1. INTRODUCTION

1.1 PROJECT OVERVIEW

An inventory management system (or inventory system) is the process by which you track your goods throughout your entire supply chain, from purchasing to production to end sales. It governs how you approach inventory management for your business. When it comes to monitoring and maintaining stocked items, Inventory management system is used to check whether the company assets, raw materialsand supplies, or finished goods that are ready to be sent to vendors or end users. Inventory managementsystem is the combination of technology (hardware and software) and processes and procedures. It can be used to construct a checklist, bill of materials, and other documentation connected to production in the industrial sector. To prevent products overstock and outages, businesses utilize inventory management software. It is a tool for organizing inventory data, which was previously typically kept in hard copy. Inventory are items that a business keeps on hand while producing the product and its component parts for sale. the variety of forms that finished goods and inventory might take. The goal ofinventory management is to maintain inventory at the lowest cost possible given the objectives to ensure ongoing provisions for ongoing activities. While making judgments on inventory management, acompromise must be reached between several cost factors. It may include the expenses associated with providing inventory, inventory keeping, and expenditures brought on by insufficient stocks.

1.2 PURPOSE

- Tracking the movement of goods between places
- Delivering goods into a warehouse or another place.
- Monitoring product sales and stock levels.
- Avoiding product damage and obsolescence.
- Avoiding losing out on sales due to stock shortages.
- Gathering, packing, and delivering goods from a warehouse.
- Sustaining a balance between excessive and insufficient inventory.
- For a cost secretarial plan to be successful, there must be proper control of accounts and equipment from the time that information is placed with the provider until they have beensuccessfully used in manufacturing.
- The planning and routing department in the automotive sector is in charge of determine ngwhere and how the job is to be done as well as issuing directions.

2. LITERATURE SURVEY
2.1 EXISTING PROBLEM

S.NO	PAPER	AUTHOR	YEAR	METHOD AND ALGORITHM	ACCURACY / PRECISION
1	Research Paperon Inventory Management .	Punam Khobragade, Roshni Selokar, Rina Maraskolhe, Prof. Manjusha Talmale	2018	Inventory Management System is software which is helpful for the businesses operate hardware stores, where storeowner keeps the records of sales and purchase. Mismanaged inventory means disappointed customers, too much cash tied up in warehouses and slower sales. This project eliminates the paper work, human faults, manual delay and speed up process.	98.6%
2	A Cloud-Based Inventory Management System Using a Smart Trolley for Automated Billing and Theft Detection.	B. Karunakara Rai, J. P. Harshitha, Radhika S. Kalagudi, B. S. Priyanka Chowdary, Palak Hora & B. Sahana	2019	Inventories are raw materials, work-in-process goods and completely finished goods that are considered to be the portionof business's assets that are ready or will be ready for sale. Formulating a suitable inventory model is one of the major concerns for an industry. The earliest scientific inventory management researches date back to the	93.6%

				second decade of the past century, but the interest in this scientific area is still great	
3	A Literature Review on Models of Inventory Management under Uncertainty	Serhii ZUIKOV	2018	Inventories are raw materials, work-in-process goods andcompletely finished goods thatare considered to be the portionof business's assets that areready or will be ready for sale. Formulating a suitableinventory model is one of themajor concerns for an industry.The earliest scientific inventory management researches date back to the second decade of the past century, but the interest in this scientific area is still great.	89.2%
4	Inventory management for retail companies: A literature review and current trends	Cinthya Vanessa Munoz Macas, Jorge Andres Espinoza Aguirre, Rodrigo Arcentales-Carrion, Mario Pena	2021	In recent years, the correct management of inventories has become a fundamental pillar for achieving success in enterprises. Unfortunately, studies suggesting the investment and adoption of advanced inventory management and control systems are not easy to find. In this context, this article aims to analyze and present an extensive literature concerning inventory management, containing multiple	82.88%

				definitions and fundamental concepts for the retail sector. A systematic literature review was carried out to determine the main trends and indicators of inventory management in Small and Medium-sized Enterprises (SMEs). This research covers five years, between 2015 and 2019, focusing specifically on the retail sector. The primary outcomes of this study are the leading inventory management systems and models, the Key Performance Indicators (KPIs)for their correct management, and the benefits and challengesfor choosing or adopting an efficient inventory control and management system. .	
5	Simulation of inventory management systems in retail stores: A case study	Puppala Sridhar, C.R. Vishnu, R Sridharan	2021	A simulation model is developed and run for particular merchandise using Arena simulation software. Rigorous experimentation is conducted with the model by altering the inputs/model characteristics, and a more effective system is proposed. Compared with the existing	87%

				traditional inventory management system, the proposed system will reduce the inventory level by 40% and lost sales by 87%. Furthermore, the proposed system is optimized using the OptQuest module in Arena simulation software. As a result, the inventory level is further reduced by 73% compared to the existing system. Store managers in various organizations may utilize the proposed methodology for improving their inventory management system.	
6	Inventory Management in Retail Store	Rohan Agarwal		The purchasing, receiving, engineering, displaying, and accounting departments all contribute to the accuracy of the inventory methods and records. Inaccurate inventory management will contribute to dispatch delays, shortage in stores, purchasing of the wrong inventory and stocking too much inventory.	90.25%

2.2 REFERENCES

[1] Sambasiva Rao K., Singh, Sukhdev. (2006). Inventory control practices in IFFCO. The Management Accountant.

[2] Pradeep Singh (2008)," Inventory and Working Capital Management- An Empirical Analysis", The ICFAI Journal of Accounting and Research.

[3] Capkun, Vedran, Hameri, Ari-Pekka & Weiss, Lawrence A. (2009). On the relationship between inventory and financial performance in manufacturing. International Journal of Operations & Production Management.

[4] Gaur, Jighyasu & Bhattacharya, Sourabh. (2011). The relationship of financial and inventory performance of manufacturing firms in Indian context. California Journal of Operations Management.

[5] Krishnankutty, Raveesh. (2011). Panel data analysis on retail inventory productivity. The Economic Research Guardian.

[6] Eneje, B. C., Nweze, A .U. & Udeh, A. (2012). Effect of Efficient Inventory Management on Profitability: Evidence from Selected Brewery Firms in Nigeria. International Journal of Current Research.

[7] Nyabwanga, Robert Nyamao & Ojera, Patrick. (2012). Inventory management practices and business performance for small scale enterprises in Kenya. KCA Journal of Business Management.

[8] Madishetti, Srinivas & Kibona, Deogratias. (2013). Impact of inventory management on the profitability of SMEs in Tanzania. Internation Commerce & Management.

[9] Anichebe, N. A. & Agu, O. A. (2013). Effect of Inventory Management on Organizational Effectiveness. Information and Knowledge Management.

[10] Panigrahi, Ashok K. (2013). Relationship between inventory management and profitability: An empirical analysis of Indian cement companies. Asia Pacific Journal of Marketing & Management Review.

[11] Srinivasa Rao Kasisomayajula (2014) "An Analytical Study on Inventory Management in Commercial Vehicle Industry in India", International Journal of Engineering Research.

[12] Sanjiv Mittal, R.K. Mittal, Gagandeep Singh, Sunil Gupta (2014)." Inventory Management in Fertiliser Industry of India: An Empirical Analysis" Asia-Pacific Journal of Management Research and Innovation.

[13] Viplesh Shardeo (2015), "Impact of Inventory Management on the Financial Performance of thefirm" IOSR Journal of Business and Management (IOSRJBM).

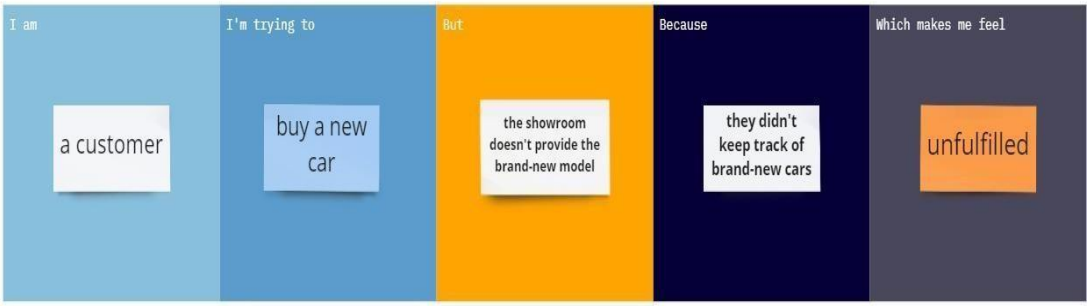
[14] Edwin Sitienei, Florence Memba (2015-16) "The Effect of Inventory Management on Profitability of Cement Manufacturing Companies in Kenya: A Case Study of Listed Cement Manufacturing Companies in Kenya" International Journal of Management and Commerce Innovations.

[15] Soni, Anita. (2012). Inventory management of engineering goods industry in Volume: 5 | Issue:8 | August 2016 ISSN - 2250-1991 | IF: 5.215 | IC Value: 77.65 216 | PARIPEX - INDIAN JOURNALOF RESEARCH Punjab: An empirical analysis. International Journal of Multidisciplinary Research.

2.3 PROBLEM STATEMENT DEFINITION

PROBLEM STATEMENT 1

PROBLEM STATEMENT 2



3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

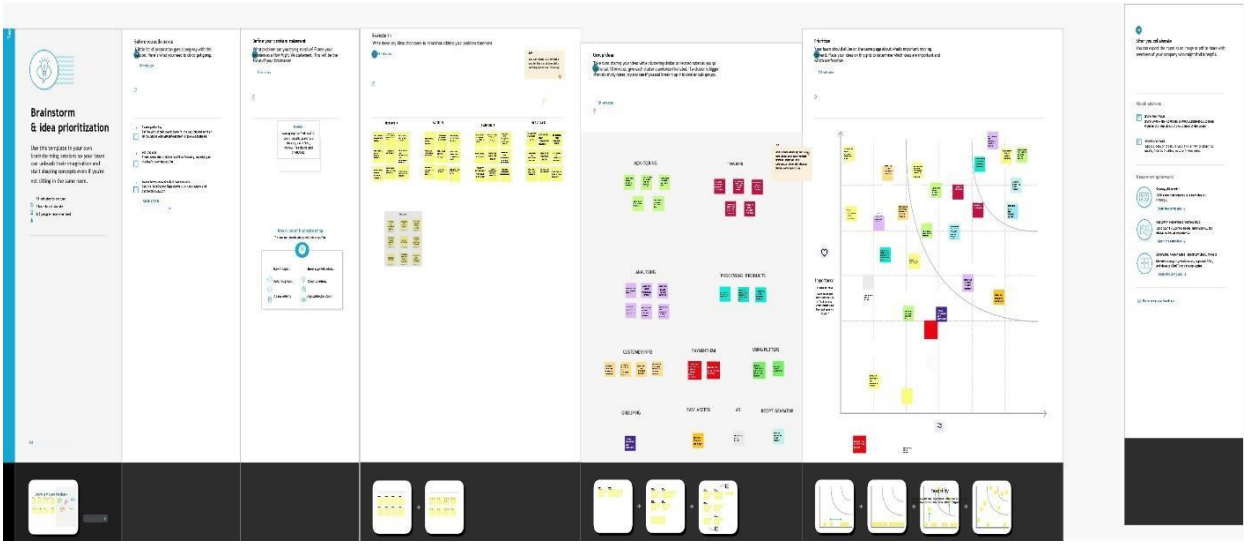
The core empathy map, which aids in identifying and describing the user's wants and pain points, is expanded upon in an empathy map canvas. Additionally, this data is useful for enhancing user experience.Teams employ user insights to map out what matters to, impacts, and how their target audience presents themselves. Using this data, personas are then developed to assist teams in visualizing and empathizing with users as people rather than just as a general marketing demographic or account number.

Empathy Map Canvas

3.2 IDEATION & BRAINSTORMING

Ideation fundamentally refers to the entire creative process of coming up with and sharing new ideas. Ideation is creative thought that usually aims to solve a problem or offer a better way to do something. It includes coming up with new ideas, developing current ideas, and determining howto put new ideas into effect.

Ideation and brainstorming, a particular method for producing fresh ideas, are frequently closely related activities. When brainstorming, a group of people are usually brought together to generate either new, broad ideas or suggestions for how to handle a particular situation or problem.



Ideation &
Brainstorming

3.3 PROPOSED SOLUTION

Making an application for retailers to maintain their inventory supplies and manage purchases, sales,stocks, etc. is the challenge that needs to be solved.

Solution description

The solution is to create an application that tracks and manages stock levels for their own product lines. The retailers create their accounts by verifying their information and entering their product stock/inventory. When finished, they can log into the application to view their supplies, sales, and change their stocks when restocking, among other things. They can identify which stocks are in high demand, and when those stocks are in danger of running out, they are alerted so they can restock them.

Uniqueness

Since we have information on stock sales, we can estimate which stocks will be the most popular so that shops may refill up on those items first. Regression analysis and historical sales data within our applicationcan be used to retrieve the data. By containerizing using a Docker application, maintenance and developmentcan also be made simpler.

Customer Satisfaction

Using the information from our application, we can buy and refill only the stocks that are needed, reducing excess stocks in the inventory that could result in product waste. We can also observe which goods are selling well and which are not doing as well as anticipated. We can request the necessary quantity of inventories from vendors and suppliers and initiate better arrangements with them as we will be aware of which products are required in large quantities.

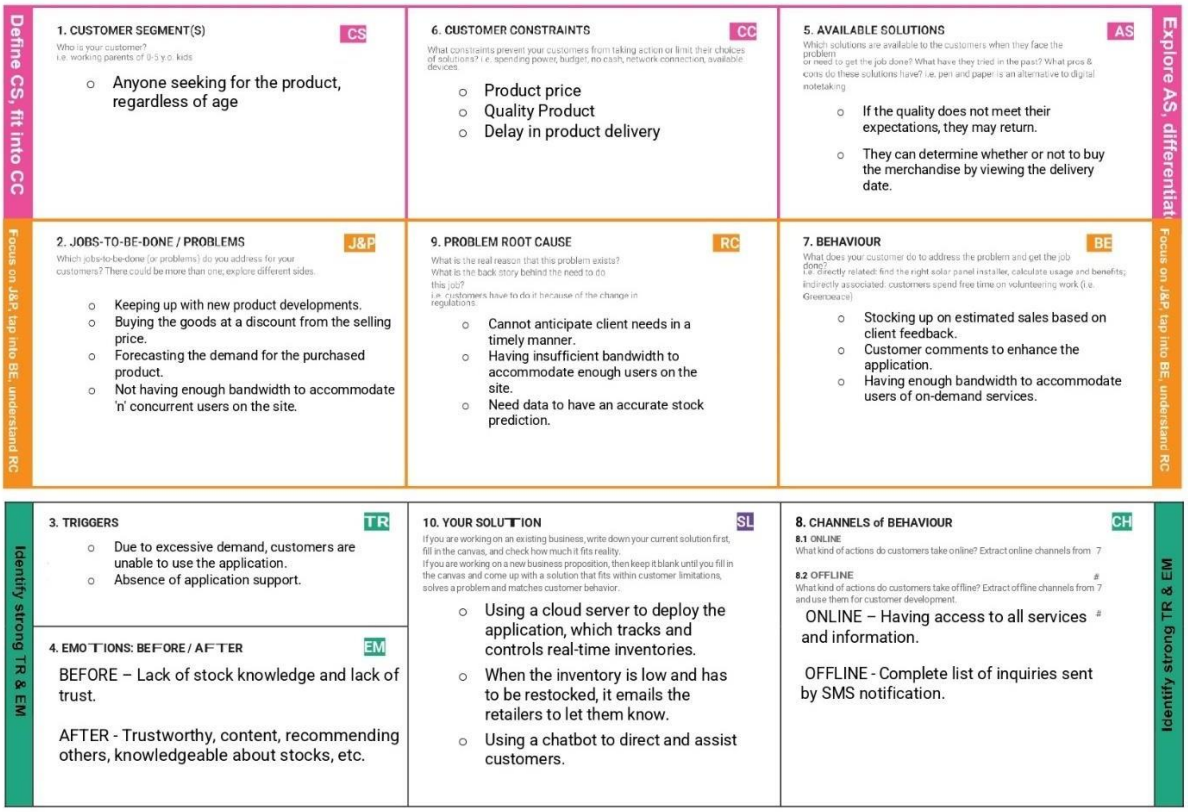
Business Model (Revenue Model)

By analyzing the predicted products that have a higher likelihood of being purchased in large quantities and eliminating unnecessary redundant products that may be excess when not ordered in the right amount, retailers can order the fast-moving products and the appropriate number of stocks from suppliers andvendors.

Scalability of the Solution

Through virtualization, scalable cloud architecture is made possible. Unlike actual machines, which have processors, memory, and other physical hardware that determines their resources and performance. Thevirtual machines we utilize on the IBM Cloud are very scalable and adaptable. Users of Kubernetes can scalethe containers in accordance with changing application requirements. Via command lines, changing the number is simple.

3.4 PROBLEM-SOLUTION FIT



4. REQUIREMENT ANALYSIS

4.1FUNCTIONAL REQUIREMENT

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via EmailConfirmation via OTP
FR-3	Sign in	Sign in to the application by LinkedIn/Gmail, Usernameand Password.
FR-4	Dashboard	Can view the product details and offers.
FR-5	Booking	The required products are selected and booked.
FR-6	Shipping	To track the delivery details of the selected product.
FR-7	Restocking	Ordering more products when the stock is low.

4.2NON-FUNCTIONAL REQUIREMENTS

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

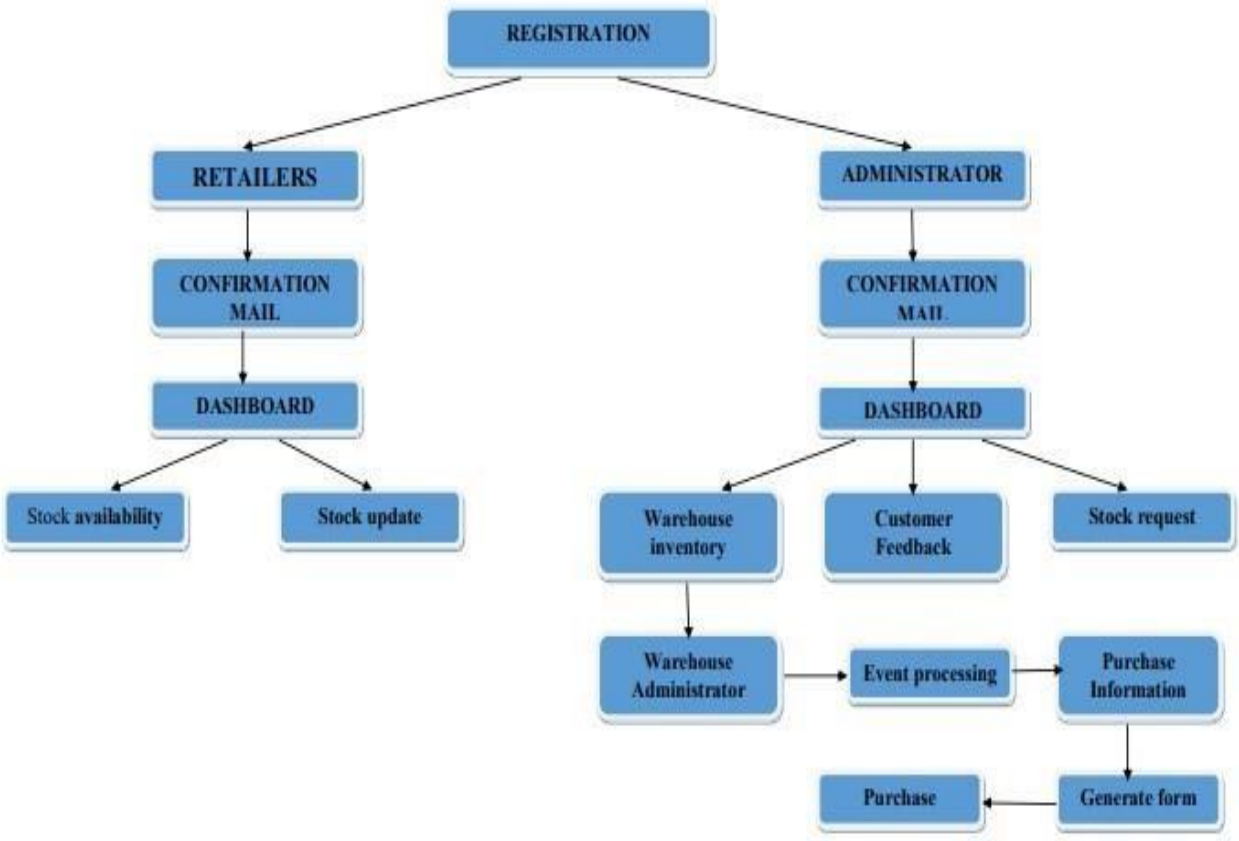
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	✓ Creating a learning curve into the site's design and development. ✓ Having a user-friendly, straightforwardwebsite. Beautiful-looking website. ✓ Making the website responsive forconsumers on both desktops and mobile devices.
NFR-2	**Security**	✓ Strong security is necessary to prevent hackers from accessing the accounts or dataof authorized users. To demonstrate authentication and authorization, log in systems is utilized. ✓ Utilizing OTP can improve security. ✓ Cookies-based security mechanism for userauthentication and enhanced website user ✓ experience
NFR-3	**Reliability**	✓ When the website is active, it should be ableto manage the necessary number of users without slowing or causing any inconvenience to the user. ✓ While running the apps, there should be fewmistakes. ✓ It should be accessible even duringdisasters.

NFR-4	**Performance**	✓ This has the advantage of cutting down the time needed for aisle and product searches, among other conveniences. ✓ It decreases expenses, saves time during restocking, and forecasts the top-selling goods. ✓ Due to the business's streamlined management system, it is more productive ✓ and profitable.
NFR-5	**Availability**	✓ To provide high availability of database servers and performances, this employs IBM ✓ DB2.
NFR-6	**Scalability**	✓ Due to DB2's excellent scalability, coding can be created and developed quickly, and new features can be added without much difficulty. ✓ High-scalability IBM Container is utilised in the Docker registry. ✓ Any new functionality can be added by reusing the code.

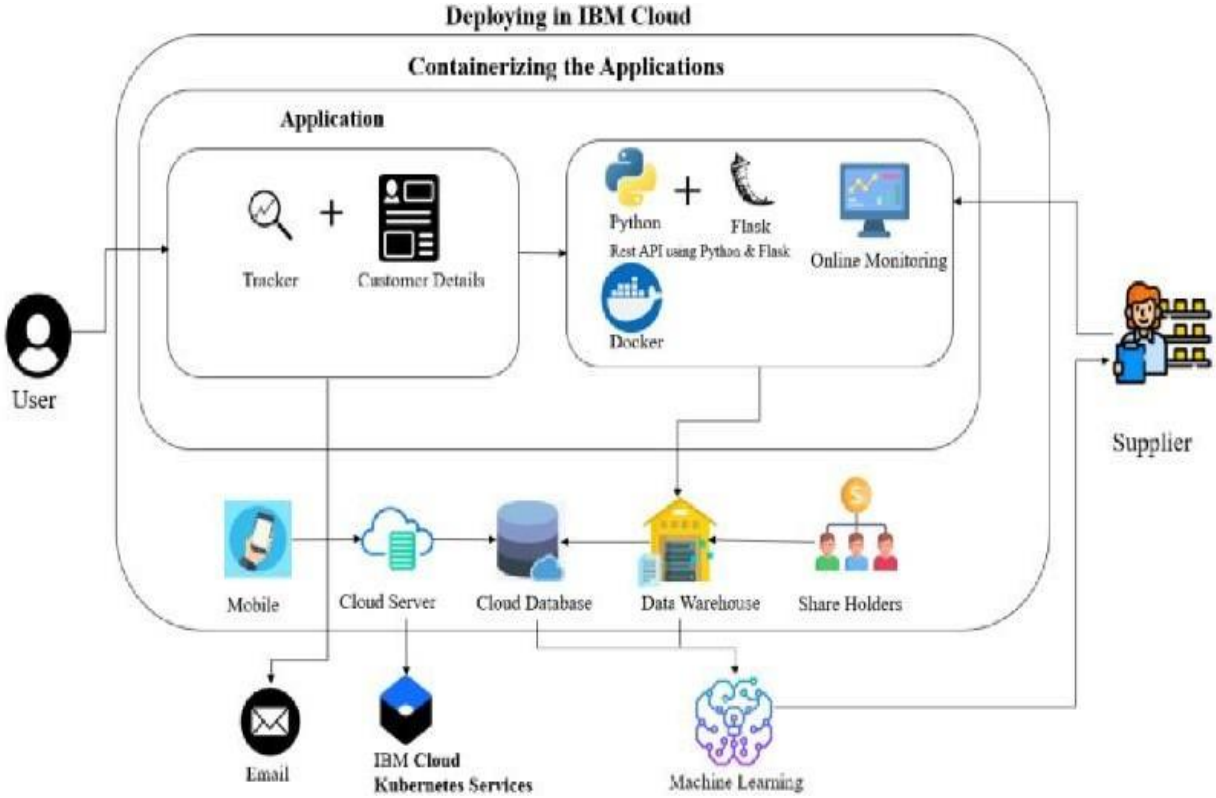
5. PROJECT DESIGN

5.1DATA FLOW DIAGRAMS

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



5.2SOLUTION & TECHNICAL ARCHITECTURE



5.3USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirmingmy password.	I can access myaccount / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once Ihave registered for the application	I can receive confirmation email& click confirm	High	Sprint-1

		USN-3	As a user, I can registerfor the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can registerfor the application through Gmail	I can register & access the dashboard by using my Gmail.	Medium	Sprint-3
	Login	USN-5	As a user, I can log intothe application by entering email & password	I can login with registered email and password.	High	Sprint-4
	Dashboard	USN-6	As a user, I have access to both the currently available products and the out-of-stock products.	Once logged in, you may view theinventory.	High	Sprint-4
	Restocking Product	USN-7	As a user, I can refill theproducts and add items that aren't already in the inventory.	Retailers have the option to refill andupdate their inventory when theproducts are not available.	Medium	Sprint-5
Customer Care Executive	Request for customer care	USN-8	As a user, I have accessto the customer service administrators and can ask questions about myconcerns.	Users can get assistance andsupport from executives by contacting customer care.	Medium	Sprint-3
Administrator	Collecting Feedback	USN-9	As a user, I have the ability to give feedback forms outlining any suggestions for enhancingor correcting any problems that I have.	Users can provide administrators withinput on problems or enhancements.	Medium	Sprint-5

6. PROJECT PLANNING & SCHEDULING
6.1SPRINT DELIVERY SCHEDULE

Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Point	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by using my email & passwordand confirming my login credentials.	3	High	Vignesh R Yashawini S Yashvandh PR Yaswant B
Sprint-1		USN-2	As a user, I can loginthrough my E-mail.	3	Medium	Vignesh R Yashawini S Yashvandh PR Yaswant B Vignesh R
Sprint-1	Confirmation	USN-3	As a user, I can receive my confirmation email once I have registeredfor the application.	2	High	Vignesh R Yashawini S Yashvandh PR Yaswant B
Sprint-1	Login	USN-4	As a user, I can log into the authorized account by entering the registered email and password.	3	Medium	Vignesh R Yashawini S Yashvandh PR Yaswant B
Sprint-2	Dashboard	USN-5	As a user, I can view the products that are available currently.	4	High	Vignesh R Yashawini S Yashvandh PR Yaswant B Vignesh R
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	Vignesh R Yashawini S Yashvandh PR Yaswant B

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	Vignesh R Yashawini S Yashvandh PR Yaswant B
Sprint-4	Request for customercare	USN-8	As a user, I am able to request customer care to get in touch with the administrators and enquire the doubts andproblems.	4	Medium	Vignesh R Yashawini S Yashvandh PR Yaswant B
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 toget it resolved.	3	Medium	Vignesh R Yashawini S Yashvandh PR Yaswant B

6.2 REPORTS FROM JIRA

Project Tracker, Velocity:

Sprint	Total Story Points	Duration	Sprint StartDate	Sprint End Date(Planned)	Story Points Completed (as on Planned EndDate)	Sprint ReleaseDate (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

Velocity:

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

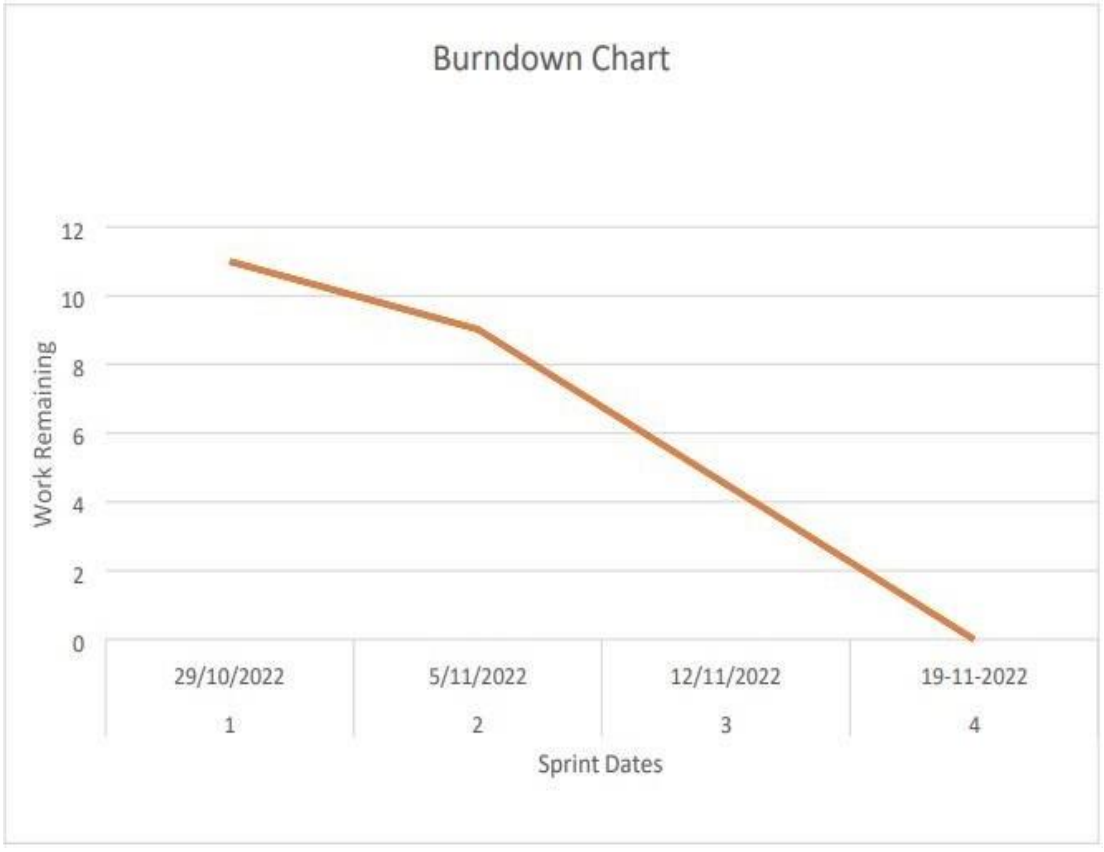
$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Our velocity should be:

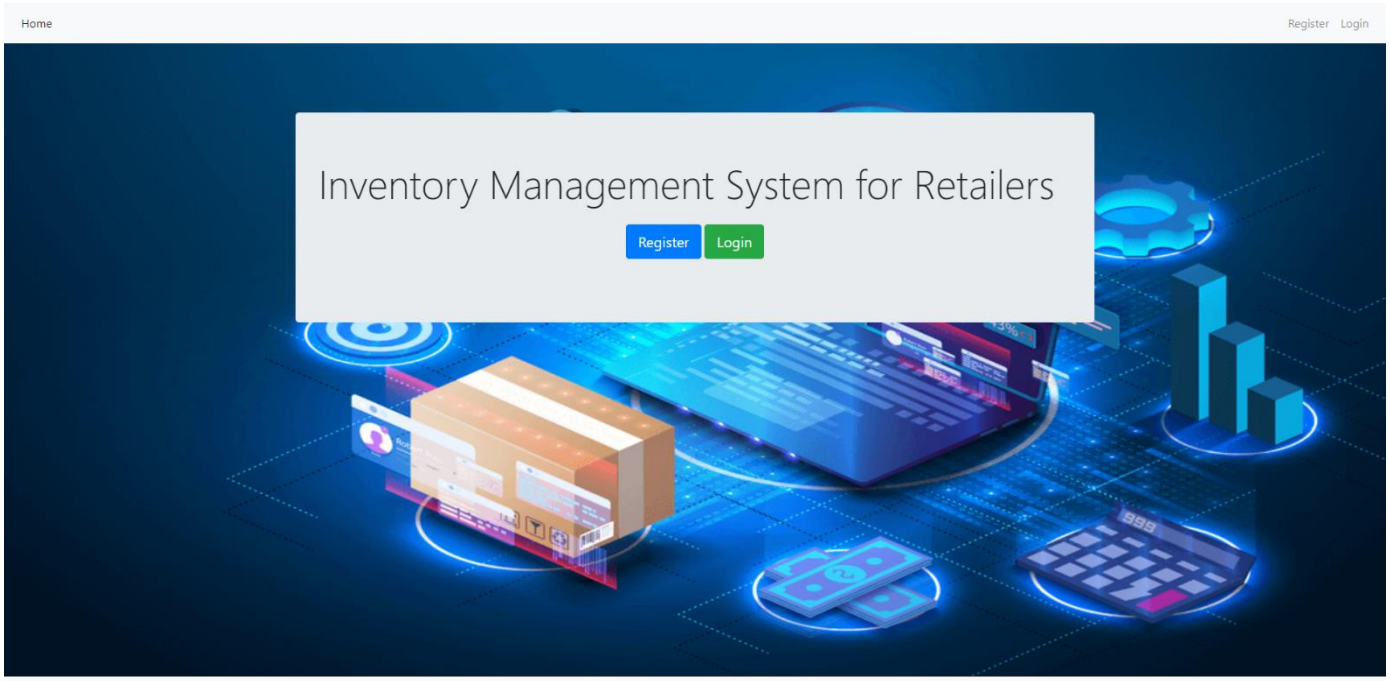
$$AV = \frac{11+7+6+7}{24} = \frac{31}{24} = 1.29$$

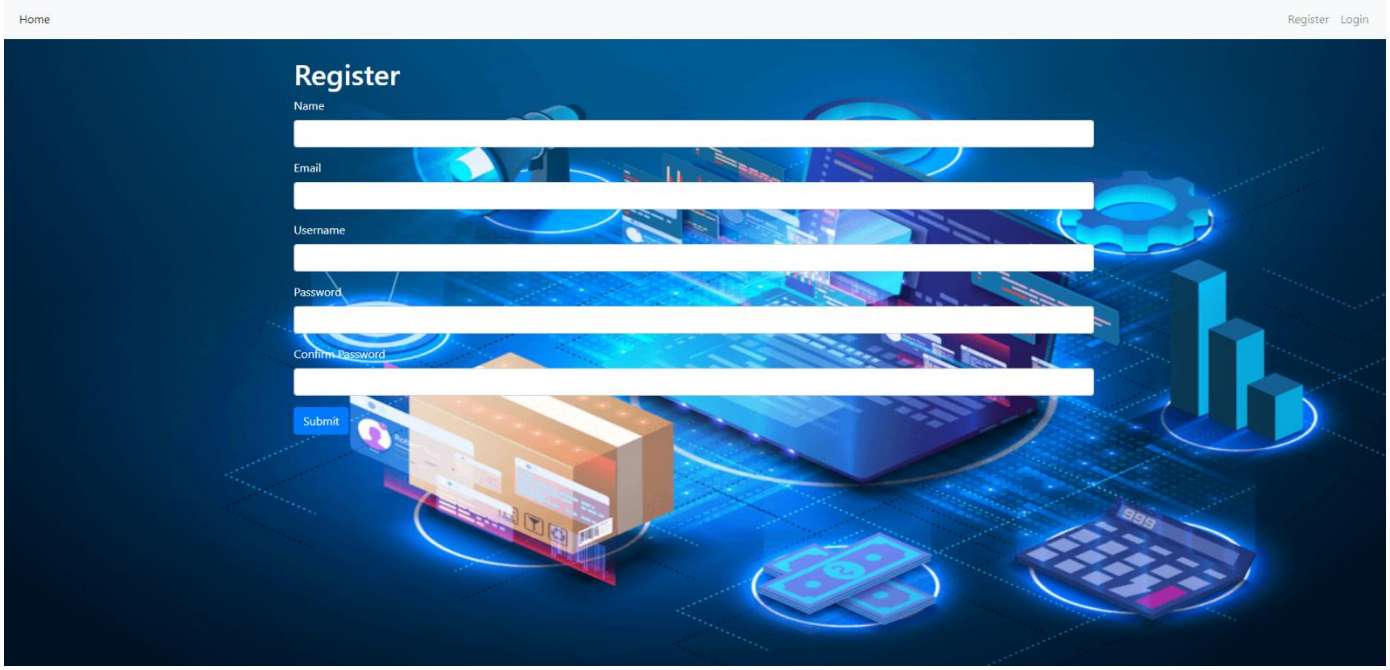
BURNDOWN CHART:

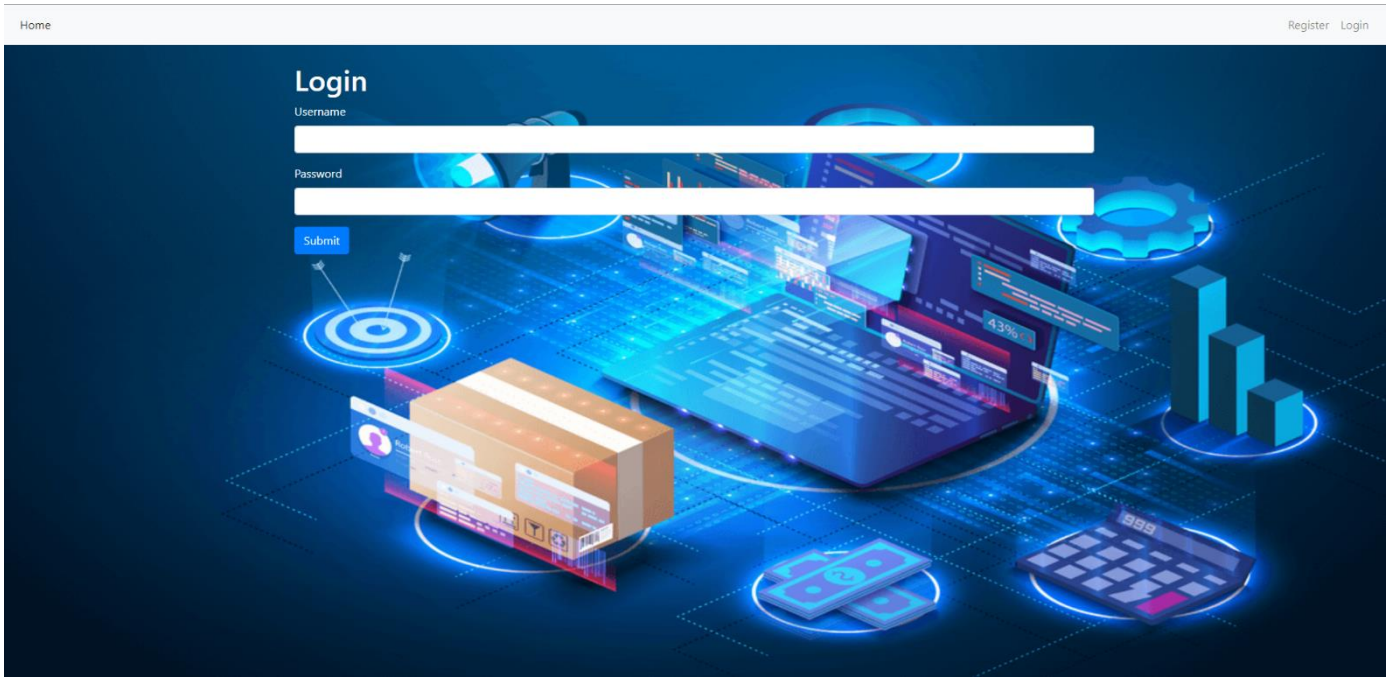
A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

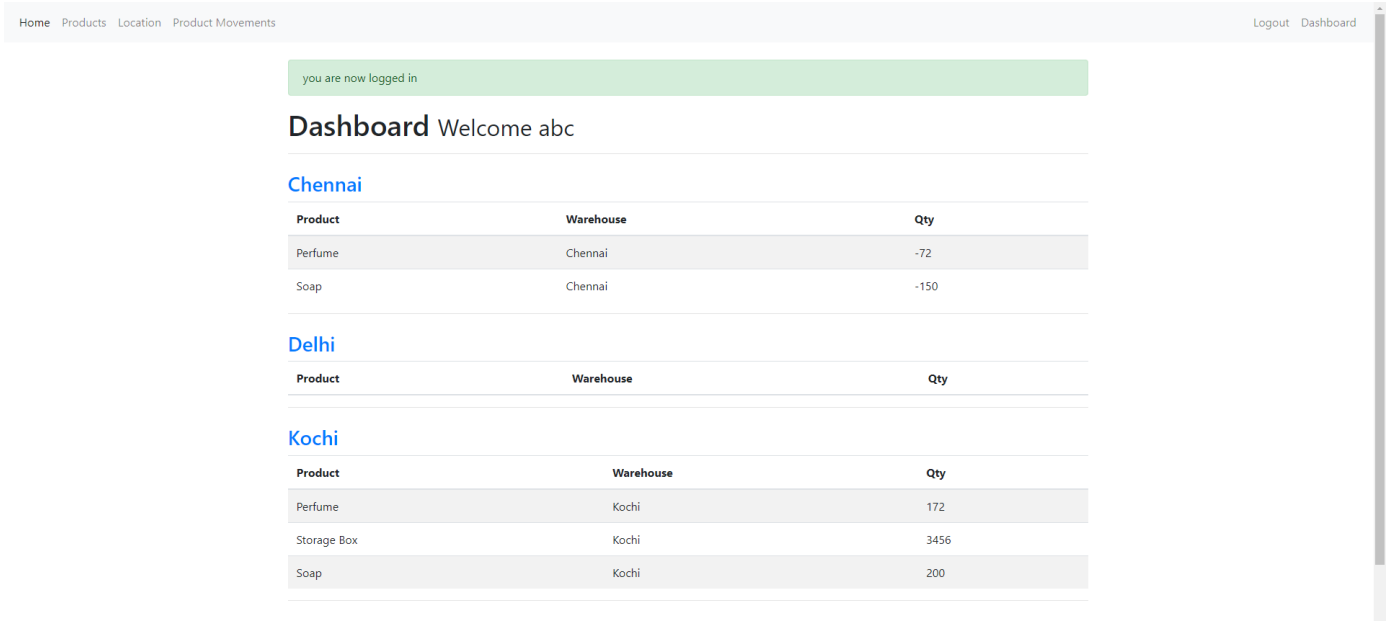


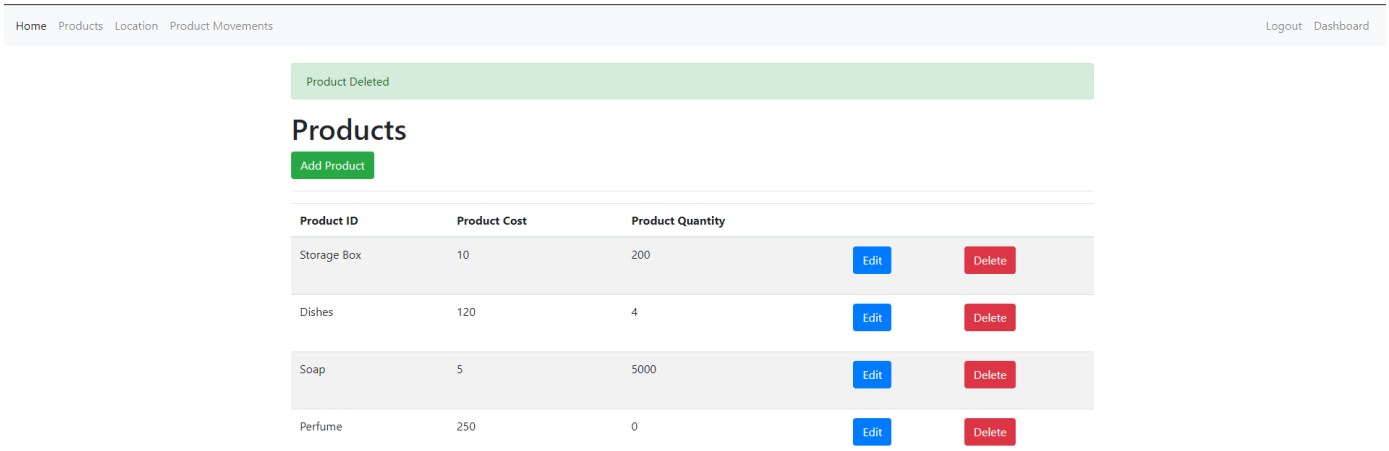
7. TESTING

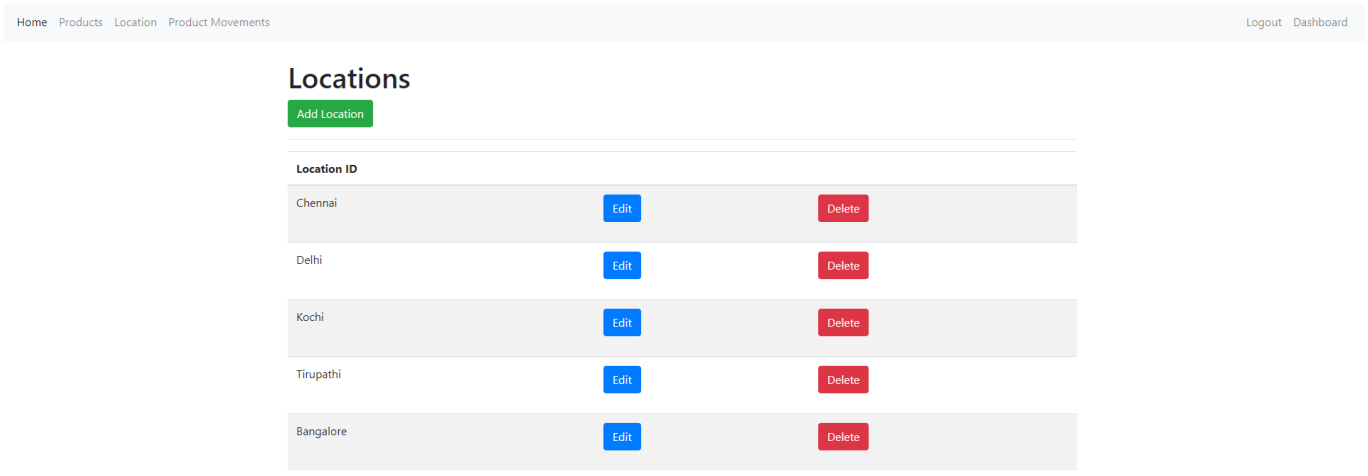

Home
Register
Login
Inventory Management System for Retailers
Register
Login

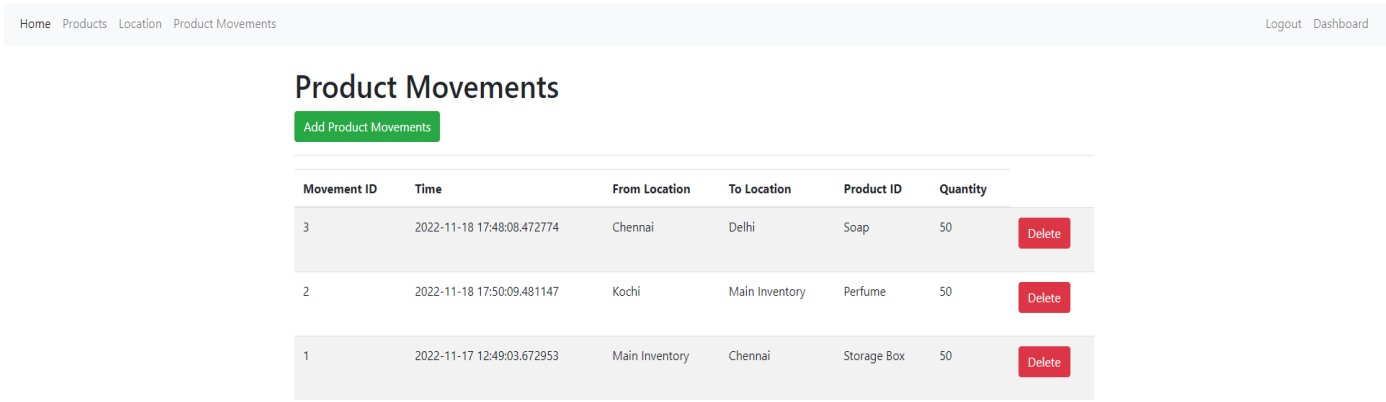

Home
Register
Login
Register
Name
Email
Username
Password
Confirm Password
Submit


Home
Register
Login
Login
Username
Password
Submit


Home
Products
Location
Product Movements
Logout
Dashboard
you are now logged in
Dashboard Welcome abc
Chennai
Product
Warehouse
Qty
Perfume
Chennai
-72
Soap
Chennai
-150
Delhi
Product
Warehouse
Qty
Kochi
Product
Warehouse
Qty
Perfume
Kochi
172
Storage Box
Kochi
3456
Soap
Kochi
200


Home Products Location Product Movements
Logout Dashboard
Product Deleted
Products
Add Product
Product ID
Product Cost
Product Quantity
Storage Box 10 200 Edit Delete
Dishes 120 4 Edit Delete
Soap 5 5000 Edit Delete
Perfume 250 0 Edit Delete


Home Products Location Product Movements
Logout Dashboard
Locations
Add Location
Location ID
Chennai Edit Delete
Delhi Edit Delete
Kochi Edit Delete
Tirupathi Edit Delete
Bangalore Edit Delete



8. RESULTS

8.1PERFORMANCE METRICS

- Accuracy

  The accuracy metric is one of the simplest Classification metrics to implement, and it can be determined as the number of correct predictions to the total number of predictions.

- Confusion Matrix

  A confusion matrix is a tabular representation of prediction outcomes of any binary classifier, which is used to describe the performance of the classification model on a set of testdata when true values are known. The confusion matrix is simple to implement, but the terminologies used in this matrix might be confusing for beginners.

9. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- Improved customer service
- Cloud-based solution
- Order Fulfillment
- Harness Customer Loyalty and Retention
- Helps move vehicles through the service bay quicker
- Mitigate Risks with Added Security
- Maximize Profit

DISADVANTAGES

- System Clash
- Reduced Physical Audits
- No solution to improve or eliminate bottlenecks in the service cycle

10. CONCLUSION

Taking proper care of our record is crucial in every business, no matter how big or little, we must understand. We must educate ourselves about the idea of effective inventory management and its applications because we can see that managers do not fully grasp it. A company's inventory management system is one of the reasons for its failure. Many customs to combat failure are present, and we can start from this point. Modern technologies can support us in managing and keeping an eyeon our inventory. We may learn, put new ideas into practice, and assess our company.

11. FUTURE SCOPE

- Collaboration with supply chain partners, coupled with a holistic approach to supply chainmanagement, will be key to effective inventory management.
- The nature of globalization will change, impacting inventory deployment decisionsdramatically.

12. APPENDIX

Source Code

app.py

```
from flask import Flask, render_template, flash, redirect, url_for, session, request, logging
from wtforms import Form, StringField, TextAreaField, PasswordField, validators, SelectField,
IntegerField
import ibm_db
from passlib.hash import sha256_crypt
from functools import wraps

from sendgrid import *

#creating an app instance
app = Flask(__name__)

app.secret_key='a'

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=31321;SECURITY=SSL;SS
LServerCertificate=DigiCertGlobalRootCA.crt;UID=kzr30386;PWD=cCdMZe6DQHHZEe5D;","","")

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

#Products
@app.route('/products')
def products():
    sql = "SELECT * FROM products"
    stmt = ibm_db.prepare(conn, sql)
    result=ibm_db.execute(stmt)

    products=[]
    row = ibm_db.fetch_assoc(stmt)
    while(row):
        products.append(row)
        row = ibm_db.fetch_assoc(stmt)
    products=tuple(products)
    #print(products)

    if result>0:
        return render_template('products.html', products = products)
    else:
        msg='No products found'
        return render_template('products.html', msg=msg)
```

```
#Locations
@app.route('/locations')
def locations():

    sql = "SELECT * FROM locations"
    stmt = ibm_db.prepare(conn, sql)
    result=ibm_db.execute(stmt)

    locations=[]
    row = ibm_db.fetch_assoc(stmt)
    while(row):
        locations.append(row)
        row = ibm_db.fetch_assoc(stmt)
    locations=tuple(locations)
    #print(locations)

    if result>0:
        return render_template('locations.html', locations = locations)
    else:
        msg='No locations found'
        return render_template('locations.html', msg=msg)

#Product Movements
@app.route('/product_movements')
def product_movements():

    sql = "SELECT * FROM productmovements"
    stmt = ibm_db.prepare(conn, sql)
    result=ibm_db.execute(stmt)

    movements=[]
    row = ibm_db.fetch_assoc(stmt)
    while(row):
        movements.append(row)
        row = ibm_db.fetch_assoc(stmt)
    movements=tuple(movements)
    #print(movements)

    if result>0:
        return render_template('product_movements.html', movements = movements)
    else:
        msg='No product movements found'
        return render_template('product_movements.html', msg=msg)

#Register Form Class
class RegisterForm(Form):
    name = StringField('Name', [validators.Length(min=1, max=50)])
    username = StringField('Username', [validators.Length(min=1, max=25)])
```

```
    email = StringField('Email', [validators.length(min=6, max=50)])
    password = PasswordField('Password', [
        validators.DataRequired(),
        validators.EqualTo('confirm', message='Passwords do not match')
    ])
    confirm = PasswordField('Confirm Password')

#user register
@app.route('/register', methods=['GET','POST'])
def register():
    form = RegisterForm(request.form)
    if request.method == 'POST' and form.validate():
        name = form.name.data
        email = form.email.data
        username = form.username.data
        password = sha256_crypt.encrypt(str(form.password.data))

        sql1="INSERT INTO users(name, email, username, password) VALUES(?,?,?,?)"
        stmt1 = ibm_db.prepare(conn, sql1)
        ibm_db.bind_param(stmt1,1,name)
        ibm_db.bind_param(stmt1,2,email)
        ibm_db.bind_param(stmt1,3,username)
        ibm_db.bind_param(stmt1,4,password)
        ibm_db.execute(stmt1)
        #for flash messages taking parameter and the category of message to be flashed
        flash("You are now registered and can log in", "success")

        #when registration is successful redirect to home
        return redirect(url_for('login'))
    return render_template('register.html', form = form)


#User login
@app.route('/login', methods = ['GET', 'POST'])
def login():
    if request.method == 'POST':
        #Get form fields
        username = request.form['username']
        password_candidate = request.form['password']

        sql1="Select * from users where username = ?"
        stmt1 = ibm_db.prepare(conn, sql1)
        ibm_db.bind_param(stmt1,1,username)
        result=ibm_db.execute(stmt1)
        d=ibm_db.fetch_assoc(stmt1)
        if result > 0:
            #Get the stored hash
            data = d
```

```
            password = data['PASSWORD']

            #compare passwords
            if sha256_crypt.verify(password_candidate, password):
                #Passed
                session['logged_in'] = True
                session['username'] = username

                flash("you are now logged in","success")
                return redirect(url_for('dashboard'))
            else:
                error = 'Invalid Login'
                return render_template('login.html', error=error)
            #Close connection
            cur.close()
        else:
            error = 'Username not found'
            return render_template('login.html', error=error)
    return render_template('login.html')

#check if user logged in
def is_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            return f(*args, **kwargs)
        else:
            flash('Unauthorized, Please login','danger')
            return redirect(url_for('login'))
    return wrap

#Logout
@app.route('/logout')
@is_logged_in
def logout():
    session.clear()
    flash("You are now logged out", "success")
    return redirect(url_for('login'))

#Dashboard
@app.route('/dashboard')
@is_logged_in
def dashboard():
    sql2="SELECT product_id, location_id, qty FROM product_balance"
    sql3="SELECT location_id FROM locations"
    stmt2 = ibm_db.prepare(conn, sql2)
    stmt3 = ibm_db.prepare(conn, sql3)
```

```
    result=ibm_db.execute(stmt2)
    ibm_db.execute(stmt3)


    products=[]
    row = ibm_db.fetch_assoc(stmt2)
    while(row):
        products.append(row)
        row = ibm_db.fetch_assoc(stmt2)
    products=tuple(products)

    locations=[]
    row2 = ibm_db.fetch_assoc(stmt3)
    while(row2):
        locations.append(row2)
        row2 = ibm_db.fetch_assoc(stmt3)
    locations=tuple(locations)

    locs = []
    for i in locations:
        locs.append(list(i.values())[0])

    if result>0:
        return render_template('dashboard.html', products = products, locations = locs)
    else:
        msg='No products found'
        return render_template('dashboard.html', msg=msg)

#Product Form Class
class ProductForm(Form):
    product_id = StringField('Product ID', [validators.Length(min=1, max=200)])
    product_cost = StringField('Product Cost', [validators.Length(min=1, max=200)])
    product_num = StringField('Product Num', [validators.Length(min=1, max=200)])

#Add Product
@app.route('/add_product', methods=['GET', 'POST'])
@is_logged_in
def add_product():
    form = ProductForm(request.form)
    if request.method == 'POST' and form.validate():
        product_id = form.product_id.data
        product_cost = form.product_cost.data
        product_num = form.product_num.data

        sql1="INSERT INTO products(product_id, product_cost, product_num) VALUES(?,?,?)"
        stmt1 = ibm_db.prepare(conn, sql1)
        ibm_db.bind_param(stmt1,1,product_id)
        ibm_db.bind_param(stmt1,2,product_cost)
```

```
        ibm_db.bind_param(stmt1,3,product_num)

        ibm_db.execute(stmt1)

        flash("Product Added", "success")

        return redirect(url_for('products'))

    return render_template('add_product.html', form=form)

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

    print(product)
    #Get form
    form = ProductForm(request.form)

    #populate product form fields
    form.product_id.data = product['PRODUCT_ID']
    form.product_cost.data = str(product['PRODUCT_COST'])
    form.product_num.data = str(product['PRODUCT_NUM'])

    if request.method == 'POST' and form.validate():
        product_id = request.form['product_id']
        product_cost = request.form['product_cost']
        product_num = request.form['product_num']

        sql2="UPDATE products SET product_id=?,product_cost=?,product_num=? WHERE
product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,product_id)
        ibm_db.bind_param(stmt2,2,product_cost)
        ibm_db.bind_param(stmt2,3,product_num)
        ibm_db.bind_param(stmt2,4,id)
        ibm_db.execute(stmt2)

        flash("Product Updated", "success")

        return redirect(url_for('products'))

    return render_template('edit_product.html', form=form)
```

```
#Delete Product
@app.route('/delete_product/<string:id>', methods=['POST'])
@is_logged_in
def delete_product(id):

    sql2="DELETE FROM products WHERE product_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,id)
    ibm_db.execute(stmt2)

    flash("Product Deleted", "success")

    return redirect(url_for('products'))

#Location Form Class
class LocationForm(Form):
    location_id = StringField('Location ID', [validators.Length(min=1, max=200)])

#Add Location
@app.route('/add_location', methods=['GET', 'POST'])
@is_logged_in
def add_location():
    form = LocationForm(request.form)
    if request.method == 'POST' and form.validate():
        location_id = form.location_id.data

        sql2="INSERT into locations VALUES(?)"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,location_id)
        ibm_db.execute(stmt2)

        flash("Location Added", "success")

        return redirect(url_for('locations'))

    return render_template('add_location.html', form=form)

#Edit Location
@app.route('/edit_location/<string:id>', methods=['GET', 'POST'])
@is_logged_in
def edit_location(id):

    sql2="SELECT * FROM locations where location_id = ?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,id)
    result=ibm_db.execute(stmt2)
    location=ibm_db.fetch_assoc(stmt2)
    #Get form
    form = LocationForm(request.form)
```

```
    print(location)

    #populate article form fields
    form.location_id.data = location['LOCATION_ID']

    if request.method == 'POST' and form.validate():
        location_id = request.form['location_id']

        sql2="UPDATE locations SET location_id=? WHERE location_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,location_id)
        ibm_db.bind_param(stmt2,2,id)
        ibm_db.execute(stmt2)

        flash("Location Updated", "success")

        return redirect(url_for('locations'))

    return render_template('edit_location.html', form=form)

#Delete Location
@app.route('/delete_location/<string:id>', methods=['POST'])
@is_logged_in
def delete_location(id):
    sql2="DELETE FROM locations WHERE location_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,id)
    ibm_db.execute(stmt2)

    flash("Location Deleted", "success")

    return redirect(url_for('locations'))

#Product Movement Form Class
class ProductMovementForm(Form):
    from_location = SelectField('From Location', choices=[])
    to_location = SelectField('To Location', choices=[])
    product_id = SelectField('Product ID', choices=[])
    qty = IntegerField('Quantity')

class CustomError(Exception):
    pass

#Add Product Movement
@app.route('/add_product_movements', methods=['GET', 'POST'])
@is_logged_in
def add_product_movements():
    form = ProductMovementForm(request.form)
```

```
sql2="SELECT product_id FROM products"
sql3="SELECT location_id FROM locations"
stmt2 = ibm_db.prepare(conn, sql2)
stmt3 = ibm_db.prepare(conn, sql3)

result=ibm_db.execute(stmt2)
ibm_db.execute(stmt3)


products=[]
row = ibm_db.fetch_assoc(stmt2)
while(row):
    products.append(row)
    row = ibm_db.fetch_assoc(stmt2)
products=tuple(products)

locations=[]
row2 = ibm_db.fetch_assoc(stmt3)
while(row2):
    locations.append(row2)
    row2 = ibm_db.fetch_assoc(stmt3)
locations=tuple(locations)

prods = []
for p in products:
    prods.append(list(p.values())[0])

locs = []
for i in locations:
    locs.append(list(i.values())[0])

form.from_location.choices = [(l,l) for l in locs]
form.from_location.choices.append(("Main Inventory","Main Inventory"))
form.to_location.choices = [(l,l) for l in locs]
form.to_location.choices.append(("Main Inventory","Main Inventory"))
form.product_id.choices = [(p,p) for p in prods]

if request.method == 'POST' and form.validate():
    from_location = form.from_location.data
    to_location = form.to_location.data
    product_id = form.product_id.data
    qty = form.qty.data


    if from_location==to_location:
        raise CustomError("Please Give different From and To Locations!!")
```

```
    elif from_location=="Main Inventory":
        sql2="SELECT * from product_balance where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,to_location)
        ibm_db.bind_param(stmt2,2,product_id)
        result=ibm_db.execute(stmt2)
        result=ibm_db.fetch_assoc(stmt2)
        print("-----------------")
        print(result)
        print("-----------------")
        app.logger.info(result)
        if result!=False:
            if(len(result))>0:
                Quantity = result["QTY"]
                q = Quantity + qty

                sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
                stmt2 = ibm_db.prepare(conn, sql2)
                ibm_db.bind_param(stmt2,1,q)
                ibm_db.bind_param(stmt2,2,to_location)
                ibm_db.bind_param(stmt2,3,product_id)
                ibm_db.execute(stmt2)

                sql2="INSERT into productmovements(from_location, to_location, product_id, qty)
VALUES(?, ?, ?, ?)"
                stmt2 = ibm_db.prepare(conn, sql2)
                ibm_db.bind_param(stmt2,1,from_location)
                ibm_db.bind_param(stmt2,2,to_location)
                ibm_db.bind_param(stmt2,3,product_id)
                ibm_db.bind_param(stmt2,4,qty)
                ibm_db.execute(stmt2)
        else:

            sql2="INSERT into product_balance(product_id, location_id, qty) values(?, ?, ?)"
            stmt2 = ibm_db.prepare(conn, sql2)
            ibm_db.bind_param(stmt2,1,product_id)
            ibm_db.bind_param(stmt2,2,to_location)
            ibm_db.bind_param(stmt2,3,qty)
            ibm_db.execute(stmt2)

            sql2="INSERT into productmovements(from_location, to_location, product_id, qty)
VALUES(?, ?, ?, ?)"
            stmt2 = ibm_db.prepare(conn, sql2)
            ibm_db.bind_param(stmt2,1,from_location)
            ibm_db.bind_param(stmt2,2,to_location)
            ibm_db.bind_param(stmt2,3,product_id)
            ibm_db.bind_param(stmt2,4,qty)
            ibm_db.execute(stmt2)
```

```
        sql = "select product_num from products where product_id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,product_id)
        current_num=ibm_db.execute(stmt)
        current_num = ibm_db.fetch_assoc(stmt)

        sql2="Update products set product_num=? where product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,current_num['PRODUCT_NUM']-qty)
        ibm_db.bind_param(stmt2,2,product_id)
        ibm_db.execute(stmt2)

        alert_num=current_num['PRODUCT_NUM']-qty

        if(alert_num<=0):
            alert("Please update the quantity of the product {}, Atleast {} number of pieces must be
added to finish the pending Product Movements!".format(product_id,-alert_num))

    elif to_location=="Main Inventory":
        sql2="SELECT * from product_balance where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,from_location)
        ibm_db.bind_param(stmt2,2,product_id)
        result=ibm_db.execute(stmt2)
        result=ibm_db.fetch_assoc(stmt2)

        app.logger.info(result)
        if result!=False:
            if(len(result))>0:
                Quantity = result["QTY"]
                q = Quantity - qty

                sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
                stmt2 = ibm_db.prepare(conn, sql2)
                ibm_db.bind_param(stmt2,1,q)
                ibm_db.bind_param(stmt2,2,to_location)
                ibm_db.bind_param(stmt2,3,product_id)
                ibm_db.execute(stmt2)

                sql2="INSERT into productmovements(from_location, to_location, product_id, qty)
VALUES(?, ?, ?, ?)"
                stmt2 = ibm_db.prepare(conn, sql2)
                ibm_db.bind_param(stmt2,1,from_location)
                ibm_db.bind_param(stmt2,2,to_location)
                ibm_db.bind_param(stmt2,3,product_id)
                ibm_db.bind_param(stmt2,4,qty)
```

```
                ibm_db.execute(stmt2)

                flash("Product Movement Added", "success")


                sql = "select product_num from products where product_id=?"
                stmt = ibm_db.prepare(conn, sql)
                ibm_db.bind_param(stmt,1,product_id)
                current_num=ibm_db.execute(stmt)
                current_num = ibm_db.fetch_assoc(stmt)

                sql2="Update products set product_num=? where product_id=?"
                stmt2 = ibm_db.prepare(conn, sql2)
                ibm_db.bind_param(stmt2,1,current_num['PRODUCT_NUM']+qty)
                ibm_db.bind_param(stmt2,2,product_id)
                ibm_db.execute(stmt2)

                alert_num=q
                if(alert_num<=0):
                   alert("Please Add {} number of {} to {} warehouse!".format(-
q,product_id,from_location))
            else:
               raise CustomError("There is no product named {} in
{}.".format(product_id,from_location))



        else: #will be executed if both from_location and to_location are specified
            f=0
            sql = "SELECT * from product_balance where location_id=? and product_id=?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt,1,from_location)
            ibm_db.bind_param(stmt,2,product_id)
            result=ibm_db.execute(stmt)
            result = ibm_db.fetch_assoc(stmt)

            if result!=False:
               if(len(result))>0:
                  Quantity = result["QTY"]
                  q = Quantity - qty

                  sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
                  stmt2 = ibm_db.prepare(conn, sql2)
                  ibm_db.bind_param(stmt2,1,q)
                  ibm_db.bind_param(stmt2,2,from_location)
                  ibm_db.bind_param(stmt2,3,product_id)
                  ibm_db.execute(stmt2)
                  f=1
```

```
                alert_num=q
                if(alert_num<=0):
                    alert("Please Add {} number of {} to {} warehouse!".format(-
q,product_id,from_location))

            else:
                raise CustomError("There is no product named {} in
{}.".format(product_id,from_location))

            if(f==1):
                sql = "SELECT * from product_balance where location_id=? and product_id=?"
                stmt = ibm_db.prepare(conn, sql)
                ibm_db.bind_param(stmt,1,to_location)
                ibm_db.bind_param(stmt,2,product_id)
                result=ibm_db.execute(stmt)
                result = ibm_db.fetch_assoc(stmt)

                if result!=False:
                    if(len(result))>0:
                        Quantity = result["QTY"]
                        q = Quantity + qty

                        sql2="UPDATE product_balance set qty=? where location_id=? and product_id=?"
                        stmt2 = ibm_db.prepare(conn, sql2)
                        ibm_db.bind_param(stmt2,1,q)
                        ibm_db.bind_param(stmt2,2,to_location)
                        ibm_db.bind_param(stmt2,3,product_id)
                        ibm_db.execute(stmt2)

                else:

                    sql2="INSERT into product_balance(product_id, location_id, qty) values(?, ?, ?)"
                    stmt2 = ibm_db.prepare(conn, sql2)
                    ibm_db.bind_param(stmt2,1,product_id)
                    ibm_db.bind_param(stmt2,2,to_location)
                    ibm_db.bind_param(stmt2,3,qty)
                    ibm_db.execute(stmt2)
                sql2="INSERT into productmovements(from_location, to_location, product_id, qty)
VALUES(?, ?, ?, ?)"
                stmt2 = ibm_db.prepare(conn, sql2)
                ibm_db.bind_param(stmt2,1,from_location)
                ibm_db.bind_param(stmt2,2,to_location)
                ibm_db.bind_param(stmt2,3,product_id)
                ibm_db.bind_param(stmt2,4,qty)
                ibm_db.execute(stmt2)

                flash("Product Movement Added", "success")
```

```
            render_template('products.html',form=form)


            return redirect(url_for('product_movements'))

        return render_template('add_product_movements.html', form=form)

    #Delete Product Movements
    @app.route('/delete_product_movements/<string:id>', methods=['POST'])
    @is_logged_in
    def delete_product_movements(id):

        sql2="DELETE FROM productmovements WHERE movement_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,id)
        ibm_db.execute(stmt2)

        flash("Product Movement Deleted", "success")

        return redirect(url_for('product_movements'))

    if __name__ == '__main__':
        app.secret_key = "secret123"
        #when the debug mode is on, we do not need to restart the server again and again
        app.run(debug=True)
```

login.html

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

        {% block body %}
        <h1>Login</h1>
        <form method="POST" action="">
            <div class="form-group">
                <label>Username</label>
                <input type="text" name="username" class="form-control"
        value={{request.form.username}}>
            </div>
            <div class="form-group">
                    <label>Password</label>
                    <input type="password" name="password" class="form-control"
        value={{request.form.password}}>
                </div>
            <p><button type="submit" class="btn btn-primary" value="Submit">Submit</button></p>
        </form>
        {% endblock %}
```

register.html

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

{% block body %}
<h1>Register</h1>
{% from "includes/_formhelpers.html" import render_field %}
<form method="POST" action="">
   <div class="form-group">
     {{render_field(form.name, class_="form-control")}}
   </div>
   <div class="form-group">
      {{render_field(form.email, class_="form-control")}}
   </div>
   <div class="form-group">
     {{render_field(form.username, class_="form-control")}}
   </div>
   <div class="form-group">
      {{render_field(form.password, class_="form-control")}}
   </div>
   <div class="form-group">
      {{render_field(form.confirm, class_="form-control")}}
   </div>
   <p><button type="submit" class="btn btn-primary" value="Submit">Submit</button></p>
</form>
{% endblock %}
```

products.html

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

{% block body %}
    <h1>Products</h1>
    <a class="btn btn-success" href="/add_product">Add Product</a>
    <hr>
    <table class="table table-striped">
       <thead>
          <tr>
             <th>Product ID</th>
             <th>Product Cost</th>
             <th>Product Quantity</th>
             <th></th>
             <th></th>
          </tr>
       </thead>
       <tbody>
            {% for product in products %}
           <tr>
            <td>{{product.PRODUCT_ID}}</td>
            <td>{{product.PRODUCT_COST}}</td>
```

```
                <td>{{product.PRODUCT_NUM}}</td>
                <td><a href="edit_product/{{product.PRODUCT_ID}}" class="btn btn-primary pull-
        right">Edit</a></td>
                <td>
                    <form action="{{url_for('delete_product', id=product.PRODUCT_ID)}}"
        method="POST">
                        <input type="hidden" name="method" value="DELETE">
                        <input type="submit" value="Delete" class="btn btn-danger">
                    </form>
                </td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        {% endblock %}
```

products_movement.html

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

        {% block body %}
            <h1>Product Movements</h1>
            <a class="btn btn-success" href="/add_product_movements">Add Product Movements</a>
            <hr>
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>Movement ID</th>
                        <th>Time</th>
                        <th>From Location</th>
                        <th>To Location</th>
                        <th>Product ID</th>
                        <th>Quantity</th>
                    </tr>
                </thead>
                <tbody>
                    {% for movement in movements %}
                    <tr>
                        <td>{{movement.MOVEMENT_ID}}</td>
                        <td>{{movement.TIME}}</td>
                        <td>{{movement.FROM_LOCATION}}</td>
                        <td>{{movement.TO_LOCATION}}</td>
                        <td>{{movement.PRODUCT_ID}}</td>
                        <td>{{movement.QTY}}</td>
                        <!--<td><a href="edit_product_movement/{{movement.MOVEMENT_ID}}"
        class="btn btn-primary pull-right">Edit</a></td>-->
                        <td>
                            <form action="{{url_for('delete_product_movements',
        id=movement.MOVEMENT_ID)}}" method="POST">
```

```
                    <input type="hidden" name="method" value="DELETE">
                    <input type="submit" value="Delete" class="btn btn-danger">
                  </form>
                </td>
              </tr>
              {% endfor %}
            </tbody>
          </table>
        {% endblock %}
```

locations.html

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

        {% block body %}
          <h1>Locations</h1>
          <a class="btn btn-success" href="/add_location">Add Location</a>
          <hr>
          <table class="table table-striped">
            <thead>
              <tr>
                <th>Location ID</th>
                <th></th>
                <th></th>
              </tr>
            </thead>
            <tbody>
                {% for location in locations %}
                <tr>
                <td>{{location.LOCATION_ID}}</td>
                <td><a href="edit_location/{{location.LOCATION_ID}}" class="btn btn-primary pull-
        right">Edit</a></td>
                <td>
                  <form action="{{url_for('delete_location', id=location.LOCATION_ID)}}"
        method="POST">
                    <input type="hidden" name="method" value="DELETE">
                    <input type="submit" value="Delete" class="btn btn-danger">
                  </form>
                </td>
              </tr>
              {% endfor %}
            </tbody>
          </table>
        {% endblock %}
```

layout.html

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

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

home.html

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

{% block body%}
<style>
    body {
      background-image: url('https://softwareauggest-blogimages.s3.ca-central-
1.amazonaws.com/blog/wp-content/uploads/2016/02/14191055/9-Top-Retail-Inventory-
Management-Software-for-SMEs-in-India-1068x578.png');
    }
    </style><br><br>
<div class="jumbotron mt-4">
    <h7 class="display-4">Inventory Management System for Retailers</h7><br><br>
    {% if session.logged_in == NULL %}
        <center><a href="/register" class="btn btn-primary btn-lg">Register</a>
        <a href="/login" class="btn btn-success btn-lg">Login</a></center>
    {% endif %}<br>
    <center><h5>Created By: Dwaraka kavyasudha, Obul Reddy , Mvs Charan , Sai
Lakshmi</h5><center>
</div>
{% endblock %}
```

edit_products.html

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

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

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

edit_products_movement.html

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

    {% block body %}
    <h1>Edit Product Movements</h1>
    {% from "includes/_formhelpers.html" import render_field %}
    <form action="" method="POST">
        <div class="form-group">
            {{ render_field(form.from_location, class_="form-control") }}
        </div>
        <div class="form-group">
            {{ render_field(form.to_location, class_="form-control") }}
        </div>
        <div class="form-group">
            {{ render_field(form.product_id, class_="form-control") }}
        </div>
        <div class="form-group">
            {{ render_field(form.qty, class_="form-control") }}
        </div>
        <p><input type="submit" value="Update" class="btn btn-primary"></p>
    </form>
    {% endblock %}
```

edit_location.html

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

    {% block body %}
    <h1>Edit </h1>
    {% from "includes/_formhelpers.html" import render_field %}
    <form action="" method="POST">
        <div class="form-group">
            {{ render_field(form.location_id, class_="form-control") }}
        </div>
        <p><input type="submit" value="Update" class="btn btn-primary"></p>
    </form>
    {% endblock %}
```

dashboard.html

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

```
{% block body %}
    <h1>Dashboard <small>Welcome {{session.username}}</small></h1>
    <hr>
     {% for location in locations %}
     <div>
      <h3 class="mt-4 text-primary" >{{location}}</h3>
      <table class="table table-striped">
        <thead>
          <tr>
            <th>Product</th>
            <th>Warehouse</th>
            <th>Qty</th>
          </tr>
        </thead>
        <tbody>
             {% for product in products %}
               {% if product.LOCATION_ID == location %}
            <tr>
             <td>{{product.PRODUCT_ID}}</td>
             <td>{{product.LOCATION_ID}}</td>
             <td>{{product.QTY}}</td>
            </tr>
               {% endif %}
            {% endfor %}
        </tbody>
      </table>
      <hr>
      </div>
   {% endfor %}
{% endblock %}
```

add_product.html

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

{% block body %}
<h1>Add Product</h1>
{% from "includes/_formhelpers.html" import render_field %}
<form action="" method="POST">
    <div class="form-group">
        {{ render_field(form.product_id, class_="form-control") }}
    </div>
    <div class="form-group">
        {{ render_field(form.product_cost, class_="form-control", type="number") }}
    </div>
    <div class="form-group">
        {{ render_field(form.product_num, class_="form-control", type="number") }}
    </div>
    <p><input type="submit" value="Add" class="btn btn-primary"></p>
```

```
</form>
{% endblock %}
```

add_product_movements.html

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

{% block body %}
<h1>Add Product Movements</h1>
{% from "includes/_formhelpers.html" import render_field %}
<form action="" method="POST">
    <div class="form-group">
        {{ render_field(form.from_location, class_="form-control") }}
    </div>
    <div class="form-group">
        {{ render_field(form.to_location, class_="form-control") }}
    </div>
    <div class="form-group">
        {{ render_field(form.product_id, class_="form-control") }}
    </div>
    <div class="form-group">
        {{ render_field(form.qty, class_="form-control", type="number") }}
    </div>
    <p><input type="submit" value="Add" class="btn btn-primary"></p>
</form>
{% endblock %}
```

add_location.html

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

{% block body %}
<h1>Add Location</h1>
{% from "includes/_formhelpers.html" import render_field %}
<form action="" method="POST">
    <div class="form-group">
        {{ render_field(form.location_id, class_="form-control") }}
    </div>
    <p><input type="submit" value="Add" class="btn btn-primary"></p>
</form>
{% endblock %}
```

form_helpers.html

```
{% macro render_field(field) %}
  {{ field.label }}
  {{ field(**kwargs)|safe }}
  {% if field.errors %}
    {% for error in field.errors %}
        <span class="help-text-inline">{{ error }}</span>
```

```
        {% endfor %}

      {% endif %}
    {% endmacro %}
```

_messages.html

```
    {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
        {% for category, message in messages %}
          <div class = "alert alert-{{ category }}">{{ message }}</div>
        {% endfor %}
      {% endif %}
    {% endwith %}

    {% if error %}
        <div class="alert alert-danger">{{error}}</div>
    {% endif %}

    {% if msg %}
        <div class="alert alert-success">{{msg}}</div>
    {% endif %}
```

_navbar.html

```
    <!-- named with underscore becuase its a partial file -->
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
       <a class="navbar-brand" href="#">IMSR</a>
       <button class="navbar-toggler" type="button" data-toggle="collapse" data-
    target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false"
    aria-label="Toggle navigation">
         <span class="navbar-toggler-icon"></span>
       </button>
       <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
          <li class="nav-item active">
            <a class="nav-link" href="/">Home</a>
          </li>
          {% if session.logged_in %}
          <li class="nav-item">
             <a class="nav-link" href="/products">Products</a>
          </li>
          {% endif %}
          {% if session.logged_in %}
          <li class="nav-item">
            <a class="nav-link" href="/locations">Location</a>
          </li>
          {% endif %}
          {% if session.logged_in %}
          <li class="nav-item">
```

```
          <a class="nav-link" href="/product_movements">Product Movements</a>
        </li>

        {% endif %}
      </ul>
      <ul class="navbar-nav navbar-right">
        {% if session.logged_in %}
          <li class="nav-item"><a class="nav-link" href="/logout">Logout</a></li>
          <li class="nav-item"><a class="nav-link" href="/dashboard">Dashboard</a></li>
        {% else %}
          <li class="nav-item"><a class="nav-link" href="/register">Register</a></li>
          <li class="nav-item"><a class="nav-link" href="/login">Login</a></li>
        {% endif %}
      </ul>
    </div>
  </nav>
```

GitHub & Project Demo Link

GitHub Link: https://github.com/IBM-EPBL/IBM-Project-27734-1660064362

Project Demo Link: https://www.loom.com/share/f49e3b10edb24d23bb1303b653d3ee8e