NALAIYA THIRAN - IBM PROJECT REPORT

ON

INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

Submitted by

TEAM ID: PNT2022TMID23391

RUBESH U	**(113219031126)**
DHINESH KUMAR K	**(113219031038)**
KISHORE D	**(113219031075)**
SANJAY G	**(113219031130)**
VALATHURU NIRANJAN	**(113219031154)**

TABLE OF CONTENTS

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.

1.2 PURPOSE

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.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

STOCK MANAGEMENT

In general, manual updating of inventory in some registers/notebooks may cause running out of stocks or to carry extra stocks. As a result of which, if an item's stock gets over, its sale is paused until a new one arrives after ordering. And carrying of extra stocks may not be sold for a long period of time

SALES PATTERN

Generally, if the sales pattern of a product is not known, the retailer will not know exactly the number of products to be ordered each time.

ITEMS MANAGEMENT

Manual updating of stocks and inventory for a large list of products is time consuming and also many errors occur if everything is done manually in a notebook/register.

2.2 REFERENCES

- **Inventory Management Challenges for B2C E-Commerce Retailers**

AUTHOR NAME: Harish Patil and Rajiv Divekar

OBJECTIVE: To study the challenges such as demand variations, reverse logistics, seasonal fluctuations, and stockless policy involved in inventory management of a B2C e-commerce business and how to mitigate the same to enhance the level of customer satisfaction by efficient inventory management.

- **Influence of Information Technology, Skills and Knowledge and Financial Resources on Inventory Management Practices Amongst Small and Medium Retailers**

  AUTHOR NAME: Tuan Zainun Tuan Mat, Nor raihan Md Johari, Maz Ainy Abdul Azis and Mohd ridzuan Hashim

  OBJECTIVE: Small-medium Enterprises (SMEs) play a vital role in the Malaysian economy. One of the rapidly growing SMEs in Malaysia is the retail industry. One important element in improving the growth of SME retailers is inventory management, as it assists the SME retailers in managing their inventories. SMEs face difficulties in securing financial resources, which inhibits the adoption of computerised inventory systems, as well as limited skill and knowledge in managing their inventory, are among the major problems that causes a less effective inventory management in retail SMEs.

- **Inventory Management and Its Effects on Customer Satisfaction**

  AUTHOR NAME: Scott Grant Eckert

  OBJECTIVE: This study examines inventory management and the role it plays in improving customer satisfaction. It looks at how food companies have been under pressure to streamline their inventory systems, and the consequences of such actions. It also examines how many retailers are trying to implement a “perfect order” system and how suppliers are constantly under pressure to meet the demands of these retailers.

- **The Effects of Inventory Management Practices on Operational Performance**

AUTHOR NAME: Jacklyne Bosibori Otundo and Dr. Walter Okibo Bichanga

OBJECTIVE: The study's general objective is to evaluate the effects of inventory management practices . i)To establish the effects of demand forecasting ii)To investigate the effects of inventory categorization iii)To determine the effects of Vendor managed inventory (VMI).

- **Simulation of inventory management systems in retail stores**

  AUTHOR NAME: Puppala Sridhar, C.R.Vishnu, R Sridharan

  OBJECTIVE: Inventory management has become a key factor in today's world of uncertainty, particularly in the retail sector. Accordingly, there is a high requirement of managing and controlling the inventory with appropriate policies to elevate the organisation's performance. In fact, a proper system has to be implemented for monitoring customer demand. This system will, in turn, assist in maintaining the right level of inventory. In this direction, the present research focuses on a retail store and explores a solution for an inventory-related problem experienced by the firm. A simulation model is developed and run for particular merchandise using Arena simulation software.

2.3 PROBLEM STATEMENT DEFINITION

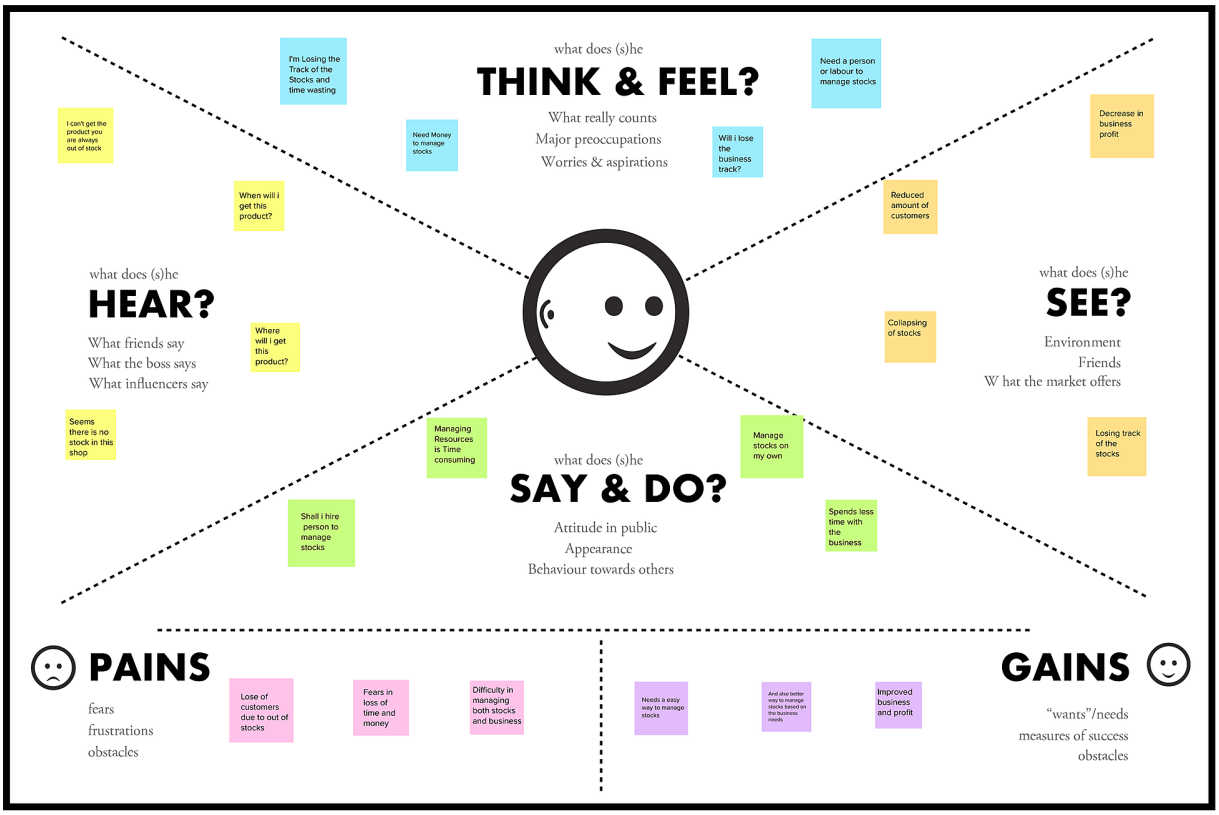
- The Retailers need to maintain their stocks such that they should not carry too much stocks or very high amount of stocks.By managing an inventory they will not run out of stocks.
- It can be accomplished with the help of an software that they can get a visual perspective of their running business and also they can easily manage their business.
- With this software they can be able to login and then update the inventory and add more stock which are newly available and also view the current stocks in the

inventory.

- The additional information like most sold products and least sold products and the lower stock items will be displayed.
- A bar chart for the sales of the product will be displayed so that the user can add more stocks based on the demand.
- If there is no stock is found the user will be sent an mail and they can add a new stock to the inventory.
- The web app enables the registered user to update the inventory and as well as add a new product if the product details are given. As everything is automated no error occurs in this process and it is not time consuming as well.
- The user will be able to make a sale to consume the products in the inventory and also review the sales pattern
- An alert is sent automatically by the inventory management system if the stock left count reaches a threshold value and as soon as the alert is received, the stocks required are ordered and as a result pausing of sale is avoided.
- A graphical representation of the sales pattern is also provided by the app from the sales undertaken till now, and from which the quantity of stocks required each month are accordingly ordered.

3. IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING

2

Brainstorm

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

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Team Lead - Rubesh U

- An Application that includes all the present date available inventory along with the quantity for both the customer and the retailer.
- To have a track of seasonal selling products and to keep those products in stock during the demand
- Predicting the Future sales analysis of the products using machine learning algorithms and past data available dataset
- Triggering the alert message when the stock falls down the threshold amount
- Providing an easy and user friendly Ecommerce site for the customers.

Team Member - Sanjay G

- Centralized transportation system among the shop branches along with the product tracking functionality.
- Customer Feedback and rating system including both the product and the retail shop service.
- Sending E-mail notifcation to the customer regarding the new arrivals and available stocks.
- Keeping a Track of the expiry dates of all the stock and announcing the discounts and offer for those products which is going to expire soon.
- Plan appropriate strategic business plans with regard to the competitors and bring the plan noticeable among the customers.

Team Member - Kishore D

- Bring RFID based product tracking system into the existence.
- Keep a record of regular customers and send them regular notice about the arrivals and exclusive offers and discounts for them.
- Can make use of excel sheet for processing the data
- Advertise the presence of the store in all the nearest geographic locations.
- Provide special discount for the frst purchase and can add key points with further purchase so future special discounts.

Team Member - Niranjan V

- Keep a proft and loss records of all the stocks.
- Make sure that the store contains all the day to day vital used from day to dawn.
- Easy and fast billing system with also provides option for the customers either through cash or through net banking.
- Deciding whether to invest in a product or not using some predictive analysis of the newly arrived product.
- Enhancing customer loyalty and providing transparency in the billing.

Team Member - Dhinesh Kumar K

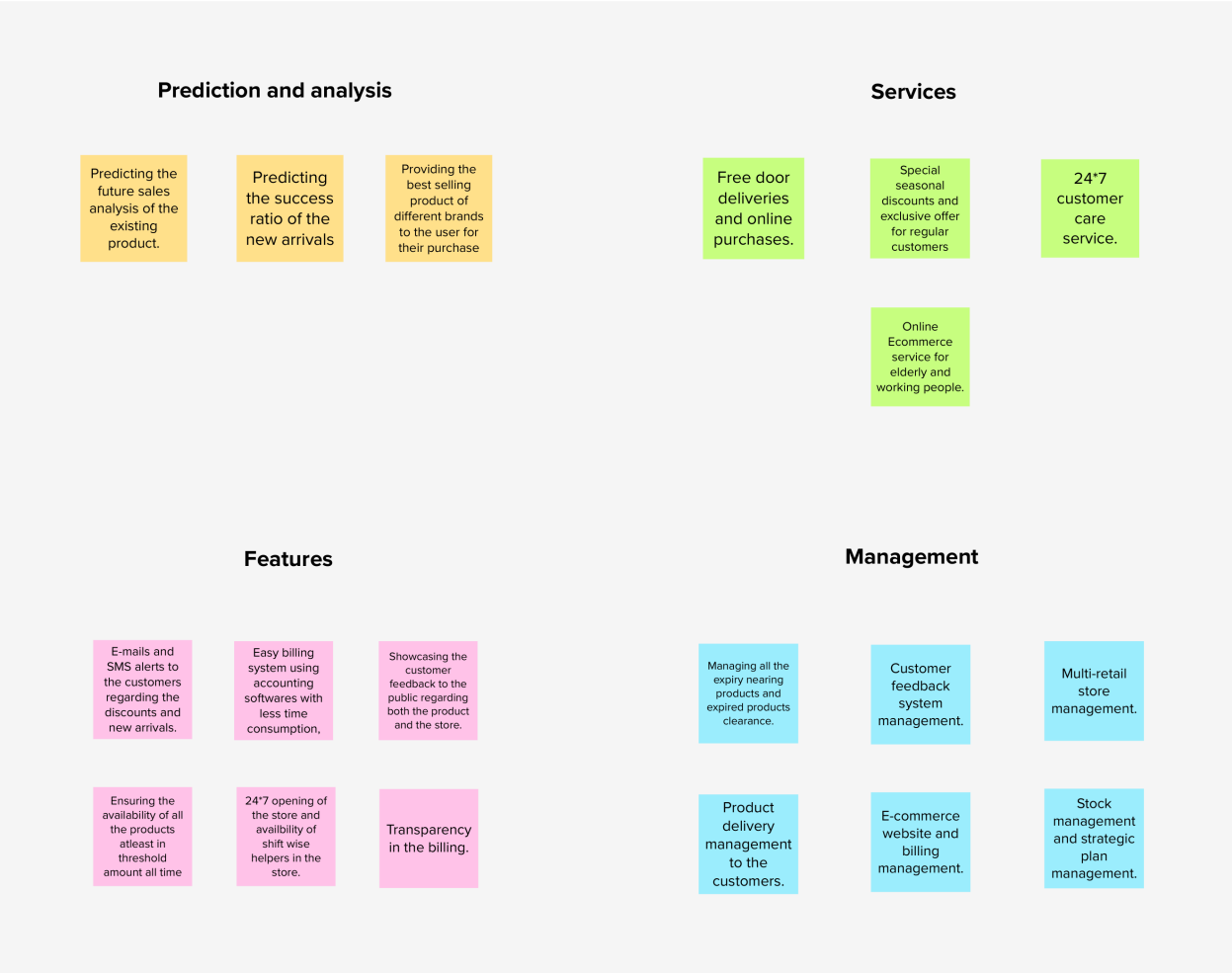
- Providing excellent user interface and user experience
- Tax and GST clearance regularly.
- Make sure to have free door deliveries to the nearest areas and to avoid late deliveries
- Scheduling all the product deliveries properly for maximum utilization of transportation.
- Alerting the user regarding the end sale discounts and real time statistics.

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



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

Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 PROPOSED SOLUTION

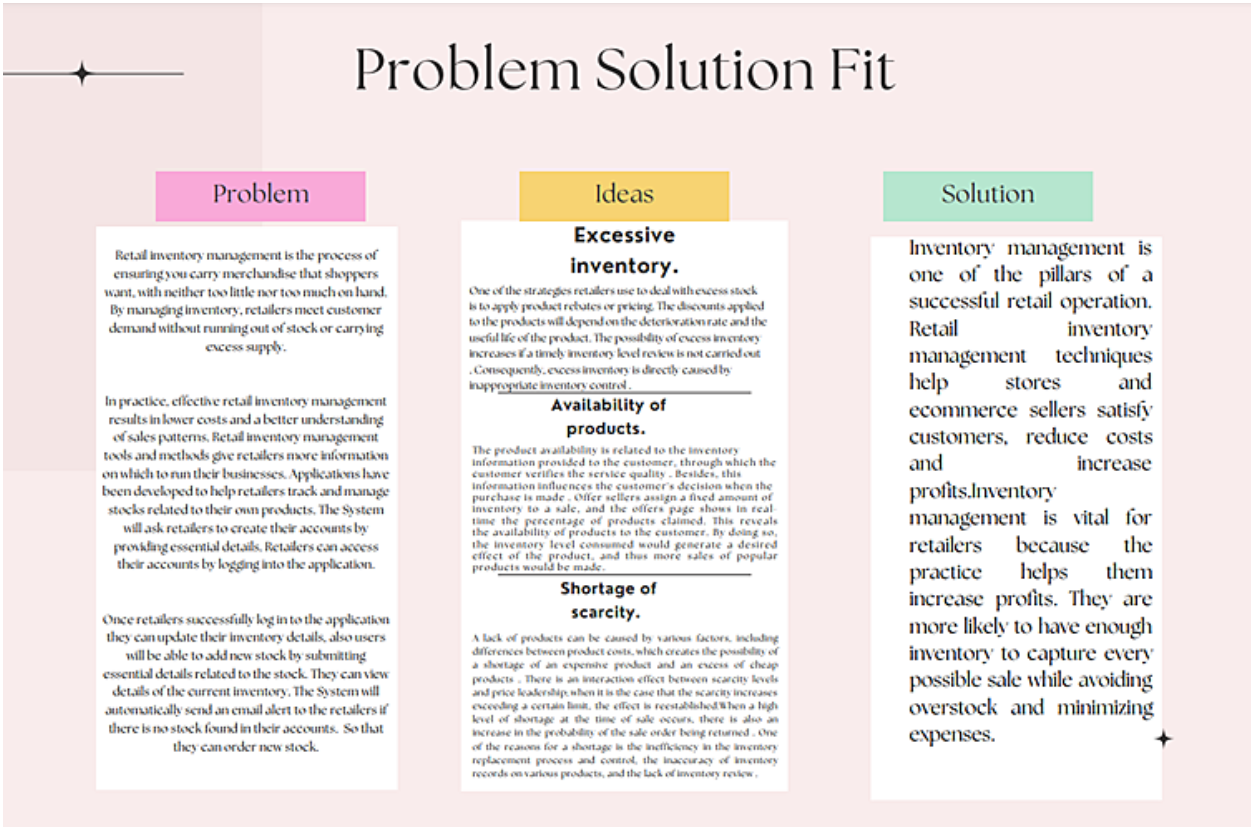
S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	· The retailers generally facing issues in recording the stocks and its threshold limit available. ·The retailers doesn't know which product is getting expired and when it is being expired. · The retailers couldn't track the availability of all the stocks up-to date. · The customers are not satisfied with the retailers store since it doesn't have enough supplements and the deliveries were not made on time
2	Idea / Solution description	· This proposed system will have a daily update system whenever a product is sold or it is renewed more. · The system will have an alert triggered to indicate both the expired product and soon going to expire products. · The product availability is tracked daily and an alert system in again kept on to indicate those products which falls below the threshold limit. · All the customers can register their accounts after which they will be given a login credentials which they can use whenever they feel like buying the stocks. · The application allows the customers to know all the present time available stocks and also when the new stock will

		be available on the store for them to buy. · Tracking the order have become easy with this application for both the retailers and the customers.
3	Novelty / Uniqueness	· Certain machine learning algorithms are used to predict the seasonal high selling products which can be made available during that time. · Prediction of the best selling brand of all certain products based on their popularity, price and customer trust and satisfaction will be implemented. · Notifications will be sent to the retailers if any product that the customers have been looking for is not available so that the product can be stocked up soon. · Notification will be sent to the customers who buys any certain products regularly when the new arrivals are stocked up. · Exclusive discounts and offers are given for regular customers to keep them engaged with the store regularly.

4	Social Impact / Customer Satisfaction	· The customers will be highly satisfied since the wasting of time while searching for an unavailable product is reduced. · The work load of the retailers will be minimized if the system is automated every day and during every purchase. · The customer satisfaction will be improved for getting appropriate response from the retailers and that too immediately
5	Business Model (Revenue Model)	· Hereby we can provide a robust and most reliable inventory management system by using: 1. ML algorithms for all the prediction purposes using all the past dataset since datasets are undoubtedly available in huge amounts. 2. Can deploy the most appropriate business advertising models. 3. To establish a loss preventing strategy. 4. And to ensure the all time, any where availability of products system. 5. Usage of freebies business strategy for dragging the customer's attention.

6	Scalability of the Solution	· This system can even work more efficiently with large volume of data. · Implementation of anyone and anywhere using system can be helpful for even a commoner to buy the products. · Daily and Each time purchase updation of the stock for preventing inventory shrinkage . · Direct chat system with the retailers and the customers for providing best customer service.

3.4 PROBLEM SOLUTION FIT



4. 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 own application Form Registration through Gmail Registration through LinkedIN Registration through Google Docs
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login through User name and password. Login through mail I'D and password. Login through OTP through mail I'd and password. Login through Phone number.
FR-4	Records of the products	Product name Product category Product I'd Stock Count Vendor details
FR-5	Login details	Login Details along with time through E-mail. Login Details along with time through phone number.
FR-6	Updation of inventory Details.	Update through E-mail Update through User account.
FR-7	Unavailability Alert	Alert Message through mail or phone number.
FR-8	Monitoring of stock	Audit monitoring through incoming and outgoing stock.
FR-9	Database	Usage of standard database for storing the data.

4.2 NON-FUNCTIONAL REQUIREMENT

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	· 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. · It can use by wide variety of client as it is very simple to learn and not complex to proceed · Easy to use, User-friendly and Responsive.
NFR-2	Security	· 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. With Registered Mail id only retailers can log into the application. So it provide authentication. · We are using login for the user and the information will be hashed so that it will be very secure to use
NFR-3	Reliability	· It will be reliable that it can update with very time period so that the accuracy will be good.

NFR-4	Performance	· User can track the record of goods available using the application. Inventory tracking helps to improve inventory management and ensures that having optimal stock available to fulfill orders.Reduces manpower , cost and saves time. Emails will be sent automatically While stocks are not available.Makes the business process more efficient.Improves organizations performance. · It will be perform fast and secure even at the lower bandwidth.
NFR-5	Availability	· The availability of product is just one way in which an inventory management system creates customer satisfaction. Inventory management systems are designed to monitor product availability, determine purchasing schedules for better customer interaction. · Prediction will be available for every user but only for premium user news,database and price alert will be alert
NFR-6	Scalability	· Scalability is an aspect or rather a functional quality of a system, software or solution.This proposed system for inventory management system can accommodate expansion without restricting the existing workflow and ensure an increase in the output or efficiency of the process · It is scalable that we are going to use data in kilobytes so that the quite amount of storage is satisfied

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

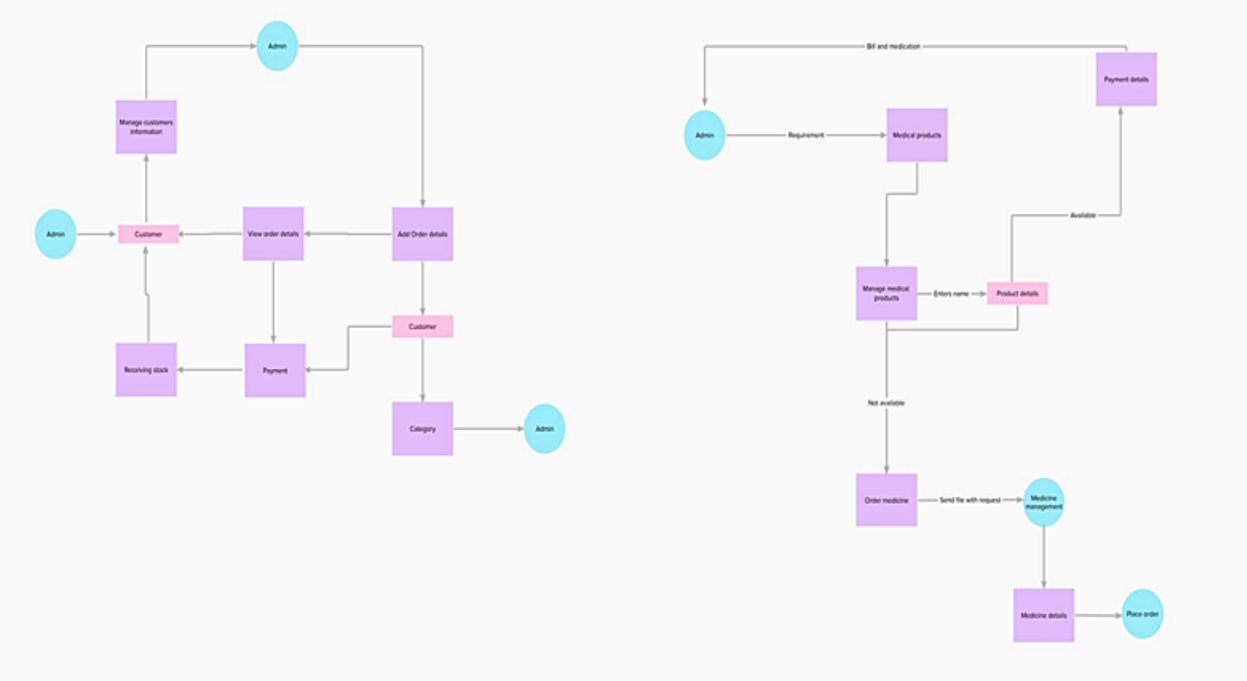


Figure : Data Flow Diagram

5.2 SOLUTION AND TECHNICAL ARCHITECHTURE

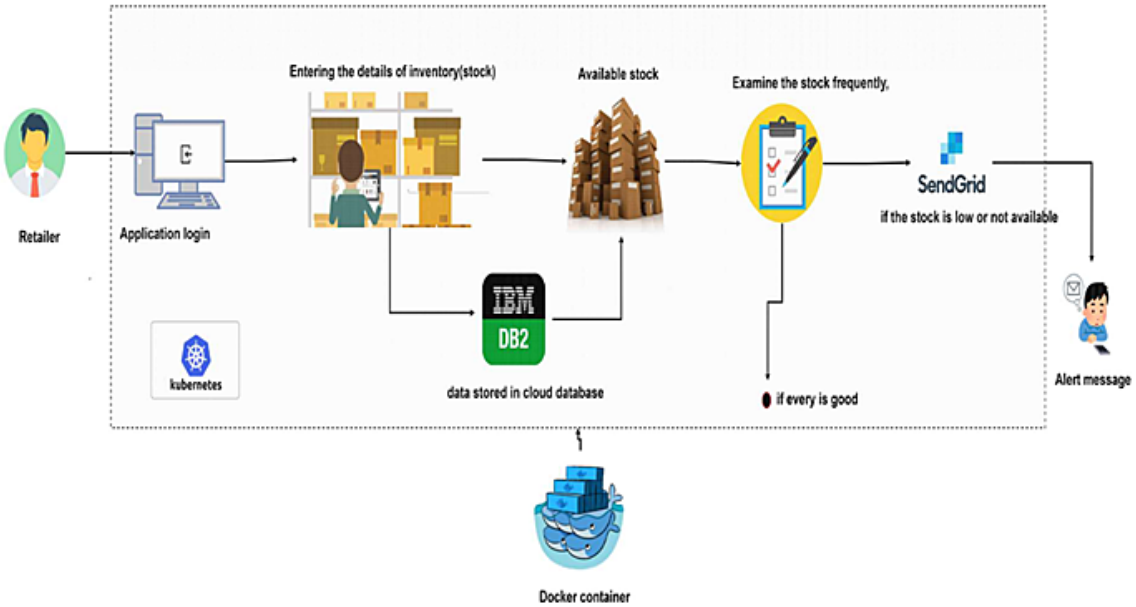


Figure : Solution Architechture

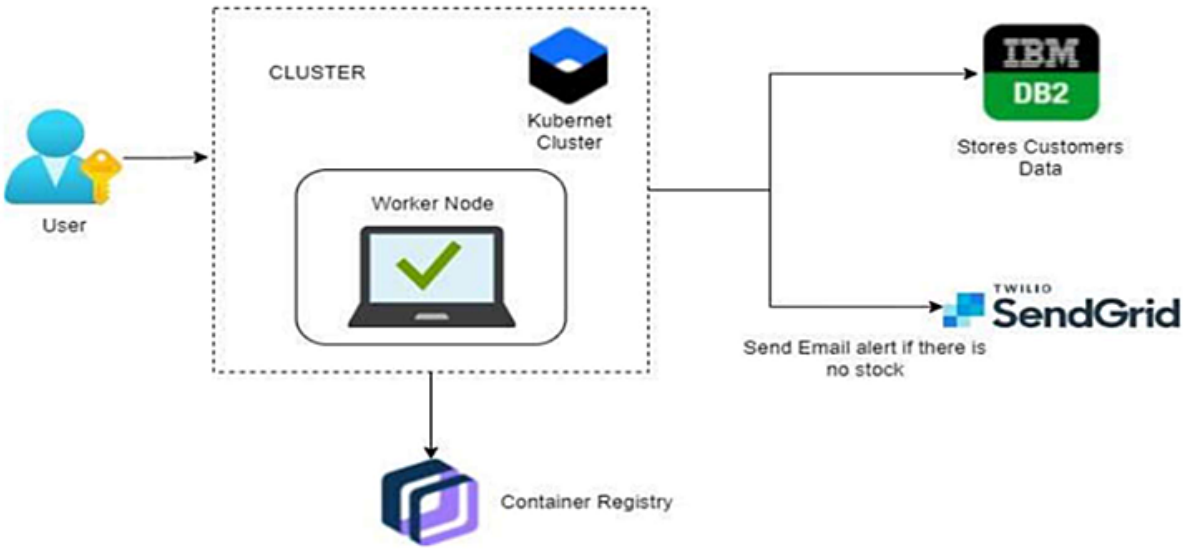


Figure : Technical Architechture

5.3 USER STORIES

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

	Login	USN – 5	As a user, I can log into the application by entering email & password	I can access my account	High	Sprint 1
	Dashboard	USN - 6	As a user,i can log into my account for the mobile	I can access my account /Dashboard	High	Sprint 1
Customer (Web user)	Registration	USN – 7	As a user,I can register for the application by entering my email, password, and confirming my password	I can access my account/Dashboard	High	Sprint 1
		USN - 8	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 - 9	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint 2
		USN – 10	As a user ,I can upload a Profile photo and add my name to my account	I can upload my Profile photo/Name in my account	Medium	Sprint 1
Customer Care Executive	Customer Support	USN – 11	As a user,I can support for customers to handle queries and complaints from their customers	I can support for customers to clear complaints	High	Sprint 1
Administrator	Responsibility	USN - 12	As a system administrator I want to be able to add new users when required so that	I Can add new users	High	Sprint 1

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

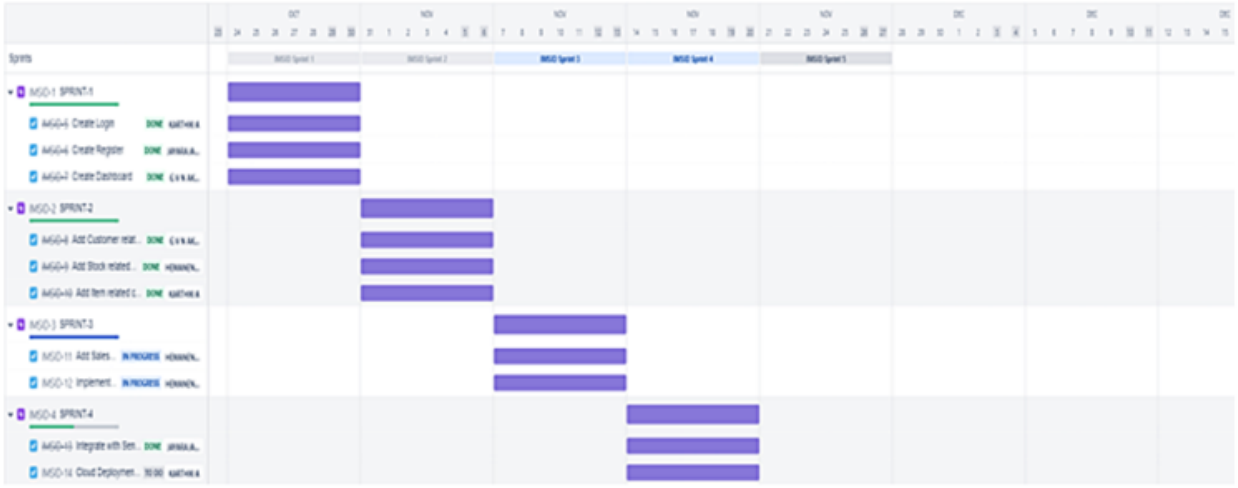
User Type	Functional Requirement (Epic)	User Story Number	User Story/Task	Story Points	Priority	Release
Sprint 1	Registration	USN – 1	User can create an account by providing business mail id and password	5	High	1,2,3,4,5
Sprint 2	Registration /Login	USN – 2	Two step authentication using one time password to provide mail id or phone number	10	High	1,2,3,4,5
Sprint 1	Login	USN – 3	Using registered mail Id	5	High	1,2,3,4,5
Sprint 1	Main dashboard	USN – 4	User need to complete account settings like giving the details about their inventory and their branches	10	High	1,2,3,4,5
Sprint 2	Hub maintenance	USN – 5	User can able to create a separate account for individual hub and he can able to create access policy to share their account with their hub managers	10	High	1,2,3,4,5
Sprint 3	Hub dashboard login	USN – 6	Hub mangers can able to login to the account to access their allotted hub details	10	High	1,2,3,4,5
Sprint 3	Hub dashboard	USN – 7	Hub mangers can able to add product details and production details. They can also provide access to their allotted space to others.	10	High	1,2,3,4,5

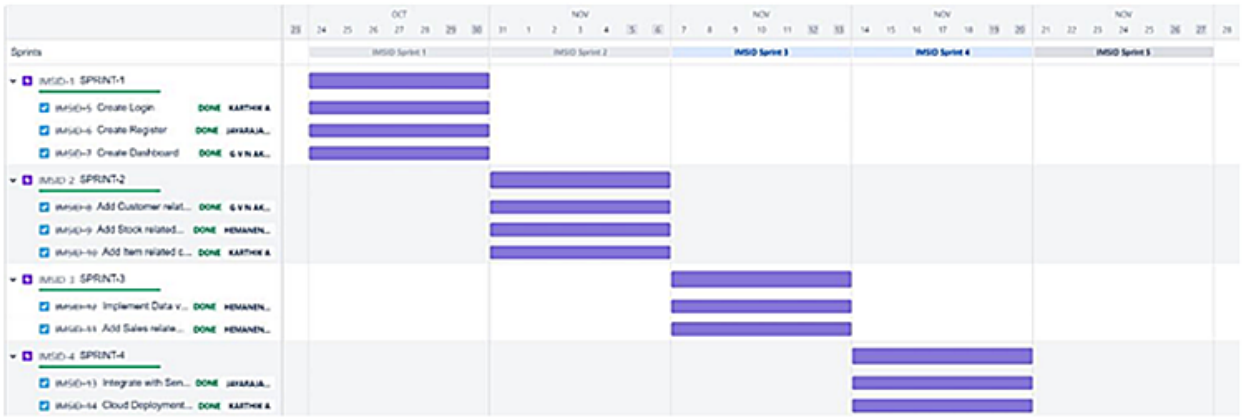
Sprint 4	Communication system	USN - 8	User and hub mangers can get the details of the stock moment via mail or chat bot .	20	Medium	1,2,3,4,5

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date	Story Points Completed (as on Planned End Date)	Sprint Release Date(Actual)
Sprint-1	7	7 Days	24 Oct 2022	30 Oct 2022	7	● (Meet Planned Date)
Sprint-2	9	7 Days	31 Oct 2022	06 Nov 2022	9	● (Meet Planned Date)
Sprint-3	5	7 Days	07 Nov 2022	13 Nov 2022	5	● (Meet Planned Date)
Sprint-4	5	7 Days	14 Nov 2022	20 Nov 2022	5	● (Meet Planned Date)

6.3 REPORTS FROM JIRA





7. CODING AND SOLUTIONING

7.1 FEATURE 1

Description

Retailer will be able to perform a sale by selecting the customer and adding items to it where the amount is calculated automatically and inventory is managed in real-time

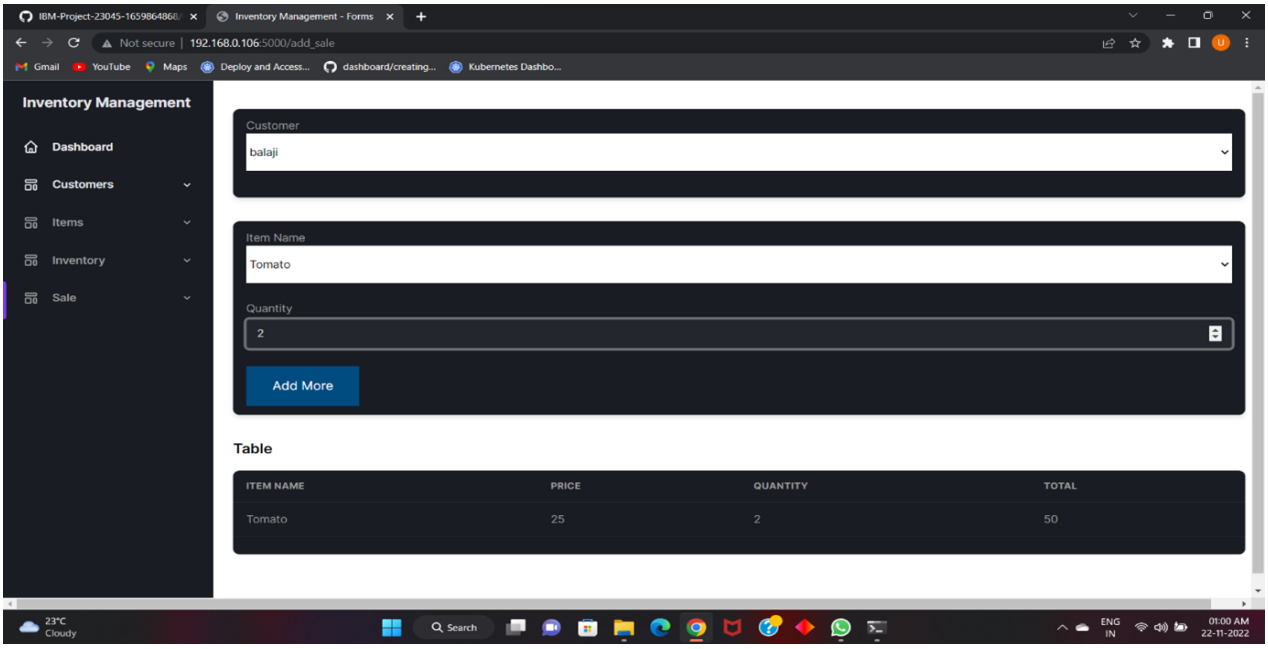
Source Code

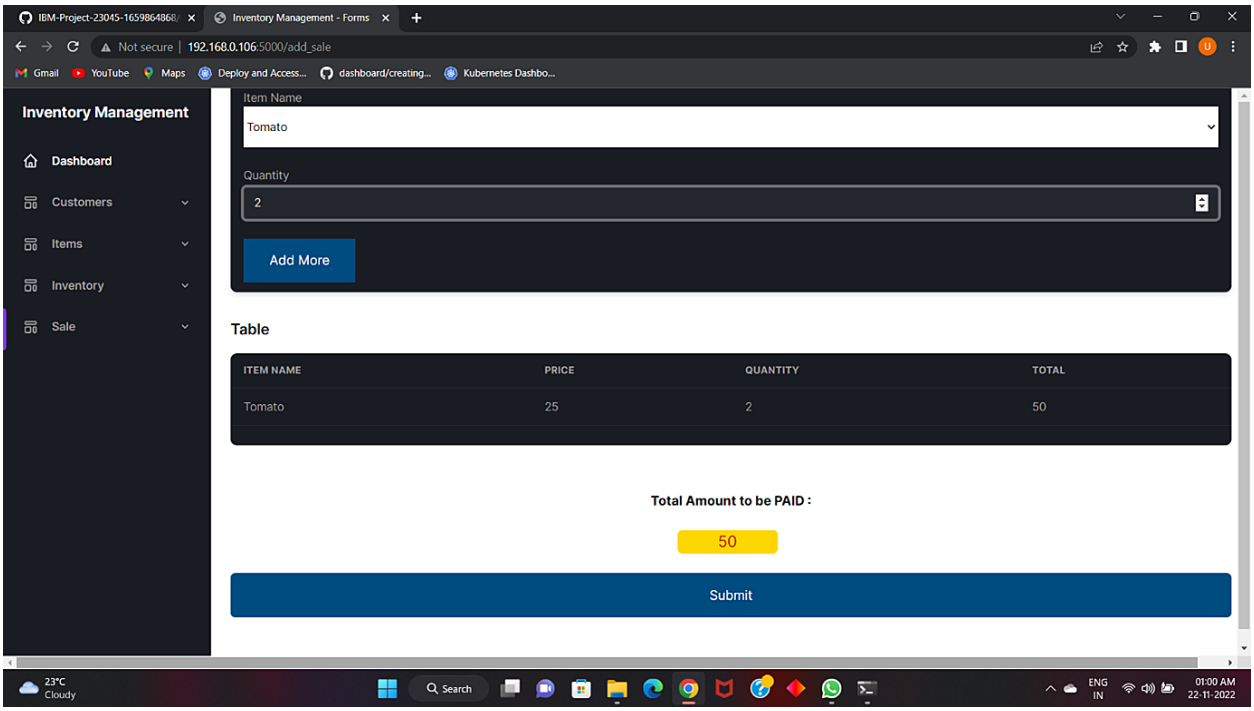
```
@app.route("/add_sale", methods=['GET','POST'])
def add_sale():
    # ITEMS
    items=list()
    query = "'select * from items"'
    exec_query = ibm_db.exec_immediate(conn, query)
    row = ibm_db.fetch_both(exec_query)
    while(row):
```

```
        items.append({"name": row["ITEM_NAME"], "quantity": row["LEFT_OUT"], "price": row["PRICE"]})
        row = ibm_db.fetch_both(exec_query)
    # CUSTOMERS
    custname=[]
    query = "'select * from customer'"
    exec_query = ibm_db.exec_immediate(conn, query)
    row = ibm_db.fetch_both(exec_query)
    while(row):
        custname.append(row["CUSTOMER_NAME"])
        row = ibm_db.fetch_both(exec_query)
    if request.method=="GET":
        return
render_template("Dashboard/add_sale.html",items=items,cname=custname,clen=len(custname),status="")
    elif request.method=="POST":
        i_array=request.form["item_array"]
        q_array=request.form["quantity_array"]
        item_list=i_array.split(",")
        quantity_list=q_array.split(",")
        cname=request.form["cname"]
        today = date.today()
        # FINDING CUSTOMER ID
        query = "'select customer_id from customer where customer_name = \'{}\'".format(cname)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        id=row["CUSTOMER_ID"]
        #SALE TABLE
        query = "'insert into sale(sale_date,customer_id) values('{}', '{}')'".format(today,id)
        exec_query = ibm_db.exec_immediate(conn, query)
        #GET SALE ID
        query = "'select sale_id from sale where sale_date = \'{}\' and customer_id=\'{}\'".format(today,id)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        sale_id=row["SALE_ID"]
        #SALE ITEM TABLE
        n=len(item_list)

        alert_mail(cname,item_list,quantity_list)

        for i in range(n):
```

```
        query = "'select item_id from items where item_name = \'{}\'".format(item_list[i])
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        item_id=row["ITEM_ID"]
        #UPDATION
        query = "'update items set left_out=left_out-\'{}\' where item_id=\'{}\'".format(quantity_list[i],item_id)
        exec_query = ibm_db.exec_immediate(conn, query)
        # INSERTION
        query = "'insert into sale_items(sale_id,quantity,item_id) values('{}', '{}',
'{}')'".format(sale_id,quantity_list[i],item_id)
        exec_query = ibm_db.exec_immediate(conn, query)
    return
render_template("Dashboard/add_sale.html",items=items,cname=custname,clen=len(custname),status="Sale
Success")
```

Screenshots





7.2 FEATURE 2

Description

Retailers could see the history of items sold in a specific period along with the quantity as a graph, which could be used to order items accordingly

Source Code

1.app.py

```
@app.route("/view_sale", methods=['GET','POST'])
def view_sale():
   items=list()
   if request.method=="GET":
      return render_template("Dashboard/view_sale.html",items=items)
   elif request.method=="POST":
      start=request.form["start_date"]
      end=request.form["end_date"]
      query = "'select item_id,count(quantity) from sale_items where sale_id in (select sale_id from sale where sale_date<= \'{}\' and sale_date>=\'{}\') group by(item_id)'".format(end,start)
      exec_query = ibm_db.exec_immediate(conn, query)
      row = ibm_db.fetch_both(exec_query)
```

```
        while(row):
            query = "'select item_name,price from items where item_id=\'{}\'".format(row[0])
            exec_query1 = ibm_db.exec_immediate(conn, query)
            row1 = ibm_db.fetch_both(exec_query1)

items.append({"item_id":row[0],"item_name":row1[0],"quantity":row[1],"amount":(row1[1]*row[1])})
            row = ibm_db.fetch_both(exec_query)
        # Item Name and Price
        return render_template("Dashboard/view_sale.html",items=items)
```

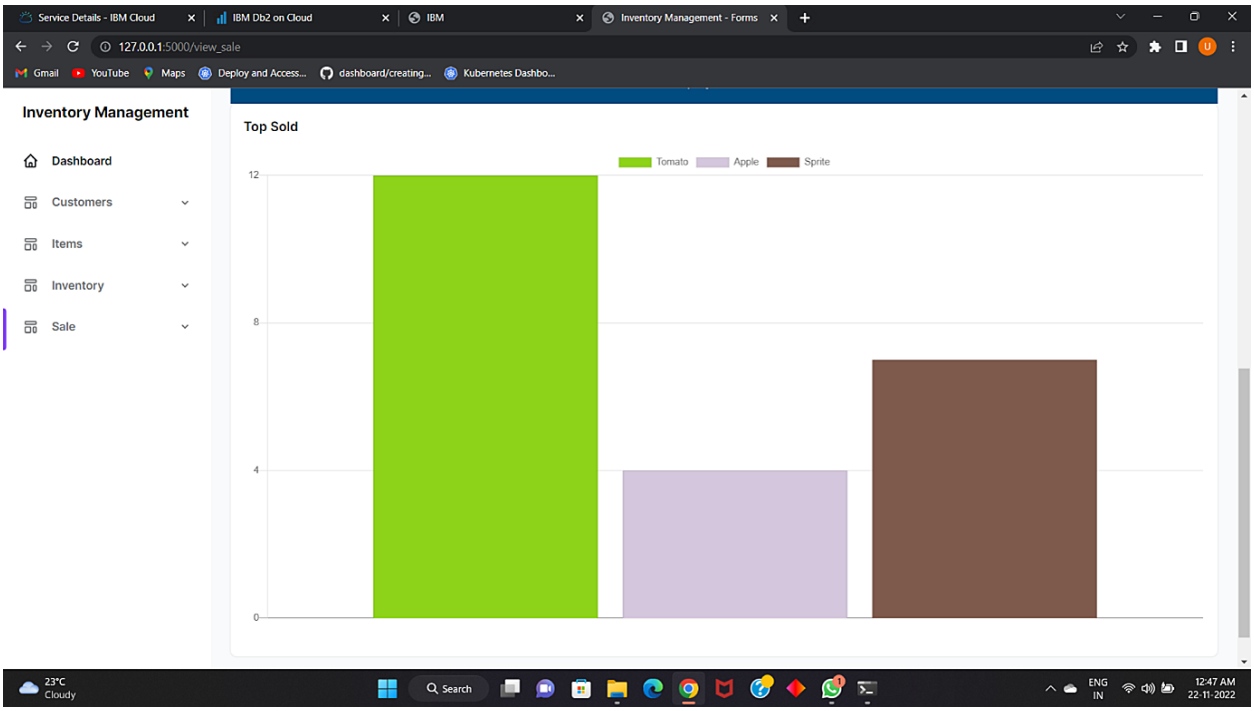
2.view_sale.html

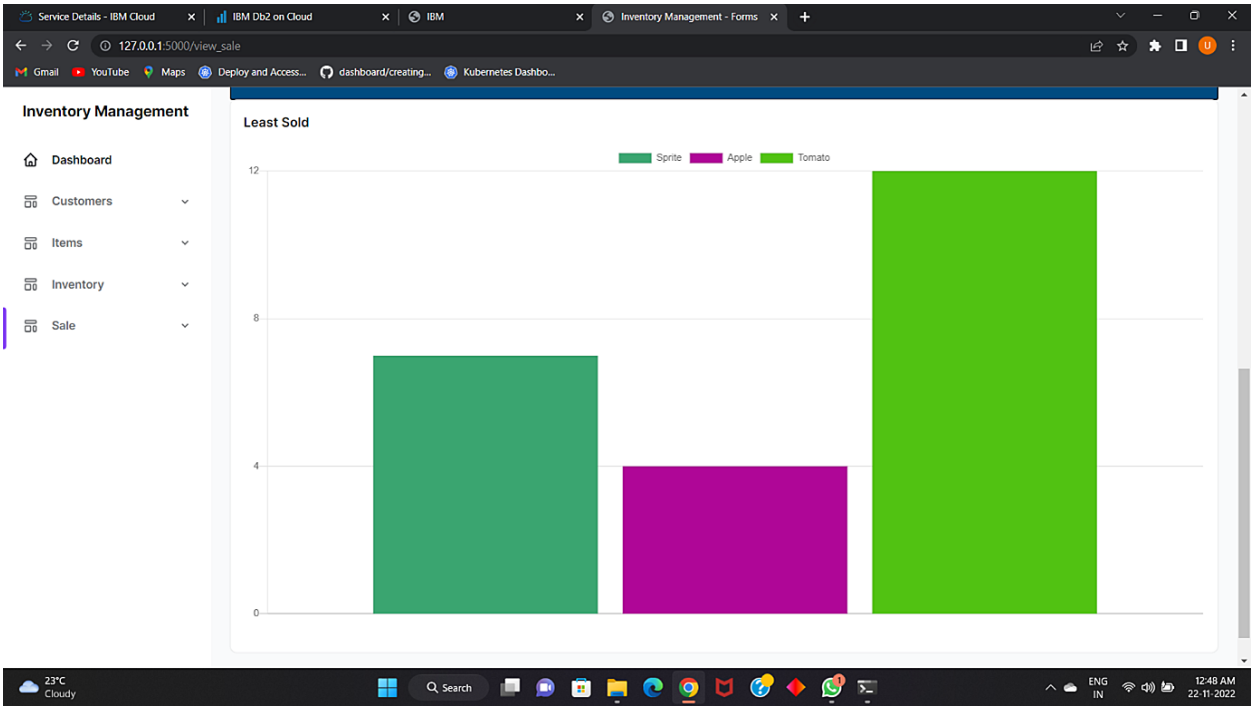
```
<script>
    x=0
    function myFunction(){
      if(x==0){
        x++;
        return
      }
      var st=document.getElementById("start").value;
      var en=document.getElementById("end").value;
      console.log(st)
      if(st>=en){
        document.getElementById("but").style.display="none";
        document.getElementById("temp").innerHTML="Start date should be before end date"
      }
      else{
        document.getElementById("temp").innerHTML=""
        document.getElementById("but").style.display="block";
      }
    }

  </script>
  <script>

    function getRandomColor() {
      var letters = '0123456789ABCDEF';
      var color = '#';
      for (var i = 0; i < 6; i++) {
```

```
    color += letters[Math.floor(Math.random() * 16)];
  }
  return color;
}




function grph(items,flag){
  console.log(items)
  var data=[]
  var label=[]
  var dataset=[]
  var n;
  var barsCtx ;
  if(flag==1){
    barsCtx = document.getElementById('bars')
    document.getElementById("last1").style.display="none"
    document.getElementById("top1").style.display="block"
    n=Math.min(10,items.length);
    for(var i=0;i<n;i++){
      data[i]=items[i]["quantity"]
      label[i]=items[i]["item_name"]
      var temp=[]
      temp[0]=data[i]
      dataset[i]={label:label[i],backgroundColor:getRandomColor(),borderWidth: 1,data: temp}
    }

  }
  else if(flag==0){
    barsCtx = document.getElementById('bars1');
    document.getElementById("top1").style.display="none"
    document.getElementById("last1").style.display="block"
    var j=0;
    n=Math.max(items.length-10,0)
    for(var i=items.length-1;i>=n;i--){
      data[j]=items[i]["quantity"]
      label[j]=items[i]["item_name"]
      var temp=[]
```

```
      temp[0]=data[j]
      dataset[j]={label:label[j],backgroundColor: getRandomColor(),borderWidth: 1,data: temp}
      j=j+1;
    }
  }
  var ind = 0,maxi = items[0]["quantity"];
  for(var i=0;i<items.length;i++){
    if(items[i]["quantity"]>maxi){
      maxi = items[i]["quantity"]
      ind = i
    }
  }
    const barConfig = {
    type: 'bar',
    data: {
      datasets: dataset,
    },
    options: {
      scales: {
        yAxes: [{
          ticks: {
            beginAtZero: true,
            min: 0,
            max: items[ind]["quantity"],
            stepSize: items[ind]["quantity"]/items.length,
          }
        }]
      }
    },
  }
  window.myBar = new Chart(barsCtx, barConfig)
 }
</script>
```

Screenshots



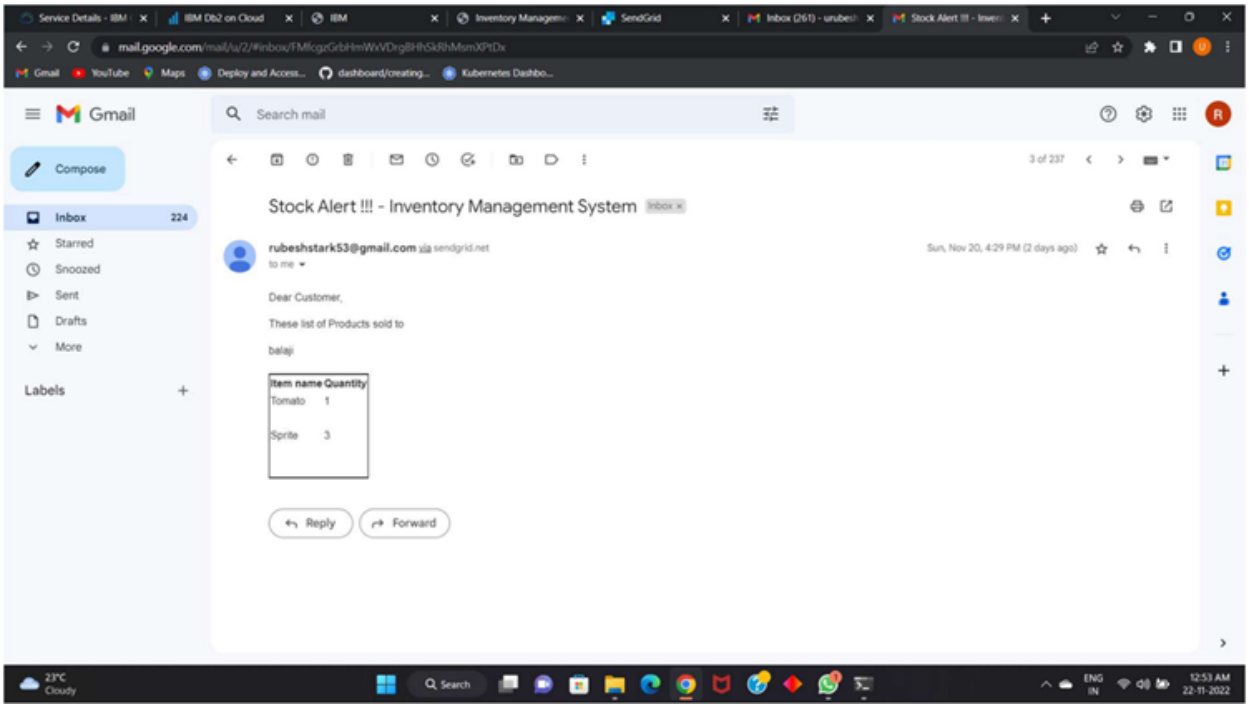


7.3 FEATURE 3

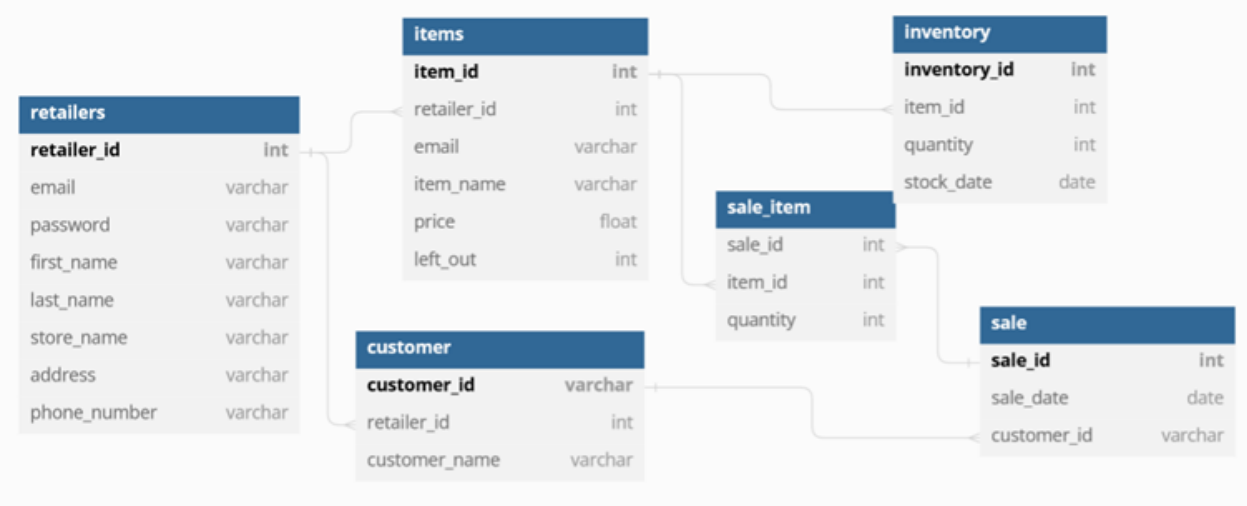
Description

After every sale, whenever an item goes below the specified threshold, the retailer would receive an email alert for the shortage of stocks

Screenshots



7.4 DATABASE SCHEMA



8. TESTING

8.1 TEST CASES

1.Sprint 1

Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Status
Functional	Login	Verify whether User is able to login into account	User should know his credentials	1. Enter email and password 2. Click Login button	Pass
Functional	Login	Verify whether a notification is displayed if the credentials are invalid	User should know his credentials	1. Enter email and password 2. Click Login button	Pass
UI	Login	Verify whether the submit button is activated only when all the fields with desired type are entered			Pass
Functional	Signup	Verify whether User is able to signup	User should have unique credentials	1. Enter name, storename, mobile number, email and password 2. Click Signup button	Pass

Functional	Signup	Verify whether a notification is displayed if the email is already used	User should have unique credentials	1. Enter name, storename, mobile number, email and password 2. Click Signup button	Pass
UI	Signup	Verify whether the submit button is activated only when all the fields with desired type are entered			Pass

Sprint 2

Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Status
Functional	Customer	Verify retailer is able to add a customer	Customer Id and Name	Enter customer id and name and click on add	Pass
Functional	Customer	Verify whether the retailer gets to know that the customer details already exists	Customer Id and Name	Enter customer id and name and click on add	Pass

UI	Customer	Verify whether the submit button gets activated only when all the required fields are entered		1. Enter the customer id and dont enter the name 2. Click submit	Pass
Functional	Customer	View the list of Customers		Click on view customers from the dashboard	Pass
Functional	Item	Verify whether the retailer is able to add a new item to the store along with the price	Item name and price	1. Enter the item name with some unique identifier 2. Click submit	Pass
Functional	Item	View the list of Items		Click on view items from the dashboard	Pass
UI	Item	Verify whether the submit button gets activated only when all the required fields are entered		1. Enter the item name and dont enter the price 2. Click submit	Pass
UI	Item	Verify whether the price field accepts only integer/floating point numbers		1. Enter alphabetic characters in the price field 2. Click submit	Pass

3.Sprint 3

Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Status
UI	Sale	Verify whether the user is able to add multiple items from the drop-down and enter its quantity to the sale		1. Select an item 2. Enter its quantity 3. Repeat the process again	Pass
Functional	Sale	Verify whether the unit price is displayed for an item and its total price is also calculated automatically		1. Select an item 2. Enter its quantity	Pass
Functional	Sale	Verify whether the user is notified when the quantity entered is above the stock available		1. Select an item 2. Enter its quantity	Pass

Functional	Sale	User should be able to see the history of date's on which an item has been sold along with the quantity		1. Select an item 2. Select from and to date	Pass
Functional	Historical Analysis	Verify whether the user is able to see the item name and its total quantity sold in the stipulated range		Select from and to date	Pass
UI	Historical Analysis	Verify whether a notification is displayed when the from date is greater than the to date		Select from and to date	Pass
UI	Historical Analysis	Verify whether a graph is displayed for the top 5 items that were sold the most in the specified range			Pass

UI	Historical Analysis	Verify whether a graph is displayed for the top 5 items that were sold the least in the specified range			Pass

4.Sprint 4

Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Status
Functional	Stock Shortage	Verify Whether the retailer receivesw an email when some items fall shortage in quantity	Send Grid	Perform a sale	Pass

8.2 USER ACCEPTANCE TESTING

Sprint 1

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	2	3	2	1	8
Duplicate	1	0	3	0	4
External	1	1	0	0	2
Fixed	0	2	1	1	4
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	4	6	6	2	18

Table : Defect Analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	0	0	0	0
Client Application	5	0	0	5
Security	4	0	0	4
Outsource Shipping	0	0	0	0
Exception Reporting	3	0	0	3
Final Report Output	0	0	0	0
Version Control	0	0	0	0

Table : Test Case Analysis

Sprint 2

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	3	4	1	2	10
Duplicate	1	1	2	0	4
External	2	1	0	0	3
Fixed	0	2	1	1	4
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	6	6	4	3	21

Table : Defect Analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	0	0	0	0
Client Application	6	0	0	6
Security	5	0	0	5
Outsource Shipping	0	0	0	0
Exception Reporting	4	0	0	4
Final Report Output	0	0	0	0
Version Control	0	0	0	0

Table : Test Case Analysis

Sprint 3

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	3	3	2	13
Duplicate	1	0	3	0	4
External	2	0	2	0	4
Fixed	0	3	3	2	8
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	8	6	11	4	29

Table : Defect Analysis

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	0	0	0	0
Client Application	7	0	0	7
Security	6	0	0	6
Outsource Shipping	0	0	0	0
Exception Reporting	5	0	0	5
Final Report Output	0	0	0	0
Version Control	0	0	0	0

Table : Test Case Analysis

Sprint 4

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	2	3	2	1	8
Duplicate	1	1	3	0	5
External	1	1	0	0	2
Fixed	0	2	4	1	4
Not Reproduced	0	0	1	0	0
Skipped	0	0	1	0	0
Won't Fix	0	0	2	0	0
Totals	4	6	6	2	19

Table : Defect Analysis

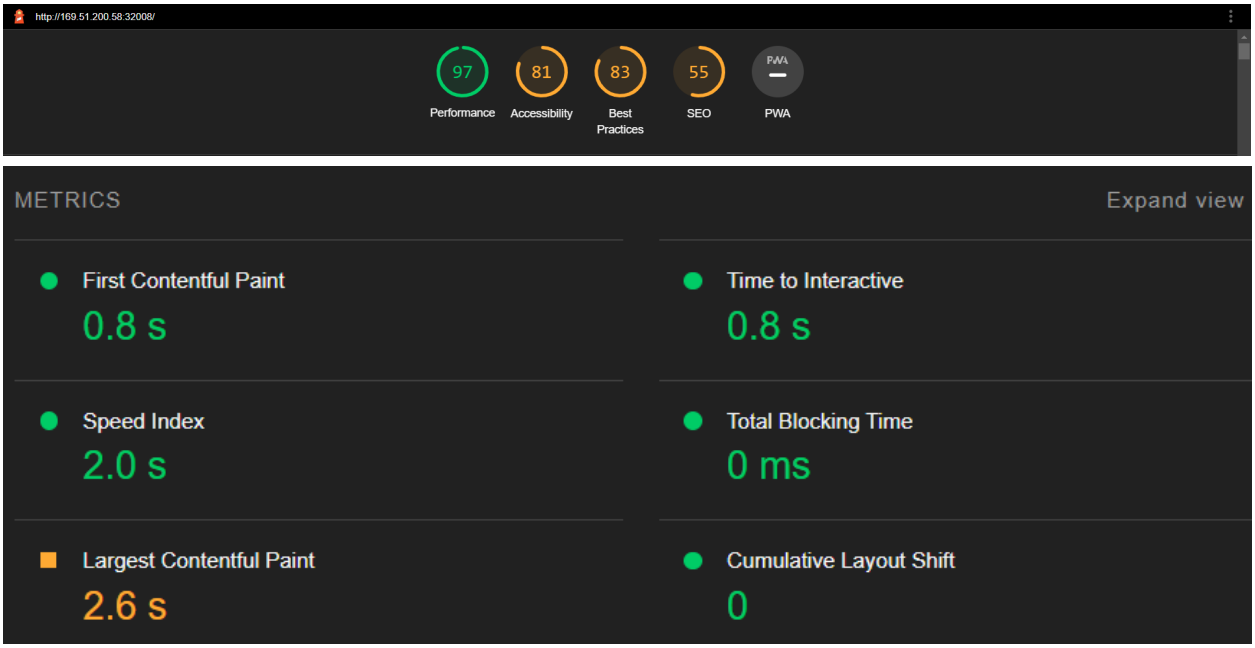
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	0	0	0	0
Client Application	4	0	0	4
Security	3	0	0	3
Outsource Shipping	0	0	0	0
Exception Reporting	4	0	0	4
Final Report Output	0	0	0	0
Version Control	0	0	0	0

Table : Test Case Analysis

9. RESULTS

9.1 PERFORMANCE METRICES

Light House Performance



10. ADVANTAGES AND DISADVANTAGES

ADVANTAGES

1. Retailers can use this application directly for their daily use where they need not depend on a developer to develop an application to manage their inventory
2. Stock alert via email could be made use to refill the items instantly and need not wait until the last moment to know it
3. Retailers can make use of the visualizations to order the items based on seasonal demands
4. Being a Web-Application and the data being stored in cloud, all the information could be accessed from any part of the world
5. No infrastructure needs to be maintained on-premises to deploy the application as everything is taken care in cloud

DISADVANTAGES

1. Could not be accessed using handheld devices
2. Single Sign-On feature is not available. User has to enter the login credentials each time and it should be difficult to remember multiple passwords
3. As the retail shops generally used to come across multiple customers on daily-basis, the data should be consistent even if the system suffers a crash

11. CONCLUSION

A Web based Application for managing the Inventory was developed to ease the work of retailers by maintaining a proper track on the items available, perform a sale, view the sales pattern and receive stock alert when items fall shortage in quantity.

12. FUTURE SCOPE

1. Single sign-on feature could be integrated into the application
2. APIs could be developed to make the system work on hand-held devices as well by designing an UI for mobile app
3. Customer specific discount feature could be added when a customer visits the store very frequently
4. Data analytics could be used to analyze the sales pattern in a specific time period

13. APPENDIX

13.1 Source Code

Directory Structure

- main.py
- static
  - images
    - bg1.jpg
    - bg2.jpg
  - js
    - charts-bars.js
    - charts-lines.js
    - charts-pie.js
    - focus-trap.js
    - init-alphine.js
  - styles
    - style.css
    - tailwind.css
    - tailwind_output.css
  - templates
    - Dashboard

- add_customer.html
- view_customer.html
- add_item.html
- view_item.html
- add_inventory.html
- view_inventory.html
- add_sale.html
- view_sale.html
- index.html

- Login
  - signup.html
  - index.html

main.py

```
from flask import *
from datetime import date
import ibm_db
from sendgrid import *
app=Flask(__name__)
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;SECURITY=SSL;PORT=31498;PROTOC
OL=TCPIP;UID=nnm68033;PWD=DUTMGiDWgJy5zlS8","")
# conn=None
retailer_id=0


cusmail = ""

@app.route("/", methods=['GET','POST'])
def login():
    if request.method=='GET':
        return render_template("Login/index.html",status="",colour="red")
    elif request.method=='POST':
        global retailer_id
        global cusmail
```

```
        email=request.form["email"]
        cusmail = request.form["email"]
        password=request.form["password"]
        query = "'select * from retailers where email = \'{}\'".format(email)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        if(row is not False):
            if(row['PASSWORD'] != password):
                return render_template("Login/index.html",status="Invalid Password",colour="red")
            else:
                temp="'select RETAILER_ID from retailers where email = \'{}\'".format(email)
                exec_query = ibm_db.exec_immediate(conn, temp)
                dict= ibm_db.fetch_both(exec_query)
                retailer_id=dict["RETAILER_ID"]
                return render_template("Dashboard/index.html")

        return render_template("Login/index.html",status="Invalid Email",colour="red")

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

    if request.method=='GET':
        return render_template("Login/signup.html",status="",colour="red")
    elif request.method=='POST':
        global cusmail
        cusmail = request.form["email"]
        email=request.form["email"]
        password=request.form["password"]
        first_name=request.form["first_name"]
        last_name=request.form["last_name"]
        store_name=request.form["store_name"]
        address=request.form["address"]
        phone_number=request.form["phone_number"]
        query = "'select * from retailers where email = \'{}\'".format(email)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        if(row is False):
            query = "'insert into retailers(email, password, first_name, last_name, store_name, address,
phone_number) values('{}', '{}', '{}', '{}', '{}', '{}', '{}')'".format(email, password, first_name, last_name,
```

```
store_name, address, phone_number)
            exec_query = ibm_db.exec_immediate(conn, query)
            return render_template("Login/signup.html",status="Signup Success",colour="green")
        else:
            return render_template("Login/signup.html",status="User Already Exists",colour="red")

@app.route("/dashboard", methods=['GET','POST'])
def dashboard():
    if request.method=="GET":
        return render_template("Dashboard/index.html")


@app.route("/add_customer", methods=['GET','POST'])
def add_customer():
    if request.method=="GET":
        return render_template("Dashboard/add_customer.html")
    elif request.method=="POST":
        name=request.form["name"]
        id=int(request.form["id"])
        query = "'select * from customer where customer_id = \'{}\'".format(id)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        if(row is False):
            query = "'insert into customer(customer_id,retailer_id,customer_name) values('{}', '{}',
'{}')'".format(id,retailer_id,name)
            exec_query = ibm_db.exec_immediate(conn, query)
            return render_template("Dashboard/add_customer.html",status="Customer Added",colour="green")
        else:
            return render_template("Dashboard/add_customer.html",status="Customer Already
Exists",colour="red")

@app.route("/view_customer", methods=['GET','POST'])
def view_customer():
    if request.method=="GET":
        query = "'select * from customer'"
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        name=[]
        id=[]
```

```
        while(row):
            name.append(row["CUSTOMER_NAME"])
            id.append(row["CUSTOMER_ID"])
            row = ibm_db.fetch_both(exec_query)
        return render_template("Dashboard/view_customer.html",name=name,id=id,len=len(name))

@app.route("/add_item", methods=['GET','POST'])
def add_item():
    if request.method=="GET":
        return render_template("Dashboard/add_item.html")
    elif request.method=="POST":
        name=request.form["name"]
        price=float(request.form["price"])
        query = "'select * from items where item_name = \'{}\'".format(name)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        if(row is False):
            query = "'insert into items(retailer_id,item_name,price,left_out) values('{}', '{}', '{}',
'{}')'".format(retailer_id,name,price,0)
            exec_query = ibm_db.exec_immediate(conn, query)
            return render_template("Dashboard/add_item.html",status="Item Added",colour="green")
        else:
            return render_template("Dashboard/add_item.html",status="Item Already Exists",colour="red")

@app.route("/view_item", methods=['GET','POST'])
def view_item():
    if request.method=="GET":
        query = "'select * from items'"
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        name=[]
        id=[]
        price=[]
        left_out=[]
        while(row):
            name.append(row["ITEM_NAME"])
            id.append(row["ITEM_ID"])
            price.append(row["PRICE"])
            left_out.append(row["LEFT_OUT"])
```

```
        row = ibm_db.fetch_both(exec_query)
    return
render_template("Dashboard/view_item.html",name=name,id=id,price=price,left_out=left_out,len=len(name))

@app.route("/add_inventory", methods=['GET','POST'])
def add_inventory():
    name=[]
    query = "'select * from items"'
    exec_query = ibm_db.exec_immediate(conn, query)
    row = ibm_db.fetch_both(exec_query)
    while(row):
        name.append(row["ITEM_NAME"])
        row = ibm_db.fetch_both(exec_query)
    if request.method=="GET":
        return render_template("Dashboard/add_inventory.html",name=name,len=len(name),status=" ")
    elif request.method=="POST":
        fname=request.form["name"]
        quantity=request.form["quantity"]
        date=request.form["date"]

        # Finding ITEM ID
        query = "'select item_id from items where item_name = \'{}\'"'.format(fname)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        id=row["ITEM_ID"]
        # INSERTION
        query = "'insert into inventory(item_id,quantity,stock_date) values('{}', '{}', '{}')"'.format(id,quantity,date)
        exec_query = ibm_db.exec_immediate(conn, query)
        #UPDATION
        query = "'update items set left_out=left_out+\'{}\' where item_id=\'{}\'"'.format(quantity,id)
        exec_query = ibm_db.exec_immediate(conn, query)
        return render_template("Dashboard/add_inventory.html",name=name,len=len(name),status="Inventory
added")

@app.route("/view_inventory", methods=['GET','POST'])
def view_inventory():
    name=[]
    query = "'select * from items"'
    exec_query = ibm_db.exec_immediate(conn, query)
```

```
    row = ibm_db.fetch_both(exec_query)
    while(row):
        name.append(row["ITEM_NAME"])
        row = ibm_db.fetch_both(exec_query)
    items=list()
    if request.method=="GET":
        return render_template("Dashboard/view_inventory.html",name=name,items=items)
    elif request.method=="POST":
        item_name=request.form["name"]
        start=request.form["start_date"]
        end=request.form["end_date"]
        query = "'select item_id from items where item_name = \'{}\'".format(item_name)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        id=row["ITEM_ID"]
        query = "'select stock_date,quantity from inventory where item_id=\'{}\' and stock_date<=\'{}\' and
stock_date>=\'{}\'".format(id,end,start)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        while(row):
            items.append({"item_name":item_name,"quantity":row[1],"stock_date":row[0]})
            row = ibm_db.fetch_both(exec_query)
        return render_template("Dashboard/view_inventory.html",name=name,items=items)

@app.route("/add_sale", methods=['GET','POST'])
def add_sale():
    # ITEMS
    items=list()
    query = "'select * from items"'
    exec_query = ibm_db.exec_immediate(conn, query)
    row = ibm_db.fetch_both(exec_query)
    while(row):
        items.append({"name": row["ITEM_NAME"], "quantity": row["LEFT_OUT"], "price": row["PRICE"]})
        row = ibm_db.fetch_both(exec_query)
    # CUSTOMERS
    custname=[]
    query = "'select * from customer"'
    exec_query = ibm_db.exec_immediate(conn, query)
    row = ibm_db.fetch_both(exec_query)
```

```
    while(row):
        custname.append(row["CUSTOMER_NAME"])
        row = ibm_db.fetch_both(exec_query)
    if request.method=="GET":
        return
render_template("Dashboard/add_sale.html",items=items,cname=custname,clen=len(custname),status="")
    elif request.method=="POST":
        i_array=request.form["item_array"]
        q_array=request.form["quantity_array"]
        item_list=i_array.split(",")
        quantity_list=q_array.split(",")
        cname=request.form["cname"]
        today = date.today()
        # FINDING CUSTOMER ID
        query = "'select customer_id from customer where customer_name = \'{}\'".format(cname)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        id=row["CUSTOMER_ID"]
        #SALE TABLE
        query = "'insert into sale(sale_date,customer_id) values('{}', '{}')'".format(today,id)
        exec_query = ibm_db.exec_immediate(conn, query)
        #GET SALE ID
        query = "'select sale_id from sale where sale_date = \'{}\' and customer_id=\'{}\'".format(today,id)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        sale_id=row["SALE_ID"]
        #SALE ITEM TABLE
        n=len(item_list)

        alert_mail(cname,item_list,quantity_list)

        for i in range(n):
            query = "'select item_id from items where item_name = \'{}\'".format(item_list[i])
            exec_query = ibm_db.exec_immediate(conn, query)
            row = ibm_db.fetch_both(exec_query)
            item_id=row["ITEM_ID"]
            #UPDATION
            query = "'update items set left_out=left_out-\'{}\' where item_id=\'{}\'".format(quantity_list[i],item_id)
            exec_query = ibm_db.exec_immediate(conn, query)
```

```
            # INSERTION
            query = "'insert into sale_items(sale_id,quantity,item_id) values('{}', '{}',
'{}')'".format(sale_id,quantity_list[i],item_id)
            exec_query = ibm_db.exec_immediate(conn, query)
        return
render_template("Dashboard/add_sale.html",items=items,cname=custname,clen=len(custname),status="Sale
Success")


@app.route("/view_sale", methods=['GET','POST'])
def view_sale():
    items=list()
    if request.method=="GET":
        return render_template("Dashboard/view_sale.html",items=items)
    elif request.method=="POST":
        start=request.form["start_date"]
        end=request.form["end_date"]
        query = "'select item_id,count(quantity) from sale_items where sale_id in (select sale_id from sale where
sale_date<= \'{}\' and sale_date>=\'{}\') group by(item_id)'".format(end,start)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        while(row):
            query = "'select item_name,price from items where item_id=\'{}\'".format(row[0])
            exec_query1 = ibm_db.exec_immediate(conn, query)
            row1 = ibm_db.fetch_both(exec_query1)

items.append({"item_id":row[0],"item_name":row1[0],"quantity":row[1],"amount":(row1[1]*row[1])})
            row = ibm_db.fetch_both(exec_query)
        # Item Name and Price
        return render_template("Dashboard/view_sale.html",items=items)


# API = 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

def alert_mail(cname,items,quantity):
    print(cname,items,quantity)
    sg =
sendgrid.SendGridAPIClient(api_key='SG.Z6VI5DEWSH2tZ6Z6VI5DEWSH2tZ6VI5DEWSH2tZ6VI5DEW
SH2t')
```

```
    from_email = Email("rubeshstark53@gmail.com")
    global cusmail

    print(cusmail)
    to_email = To(cusmail)

    subject = "Stock Alert !!! - Inventory Management System"

    msg = '''
    <html>
    <body>
        <p>Dear Customer,</p>
        <p>These list of Products sold to </p>
    '''

    msg+=cname

    msg+="<br/><br/><table style='border: 1px solid black; border-collapse: collapse;'><tr><th>Item
name</th><th>Quantity</th></tr>"

    for i in range(len(items)):

        msg +="<tr><td>"+items[i]+"</td> <td>"+quantity[i]+"</td></tr><br>"
    '''

    </table>


    <body>

    </html>
    '''

    content = Content("text/html", msg)

    mail = Mail(from_email, to_email, subject, content)
    mail_json = mail.get()
    response = sg.client.mail.send.post(request_body=mail_json)
```

```
    print(response.status_code)
    print(response.headers)


if __name__=="__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

signup.html

```
<html>
<head>
    <link rel="stylesheet" href={{url_for('static', filename='styles/style.css')}}>
</head>
<body>
<br>
<br>
    <div class="cont">
        <i>
        <div class="form sign-in">
            <h2>Create your Account</h2>
            </br>
            <p style="color:{{colour}};text-align:center;">{{status}}</p>
                <form action="/signup" autocomplete="ON" method="POST">
                <label>
                    <span>Email</span>
                    <input type="email" name="email" required/>
                </label>
                <label>
                    <span>First Name</span>
                    <input type="text" name="first_name" required/>
                </label>
                <label>
                    <span>Last Name</span>
                    <input type="text" name="last_name" required/>
                </label>
                <label>
                    <span>Password</span>
                    <input type="password" name="password" required/>
```

```
            </label>
            <label>
                <span>Store Name</span>
                <input type="text" name="store_name" required/>
            </label>
            <label>
                <span>Address</span>
                <input type="text" name="address" required/>
            </label>
            <label>
                <span>Phone Number</span>
                <input type="text" name="phone_number" required/>
            </label>
            <button type="submit" class="submit">Sign Up</button>
            </form>

        </div>


        <div class="sub-cont">
            <div class="img">
                <div class="img__text m--up">

                    <h3>If you already has an account, just sign in.<h3>
                </div>
                <div class="img__btn">
                    <a href="/"><span>Login</span></a>
                </div>
            </div>
        </div>
        </i>
    </div>

</body>
</html>
```

add_sale.html

```
<!DOCTYPE html>
```

```
<html :class="{ 'theme-dark': dark }" x-data="data()" lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Inventory Management - Forms</title>
  <link
   href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700;800&display=swap"
   rel="stylesheet"
  />
  <link rel="stylesheet" href={{url_for('static', filename='styles/tailwind_output.css')}} />
  <script
   src="https://cdn.jsdelivr.net/gh/alpinejs/alpine@v2.x.x/dist/alpine.min.js"
   defer
  ></script>
  <script src="{{url_for('static', filename='js/init-alpine.js')}}"></script>
 </head>
 <body onload="init({{items}})">
  <div
   class="flex h-screen bg-gray-50 dark:bg-gray-900"
   :class="{ 'overflow-hidden': isSideMenuOpen}"
  >
   <!-- Desktop sidebar -->
   <aside
    class="z-20 hidden w-64 overflow-y-auto bg-white dark:bg-gray-800 md:block flex-shrink-0"
   >
    <div class="py-4 text-gray-500 dark:text-gray-400">
     <a
      class="ml-6 text-lg font-bold text-gray-800 dark:text-gray-200"
      href="#"
     >
      Inventory Management
     </a>
     <ul class="mt-6">
      <li class="relative px-6 py-3">
       <a
        class="inline-flex items-center w-full text-sm font-semibold text-gray-800 transition-colors duration-
150 hover:text-gray-800 dark:hover:text-gray-200 dark:text-gray-100"
        href="/dashboard"
       >
```

```
            <svg
              class="w-5 h-5"
              aria-hidden="true"
              fill="none"
              stroke-linecap="round"
              stroke-linejoin="round"
              stroke-width="2"
              viewBox="0 0 24 24"
              stroke="currentColor"
            >
              <path
                d="M3 12l2-2m0 0l7-7 7 7M5 10v10a1 1 0 001 1h3m10-11l2 2m-2-2v10a1 1 0 01-1 1h-3m-6 0a1 1 0 001-1v-4a1 1 0 011-1h2a1 1 0 011 1v4a1 1 0 001 1m-6 0h6"
              ></path>
            </svg>
            <span class="ml-4">Dashboard</span>
          </a>
        </li>
      </ul>
      <ul>
        <li class="relative px-6 py-3">
          <button
            class="inline-flex items-center justify-between w-full text-sm font-semibold transition-colors duration-150 hover:text-gray-800 dark:hover:text-gray-200"
            @click="togglePagesMenuCustomer"
            aria-haspopup="true"
          >
            <span class="inline-flex items-center">
              <svg
                class="w-5 h-5"
                aria-hidden="true"
                fill="none"
                stroke-linecap="round"
                stroke-linejoin="round"
                stroke-width="2"
                viewBox="0 0 24 24"
                stroke="currentColor"
              >
                <path
```

```
                d="M4 5a1 1 0 011-1h14a1 1 0 011 1v2a1 1 0 01-1 1H5a1 1 0 01-1-1V5zM4 13a1 1 0 011-1h6a1 1 0 011 1v6a1 1 0 01-1 1H5a1 1 0 01-1-1v-6zM16 13a1 1 0 011-1h2a1 1 0 011 1v6a1 1 0 01-1 1h-2a1 1 0 01-1-1v-6z"
              ></path>
            </svg>
            <span class="ml-4">Customers</span>
          </span>
          <svg
            class="w-4 h-4"
            aria-hidden="true"
            fill="currentColor"
            viewBox="0 0 20 20"
          >
            <path
              fill-rule="evenodd"
              d="M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0 010-1.414z"
              clip-rule="evenodd"
            ></path>
          </svg>
        </button>
        <template x-if="isPagesMenuOpenCustomer">
          <ul
            x-transition:enter="transition-all ease-in-out duration-300"
            x-transition:enter-start="opacity-25 max-h-0"
            x-transition:enter-end="opacity-100 max-h-xl"
            x-transition:leave="transition-all ease-in-out duration-300"
            x-transition:leave-start="opacity-100 max-h-xl"
            x-transition:leave-end="opacity-0 max-h-0"
            class="p-2 mt-2 space-y-2 overflow-hidden text-sm font-medium text-gray-500 rounded-md shadow-inner bg-gray-50 dark:text-gray-400 dark:bg-gray-900"
            aria-label="submenu"
          >
            <li
              class="px-2 py-1 transition-colors duration-150 hover:text-gray-800 dark:hover:text-gray-200"
            >
              <a class="w-full" href="/add_customer">Add customer</a>
            </li>
            <li
```

```
              class="px-2 py-1 transition-colors duration-150 hover:text-gray-800 dark:hover:text-gray-200"
            >
              <a class="w-full" href="/view_customer">View customer</a>
            </li>

            </ul>
          </template>
        </li>
        <li class="relative px-6 py-3">
          <button
            class="inline-flex items-center justify-between w-full text-sm font-semibold transition-colors
duration-150 hover:text-gray-800 dark:hover:text-gray-200"
            @click="togglePagesMenuItem"
            aria-haspopup="true"
          >
            <span class="inline-flex items-center">
              <svg
                class="w-5 h-5"
                aria-hidden="true"
                fill="none"
                stroke-linecap="round"
                stroke-linejoin="round"
                stroke-width="2"
                viewBox="0 0 24 24"
                stroke="currentColor"
              >
                <path
                  d="M4 5a1 1 0 011-1h14a1 1 0 011 1v2a1 1 0 01-1 1H5a1 1 0 01-1-1V5zM4 13a1 1 0 011-
1h6a1 1 0 011 1v6a1 1 0 01-1 1H5a1 1 0 01-1-1v-6zM16 13a1 1 0 011-1h2a1 1 0 011 1v6a1 1 0 01-1 1h-2a1 1
0 01-1-1v-6z"
                ></path>
              </svg>
              <span class="ml-4">Items</span>
            </span>
            <svg
              class="w-4 h-4"
              aria-hidden="true"
              fill="currentColor"
              viewBox="0 0 20 20"
```

```
            >
              <path
                fill-rule="evenodd"
                d="M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0 010-1.414z"
                clip-rule="evenodd"
              ></path>
            </svg>
          </button>
          <template x-if="isPagesMenuOpenItem">
            <ul
              x-transition:enter="transition-all ease-in-out duration-300"
              x-transition:enter-start="opacity-25 max-h-0"
              x-transition:enter-end="opacity-100 max-h-xl"
              x-transition:leave="transition-all ease-in-out duration-300"
              x-transition:leave-start="opacity-100 max-h-xl"
              x-transition:leave-end="opacity-0 max-h-0"
              class="p-2 mt-2 space-y-2 overflow-hidden text-sm font-medium text-gray-500 rounded-md shadow-inner bg-gray-50 dark:text-gray-400 dark:bg-gray-900"
              aria-label="submenu"
            >
              <li
                class="px-2 py-1 transition-colors duration-150 hover:text-gray-800 dark:hover:text-gray-200"
              >
                <a class="w-full" href="/add_item">Add Item</a>
              </li>
              <li
              class="px-2 py-1 transition-colors duration-150 hover:text-gray-800 dark:hover:text-gray-200"
            >
              <a class="w-full" href="/view_item">View Item</a>
            </li>

            </ul>
          </template>
        </li>
        <li class="relative px-6 py-3">
          <button
            class="inline-flex items-center justify-between w-full text-sm font-semibold transition-colors duration-150 hover:text-gray-800 dark:hover:text-gray-200"
```

```
            @click="togglePagesMenuInventory"
            aria-haspopup="true"
          >
            <span class="inline-flex items-center">
              <svg
                class="w-5 h-5"
                aria-hidden="true"
                fill="none"
                stroke-linecap="round"
                stroke-linejoin="round"
                stroke-width="2"
                viewBox="0 0 24 24"
                stroke="currentColor"
              >
                <path
                  d="M4 5a1 1 0 011-1h14a1 1 0 011 1v2a1 1 0 01-1 1H5a1 1 0 01-1-1V5zM4 13a1 1 0 011-1h6a1 1 0 011 1v6a1 1 0 01-1 1H5a1 1 0 01-1-1v-6zM16 13a1 1 0 011-1h2a1 1 0 011 1v6a1 1 0 01-1 1h-2a1 1 0 01-1-1v-6z"
                ></path>
              </svg>
              <span class="ml-4">Inventory</span>
            </span>
            <svg
              class="w-4 h-4"
              aria-hidden="true"
              fill="currentColor"
              viewBox="0 0 20 20"
            >
              <path
                fill-rule="evenodd"
                d="M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0 010-1.414z"
                clip-rule="evenodd"
              ></path>
            </svg>
          </button>
          <template x-if="isPagesMenuOpenInventory">
            <ul
              x-transition:enter="transition-all ease-in-out duration-300"
```

```
            x-transition:enter-start="opacity-25 max-h-0"
            x-transition:enter-end="opacity-100 max-h-xl"
            x-transition:leave="transition-all ease-in-out duration-300"
            x-transition:leave-start="opacity-100 max-h-xl"
            x-transition:leave-end="opacity-0 max-h-0"
            class="p-2 mt-2 space-y-2 overflow-hidden text-sm font-medium text-gray-500 rounded-md
shadow-inner bg-gray-50 dark:text-gray-400 dark:bg-gray-900"
            aria-label="submenu"
          >
            <li
              class="px-2 py-1 transition-colors duration-150 hover:text-gray-800 dark:hover:text-gray-200"
            >
              <a class="w-full" href="/add_inventory">Add Inventory</a>
            </li>
            <li
            class="px-2 py-1 transition-colors duration-150 hover:text-gray-800 dark:hover:text-gray-200"
          >
            <a class="w-full" href="/view_inventory">View Inventory</a>
          </li>
          </ul>
        </template>
      </li>
      <li class="relative px-6 py-3">
        <span
          class="absolute inset-y-0 left-0 w-1 bg-purple-600 rounded-tr-lg rounded-br-lg"
          aria-hidden="true"
        ></span>
        <button
          class="inline-flex items-center justify-between w-full text-sm font-semibold transition-colors
duration-150 hover:text-gray-800 dark:hover:text-gray-200"
          @click="togglePagesMenuSale"
          aria-haspopup="true"
        >
          <span class="inline-flex items-center">
            <svg
              class="w-5 h-5"
              aria-hidden="true"
              fill="none"
              stroke-linecap="round"
```

```
                stroke-linejoin="round"
                stroke-width="2"
                viewBox="0 0 24 24"
                stroke="currentColor"
              >
                <path
                  d="M4 5a1 1 0 011-1h14a1 1 0 011 1v2a1 1 0 01-1 1H5a1 1 0 01-1-1V5zM4 13a1 1 0 011-1h6a1 1 0 011 1v6a1 1 0 01-1 1H5a1 1 0 01-1-1v-6zM16 13a1 1 0 011-1h2a1 1 0 011 1v6a1 1 0 01-1 1h-2a1 1 0 01-1-1v-6z"
                ></path>
              </svg>
              <span class="ml-4">Sale</span>
            </span>
            <svg
              class="w-4 h-4"
              aria-hidden="true"
              fill="currentColor"
              viewBox="0 0 20 20"
            >
              <path
                fill-rule="evenodd"
                d="M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0 010-1.414z"
                clip-rule="evenodd"
              ></path>
            </svg>
          </button>
          <template x-if="isPagesMenuOpenSale">
            <ul
              x-transition:enter="transition-all ease-in-out duration-300"
              x-transition:enter-start="opacity-25 max-h-0"
              x-transition:enter-end="opacity-100 max-h-xl"
              x-transition:leave="transition-all ease-in-out duration-300"
              x-transition:leave-start="opacity-100 max-h-xl"
              x-transition:leave-end="opacity-0 max-h-0"
              class="p-2 mt-2 space-y-2 overflow-hidden text-sm font-medium text-gray-500 rounded-md shadow-inner bg-gray-50 dark:text-gray-400 dark:bg-gray-900"
              aria-label="submenu"
            >
```

```
            <li
              class="px-2 py-1 transition-colors duration-150 hover:text-gray-800 dark:hover:text-gray-200"
            >
              <a class="w-full" href="/add_sale">Add Sale</a>
            </li>
            <li
            class="px-2 py-1 transition-colors duration-150 hover:text-gray-800 dark:hover:text-gray-200"
          >
            <a class="w-full" href="/view_sale">View Sale</a>
          </li>
          </ul>
        </template>
      </li>

    </ul>
  </div>
</aside>
<!-- Mobile sidebar -->
<!-- Backdrop -->
<div
  x-show="isSideMenuOpen"
  x-transition:enter="transition ease-in-out duration-150"
  x-transition:enter-start="opacity-0"
  x-transition:enter-end="opacity-100"
  x-transition:leave="transition ease-in-out duration-150"
  x-transition:leave-start="opacity-100"
  x-transition:leave-end="opacity-0"
  class="fixed inset-0 z-10 flex items-end bg-black bg-opacity-50 sm:items-center sm:justify-center"
></div>
<div class="flex flex-col flex-1">
  <main class="h-full pb-16 overflow-y-auto" style="background-color:white;">
    <div class="container px-6 mx-auto grid">
      <h2
        class="my-6 text-2xl font-semibold text-gray-700 dark:text-gray-200"
        style="color:black;" >
        Add Sale
      </h2>
```

```
        <!-- General elements -->
        <h4
          class="mb-4 text-lg font-semibold text-gray-600 dark:text-gray-300"
        >

        </h4>
        <p style="color:green;text-align:center;">{{status}}</p>
        <form action="/add_sale" method="post" autocomplete="on">
        <div
          class="px-4 py-3 mb-8 bg-white rounded-lg shadow-md dark:bg-gray-800"
        >
        <label class="block text-sm">
            <span class="text-gray-700 dark:text-gray-400">Customer </span>
            <select style="width: 1200px;height: 50px;" name="cname">
              {%for i in range(clen)%}
              <option style="min-height: 50px;" value="{{cname[i]}}">{{cname[i]}}</option>
              {%endfor%}
            </select>
          </label>
        </br>
        </div>
        <div
          class="px-4 py-3 mb-8 bg-white rounded-lg shadow-md dark:bg-gray-800"
        >
        <p style="color:red;text-align:center;" id="quantStatus"></p>
        <label class="block text-sm">
            <span class="text-gray-700 dark:text-gray-400">Item Name</span>
        <select style="width: 1200px;height: 50px;" name="iname" id="iname">
          {%for i in items%}
          <option style="min-height: 50px;" value='{{i["name"]}}'>{{i["name"]}}</option>
          {%endfor%}
        </select>
          </label>
        </br>
          <label class="block text-sm">
            <span class="text-gray-700 dark:text-gray-400">Quantity</span>
            <input
              class="block w-full mt-1 text-sm dark:border-gray-600 dark:bg-gray-700 focus:border-purple-400
focus:outline-none focus:shadow-outline-purple dark:text-gray-300 dark:focus:shadow-outline-gray form-
```

```
input"
              name="fname" type="number" name="quantity" id="quantity"
             />
            </label>

          </br>
            <button type="button" onclick="myFunction();" style="background-color: rgb(0, 75,
128);color:white;padding: 15px 32px;">Add More</button>
          </div>
          <h4
          class="mb-4 text-lg font-semibold text-gray-600 dark:text-gray-300"
          style="color:black;" >
          Table
        </h4>
        <div class="w-full mb-8 overflow-hidden rounded-lg shadow-xs">
          <div class="w-full overflow-x-auto">
            <table class="w-full whitespace-no-wrap" id="myTable">
              <thead>
                <tr
                  class="text-xs font-semibold tracking-wide text-left text-gray-500 uppercase border-b dark:border-
gray-700 bg-gray-50 dark:text-gray-400 dark:bg-gray-800"
                >
                  <th class="px-4 py-3">Item Name</th>
                  <th class="px-4 py-3">Price</th>
                  <th class="px-4 py-3">Quantity</th>
                  <th class="px-4 py-3">Total</th>
                </tr>
              </thead>
              <tbody
                class="bg-white divide-y dark:divide-gray-700 dark:bg-gray-800"
              >
                    <tr class="text-gray-700 dark:text-gray-400">
                      <td class="px-4 py-3 text-sm">
                      </td>
                      <td class="px-4 py-3 text-sm">
                      </td>
                      <td class="px-4 py-3 text-sm">
                      </td>
                      <td class="px-4 py-3 text-sm">
```

```
                    </td>
                  </tr>
              </tbody>
            </table>
          </div>
        </div>
      </br>
        <p style="display:none">
          <input type="text" type="hidden" name="item_array" value="" id="item_array"/>
          <input type="text" type="hidden" name="quantity_array" value="" id="quantity_array"/>
        </p>
        <p style="text-align:center;font-weight: bold;">Total Amount to be PAID :</p>
        </br>
        <p style="color:rgb(192, 20, 20);text-align:center;background-color: gold;width:10%;border-radius:
6px;margin-left: 550px;font-size: larger;"  id="total">0</p>
      </br>
        <button type="submit" style="border-radius:6px;background-color: rgb(0, 75, 128);color:white;padding:
15px 32px;width: 100%;display: none;" id="submitBut">Submit</button>
    </form>
      </main>
    </div>
  </div>
  <script>
    var y=0,tot=0,n=0;
    var it,q,price;
    var item=[],quantity=[];
    var items={},sale={};
    function init(item){
      for(var i=0;i<item.length;i++){
        var list=[]
        list[1]=item[i]["quantity"]
        list[0]=item[i]["price"]
        items[item[i]["name"]]=list
      }
    }
    function myFunction() {
        it=document.getElementById("iname").value;
        q=document.getElementById("quantity").value;
        if(parseInt(items[it][1])<q){
```

```
        document.getElementById("quantStatus").innerHTML="Stock left for "+it+" is only "+items[it][1];
      }
      else if(q==0){
          document.getElementById("quantStatus").innerHTML="Quantity cannot be Zero";
      }
      else{
        items[it][1]-=q;
        price=items[it][0];
        if(it && q){
            finalCall();
        }
        item[y]=it
        quantity[y]=q
        document.getElementById("item_array").value=item;
        document.getElementById("quantity_array").value=quantity;
        y=y+1;
        document.getElementById("submitBut").style.display="block";
      }
  }
  function finalCall(){
    var table = document.getElementById("myTable");
    for(var i=1;i<n+1;i++){
      table.deleteRow(1);
    }
    n=0;
    var x=1;
    var list=[]
    list[0]=price;
    list[1]=parseInt(q);
    list[2]=q*price;
    tot+=list[2];
    if(sale[it]){
      sale[it][1]+=list[1]
      sale[it][2]+=list[2]
      console.log("Repeated",sale[it])
    }
    else{
      sale[it]=list;
      console.log("New")
```

```
        }

        for(var key in sale){
          var row = table.insertRow(x);
          row.className = "text-gray-700 dark:text-gray-400"
          var cell1 = row.insertCell(0);
          var cell2 = row.insertCell(1);
          var cell3 = row.insertCell(2);
          var cell4 = row.insertCell(3)
          cell1.innerHTML = key;
          console.log(key)
          cell1.className = "px-4 py-3 text-sm temptable"
          cell2.innerHTML = sale[key][0];
          cell2.className = "px-4 py-3 text-sm temptable"
          cell3.innerHTML = sale[key][1];
          cell3.className = "px-4 py-3 text-sm temptable"
          cell4.innerHTML = sale[key][2];
          cell4.className = "px-4 py-3 text-sm temptable"
          x++;
          n++;
        }
        document.getElementById("quantity").value=" ";
        document.getElementById("iname").value=" ";
        document.getElementById("total").innerHTML=tot;
      }
      </script>
  </body>
</html>
```

13.2 GITHUB AND PROJECT DEMO LINK

Github Link :

https://github.com/IBM-EPBL/IBM-Project-23045-1659864868

Video Demo Link :

https://drive.google.com/file/d/1tBhYWAdjDvMsTBKrTebLDgbsOcX-YIiy/view?usp=share_link

Deployment Link :

http://169.51.200.58:32008/