INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

(CLOUD APPLICATION DEVELOPMENT)

A NALAIYATHIRAN PROJECT REPORT

SUBMITTED BY

JESE OWENS A	- 610519104043
GUNADHARSHINI G	-610519104034
KALPANACHAWLA R	-610519104047
KEERTHIKA R	- 610519104052

TEAM ID: PNT2022TMID29793

PROJECT REPORT FORMAT

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

Retail businesses sell items or services to customers for their consumption,use,orpleasure. They typically sell items and services in-store but some items may be sold online or over the phone and then shipped to the customer. Examples of retail businesses include clothing, drug, grocery, and convenience store. The primary purpose of inventory management is to ensure there is enough goods or materials to meet demand without creating overstock, or excess inventory.

2. LITERATURE SURVEY

2.1 Existing problem

One of the biggest problems in the warehouse would be to track the products in the inventory that need to be shipped or transported to customers.What happens if the stocks are accessible? The deliveries reach the customers late which eventually degrades the brand's value and attracts low ratings on social media channels.43 percent of small businesses in the United States don't track inventory or do so using a manual system. (Source)Using manual inventory processes to track your inventory will end up in loads of confusion and be seen as one of the major inventory management issues too.Unable to track the exact product stock's location and no update if the product has been delivered or not leads to customer's inability to order new products with projecting in-efficiency. This gap in updates is also mainly due to the involvement of fast processes like multiple order tracking and the need for real-time inventory details.

2.2 References

A Literature Review On Models Inventory Managemnt Under Uncertainity

Serhii Ziukov

Inventories are raw materials, work-in-process goods and completely finished goods that are considered to be the portion of 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 second decade of the past century, but the interest in this scientific area is still great. Again considering the reliability of any process is an important feature in the research activities. Values of some factors are very hard to define or almost unreal. In such cases, fuzzy models of inventory management take an important place. This paper analyzes possible parameters of existing models of inventory control. An attempt is made to provide an up-to-date review of existing literature, concentrating on descriptions of the characteristics and types of inventory control model that have been developed

A Study Of Inventry Management System Case Study

Tariq Sheak

Inventory management is a challenging problem area in supply chain management. Companies need to have inventories in warehouses in order to fulfil customer demand, meanwhile these inventories have holding costs and this is frozen fund that can be lost. Therefore, the task of inventory management is to find the quantity of inventories that will fulfil the demand, avoiding overstocks. This paper presents a case study for the steel manufacturing industry (Small Scale Industry) on inventory management. The relationship between the inventory management and company performance was determined based on inventory days and return on asset (ROA) analysis. The research found that company X had a few inventory problems such as unorganized inventory arrangement, large amount of inventory days / no cycle counting and no accurate records balance due to unskilled workers. The study also proved that there was a significant relationship between return on asset (ROA) and inventory days. This paper also provides recommendation to the company and for further research.

Determinants Of Effective Inventory Management A Study Of Consumer Durable Retailer

Vibhuti Tripathi and Priyanka Kochhar

Global retailing is evolving into hi-tech business Indian retail market is poised to follow the path with large number of global players eyeing the market. 10.6% of growth Indian Retail market has experienced between the year 2010 and 2012. Revolution in Indian Retail has been brought about with the change in demographic profile, increase in income levels, urbanization, technology and globalization. Growth comes along with challenges, which retailers have to manage in order to sustain in the competitive environment. It is imperative for the retailers to offer adequate service levels to sustain. The determinants of service levels for a retailer can be linked to availability and variety which aspects can be maintained through a healthy inventory. The paper attempts to address the objective of identifying determinants of effective inventory management. An exploratory survey was conducted among 60 retailers from Allahabad, Lucknow, and New Delhi dealing with consumer durables. Data was treated with factor analysis and Regression analysis. Four Factors of Retailer Size, Supplier Relationship, Service Level and Demand Uncertainty emerged as the determinants of effective inventory

2.3 Problem Statement Definition

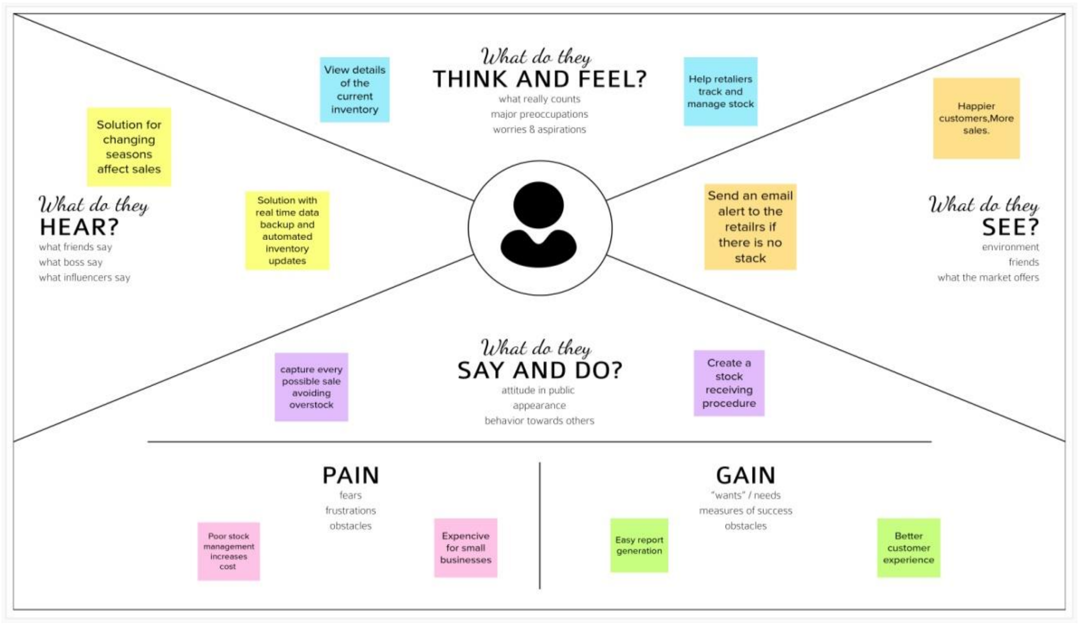
.

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 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 been developed to help retailers track and manage stocks related to their own products. The System will ask retailers to create their accounts by providing essential details. Retailers can access their accounts by logging into the application.

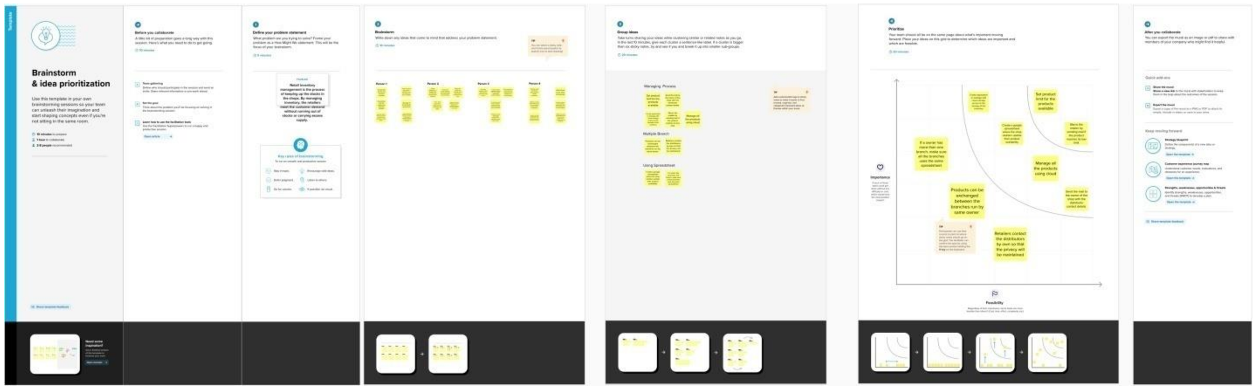
Once retailers successfully log in to the application they can update their inventory details, also users will be able to add new stock by submitting essential details related to the stock. They can view details of the current inventory. The System will automatically send an email alert to the retailers if there is no stock found in their accounts. So that they can order new stock.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas


What do they
THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
View details of the current inventory
Help retailers track and manage stock
Happier customers,More sales.
Solution for changing seasons affect sales
What do they
HEAR?
what friends say
what boss say
what influencers say
Solution with real time data backup and automated inventory updates
Send an email alert to the retailrs if there is no stack
What do they
SEE?
environment
friends
what the market offers
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
capture every possible sale avoiding overstock
Create a stock receiving procedure
PAIN
fears
frustrations
obstacles
Poor stock management increases cost
Expencive for small businesses
GAIN
"wants" / needs
measures of success
obstacles
Easy report generation
Better customer experience

3.2 Ideation & Brainstorming

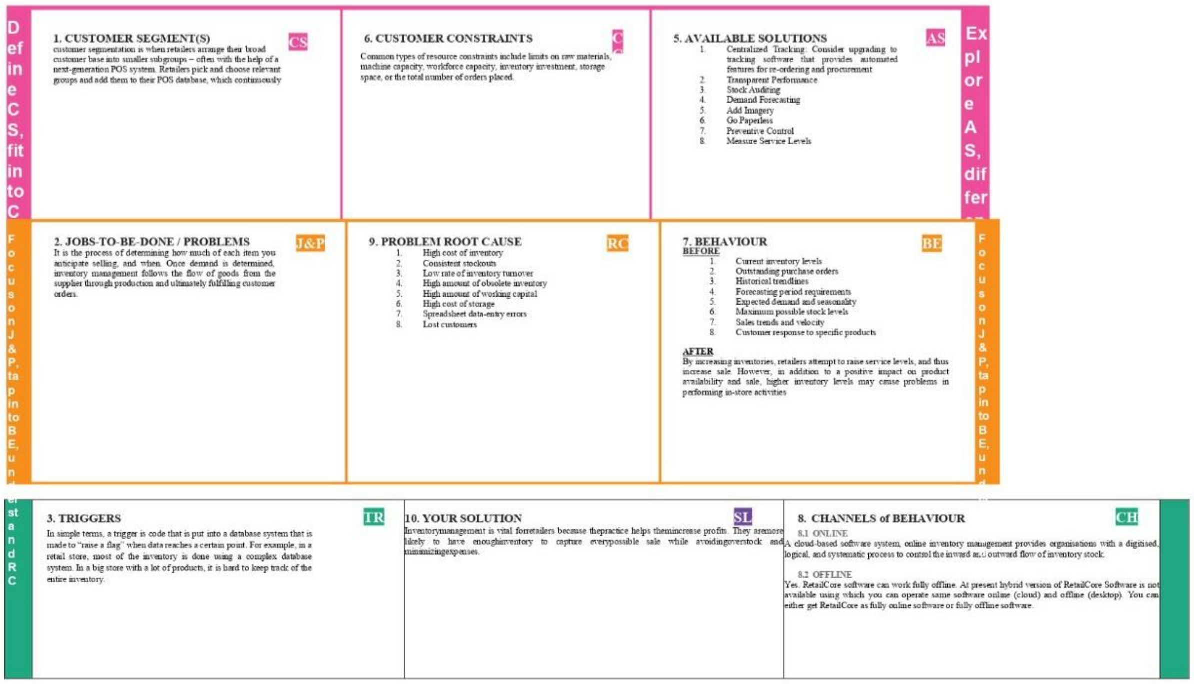


3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	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.
2.	Idea / Solution description	Inventory management software for the retail industry focuses on recording the stock levels that a retail company has within an inventory database that updates in real-time. These solutions streamline order management as they enable businesses to automatically reorder stock that is running low before it runs out, ensuring no sales opportunities are

		missed. On the other hand, this also means that overstocking can be avoided if certain products aren't selling as expected.
3.	Novelty / Uniqueness	• Applications have been developed to help retailers track and manage stocks related to their own products. • Reduces the time for managing inventory by keeping record in place.
4.	Social Impact / Customer Satisfaction	The results indicate that higher levels of inventory management practice can lead to an enhanced competitive advantage and improved organizational performance.
5.	Business Model (Revenue Model)	By providing service to the small and large scale retailers.
6.	Scalability of the Solution	Switch from spreadsheets to a cloud-based automated inventory management solution. Sync online orders, Scan barcodes, Manage multiple warehouses, Packages and Shipments. SKU Generator. Reporting & Analytics.

3.4 Problem Solution fit


Define CS, fit into C
Explore AS, differ
Focus on J&P, tap into BE, un
erstand RC
CS
CC
AS
J&P
RC
BE
TR
SL
CH

1. CUSTOMER SEGMENT(S)

customer segmentation is when retailers arrange their broad customer base into smaller subgroups – often with the help of a next-generation POS system. Retailers pick and choose relevant groups and add them to their POS database, which continuously

6. CUSTOMER CONSTRAINTS

Common types of resource constraints include limits on raw materials, machine capacity, workforce capacity, inventory investment, storage space, or the total number of orders placed.

5. AVAILABLE SOLUTIONS

1. Centralized Tracking: Consider upgrading to tracking software that provides automated features for re-ordering and procurement
2. Transparent Performance
3. Stock Auditing
4. Demand Forecasting
5. Add Imagery
6. Go Paperless
7. Preventive Control
8. Measure Service Levels

2. JOBS-TO-BE-DONE / PROBLEMS

It is the process of determining how much of each item you anticipate selling, and when. Once demand is determined, inventory management follows the flow of goods from the supplier through production and ultimately fulfilling customer orders.

9. PROBLEM ROOT CAUSE

1. High cost of inventory
2. Consistent stockouts
3. Low rate of inventory turnover
4. High amount of obsolete inventory
5. High amount of working capital
6. High cost of storage
7. Spreadsheet data-entry errors
8. Lost customers

7. BEHAVIOUR

BEFORE

1. Current inventory levels
2. Outstanding purchase orders
3. Historical trendlines
4. Forecasting period requirements
5. Expected demand and seasonality
6. Maximum possible stock levels
7. Sales trends and velocity
8. Customer response to specific products

AFTER

By increasing inventories, retailers attempt to raise service levels, and thus increase sale. However, in addition to a positive impact on product availability and sale, higher inventory levels may cause problems in performing in-store activities

3. TRIGGERS

In simple terms, a trigger is code that is put into a database system that is made to "raise a flag" when data reaches a certain point. For example, in a retail store, most of the inventory is done using a complex database system. In a big store with a lot of products, it is hard to keep track of the entire inventory.

10. YOUR SOLUTION

Inventorymanagement is vital forretailers because thepractice helps themincrease profits. They aremore likely to have enoughinventory to capture everypossible sale while avoidingoverstock and minimizingexpenses.

8. CHANNELS of BEHAVIOUR

8.1 ONLINE

A cloud-based software system, online inventory management provides organisations with a digitised, logical, and systematic process to control the inward and outward flow of inventory stock.

8.2 OFFLINE

Yes. RetailCore software can work fully offline. At present hybrid version of RetailCore Software is not available using which you can operate same software online (cloud) and offline (desktop). You can either get RetailCore as fully online software or fully offline software.

4 REQUIREMENT ANALYSIS

4.1 Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Easy Inventory Management	The software makes the process of inventory management a lot easier which saves money and time both. It assists to automate the business processes and guides to make smarter decisions.
FR-2	Inventory Reports	The software is meant to generate automated reports. You can get any report such as a low stock report, inventory validation report, inventory forecast report.
FR-3	Productivity and Efficiency	Inventory management software enables us to increase productivity and efficiency by implementing automated daily manual tasks. This will assist you to maximize the growth of your business. The software saves uncountable hours and gives the opportunity to print shipping labels, process and dispatch orders, manage stock, create and update the listing on the system.
FR-4	Avoid Stock-outs and Over-stock	When it comes to maintaining the balance sheet of inventories and its management, it is a difficult and challenging task to handle. Case of less stock leads to stock-out which not only disrupt customer relation but cause a possible loss whereas in case of over-stock its storage creates a problem. With inventory management software installed, you can set a limit for re-ordering so that stock when drops it gets automatically re-ordered.
FR-5	Inventory Alerts	Various modules trigger inventory alerts that help reduce waste, optimize inventory financials and manage customer expectations. These features include automated email or SMS messages that alert you to low inventory levels and shipment and supply chain delays. Inventory alerts offer an added level of operational control for sales forecasting, materials planning, shipping logistics and supplier management.

FR-6	Inventory Barcoding	Barcoding software helps eliminate data entry errors and automate business functions that require communication with other parts of the system. Collecting, storing and organizing digital inventory data makes inventory operations faster and more accurate.

4.2 Non-Functional requirements

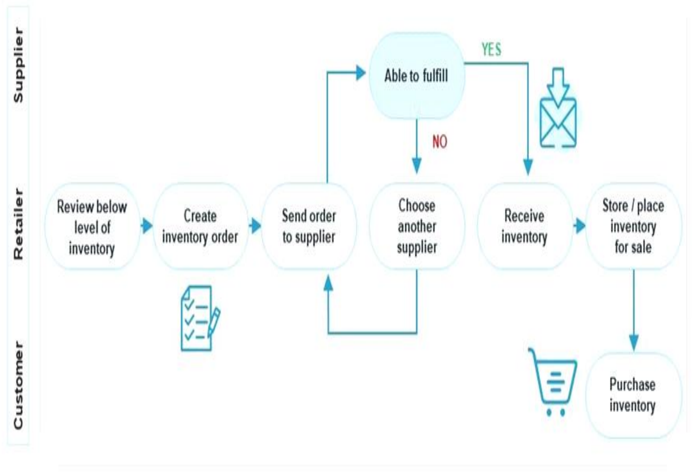
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	Usability examines the effectiveness of the inventory software. If it takes hours for your staff to learn the software, then it's not worth it. You should remember to choose a solution that simplifies inventory management.
NFR-2	**Security**	Inventory security is the process of ensuring the safety and optimum management control of stored goods. It is of central importance for optimum warehouse management because the performance of a company stands or falls with the safety and efficiency of a warehouse.
NFR-3	**Reliability**	Improving inventory reliability is important for several reasons. Your delivery reliability depends on it, but also consider the costs involved.
NFR-4	**Performance**	Inventory Performance is a measure of how effectively and efficiently inventory is used and replenished. The goal of inventory performance metrics is to compare actual on-hand dollars versus forecasted cost of goods sold.
NFR-5	**Availability**	Inventory availability refers to whether a specific item is available for customer orders. Additional information provided by retailers may include the quantity available. If the item is not in stock, the retailer may indicate its status
NFR-6	**Scalability**	The inventory management software or app you choose be able to grow as your business does? The last thing you want is to have to manually re-enter all your inventory if you outgrow your current system.

5 PROJECT DESIGN

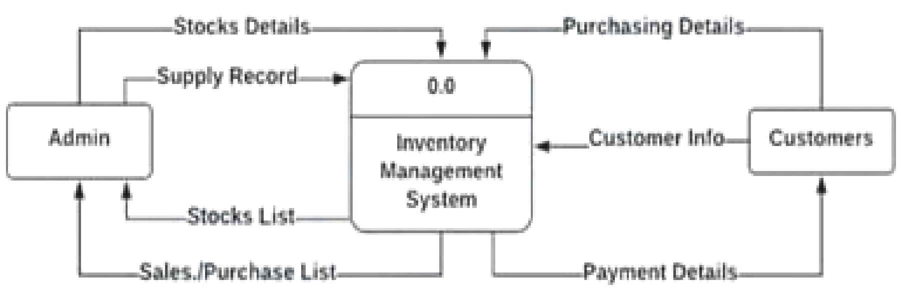
5.1 Data Flow Diagrams

The Data Flow Diagram (DFD) depicts the data flow and transformations that occur when data enters and exits a system. This DFD represents and describes the inventory management system as a whole, including input, processing, and output.

Example: (Simplified)



Example: DFD Level 0 (Industry Standard)



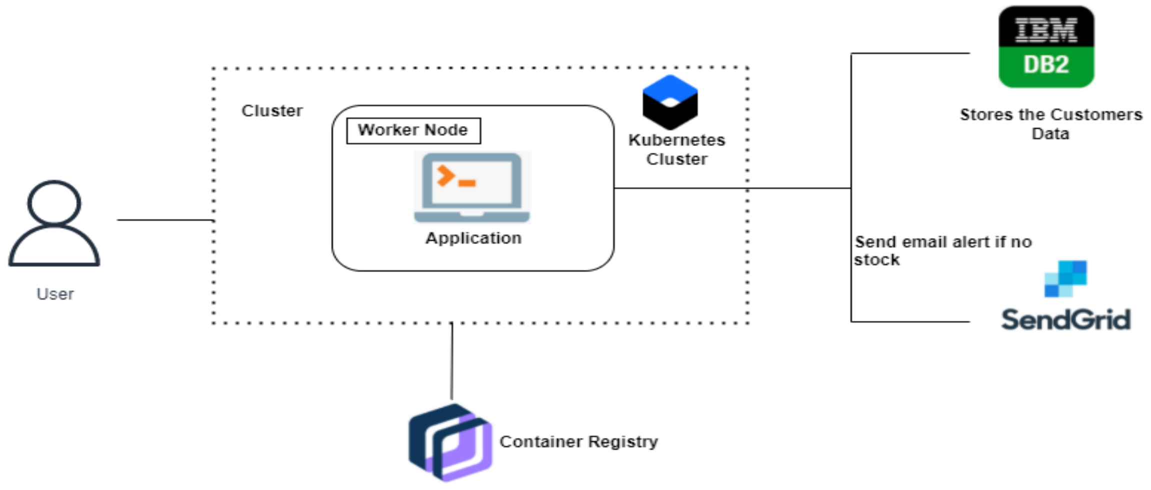
5.2 Solution & Technical Architecture

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

- Inventory management is vital for retailers because the practice helps them increase profits. They are more likely to have enough inventory to capture every possible sale while avoiding overstock and minimizing expenses.
- Inventory management software allows you to automate the tracking process and saves you a substantial amount of time. Look for software that integrates with your retail POS, so your inventory count is displayed in real time.
- Catalog data on problem stock, such as location, cost and quantity. This will help you monitor shelf life and prevent waste.

- The inventory management requires a strategic approach to operate optimally. Inventory management systems are not only used by large companies but also by small and medium-sized businesses. Choose the best stock system that suits your business. Integrating the Purchasing System with the stock system will help you plan inventory costs, control the procurement of goods, and manage orders better.

Example - Solution Architecture Diagram:



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Administration	USN-1	As a store manager,I want the available batch numbers and expiration dates that are currently valies for a product (are in use in the countr) to auto-populate a drop-down list So I don't have to manually enter the batch number for each transaction	System can receive an upload/transfer from a national ERP with product/batch/expiry Administrator can manually add/remove/edit a product/batch/expirty via CRUD screen	High	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
	Receive Stock	USN-2	As a store manager at any level, I want to receive a shipment for an order that has already been recorded in the system at a higher level So that my stock balances for all received goods are accurate	The receiving facilities stock cards for all commodities are updated (no change to sending facilities stock cards). Transaction type is receipt (not adjustment).	High	Sprint-1
		USN-3	As a store manager and/or facility in charge I want to receive a shipment of goods (including verifying received commodity counts, identifying any damaged stock, signing the POD, etc.) So that I can replenish my commodities	P1. Recipient can view shipping notice electronically (including products shipped, batches, quantity, etc), and enter actual received quantity, and any damaged stock count P2. Stock for shipment is removed from "in transit" status and transfered to the reccipient 3. Stock card(s) at the receipeint facility are incremented by the quantity received 4. Receipt date is recorded (for OTIF - may be different than date entered in the system)	Low	Sprint-2
	Stock Reports	USN-4	As a logistics manager at any level I want a list of facilities that are currently stocked out for a certain product(s) So that I can issue emergency stock		Medium	Sprint-1
	Login	USN-5	As a store room manager, I want my requisition automatically populated with current stock, stock consumed during the month, number of stock out days, etc. So that my requisition is accurate, and I save data entry time		High	Sprint-1
	Dashboard		As a store manager, I want my stock levels to be automatically updated with stock that I receive via informed push			

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
			So that my stock balance is accurate			
Customer (Web user)			As a national, regional, or district logistics manager I want predictive information on weeks of stock available at facilities (facilities that will stock out soon, based on current stock levels and historical consumption data) So that I prevent a stock out before it happens			
Customer Care Executive			As a store manager, I want to record stock on hand of all commodities at the end of my inventory period, or at any time So that we "true up" the right amount of stock			
Administrator			As a store room manager, I want my requisition automatically populated with current stock, stock consumed during the month, number of stock out days, etc. So that my requisition is accurate, and I save data entry time			

6 PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

7

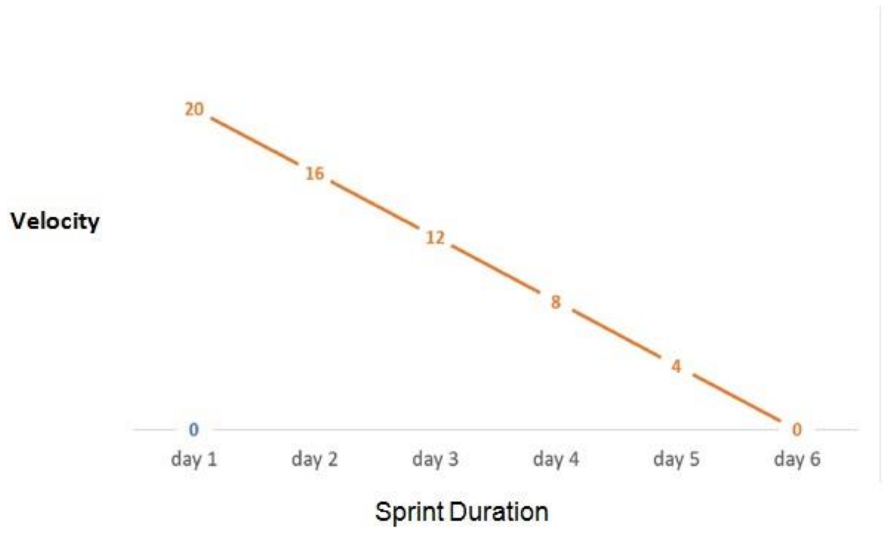
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering email, password and conforming my password.	8	High	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika
Sprint-1	Confirmation	USN-2	As a user, I will receive conformation email once I have registered for the application.	5	Low	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password.	7	High	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Dashboard	USN-4	As a User, I must able to see the products details & availability.	7	High	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika
Sprint-2	Inventory	USN-5	As a user, I should able to add the products and delete the products.	6	Medium	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika
Sprint 2		USN-6	As a User, I should get the alert on stock is unavailability.	7	Medium	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika
Sprint 3	Location	USN-7	As a User, I should able to identify stock location.	5	High	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika
Sprint 3		USN-8	As a User, I should able to using barcode.	5	Low	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika
Sprint 3		USN-9	I should be able to give response to the customers so I should implement the Chatbot.	5	High	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika
Sprint 3	Stock update	USN-10	As a user, I can add products which are not available in the dashboard to the stock list.	5		A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika
Sprint 4	Feedback	USN-11	I should able to get the Feedback from theUsers.	5	High	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika
Sprint 4	Maintenance	USN-12	As a administrator, I should able to create a centralized records of all product.	1 5	High	A.Jese Owens G.Gunadharshini R.Kalpanachawla R.Keerthika

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	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

BRUNDOWN CHART



7. CODING & SOLUTIONING

```
from __future__ import print_function
from flask import Flask, render_template, url_for, request, redirect,
session
import sqlite3 as sql
import re
import os
import ibm_db

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b1bc1829-
6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=3
2304;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=xf
y44860;PWD=ngBjSkgGQgRZxqKV;",'','')

app=Flask(__name__)
app.secret_key ='asd'

@app.route('/main')
@app.route('/')
def main():
    return render_template('main.html')

@app.route('/dashboard')
def dashboard():
    sql = "SELECT COUNT(*) FROM shops"
    stmt = ibm_db.exec_immediate(conn, sql)
    shop = ibm_db.fetch_both(stmt)
    numshop=shop["1"]
    sql = "SELECT COUNT(*) FROM products"
    stmt = ibm_db.exec_immediate(conn, sql)
    prod = ibm_db.fetch_both(stmt)
```

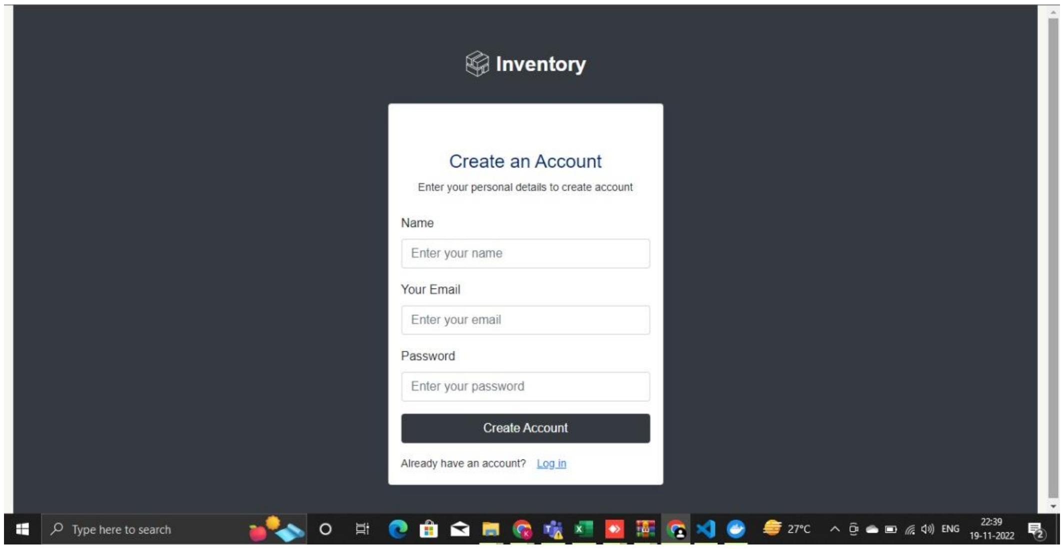
```
        numprod=prod['1']
        sql = "SELECT COUNT(*) FROM location"
        stmt = ibm_db.exec_immediate(conn, sql)
        location = ibm_db.fetch_both(stmt)
        numlocation=location['1']


        if shop | prod | location:
         return render_template('dashboard.html', numshop =
numshop,numprod = numprod ,numlocation = numlocation)


    @app.route('/products',methods = ['POST', 'GET'])
    def products():
       if request.method == 'POST':
         name = request.form['name']
         quantity = request.form['quantity']
         cost = request.form['cost']

         insert_sql = "INSERT INTO PRODUCTS VALUES (?,?,?)"
         prep_stmt = ibm_db.prepare(conn, insert_sql)
         ibm_db.bind_param(prep_stmt, 1, name)
         ibm_db.bind_param(prep_stmt, 2, quantity)
         ibm_db.bind_param(prep_stmt, 3, cost)
         ibm_db.execute(prep_stmt)

         products = []
         sql = "SELECT * FROM PRODUCTS"
         stmt = ibm_db.exec_immediate(conn, sql)
         dictionary = ibm_db.fetch_both(stmt)
         while dictionary != False:
           products.append(dictionary)
           dictionary = ibm_db.fetch_both(stmt)
         if products:
```

```
        return render_template("products.html", products = products)
    return render_template('products.html')

@app.route('/location',methods = ['POST', 'GET'])
def location():
    if request.method == 'POST':
      name = request.form['name']
      insert_sql = "INSERT INTO location VALUES (?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, name)
      ibm_db.execute(prep_stmt)

      location = []
      sql = "SELECT * FROM location"
      stmt = ibm_db.exec_immediate(conn, sql)
      dictionary = ibm_db.fetch_both(stmt)
      while dictionary != False:
        location.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
      if location:
        return render_template("location.html", location = location)
    return render_template('location.html')

@app.route('/shops',methods = ['POST', 'GET'])
def shops():
    if request.method == 'POST':
      name = request.form['name']
      status = request.form['status']

      insert_sql = "INSERT INTO SHOPS VALUES (?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, name)
      ibm_db.bind_param(prep_stmt, 2, status)
      ibm_db.execute(prep_stmt)
      shops = []
      sql = "SELECT * FROM SHOPS"
      stmt = ibm_db.exec_immediate(conn, sql)
      dictionary = ibm_db.fetch_both(stmt)
      while dictionary != False:
        shops.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)
      if shops:
        return render_template("shops.html", shops = shops)
    return render_template('shops.html')

@app.route('/delete/<name>')
def delete(name):
  sql = f"SELECT * FROM products WHERE name='{(name)}'"
  stmt = ibm_db.exec_immediate(conn, sql)
  products = ibm_db.fetch_row(stmt)
  if products:
    sql = f"DELETE FROM products WHERE name='{(name)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    products = []
    sql = "SELECT * FROM products"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
      products.append(dictionary)
      dictionary = ibm_db.fetch_both(stmt)
    if products:
      return render_template("products.html", products = products)
  return render_template("products.html")

@app.route('/dele/<name>')
def dele(name):
  sql = f"SELECT * FROM shops WHERE name='{(name)}'"
  stmt = ibm_db.exec_immediate(conn, sql)
  store = ibm_db.fetch_row(stmt)
  if store:
    sql = f"DELETE FROM shops WHERE name='{(name)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    shops = []
    sql = "SELECT * FROM shops"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
      shops.append(dictionary)
      dictionary = ibm_db.fetch_both(stmt)
    if shops:
      return render_template("shops.html", shops = shops)
  return render_template("shops.html")

@app.route('/dellocation/<name>')
def dellocation(name):
  sql = f"SELECT * FROM location WHERE name='{(name)}'"
  stmt = ibm_db.exec_immediate(conn, sql)
  location = ibm_db.fetch_row(stmt)
  if location:
    sql = f"DELETE FROM location WHERE name='{(name)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    location = []
    sql = "SELECT * FROM location"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
      location.append(dictionary)
      dictionary = ibm_db.fetch_both(stmt)
    if location:
      return render_template("location.html",  location =  location)
  return render_template("location.html")
```

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

@app.route('/user/<id>')
def user_info(id):
    with sql.connect('inventorymanagement.db') as con:
        con.row_factory=sql.Row
        cur =con.cursor()
        cur.execute(f'SELECT * FROM register WHERE email="{id}"')
        user = cur.fetchall()
    return render_template("user_info.html", user=user[0])

@app.route('/signin',methods =['GET', 'POST'])
def signin():
    global userid
    msg = ''
    if request.method == 'POST' :
        un = request.form['username']
        pd = request.form['password']
        sql = "SELECT * FROM register WHERE username =? AND
password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,un)
        ibm_db.bind_param(stmt,2,pd)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print (account)
        if account:
            session['loggedin'] = True
            session['id'] = account['USERNAME']
            userid=  account['USERNAME']
            session['username'] = account['USERNAME']
            return render_template('dashboard.html', msg = msg)
        else:
            msg = 'Incorrect username / password !'
    return render_template('signin.html', msg = msg)


@app.route('/signup', methods=['POST','GET'])
def signup():
    msg=''
    if request.method == "POST":
        username=request.form['username']
        email=request.form['email']
        pw=request.form['password']
        sql='SELECT * FROM register WHERE email =?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,email)
        ibm_db.execute(stmt)
        acnt=ibm_db.fetch_assoc(stmt)
        print(acnt)

        if acnt:
            msg='Account already exits!!'

        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg='Please enter the avalid email address'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg='name must contain only character and number'
        else:
            insert_sql='INSERT INTO register VALUES (?,?,?)'
            pstmt=ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt,1,username)
            ibm_db.bind_param(pstmt,2,email)
            ibm_db.bind_param(pstmt,3,pw)
            ibm_db.execute(pstmt)
            msg='You have successfully registered click signin!!'
            return render_template("signin.html")


        elif request.method == 'POST':
            msg="fill out the form first!"
        return render_template("signup.html",msg=msg)


    """ import sendgrid
    from sendgrid.helpers.mail import *


    sg = sendgrid.SendGridAPIClient(api_key='SENDGRID_API_KEY')
    from_email = Email("110119106302@smartinternz.com")
    to_email = To("110119106302@aalimec.ac.in")
    subject = "Sending with SendGrid is Fun"
    content = Content("text/plain", "and easy to do anywhere, even with
Python")
    mail = Mail(from_email, to_email, subject, content)
    response = sg.client.mail.send.post(request_body=mail.get())
    print(response.status_code)
    print(response.body)
    print(response.headers) """




    if __name__ == '__main__':
        #app.run(debug=True)
      app.run(host='0.0.0.0')
```

8.RESULT

REGISTER PAGE



Inventory

Create an Account

Enter your personal details to create account

Name

Enter your name

Your Email

Enter your email

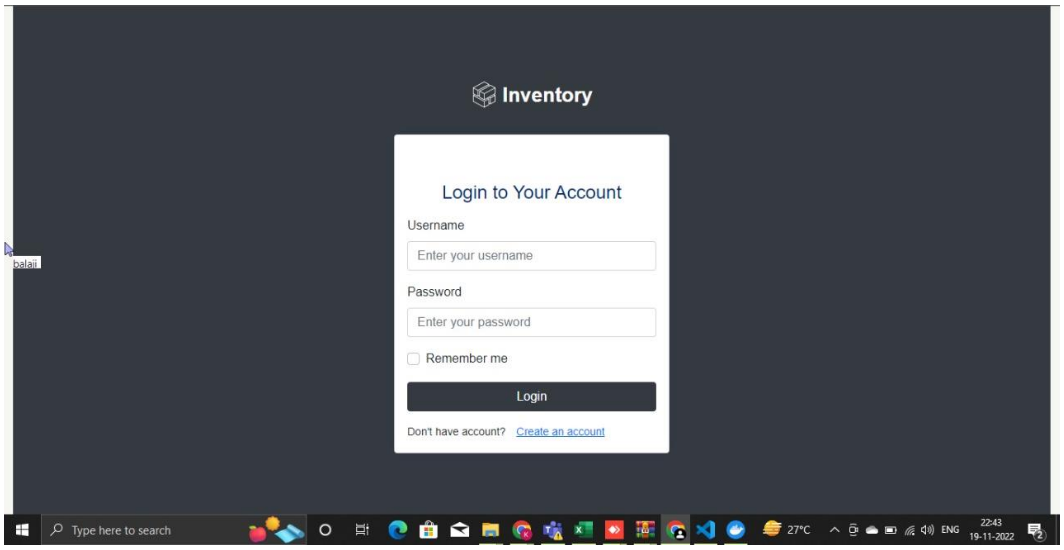
Password

Enter your password

Create Account

Already have an account? Log in

LOGIN PAGE



Inventory

Login to Your Account

Username

Enter your username

Password

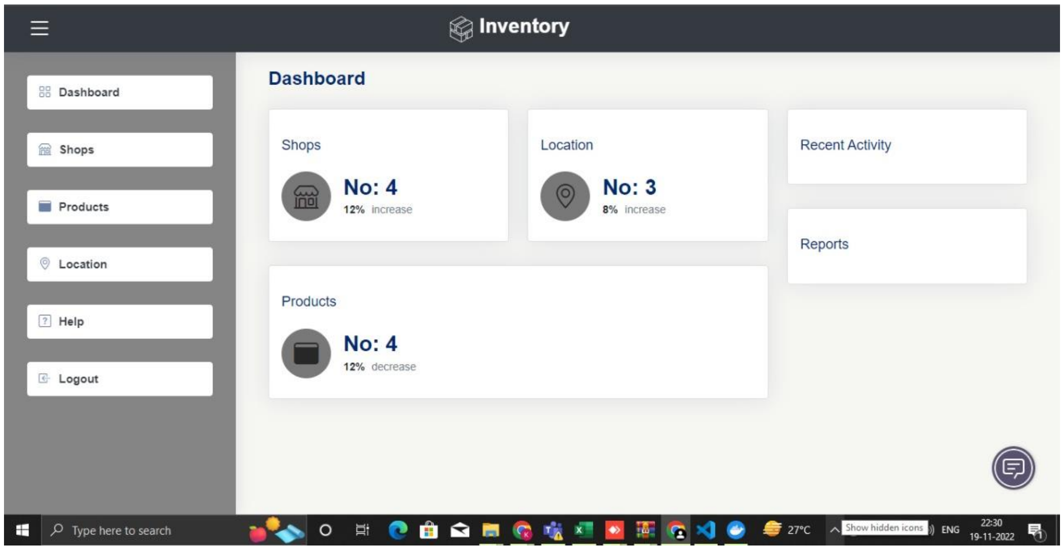
Enter your password

Remember me

Login

Don't have account? Create an account

DASHBOARD



Dashboard

Shops
No: 4
12% increase

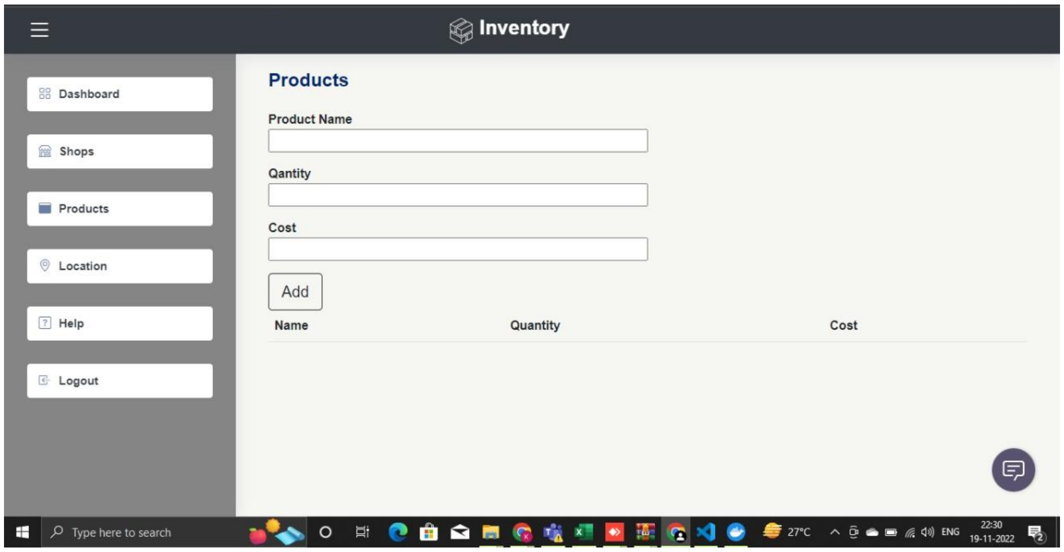
Location
No: 3
8% increase

Recent Activity

Reports

Products
No: 4
12% decrease

PRODUCTS



Products

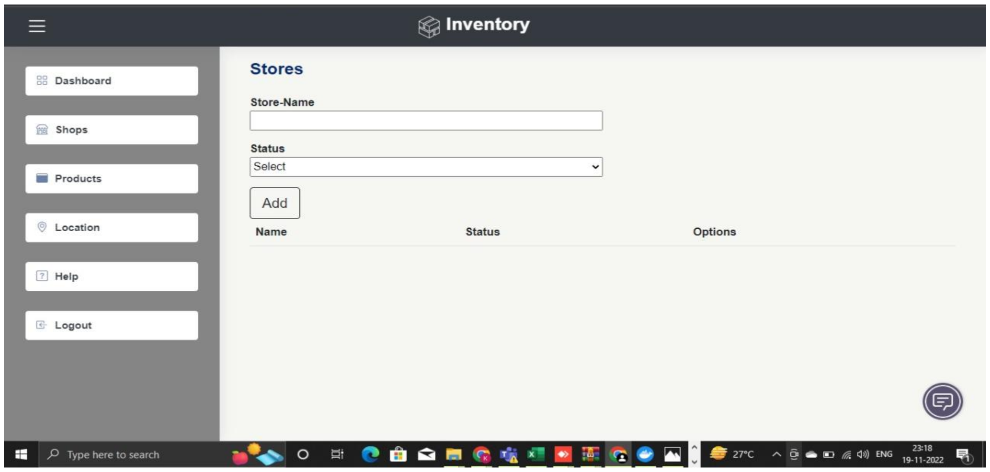
Product Name

Qantity

Cost

Name	Quantity	Cost

SHOPS



Stores

Store-Name

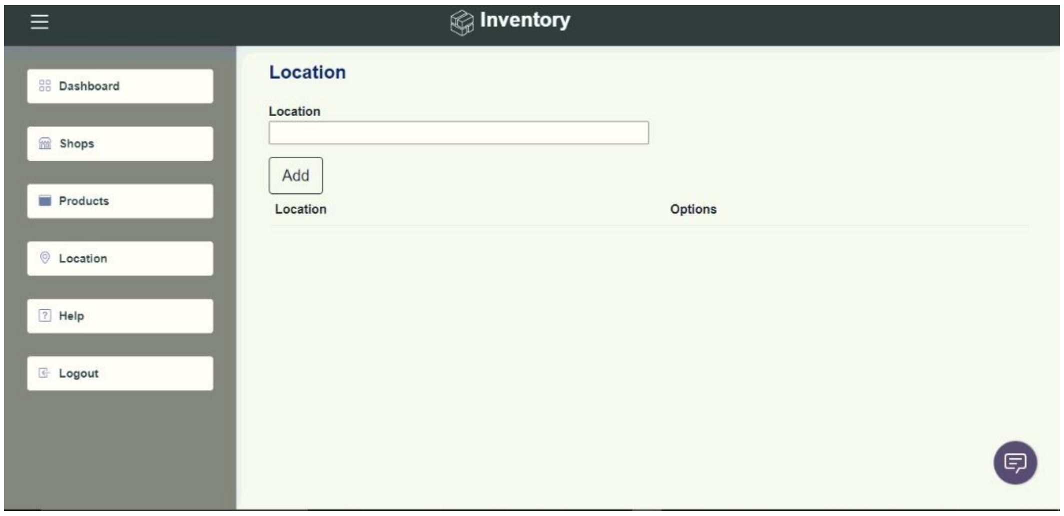
Status

Select

Add

Name	Status	Options

LOCATION



Location

Location

Add

Location	Options

9 ADVANTAGES AND DISADVANTAGES

Advantages:

1. It helps to maintain the right amount of stocks

2. It leads to a more organized warehouse

3. It saves time and money

4. A well-structured inventory management system leads to improved customer retention

5. Reduction in **holding costs**

Disadvantages:

1. Holding inventory can result to a greater risk of loss to devaluation

2. Even with an efficient inventory management method, you can control but not eliminate business risk

3. The control of inventory is complex because of the many functions it performs. It should thus be viewed as a shared responsibility.

4. some methods and strategies of inventory management can be relatively complex and difficult to understand on the part of the staff

5. Excess inventory can create storage issues.

10. CONCLUSION

Inventory management is a process of keeping in-depth track of all your products. With proper inventory management, you will ensure that your company correctly orders, handles, and tracks your entire stock. On the most fundamental level, inventory control is aimed to make sure that your shelves or virtual platforms are adequately stocked and you know your current inventory levels at any moment. To conclude, Inventory Management System is a simple desktop based application basically suitable for small organization. It has every basic items which are used for the small organization. Our team is successful in making the application where we can update, insert and delete the item as per the requirement. This application also provides a simple report on daily basis to know the daily sales and purchase details. This application matches for small organization where there small limited if godwoms. Through it has some limitations, our team strongly believes that the implementation of this system will surely benefit the organization.

11. GitHub Link

https://github.com/IBM-EPBL/IBM-Project-36429-1660295012