panel for the 'PRODUCTS' table. The left panel lists the tables, and the right panel displays the table's structure. The table has two columns: 'PRODUCTID' and 'PRODUCTNAME', both of type 'VARCHAR' with a length of 30 and are nullable. The table is approximately 3 rows (32.0 KB) and was updated on 2022-11-18 03:22:40. A 'View data' button is located at the bottom of the table definition panel. The bottom status bar shows the time as 7:25 PM on 11/18/2022.

This screenshot shows the 'Table definition' panel for the 'THRESHOLD_VALUE' table. The left panel lists the tables, and the right panel displays the table's structure. The table has two columns: 'EMAIL' of type 'VARCHAR' with a length of 30, and 'TH_VALUE' of type 'DOUBLE'. Both columns are nullable. The table is approximately 1 row (32.0 KB) and was updated on 2022-11-18 07:17:57. A 'View data' button is located at the bottom of the table definition panel. The bottom status bar shows the time as 7:26 PM on 11/18/2022.

mycluster mycluster Manage F Service D IBM D x IBM-EPBL Meet Setting up IBM-Proje Integrat IBM DESCRIBE +

bpe61bfd0365e9u4psdglite.db2.cloud.ibm.com/cn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3A...

WhatsApp Web Web Development CyberSecurity IBM

IBM Db2 on Cloud

Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

Find schemas or tables Refresh

Tables New table +

Name	Schema	Properties
PRODUCTS	GLR36049	...
THRESHOLD_VALUE	GLR36049	...
USERPRODUCTS	GLR36049	...
USERS	GLR36049	...

Total: 4, selected: 0

Table definition

USERPRODUCTS
Approximate 3 rows (32.0 KB)
Updated on 2022-11-18 07:18:11

Name	Data type	Nullable	Length	Scale
PRODUCTID	VARCHAR	N	30	0
USERNAME	VARCHAR	N	30	0
AVAILABLESTOCK	INTEGER	N		0
UNITPRICE	INTEGER	N		0

View data

Type here to search 7:26 PM 11/18/2022

mycluster mycluster Manage F Service D IBM D x IBM-EPBL Meet Setting up IBM-Proje Integrat IBM DESCRIBE +

bpe61bfd0365e9u4psdglite.db2.cloud.ibm.com/cn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3A...

WhatsApp Web Web Development CyberSecurity IBM

IBM Db2 on Cloud

Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

Find schemas or tables Refresh

Tables New table +

Name	Schema	Properties
PRODUCTS	GLR36049	...
THRESHOLD_VALUE	GLR36049	...
USERPRODUCTS	GLR36049	...
USERS	GLR36049	...

Total: 4, selected: 0

Table definition

USERS
Approximate 1 rows (32.0 KB)
Updated on 2022-11-18 03:20:42

Name	Data type	Nullable	Length	Scale
EMAIL	VARCHAR	N	30	0
USERNAME	VARCHAR	N	30	0
PASSWORD	VARCHAR	N	30	0

View data

Type here to search 7:26 PM 11/18/2022

CHAPTER 8 - TESTING :

After finishing the development , we tested using Selinium IDE :

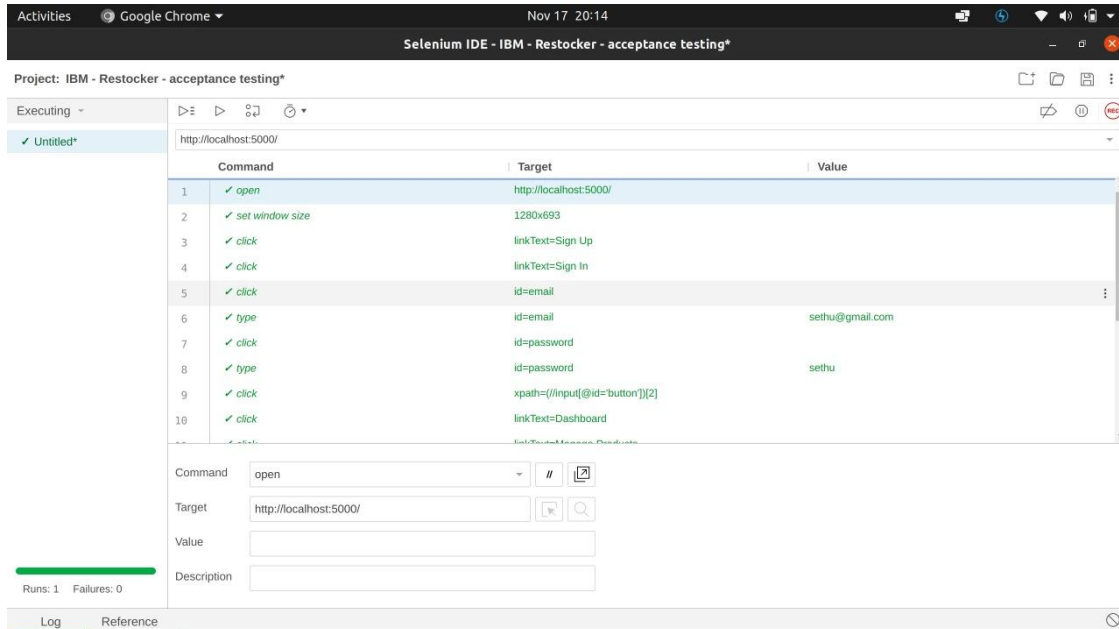


Fig 8.1

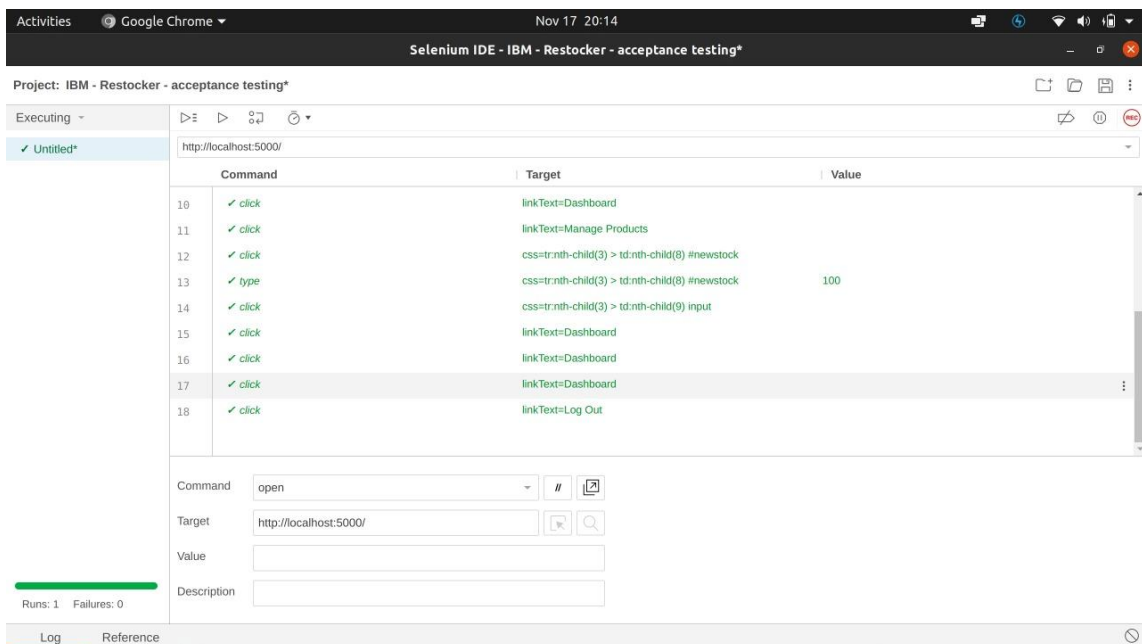


Fig 8.2

Test case ID	Feature Type	Component	Test Scenario	Steps to Execute
SignUpPage_TC_001	Functional	Sign Up page	Verify the user is able to see the Sign up page when the user clicks the sign up button in navigation bar	1. Enter the url and go 2. Click the sign up link in the navigation bar. 3. Verify the sign up page is visible or not.
SignUpPage_TC_002	UI	Sign Up page	Verify the UI elements in the Sign up page	1. Enter the url and go 2. Click the sign up link in the navigation bar. 3. Verify the below mentioned ui elements: a. name text box b. email text box. c. password text box. d. repeat password text box. e. sign up button f. role type radio button
SignUpPage_TC_003	Functional	Sign Up page	Verify the user is able to register into the application by providing valid details	1. Enter the url and go 2. Click the sign up link in the navigation bar. 3. Enter valid details in the text boxes. 4. Verify the confirmation message.
SignInPage_TC_001	Functional	Sign In page	Verify the user is able to see the sign in page when the user clicks the sign in button in navigation bar	1. Enter the url and go 2. Click the sign in link in the navigation bar. 3. Verify the sign in page is visible or not.
SignInPage_TC_002	UI	Sign In page	Verify the UI elements in the Sign in page	1. Enter the url and go 2. Click the sign in link in the navigation bar. 3. Verify the below mentioned ui elements: a. email text box. b. password text box. c. sign in button
SignInPage_TC_003	Functional	Sign In page	Verify the user is able to login into the application by providing valid details	1. Enter the url and go 2. Click the sign in link in the navigation bar. 3. Enter valid details in the text boxes. 4. Verify the user is able to login.

DashboardPage_TC_001	Functional	Dashboard	Verify whether the user is able to see the list of products stored in the warehouse	1. Enter the url and go 2. Verify whether products are visible or not.
DashboardPage_TC_002	UI	Dashboard	Verify the UI elements in the dashboard page	. Enter the url and go 2. Verify the below mentioned ui elements: a. A navbar b. add products button c. logout button
				3. Click the update button. 4. Verify whether the user information is updated successfully.
AddProductForm_TC_001	Functional	Add Product page	Verify the user is able to add a product to the warehouse	1. Enter the url and go 2. Click the request link near the warehouse name. 3. Enter valid details in the text boxes. 4. Click the add button. 5. Verify whether the product is added successfully.
Notification_TC_001	Functional	Dashboard	Verify whether the user gets email notification when the product count reaches threshold	1. Enter the url and go 2. Go to the dashboard. 3. Remove products so that the product count reaches below threshold level.
Logout_TC_001	Functional	Dashboard	Verify the user is able to logout	1. Enter the url and go 2. Click the logout button

9.RESULTS

9.1 *Registration Module*

9.1.1 Sign Up

New users can register themselves using the sign up button by entering their name and email ID or donor can create an account to use in the blood/plasmadonor application and create a password for account verification and create an identity.

9.1.2 Sign In

Already Registered Users can log-in and proceed further.

9.2 *Products Module*

- Add New Products

Users can enter manually various different Products, their ID and stock.

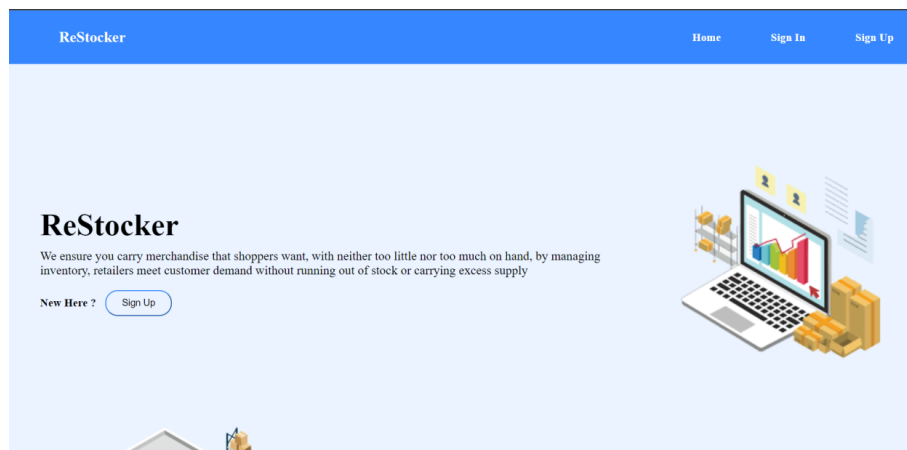
- Display all the Products

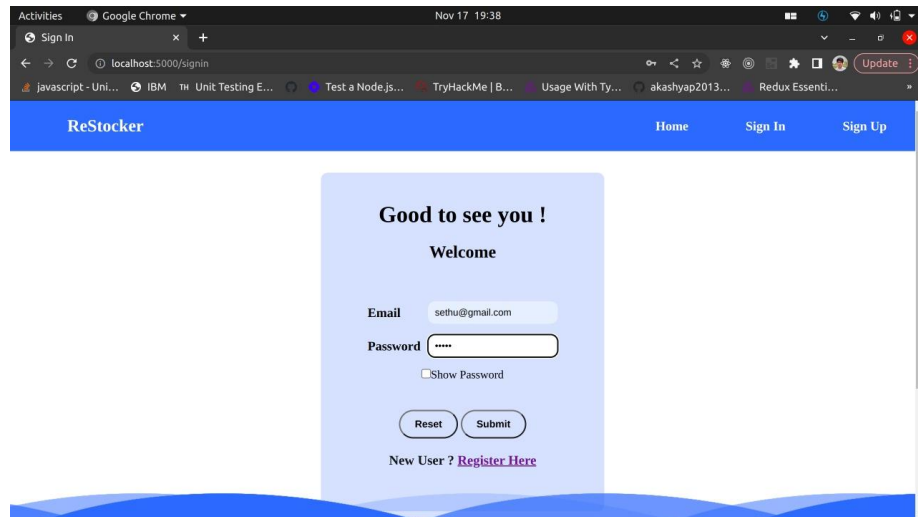
User can view various product categories and the number of stocks in each of them respectively.

- Dashboard

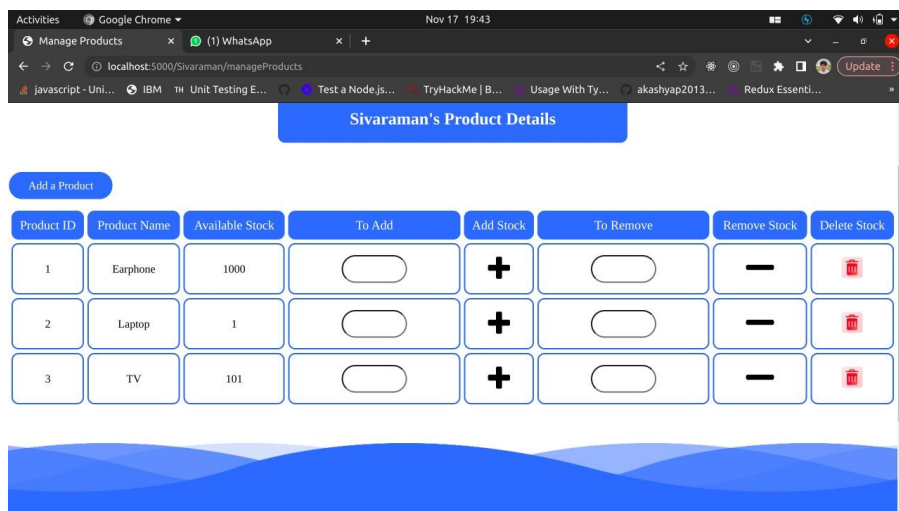
Dashboard to provide infographic detail of all the products.

9.3 *Screen Layouts*

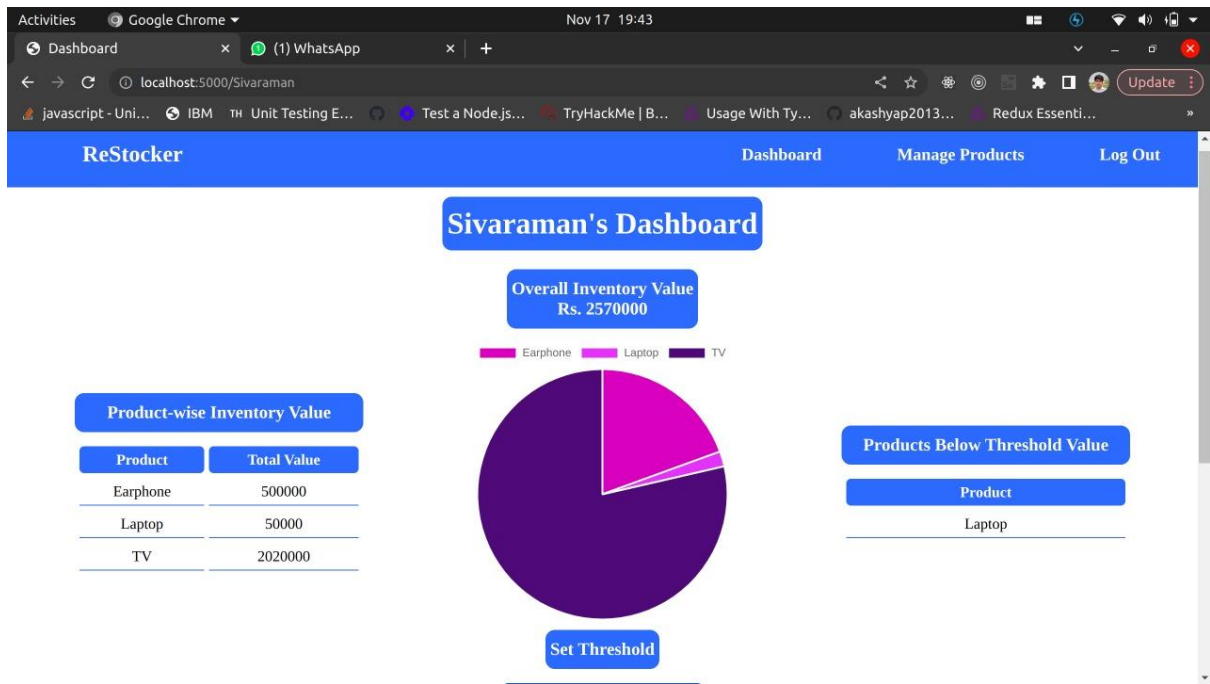




Login page



Products page



Dashboard page

After signing up and coming to products page, Add the products by entering the product name, ID and number of stocks and we can view the dashboard.

10 ADVANTAGES & DISADVANTAGES

Advantages

- *Speed*

This website is fast and offers great accuracy as compared to manual registered keeping.

- *Maintenance*

Less maintenance is required

- *User Friendly*

It is very easy to use and understand. It is easily workable and accessible for everyone.

- *Fast Results*

It would help you to provide plasma donors easily depending upon the availability of it.

Disadvantages

- *Internet*

It would require an internet connection for the working of the website.

- *Auto- Verification*

It cannot automatically verify the genuine users.

11 CONCLUSION

Although the government is carrying out Covid vaccination campaigns on a large scale, the number of vaccines produced is not enough for all the population to get vaccinated at present. And with the corona positive cases rising every day, saving lives has become the prime matter of concern. As per the data provided by WHO more than 3 million people have died due to the coronavirus. However, apart from vaccination, there is another scientific method by which a covid infected person can be treated and the death risk can be reduced. This plasma therapy is an experimental approach to treat corona-positive patients and help them recover. This plasma therapy is considered to be safe & promising. A person who has recovered from Covid can donate his/her plasma to a person who is infected with the coronavirus.

This system proposed here aims at connecting the donors & the patients by an online application. By using this application, the users can either raise a request for plasma donation or requirement. Both parties can Accept or Reject the request. User has to Upload a Covid Negative report to be able to Donate Plasma. This system is used if anyone needs a Plasma Donor Blood and Plasma donation is a kind of citizen's social responsibility in which an individual can willingly donate blood/plasma via our app. This Application has been created with the concept and has sought to make sure that the donor gives blood/plasma to community. This model is made user friendly so anybody can view and maintain his/her account. This application will break the chain of business through blood/plasma and help the poor to find donor at free of cost. This project will help new blood/plasma banks improve their services and progress from traditional to user-friendly frameworks.

12 FUTURE SCOPE

Plasma Application can be developed to further improve user accessibility via integrating this application with various social networks application program interfaces (APIs). Consequently, users can login and sign up using various social networks. This would increase number of donors and enhances the process of blood donation.

User interface (UI) can be improved in future to accommodate global audience by supporting different languages across countries. Data scraping can be done from different social networks and can be shown in the Blood/Plasma Request Feeds. Appointments can be synchronized with Google and Outlook calendars for the ease of users.

Donor and Beneficiary Stories feature aims to create a sense of belonging to the community. Donors will be able to view and share personal experiences about their donation; Beneficiaries can share their experiences of receiving blood transfusion which contributed to their improved health and lives.

Live Check-in Process feature aims to provide a better experience with regards to the waiting time when the user is in the process of donation. We hypothesise that a more efficient experience will help the user look forward to his blood/plasma donation appointments.

13 APPENDIX

- **GitHub Link :**
<https://github.com/IBM-EPBL/IBM-Project-25616-1659968559>
- **Website :**
<http://169.51.204.79:30105/>
- **Project Demo Link :**
<https://drive.google.com/file/d/1SDRaPakIWJnCyK4LRnJ3iqIDIBjhod5H/view?usp=sharing>

SOURCE CODE:

addProduct.html:

```
{% extends 'base.html' %}
{% block title %}
    Add a Product
{% endblock title %}

{% block content %}
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="../static/form.css">
</head>

<body>
    <form class="form" method="POST">
        <h1>Add a Product</h1>
        <h2>{{ username }}</h2>
        <div class="form_input">
            {% if error %}
                <label style="color:red;">{{ error }}</label>
            {% endif %}
            <div class="form-element">
                <label for="productid">Product ID</label>
                <input type="text" name="pid" id="pid" placeholder="Enter Product ID" required>
            </div>
            <div class="form-element">
                <label for="productid">Product Name</label>
                <input type="text" name="pname" id="pname" placeholder="Enter Product Name" required>
            </div>
            <div class="form-element">
                <label for="productid">Stock Amount</label>
                <input type="number" name="stock" id="stock" placeholder="Enter Stock Amount" required>
            </div>
        </div>
    </form>
</body>
</html>
```



```

        <div class="form-element">
            <label for="productid">Price/Unit (in INR)</label>
            <input type="number" name="unitprice" id="unitprice" placeholder="Enter Price per Unit" required>
        </div>
    </div>
    <div class="btn_container">
        <input type="reset" id="button" value="Reset">
        <input type="submit" id="button" value="Add Product">
    </div>
</form>
</body>

</html>
{% endblock content %}

```

Base.html:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}{% endblock title %}</title>
    <link rel="stylesheet" href="../static/style.css">
    <script src="https://kit.fontawesome.com/aff88e4fcf.js" crossorigin="anonymous"></script>
</head>
<body>
    <div class="nav_container">
        <h2 style="color:#fff" class="logo">ReStocker</h2>
        <nav class="navigation">
            <ul>
                {% if success %}
                <li><a href="{{ url_for('.dashboard', username=username) }}">Dashboard</a></li>
                <li><a href="{{ url_for('.manageProducts', username=username) }}">Manage Products</a></li>
                <li><a href="{{ url_for('.logout') }}">Log Out</a></li>
                {% else %}
                <li><a href="/">Home</a></li>
                <li><a href="signin">Sign In</a></li>
                <li><a href="signup">Sign Up</a></li>
                {% endif %}
            </ul>
        </nav>
    </div>
    {% block content %}
    {% endblock content %}
    <div class="contact">
        <div class="waves">
            <div class="wave" id="wave1"></div>
            <div class="wave" id="wave2"></div>
            <div class="wave" id="wave3"></div>
            <div class="wave" id="wave4"></div>
        </div>
        <ul class="social_icon">
            <li><a href="#"><i class="fa-brands fa-facebook"></i></a></li>
            <li><a href="#"><i class="fa-brands fa-instagram"></i></a></li>
            <li><a href="#"><i class="fa-brands fa-linkedin"></i></a></li>
            <li><a href="#"><i class="fa-brands fa-github"></i></a></li>
        </ul>
        <ul class="bottom_nav">
            {% if success %}

```

```
<li><a href="{ url_for('.dashboard', username=username) }" >Dashboard</a></li>
    <li><a href="{ url_for('.manageProducts', username=username) }" >Manage Products</a></li>
    <li><a href="{ url_for('.logout') }" >Log Out</a></li>
    {% else %}
    <li><a href="/" >Home</a></li>
    <li><a href="signin" >Sign In</a></li>
    <li><a href="signup" >Sign Up</a></li>
    {% endif %}
</ul>
<h3>Copyrights</h3>
<ul class="owners">
    <li>Venkatesha Prabhu</li>
    <li>Sethuram</li>
    <li>Sivaraman</li>
    <li>Dhinesh</li>
</ul>
</div>
</body>
</html>
```

Dashboard.html:

```
{% extends 'base.html' %}
{% block title %}
Dashboard
{% endblock title %}

{% block content %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/static/dashboard.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body class="dashboard">
    <h1 class="heading">{{username}}'s Dashboard</h1>
    <div class="overall_inventory">
        <h3>Overall Inventory Value<br>
        Rs. {{overallValue}}
        </h3>
    </div>
    <div class="main_inventory">
        <div class="prod_inventory">
            <h3>Product-wise Inventory Value</h3>
            <table>
                <thead>
                    <th>Product</th>
                    <th>Total Value</th>
                </thead>
                <tbody>
                    {% for product in products %}
                    <tr>
                        <td>{{product['PRODUCTNAME']}}</td>
                        <td>{{product['UNITPRICE']*product['AVAILABLESTOCK']}}</td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
        <div class="chart_container">
            <canvas id="pie-chart"></canvas>
        </div>
        <div class="thres_inventory">
            <h3>Products Below Threshold Value</h3>
            <table>
                <thead>
                    <th>Product</th>
                </thead>
                <tbody>
                    {% for product in productsBelowThValue %}
                    <tr>
                        <td>{{product}}</td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
```

```

        </tbody>
    </table>
</div>
</div>
<h3>Set Threshold</h3>
<form class="thres_set" method="POST">
    <input type="number" id="threshold" name="threshold" placeholder="0" required>
    <input type="submit" value="Set Limit">
</form>
<script>
    const randomColor = () => Math.floor(Math.random()*16777215).toString(16);
    const colors = [];

    var labels=JSON.parse('{{product|tojson}}');
    var data=JSON.parse('{{price|tojson}}');

    for(let i = 0;i<labels.length;++i){
        colors[i] = "#" + randomColor();
    }

    const ctx = document.getElementById('pie-chart');
    const myChart = new Chart(ctx, {
        type: 'pie',
        data: {
            labels: labels,
            datasets: [{
                label: 'Products In Inventory',
                data: data,
                backgroundColor: colors,
            }]
        }
    });
</script>
</body>
</html>

```

{% endblock content %}

Home.html:

{% extends 'base.html' %}

{% block title %}

Home

{% endblock title %}

{% block content %}

{% if isRegistered %}

<script>

```

    window.onload=function(){
        alert("Registration Successful!, Please Login");
    }

```

</script>

{% endif %}

<div class="grid_container">

<div class="desc_container">

<h1>ReStocker</h1>

<p>

We ensure 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

</p>

<h4>New Here ? <button id="button">Sign Up</button></h4>

</div>

<div class="desc_image">

</div>

<div class="about_image">

</div>

<div class="about_container">

<h2>About</h2>

```

<p>
    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
</p>
</div>
</div>
{% endblock content %}

```

ProductsM.html:

```

{% extends 'base.html' %}
{% block title %}
    Manage Products
{% endblock title %}

{% block content %}
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="../static/productsM.css">
    {% if alertUser %}
    <script>
        window.onload=function(){
            alert("Set Threshold Value on Dashboard before Adding a Product.");
        }
    </script>
    {% endif %}
</head>

<body>
    <h2 class="heading">{{username}}'s Product Details</h1>
    <a class="addProd" id="button" href="{{url_for('.addProducts', username=username)}}">Add a Product</a>
    <table class="productsTable">
        <tr>
            <td><label class="info head">Product ID</label></td>
            <td><label class="info head">Product Name</label></td>
            <td><label class="info head">Available Stock</label></td>
            <td><label class="info head">To Add</label></td>
            <td><label class="info head">Add Stock</label></td>
            <td><label class="info head">To Remove</label></td>
            <td><label class="info head">Remove Stock</label></td>
            <td><label class="info head">Delete Stock</label></td>
        </tr>
        {% for product in products %}
        <tr>
            <td><label class="info">{{product['PRODUCTID']}}</label></td>
            <td><label class="info">{{product['PRODUCTNAME']}}</label></td>
            <td><label class="info">{{product['AVAILABLESTOCK']}}</label></td>
            <form method="POST"
                action="{{url_for('.editProduct', username=username, action='add', pid=product['PRODUCTID'])}}">
            <td>
                <label class="info">
                    <input type="number" name="newstock" id="newstock" required />

```

```

        </label>
    </td>
    <td>
        <label class="info">
            <input type="image" src="../static/add.png"/>
        </label>
    </td>
</form>
<form method="POST"
    action="{{url_for('.editProduct', username=username, action='remove', pid=product['PRODUCTID'])}}">
    <td>
        <label class="info">
            <input type="number" name="newstock" id="newstock" required />
        </label>
    </td>
    <td>
        <label class="info">
            <input type="image" src="../static/minus.png"/>
        </label>
    </td>
</form>
    <td>
        <label class="info">
            <a class="delProd" href="{{url_for('.deleteProduct', username=username,
pid=product['PRODUCTID'])}}">
                
            </a>
        </label>
    </td>
</tr>
</tbody>
</table>
</body>

</html>
{% endblock content %}

```

Signin.html :

```
{% extends 'base.html' %}
```

```
{% block title %}
```

```
    Sign In
```

```
{% endblock title %}
```

```
{% block content %}
```

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <link rel="stylesheet" href="../static/form.css">
```

```
<script>
```

```
    function setVisible(){
```

```
        var value = document.getElementById("password");
```

```
        if (value.type === "password") {
```

```
            value.type = "text";
```

```
        } else {
```

```

        value.type = "password";
    }
}
</script>
</head>

<body>
  <form class="form" method="POST">
    <h1>Good to see you !</h1>
    <h2>Welcome</h2>
    <div class="form_input">
      {% if error %}
      <label style="color:red;">{{error}}</label> <br>
      {% endif %}
      <div class="form-element">
        <label for="email">Email</label>
        <input type="email" name="email" id="email" placeholder="Email" required>
      </div>
      <div class="form-element">
        <label for="password">Password</label>
        <input type="password" name="password" id="password" placeholder="Password" required>
      </div>
      <input type="checkbox" onclick="setVisible()">Show Password
    </div>
    <div class="btn_container">
      <input type="reset" id="button" value="Reset">
      <input type="submit" id="button" value="Submit">
    </div>
    <h3>New User ? <a href="signup">Register Here</a></h3>
  </form>
</body>

</html>
{% endblock content %}

```

Signup.html:

```

{% extends 'base.html' %}
{% block title %}
  Sign Up
{% endblock title %}

{% block content %}
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="../static/form.css">
</head>

<body>
  <form class="form" method="POST">
    <h1>Hello Friend !</h1>
    <h2>Welcome to the ReStocker</h2>
    {% if error %}
      <label style="color:red;">{{error}}</label>
    {% endif %}

```

```

<div class="form-element">

    <label for="username">Username</label>
    <input type="text" name="username" id="username" placeholder="Username" required>
</div>
<div class="form-element">
    <label for="email">Email</label>
    <input type="email" name="email" id="email" placeholder="Email" required>
</div>
<div class="form-element">
    <label for="password">Password</label>
    <input type="password" name="password" id="password" placeholder="Password" required>
</div>
<div class="btn_container">
    <input type="reset" id="button" value="Reset">
    <input type="submit" id="button" value="Submit">
</div>
<h3>Existing User ? <a href="signin">Login Here</a></h3>
</form>
</body>
</html>

```

{% endblock content %}

App.py:

```

from flask import Flask
app=Flask(__name__)
app.secret_key = "121212"

```

deployment.yaml:

```

apiVersion: v1
kind: Service
metadata:
  name: restocker-service
spec:
  selector:
    app: restocker
  ports:
    - port: 5000
    type: NodePort
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: restocker-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: restocker
  template:
    metadata:
      labels:
        app: restocker
    spec:
      containers:
        - name: restocker
          image:
            icr.io/ibmcr/restocker@sha256:cc14024365de9d07f408062bb0e8b779bc265c0024b345fa5e5b369325400a14
          imagePullPolicy: Always

```

ports:

- containerPort: 5000

Dockerfile:

```
FROM python:3.10.7
WORKDIR /app
ADD . /app
COPY requirements.txt /app
RUN python3 -m pip install -r requirements.txt
EXPOSE 5000
CMD ["python", "run.py"]
```

ROUTER.PY:

```
from app import app
from flask import request, redirect, url_for, render_template, session, json
import ibm_db
from sendGrid import mailto, checkstatus, getProductsBelowThValue
```

```
#Connect to DB
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=fbd88901-ebdb-4a4f-a32e-
9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32731;SECURITY=SSL;SSLServerCertificate
=DigiCertGlobalRootCA.crt;UID=glr36049;PWD=dcAymdrrrHG3zGIs;", "", "")
```

```
@app.route("/")
```

```
def root():
```

```
    return render_template("home.html", title="Home")
```

```
@app.route("/signin", methods=('POST', 'GET'))
```

```
def signin():
```

```
    error = None
```

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

```
        email = request.form['email']
```

```
        password = request.form['password']
```

```
        sql = "SELECT username FROM users WHERE password = '{}' AND email = '{}".format(password,
email)
```

```
        stmt = ibm_db.exec_immediate(conn, sql)
```

```
        fetchUser = ibm_db.fetch_assoc(stmt)
```

```
        if fetchUser == False:
```

```
            error = "Incorrect Username/Password."
```

```
        if error is None:
```

```
            user = fetchUser["USERNAME"]
```

```
            session['loggedIn'] = True
```

```
            session['id'] = user
```

```
            session['email'] = email
```

```
            return redirect(url_for('.dashboard', username=user))
```

```
        return render_template('signin.html', error=error)
```

```
@app.route("/signup", methods=('POST', 'GET'))
```

```
def signup():
```

```
    error=None
```

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

```
        username = request.form['username']
```



```

email = request.form['email']
password = request.form['password']

checkUser = "SELECT * FROM users WHERE username = '{}'.format(username)
stmt = ibm_db.exec_immediate(conn, checkUser)
findUser = ibm_db.fetch_assoc(stmt)
if findUser == False:
    sql = "INSERT INTO users (email, username, password) VALUES ('{}', '{}', '{}');".format(email,
username, password)
    ibm_db.exec_immediate(conn, sql)
    return render_template('home.html', title="Home", isRegistered=True)
    error="Username already exists."
return render_template("signup.html", error=error)

@app.route("/<username>/dashboard", methods=('POST', 'GET'))
@app.route("/<username>", methods=('POST', 'GET'))
def dashboard(username):
    fetchPrices = "SELECT p.productname,up.unitprice,up.availablestock FROM products p,userproducts up
WHERE p.productid=up.productid AND up.username='{}'.format(username)
    stmt = ibm_db.exec_immediate(conn, fetchPrices)
    productInfo = ibm_db.fetch_assoc(stmt)
    allproducts=[]
    product=[]
    price=[]
    overallValue=0
    while productInfo != False:
        allproducts.append(productInfo)
        product.append(productInfo['PRODUCTNAME'])
        price.append(productInfo['UNITPRICE']*productInfo['AVAILABLESTOCK'])
        overallValue=overallValue+(productInfo['UNITPRICE']*productInfo['AVAILABLESTOCK'])
        productInfo = ibm_db.fetch_assoc(stmt)

    fetchPrices = "SELECT * FROM threshold_value WHERE email='{}';".format(session['email'])
    stmt = ibm_db.exec_immediate(conn, fetchPrices)
    productInfo = ibm_db.fetch_assoc(stmt)
    if productInfo == False:
        productsBelowThValue = []
    else:
        productsBelowThValue = getProductsBelowThValue(conn, session['email'], username)

    if request.method=='POST':
        th_value = request.form['threshold']
        sql = "INSERT INTO threshold_value (email, th_value) VALUES ('{}', '{}');".format(session['email'],
th_value)
        ibm_db.exec_immediate(conn, sql)
        return redirect(url_for('.dashboard', username=username))
    return render_template("dashboard.html", username=username, success=True, products=allproducts,
product=product, price=price, overallValue=overallValue, productsBelowThValue=productsBelowThValue)

@app.route("/<username>/manageProducts", methods=('POST', 'GET'))
def manageProducts(username):
    sql = "SELECT up.productid,p.productname,up.availablestock FROM products p, users u, userproducts up
WHERE u.username=up.username AND p.productid=up.productid AND u.username='{}';".format(username)
    stmt = ibm_db.exec_immediate(conn, sql)
    getProducts = ibm_db.fetch_assoc(stmt)
    products = []
    while getProducts != False:
        products.append(getProducts)

```

```
getProducts = ibm_db.fetch_assoc(stmt)
return render_template("productsM.html", username=username, success=True, products=products)
```

```
@app.route("/<username>/manageProducts/edit=<pid>,action=<action>", methods=('POST', 'GET'))
```

```
def editProduct(username, pid, action):
```

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

```
        stock = int(request.form['newstock'])
```

```
        checkAvailable = "SELECT * FROM userproducts WHERE productid='{}' AND
```

```
username='{}';".format(pid, username)
```

```
        statement = ibm_db.exec_immediate(conn, checkAvailable)
```

```
        productDetails = ibm_db.fetch_assoc(statement)
```

```
        available = productDetails['AVAILABLESTOCK']
```

```
        if action == "add":
```

```
            stock = available + stock
```

```
        else:
```

```
            stock = available - stock
```

```
            checkstatus(conn, session['email'], username)
```

```
            updateQuery = "UPDATE userproducts SET availablestock='{}' WHERE productid='{}' AND
```

```
username='{}';".format(stock, pid, username)
```

```
            ibm_db.exec_immediate(conn, updateQuery)
```

```
            return redirect(url_for('.manageProducts', username=username))
```

```
@app.route("/<username>/manageProducts/delete=<pid>", methods=('POST', 'GET'))
```

```
def deleteProduct(username, pid):
```

```
    deleteQuery = "DELETE FROM userproducts WHERE productid='{}' AND username='{}';".format(pid,
```

```
username)
```

```
    ibm_db.exec_immediate(conn, deleteQuery)
```

```
    return redirect(url_for('.manageProducts', username=username))
```

```
@app.route("/<username>/addProduct", methods=('POST', 'GET'))
```

```
def addProducts(username):
```

```
    error = None
```

```
    isValueSet = "SELECT * FROM threshold_value WHERE email='{}';".format(session['email'])
```

```
    statement = ibm_db.exec_immediate(conn, isValueSet)
```

```
    isSet = ibm_db.fetch_assoc(statement)
```

```
    if isSet == False:
```

```
        return render_template("productsM.html", username=username, alertUser=True)
```

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

```
        pid = request.form['pid']
```

```
        pname = request.form['pname']
```

```
        addstock = int(request.form['stock'])
```

```
        unitprice = int(request.form['unitprice'])
```

```
        checkDuplicate = "SELECT * FROM products WHERE productid='{}' AND
```

```
productname<>'{}';".format(pid, pname)
```

```
        statement = ibm_db.exec_immediate(conn, checkDuplicate)
```

```
        productDetails = ibm_db.fetch_assoc(statement)
```

```
        if productDetails != False:
```

```
            error = "Product ID is already assigned"
```

```
            return render_template("addProduct.html", username=username, success=True, error=error)
```

```
        checkAvailable = "SELECT * FROM userproducts WHERE productid='{}' AND
```

```
username='{}';".format(pid, username)
```

```
        statement = ibm_db.exec_immediate(conn, checkAvailable)
```

```
        productDetails = ibm_db.fetch_assoc(statement)
```

```
        if productDetails == False:
```

```

        checkProduct = "SELECT * FROM products WHERE productid='{ }';".format(pid)
        statement = ibm_db.exec_immediate(conn, checkProduct)
        productDetails = ibm_db.fetch_assoc(statement)
        if productDetails == False:
            addProduct = "INSERT INTO products(productid, productname) VALUES ('{ }', '{ }');".format(pid,
pname)

            ibm_db.exec_immediate(conn, addProduct)

        updateStock = "INSERT INTO userproducts(productid, username, availablestock, unitprice) VALUES
('{ }', '{ }', '{ }', '{ }');".format(pid, username, addstock, unitprice)
        ibm_db.exec_immediate(conn, updateStock)

    else:
        addstock = addstock + productDetails['AVAILABLESTOCK']
        updateStock = "UPDATE userproducts SET availablestock='{ }' WHERE productid='{ }' AND
username='{ }';".format(addstock, pid, username)
        ibm_db.exec_immediate(conn, updateStock)
        return redirect(url_for('.manageProducts', username=username))
    return render_template("addProduct.html", username=username, success=True, error=error)

@app.route('/')
def logout():
    session.clear()
    return render_template('home.html')

```

Run.py:

```

import app
from router import *
if __name__=="__main__":
    app.run(host='0.0.0.0')

```

SendGrid.py:

```

from sendgrid import SendGridAPIClient, Mail
import ibm_db

```

```

def mailto(email, subject, content):
    message = Mail(from_email='sethurat@student.tce.edu', to_emails=email,
        subject=subject, html_content='<strong> { } </strong>'.format(content))
    try:
        sg = SendGridAPIClient("Enter your api key here")
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e.message)

def checkstatus(userDB, email, username):
    sql = "SELECT th_value FROM threshold_value WHERE email='{ }'".format(email)
    stmt = ibm_db.exec_immediate(userDB, sql)
    res = ibm_db.fetch_assoc(stmt)
    th_value = res['TH_VALUE']
    sql = "SELECT * FROM USERPRODUCTS WHERE username='{ }'".format(username)
    stmt = ibm_db.exec_immediate(userDB, sql)
    res = ibm_db.fetch_assoc(stmt)
    while res != False:

```

```

curr_stocks = int(res['AVAILABLESTOCK'])
if curr_stocks < th_value:
    pid = res['PRODUCTID']
    sql = "SELECT productname FROM PRODUCTS WHERE productid='{0}'".format(pid)
    stmt = ibm_db.exec_immediate(userDB, sql)
    res = ibm_db.fetch_assoc(stmt)
    prod_name = res['PRODUCTNAME']
    mailto(email, "threshold alert", "Alert! product '{0}' available stocks had dropped below the threshold
limit '{1}'".format(prod_name, th_value))
    res = ibm_db.fetch_assoc(stmt)

def getProductsBelowThValue(userDB, email, username):
    prodList = []
    sql = "SELECT th_value FROM threshold_value WHERE email='{0}'".format(email)
    stmt = ibm_db.exec_immediate(userDB, sql)
    res = ibm_db.fetch_assoc(stmt)
    th_value = res['TH_VALUE']
    sql = "SELECT * FROM USERPRODUCTS WHERE username='{0}'".format(username)
    stmt = ibm_db.exec_immediate(userDB, sql)
    res = ibm_db.fetch_assoc(stmt)
    while res != False:
        curr_stocks = int(res['AVAILABLESTOCK'])
        if curr_stocks < th_value:
            pid = res['PRODUCTID']
            sql = "SELECT productname FROM PRODUCTS WHERE productid='{0}'".format(pid)
            stmt = ibm_db.exec_immediate(userDB, sql)
            res = ibm_db.fetch_assoc(stmt)
            prod_name = res['PRODUCTNAME']
            prodList.append(prod_name)
            res = ibm_db.fetch_assoc(stmt)

    return prodList

```

STYLE.CSS:

```

*{
    padding:0;
    margin:0;
    box-sizing: border-box;
}
.grid_container{
    display:grid;
    grid-template-areas:
        'nav nav nav'
        'desc desc desc_image'
        'about_image about about'
        'contact contact contact';
    grid-template-columns: repeat(3,minmax(1fr,2fr));
    background-color: rgba(53, 134, 255,0.1);
}
.nav_container{
    grid-area:nav;
    position:relative;
    top:0;
    width:100%;
    height:12vh;
    display:flex;
    justify-content: space-between;
    align-items: center;

```

```

    background-color: #3586ff;
}
.nav_container .logo{
    margin-left:80px;
}
.nav_container .navigation ul{
    display:flex;
    justify-content: space-between;
}
.nav_container .navigation ul li{
    margin:0 20px;
    list-style-type:none;
}
.nav_container .navigation ul li a{
    text-decoration: none;
    padding:10px 20px;
    font-size:18px;
    color:#fff;
    font-weight: bold;
}
.nav_container .navigation ul .active{
    border-radius:20px;
}
.nav_container .sign_in_up_container{
    margin-right:4%;
}
.nav_container .sign_in_up_container button{
    padding:10px 25px;
    margin:0 10px;
    font-size:15px;
    background-color: transparent;
    border-radius: 20px;
    color:#fff;
    border-color: #fff;
}
.desc_container{
    grid-area:desc;
    height:88vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
    padding:0 50px;
}
.desc_container h1{
    font-size:50px;
    font-weight: bold;
    margin-bottom:10px;
}
.desc_container p{
    font-size:20px;
    margin-bottom:20px;
}
.desc_container h4{
    font-size:18px;
    font-weight: bolder;
}
.desc_container button{
    padding:10px 25px;
    margin:0 10px;
    font-size:15px;
    background-color: transparent;

```

```
border-radius: 20px;
border-color:#3586ff;
cursor: pointer;
}
.desc_image{
grid-area: desc_image;
display: flex;
align-items: center;
}
.desc_image img{
min-width: 400px;
min-height: 350px;
}
.about_image{
grid-area:about_image;
position: relative;
display: flex;
}
.about_image img{
max-height:350px;
position: relative;
top:-80px;
}
.about_container{
grid-area:about;
height:70vh;
display: flex;
flex-direction: column;
justify-content: start;
padding:0 50px;
}
.about_container h2{
font-size:40px;
text-align: center;
margin-bottom:10px;
}
.about_container p{
font-size: 20px;
text-align: center;
}
.contact{
grid-area:contact;
position: relative;
width:100%;
min-height:200px;
background-color: #3586ff;
}
.contact ul{
display:flex;
justify-content: center;
margin:20px 0;
}
.contact ul li{
padding:0 20px;
list-style-type: none;
color:#fff;
}
.contact ul li a{
text-decoration: none;
color:#fff;
display:inline-block;
```

```

    transition: 0.5s;
}
.contact .social_icon li a{
    font-size:32px;
}
.contact .social_icon li a:hover{
    transform: translateY(-10px);
}
.contact .bottom_nav li a{
    font-size:18px;
}
.contact h3{
    color:#fff;
    text-transform: uppercase;
    text-align: center;
    margin-top:30px;
    margin-bottom:-10px;
}
.contact .owners li{
    font-size:16px;
}
.wave{
    width:100%;
    height:100px;
    position:absolute;
    top:-100px;
    left:0;
    background:url(wave.png);
    background-size: 1000px 100px;
}
.wave#wave1{
    z-index:1000;
    opacity:1;
    bottom:0;
    animation:animatewave 4s linear infinite;
}
.wave#wave2{
    z-index:999;
    opacity:0.5;
    bottom:10px;
    animation:animatewave_2 4s linear infinite;
}
.wave#wave3{
    z-index:1000;
    opacity:0.2;
    bottom:15px;
    animation:animatewave 3s linear infinite;
}
.wave#wave4{
    z-index:999;
    opacity:0.7;
    bottom:20px;
    animation:animatewave_2 3s linear infinite;
}
@keyframes animatewave {
    0%{
        background-position-x: 1000px;
    }
    100%{
        background-position-x: 0px;
    }
}

```

```
}  
@keyframes animatewave_2 {  
  0%{  
    background-position-x: 0px;  
  }  
  100%{  
    background-position-x: 1000px;  
  }  
}
```