

INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

Team ID	PNT2022TMID15205
Project Name	INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

TABLE OF CONTENTS

- 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 & Estimation
 - 6.2 Sprint Delivery Schedule
 - 6.3 Reports from JIRA
- 7. CODING & SOLUTIONING**
 - 7.1 Feature 1
 - 7.2 Feature 2
 - 7.3 Database Schema (if Applicable)
- 8. TESTING**
 - 8.1 Test Cases
 - 8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1. INTRODUCTION :

1.1 Project Overview :

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.

1.2 Purpose :

The emergence of the internet has been the greatest technological advancement after the industrial age. From the recent studies on internet penetration and usage in India it has been concluded that many Indians are using the internet to pay bills, purchase products online apart from regular surfing, checking e-mail and socialising on multiple social networks. The number is expected to grow from time as the internet becomes more pervasive and secure. The rise of the internet has created opportunities for entrepreneurs, and has changed the business landscape of e-commerce.

Managing inventory to create higher inventory turnover and just in time delivery practices is one of the most important processes for online retailers. Flexible systems that respond to customer demand and inventory uncertainties are most important in e-commerce.

2. LITERATURE SURVEY

2.1 Existing problem :

- Lack of Inventory Visibility. ...
- Inefficient Inventory Management Process or Software. ...
- Tracking Obsolete Material. ...
- Identifying Incorrectly Located Materials. ...
- Keeping up with Overstocks. ...
- Managing Inventory Waste & Defects. ...
- Lack of Centralized Inventory Hub. ...
- Changing Demand.

2.2 References :

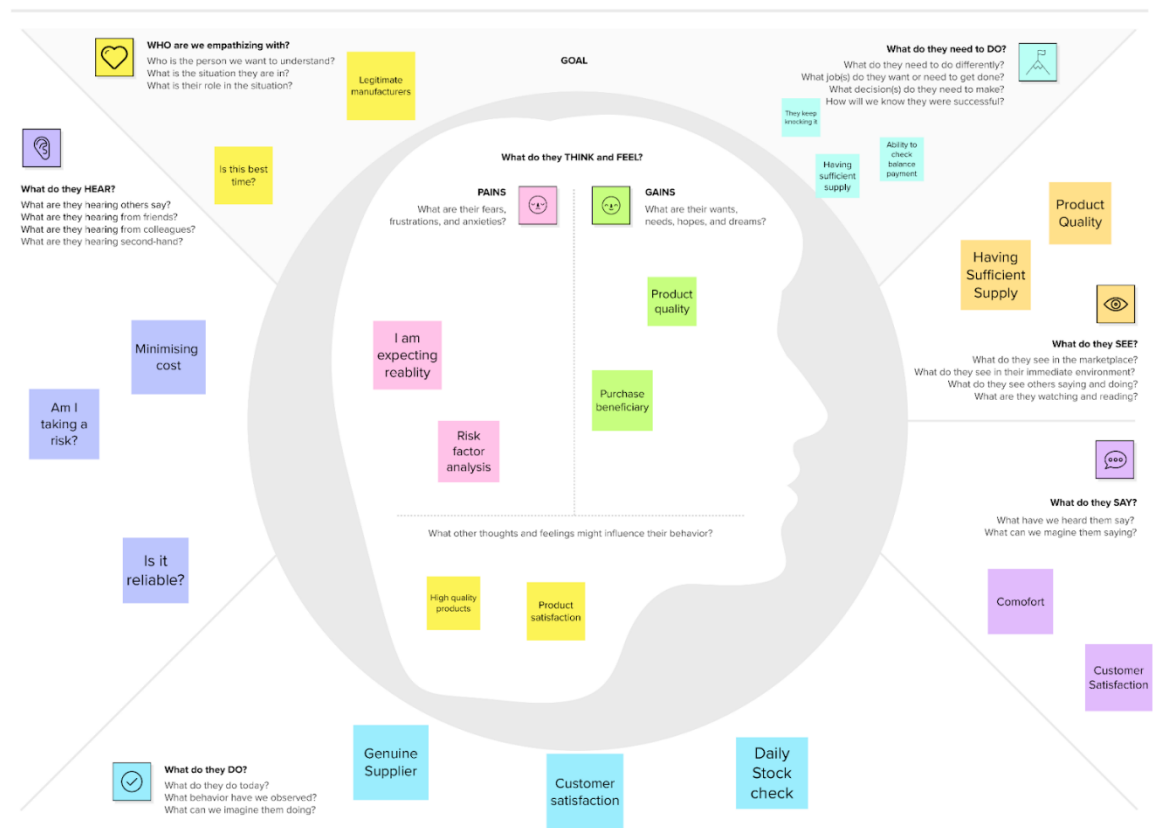
- [1] Patil, Harish, and Brig Rajiv Divekar. "Inventory management challenges for B2C e-commerce retailers." *Procedia Economics and Finance* 11 (2014): 561-571.
- [2] Eckert, Scott Grant. "Inventory management and its effects on customer satisfaction." *Journal of Business and public Policy* 1.3 (2007): 1-13.
- [3] Madadi, Alireza, Mary E. Kurz, and Jalal Ashayeri. "Multi-level inventory management decisions with transportation cost consideration." *Transportation Research Part E: Logistics and Transportation Review* 46.5 (2010): 719-734.
- [4] Zhang, Mingyang, et al. "Inventory Management of Perishable Goods with Overconfident Retailers." *Mathematics* 10.10 (2022): 1716.
- [5] Li, S. G., and X. Kuo. "The inventory management system for automobile spare parts in a central warehouse." *Expert Systems with Applications* 34.2 (2008): 1144-1153.
- [6] Ehrental, J. C. F., Dorothée Honhon, and Tom Van Woensel. "Demand seasonality in retail inventory management." *European Journal of Operational Research* 238.2 (2014): 527-539.
- [7] Bose, R., Mondal, H., Sarkar, I., & Roy, S. (2022). Design of smart inventory management system for construction sector based on IoT and cloud computing. *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, 2, 100051.
- [8] Beheshti, Hooshang M. "A decision support system for improving performance of inventory management in a supply chain network." *International Journal of Productivity and Performance Management* (2010).
- [9] Taleizadeh, Ata Allah, et al. "Stock replenishment policies for a vendor-managed inventory in a retailing system." *Journal of Retailing and Consumer Services* 55 (2020): 102137.
- [10] Ji, Jindi, et al. "Distribution network planning and inventory management in a multi-retailing supply chain." *Journal of Physics: Conference Series*. Vol. 1903. No. 1. IOP Publishing, 2021.

2.3 Problem Statement Definition :

The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized. Good planning and sales forecast before setting optimal inventory levels, appropriate inventory management requires close coordination between the areas of sales, purchasing and finance.

3. IDEATION & PROPOSED SOLUTION :

3.1 Empathy Map Canvas :



3.2 Ideation & Brainstorming

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

Template



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

 10 minutes to prepare
 1 hour to collaborate
 2-8 people recommended

[Share template feedback](#)

 **Before you collaborate**

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

 10 minutes

A Team gathering
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B Set the goal
Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

 5 minutes

PROBLEM

The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.

Key rules of brainstorming

To run an smooth and productive session

 Stay in topic.

 Encourage wild ideas.

 Defer judgment.

 Listen to others.

 Go for volume.

 If possible, be visual.

 Need some inspiration?



See a finished version of this template to kickstart your work.

[Open example](#) →

Step-2: Brainstorm, Idea Listing and Grouping :

2

Brainstorm

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

Ⓢ 10 minutes

TIP

You can select a sticky note and drag it to the panel (switch to the **Sticky Notes** panel to start drawing).

[illegible]

3

Group ideas

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

20 minutes

The diagram illustrates a business growth strategy through several stages:

- Product Inventor** (Top Left):
 - Initial state: **Small idea** (blue), **Small idea** (blue), **Small idea** (blue).
 - Development: **Small idea** (blue) leads to **Small idea** (blue) and **Small idea** (blue).
 - Expansion: **Small idea** (blue) leads to **Small idea** (blue) and **Small idea** (blue).
 - Final state: **Small idea** (blue), **Small idea** (blue), **Small idea** (blue).
- Customer Manager** (Top Right):
 - Initial state: **Small idea** (blue), **Small idea** (blue), **Small idea** (blue).
 - Development: **Small idea** (blue) leads to **Small idea** (blue) and **Small idea** (blue).
 - Expansion: **Small idea** (blue) leads to **Small idea** (blue) and **Small idea** (blue).
 - Final state: **Small idea** (blue), **Small idea** (blue), **Small idea** (blue).
- Sales Product** (Middle):
 - Initial state: **Small idea** (blue), **Small idea** (blue), **Small idea** (blue).
 - Development: **Small idea** (blue) leads to **Small idea** (blue) and **Small idea** (blue).
 - Expansion: **Small idea** (blue) leads to **Small idea** (blue) and **Small idea** (blue).
 - Final state: **Small idea** (blue), **Small idea** (blue), **Small idea** (blue).
- Account Statistics** (Bottom Left):
 - Initial state: **Small idea** (blue), **Small idea** (blue), **Small idea** (blue).
 - Development: **Small idea** (blue) leads to **Small idea** (blue) and **Small idea** (blue).
 - Expansion: **Small idea** (blue) leads to **Small idea** (blue) and **Small idea** (blue).
 - Final state: **Small idea** (blue), **Small idea** (blue), **Small idea** (blue).
- Expansion of products** (Bottom Right):
 - Initial state: **Small idea** (blue), **Small idea** (blue), **Small idea** (blue).
 - Development: **Small idea** (blue) leads to **Small idea** (blue) and **Small idea** (blue).
 - Expansion: **Small idea** (blue) leads to **Small idea** (blue) and **Small idea** (blue).
 - Final state: **Small idea** (blue), **Small idea** (blue), **Small idea** (blue).



Step-3: Idea Prioritization :

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes

Importance
If lots of things could get done without any difficulty or cost, which would bring the most positive impact?

Feasibility
Degree of how important, which makes one more feasible than others? (Cost, time, complexity, etc.)

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

- Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.
- Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

- Strategy blueprint**
Define the components of a new idea or strategy.
[Open the template →](#)
- Customer experience journey map**
Understand customer needs, painpoints, and obstacles for an experience.
[Open the template →](#)
- Strengths, weaknesses, opportunities & threats**
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
[Open the template →](#)

[Share template feedback](#)

Entire pitch :

[illegible]

3.3 Proposed Solution :

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized. Good planning and sales forecast before setting optimal inventory levels, appropriate inventory management requires close coordination between the areas of sales, purchasing and finance.
	Idea / Solution description	No matter the size of your business, employing some of these common inventory management techniques can be a great way to take control of your stock. Here are a few to consider: <ol style="list-style-type: none"> 1. <u>Just-in-time (JIT) inventory.</u> 2. <u>ABC inventory analysis.</u> 3. Dropshipping. 4. Bulk shipments 5. Consignment 6. Cross-docking 7. Cycle counting.
	Novelty / Uniqueness	One of the biggest features of an inventory management system for small businesses is the ability to tell you how much you have left of each product. However, some systems can also help you predict future inventory needs. Data analysis uses information from your inventory cycle to help you make better business decisions. For example, it can track your historic sales to predict when you will get a surge of purchases, so you can buy extra inventory and <u>prepare better for your peak season.</u>
0.	Social Impact / Customer Satisfaction	<ul style="list-style-type: none"> • Delivering great customer service and a positive customer experience every time begins long before a sales transaction takes place. By developing smarter practices and procedures to increase inventory accuracy you can improve lead times, save money and ensure greater satisfaction and consumer loyalty. • Optimal inventory management is all about having what the customer wants, when and where they want it, while exceptional customer service is about meeting and exceeding consumer expectations. Having the right inventory control system will enable you to deliver on both.
0.	Business Model (Revenue Model)	<ul style="list-style-type: none"> • Point of sale integration • Inventory catalog • Automated reordering • E-commerce integrations • Product cost analysis • Barcoding

		1.Forecasting
0.	Scalability of the Solution	Inventory management platforms include demand forecasting tools. This feature integrates with accounting and sales data to help you predict demand and schedule orders based on shifting customer preferences, material availability or seasonal trends.

3.4 Problem Solution fit :

<p>Define CS, fit into CC</p> <p>1. CUSTOMER SEGMENT(S) Who is your customer? i.e. working parents of 0-5 y.o. kids</p> <p>CS</p> <p>Manufacturers</p>	<p>6. CUSTOMER CONSTRAINTS</p> <p>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.</p> <p>CC</p> <ul style="list-style-type: none"> Machine capacity Workforce capacity Inventory investment Storage space or the total number of orders placed. 	<p>5. AVAILABLE SOLUTIONS Which solutions are available to the customers when they face the problem</p> <p>AS</p> <p>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 notetaking</p> <p>You can take advantage of bulk savings</p> <p>You need space for your products</p>
<p>Focus on J&P, tap into BE, understand RC</p> <p>2. JOBS-TO-BE-DONE / PROBLEMS Which job-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.</p> <p>J&P</p> <ul style="list-style-type: none"> Inconsistent Tracking Insufficient Order Management Increasing Competition Evolving Packaging 	<p>9. PROBLEM ROOT CAUSE 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.</p> <p>RC</p> <ul style="list-style-type: none"> Poor Production Planning Lack of Expertise Inefficient Processes 	<p>7. BEHAVIOUR What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)</p> <p>BE</p> <p>Which stock sells well and which doesn't, by location and sales channel.</p> <p>How changing seasons affect sales</p>

Identify Strong TR & EM	<div><div>3. TRIGGERS</div><div>TR</div></div> <p>What triggers customers to act? i.e. seeing their robotic installing solar panels, reading about a more efficient solution in the news.</p> <p>To manage changing trends, such as packaging initiatives to reduce plastic waste. Categorize stock by packaging type, dimensions and product. Use this information to control shipping costs and storage location better.</p>	<div><div>10. YOUR SOLUTION</div><div>SL</div></div> <p>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.</p> <p>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 robotic.</p> <ul style="list-style-type: none">• Centralized Tracking• Stock Auditing• Add Imagery• Safety Stock• Multi-Location Warehousing• Reduce Human Error• Optimize Space• Leverage Lead Times	<div><div>8. CHANNELS OF BEHAVIOUR</div><div>CH</div></div> <div><div>8.1 ONLINE</div><p>What kind of actions do customers take online? Extract online channels from #7</p></div> <div><div>8.2 OFFLINE</div><p>What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.</p></div>	
	<div><div>4. EMOTIONS: BEFORE / AFTER</div><div>EM</div></div> <p>How do customers feel when they face a problem or a job and afterwards?</p> <p>i.e. lost, insecure > confident, in control - use it in your communication strategy & design.</p> <p>Emotions :</p> <p>Before: Complexed</p> <p>After : Good Satisfaction</p>			<p>Online:</p> <p>Shopping and shipping</p> <p>Offline:</p> <p>Demanding and less moving product to kept in front section.</p>

4. REQUIREMENT ANALYSIS :

4.1 Functional requirement :

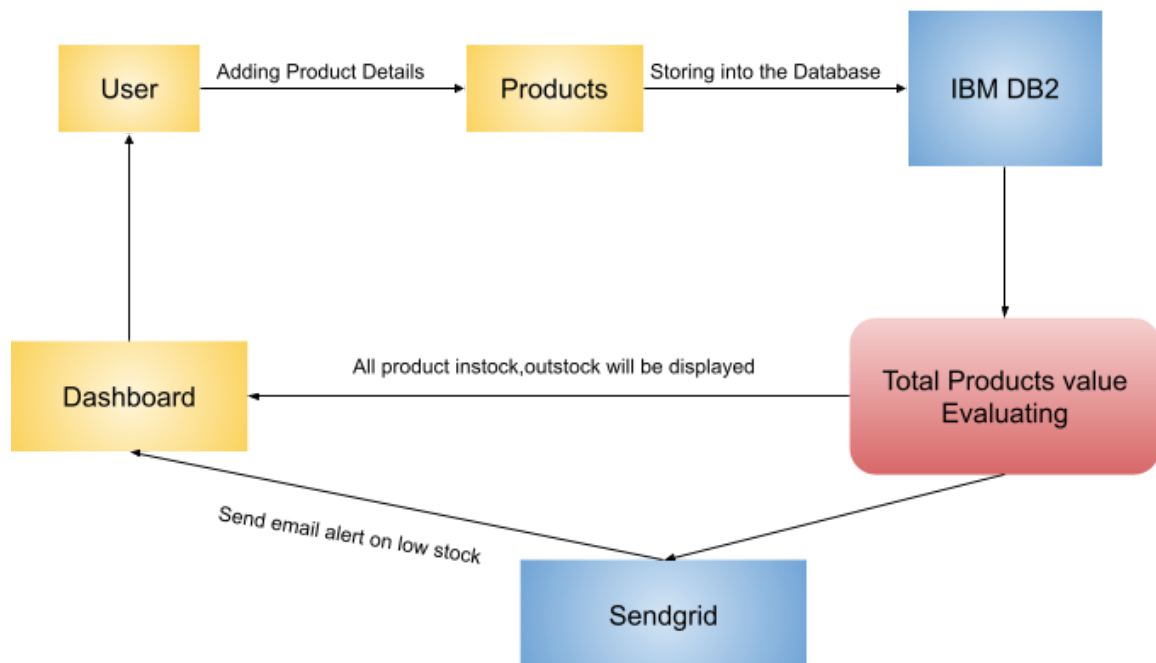
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email,OTP,Mobile-number
FR-3	Add,delete and update the stocks	Add or delete or update the stocks using the functions available in the inventory
FR-4	Indication	Receive an indication when an item I'm trying to buy is too much than the stock available
FR-5	Location	Able to make purchase from a particular location
FR-6	Re-order	Able to order the products after a certain period of time

4.2 Non-Functional requirements :

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Stocks/products are easily able to track and manage
NFR-2	Security	By providing one-time-password and 2-step authentication
NFR-3	Reliability	More number of stocks can be easily managed and retrieved
NFR-4	Performance	Website navigating and loading the content in few seconds.
NFR-5	Availability	Retailers, business and traders of all kind can make use of inventory management
NFR-6	Scalability	From low-scale retailer to high-scale retailer can very well make use of inventory management

5. PROJECT DESIGN :

5.1 Data Flow Diagrams :



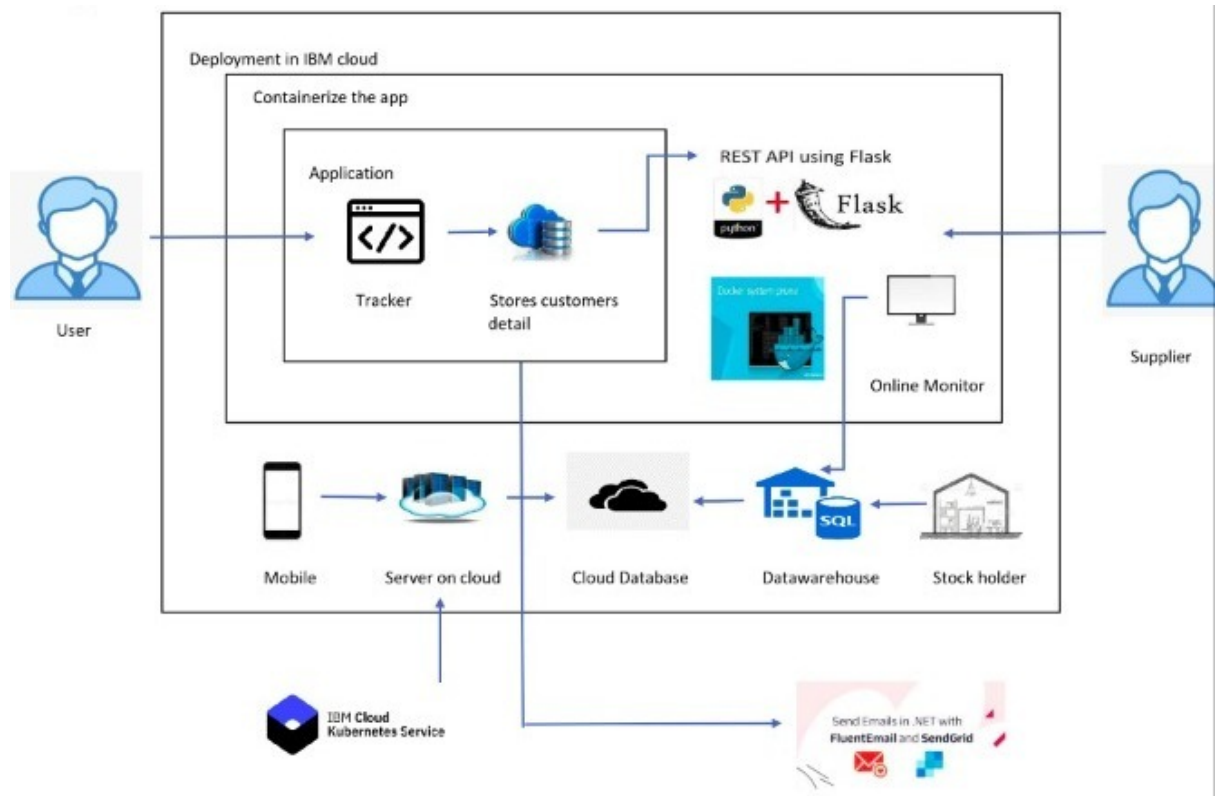
5.2 Solution & Technical Architecture :

Proposed Solutions :

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized. Good planning and sales forecast before setting optimal inventory levels, appropriate inventory management requires close coordination between the areas of sales, purchasing and finance.
	Idea / Solution description	No matter the size of your business, employing some of these common inventory management techniques can be a great way to take control of your stock. Here are a few to consider: <ol style="list-style-type: none">1. <u>Just-in-time (JIT) inventory.</u>2. <u>ABC inventory analysis.</u>3. Dropshipping.4. Bulk shipments5. Consignment6. Cross-docking7. Cycle counting.

	Novelty / Uniqueness	One of the biggest features of an inventory management system for small businesses is the ability to tell you how much you have left of each product. However, some systems can also help you predict future inventory needs. Data analysis uses information from your inventory cycle to help you make better business decisions. For example, it can track your historic sales to predict when you will get a surge of purchases, so you can buy extra inventory and <u>prepare better for your peak season.</u>
0.	Social Impact / Customer Satisfaction	<ul style="list-style-type: none"> • Delivering great customer service and a positive customer experience every time begins long before a sales transaction takes place. By developing smarter practices and procedures to increase inventory accuracy you can improve lead times, save money and ensure greater satisfaction and consumer loyalty. • Optimal inventory management is all about having what the customer wants, when and where they want it, while exceptional customer service is about meeting and exceeding consumer expectations. Having the right inventory control system will enable you to deliver on both.
0.	Business Model (Revenue Model)	<ul style="list-style-type: none"> • Point of sale integration • Inventory catalog • Automated reordering • E-commerce integrations • Product cost analysis • Barcoding • Forecasting
0.	Scalability of the Solution	Inventory management platforms include demand forecasting tools. This feature integrates with accounting and sales data to help you predict demand and schedule orders based on shifting customer preferences, material availability or seasonal trends.

Technical Architecture :



5.3 User Stories :

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Retailer	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	Medium	Sprint-1
	Login	USN-3	As a user, I can log into the application by entering email & password	I can access my account / dashboard	High	Sprint-1
	Items	USN-5	As a user, I can add the items.	I can create a new type of item	High	Sprint-2
		USN-6	As a user, I can see the items	I can be able to see the items that can be	Low	Sprint-2

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
				added to the inventory		
	Inventory	USN-7	As a user, I can add the items to inventory.	I can add items to the inventory with quantity	High	Sprint-2
		USN-8	As a user, I can see the items in the inventory.	I can see the inventory items with quantity	Low	Sprint-2
	Indication	USN-9	As a user, I can be able to receive indication	I receive a notification when the stock running low	High	Sprint-3
	Location	USN-10	As a user, I can be able to see items from a particular store location	I can be able to make purchase from a particular location	Medium	Sprint-3
		USN-11	As a user, I can add a new location of my store	I can be able to add new store locations	Medium	Sprint - 3
Customer	Purchase	USN -12	As a customer, I can be able to purchase good from the particular location of the store	I can able to purchase from the store	High	Sprint - 4
Retailer & Customer	Deployment	USN-13	As a user, I can access the software in the web	I can access the software in web	High	Sprint -4

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	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.	High	Split between all 4
Sprint 1		USN-2	As a user, I will receive confirmation email once I have registered for the application	Medium	Split between all 4
Sprint 1	Login	USN-3	As a user, I can log into the application by entering email & password	High	Lakshmikant & Kiran Shrinivaas
Sprint 2	Items	USN-5	As a user, I can add the items.	High	Mukesh Varman
Sprint 2		USN-6	As a user, I can see the items	Low	Lakshmikant

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Priority	Team Members
Sprint 2	Inventory	USN-7	As a user, I can add the items to inventory.	High	Maheshkumar
Sprint 2		USN-8	As a user, I can see the items in the inventory.	Low	Kiran Shrinivaas
Sprint 3	Indication	USN-9	As a user, I can be able to receive indication	High	Maheshkumar
Sprint 3	Location	USN-10	As a user, I can be able to see items from a particular store location	Medium	Mukesh Varman
Sprint 3		USN-11	As a user, I can add a new location of my store	Medium	Lakshmikant
Sprint 4	Purchase	USN -12	As a customer, I can be able to purchase good from the particular location of the store	High	Kiran Shrinivaas
Sprint 4	Deployment	USN-13	As a user, I can access the software in the web	High	Maheshkumar

Estimation :

Sprint	Total Story Points	Duration	Average Velocity
Sprint 1	20	7 Days	$20 / 7 = 2.85$
Sprint 2	20	7 Days	$20 / 7 = 2.85$
Sprint 3	20	7 Days	$20 / 7 = 2.85$
Sprint 4	20	7 Days	$20 / 7 = 2.85$
Total	80	28	$80 / 28 = 2.85$

6.2 Sprint Delivery Schedule :

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	7 Days	24 Oct 2022	30 Oct 2022	20	30 Oct 2022
Sprint-2	20	7 Days	31 Oct 2022	06 Nov 2022	20	07 Nov 2022
Sprint-3	20	7 Days	07 Nov 2022	14 Nov 2022	20	14 Nov 2022
Sprint-4	20	7 Days	14 Nov 2022	21 Nov 2022	20	21 Nov 2022

6.3 Reports from JIRA :

Project Planning :

Burndown Chart:

Projects / Inventory Management System for Retailers / Reports

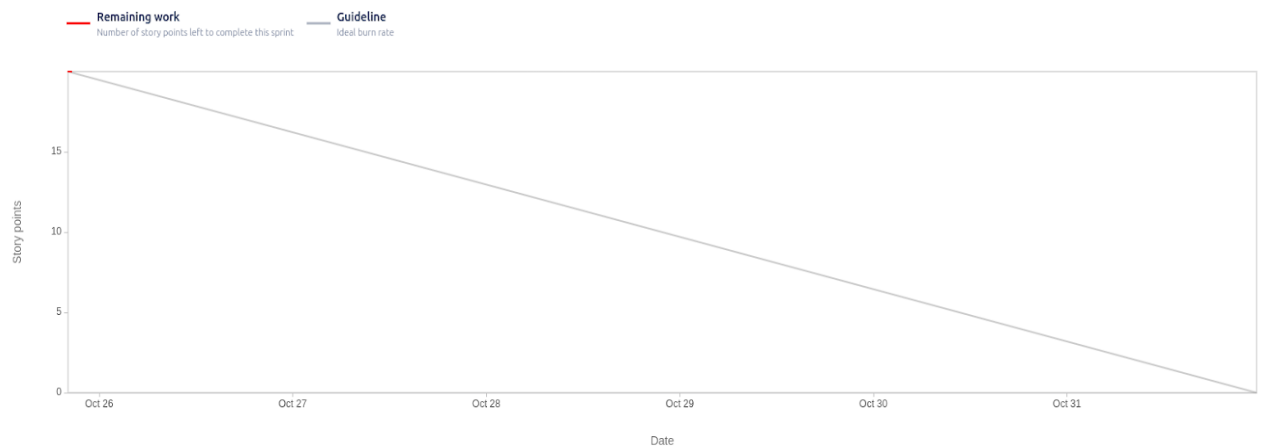
Sprint burndown chart

[How to read this report](#)

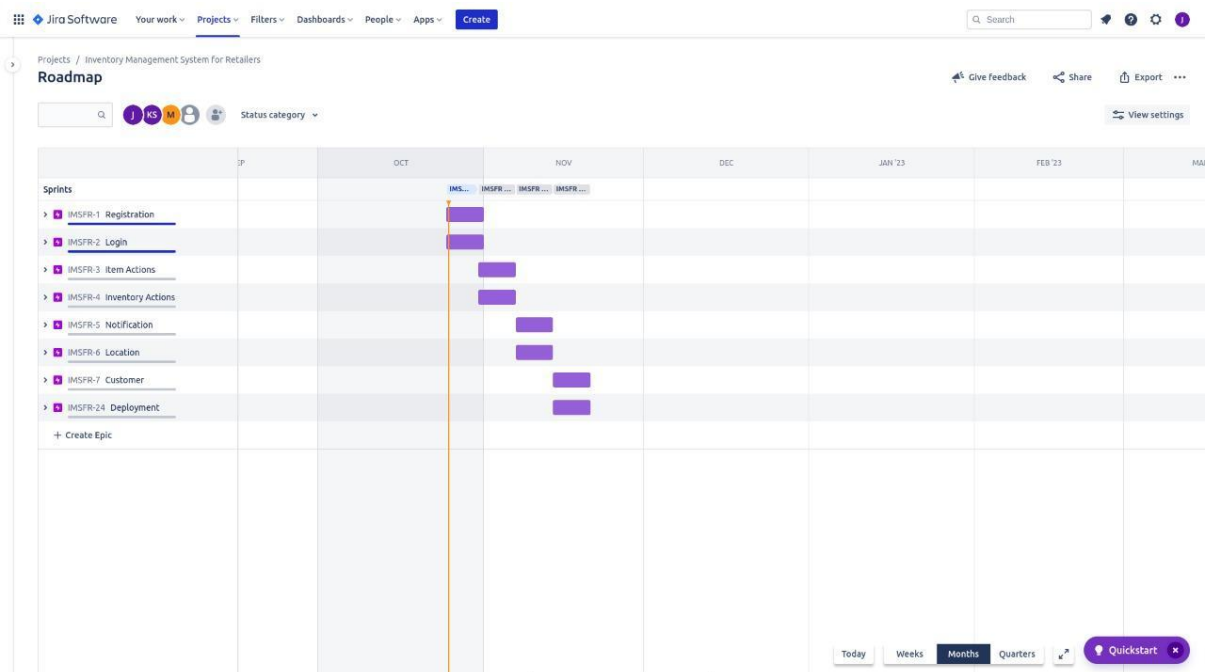
Sprint
IMSFR Sprint 1

Estimation field
Story points

Date - October 25th, 2022 - October 31st, 2022



Roadmap:



Backlog :

Projects / Inventory Management System for Retailers

Backlog

Search: [] Filter: [J] [KS] [M] [] Epic: [v] Insights [i]

IMSFR Sprint 1 25 Oct - 31 Oct (2 issues) Complete sprint [x]

- IMSFR-8 Retailer Registration REGISTRATION 10 IN PROGRESS [i]
- IMSFR-9 Retailer Login LOGIN 10 IN PROGRESS [i]

+ Create issue

IMSFR Sprint 2 31 Oct - 6 Nov (4 issues) Start sprint [x]

- IMSFR-10 Adding new item type ITEM ACTIONS 5 TO DO [M]
- IMSFR-11 Displaying item types ITEM ACTIONS 5 TO DO [i]
- IMSFR-12 Adding item to the inventory INVENTORY ACTIONS 5 TO DO [i]
- IMSFR-13 Displaying items of the inventory INVENTORY ACTIONS 5 TO DO [i]

+ Create issue

IMSFR Sprint 3 7 Nov - 13 Nov (4 issues) Start sprint [x]

- IMSFR-14 Setting threshold for the inventory items NOTIFICATION 5 TO DO [M]
- IMSFR-15 Send notification when threshold reached NOTIFICATION 5 TO DO [i]
- IMSFR-16 Add store location LOCATION 5 TO DO [i]
- IMSFR-17 Actions based on store location LOCATION 5 TO DO [i]

+ Create issue

IMSFR Sprint 4 14 Nov - 20 Nov (6 issues) Start sprint [x]

- IMSFR-18 List all items of the store CUSTOMER 3 TO DO [M]
- IMSFR-19 Filter by locations of the store CUSTOMER 3 TO DO [i]
- IMSFR-20 Make a purchase of a stock CUSTOMER 3 TO DO [i]
- IMSFR-21 Indication when trying to make a purchase that doesn't meet the threshold CUSTOMER 3 TO DO [i]
- IMSFR-22 Create dockerfile DEPLOYMENT 4 TO DO [i]
- IMSFR-23 Deploy app in kubernetes DEPLOYMENT 4 TO DO [i]

+ Create issue

Backlog (0 issues) Create sprint [x]

Your backlog is empty.

+ Create issue Quickstart [x]

7. CODING & SOLUTIONING

7.1 Registration & Login

```
def register():
    form = RegistrationForm()

    if form.validate():
        retailer_data = {
            'name': form.name.data,
            'email': form.email.data,
            'address': form.address.data,
            'is_active': False
        }
```

```

        retailer = Retailer(**retailer_data)
        retailer.set_password(form.password.data)

        send_confirmation_email(retailer)

        db.session.add(retailer)
        db.session.commit()
        response_data = {'id': retailer.id}
        return response.success(status_code=RESOURCE_CREATED,
data=response_data, message=REGISTRATION_SUCCESS)

    return response.error(status_code=UNPROCESSABLE_ENTITY,
data=form.errors, message=INVALID_DATA)

def login():
    form = LoginForm()

    if form.validate():
        retailer =
Retailer.query.filter_by(email=form.email.data).scalar()

        if retailer is None or (not
retailer.check_password(form.password.data)):
            return response.error(status_code=UNAUTHORIZED_ACCESS,
message=INVALID_DATA)
            elif not retailer.is_active:
                return response.error(status_code=UNAUTHORIZED_ACCESS,
message=NOT_ACTIVE)

        access_token = create_access_token(retailer)
        response_data = {
            'jwt_token': access_token
        }
        return response.success(status_code=REQUEST_COMPLETED,
data=response_data, message=LOGIN_SUCCESS)

    return response.error(status_code=UNPROCESSABLE_ENTITY,
data=form.errors, message=INVALID_DATA)

```

7.2 Sending Mails

```
def send_mail(subject, to_emails, plain_text_content=None,
html_content=None,
from_email=current_app.config['EMAIL_CONFIRMATION_SENDER_EMAIL']):
    message = Mail(
        from_email=from_email,
        to_emails=to_emails,
        subject=subject,
        plain_text_content=plain_text_content,
        html_content=html_content)

    SendGridAPIClient(current_app.config['SENDGRID_API_KEY']).send(message)

def send_confirmation_email(user):
    try:
        to_emails = [user.email]
        confirmation_token = user.get_confirmation_token()
        confirmation_link = url_for(
            'auth.confirm_email', token=confirmation_token,
            _external=True)
        subject = 'Inventory: Confirm Your Mail Now!'
        plain_text_content = 'Please confirm your account by clicking
the confirmation link below.'
        html_content = f'''
            <a href='{confirmation_link}'
target='_blank'>Click Here to Confirm Your Account!</a>
        '''
        send_mail(subject=subject, to_emails=to_emails,
            plain_text_content=plain_text_content,
            html_content=html_content)
    except Exception as e:
        current_app.log_exception(e)

def send_restock_mail(user, product_name):
    try:
        to_emails = [user.email]
        subject = 'Inventory: Product reached the Threshold.'
        plain_text_content = f'Please start think about restocking the
product {product_name}'
```

```

        send_mail(subject=subject, to_emails=to_emails,
                  plain_text_content=plain_text_content)
    except Exception as e:
        current_app.log_exception(e)

```

7.3 Products

```

@jwt_required()
def create():
    form = ProductForm()

    if form.validate():
        product_data = {
            'name': form.name.data,
            'description': form.description.data,
            'retailer_id': current_user.id
        }
        product = Product(**product_data)

        db.session.add(product)
        db.session.commit()
        response_data = {'id': product.id}
        return response.success(status_code=RESOURCE_CREATED,
                                data=response_data, message=PRODUCT_CREATED)

    return response.error(status_code=UNPROCESSABLE_ENTITY,
                            data=form.errors, message=INVALID_DATA)

@jwt_required()
def get_all():
    products = [product.to_dict() for product in current_user.products]

    return response.success(status_code=REQUEST_COMPLETED,
                            data=products, message=ALL_PRODUCTS)

@jwt_required()
@load_product_by_id
def get_by_id(product):
    return response.success(status_code=REQUEST_COMPLETED,
                            data=product.to_dict(), message=PRODUCT)

```

```

@jwt_required()
@load_product_by_id
def update_by_id(product):
    form = ProductEditForm()

    if form.validate():
        form_description = form.description.data
        if form_description != product.description:
            product.description = form_description

        db.session.add(product)
        db.session.commit()

        response_data = {'description': form_description}
        return response.success(status_code=REQUEST_COMPLETED,
data=response_data, message=PRODUCT_UPDATED)

    return response.error(status_code=UNPROCESSABLE_ENTITY,
data=form.errors, message=INVALID_DATA)

@jwt_required()
@load_product_by_id
def delete_by_id(product):
    db.session.delete(product)
    db.session.commit()

    return response.success(status_code=REQUEST_COMPLETED,
data=product.to_dict(), message=PRODUCT_DELETED)

```

7.4 Locations

```

@jwt_required()
def create():
    form = LocationForm()

    if form.validate():
        location_data = {
            'name': form.name.data,
            'address': form.address.data,
            'retailer_id': current_user.id
        }
        location = Location(**location_data)

```

```

        db.session.add(location)
        db.session.commit()
        response_data = {'id': location.id}
        return response.success(status_code=RESOURCE_CREATED,
data=response_data, message=LOCATION_CREATED)

    return response.error(status_code=UNPROCESSABLE_ENTITY,
data=form.errors, message=INVALID_DATA)

@jwt_required()
def get_all():
    locations = [location.to_dict() for location in
current_user.locations]

    return response.success(status_code=REQUEST_COMPLETED,
data=locations, message=ALL_LOCATIONS)

@jwt_required()
@load_location_by_id
def get_by_id(location):
    return response.success(status_code=REQUEST_COMPLETED,
data=location.to_dict(), message=LOCATION)

@jwt_required()
@load_location_by_id
def update_by_id(location):
    form = LocationEditForm()

    if form.validate():
        form_address = form.address.data
        if form_address != location.address:
            location.description = form_address

        db.session.add(location)
        db.session.commit()

        response_data = {'address': form_address}
        return response.success(status_code=REQUEST_COMPLETED,
data=response_data, message=LOCATION_UPDATED)

```

```

        return response.error(status_code=UNPROCESSABLE_ENTITY,
data=form.errors, message=INVALID_DATA)

@jwt_required()
@load_location_by_id
def delete_by_id(location):
    db.session.delete(location)
    db.session.commit()

    return response.success(status_code=REQUEST_COMPLETED,
data=location.to_dict(), message=LOCATION_DELETED)

```

7.5 Inventory

```

@jwt_required()
def create():
    form = InventoryForm()

    if form.validate():
        product_id = Product.query.with_entities(Product.id).filter_by(
            name=form.product_name.data,
retailer_id=current_user.id).scalar()
        location_id =
Location.query.with_entities(Location.id).filter_by(
            name=form.location_name.data,
retailer_id=current_user.id).scalar()

        already_exists = Inventory.query.filter_by(
            product_id=product_id, location_id=location_id).scalar()

        if already_exists:
            return response.error(status_code=UNPROCESSABLE_ENTITY,
message=ALREADY_EXISTS)

        inventory_data = {
            'product_id': product_id,
            'location_id': location_id,
            'quantity': form.quantity.data,
            'threshold': form.threshold.data
        }

        inventory = Inventory(**inventory_data)

```



```

        db.session.add(inventory)
        db.session.commit()
        response_data = {'id': inventory.id}
        return response.success(status_code=RESOURCE_CREATED,
data=response_data, message=INVENTORY_RECORD_CREATED)

        return response.error(status_code=UNPROCESSABLE_ENTITY,
data=form.errors, message=INVALID_DATA)

@jwt_required()
def get_all():
    product_ids = {product.id for product in current_user.products}
    inventory_items = Inventory.query.filter(
        Inventory.product_id.in_(product_ids)).all()
    inventory_items = [inventory_item.to_dict()
        for inventory_item in inventory_items]

    return response.success(status_code=REQUEST_COMPLETED,
data=inventory_items, message=ALL_INVENTORY_ITEMS)

@jwt_required()
@load_inventory_item_by_id
def get_by_id(inventory_item):
    return response.success(status_code=REQUEST_COMPLETED,
data=inventory_item.to_dict(), message=INVENTORY_ITEM)

@jwt_required()
@load_inventory_item_by_id
def update_by_id(inventory_item):
    form = InventoryEditForm()

    if form.validate():
        form_quantity = form.quantity.data
        if form_quantity != inventory_item.quantity:
            inventory_item.quantity = form_quantity

        form_threshold = form.threshold.data
        if form_threshold != inventory_item.threshold:
            inventory_item.threshold = form_threshold

```

```

        db.session.add(inventory_item)
        db.session.commit()

        response_data = {'quantity': form_quantity,
                        'threshold': form_threshold}

        return response.success(status_code=REQUEST_COMPLETED,
data=response_data, message=INVENTORY_ITEM_UPDATED)

        return response.error(status_code=UNPROCESSABLE_ENTITY,
data=form.errors, message=INVALID_DATA)

@jwt_required()
@load_inventory_item_by_id
def delete_by_id(inventory_item):
    db.session.delete(inventory_item)
    db.session.commit()
    response_data = {'id': inventory_item.id}
    return response.success(status_code=REQUEST_COMPLETED,
data=response_data, message=INVENTORY_ITEM_DELETED)

```

7.6 Marketplace

```

def get_all_retailers():
    retailers = [{'id': retailer.id, 'name': retailer.name}
                for retailer in
Retailer.query.with_entities(Retailer.id, Retailer.name).all()]
    return response.success(status_code=REQUEST_COMPLETED,
data=retailers, message=ALL_RETAILERS)

def get_all_locations_for_a_retailer(retailer_id):
    locations_for_a_retailer = Location.query.with_entities(Location.id,
Location.name).filter_by(
        retailer_id=retailer_id).all()
    locations_for_a_retailer = [{'id': location.id, 'name':
location.name}
                                for location in
locations_for_a_retailer]
    return response.success(status_code=REQUEST_COMPLETED,
data=locations_for_a_retailer, message=LOCATIONS_FOR_A_RETAILER)

```

```

def get_all_products_for_a_location(location_id):
    products_for_a_location =
Product.query.with_entities(Product.name).where(

Product.id.in_(Inventory.query.with_entities(Inventory.product_id).filt
er_by(location_id=location_id))).all()
    products_for_a_location = [{'name': product.name}
                                for product in products_for_a_location]
    return response.success(status_code=REQUEST_COMPLETED,
data=products_for_a_location, message=PRODUCTS_FOR_A_LOCATION)

def complete_purchase_order(retailer_id):
    form = PurchaseOrderForm()

    if form.validate():
        form_product_name = form.product_name.data
        product_id = Product.query.with_entities(
            Product.id).filter_by(name=form_product_name,
retailer_id=retailer_id).scalar()
        location_id = Location.query.with_entities(
            Location.id).filter_by(name=form.location_name.data,
retailer_id=retailer_id).scalar()

        existing_product_record = Inventory.query.filter_by(
            product_id=product_id, location_id=location_id).scalar()

        if not existing_product_record:
            return response.error(status_code=UNPROCESSABLE_ENTITY,
message=PRODUCT_NOT_EXIST_IN_THE_LOCATION)

        form_quantity = form.quantity.data
        if form_quantity > existing_product_record.quantity:
            return response.error(status_code=UNPROCESSABLE_ENTITY,
message=TOO_MUCH_QUANTITY)

        existing_product_record.quantity -= form_quantity

        if existing_product_record.quantity <=
existing_product_record.threshold:
            retailer =
Retailer.query.filter_by(id=Product.query.with_entities(

```

```

Product.retailer_id).filter_by(id=existing_product_record.product_id).s
calar()).scalar()

    send_restock_mail(retailer, form_product_name)

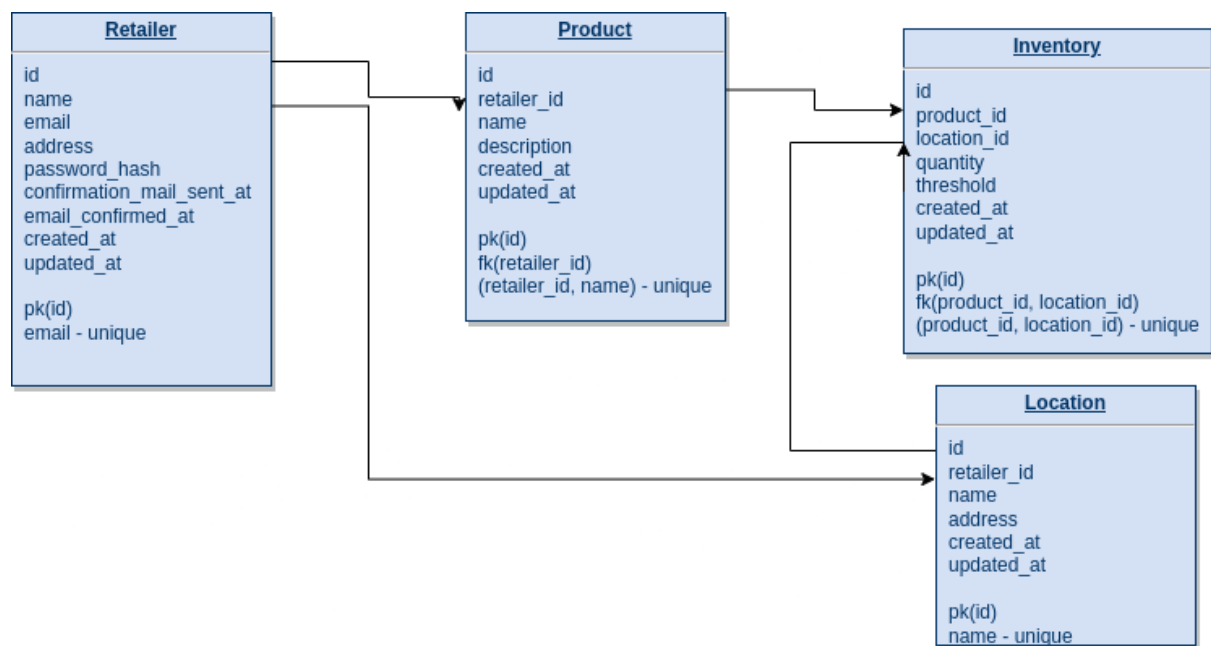
    db.session.add(existing_product_record)
    db.session.commit()

    respose_data = {'bought': form_product_name, 'quantity':
form_quantity}
    return response.success(status_code=REQUEST_COMPLETED,
                           data=respose_data,
message=PURCHASE_SUCCESSFUL)

    return response.error(status_code=UNPROCESSABLE_ENTITY,
                          data=form.errors, message=INVALID_DATA)

```

7.7 DB Models



8. Testing

8.1 Test Cases

<https://github.com/IBM-EPBL/IBM-Project-10417-1659179409/tree/main/Project%20Development%20Phase/User%20Acceptance%20Testing>

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_001	Functional	Home Page	Verify user is able to see the Login/Signup popup when user land on the page		1.Enter URL, and click go 2.Verify login/signup popup displayed or not		Login/signup popup should display	Working as expected	Pass		N		
LoginPage_TC_002	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL, and click go 2.Click on respective button 3.Verify login/signup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link e.Last password? Recovery password link		Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery password link	Working as expected	Pass		N		
LoginPage_TC_003	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL, and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: test@gmail.com password: Testing123	User should navigate to user account homepage.	Working as expected	Pass		N		
LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL, and click go 2.Click on button 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: test@gmail.com password: Testing6	Application should show 'incorrect email or password' validation message.	Working as expected	Pass		N		
ProductPage_TC_001	Functional	Product page	Verify user is able to see the products page		1. Go to the site 2. Click on products button		Application should show the products page	Working as expected	Pass		N		
ProductPage_TC_002	Functional	Product page	Verify user is able to add a product		1. Go to the site 2. Click on products button 3. Click on add 4. Fill necessary details		Application should create a product	Working as expected	Pass		N		
ProductPage_TC_003	Functional	Product page	Verify user is able to delete a product		1. Go to the site 2. Click on products button 3. Click on delete		Application should delete a product	Working as expected	Pass		N		
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
ProductPage_TC_002	Functional	Product page	Verify user is able to add a product		1. Go to the site 2. Click on products button 3. Click on add 4. Fill necessary details		Application should create a product	Working as expected	Pass		N		
ProductPage_TC_003	Functional	Product page	Verify user is able to delete a product		1. Go to the site 2. Click on products button 3. Click on delete		Application should delete a product	Working as expected	Pass		N		
ProductPage_TC_004	Functional	Product page	Verify user is able to update a product		1. Go to the site 2. Click on products button 3. Click on update 4. Fill necessary details		Application should update a product	Working as expected	Pass		N		
LocationPage_TC_001	Functional	Location page	Verify user is able to create a location		1. Go to the site 2. Click on locations button 3. Click on add 4. Fill necessary details		Application should add a location	Working as expected	Pass		N		
LocationPage_TC_002	Functional	Location page	Verify user is able to delete a location		1. Go to the site 2. Click on locations button 3. Click on delete		Application should delete a location	Working as expected	Pass		N		
LocationPage_TC_003	Functional	Location page	Verify user is able to update a location		1. Go to the site 2. Click on locations button 3. Click on update 4. Fill necessary details		Application should update a location	Working as expected	Pass		N		
InventoryPage_TC_001	Functional	Inventory page	Verify user is able to create a inventory record		1. Go to the site 2. Click on inventory button 3. Click on add 4. Fill necessary details		Application should create an inventory	Working as expected	Pass		N		
InventoryPage_TC_002	Functional	Inventory page	Verify user is able to delete a inventory record		1. Go to the site 2. Click on inventory button 3. Click on delete		Application should delete an inventory	Working as expected	Pass		N		
InventoryPage_TC_003	Functional	Inventory page	Verify user is able to update a inventory record		1. Go to the site 2. Click on inventory button 3. Click on update 4. Fill necessary details		Application should update an inventory	Working as expected	Pass		N		

8.2 Defect Analysis

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	0	0	0	0
Duplicate	0	0	2	0	2
External	0	0	0	0	0
Fixed	0	0	0	3	0
Not Reproduced	0	0	0	1	1
Skipped	0	0	0	0	0
Won't Fix	0	0	3	0	3
Totals	0	0	5	4	6

8.3 Test Case Analysis

Section	Total Cases	Not Tested	Fail	Pass
Login	4	0	0	4
Product	4	0	0	4
Location	3	0	0	3
Inventory	3	0	0	3
Marketplace	1	0	0	1

9. Performance Metrics

NFT - Risk Assessment									
S.No	Project Name	Scope/Feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volumen Changes	Risk Score	Justification
1	Inventory Management	New	Low	No Changes	Moderate		>5 to 10%	ORANGE	
NFT - Detailed Test Plan									
S.No	Project Overview		NFT Test approach		assumptions/Dependencies/Risk		Approvals/SignOff		
1	Inventory Management System		Manual		Device with internet				
End Of Test Report									
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff	
1	Inventory Management Sys Manual			As expected, working fine		Laptop or Desktop gives better use	None		
				Team ID	PNT/2022/MID/15205				
				Project Name	Inventory Management sSystem For Retailer				

10. Advantages & Disadvantages

- **Allows to monitor product trends**
- **Allows to restock quickly**
- **Allows to understand user needs**
- **Allows to understand demographics of location**

11. Conclusion

So we've created an application that helps retailers to manage their inventories in various locations. They can also receive email whenever the stock count meets the threshold.

12. Future Scope

This is currently implemented as a web application. But a mobile app was also needed in order to connect and make the retailer's life easier. Future scope is to add mobile app support. Listing the purchases happened for each retailer. Showing graphs for the trends.

13. Appendix

- Github - <https://github.com/IBM-EPBL/IBM-Project-10417-1659179409>
- Application - <http://159.122.178.236:31098/register.html>
- Video link - <https://www.loom.com/share/117c36a1fa1344828258e0187eccea95>