INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

Nalaiya Thiran

Professional Readiness for Innovation, Employability & Entrepreneurship

A Project Report

Submitted by

HARIHARAN.D - (311419106010)
AJAY ANAND .P.V - (311419106002)
ARUN KARTHIK.D - (311419106004)
UDAYAKUMAR.A - (311419106032)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

MEENAKSHI COLLEGE OF ENGINEERING,

WEST K.K. NAGAR

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

FACULTY MENTOR
R.ANANDHA PRABA

FACULTY EVALUATOR
D.SATHEESWARI

PROJECT REPORT

Date	19 November 2022
Team ID	PNT2022TMID27727
Project Name	Inventory Management System For Retailers

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. In today's more turbulent environment there is no longer any possibility of manufacturing and marketing acting independently of each other. It is now generally accepted that the need to understand and meet customer requirements is a prerequisite for survival. At the same time, in the search for improved cost competitiveness, manufacturing management has been the subject of massive renaissance. The last decade has seen the rapid introduction of flexible manufacturing systems, of new approaches to inventory based on materials requirement planning (MRP) and just in time (JIT) methods, a sustained emphasis on quality.

1.2 Purpose:

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. In the industries there will be a competitor who will be a low cost producer and will have greater sales volume in that sector. This is partly due to economies of scale, which enable fixed costs to spread over a greater volume but more particularly to the impact of the experience curve.It is possible to identify and predict improvements in the rate of output of workers as they become more skilled in the processes and tasks on which they work. Bruce Henderson extended this concept by demonstrating that all costs, not just production costs, would decline at a given rate as volume increased. This cost decline applies only to value added, i.e. costs other than bought in supplies. Traditionally it has been suggested that the main route to cost reduction was by gaining greater sales volume and there can be no doubt about the close linkage between relative market share and relative costs. However it must also be recognized that logistics management can provide a multitude of ways to increase efficiency and productivity and hence contribute significantly to reduced unit costs.

2. Literature Survey

2.1 EXISTING PROBLEM

Inventory Management System is an integral part of all organizations to manage the information about availability of items in stock and its issues and returns. In this post we will learn how to create a simple online Inventory Management System that allows you to add items, accept requests from employees and Issue items against their requests. The non consumable items can be returned thus updating the stock.

2.2 REFERENCES:

Ahmad, K Mohamed Zabri, S & Mohamed, MIP 2014,

1. The first study analyzed a decision maker having the capability to buy from two different suppliers and using the "periodic-review inventory model" (Fox, Metters & Semple, 2006, p. 389). The first supplier was defined by high variable cost and insignificant fixed cost. The second supplier had low variable cost and high fixed cost. When using different suppliers, there were tradeoffs between variable and fixed cost. the second study, Tracey, Lim and Voderembse (2005) conduct an empirical test to examine the impact of supply-chain management (SCM)

Atkinson, C. (2005, May 9). Today's inventory management. Inventory Management Review. Retrieved March 20, 2007,

2. Generally speaking, retailer selection in a VMI system is a kind of partner selection in a supply chain. The operations research offers a range of methods and techniques for the supplier selection problem. Typical examples of such methods are multi-criteria decision-making approaches and integer programming models.

Multifunctional Barcode Inventory System for Retailing. Are You Ready for It? Ling Shi Cai, Leau Yu Beng, Charlie Albert Lasuin, Tan Soo Fun, Chin Pei Yee

3. The main problem of a sophisticated inventory management usage was due to the inadequacy of qualified personnel as well as the management's attitude. In a later study, Ayad (2008) examined key factors within the control of store managers to optimize the inventory and store.The results found that different stores within the same companies and different departments within the same stores delivered different results, mainly due to human factors, specifically in terms of critical thinking, functional knowledge, and leadership. Strohhecker and Grobler (2013) focused on the physiological traits of inventory managers by investigating the influence of four personal traits.

Role of Emerging Technology for Building Smart Hospital Information System Vaibhav Thakarea, Gauri Khir

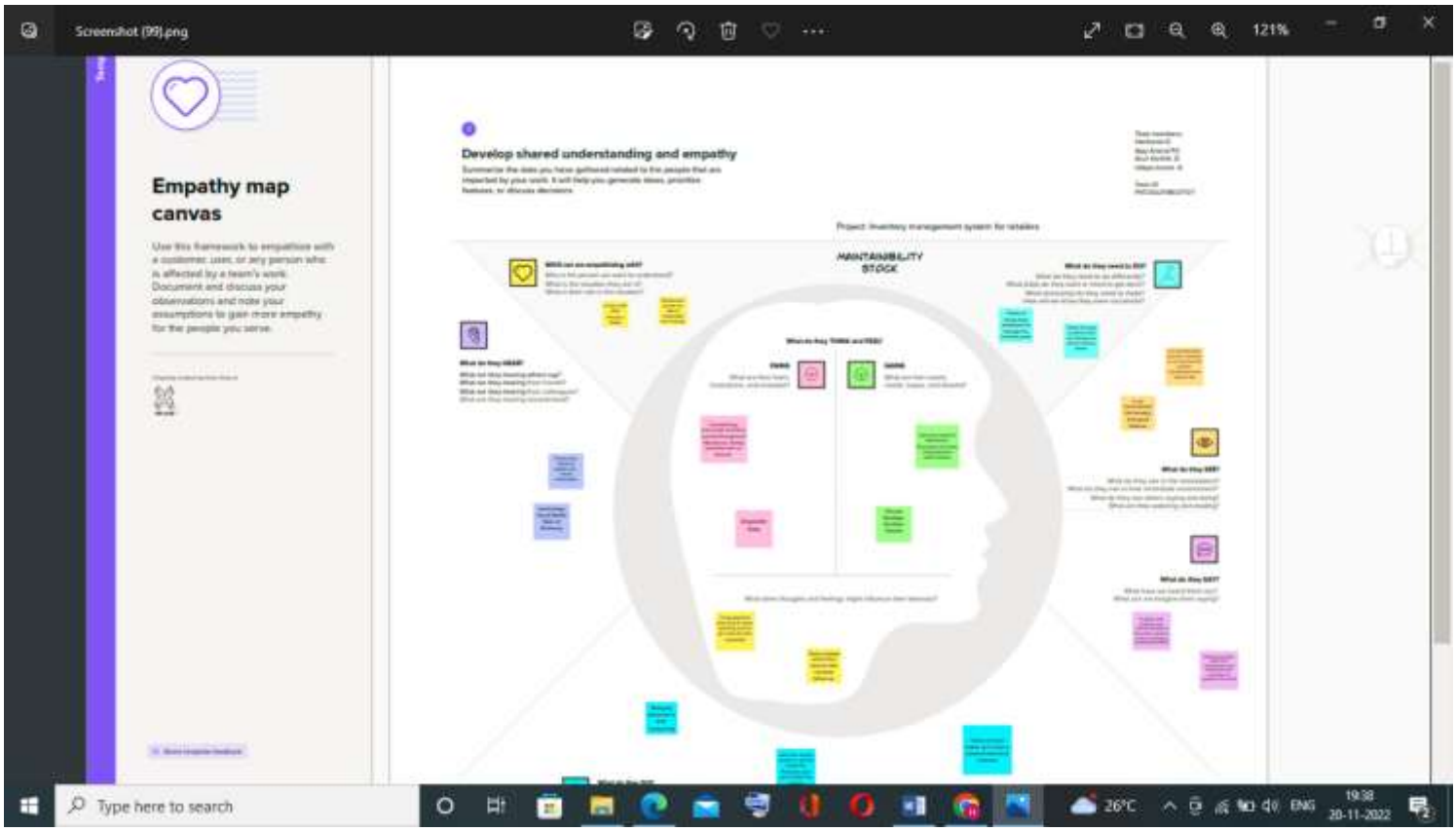
4. The emergence of the internet has been the greatest technological advancement after the industrial age. From the recent studies on internet penetration and usage in India it has been concluded that many Indians are using the internet to pay bills, purchase products online apart from regular surfing,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

2.3 PROBLEM STATEMENT DEFINITION

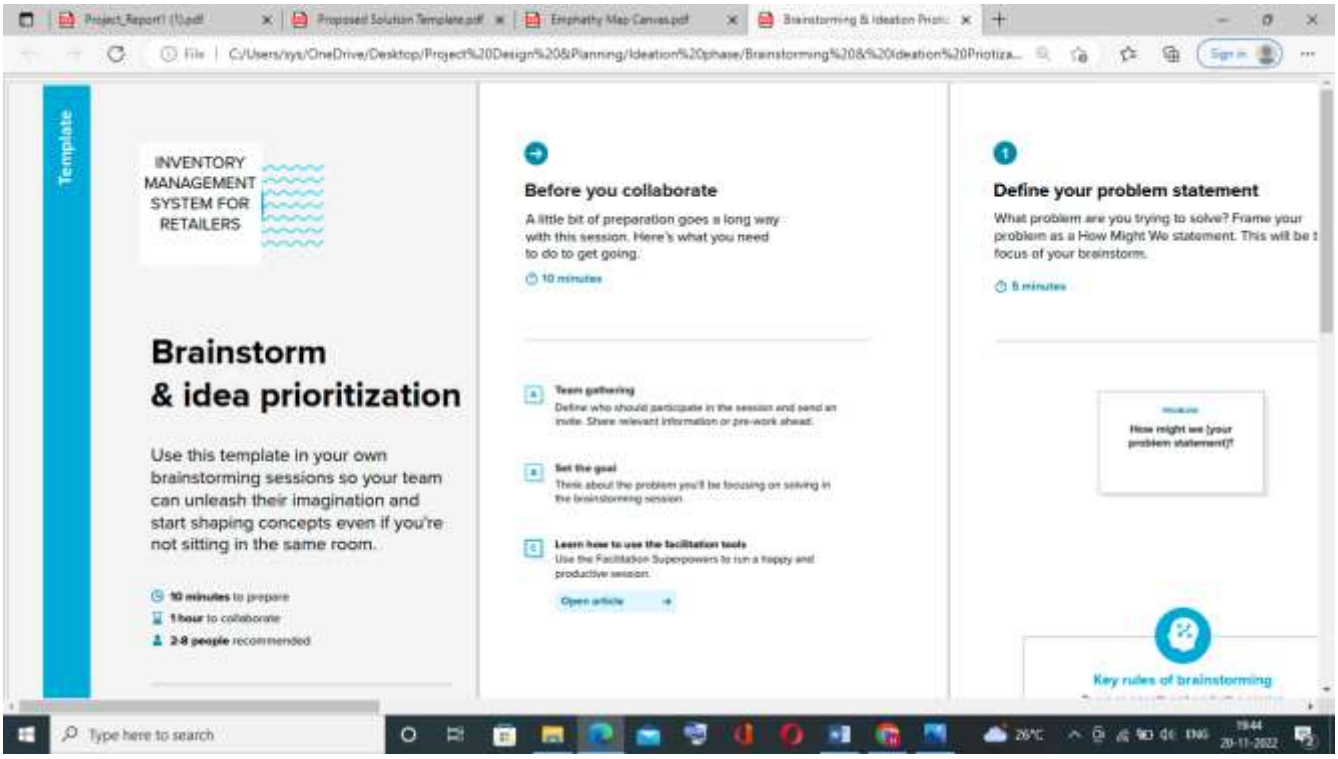
The Problem faced by the retailers is that they do not have any system to record and keep their inventory data. It is difficult for the owner to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized. The inventory process involves multiple intricate aspects that drive accurate product delivery. Even a single error in the process can have expensive and long-term consequences. This will eventually affect the company's growth and reputation. Thus, retail companies need to understand and analyze the risks involved in inventory management. Only then can companies find proactive solutions to the problems.

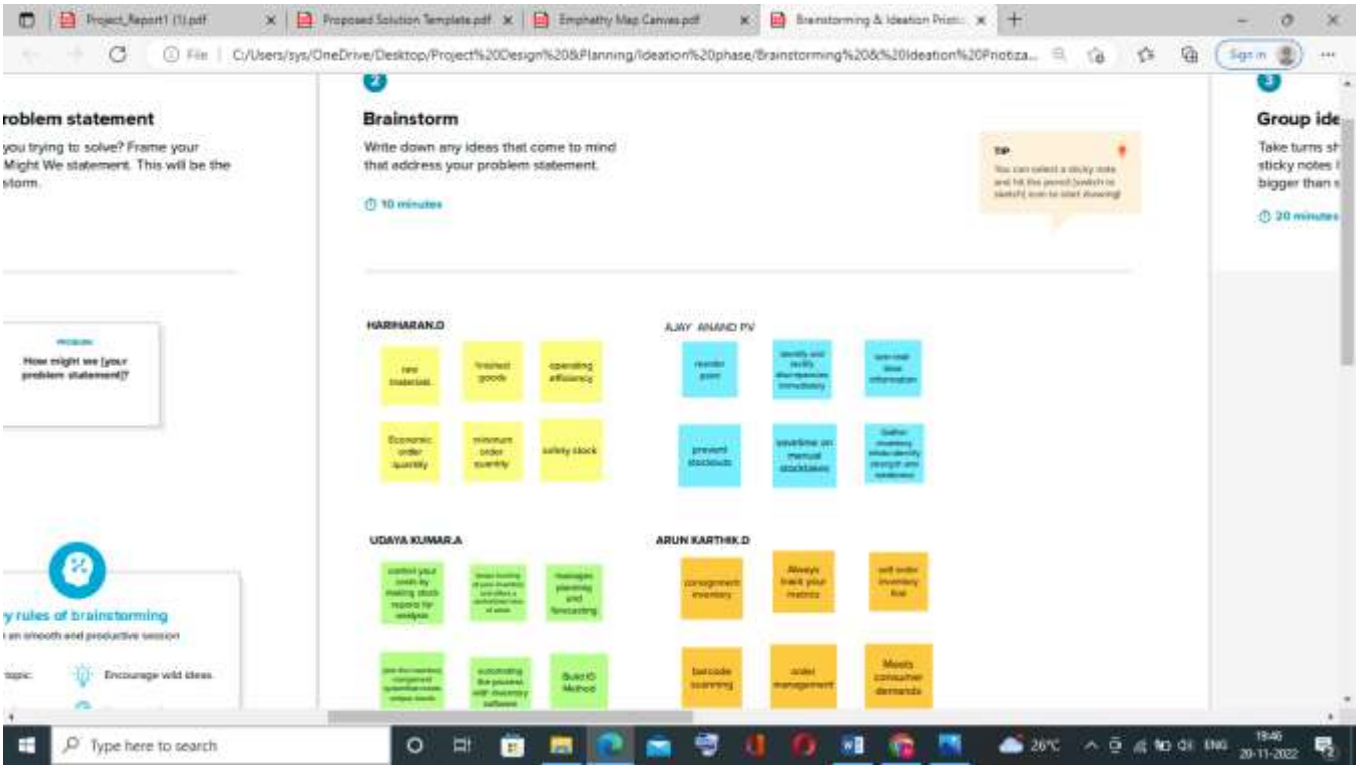
3. IDEATION & PROPOSED SOLUTION

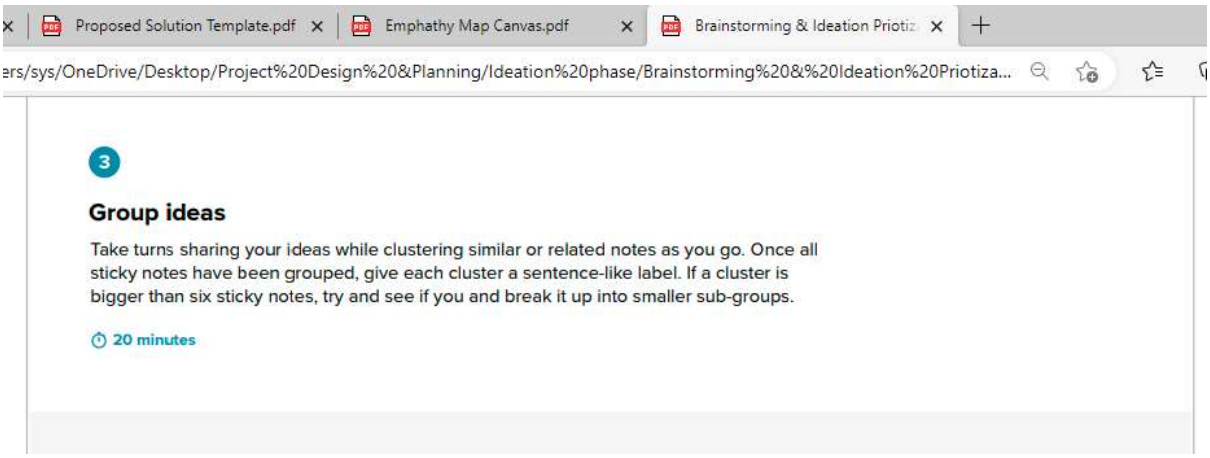
3.1 EMPATHY MAP CANVAS

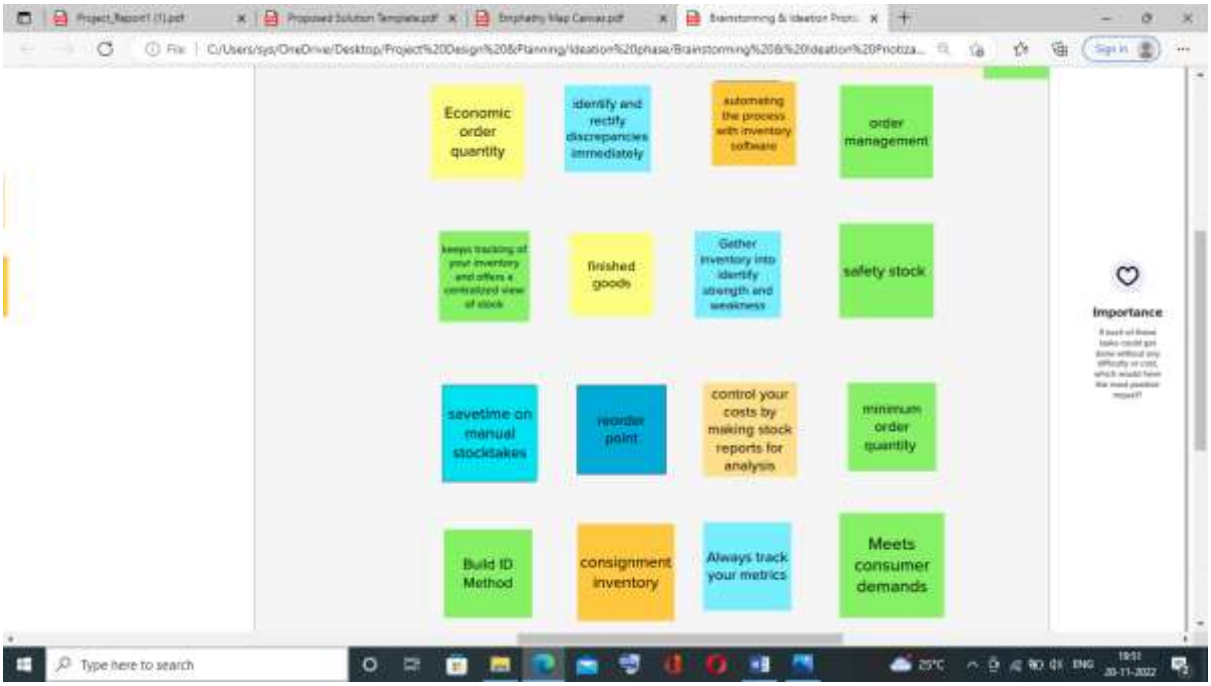


3.2 Ideation & Brainstroming

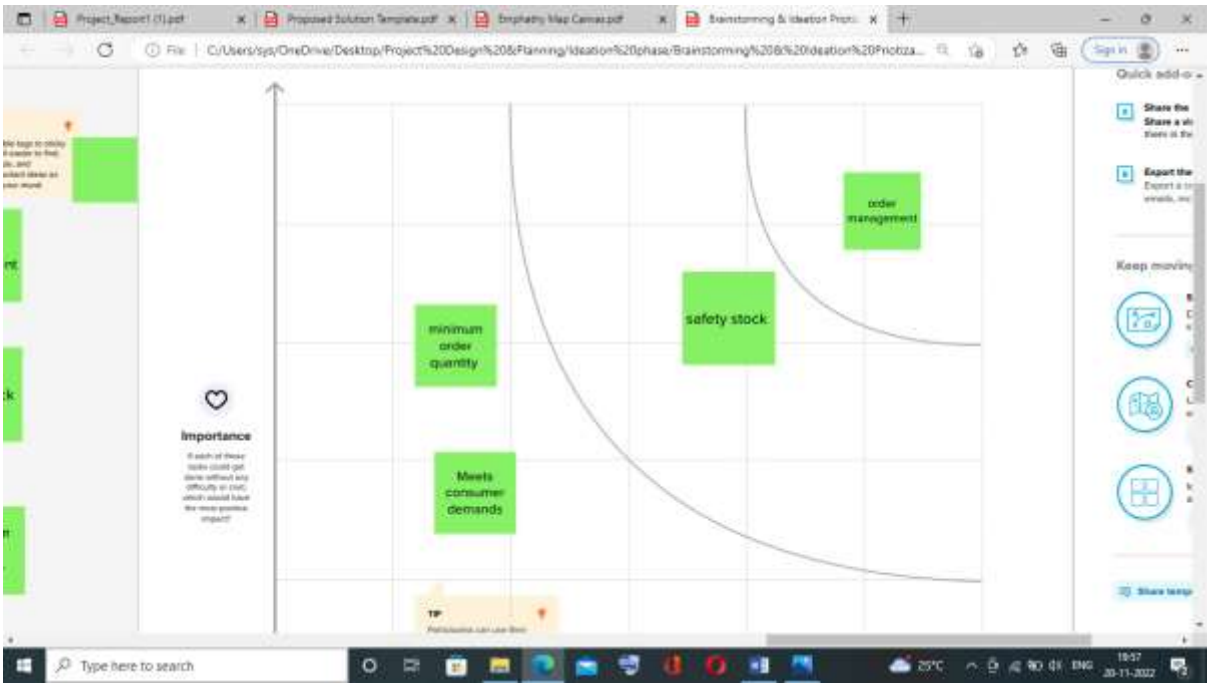




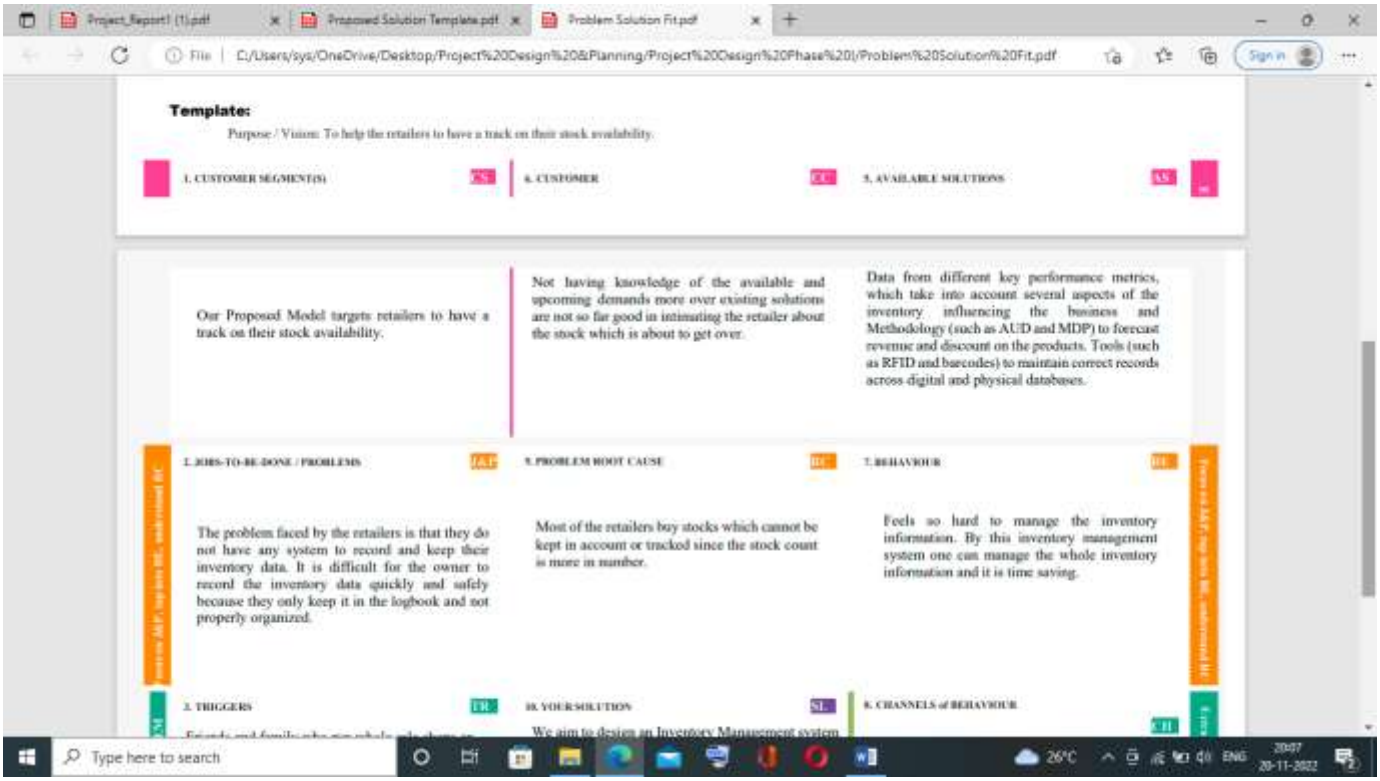

Proposed Solution Template.pdf
Emphathy Map Canvas.pdf
Brainstorming & Ideation Prioti
ers/sys/OneDrive/Desktop/Project%20Design%20&Planning/Ideation%20phase/Brainstorming%20&%20Ideation%20Priotiza...
3
Group ideas
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.
20 minutes


Economic order quantity
identify and rectify discrepancies immediately
automating the process with inventory software
order management
finished goods
Gather inventory into identify strength and weaknees
safety stock
Importance
savetime on manual stocktakes
reorder point
control your costs by making stock reports for analysis
minimum order quantity
Build ID Method
consignment inventory
Always track your metrics
Meets consumer demands
Type here to search


Proposed Solution Template.pdf
Emphathy Map Canvas.pdf
Brainstorming & Ideation Prioti
File | C:/Users/sys/OneDrive/Desktop/Project%20Design%20&Planning/Ideation%20phase/Brainstorming%20&%20Ideation%20Priotiza...
4
Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes



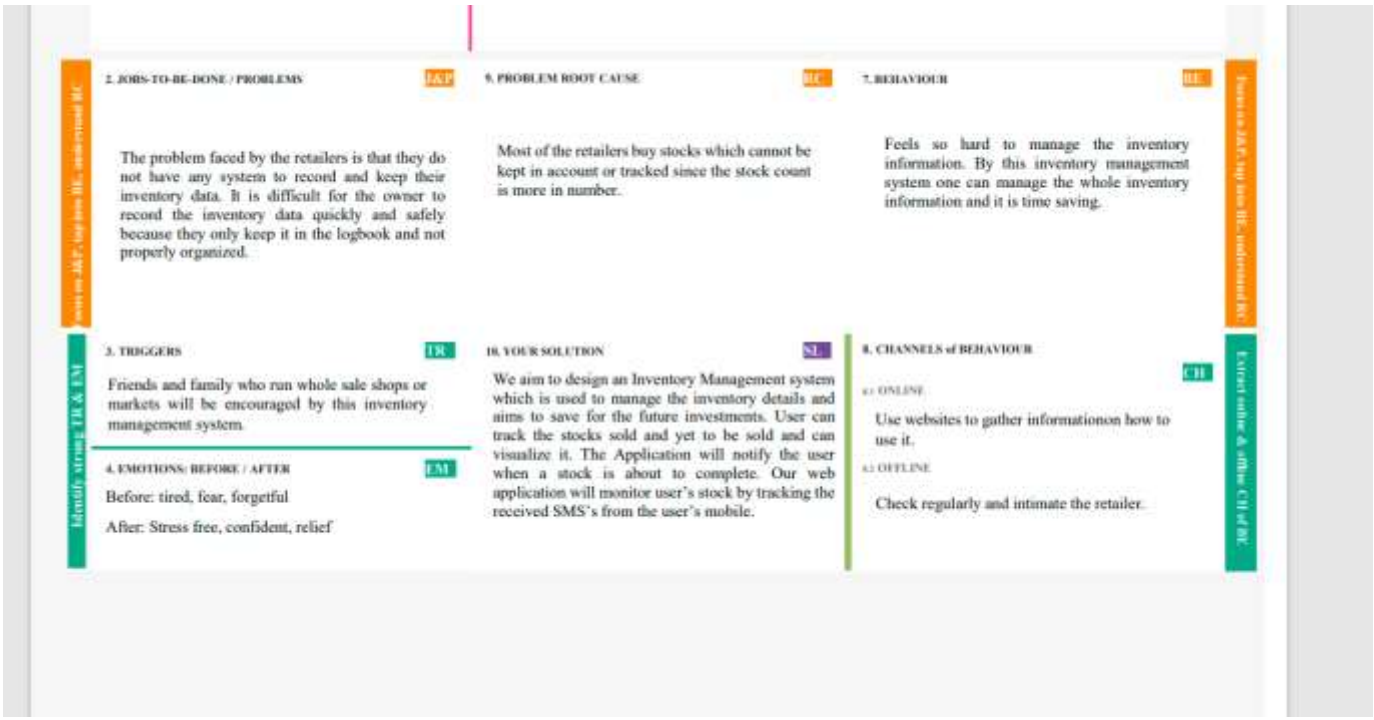
3.3 Proposed Solution



Template:

Purpose / Vision: To help the retailers to have a track on their stock availability.

1. CUSTOMER SEGMENT(S)	6. CUSTOMER	5. AVAILABLE SOLUTIONS
Our Proposed Model targets retailers to have a track on their stock availability.	Not having knowledge of the available and upcoming demands more over existing solutions are not so far good in intimating the retailer about the stock which is about to get over.	Data from different key performance metrics, which take into account several aspects of the inventory influencing the business and Methodology (such as AUD and MDP) to forecast revenue and discount on the products. Tools (such as RFID and barcodes) to maintain correct records across digital and physical databases.
2. JOBS-TO-BE-DONE / PROBLEMS	**9. PROBLEM ROOT CAUSE**	**7. BEHAVIOUR**
The problem faced by the retailers is that they do not have any system to record and keep their inventory data. It is difficult for the owner to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.	Most of the retailers buy stocks which cannot be kept in account or tracked since the stock count is more in number.	Feels so hard to manage the inventory information. By this inventory management system one can manage the whole inventory information and it is time saving.
3. TRIGGERS	**10. YOUR SOLUTION**	**8. CHANNELS of BEHAVIOUR**
	We aim to design an Inventory Management system	



3.4 Problem Solution fit

4.

Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.NO.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The Problem faced by the retailers is that they do not have any system to record and keep their inventory data. It is difficult for the owner to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.
2.	Idea / Solution description	We aim to design an Inventory Management system which is used to manage the inventory details and aims to save for the future stocks sold and can visualize it. The Application will notify the user when a stock is about to complete. Our web application will monitor user's stock by tracking the received SMS's from the user's mobile.
3.	Novelty / Uniqueness	Retailers get notified when the stock is about to get over and intimates the user to buy more stock. Providing Key Performance Indicator for analysing stock. Demand based advanced stock pre-order.
4.	Social Impact / Customer Satisfaction	Encourages user to track stock availability and increase profit. It helps to make a better budget that he will have a financial control.
5.	Business Model (Revenue Model)	The low cost requirement for designing this proposed model makes it more reliable and user friendly.
6.	Scalability of the Solution	With efficient usage of IBM cloud, this proposed model will be able to handle large number of user data. This makes a huge number of users to easily access and efficiently use it.

REQUIREMENT ANALYSIS

4.1 Functional 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 Email Confirmation Confirmation via OTP
FR-3	User Login	Login via google Login with password and Email ID
FR-4	Admin Login	Login via google Login with password and Email ID
FR-5	Query Form	Description of the issues contact immediately
FR-6	E-mail	Login alertness
FR-7	Feedback	Customer Feedback

4.2 Non-functional Requirements

Non-functional Requirements:

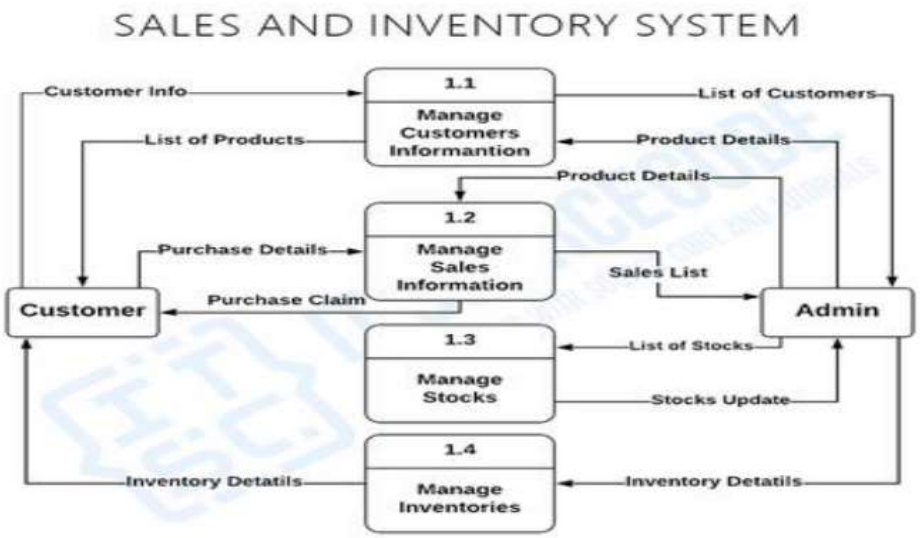
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	To provide solution to the problem
NFR-2	**Security**	Track of login authentication
NFR-3	**Reliability**	Tracking of decade status through mail
NFR-4	**Performance**	Effective development of web application
NFR-5	**Availability**	24/7
NFR-6	**Scalability**	Agents scalability as per number of customers

5. PROJECT DESIGN

5.1 Data Flow Diagrams

Data flow Diagrams:



5.2 Solution & Technical Architecture

Technical Architecture:

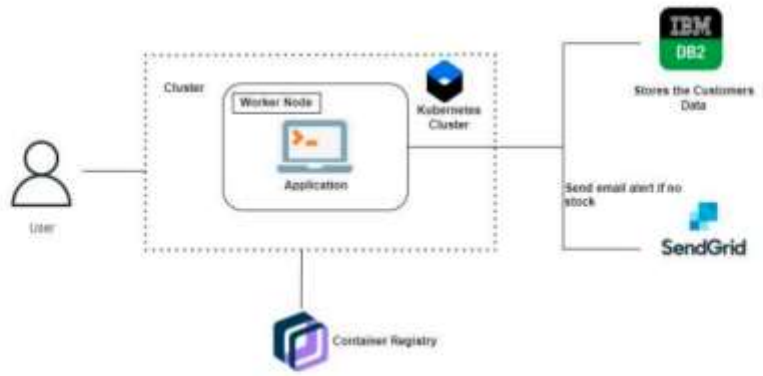


Table-1 : Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Bootstrap etc.
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	SqlAlchemy, Sqlite etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	JOB API, etc.
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-2: **Application Characteristics:**

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Flask, Bootstrap, Kubernetes
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Kubernetes, docker
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	Distributed Servers
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Use of CDN

5.3 User Stories

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a customer, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	login	USN-2	As a customer, I can login to the application by entering correct email and password.	I can access my account/dashboard.	High	Sprint-1
	Dashboard	USN-3	As a customer, I can see all the orders raised by me.	I get all the info needed in my dashboard.	Low	Sprint-2
	Order creation	USN-4	As a customer, I can place my order with the detailed description of my query	I can ask my query	Medium	Sprint-2
	Address Column	USN-5	As a customer, I can have conversations with the assigned agent and get my queries clarified	My queries are clarified.	High	Sprint-3
	Forgot password	USN-6	As a customer, I can reset my password by this option incase I forgot my old password.	I get access to my account again	Medium	Sprint-4
	Order details	USN-7	As a Customer ,I can see the current stats of order.	I get abetter understanding	Medium	Sprint-4
Agent (web user)	Login	USN-1	As an agent I can login to the application by entering Correct email and password.	I can access my account / dashboard.	High	Sprint-3
	Dashboard	USN-2	As an agent, I can see the order details assigned to me by admin.	I can see the tickets to which I could answer.	High	Sprint-3
	Address column	USN-3	As an agent, I get to have conversations with the customer and clear his/er dobuts	I can clarify the issues.	High	Sprint-3
	Forgot password	USN-4	As an agent I can reset my password by this option in case I forgot my old password.	I get access to my account again.	Medium	Sprint-4

DATA FLOW DIAGRAM & USER STORIES

DATA FLOW DIAGRAM & USER STORIES

Admin (Mobile user)	Login	USN-1	As a admin, I can login to the appliaction by entering Correct email and password	I can access my account/dashboard	High	Sprint-1
	Dashboard	USN-2	As an admin I can see all the orders raised in the entire system and lot more	I can assign agents by seeing those order.	High	Sprint-1
	Agent creation	USN-3	As an admin I can create an agent for clarifying the customers queries	I can create agents.	High	Sprint-2
	Assignment agent	USN-4	As an admin I can assign an agent for each order created by the customer.	Enable agent to clarify the queries.	High	Sprint-1
	Forgot password	USN-5	As an admin I can reset my password by this option in case I forgot my old password.	I get access to my account.	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint planning & Estimation

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

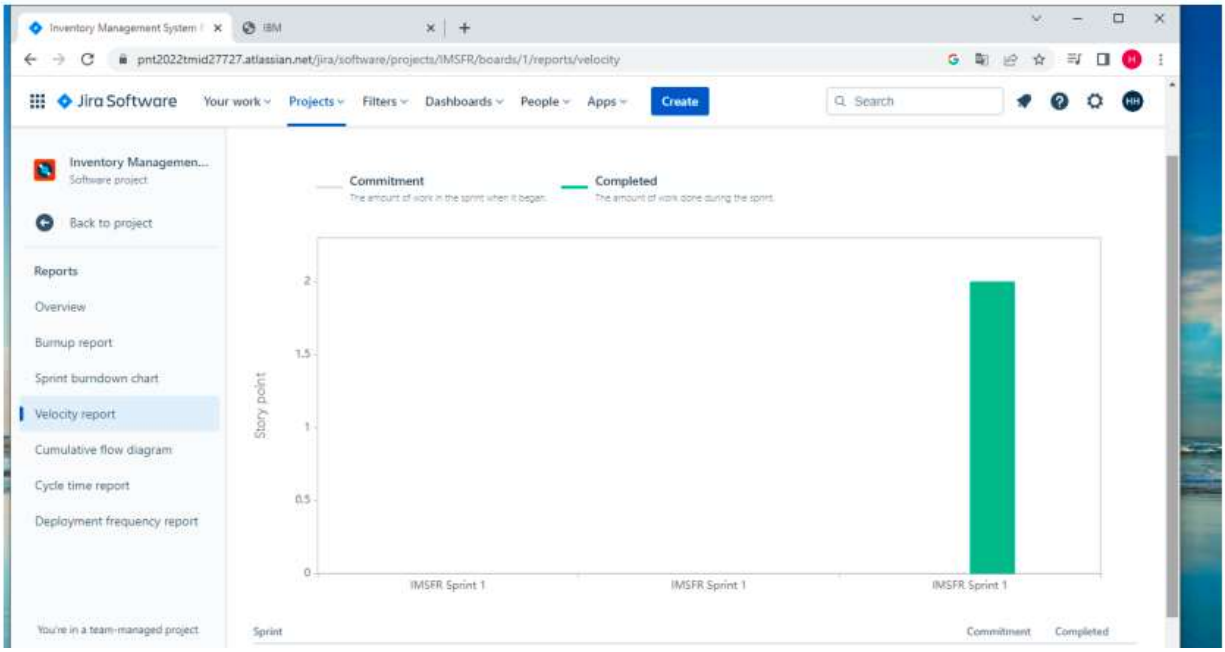
Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Hariharan.D
Sprint-2	Login	USN-2	As an admin,I can see all the orders raised in the entire system lot more	1	High	Arun Karthik.D
Sprint-3	Dashboard	USN-3	As a agent, I can see the order details assigned to me by admin	2	High	Ajay Anand.P.V.
Sprint-4	Assignment agent	USN-4	As an admin I can assign an agent for each order created by the customer	2	High	Udayakumar.A

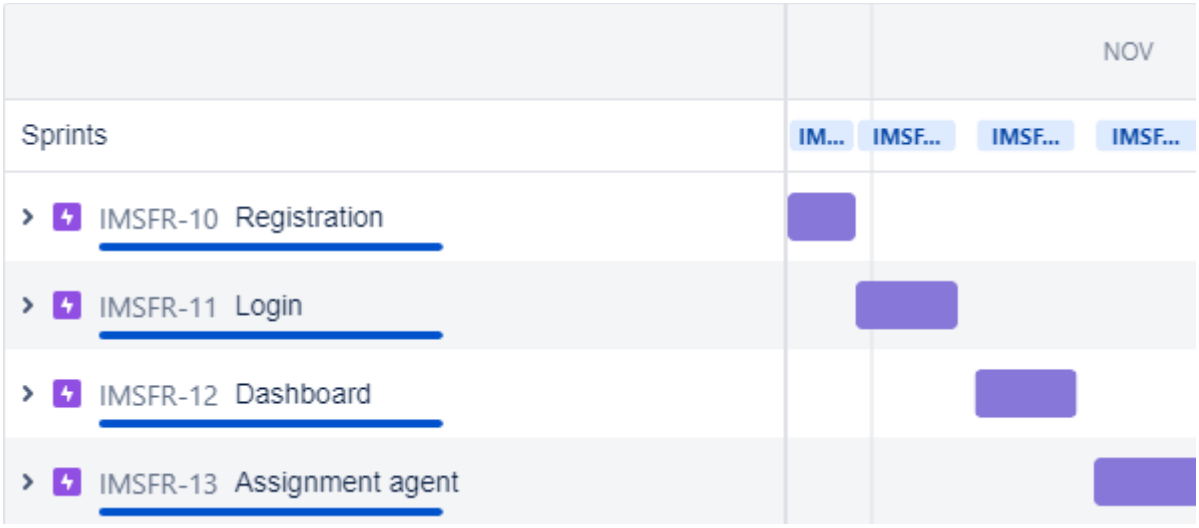
6.2 Sprint Delivery estimation

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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



6.3 Reports from JIRA



7.CODING & SOLUTIONING

7.1 Feature 1

- track raw material and finished goods for manufacturers
- track lot numbers, FDA, and recall
- support for kitting and costing of kits from components and labor

7.2 FEATURE 2

- Business owners manage the inventory well with the help of inventory software. Managers balance the demand and supply of the company products efficiently.
- This is why businesses are capable of generating a huge amount of revenue on an annual basis.

7.3 DATABASE SCHEMA (IF APPLICABLE)

- The shop has an inventory of products. Each product has a price, but this price should vary depending on sales.
- Customers can make orders for multiple products at a time, and should be able to see their order history.
- When the order has been completed, there should be a track and trace number.

```
import os
import numpy as np
from flask import Flask, render_template, request, send_from_directory, url_for
from gevent.pywsgi import WSGIServer
 from keras.models import load_model
 from keras.preprocessing import image
 from PIL import Image
from werkzeug.utils import redirect, secure_filename

UPLOAD_FOLDER = 'D:/NalaiyaThiran/projFiles/data
app = Flask( name )
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
model = load_model("./model/mnist_digit_recog_cnn.h5")
@app.route('/')
def index():
return render_template('index.html')
```

```
@app.route('/web', methods=['GET', 'POST'])
def web():
 if request.method == "POST":
f = request.files["image"]
basepath = os.path.dirname( file )
filepath = os.path.join(basepath, 'data', f.filename)
f.save(filepath)
 # img = image.load_img(filepath, target_size=(64, 64))
# x = image.img_to_array(img)
 # x = np.expand_dims(x, axis=0)

 # filepath = secure_filename(f.filename)
# f.save(os.path.join(app.config['UPLOAD_FOLDER'], filepath))

 # upload_img = os.path.join(UPLOAD_FOLDER, filepath)
img = Image.open(filepath).convert("L") # convert image to monochrome
 img = img.resize((28, 28)) # resizing of input image
im2arr = np.array(img) # converting to image
im2arr = im2arr.reshape(1, 28, 28, 1)# reshaping according to our
requirement
 pred = model.predict(im2arr)
 num = np.argmax(pred, axis=1)# printing our Labels
return render_template('web.html', num=str(num[0]))
return render_template('web.html')
if name == ' main ':
app.run(debug=True, threaded=False)
```

8. TESTING

8.1 Test cases

Activity	Activity Name	Detailed activity Description	Assigned to	Duration	Status
7.	Create IBM DB2 and connect with python	Create a IBM db2 service in ibm cloud and connect with python code using DB	Hariharan.D	09th to 15th November	completed
Integrating SendGrid services					
8.	Sendgrid Integration with Python code	To send emails from the applications we need to integrate the SendGrid service	Arun karthik.D	14 to 19th November	completed
Deployment of App in IBM cloud					
9.	Containerize the App	Need to create a docker image on the application and push into the IBM container registry	Arun karthik.D Udayakumar.A	08th November to 14th November	completed
10.	Upload image to IBM container registry	Upload the image to IBM container registry	Ajay Anand P.V	08th November to 14th November	completed
11.	Deploy in Kubernetes cluster	Once the image is uploaded in IBM Container registry deploy the image to IBM Kubernetes cluster	Hariharan.D	08th November to 14th November	completed

8.2UserUserAcceptance Testing

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

Purpose of UAT

The main Purpose of UAT is to validate end to end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is kind of black box testing where two or more end-users will be involved.UAT

is performed by –

• Client

• End users

Need of User Acceptance Testing arises once software has undergone Unit, Integration and System testing because developers might have built software based on requirements document by their own understanding and further required changes during development may not be effectively communicated to them, so for testing whether the final product is accepted by client/end-user, user acceptance testing is needed.

Developers code software based on requirements document which is their “own” understanding of the requirements and may not actually be what the client needs from the software.

Requirements changes during the course of the project may not be communicated effectively to the developer.

8.2 User Acceptance Testing

Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	3	2	2	1	8
Duplicate	0	1	2	1	4
External	1	3	2	1	7
Fixed	4	2	3	15	24
Not Reproduced	0	0	1	1	2
Skipped	1	0	1	1	3
Won't Fix	2	3	2	1	8
Totals	11	11	13	21	56

Test Case Analysis

This reportshows the numberof test cases that have passed, failed,and untested.

Section	Total Cases	Not Tested	Fail	Pass
Login	8	0	0	8
Dashboard	19	0	0	19
Db2 Database	9	0	0	9
Flask Application	4	0	0	4

9. RESULTS

9.1 Performance Metrices

Performance metrics are figures and facts that quantify an organization's capabilities, actions, and general level of excellence. Profit, revenue, customer satisfaction, ROI, customer reviews, overall quality, individual reviews, and reputation in marketplaces are only a few examples of different performance measures. Be aware that performance measurements can vary depending on which industry they are seen through. Performance metrics are crucial to the success of any firm. Since performance metrics support and ensure an organization's success, it is imperative for any business to select its performance metrics and then pay attention to those areas. Some crucial success factors are beneficial if they are monitored and acknowledged. Business measurements must be constantly monitored to ensure they are producing the important answers and that the appropriate questions are being posed.

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

1. It helps to maintain the right amount of stocks: contrary to the belief that is held by some people, inventory management does not seek to reduce the amount of inventory that you have in stock, however, it seeks to maintain an equilibrium point where your inventory is working at a maximum efficiency and you do not have to have many stocks or too few stocks at hand at any particular point in time. The goal is to find that zone where you are never losing money in your inventory in either direction. With the aid of an efficient inventory management strategy, it is easy to improve the accuracy of inventory order.

2. It leads to a more organized warehouse: with the aid of a good inventory management system, you can easily organize your warehouse. If your warehouse is not organized, you will find it very difficult to manage your inventory. A lot of businesses choose to optimize their warehouse by putting the items that have the highest sales together in a place that is easy to access in the warehouse. This ultimately helps to speed up order fulfilment and keeps clients happy.

3. It saves time and money: an effective inventory management system can translate to time and money saved on the part of the business. By keeping track of the product that you already have at hand, you can save yourself the hassles of having to do an inventory recount in order to ensure your records are accurate. It also allows you to save cash that would have otherwise been spent on slow moving products.

4. Improves efficiency and productivity: inventory management devices like bar code scanners and inventory management software can help to greatly increase the efficiency and productivity of a business. They do this by eliminating the manual way of doing things thus allowing employees to do other more important things for the business.

DISADVANTAGES:

1. Bureaucracy: even though inventory management allows employees at every level of the company to read and manipulate company stock and product inventory, the infrastructure required to build such a system adds a layer of bureaucracy to the whole process and the business in general. In instances where inventory control is in-house, this includes the number of new hires that are not present to regulate the warehouse and facilitate transactions.

2. Impersonal touch: another disadvantage of inventory management is a lack of personal touch. Large supply chain management systems make products more accessible across the globe and most provide customer service support in case of difficulty, but the increase in infrastructure can often mean a decrease in the personal touch that helps a company to stand out above the rest.

3. Production problem: even though inventory management can reveal to you the amount of stock you have at hand and the amount that you have sold off, it can also hide production problems that could lead to customer service disasters. Since the management places almost all of its focus on inventory management to the detriment of quality control, broken or incorrect items that would normally be discarded are shipped along with wholesome items.

4. Increased space is need to hold the inventory: in order to hold inventory, you will need to have space so unless the goods you deal in are really small in size, then you will need a warehouse to store it. In addition, you will also need to buy shelves and racks to store your goods, forklifts to move around the stock and of course staff. The optimum level of inventory for a business could still be a lot of goods and they will need space to be stored in and in some cases additional operational costs to manage the inventory. This will in turn increase cost and impact negatively on the amount of profit the business makes.

11. CONCLUSION:

To conclude, Inventory Management System is a simple desktop based application basically suitable for small organization. It has every basic items which are used forthe small organization. Our team is successful in making the application where 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.

12. FUTURE SCOPE:

The future of inventory is going to be flooded with updates in technology. From virtual reality, to artificial intelligence, to digital signage, and even inventory-less stores, there are constant iterations made in this industry to accelerate business and attract customers. And it's only growing from here.

Inventory management systems have become more real-time, giving retailers more data about demographics, spending habits, shopping preferences, and more. With this constant increase in inventory visibility, retailers aim to better their accuracy with their inventory, and continue to appeal to their consumers.

Take Nordstrom, for example. Nordstrom has recently introduced a new inventory strategy – only with no inventory. They've introduced a store concept called Nordstrom Local, a tiny space where instead of focusing on the inventory, they focus on the customers. Customers crave fast and convenient methods of shopping, and Nordstrom took that and ran with it. The store has alteration services, personal styling, nail salons, beverages, online order pick ups, just no inventory. By introducing this store concept, they are proving their loyalty to their customers and giving them what they really want.

13. APPENDIX

SOURCE CODE

Register.html

```
<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<style>
body {
  font-family: Arial, Helvetica, sans-serif;
  background-color: black;
}

* {
  box-sizing: border-box;
}

/* Add padding to containers */
.container {
  padding: 16px;
  background-color: white;
}

/* Full-width input fields */
input[type=text], input[type=password] {
  width: 100%;

  padding: 15px;
  margin: 5px 0 22px 0;
  display: inline-block;
  border: none;
  background: #f1f1f1;
}

input[type=text]:focus, input[type=password]:focus {
  background-color: #ddd;
  outline: none;
}

/* Overwrite default styles of hr */
```

```
hr {
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

/* Set a style for the submit button */
.registerbtn {
  background-color: #04AA6D;
  color: white;
  padding: 16px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
  opacity: 0.9;
}

.registerbtn:hover {
  opacity: 1;
}

/* Add a blue text color to links */

{
  color: dodgerblue;
}

/* Set a grey background color and center the text of the "sign in" section */
.signin {
  background-color: #f1f1f1;
  text-align: center;
}
```

```
</style>
</head>
<body>

<form action=" " method="post">
 <div class="container">
   <h1>Register</h1>
   <p>Please fill in this form to create an account.</p>

   <hr>

   <label for="email"><b>Email</b></label>
   <input type="text" placeholder="Enter Email" name="email" id="email" required>

   <label for="psw"><b>Password</b></label>
   <input type="password" placeholder="Enter Password" name="psw" id="psw" required>

   <label for="psw-repeat"><b>Repeat Password</b></label>
   <input type="password" placeholder="Repeat Password" name="psw-repeat" id="psw-repeat" required>
   <hr>
   <p>By creating an account you agree to our <a href="#">Terms & Privacy</a>.</p>

   <input type="submit" name="btn" value="submit">
 </div>


 <div class="container signin">
   <p>Already have an account? <a href="#">Sign in</a>.</p>
 </div>
</form>
<?php
include "config.php";
if(isset($_POST["btn"])) {
```

```
$email = $_POST['email'];
$password = $_POST['password'];
$confirm_password = $_POST['confirm_password'];

if($password == $confirm_password){

//INSERT INTO `Registration` (`Email`, `password`, `Confirm password`) VALUES ('', '', '')



} else {

//display error message

}

?>
</body>
</html>
```

App.py

```
from flask import Flask, render_template, request, redirect, session
import sqlite3 as sql

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


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


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

```
def about():
  return render_template('about.html')

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

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

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



@app.route('/data',methods = ['POST', 'GET'])
def data():
  if request.method == 'POST':
    try:
      username = request.form['username']
      email = request.form['email']
      password = request.form['password']



      with sql.connect("student_database.db") as con:
        cur = con.cursor()
        cur.execute("INSERT INTO students (username,email,password) VALUES
(?,?,?)",(username,email,password) )
        con.commit()
        msg = "Record successfully added!"
    except:
      con.rollback()
```

```
        msg = "error in insert operation"

      finally:
        return render_template("list.html",msg = msg)
        con.close()

@app.route('/list')
def list():
   con = sql.connect("student_database.db")
   con.row_factory = sql.Row

   cur = con.cursor()
   cur.execute("select * from students")

   students = cur.fetchall()
   return render_template("list.html", students = students)

if __name__ == '__main__':
   app.run(debug = True)



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

```
body {
    text-align:center;
    background-color: rgb(161, 187, 213);
}
```

```
 button{float:left ;


}
p {
    color: white;
    font-family: Shanti;
    font-size: 1.2em;
    display: inline-block;
    margin:  20px;
}
img {
    margin: 60px 0 30px 0;
    width: 250px;
}
input {
  width: 300px;
  margin: 20px 20px;
  height: 50px;
  border: none;
  border-radius: 10px;
  font-family: Shanti;
  font-size: 1.3em;
  text-align: center;
}

input:focus {
    outline: none;
    border: solid 5px #00FFCE;
}

#greet {
    background-color: rgb(27, 193, 132);
    border:  none;
```

```
   width:  200px;
   color: white;
}

#greet:hover {
   background-color: rgb(21, 219, 91);
}
```

Product.html

```
<!DOCTYPE html>
<html>
<head>
{% if title%}
            <title>Inventory {{title}}</title>
        {% else %}
                <title>Inventory</title>
        {% endif %}
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-light">
                <a class="navbar-brand" href="#"><b>Inventory Management System</b></a>
                        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle
navigation">
                          <span class="navbar-toggler-icon"></span>
                        </button>
                        <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
                          <div class="navbar-nav">
                            <a class="nav-item nav-link" href="/home">Home</a>
                            <a class="nav-item nav-link active" href="/Product">Product <span class="sr-
only">(current)</span></a>
```

```
        <a class="nav-item nav-link" href="/stock">Stock</a>
                                    <!-- <a class="nav-item nav-link" href="/Location">Location</a> -->
                                    <!-- <a class="nav-item nav-link" href="/ProductMovement">Product Movement</a>
-->
                                  </div>
                                 </div>
                                </nav>
 </br>
<div class="container">
 <h2>Product Information</h2>

 <div class="float-md-right">
 <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">ADD
PRODUCT</button></div>
 <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel"
aria-hidden="true">
 <div class="modal-dialog" role="document">
 <div class="modal-content">
    <div class="modal-header">
     <h5 class="modal-title" id="exampleModalLabel">ADD PRODUCT</h5>
     <button type="button" class="close" data-dismiss="modal" aria-label="Close">
       <span aria-hidden="true">&times;</span>
     </button>
    </div>
    <div class="modal-body">
        <div class="col-md-4">
     <form class="form-group" action = "{{ url_for('addProduct') }}" method = "POST">
        <div class="form-group">
          Name
          <input class="form-group" type="text" name="pn" placeholder="Product Name" id="p_name" required>
        </div>
         <div class="form-group">
          Description
          <input class="form-group" type="text" name="pd" placeholder="Product Description"
id="P_description" required>
```

```
          </div>
           <div class="form-group">
            QTY
             <input class="form-group" type="text" name="pq" placeholder="Product QTY" id="P_QTY" required>
          </div>
    </div>
    <div class="modal-footer">
      <input class="btn btn-success" class="form-control"  type="Submit" value="Submit" />
      <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
    </div>
   </div>
 </form></div>
 </form>
 </div>
</div>

 </br>
</br>

 <table class="table table-striped">
   <thead>
     <tr>
       <th class="text-info">Product id</th>
       <th class="text-info">Product name</th>
       <th class="text-info">Product Description</th>
       <th class="text-info">Qty</th>
       <th class="text-info">Validity</th>
       <th class="text-success">Edit</th>
       <th class="text-danger">Delete</th>
     </tr>
   </thead>
   <tbody>
     {% for row in rows %}
```

```
        <tr>
          <td>{{row["productID"]}}</td>
          <td>{{row["productName"]}}</td>
          <td> {{ row["productDescription"]}}</td>
          <td>{{row['QTY']}}</td>
          <td>{{row['Validity']}}</td>
          <td>
              <a><button class="btn btn-primary"
OnClick='showModal({{row["productID"]}},"{{row["productName"]}}","{{row["productDescription"]}}","{{row["Q
TY"]}}","{{row["Validity"]}}");'>Edit</button></a></td>
                <td> <a href='deleteProduct/{{row["productID"]}}'><button class="btn btn-
danger">Delete</button></a>
              </td>

        </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
 <div class="modal fade" id="myModal">
    <div class="modal-dialog">
    <div class="modal-content">

      <!-- Modal Header -->
      <div class="modal-header">
        <h4 class="modal-title">Edit Product</h4>
        <button type="button" class="close" data-dismiss="modal">&times;</button>
      </div>

      <!-- Modal body -->
      <div class="modal-body">
        <form class="form-group"  action = "{{ url_for('editProduct') }}" method = "POST">
          <div class="form-group">
            <input type="text" readonly class="form-control-plaintext  form-control-lg" name="ProductID"
id="ProductID" style="display: none;" value="00">
```

```
            </div>
            <div class="form-group">
              <label for="NEWProductName" class="sr-only"></label>
              <input class="form-control form-control-lg" type="text" name="NEWProductName" placeholder="New Product Name" id="NEWProductName" required>
            </div>
             <div class="form-group">
              <label for="NEWProductDescription" class="sr-only"></label>
              <input class="form-control form-control-lg" type="text" name="NEWProductDescription" placeholder="NEW Product Description" id="NEWProductDescription" required>
            </div>
            <div class="form-group">
              <label for="NEWProductQty" class="sr-only"></label>
              <input class="form-control form-control-lg" type="text" name="NEWProductQty" placeholder="NEW Product QTY" id="NEWProductQty" required>
            </div>
            <button type="submit" class="btn btn-success mb-2 font-weight-bold">EDIT PRODUCT</button>
          </form>
        </div>
      </div>
    </div>
  </div>
 </div>
</body>
</html>
<script type="text/javascript">

   function showModal(id,oldname,oldDescription,oldqty){
          $('#ProductID').val(id);
          $('#NEWProductName').val(oldname);
          $('#NEWProductDescription').val(oldDescription);
          $('#NEWProductQty').val(oldqty);
          $('#myModal').modal('toggle');
       }
```

```
  </script>
</script>
```

Stock.html

```
<!DOCTYPE html>
<html>
<head>
 <title>Stock balance</title>
 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css">
 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>
 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>
</head>
<body>
          <table class="table table-striped">
   <thead>
    <tr>
     <!-- <th class="text-info">Location Name</th> -->
     <th class="text-info">Product Name</th>
       <th class="text-info">Product QTY</th>
       <th class="text-info">Product Validity</th>
       <th class="btn btn-danger"><a href="/home"><button>Home Page</button></a></th>

    </tr>
   </thead>
   <tbody>
    {% for row in rows %}
        <tr>
          <td>{{row["productName"]}}</td>
          <td>{{row["QTY"]}}</td>
          <td>{{row["Validity"]}}</td>
```

```
          </tr>
        {% endfor %}
    </tbody>
  </table>
</div>

          </form>
        </div>
      </div>
    </div>
  </div>
 </div>
</body>
</html>
```

App.py

```
from flask import Flask,render_template,redirect, url_for
from flask import request
import sqlite3 as sql
# import os
# PEOPLE_FOLDER = os.path.join('static', 'people_photo')




from cloudant.client import Cloudant

client=Cloudant.iam('71ba514f-7497-4541-b670-2cca557572d7-
bluemix','jbwiZagjHPeEaY_Ff_WeZowtqr_cjHjP5fUsGtkaqfDH',connect=True)
my_database=client.create_database('my_database')



con = sql.connect("database.db",check_same_thread=False)
con.row_factory = sql.Row
```

```
cur = con.cursor()


app = Flask(__name__)
# PEOPLE_FOLDER = os.path.join('static', 'people_photo')
# app.config['UPLOAD_FOLDER'] = PEOPLE_FOLDER


@app.route('/')
def login():
  return render_template('login.html')
# def show_login():
  # full_filename = os.path.join(app.config['UPLOAD_FOLDER'], 'logo.png')
  # return render_template("login.html",user_image = full_filename)


# @app.route('/home')
# def home():
#    return redirect(url_for('home'))


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



@app.route("/home")
def home():
   return render_template('home.html')
   #stock balance
@app.route("/stock")
def stock():
  con = sql.connect("database.db")
  con.row_factory = sql.Row


  cur = con.cursor()
  cur.execute("select * from Product")
```

```
  rows = cur.fetchall();
  return  render_template('stock.html',rows = rows)



#register
@app.route('/afterreg',methods=['POST'])
def afterreg():
   x=[x for x in request.form.values()]
   print(x)
   data={
      '_id':x[1],
      'name':x[0],
      'psw':x[2]
   }
   print(data)
   query={'_id':{'$eq':data['_id']}}
   docs=my_database.get_query_result(query)
   print(docs)


   print(len(docs.all()))


   if(len(docs.all())==0):
      url=my_database.create_document(data)
      return render_template('login.html',pred="Registeration successfull, Please login your details")
   else:
      return render_template('register.html',pred="You re already member, Please login using r details")


#login
@app.route('/afterlogin',methods=['POST'])
def afterlogin():
            user = request.form['_id']
            passw = request.form['psw']
```

```
        print(user,passw)
        query = {'_id': {'$eq': user}}
        docs = my_database.get_query_result(query)
        print(docs)
        print(len(docs.all()))
        # if(len(docs,all())==0):
        #         return render_template('login.html',pred="The username is not found.")
        # else:
        if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):
                return render_template('home.html')
        else:
                print('Invalid User')



                    #Product Page
@app.route("/Product")
def Product():
  con = sql.connect("database.db")
  con.row_factory = sql.Row

  cur = con.cursor()
  cur.execute("select * from Product")

  rows = cur.fetchall();
  return  render_template('Product.html',rows = rows)

                    #ADD Product
@app.route('/addProduct',methods = ['POST'])
def addProduct():
  if request.method == 'POST':
    try:
      pn = request.form['pn']
      pd = request.form['pd']
```

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



        with sql.connect("database.db") as con:
          cur = con.cursor()
          cur.execute("INSERT INTO Product (productName,productDescription,QTY) VALUES (?,?,?)",(pn,pd,pq) )


          con.commit()
          msg = "Record added"
      except:
        con.rollback()
        msg = "error in  operation"


      finally:


        return redirect(url_for('Product')+"?msg="+msg)
        con.close()
                            #Edit Product
@app.route('/editProduct',methods = ['POST'])
def editProduct():
  if request.method == 'POST':
      try:
        productID = request.form['ProductID']
        productName = request.form['NEWProductName']
        productDescription=request.form['NEWProductDescription']
        ProductQty=request.form['NEWProductQty']
        cur.execute("UPDATE Product SET productName = ?,productDescription = ?, QTY = ? WHERE productID =
?",(productName,productDescription,ProductQty,productID) )


        con.commit()
        msg = "Product Edited "
      except:
        con.rollback()
        msg = "error in operation"
```

```
    finally:
      return redirect(url_for('Product')+"?msg="+msg)
      con.close()


                    #Delete Product
@app.route('/deleteProduct/<productID>')
def deleteProduct(productID):
    try:
        cur.execute("DELETE FROM Product WHERE productID = ?",(productID,))


        con.commit()
        msg = "Product Deleted"
    except:
        con.rollback()
        msg = "error in operation"


    finally:
        return redirect(url_for('Product')+"?msg="+msg)
        con.close()
                    #Location Page
@app.route("/Location")
def Location():
  con = sql.connect("database.db")
  con.row_factory = sql.Row


  cur = con.cursor()
  cur.execute("select * from locations")


  rows = cur.fetchall();
  return  render_template('Location.html',rows = rows)
                     #ADD Locations
@app.route('/addlocation',methods = ['POST'])
```

```
def addlocation():
  if request.method == 'POST':
    try:
      ln = request.form['ln']

      with sql.connect("database.db") as con:
        cur = con.cursor()
        cur.execute("INSERT INTO locations (locationName) VALUES (?)",(ln,))

        con.commit()
        msg = "successfully added"
    except:
        con.rollback()
        msg = "error in operation"

    finally:
        return redirect(url_for('Location')+"?msg="+msg)
        con.close()
                    # Edit Location
@app.route('/editlocation',methods = ['POST'])
def editlocation():
  if request.method == 'POST':
    try:
      locationID = request.form['locationID']
      locationName = request.form['NEWLocationName']
      cur.execute("UPDATE locations SET locationName = ? WHERE locationID = ?",(locationName,locationID) )

      con.commit()
      msg = "location Edit Successfully"
    except:
      con.rollback()
      msg = "error operation"
```

```
    finally:
      return redirect(url_for('Location')+"?msg="+msg)
      con.close()
                # Delete Location
@app.route('/deletelocation/<locationID>')
def deletelocation(locationID):
    try:
        cur.execute("DELETE FROM locations WHERE locationID = ? ",(locationID))

        con.commit()
        msg = "location Delete Successfully"
    except:
        con.rollback()
        msg = "error operation"

    finally:
        return redirect(url_for('Location')+"?msg="+msg)
        con.close()

@app.route('/ProductMovement')
def ProductMovement():
  cur.execute("select * from product_movement")

  rows = cur.fetchall()

  cur.execute("select * from products")
  productRows = cur.fetchall()

  cur.execute("select * from locations")
  locationRows = cur.fetchall()

  for pr in productRows:
    for lr in locationRows:
```

```
        cur.execute("SELECT * FROM Balance WHERE locationName = ? AND productName = ?
",(lr["locationName"],pr["productName"]))

        data = cur.fetchall()


        if len(data) == 0:

          cur.execute("INSERT INTO Balance (locationName, productName, qty)VALUES
(?,?,?)",(lr["locationName"],pr["productName"],0))

          con.commit()



   return render_template('ProductMovement.html',rows = rows,  productRows =  productRows, locationRows =
locationRows)


           # ADD ProductMovement


@app.route('/addProductMovement',methods = ['POST'])

def addProductMovement():

   if request.method == 'POST':

      try:

        pn = request.form['pn']

        datetime = request.form['datetime']

        fromlocation = request.form['fromlocation']

        tolocation = request.form['tolocation']

        pq =request.form['pq']



        with sql.connect("database.db") as con:

          cur = con.cursor()

          cur.execute("INSERT INTO product_movement (productName,Timing,fromlocation,tolocation,QTY)
VALUES (?,?,?,?,?)",(pn,datetime,fromlocation,tolocation,pq) )


          con.commit()

          msg = "Record added"

      except:

        con.rollback()
```

```
        msg = "error in  operation"


    finally:
        return redirect(url_for('ProductMovement')+"?msg="+msg)
        con.close()
            #Edit ProductMovement
@app.route('/editProductMovement',methods = ['POST'])
def editProductMovement():
  if request.method == 'POST':
    try:
      movementID = request.form['movementID']
      ProductName = request.form['NEWProductName']
      datetime = request.form['NEWDateTime']
      fromlocation = request.form['NEWfromlocation']
      tolocation = request.form['NEWtolocation']
      qty=request.form['NEWProductQty']
      cur.execute("UPDATE product_movement SET productName = ?,Timing = ?,fromlocation = ?,tolocation =
?,QTY = ? WHERE movementID = ?",(ProductName,datetime,fromlocation,tolocation,qty,movementID),)


      con.commit()
      msg = " movement Edit Successfully"
    except:
      con.rollback()
      msg = "error operation"


    finally:
      return redirect(url_for('ProductMovement')+"?msg="+msg)
      con.close()
        #Delete Product Movement
@app.route('/deleteprouctmovement/<movementID>')
def deleteprouctmovement(movementID):
    try:
        cur.execute("DELETE FROM product_movement WHERE movementID = ? ",(movementID))
```

```
            con.commit()
            msg = "movement Delete Successfully"
        except:
            con.rollback()
            msg = "error operation"

        finally:
            return redirect(url_for('ProductMovement')+"?msg="+msg)
            con.close()



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

Github repo link – https://github.com/IBM-EPBL/IBM-Project-11569-1659334827

Demo video link - https://youtu.be/wK8O_7rE5Uc