

<b>Team ID</b>	<b>PNT2022TMID30805</b>
<b>Project Name</b>	<b>Inventory Management System For Retailers</b>
<b>Team Members</b>	<b>620119104077 - S RAAGUL</b> <b>620119104078 - P RAGAVAN</b> <b>620119104101 - S TAMILSELVAN</b> <b>620119104108 - K VELMURUGAN</b>

## **TABLE OF CONTENT**

<b>CHAPTER NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
<b>1.</b>	<b>INTRODUCTION</b>	
	1.1 Project Overview	1
	1.2 Purpose	1
<b>2.</b>	<b>LITERATURE SURVEY</b>	
	2.1 Existing Problem	2
	2.2 References	2
	2.3 Problem Statement Definition	3
<b>3.</b>	<b>IDEATION &amp; PROPOSED SOLUTION</b>	
	3.1 Empathy Map Canvas	4
	3.2. Ideation & Brainstorming	5
	3.3 Proposed Solution	9
	3.4 Problem Solution Fit	10
<b>4.</b>	<b>REQUIREMENT ANALYSIS</b>	
	4.1 Functional Requirements	11
	4.2 Non-Functional Requirements	12

<b>5.</b>	<b>PROJECT DESIGN</b>	
	5.1 Data Flow Diagram	13
	5.2 Solution & Technical Architecture	14
	5.3 User Stories	15
<b>6.</b>	<b>PROJECT PLANNING &amp; SCHEDULING</b>	
	6.1 Sprint Planning & Estimation	16
	6.2 Sprint Delivery Schedule	17
	6.3 Reports from JIRA	17
<b>7.</b>	<b>CODING &amp; SOLUTIONING</b>	
	7.1 Feature 1	18
	7.2 Feature 2	18
	7.3 Feature 3	19
	7.4 Feature 4	19
	7.5 Feature 5	21
	7.6 Feature 6	22
	7.7 Feature 7	23

<b>8.</b>	<b>TESTING</b>	
	8.1 Test Cases	24
	8.2 User Acceptance Testing	25
<b>9.</b>	<b>RESULTS</b>	
	9.1 Performance Metrics	29
<b>10.</b>	<b>ADVANTAGES &amp; DISADVANTAGES</b>	30
<b>11.</b>	<b>CONCLUSION</b>	31
<b>12.</b>	<b>FUTURE SCOPE</b>	32
<b>13.</b>	<b>APPENDIX</b>	
	Source Code	33
	GitHub Link	47

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 PROJECT OVERVIEW**

Our project is a cloud based web application that is specifically implemented to make the lives of warehouse workers much easier. It is an inventory management system for all the retailers out there in the market where they can manage, add, delete and track their goods that are being imported and exported through all locations. By managing inventory, retailers meet customer demand without running out of stock or carrying excess supply. This results in lower costs and gives them a better understanding on sales patterns.

### **1.2 PURPOSE**

The goal is to assist shops in keeping track of and managing stock levels for their own products. The system will ask the merchants to set up their accounts by giving necessary information. Retailers can update their inventory details after successfully logging into the programme. Users can also add new merchandise by providing the necessary stock-related information. They have access to their merchandise at any time. Additionally, we made use of the Send Grid email service, which, in the event that there is no stock found in the accounts of the retailers, notifies them via email. At that time, they can also place new stock orders.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 EXISTING PROBLEM**

Warehouses of a single organization can be in different locations. It makes it really hard for the admin to keep track of all the goods across all the warehouses. Management of these information is really essential for purchasing goods on the proper time. Also these data can be used to get an insight on the recent trends for efficient purchase of goods. Also, manual tracking leads to a lot of human errors. There also exists some communication gaps between the workers and the admin which makes it even harder to keep track of the products across the warehouses.

#### **2.2 REFERENCES**

1. G. Hançerlioğulları, A. Şen, y E. A. Aktunç, “Demand uncertainty and inventory turnover performance: an empirical analysis of the US retail industry”, *International Journal of Physical Distribution and Logistics Management*, vol. 46, number. 6–7, pp. 681–708, 2016, doi: 10.1108/IJPDLM-12-2014-0303
2. Y. Wang, S. W. Wallace, B. Shen, y T.-M. Choi, “Service supply chain management: A review of operational models”, *European Journal of Operational Research*, vol. 247, numb. 3, pp. 685–698, 2015
3. S. Mahar y P. D. Wright, “The value of postponing online fulfillment decisions in multi-channel retail/e-tail organizations”, *Computers & operations research*, vol. 36, numb. 11, pp. 3061–3072, 2009
4. M. Barratt, T. J. Kull, y A. C. Sodero, “Inventory record inaccuracy dynamics and the role of employees within multi-channel distribution center inventory systems”, *Journal of Operations Management*, vol. 63, numb. 1, pp. 6–24, Nov. 2018, doi: 10.1016/j.jom.2018.09.003

5. A. Ros s, M. Khajehnezhad, W. Otieno, y O. Aydas , “Integrated location inventory modelling under forward and reverse product flows in the used merchandise retail sector: A multi-echelon formulation”, *European Journal of Operational Research*, vol. 259, numb. 2, pp. 664–676, 2017, doi: 10.1016/j.ejor.2016.10.036

## **2.3 PROBLEM STATEMENT DEFINITION**

Retail inventory management is the process of ensuring you carry merchandise that shoppers want, with neither too little nor too much on hand. By managing inventory, retailers meet customer demand without running out of stock or carrying excess supply. In practice, effective retail inventory management results in lower costs and a better understanding of sales patterns. Retail inventory management tools and methods give retailers more information on which to run their businesses. Applications have been developed to help retailers track and manage stocks related to their own products. The System will ask retailers to create their accounts by providing essential details. Retailers can access their accounts by logging into the application. Once retailers successfully log in to the application they can update their inventory details, also users will be able to add new stock by submitting essential details related to the stock. They can view details of the current inventory. The System will automatically send an email alert to the retailers if there is no stock found in their accounts. So that they can order new stock.

## CHAPTER 3

### IDEATION & PROPOSED SOLUTION

#### 3.1 EMPATHY MAP CANVAS





### 3.2 IDEATION AND BRAINSTORMING



**Fig 3.1: Problem Definition**

## Brainstorm

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

🕒 10 minutes

### S Raagul

Enhanced user interface	Periodic analysis of Sales reports	Send mail when minimum stock limit is reached
Maintain proper product categories	Managing customer feedback	Tax calculations

### P Ragavan

Avoid overstocking of products	Enable remote access of software	Managing customer account
Maintain records for the product	Enabling multiple payment options	Occasional discounts for the products

### S Tamilselvan

Analyze low and high selling products	Monitor products for cost changes	Enabling customer return policy
Provide product insights	Managing multiple orders	Display graphs to show clear picture

### K Velmurugan

Display a dashboard containing stock details	Payment status tracking	Managing stock details
Maintaining a unique product number	Profit and Loss analysis	Data privacy for customers

Fig 3.2: Brainstorm

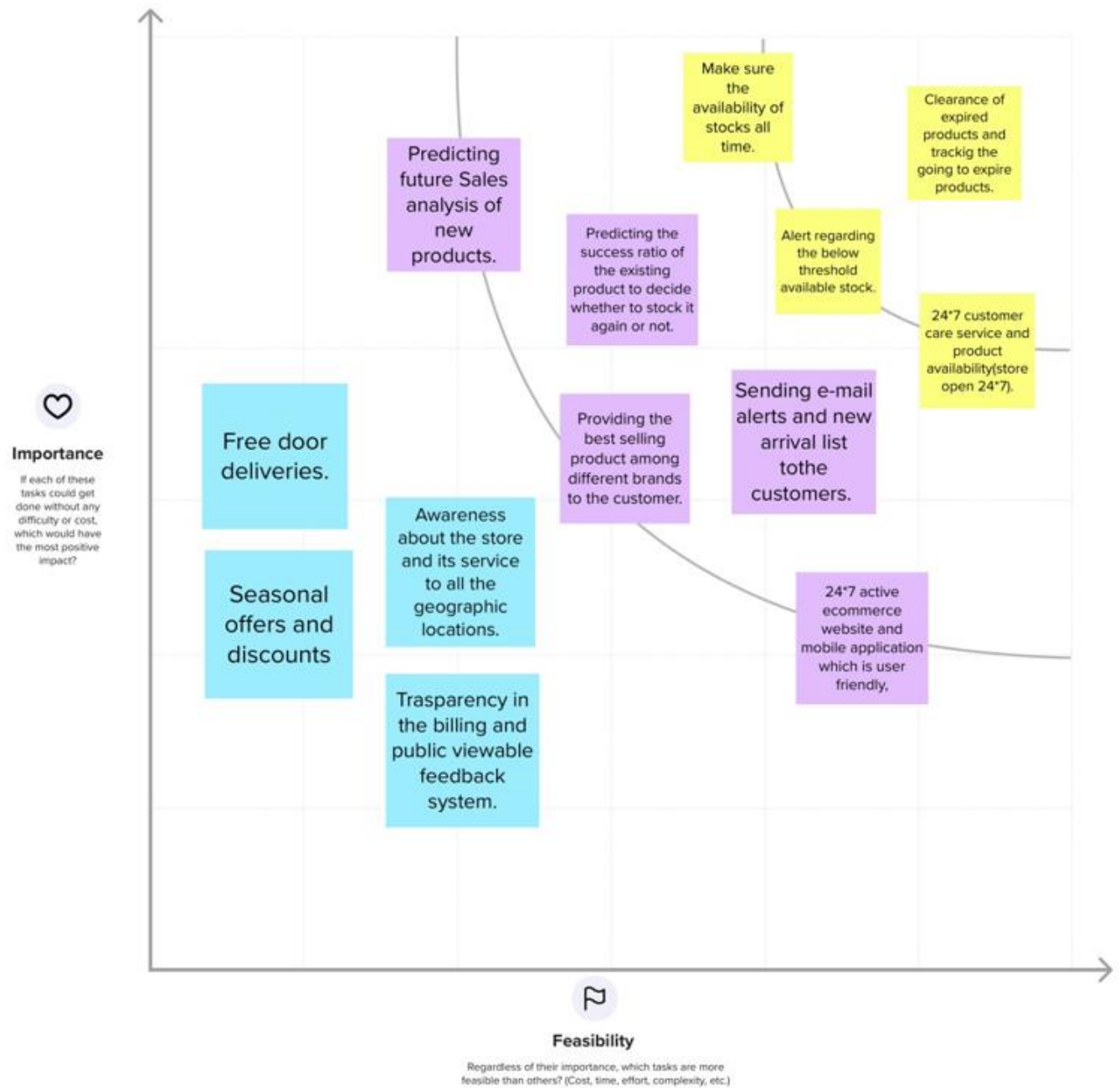
### Group ideas

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

⌚ 20 minutes



**Fig 3.3: Group ideas**



**Fig 3.4: Prioritize**

### 3.3 PROPOSED SOLUTION

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	Inventory systems, demand is usually uncertain, and the lead-time can also vary. To avoid shortages, managers often maintain a safety stock. In such situations, it is not clear what order quantities and reorder points will minimize expected total inventory cost.
2.	Idea / Solution description	To develop an end-to-end web application which in default shows the amount of stock present in the inventory at that time. Users can add or reduce the number of goods based on purchase and sales.
3.	Novelty / Uniqueness	Track inventory across multiple locations and automatically notify when products count reaches a certain limit. This helps in saving time.
4.	Social Impact / Customer Satisfaction	It makes the life of retailers easier as it helps them keeping track of items that are stored in their warehouse.
5.	Business Model (Revenue Model)	We can charge users based on the number of warehouses they add
6.	Scalability of the Solution	Inventory data can be scaled up and scaled down based on the number of available inventory in the warehouse.

### 3.4 PROBLEM SOLUTION FIT

Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> <p>Our proposed model targets the distributors, wholesalers, manufacturers and retailers to track their stocks.</p>	<b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span> <p>Too much stock on hand can be just as hazardous as not enough. Overstock negatively affects a company's cash flow and causes issues with storage and loss of inventory. Also doesn't come to know about the stocks which is to be short.</p>	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> <p>It is laborious and unsafe to manage inventory with paperwork and manual procedures. Additionally, scaling across several warehouses with a lot of goods is difficult. Provide workers with the appropriate inventory tools for the job. Software is required to replace manual inventory tracking, and purchase orders and invoices must be processed without the use of paper.</p>	Explore AS, differentiate
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>J&amp;P</span> <p>The problem faced by them is that it is difficult to manage the large amount of inventory data. They have maintain the hardcopy of the inventory, it is difficult to organize properly. Pen and paper work is too tedious.</p>	<b>9. PROBLEM ROOT CAUSE</b> <span>RC</span> <p>Difficulty in managing the large amount of stocks using pen and paper and struggles in managing the stocks data without centralized data storage.</p>	<b>7. BEHAVIOUR</b> <span>BE</span> <p>It is time-consuming, redundant, and prone to errors to use manual inventory tracking techniques across various programmes and spreadsheets. An integrated central inventory management system with accounting capabilities might be helpful for even small retailers.</p>	
Identify strong TR & EM	<b>3. TRIGGERS</b> <span>TR</span> <p>This inventory management method will inspire distributors, retailers who own markets or wholesale enterprises by making them to handle the data easily.</p>	<b>10. YOUR SOLUTION</b> <span>SL</span> <p>Our aim is to design the inventory management system to increase the scalability of the retailers business with the help of automated inventory management system and also aim to save the time. The customer can able to track the sold stocks and availability of stocks. They get notified when the stock is about to end.</p>	<b>8. CHANNELS of BEHAVIOUR</b> <span>CH</span> <b>8.1 ONLINE</b> <p>Collecting information from various websites and utilise it efficiently.</p>	Extract online & offline CH of BE
	<b>4. EMOTIONS: BEFORE / AFTER</b> <span>EM</span> <p>Before: Depressed, Worn out of managing stocks.  After : Stress less, Enthusiastic in works.</p>		<b>8.2 OFFLINE</b> <p>Collecting feedbacks to improve the efficiency of the system.</p>	

## CHAPTER 4

### REQUIREMENT ANALYSIS

#### 4.1 FUNCTIONAL REQUIREMENTS

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Login	Login with username Login with password
FR-3	Product record	Product ID Product name Product Count Minimum count to trigger reorder notification Maximum count Product category Vendor details
FR-4	Email Notification	Email through SendGridReduced stock quantity Email to both retailer and seller
FR-5	Audit Monitoring	Monitor incoming and outgoing stock

## 4.2 NON FUNCTIONAL REQUIREMENTS

<b>NFR No.</b>	<b>Non-Functional Requirement</b>	<b>Description</b>
NFR-1	<b>Usability</b>	Highly portable, User-friendly and highly responsive UI for easy access
NFR-2	<b>Security</b>	Access Control, User privileges, Password management features, Hashed Password Storage
NFR-3	<b>Reliability</b>	Secure server for reliable and fault tolerant connection
NFR-4	<b>Performance</b>	The System shall be able to handle multiple requests at any given point in time and generate an appropriate response.
NFR-5	<b>Availability</b>	It is a cloud-based web application so user can access without any platform limitations, just using a browser with an internet connection is enough for use the application
NFR-6	<b>Scalability</b>	As the business grows, the users can keep track of stocks in multiple warehouses located at various locations without any hassle



## CHAPTER 5

### PROJECT DESIGN

#### 5.1 DATA FLOW DIAGRAM

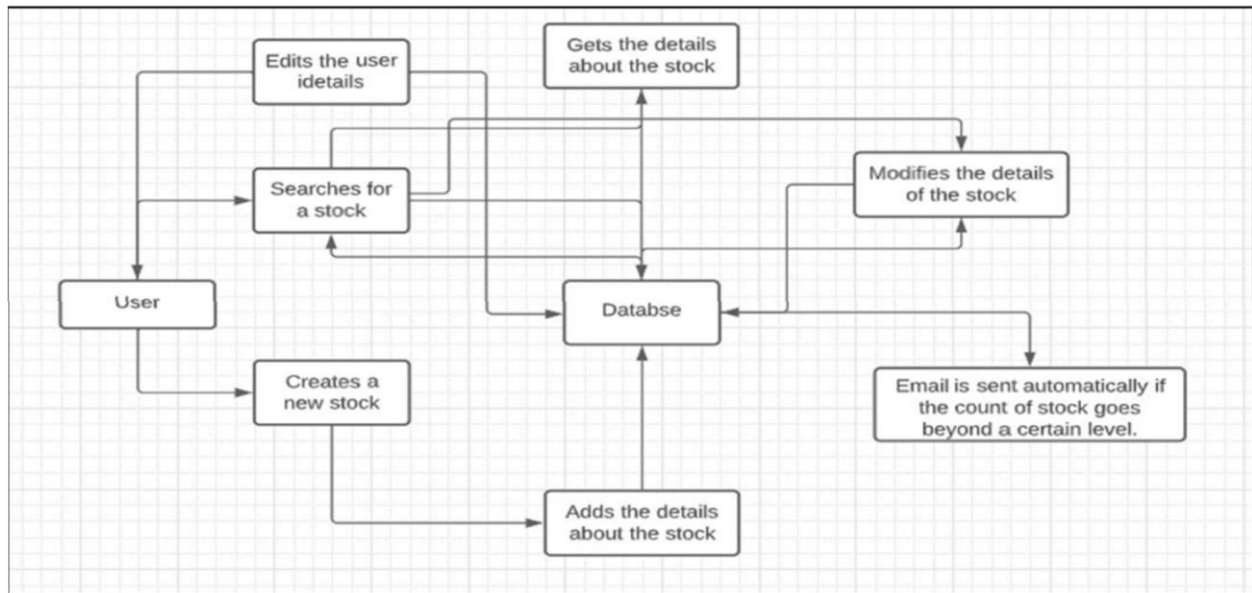
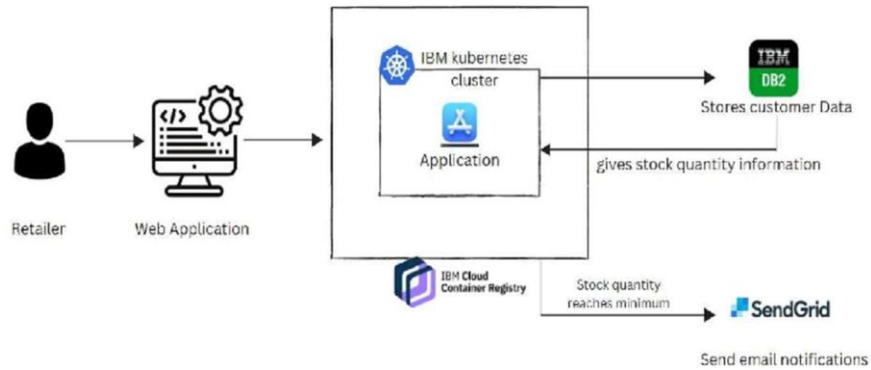
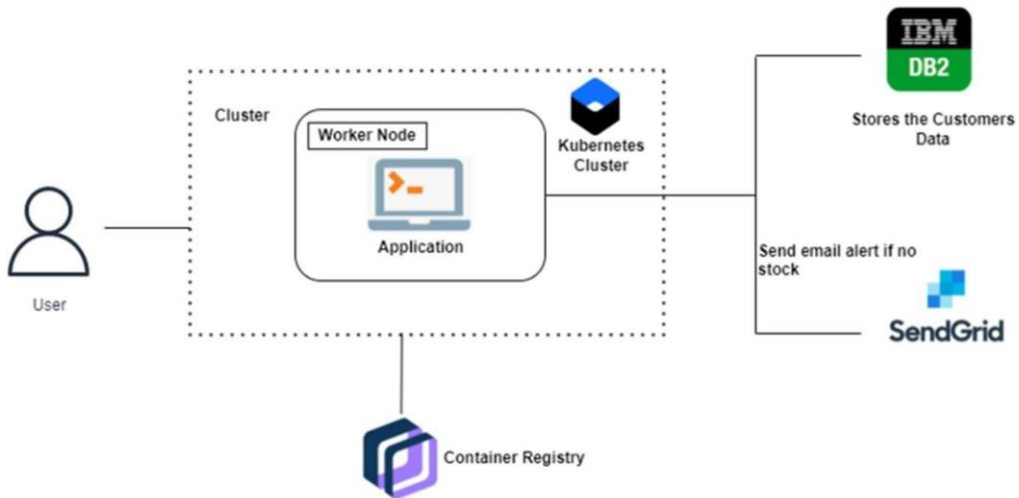


Fig 5.1: Data Flow Diagram of Inventory Management

## 5.2 SOLUTION AND TECHNICAL ARCHITECTURE



**Fig 5.2: Solution Architecture Diagram**



**Fig 5.3: Technical Architecture**

## 5.3 USER STORIES

User Type	Functional Requirement(Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Webuser)	Registration	USN-1	As a user, I can register for the application by entering my email and password and confirming my password.	I can access my account /dashboard	High	Sprint-1
	Login and Authentication	USN-2	As a user, I can log into the application by entering email & password	I can Sign In	High	Sprint-1
	Management	USN-4	As a user, I can add warehouses and add products to them	I can add warehouses	High	Sprint-3
	Dashboard	USN-3	As a user, I can log into my account and access the Dashboard	I can access the Dashboard	High	Sprint-2
	Notification	USN-5	As a user, I should get mail if certain products count goes below the threshold count specified by me	I should receive notification mail	Medium	Sprint-4

	USN-6	As an admin user, I can edit my details and change my Inventory name	I can edit my details and change inventory name	High	Sprint-2
Management	USN-7	As an admin user, I can add warehouses and add/remove products to them	I can add warehouses and add/ remove products	High	Sprint-3
	USN-8	As a normal user, I can add warehouses and remove products to them	I can remove products	High	Sprint-3
Notification	USN-9	As a user, I should get mail if certain products count goes below the threshold count specified by me  As an admin user, I should get mail if certain products count goes below the threshold count specified by me	I should receive notification mail	Medium	Sprint-4

## CHAPTER 6

### PROJECT PLANNING AND SCHEDULING

#### 6.1 Sprint Planning and Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by using my email & password and confirming my login credentials.	3	High	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-1		USN-2	As a user, I can login through my E-mail.	3	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-1	Confirmation	USN-3	As a user, I can receive my confirmation email once I have registered for the application.	2	High	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-1	Login	USN-4	As a user, I can log in to the authorized account by entering the registered email and password.	3	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN

Sprint-2	Dashboard	USN-5	As a user, I can view the products that are available currently.	4	High	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-2	Stocks update	USN-6	As a user, I can add products which are not available in the inventory and restock the products.	3	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-3	Sales prediction	USN-7	As a user, I can get access to sales prediction tool which can help me to predict better restock management of product.	6	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-4	Request for customer care	USN-8	As a user, I am able to request customer care to get in touch with the administrators and enquire the doubts and problems.	4	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN
Sprint-4	Giving feedback	USN-9	As a user, I am able to send feedback forms reporting any ideas for improving or resolving any issues I am facing to get it resolved.	3	Medium	S RAAGUL P RAGAVAN S TAMILSELVAN K VELMURUGAN

## 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:

The screenshot shows the JIRA Backlog for the 'smart railways' project. The left sidebar contains navigation links for Planning (Roadmap, Backlog, Board), Development (Code), Project pages, Add shortcut, and Project settings. The main content area displays the Backlog with three sprints:

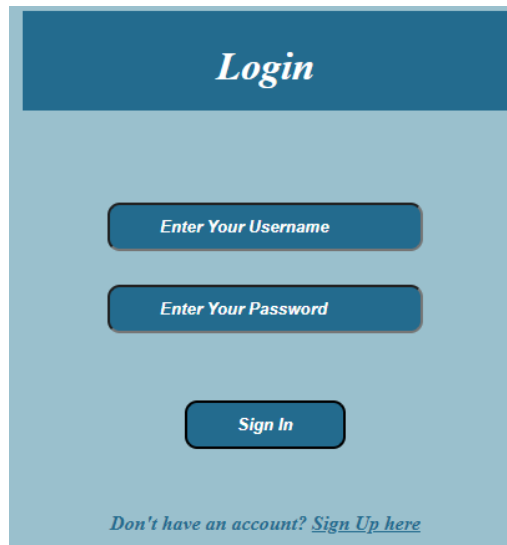
- SR Sprint 1** (24 Oct - 29 Oct, 2 issues):
  - SR-1 Create Login Page (DONE)
  - SR-2 Create Registration Page (DONE)
- SR Sprint 2** (31 Oct - 5 Nov, 2 issues):
  - SR-3 Create Seats Booking Page (DONE)
  - SR-4 Create Ticket Booking Portal (DONE)
- SR Sprint 3** (7 Nov - 12 Nov, 2 issues):
  - SR-5 Create Ticket Generation (DONE)

A 'Quickstart' button is visible at the bottom right of the sprint list.

## CHAPTER 7

### CODING & SOLUTIONING

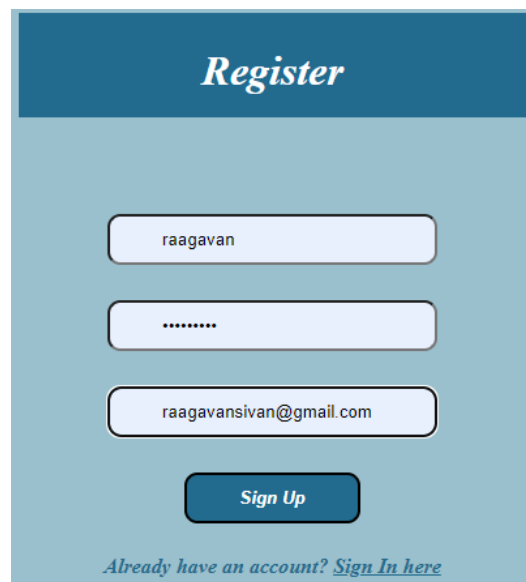
#### 7.1 FEATURE 1:



The image shows a login form with a dark blue header containing the word "Login" in white italicized font. Below the header, there are three rounded rectangular input fields: the first contains "Enter Your Username", the second contains "Enter Your Password", and the third is a button labeled "Sign In". At the bottom, there is a link that says "Don't have an account? [Sign Up here](#)".

Figure 7.1 : Sign in Page

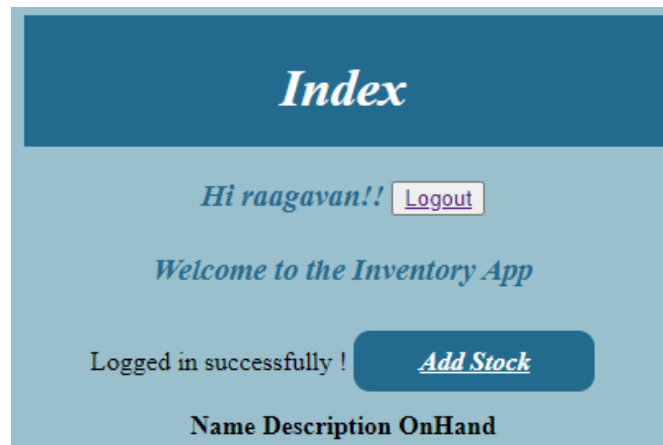
#### 7.2 FEATURE 2:



The image shows a register form with a dark blue header containing the word "Register" in white italicized font. Below the header, there are three rounded rectangular input fields: the first contains "raagavan", the second contains "\*\*\*\*\*", and the third contains "raagavansivan@gmail.com". Below these fields is a button labeled "Sign Up". At the bottom, there is a link that says "Already have an account? [Sign In here](#)".

Figure 7.2 : Sign up Page

### 7.3 FEATURE 3:



The Index Page features a dark blue header with the word "Index" in white italicized font. Below the header, the text "Hi raagavan!!" is displayed in a dark blue italicized font, followed by a "Logout" button with a purple border. A "Welcome to the Inventory App" message is shown in a dark blue italicized font. A "Logged in successfully !" message is displayed in a dark blue italicized font, followed by an "Add Stock" button with a dark blue background and white italicized text. At the bottom, a table header is visible with the columns "Name", "Description", and "OnHand".

*Index*

*Hi raagavan!!* [Logout](#)

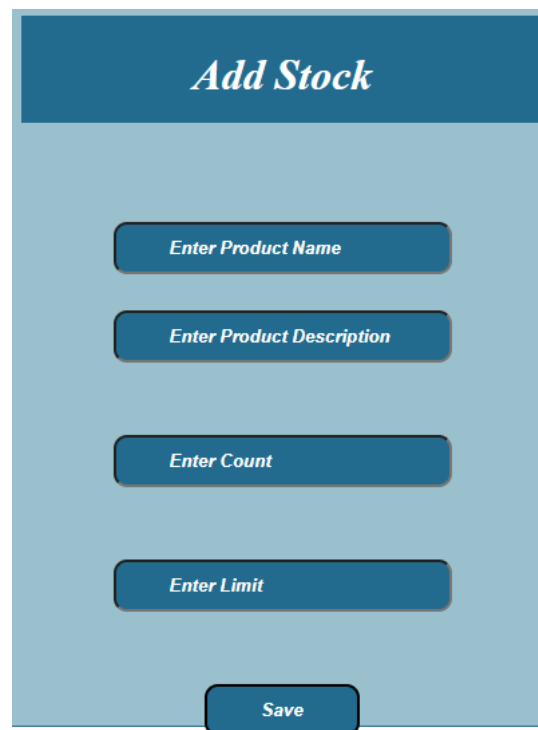
*Welcome to the Inventory App*

*Logged in successfully !* [Add Stock](#)

**Name Description OnHand**

Figure 7.3 : Index Page

### 7.4 FEATURE 4:



The Add Stock form has a dark blue header with the text "Add Stock" in white italicized font. Below the header, there are four input fields with dark blue backgrounds and white italicized text: "Enter Product Name", "Enter Product Description", "Enter Count", and "Enter Limit". At the bottom, there is a "Save" button with a dark blue background and white italicized text.

*Add Stock*

*Enter Product Name*

*Enter Product Description*

*Enter Count*

*Enter Limit*

*Save*

## Add Stock

## Index

Hi raagavan!!

Welcome to the Inventory App

Stock has been added!

Name	Description	OnHand
Apple	Apple Iphone pro	10 <input type="button" value="Add Sale"/>

## Add Stock

## Index

Hi raagavan!!

Welcome to the Inventory App

Stock has been added!

Name	Description	OnHand
Apple	Apple Iphone pro	10 <input type="button" value="Add Sale"/>
LG	LG TV	20 <input type="button" value="Add Sale"/>

Figure 7.4 : Stock Adding Page



## 7.5 FEATURE 5:

Add Sale

Product Name: Apple

Enter

Save

Add Sale

Product Name: Apple

2

Save

Index

Hi raagavan!! [Logout](#)

Welcome to the Inventory App

Stock onhand decreased ! [Add Stock](#)

Name	Description	OnHand	
Apple	Apple Iphone pro 8	8	<a href="#">Add Sale</a>
LG	LG TV	40	<a href="#">Add Sale</a>

Figure 7.5 : Add Sale Page

## 7.6 FEATURE 6:

### Add Stock

### Index

*Hi raagavan!!*

Welcome to the Inventory App

Stock onhand increased !

Name	Description	OnHand	
Apple	Apple Iphone pro	16	<input type="button" value="Add Sale"/>
LG	LG TV	40	<input type="button" value="Add Sale"/>

### Add Sale

*Product Name: Apple*

### Index

*Hi raagavan!!*

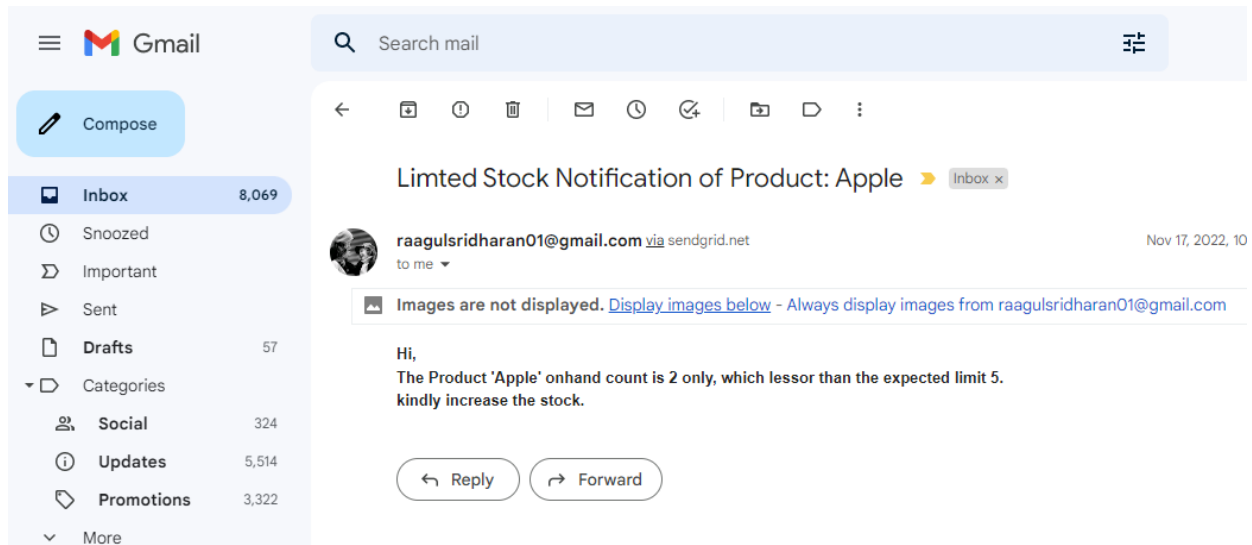
Welcome to the Inventory App

Stock onhand decreased !

Name	Description	OnHand	
Apple	Apple Iphone pro	2	<input type="button" value="Add Sale"/>
LG	LG TV	40	<input type="button" value="Add Sale"/>

Figure 7.6 : Check Stock Page

## 7.7 FEATURE 7 :



**Figure 7.7 : Stock Alert Mail**

## CHAPTER 8

### TESTING

#### 8.1 TEST CASES :

Test cases	Result
Verify whether the user is able to see the sign in page	Positive
Verify whether the user is able to go to the sign up page	Positive
Verify whether the user is able to create a new account	Positive
Verify whether the user is able to able choose their preferred role when creation	Positive
Verify whether the user is able to login using email id and password	Positive
Verify whether the user is able to see the dashboard after login	Positive
Verify whether the user is able to view their profile after clicking view profile button	Positive
Verify whether the user is able to edit their profile info after clicking edit profile	Positive
Verify whether the user is able to create a new warehouse by choosing create warehouse	Positive
Verify whether the user is able to add new products to the warehouse	Positive
Verify whether the user is able to view the list of products in the dashboard	Positive
Verify whether the user is able to add/remove the product count	Positive
Verify whether the user receives notification mail when the product count reaches threshold	Postive
Verify whether the user is able to logout	Positive

## 8.2 USER ACCEPTANCE TESTING:

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 signup 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	<ol style="list-style-type: none"> <li>1. Enter the url and go</li> <li>2. Click the sign up link in the navigation bar.</li> <li>3. Enter valid details in the text boxes.</li> <li>4. Verify the confirmation message.</li> </ol>
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	<ol style="list-style-type: none"> <li>1. Enter the url and go</li> <li>2. Click the sign in link in the navigation bar.</li> <li>3. Verify the sign in page is visible or not.</li> </ol>
SignInPage_TC_002	UI	Sign In page	Verify the UI elements in the Sign in page	<ol style="list-style-type: none"> <li>1. Enter the url and go</li> <li>2. Click the sign in link in the navigation bar.</li> <li>3. Verify the below mentioned ui elements: <ol style="list-style-type: none"> <li>a. email text box.</li> <li>b. password text box.</li> <li>c. sign in button</li> </ol> </li> </ol>
SignInPage_TC_003	Functional	Sign In page	Verify the user is able to login into the application by providing valid details	<ol style="list-style-type: none"> <li>1. Enter the url and go</li> <li>2. Click the sign in link in the navigation bar.</li> <li>3. Enter valid details in the text boxes.</li> <li>4. Verify the user is able to login.</li> </ol>
DashboardPage_TC_001	Functional	Dashboard	Verify whether the user is able to see the list of products stored in the warehouse	<ol style="list-style-type: none"> <li>1. Enter the url and go</li> <li>2. Verify whether products are visible or not.</li> </ol>

DashboardPage_TC_002	UI	Dashboard	Verify the UI elements in the dashboard page	<p>. Enter the url and go</p> <p>2. Verify the below mentioned ui elements:</p> <p>a. A navbar</p> <p>b. list of table each representing a warehouse location</p> <p>c. view profile button</p> <p>d. add products button</p> <p>e. logout button</p> <p>f. add warehouses button</p>
EditProfilePage_TC_001	Functional	Edit profile page	Verify the user is able to change user details by providing valid details	<p>1. Enter the url and go</p> <p>2. Enter valid details in the text boxes.</p>
				<p>3. Click the update button.</p> <p>4. Verify whether the user information is updated successfully.</p>
EditProfilePage_TC_002	UI	Edit profile page	Verify the UI elements in the edit profile page	<p>1. Enter the url and go</p> <p>2. Click the edit profile button in the navigation bar.</p> <p>3. Verify the below mentioned ui elements:</p> <p>a. name text box</p> <p>b. email text box.</p> <p>c. password text box.</p> <p>d. inventory name text box.</p> <p>e. an update button</p>

AddProductForm_TC_001	Functional	Add Product page	Verify the user is able to add a product to the warehouse	<ol style="list-style-type: none"> <li>1. Enter the url and go</li> <li>2. Click the request link near the warehouse name.</li> <li>3. Enter valid details in the text boxes.</li> <li>4. Click the add button.</li> <li>5. Verify whether the product is added successfully.</li> </ol>
AddWarehouse_TC_001	Functional	Add warehouse page	Verify the user is able to add a warehouse	<ol style="list-style-type: none"> <li>1. Enter the url and go</li> <li>2. Go to add warehouse page.</li> <li>3. Enter the details and click add button.</li> </ol>
Notification_TC_001	Functional	Dashboard	Verify whether the user gets email notification when the product count reached threshold	<ol style="list-style-type: none"> <li>1. Enter the url and go</li> <li>2. Go to the dashboard.</li> <li>3. Remove products so that the product count reaches below threshold level.</li> </ol>
Logout_TC_001	Functional	Dashboard	Verify the user is able to logout	<ol style="list-style-type: none"> <li>1. Enter the url and go</li> <li>2. Click the logout button</li> </ol>



## **CHAPTER 9**

### **RESULTS**

#### **9.1 PERFORMANCE METRICS:**

- |                               |            |
|-------------------------------|------------|
| 1. Hours worked               | : 47 hours |
| 2. Stick to Timelines         | : 100%     |
| 3. Consistency of the product | : 78%      |
| 4. Efficiency of the product  | : 85%      |
| 5. Quality of the product     | : 89%      |

## **CHAPTER 10**

### **ADVANTAGES AND DISADVANTAGES**

#### **ADVANTAGES**

- Measured pay for each use
- Effective management
- More convenient to access from anywhere
- Can include multiple stocks

#### **DISADVANTAGES**

- Only operates when the internet is on.
- On the client side machine, latency can be seen.
- Requires upkeep to maintain stability

## **CHAPTER 11**

### **CONCLUSION**

Therefore, the major goal of this project was to create an easy-to-use management system software for merchants so that they could easily keep track of their inventory. With that said, we have already completed our project. It was a terrific experience and we learned a lot of new technologies while putting this project into practice.

## **CHAPTER 12**

### **FUTURE SCOPE**

#### **12.1 FUTURE SCOPE :**

- Successful businesses will consider inventory as a strategic asset as opposed to an annoying expense or a necessary evil.
- Effective inventory management will depend on collaboration with supply chain partners and a holistic approach to supply chain management. Globalization's characteristics will alter, having a significant impact on decisions on how to deploy inventories.
- The main drivers for changing supply chain and inventory strategy will be an increased emphasis on supply chain security and worries about the quality of inventory itself.
- An inventory system can be used to value goods, track inventory changes, and prepare for future inventory levels, among other things. At the end of each period, the inventory value provides

## CHAPTER 13

### APPENDIX

#### 13.1 SOURCE CODE :

**app.py :**

```
import re
from flask import Flask, render_template, request, redirect, url_for, session
from pymongo_get_database import get_database
import logging
logging.basicConfig(filename='record.log', level=logging.DEBUG, format=f'%(asctime)s %(levelname)s %(name)s %(threadName)s : %(message)s')
dbname = get_database()

import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

app = Flask(__name__)

app.secret_key = 'your secret key'
print("test")

@app.route('/')
@app.route('/login', methods=['GET', 'POST'])
def login():
    msg = "
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
        username = request.form['username']
        password = request.form['password']
        users = dbname["Users"]
```

```

account = users.find_one({"username": username, "password": password})
if account:
    session['loggedin'] = True
    session['id'] = str(account['_id'])
    session['username'] = account['username']
    session['email'] = account['email']
    msg = 'Logged in successfully !'
    stocks = dbname["Stocks"].find({ "email": session['email']})
    return render_template('index.html', msg=msg, stocks=stocks)
else:
    msg = 'Incorrect username / password !'
return render_template('login.html', msg = msg)

```

```
@app.route('/logout')
```

```
def logout():
```

```

    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    session.pop('email', None)
    return redirect(url_for('login'))

```

```
@app.route('/register', methods=['GET', 'POST'])
```

```
def register():
```

```
    msg = "
```

```
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form and 'email'
```

```
in request.form :
```

```

    username = request.form['username']
    password = request.form['password']
    email = request.form['email']
    users = dbname["Users"]
    account = users.find_one({"username" : username})
    app.logger.info('account:%s', account)

```

```

if account:
    msg = 'Account already exists !'
elif not re.match(r'^@]+@[^@]+\.[^@]+', email):
    msg = 'Invalid email address !'
elif not re.match(r'[A-Za-z0-9]+', username):
    msg = 'Username must contain only characters and numbers !'
elif not username or not password or not email:
    msg = 'Please fill out the form !'
else:
    users.insert_one({"username":username, "password": password, "email": email})
    msg = 'You have successfully registered !'

elif request.method == 'POST':
    msg = 'Please fill out the form !'

return render_template('register.html', msg = msg)

```

```
@app.route('/add_stock', methods=['GET', 'POST'])
```

```
def add_stock():
```

```
    msg = "
```

```
    stocks = dbname["Stocks"]
```

```
    if request.method == 'POST' and 'name' in request.form and 'description' in request.form and 'onhand' in request.form :
```

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

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

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

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

```
        stock = stocks.find_one({"name" : name, "email": session["email"]})
```

```
        app.logger.info('stock:%s', stock)
```

```
        if stock:
```

```
            stocks.update_one({"name" : name, "email": session["email"]}, {"$set": { "onhand":
```

```
stock["onhand"] + onhand }})
```

```
            msg = 'Stock onhand increased !'
```

```
            return render_template('index.html', msg = msg, stocks = stocks.find({"email":
```

```

session["email"])))
    elif onhand <= 0:
        msg = 'Invalid on hand count !'
    elif limit <= 0:
        msg = 'Invalid limit count !'
    elif not name or not description or not onhand:
        msg = 'Please fill out the form !'
    else:
        stocks.insert_one({"name":name, "description": description, "onhand": onhand, "limit":
limit, "email": session["email"]})
        msg = 'Stock has been added!'
        return render_template('index.html', msg = msg, stocks = stocks.find({"email":
session["email"]}))
    elif request.method == 'POST':
        msg = 'Please fill out the form !'
        return render_template('add_stock.html', msg = msg)

@app.route('/add_sale', methods =['GET', 'POST'])
def add_sale():
    msg = "
    stocks = dbname["Stocks"]
    if request.method == 'POST' and 'name' in request.form and 'count' in request.form:
        name = request.form['name']
        count = int(request.form['count'])
        stock = stocks.find_one({"name" : name, "email": session["email"]})
        app.logger.info('stock:%s', stock)
        if stock:
            onhand = stock["onhand"] - count
            stocks.update_one({"name" : name, "email": session["email"]}, {"$set": { "onhand":
onhand } })

            msg = 'Stock onhand decreased !'
            if onhand < stock["limit"]:

```



```

        send_mail(stock["email"],r"Limited Stock Notification of Product: "+name,
"<strong>Hi, <br> The Product '"+ name +"' onhand count is { } only, which lessor than the expected limit { }.
<br> kindly increase the stock.</strong>".format(onhand, stock["limit"]))

        return render_template('index.html', msg = msg, stocks = stocks.find({"email":
session["email"]})))

        elif count <= 0:

            msg = 'Invalid Sale count !'

        elif not name or not count:

            msg = 'Please fill out the form !'

        else:

            sales = dbname["Sales"]

            sales.insert_one({"name":name, "count": count})

            msg = 'Sale has been added!'

            return render_template('index.html', msg = msg, stocks = stocks.find({"email":
session["email"]})))

        elif request.method == 'POST':

            msg = 'Please fill out the form !'

        elif request.method == 'GET' and request.args.get("name"):

            app.logger.info('argument name:%s', request.args.get("name"))

            stock = stocks.find_one({"name" : request.args.get("name")})

            return render_template('add_sale.html', stock = stock)

def send_mail(to_emails,subject,html_content):

    message = Mail(

        from_email='raagulsridharan01@gmail.com',

        to_emails=to_emails,

        subject=subject,

        html_content=html_content)

    try:

        sg =

SendGridAPIClient('SG.RSAT_rFmSnqYuhmOINYHcw.h3yLAfTtXONesb2gYbxXhBLyuNBMPPP2qOM2v

SnM_tw')

```

```

        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e.message)

```

## login.html

```

<html>
    <head>
        <meta charset="UTF-8">
        <title> Login </title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    </head>
    <body>
        <div align="center">
            <div align="center" class="border">
                <div class="header">
                    <h1 class="word">Login</h1>
                </div></br></br></br>
                <h2 class="word">
                    <form action="{{ url_for('login') }}" method="post">
                        <div class="msg">{{ msg }}</div>
                        <input id="username" name="username" type="text" placeholder="Enter
Your Username" class="textbox"/></br></br>
                        <input id="password" name="password" type="password"
placeholder="Enter Your Password" class="textbox"/></br></br></br>
                        <input type="submit" class="btn" value="Sign In"></br></br>
                    </form>
                </h2>
                <p class="bottom">Don't have an account? <a class="bottom"

```

href="{{ url\_for('register') }}"> Sign Up here</a></p>

</div>

</div>

</body>

</html>

## register.html

<html>

<head>

<meta charset="UTF-8">

<title> Register </title>

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

</head>

<body>

<div align="center">

<div align="center" class="border">

<div class="header">

<h1 class="word">Register</h1>

</div><br><br><br>

<h2 class="word">

<form action="{{ url\_for('register') }}" method="post">

<div class="msg">{{ msg }}</div>

<input id="username" name="username" type="text" placeholder="Enter Your Username" class="textbox"/><br><br>

<input id="password" name="password" type="password" placeholder="Enter Your Password" class="textbox"/><br><br>

<input id="email" name="email" type="text" placeholder="Enter Your Email ID" class="textbox"/><br><br>

<input type="submit" class="btn" value="Sign Up"><br>

</form>

</h2>

<p class="bottom">Already have an account? <a class="bottom"

```

href="{{ url_for('login') }}"> Sign In here</a></p>
    </div>
    </div>
</body>
</html>

```

## index.html

```

<html>
    <head>
        <meta charset="UTF-8">
        <title> Index </title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

    </head>
    <body>
        <div align="center">
            <div align="center" class="border">
                <div class="header">
                    <h1 class="word">Index</h1>

                </div>

                <h3 class="bottom">
                    Hi {{ session.username }}!!

                <button><a href="{{ url_for('logout') }}" class="btn-sm">Logout</a></button></br></br>
Welcome to the Inventory App

                </h3>

            </div>{{ msg }}
            <a href="{{ url_for('add_stock') }}" class="btn">Add Stock</a></br>
        </br>
        <table>
            <tr>
                <!-- <th>Stock#</th> -->

                <th>Name</th>

```

```

        <th>Description</th>
        <th>OnHand</th>
    </tr>
    {% for stock in stocks %}
    <tr>
        <!-- <td></td> -->
        <td>{{ stock.name }}</td>
        <td>{{ stock.description }}</td>
        <td>{{ stock.onhand }}</td>
        <td><button><a href="{{ url_for('add_sale', name=stock.name) }}">Add
Sale</a></button></td>
    </tr>
    {% endfor %}
</table>
<br><br>
</div>
</div>
</body>
</html>

```

## addstock.html

```

<html>
    <head>
        <meta charset="UTF-8">
        <title> Add Stock </title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    </head>
    <body>
        <div align="center">
            <div align="center" class="border">
                <div class="header">
                    <h1 class="word">Add Stock</h1>

```

```

</div></br></br></br>
<h2 class="word">
    <form action="{{ url_for('add_stock') }}" method="post">
        <div class="msg">{{ msg }}</div>
            <input id="name" name="name" type="text" placeholder="Enter Product
Name" class="textbox"/></br></br>
                <input id="description" name="description" type="text"
placeholder="Enter Product Description" class="textbox"/></br></br></br>
                    <input id="onhand" name="onhand" type="number" placeholder="Enter
Count" class="textbox"/></br></br></br>
                        <input id="limit" name="limit" type="number" placeholder="Enter
Limit" class="textbox"/></br></br></br>
                            <input type="submit" class="btn" value="Save"></br></br>
                                </form>
                                    </h2>
                                        </div>
                                            </div>
                                                </body>
                                                    </html>

```

## addsale.html

```

<html>
    <head>
        <meta charset="UTF-8">
        <title> Add Sale </title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    </head>
    <body>
        <div align="center">
            <div align="center" class="border">
                <div class="header">
                    <h1 class="word">Add Sale</h1>

```

```

</div></br></br>
<h2 class="word">
    <form action="{{ url_for('add_sale') }}" method="post">
        <div class="msg">{{ msg }}</div>
            <label>Product Name: {{ stock.name }}</label></br></br>
            <input id="name" name="name" type="hidden"
value="{{ stock.name }}" placeholder="Enter Product Name" class="textbox"/></br></br>
            <input id="count" name="count" type="number" min="1 "
max="{{ stock.onhand }}" placeholder="Enter Count" class="textbox"/></br></br></br>
            <input type="submit" class="btn" value="Save"></br></br>
        </form>
    </h2>
</div>
</div>
</body>
</html>

```

### style.css

```

.header{
    padding: 5px 120px;
    width: 150px;
    height: 70px;
    background-color: #236B8E;
}

.border{
    padding: 80px 50px;
    width: 400px;
    height: 450px;
    border: 1px solid #236B8E;
    border-radius: 0px;
}

```

```
background-color: #9AC0CD;  
}
```

```
.btn {  
padding: 10px 40px;  
background-color: #236B8E;  
color: #FFFFFF;  
font-style: oblique;  
font-weight: bold;  
border-radius: 10px;  
}
```

```
.textbox{  
padding: 10px 40px;  
background-color: #236B8E;  
text-color: #FFFFFF;  
border-radius: 10px;  
}
```

```
::placeholder {  
color: #FFFFFF;  
opacity: 1;  
font-style: oblique;  
font-weight: bold;  
}
```

```
.word{  
color: #FFFFFF;  
font-style: oblique;  
font-weight: bold;  
}
```



```
.bottom{
  color: #236B8E;
  font-style: oblique;
  font-weight: bold;
}
```

## **Dockerfile**

```
FROM python:alpine3.11
LABEL maintainer="Raagul Sridharan, raagulsridharan01@gmail.com"
WORKDIR /app
COPY . /app
RUN pip install --no-cache-dir -r requirements.txt
EXPOSE 5000
CMD [ "python", "-m", "flask", "--debug", "run"]
```

## **deployment.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: inventory-deployment
  labels:
    app: inventory
spec:
  replicas: 1
  selector:
    matchLabels:
      app: inventory
  template:
    metadata:
```

```
labels:
  app: inventory
spec:
  containers:
  - name: inventory
    image: icr.io/inventoryns/inventory
  ports:
  - containerPort: 5000
```

### **Service.yaml**

```
apiVersion: v1
kind: Service
metadata:
  name: inventory
  labels:
    run: inventory
spec:
  ports:
  - port: 5000
    protocol: TCP
  selector:
    run: inventory
```

### **loadbalance.yaml**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/name: load-balancer-example
  name: hello-world
spec:
```

```
replicas: 1
selector:
  matchLabels:
    app.kubernetes.io/name: load-balancer-example
template:
  metadata:
    labels:
      app.kubernetes.io/name: load-balancer-example
  spec:
    containers:
      - image: icr.io/inventoryns/inventory
        name: hello-world
        ports:
          - containerPort: 5000
```

## 13.2 GITHUB LINK :

<https://github.com/IBM-EPBL/IBM-Project-29964-1660135972>