

# **INVENTORY MANAGEMENT SYSTEM FOR RETAILERS**

(CLOUD APPLICATION DEVELOPMENT)

*A NAALAIYA THIRAN PROJECT REPORT*

SUBMITTED BY

RISHI RAGUL.R	-	610519104082
NETHA AJI.M	-	610519104071
SEKAR.K.A	-	610519104090
RUTHIRA KANNAN.K	-	610519104084
PRAKASHMANI.K	-	610519104076

**TEAM ID: PNT2022TMID29830**

# Project Report Format

1. **INTRODUCTION**
  - 1.1 Project Overview
  - 1.2 Purpose
2. **LITERATURE SURVEY**
  - 2.1 Existing problem
  - 2.2 References
  - 2.3 Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
  - 3.1 Empathy Map Canvas
  - 3.2 Ideation & Brainstorming
  - 3.3 Proposed Solution
  - 3.4 Problem Solution fit
4. **REQUIREMENT ANALYSIS**
  - 4.1 Functional requirement
  - 4.2 Non-Functional requirements
5. **PROJECT DESIGN**
  - 5.1 Data Flow Diagrams
  - 5.2 Solution & Technical Architecture
  - 5.3 User Stories
6. **PROJECT PLANNING & SCHEDULING**
  - 6.1 Sprint Planning & Estimation
  - 6.2 Sprint Delivery Schedule
  - 6.3 Reports from JIRA
7. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
  - 7.1 Feature 1
  - 7.2 Feature 2
  - 7.3 Database Schema (if Applicable)
8. **RESULTS**
  - 8.1 Performance Metrics
9. **ADVANTAGES & DISADVANTAGES**
10. **CONCLUSION**
11. **GitHub & Project Demo Link**

# **1.INTRODUCTION**

## **1.1 Project overview**

The project Inventory Management System is a complete desktop based application designed on .Net technology using Visual Studio Software. The main aim of the project is to develop Inventory Management System Model software in which all the information regarding the stock of the organization will be presented. It is an intranet based desktop application which has admin component to manage the inventory and maintenance of the inventory system

## **1.2 Purpose**

The project is aimed at developing a cloud based application named Inventory Management system for managing the inventory system for retailers. The Inventory Management System refers to the system and process to manage the stock of organization. This system can be used to store the details of the inventory, stock maintenance, update the inventory based on the sales details, generate sales and inventory report on customer data. In this system we are solving different problem affecting to direct sales management and purchase management. Without proper inventory control, a large retail store may run out of stock on an important item. This application will alert when there is a out of stock through messages or email then the customer will know while ordering if the requested materials is in stock or not. Once the retailers login to the application they can update their inventory stocks, then the customer/user will be able to add new stocks by submitting details related to the stocks, they can view the details of the inventory before ordering. The main objective of the project is to help retailers track and mange stock related to their own products.

## 2.LITERATURE SURVEY

### 2.1 Existing problem

**AUTHOR NAME:** K. Ranganath, V.Vijaya Lakshmi

**OBJECTIVE:** Inventory Management is a crucial aspect of managing a company successfully. Inventory is a vital part of current assets mainly in manufacturing concerns. Huge funds are committed to inventories as to ensure smooth flow of production to meet consumer demand. Maintaining Inventory also involves holding or carrying costs along with opportunity cost. An efficient inventory management ensures continuous production by maintaining inventory at a satisfactory level. It also minimizes capital investment and cost of inventory by avoiding stock-pile of product. Efficient and Effective Inventory Management goes a long way in successful running and survival of business firm

**Reference :** Indian Journal of Research

**AUTHOR NAME:** Mario Pena

**OBJECTIVE:** In this article aims to analyze and present an extensive literature concerning inventory management, containing multiple definitions and fundamental concepts for the retail sector. A systematic literature review was carried out to determine the main trends and indicators of inventory management in Small and Medium-sized Enterprises (SMEs). This research covers five years, between 2015 and 2019, focusing specifically on the retail sector. The primary outcomes of this study are the leading inventory management systems and models, the Key Performance Indicators (KPIs) for their correct management, and the benefits and challenges for choosing or adopting an efficient inventory control and management system.

**Reference :** 2021 Second International Conference on Information Systems and Software Technologies (ICI2ST)

**AUTHOR NAME:** Punam Khobragade\*, Roshni Selokar\*, Rina Maraskolhe\*

Prof.Manjusha Talmale+

**OBJECTIVE:** Inventory Management System is software which is helpful for the businesses operate hardware stores, where storeowner keeps the records of sales

and purchase. Mismanaged inventory means disappointed customers, too much cash tied up in warehouses and slower sales. This project eliminates the paper work, human faults, manual delay and speed up process. Inventory Management System will have the ability to track sales and available inventory, tells a storeowner when it's time to reorder and how much to purchase. Inventory Management System is a windows application developed for Windows operating systems which focused in the area of Inventory control and generates the various required reports.

**Reference:** International Research Journal of Engineering and Technology (IRJET)

## **2.3 Problem Statement Definition**

The project is aimed at developing a cloud based application named Inventory Management system for managing the inventory system for retailers. The Inventory Management System refers to the system and process to manage the stock of organization.

This system can be used to store the details of the inventory, stock maintenance, update the inventory based on the sales details, generate sales and inventory report on customer data. In this system we are solving different problem affecting to direct sales management and purchase management. Without proper inventory control, a large retail store may run out of stock on an important item. This application will alert when there is a out of stock through messages or email then the customer will know while ordering if the requested materials is in stock or not.

Once the retailers login to the application they can update their inventory stocks, then the customer/user will be able to add new stocks by submitting details related to the stocks, they can view the details of the inventory before ordering. The main objective of the project is to help retailers track and mange stock related to their own products.

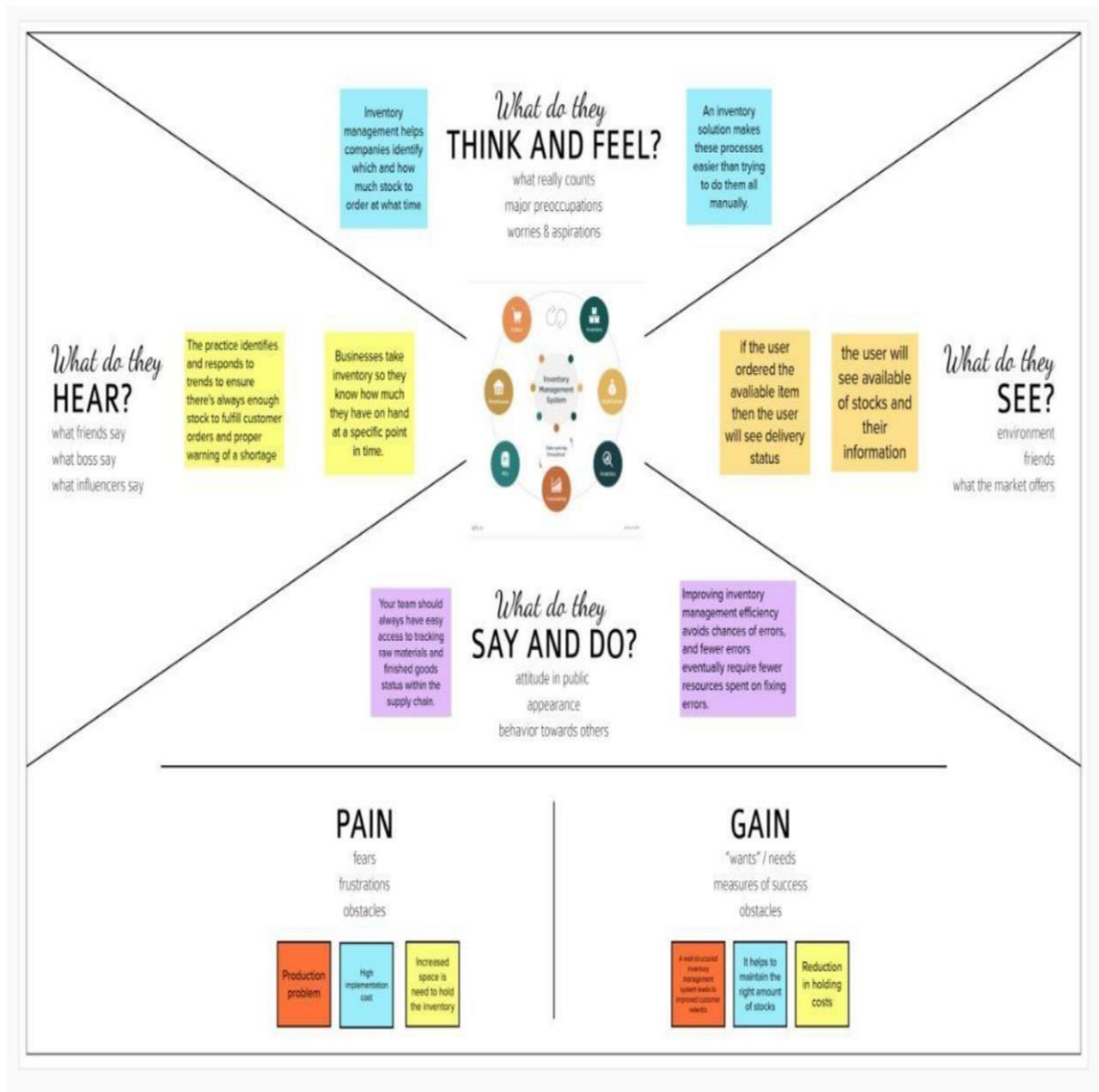
## **3.IDEATION & PROPOSED SOLUTION**

### **3.1 Empathy Map Canvas**

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

#### **Inventory Management System for Retailers:**

The project is aimed at developing a cloud based application named Inventory Management system for managing the inventory system for retailers. The Inventory Management System refers to the system and process to manage the stock of organization. This system can be used to store the details of the inventory, stock maintenance, update the inventory based on the sales details, generate sales and inventory report on customer data. In this system we are solving different problems affecting direct sales management and purchase management. Without proper inventory control, a large retail store may run out of stock on an important item. This application will alert when there is a stock out through messages or email then the customer will know while ordering if the requested materials are in stock or not. Once the retailers login to the application they can update their inventory stocks, then the customer/user will be able to add new stocks by submitting details related to the stocks, they can view the details of the inventory before ordering. The main objective of the project is to help retailers track and manage stock related to their own products.



## 3.2 Ideation & Brainstorming

Brainstorm & Idea: The main idea of the project is to help retailers track and manage stock related to their own products. The requirements from the shopkeeper to create backup inventory within limited time and in high accuracy makes us to come up with automation solution by using desktop. The main goals are:

1. To track and maintain the stocks for retailers.
2. Alert if the stock is going to finish to retailers to add that products.
3. Send a notification regarding offers to shopkeepers.

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

1

#### Define your problem statement

The project is aimed at developing a cloud based application named Inventory Management system for managing the inventory system for retailers.

🕒 5 minutes

PROBLEM

How might we [your problem statement]?



#### Key rules of brainstorming

To run a smooth and productive session



Stay in topic.



Encourage wild ideas.



Defer judgment.



Listen to others.



Go for volume.



If possible, be visual.



## Step-2: Brainstorm, Idea Listing and Grouping

2

### Brainstorm

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

🕒 10 minutes

**Rishi Ragul.R**

push notification  
to retailers  
through mail  
when the stock  
is going to finish

**Nethaaji.M**

The delviery  
status of the  
stock will be  
send to  
shopkeepers

**Sekar.K**

Top selling  
items will be  
displayed for  
the shop  
keepers

**Ruthira Kannan.K**

The manufacture  
date and the expiry  
date of the product  
will be displayed  
when the product  
is selected

**Prakashmani.K**

convert a  
sales order  
into an  
invoice,send  
it via email

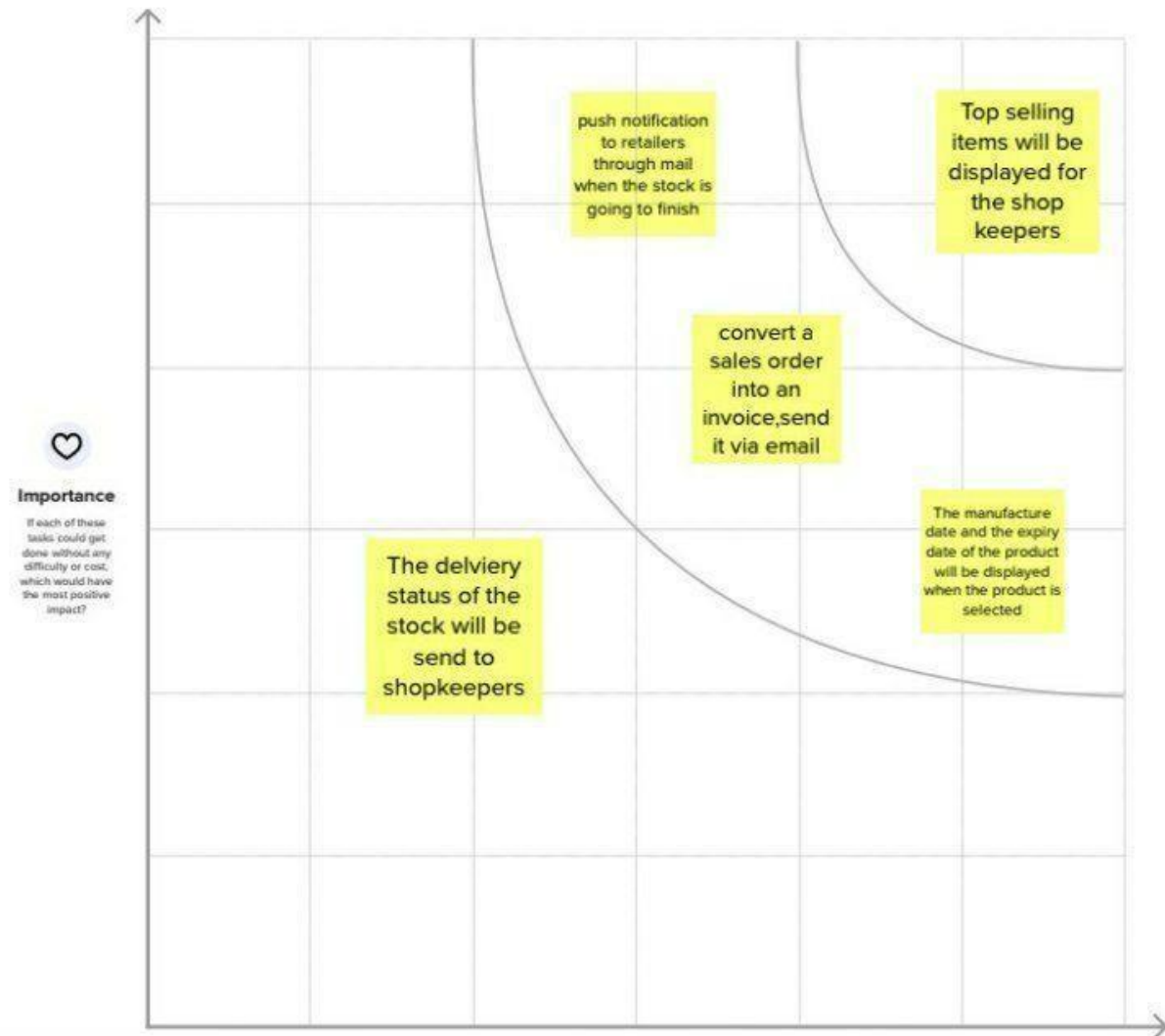
## Step-3: Idea Prioritization

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

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The main objective of the project is to help retailers track and manage stock related to their own products.
2.	Idea / Solution description	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.
3.	Novelty / Uniqueness	In this application we send an alert message to the retailers if the stock is going to finish, through which the retailers can manage the stock
4.	Social Impact / Customer Satisfaction	Through this application the retailers can manage the stocks effectively and the retailers can save the time by only updating the products in this application instead of doing manually.
5.	Business Model (Revenue Model)	Particular amount of rupees are charged for a referral.
6.	Scalability of the Solution	Chatbots facility is given.

### 3.4 Problem Solution fit

Project Title: Inventory Management system for Retailers

Project Design Phase – 1  
Problem Solution Fit

Team Id: PNT2022TMID29830

Define CS, fit into	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> Who is your customer? <p>The retailers and shopkeepers are the customers in our project, the retailers can update and maintain the inventory and shopkeepers can check for their product in the inventory</p>	<b>6. CUSTOMER</b> <span>CC</span> What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices. <p>The main constraint is that retailers can maintain stocks manually, but in this project after the product is purchased the inventory will be automatically updated</p>	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e., pen and paper is an alternative to digital notetaking <p>Maintain excel sheet. Inventory dairy to maintain stocks. Pros: Through this method consumption of time is high Cons: By our project we can maintain stocks in cloud and maintain inventory neatly.</p>	Explore AS,
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>J&amp;P</span> Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides. <p>Theory, there are three types of jobs-to-be done your customer is trying to get done - functional, emotional and consumption</p>	<b>9. PROBLEM ROOT CAUSE</b> <span>RC</span> What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e., customers have to do it because of the change in regulations. <p>Lack of real time engagement in many scenarios, the retailers are need instant inventory report and notifications on products availability to retailers.</p>	<b>7. BEHAVIOUR</b> <span>BE</span> What does your customer do to address the problem and get the job done? i.e., directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace) <p>Stay calm. Treat the situation with humor rather than getting angry. Distract their attention, rather than getting confrontational. If other people are present, explain to them that the behavior is because of an illness and is not personal</p>	
Identify strong TR & EM	<b>3. TRIGGERS</b> <span>TR</span> These interactions/triggers are pieces of information which users create through their everyday actions that indicate they are either a information or not. 	<b>10. YOUR SOLUTION</b> <span>SL</span> The database created for the retailers and connected with the website through which the retailers can see the products status and availability, and shopkeeper can see the products in the inventory for their purchase after purchase the inventory will be automatically updated.	<b>8. CHANNELS of BEHAVIOUR</b> <span>CH</span> <b>8.1 ONLINE</b> The retailers and shopkeepers can login to website and ask any related queries to that bot and agent, the bot/ agent can able to answer the queries. <b>8.2 OFFLINE</b> The retailers can directly visit to the respected organization or show room and ask any related questions.	Extract online & offline CH of BE
	<b>4. EMOTIONS: BEFORE / AFTER</b> <span>EM</span> The customer feels very bad. Do not try to talk over the customer or argue with them. Let the customer have their say, even if you know that they are mistaken and don't have all the information, or you can anticipate what they are asking.			

## 4.REQUIREMENT ANALYSIS

### 4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
1	User Registration	<ul style="list-style-type: none"><li>• Registration through registration form.</li><li>• Registration through One-Tap Google Sign- in.</li></ul>
2	User Confirmation	<ul style="list-style-type: none"><li>• Authentication via Google Authentication.</li><li>• Confirmation via Email.</li><li>Confirmation via OTP.</li></ul>
3	Product Management	<ul style="list-style-type: none"><li>• Quickly produce reports for single or multiple products.</li><li>• Track information of dead and fastmoving products.</li><li>Track information of suppliers and manufacturers of the product.</li></ul>
4	Audit Monitoring	<ul style="list-style-type: none"><li>• The technique of tracking crucial data is known as audit tracking.</li><li>Monitor the financial expenses carried out throughout the whole time (from receiving order of the product to delivery of the product).</li></ul>
5	Historical Data	Data of everything should be stored for analytics and forecasting

6	CRM (Customer Relationship Management)	<ul style="list-style-type: none"> <li>Track the customer experience via ratings given by them.</li> <li>Get customer reviews regularly or atleast at the time of product delivery to work on customer satisfaction.</li> </ul>
		<ul style="list-style-type: none"> <li>User-friendly GUI to increase the customer base from only techies to normal people.</li> </ul>
7	Security Policy	<ul style="list-style-type: none"> <li>User data collected must be as secure as possible.</li> <li>User data must not be misused. They can only be used for user preferred advertising purposes.</li> </ul>

## 4.2 Non-functional Requirements:

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

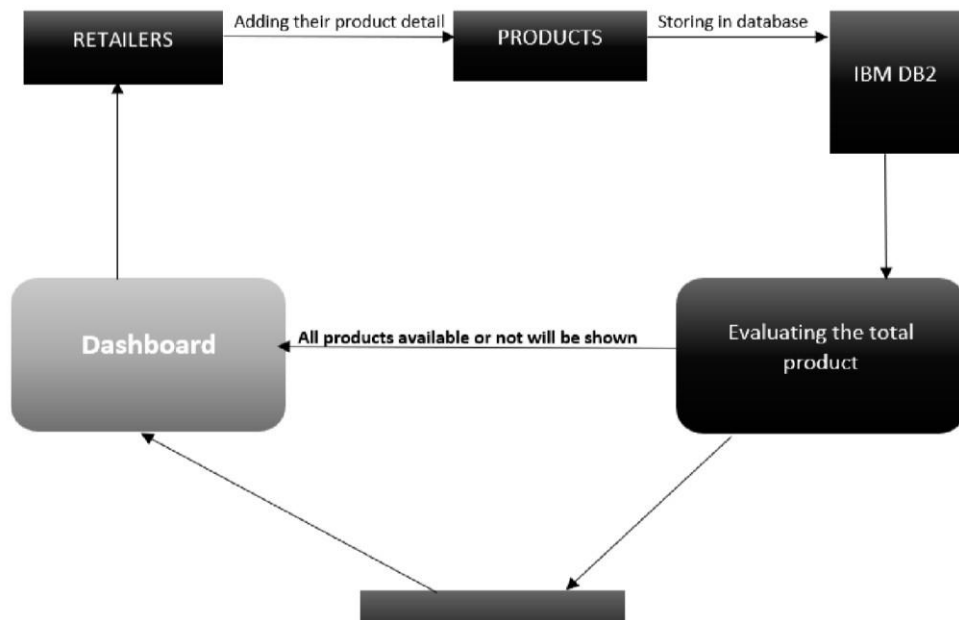
FR No.	Non-Functional Requirement	Description
1	Usability	<p>The UI should be accessible to everybody despite of there diversity in languages.</p> <p>People with some impairments should also be able to use the application with ease. (Example, integrate google assistant so that blind people can use it).</p>
2	Security	<ul style="list-style-type: none"> <li>The security requirements deal with the primary security. Only authorized users can access the system with their credentials.</li> </ul> <p>Administrator or the concerned security team should be alerted on any unauthorized access or data breaches so as to rectify it immediately.</p>

3	Reliability	<ul style="list-style-type: none"> <li>• The software should be able to connect to the database in the event of the server being down due to a hardware or software failure.</li> <li>• The users must be intimated by the periodic maintenance break of the server so that they will be aware of it.</li> </ul>
4	Performance	<ul style="list-style-type: none"> <li>• Performance of the app should be reliable with high-end servers on which the software is running.</li> </ul>
5	Availability	<ul style="list-style-type: none"> <li>• The software should be available to the users 24/7 with all functionalities working.</li> </ul> <p>New module deployment should not impact the availability of existing modules and their functionalities.</p>
6	Scalability	The whole software deployed must be easily scalable as the customer base increases.

## 5. PROJECT DESIGN

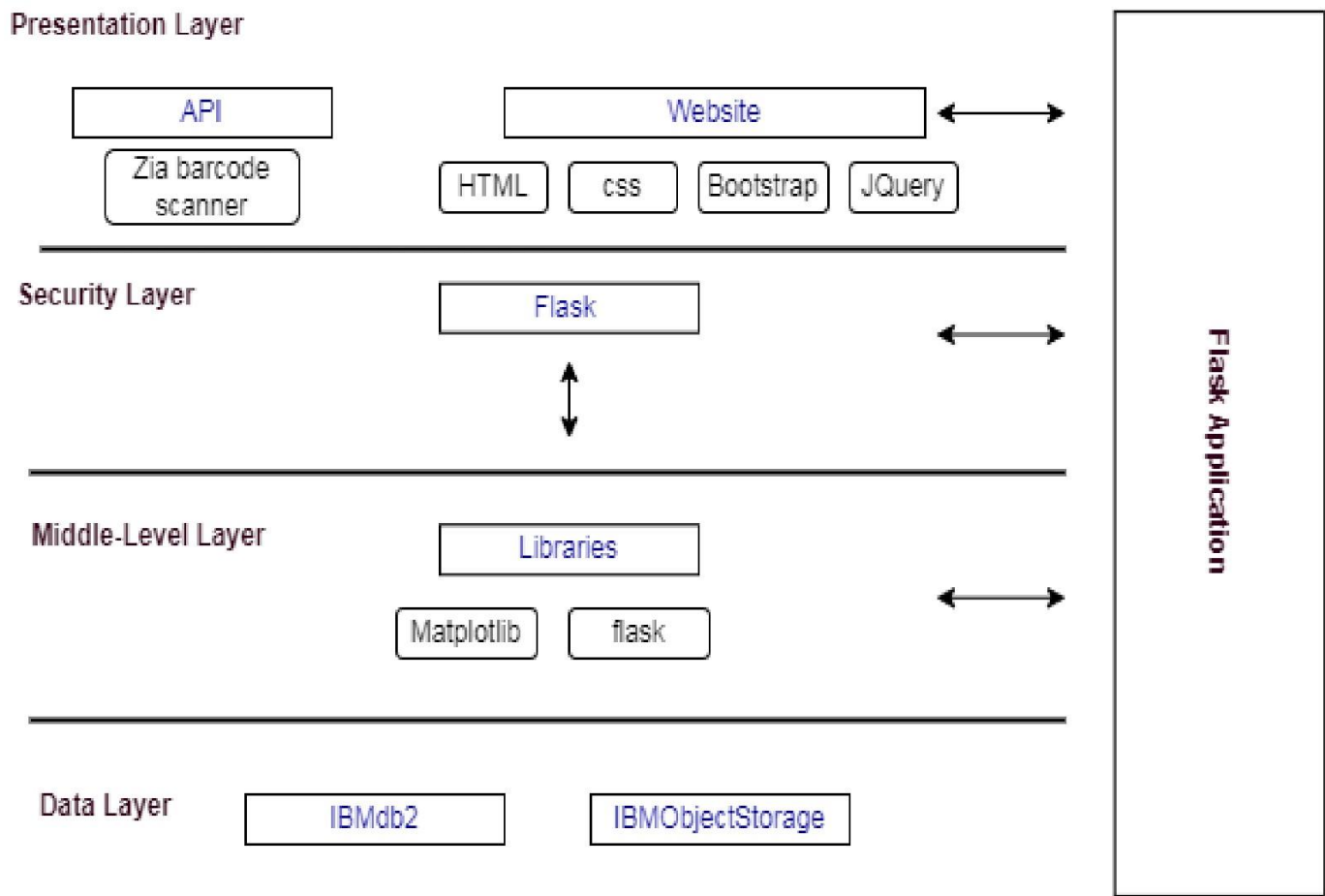
### 5.1 Data Flow Diagrams

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





5.2 Solution & Technical Architecture



**Table-1 : Components & Technologies:**

S.No	Component	Description	Technology
1	User Interface	Web UI with Chatbot	HTML, CSS, Bootstrap, jQuery
2	Calculating Products Count	By entering barcode details into the application	Zia Barcode Scanner
3	Showing high demand product	By the products data in IBMdb2	Data Visualization using Python Bar plot by Matplot Library
4.	Alert and Notification	Alerting the retailers regarding the low stock count of the product	SendGrid
5	Chat	Chat with Watson assistant	IBM Watson Assistant
6	Cloud Database	Database Service on Cloud	IBM DB2
7	File Storage	File storage requirements	IBM Object Storage
8	External API-1 Barcode	To Scan the product barcode	Zia Barcode Scanner
9	Infrastructure (Server / Cloud)	Cloud Server Configuration	Cloud Foundry, Kubernetes

**Table-2: Application Characteristics:**

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Styling our page, Python flask microframework	Python Flask, Bootstrap
2.	Security Implementations	For securing our cloud data	SSL Certificates
3.	Scalable Architecture	Three – tier architecture (MVC)	Web server - HTML, CSS, JavaScript Application server - Python Flask, Docker, Container Registry Database server - IBM DB2
4.	Availability	availability of application	IBM Load Balancer
5.	Performance	5 requests per seconds, Use of Local Machine Cache Memory	IBM Cloud, CDN

## 5.3 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
Retailer(Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I will be redirected to login page	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can verify the OTP number	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can access my account / dashboard .	High	Sprint-1
	Dashboard	USN-6	As a user,I can update stock in & out count details	I can access my account / dashboard	High	Sprint-1
	Dashboard	USN-7	As Administrator can hold the extract details of patient and available of plasma	Updation can be made through barcode scanning	High	Sprint-2

		USN-8	As a user,I can check the total product details	I can view the value of total	Medium	Sprint-2
				products in the stock .		

		USN-9	As a user,I can check the high demand product details	I can update sales details of the products	High	Sprint-2
		USN-10	As a user,I can generate the invoice details user also help the user	I can add incoming stock details .	High	Sprint-1

## 6.PROJECT PLANNING & SCHEDULING

### 6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Panel(Shopkeeper Panel)	USN-1	The user will login into the website and select their respective Domain (shopkeepers/Retailers). The role of shopkeeper is to check out the the	20	High	Rishi Ragul.R  Nethaaji.M

			availability of stocks needed by the shopkeeper..			
Sprint-2	Retailer Panel	USN-2	The role of the retailers is to check out the inventory details and update the stocks in inventory and maintain the stocks by connecting with database	20	High	RuthiraKannan.K Sekar.K.A
Sprint-3	Admin Panel	USN-3	The shopkeeper details and purchases are maintained in a database. And to maintain the inventory for the retailers while updating and maintaining the database.	20	High	Prakashmani.K  Rishi Ragul.R
Sprint-4	Chat Bot	USN-4	The user can directly talk to Chatbot regarding the services. Get the recommendations based on information provided by the user.	20	High	Sekar.K.A  Nethaaji.M

Sprint-5	Final Delivery	USN-5	Container of applications using docker kubernetes and deployment the application. Create the documentation and final submit the application	20	High	Rishi Ragul.R  Prakashmani.K  Ruthira Kannan.K
----------	----------------	-------	---	----	------	--

### Project Tracker, Velocity & Burndown Chart

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Completed on Planned Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	3 Days	31 Oct 2022	05 Nov 2022	20	02Nov 2022
Sprint-3	20	3 Days	07 Nov 2022	12 Nov 2022	20	06 Nov 2022
Sprint-4	20	6 Days	14Nov 2022	19 Nov 2022	20	12 Nov 2022
Sprint-5	20	6 Days	13 Nov2022	19 Nov 2022	20	19 Nov 2022

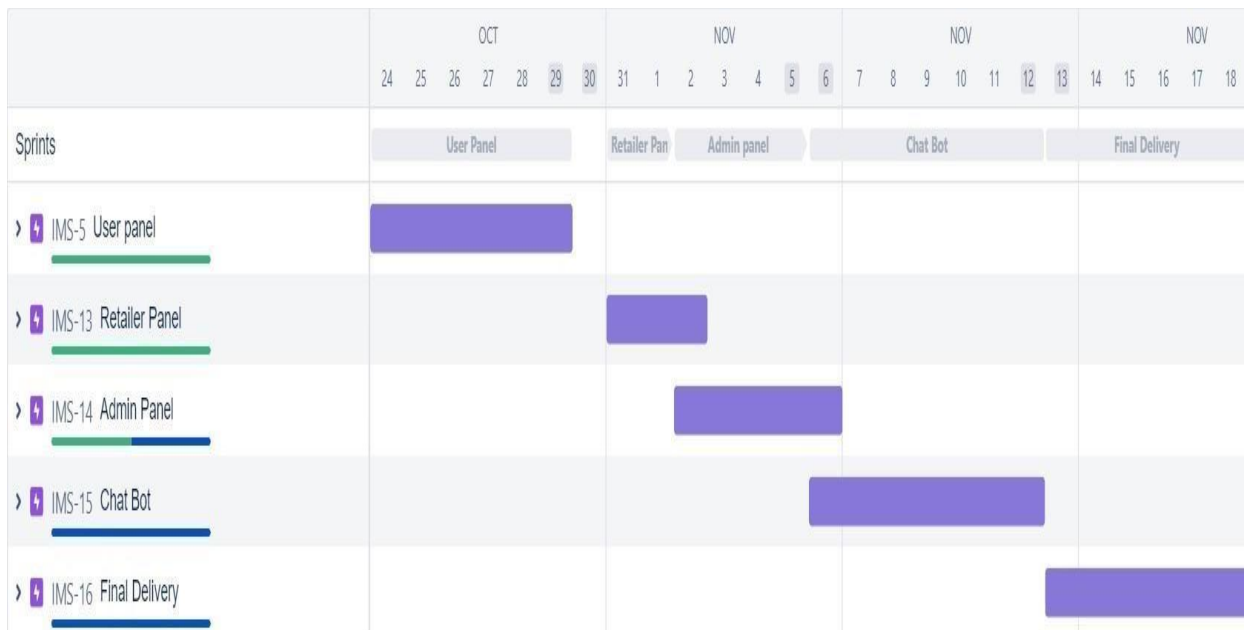
### Velocity:

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

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

## Burndown Chart:

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



## 7.CODING

```
from flask import Flask,redirect, render_template, request, session, url_for, flash
from werkzeug.utils import secure_filename
from importlib.resources import contents
from tkinter import S
from turtle import title
from turtle import from
from turtle import model
from sqlalchemy import PrimaryKeyConstraint
import ibm_db
from markupsafe import escape
import os
import requests
import json
url = "https://www.fast2sms.com/dev/bulkV2"

headers = {
    'authorization':
'xe1nQcD4A8tFWRIXHyRhPLqKEl7Zpg9VUCYGd3j5MTiSzO6omNSP2LpXfBuqC5J6kTa04KmlgVZ7GHEA',
    'Content-Type': "application/x-www-form-urlencoded",
    'Cache-Control': "no-cache"
}

app = Flask(name)
app.secret_key='oiuytrekjhgfd\\124!@#$$%^'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=0c77d6f2-5da9-48a981f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31198;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=tfc22793;PWD=IGXPWDZrWRG7eqRu","","")
print(conn)
print("connection successful...")

@app.route('/')
def home():
```



```
message = "TEAM ID : PNT2022TMID29830" + " " + "BATCH ID : B1-1M3E "  
return render_template('index.html',mes=message)
```

```
@app.route('/login', methods=['GET','POST'])  
def login(): return  
render_template('login.html')
```

```
@app.route('/register', methods = ['GET','POST'])  
def register(): return  
render_template('register.html')
```

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

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

```
@app.route('/changepass', methods = ['GET','POST'])  
def changepass(): return  
render_template('changepassword.html')
```

```
@app.route('/signup', methods=['GET', 'POST']) def  
signup():  
    if request.method == 'POST':  
  
        name = request.form['name']  
email = request.form['email']  
        phonenumber = request.form['phonenumber']  
        password = request.form['password']
```

```
sql = "SELECT * FROM shopkeeper WHERE email=?"
```

```

        stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,email)
    ibm_db.execute(stmt)        account =
    ibm_db.fetch_assoc(stmt)    if
    account:
        return render_template('index.html', msg="You are already a member, please
        login using your details....")

```

else:

```

        insert_sql = f"INSERT INTO shopkeeper VALUES (?, ?, ?, ?);"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, name)        ibm_db.bind_param(prepare_stmt,
    2, email)        ibm_db.bind_param(prepare_stmt, 3, phonenumber)
    ibm_db.bind_param(prepare_stmt, 4, password)        ibm_db.execute(prepare_stmt)
        return render_template('login.html', msg="User Data saved successfully..")

```

```

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

```

```

def add():    if request.method == 'POST':

```

```

        productid= request.form['product_id']
    productname= request.form['product_name']
        quantity = request.form['quantity']
        price= request.form['price']

```

```

        insert_sql = f"INSERT INTO inventory VALUES (?, ?, ?, ?);"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, productid)        ibm_db.bind_param(prepare_stmt,
    2, productname)        ibm_db.bind_param(prepare_stmt, 3, quantity)
    ibm_db.bind_param(prepare_stmt, 4, price)        ibm_db.execute(prepare_stmt)
    return redirect(url_for('customize'))

```

```

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

```

```

def delete():    if request.method == 'POST':

```

```

    productid= request.form['productid']
        sql = f"SELECT * FROM inventory WHERE productid = ?"
        stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,productid)
        cusdel = ibm_db.execute(stmt)

```

```

if cusdel:
    sql = f"delete from inventory where productid = ?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,productid)
ibm_db.execute(stmt)
    return redirect(url_for("customize"))

```

```

Cse Rishi, [19-11-2022 20:55] def
executeCountQuery(tableName,columnName):    sql = "SELECT
COUNT("+columnName+") FROM "+ tableName
    stmt=ibm_db.exec_immediate(conn,sql)
count_id=ibm_db.fetch_both(stmt)
    return count_id.get('1')

```

```

@app.route('/dashboard') def
dashboard():

```

```

    sql = "SELECT COUNT(productid) FROM inventory where quantity <= 3"
stmt=ibm_db.exec_immediate(conn,sql)    count_id=ibm_db.fetch_both(stmt)
    print(count_id)
data = {
    'totalcount' : executeCountQuery("inventory","productid"),
    'customer' : executeCountQuery("shopkeeper","email"),
    'lowstocks' : count_id.get('1')
}
    return render_template('dashboard.html', data = data)

```

```

@app.route('/update',methods=['GET', 'POST']) def
update():

```

```

    if request.method == 'POST':
productid = request.form["productid"]
        quantity = request.form["quantity"]
        sql="SELECT quantity FROM inventory WHERE productid = "+productid
stmt = ibm_db.exec_immediate(conn,sql)
        oldQuantity=ibm_db.fetch_both(stmt).get('QUANTITY')
        newQuantity= int(quantity) + oldQuantity
        sql = "UPDATE inventory SET quantity=? WHERE productid = ?"
stmt = ibm_db.prepare(conn,sql)

```

```

ibm_db.bind_param(stmt,1,newQuantity)
ibm_db.bind_param(stmt,2,productid)
    ibm_db.execute(stmt)
flash("Updated Successfully")
    return redirect(url_for('customize'))

```

```

@app.route('/lowstockdis') def
lowstockdis():
lowstockdis= []
    sql = "SELECT * FROM inventory WHERE quantity<=3"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)

    while dictionary != False:
        lowstockdis.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)
    print(dictionary)

    if lowstockdis:
        sql = "SELECT * FROM inventory"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)
        return render_template('lowstocks.html', lowstockdis = lowstockdis)

```

```

@app.route('/forget', methods=['GET', 'POST'])
def forget():    if request.method == 'POST':
cm = request.form["Email"]        cp =
request.form["oldpassword"]        co =
request.form["newpass"]
        sql = "UPDATE retailer SET password= ? WHERE email = ?"
        stmt = ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,co)        ibm_db.bind_param(stmt,2,cm)
        ibm_db.execute(stmt)
        return redirect(url_for('home'))

```

```

@app.route('/signin', methods=['GET', 'POST']) def
signin():

```

```

app.secret_key='oiuytrekjhgfd\\124!@#$$%^'
if request.method == 'POST':
    mail = request.form['em']
    password = request.form['pass']
    print(mail, password)

```

```

    sql = f"select * from retailer where email='{escape(mail)}' and password=
    '{escape(password)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    data = ibm_db.fetch_both(stmt)

```

```

if data:
    session["email"] = escape(mail)
    session["password"] = escape(password)
    return redirect(url_for('dashboard'))

```

```

else:
    return redirect(url_for("login",msg = "Account does not exists or invalid"))
@app.route('/signinsk', methods=['GET', 'POST']) def signinsk():

```

```

app.secret_key='oiuytrekjhgfd\\124!@#$$%^'
if request.method == 'POST':
    mail = request.form['em']
    password = request.form['pass']

```

```

    sql = f"select * from shopkeeper where email='{escape(mail)}' and password=
    '{escape(password)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    data = ibm_db.fetch_both(stmt)

```

```

if data:
    session["email"] = escape(mail)
    session["password"] = escape(password)
    return redirect(url_for('dashboardsk'))

```

```

else:
    return redirect(url_for("login",msg = "Account does not exists or invalid"))

```

Cse Rishi, [19-11-2022 20:55]

```
@app.route('/inventory') def  
inventory():    inventory = []
```

```
    sql = "SELECT * FROM INVENTORY "  
stmt = ibm_db.exec_immediate(conn, sql)  
dictionary = ibm_db.fetch_both(stmt)    while  
dictionary != False:  
    inventory.append(dictionary)  
dictionary = ibm_db.fetch_both(stmt)
```

```
    if inventory:  
        sql = "SELECT * FROM INVENTORY"  
stmt = ibm_db.exec_immediate(conn, sql)  
user = ibm_db.fetch_both(stmt)  
length=len(inventory)  
alertmessage="  
lowStockItemsString = ""  
for i in inventory:    if  
i[2] == 1 :  
    lowStockItemsString = lowStockItemsString + i[1]+" "  
print(lowStockItemsString)    my_data = {  
    # Your default Sender ID  
    'sender_id': 'FTWSMS',  
  
    # Put your message here!  
    'message': 'low stocks on following items'+ lowStockItemsString,  
  
    'language': 'english',  
    'route': 'p',  
  
    # You can send sms to multiple numbers  
    # separated by comma.  
  
    'numbers': '6385517795'  
}  
response = requests.request("POST",url,data = my_data,headers = headers)  
#load json data from source  
returned_msg = json.loads(response.text)
```

```

    # print the send message
    print(returned_msg['message'])
    return render_template('inventory.html', inventory = inventory)

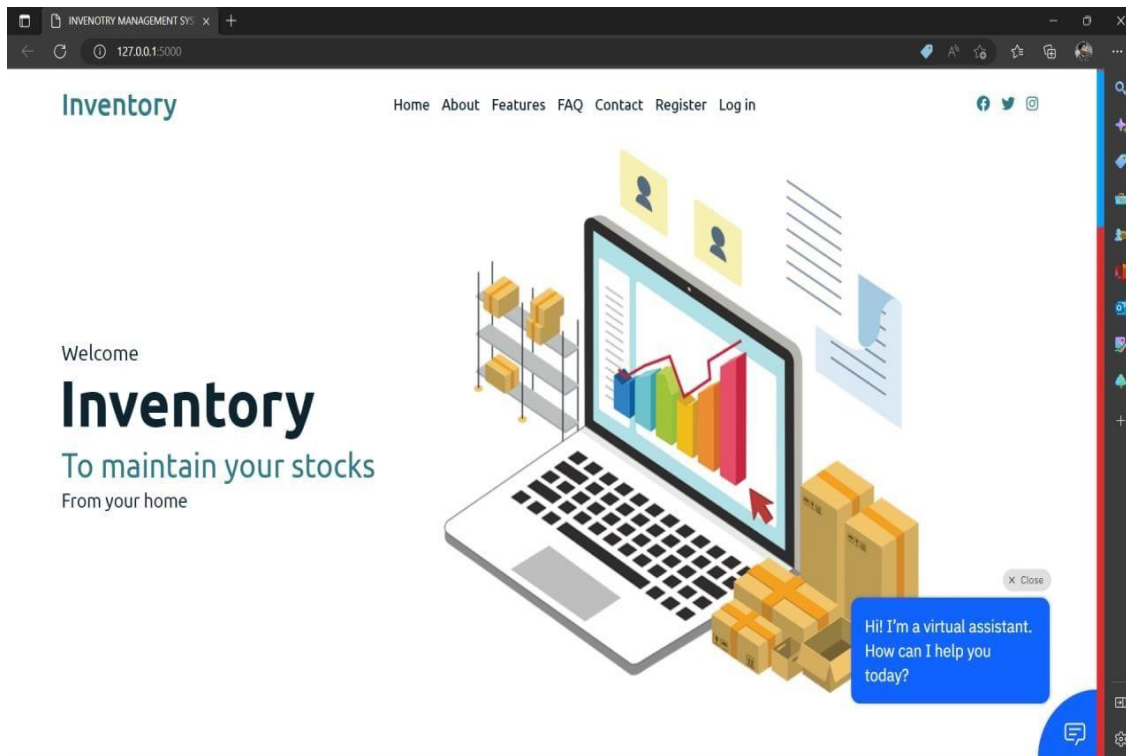
@app.route('/customer')
def customer():
    customer = []
    sql = f"SELECT NAME FROM SHOPKEEPER"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)    while
    dictionary != False:
        customer.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)    if
    customer:
        sql = f"SELECT NAME FROM SHOPKEEPER"
        stmt = ibm_db.exec_immediate(conn, sql)        user =
        ibm_db.fetch_both(stmt)
        return render_template('customers.html', customer = customer)

if name == "main":
    app.run(host='0.0.0.0',port=5000,debug=True)

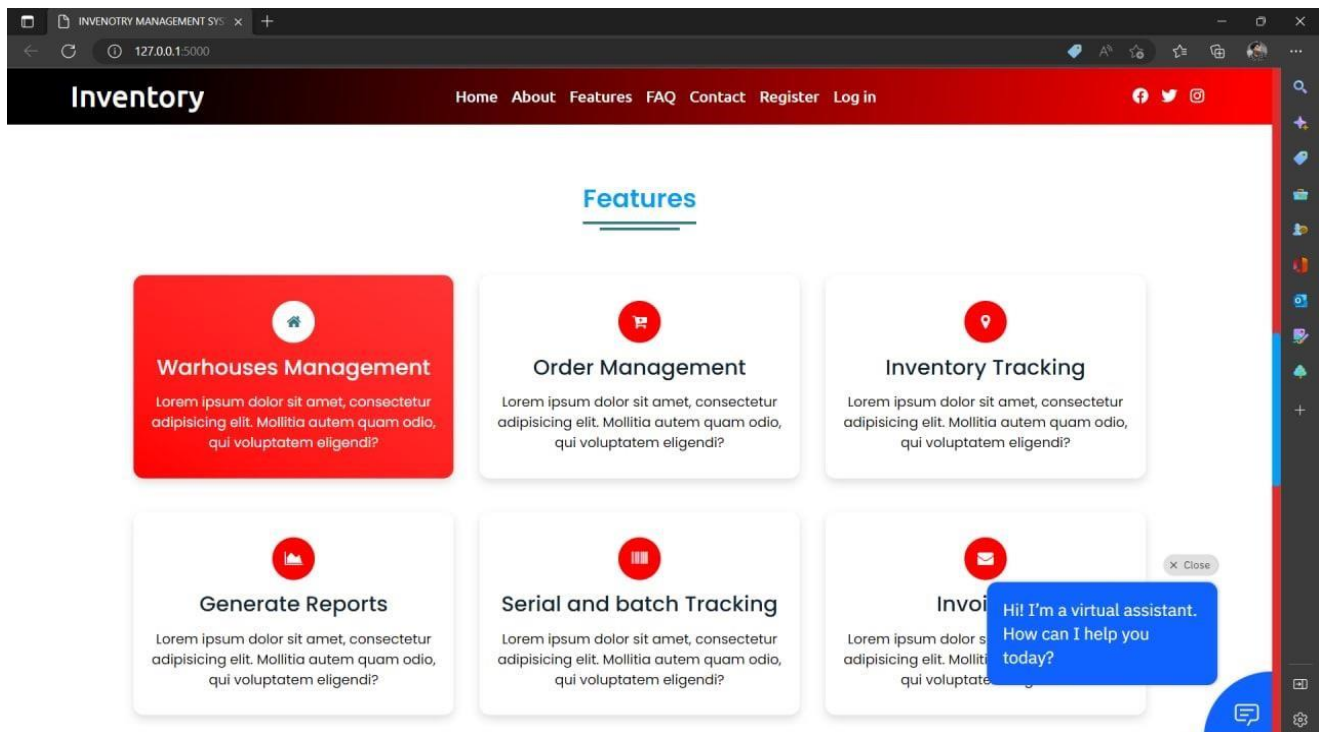
```

## 8 RESULTS

### HOME PAGE:



### FEATURES:





REGISTRATION:

Signup

127.0.0.1:5000/register

SHOPKEEPER REGISTER

name

134

phone no.


\*\*\*\*

Already have an account [login](#)

REGISTER

I'm Admin!

Enter your personal details and start journey



RETAILER AND SHOPKEEPER LOGIN

Login Form

127.0.0.1:5000/login

RETAILER LOGIN

EMAIL

PASSWORD

[FORGOT PASSWORD?](#)

LOG IN

SHOPKEEPER LOGIN

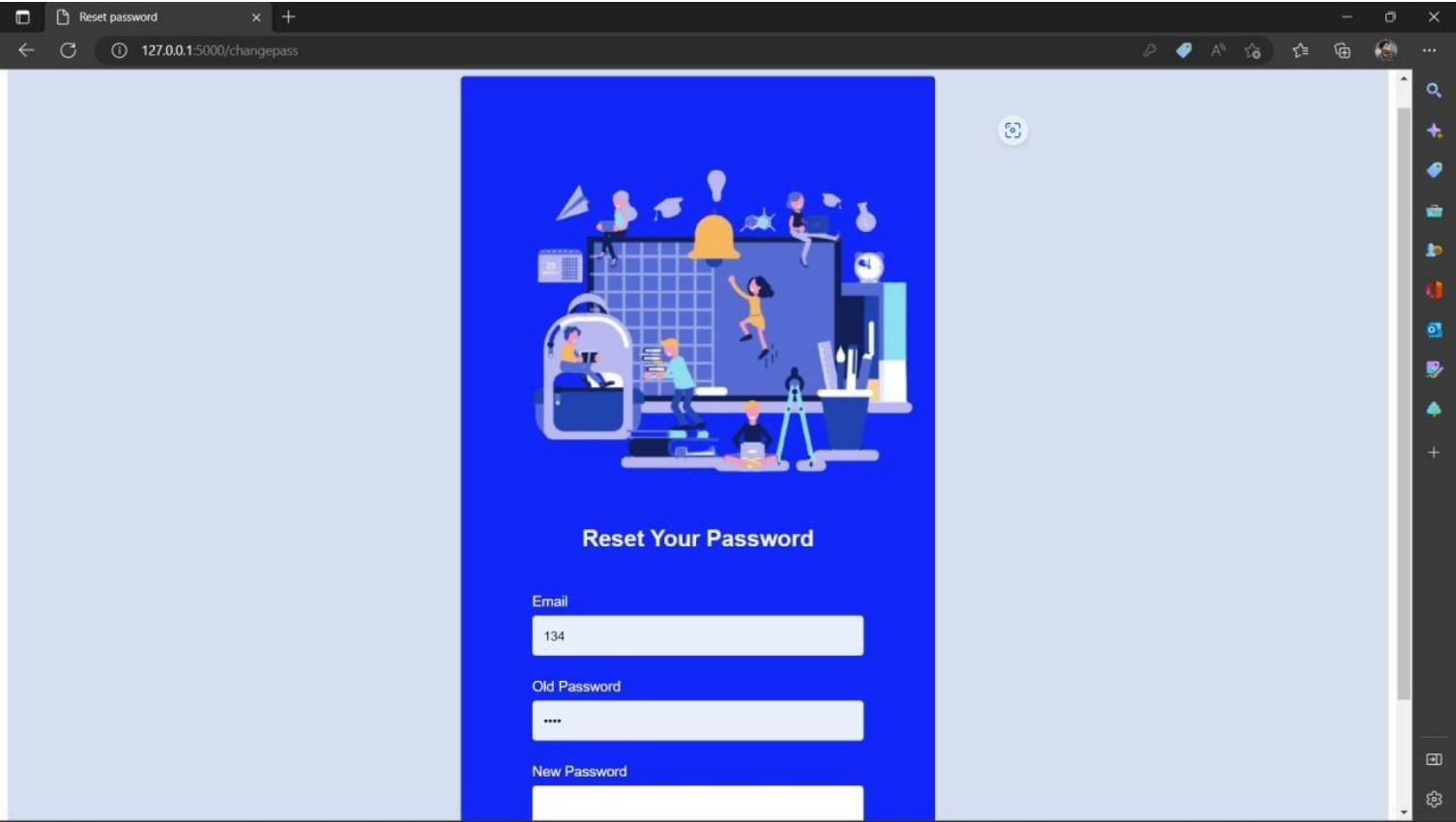
134

\*\*\*\*

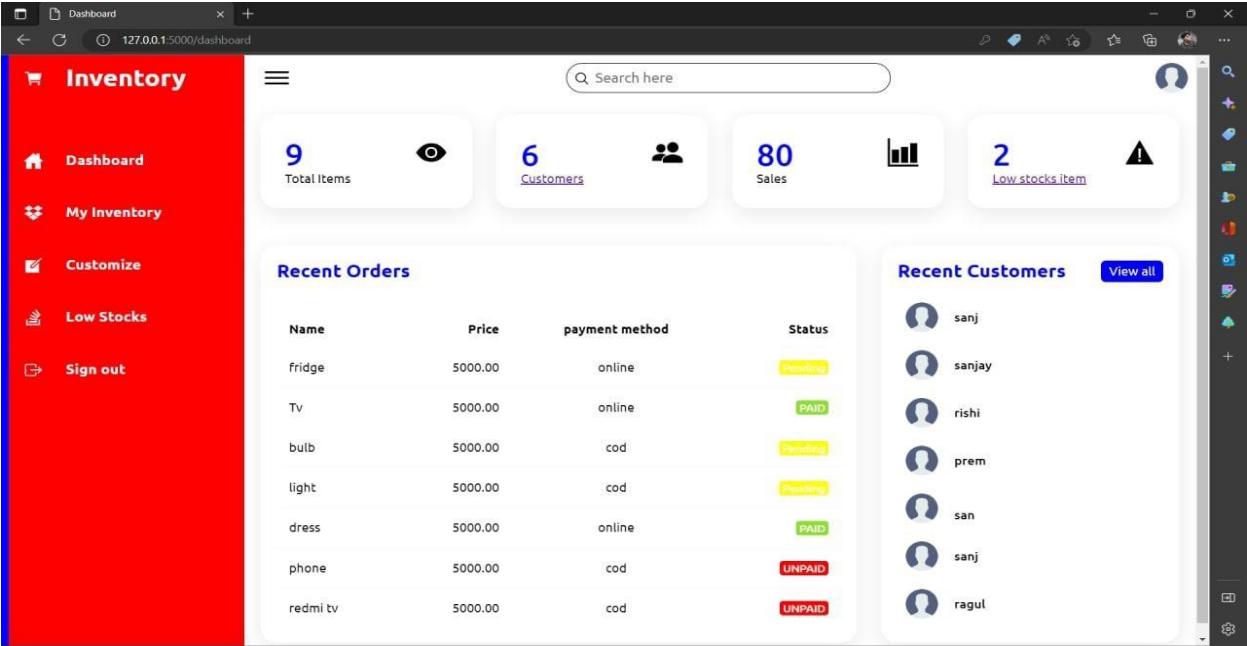
[FORGOT PASSWORD?](#)

LOG IN

RESET PASSWORD:



DASHBOARD:



# MY INVENTORY:

127.0.0.1:5000/inventory

MY INVENTORY

INVENTORY			
PRODUCT_ID	PRODUCT_NAME	QUANTITY	PRICE
2	zebronic companion 104 wireless keyboard	10	380
3	zeb juke bar 3901	17	2000
4	sony inzone m9 monitor	2	2000
5	samsung monitor m8	10	2500
6	intel core i7 13700k 13th gen	10	25000
8	tp-link pc-usb bluetooth adaptor	10	200
9	zeb usb 150w/1 wifi adaptor	10	200
10	lenovo pc fhd webcam	10	2000
7	zebronic zeb-dazzle wireless mouse	1	2000

# CUSTOMIZING INVENTORY:

Document1127.0.0.1:5000/customize

CUSTOMIZE YOUR INVENTORY

PRODUCT ID

PRODUCT NAME

QUANTITY

PRICE

PRODUCT ID

PRODUCT NAME

QUANTITY

PRICE

ADD

TO DELETE THE PRODUCT

PRODUCT ID

productid

DELETE

TO UPDATE THE PRODUCT

PRODUCT ID

productid


QUANTITY

UPDATE

# LOW STOCKS:

127.0.0.1:5000/lowstockdis

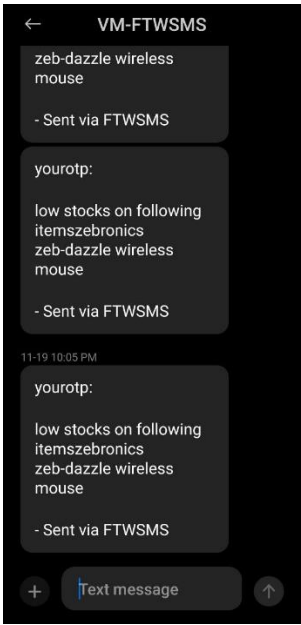
127.0.0.1:5000/lowstockdis



LOW STOCKS

LOW STOCK ITEMS			
PRODUCT_ID	PRODUCT_NAME	QUANTITY	PRICE
4	sony inzone m9 monitor	2	2000
7	zebronics zeb-dazzle wireless mouse	1	2000

# INSTEAD OF SENDGRID WE USE FAST 2 SMS SERVICE



## **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.