INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

(*Cloud Application Development*)

1. INTRODUCTION

a. 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.

b. 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

a. 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

b. References

i. 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.

ii. 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.

iii. 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.

iv. **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)

v. **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.

c. Problem Statement Definition

A web application is to be designed and developed, which helps to address the above-mentioned issues.

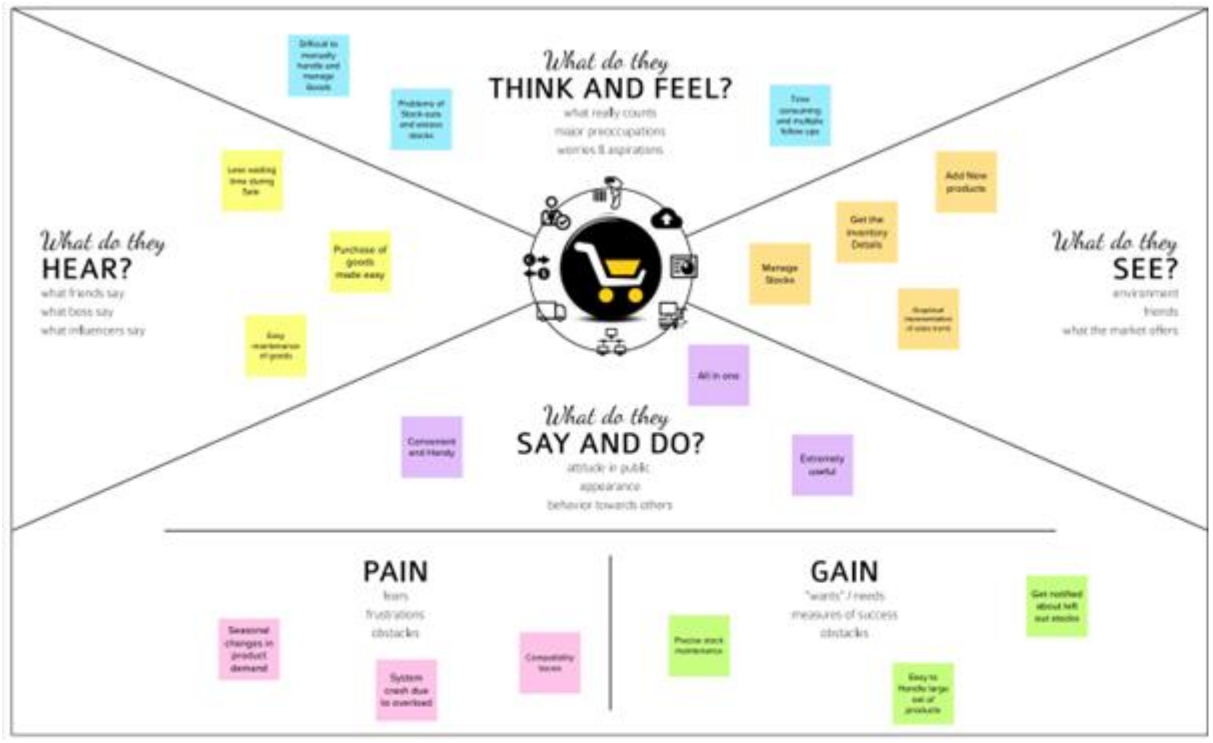
- The web app enables the registered user to update the inventory and as well as add a new product if the product details are given. As everything is automated no error occurs in this process and it is not time consuming as well.
- The user will be able to make a sale to consume the products in the inventory 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.

- If an item is left unsold for a long period of time, it can also be checked and ordered as required in the upcoming orders.

- 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.

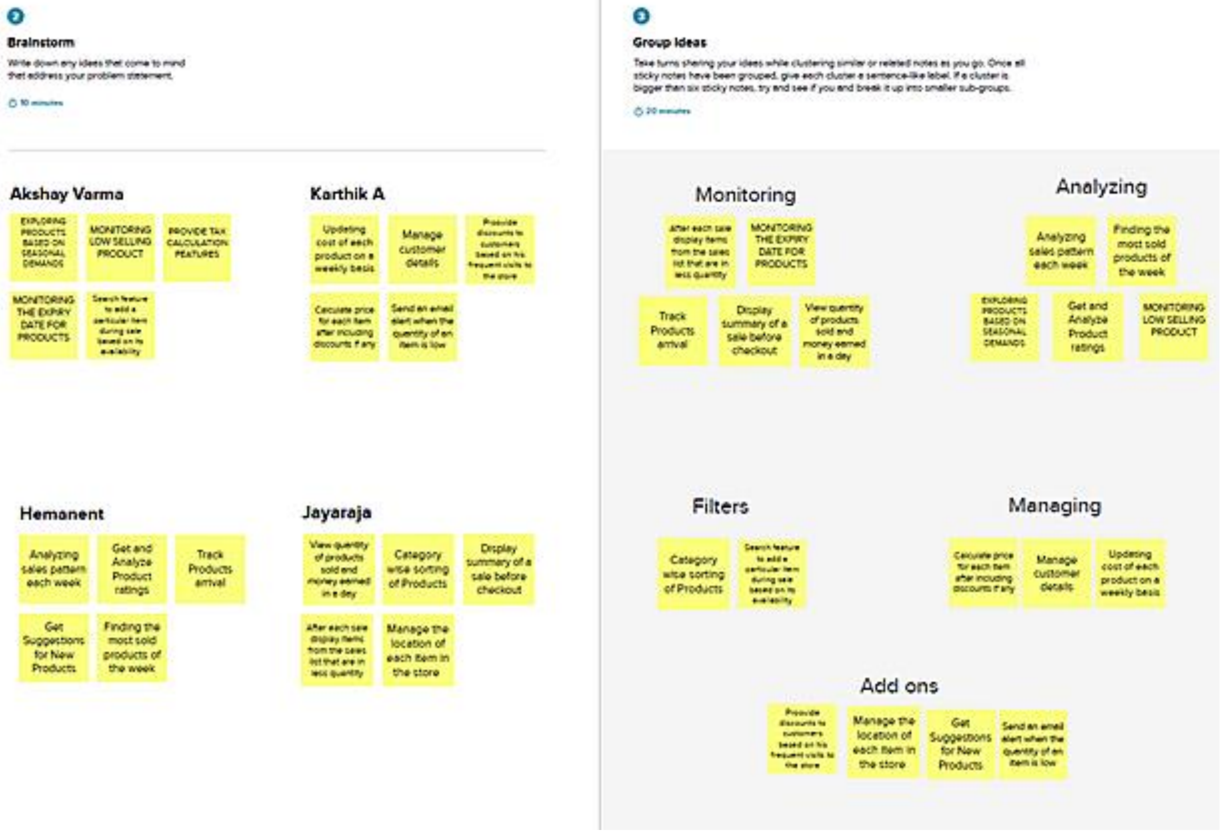
3. **IDEATION AND PROPOSED SOLUTION**

a. **Empathy Map Canvas**



Difficult to manually handle and manage Goods
Problems of Stock-outs and excess stocks
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Time consuming and multiple follow ups
Add New products
Get the inventory Details
Manage Stocks
Graphical representation of sales trend
What do they SEE?
environment
friends
what the market offers
Less waiting time during Sale
Purchase of goods made easy
Easy maintenance of goods
What do they HEAR?
what friends say
what boss say
what influencers say
All in one
Convenient and Handy
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Extremely useful
PAIN
fears
frustrations
obstacles
Seasonal changes in product demand
System crash due to overload
Compatibility issues
GAIN
"wants" / needs
measures of success
obstacles
Precise stock maintenance
Easy to Handle large set of products
Get notified about left out stocks

b. **Ideation and Brainstorming**



2

Brainstorm

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

10 minutes

Akshay Varma

- EXPLORING PRODUCTS BASED ON SEASONAL DEMANDS
- MONITORING LOW SELLING PRODUCT
- PROVIDE TAX CALCULATION FEATURES
- MONITORING THE EXPIRY DATE FOR PRODUCTS
- Search feature to add a particular item during sale based on its availability

Karthik A

- Updating cost of each product on a weekly basis
- Manage customer details
- Provide discounts to customers based on his frequent visits to the store
- Calculate price for each item after including discounts if any
- Send an email alert when the quantity of an item is low

Hemanent

- Analyzing sales pattern each week
- Get and Analyze Product ratings
- Track Products arrival
- Get Suggestions for New Products
- Finding the most sold products of the week

Jayaraja

- View quantity of products sold and money earned in a day
- Category wise sorting of Products
- Display summary of a sale before checkout
- After each sale display items from the sales list that are in less quantity
- Manage the location of each item in the store

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

Monitoring

- After each sale display items from the sales list that are in less quantity
- MONITORING THE EXPIRY DATE FOR PRODUCTS
- Track Products arrival
- Display summary of a sale before checkout
- View quantity of products sold and money earned in a day

Analyzing

- Analyzing sales pattern each week
- Finding the most sold products of the week
- EXPLORING PRODUCTS BASED ON SEASONAL DEMANDS
- Get and Analyze Product ratings
- MONITORING LOW SELLING PRODUCT

Filters

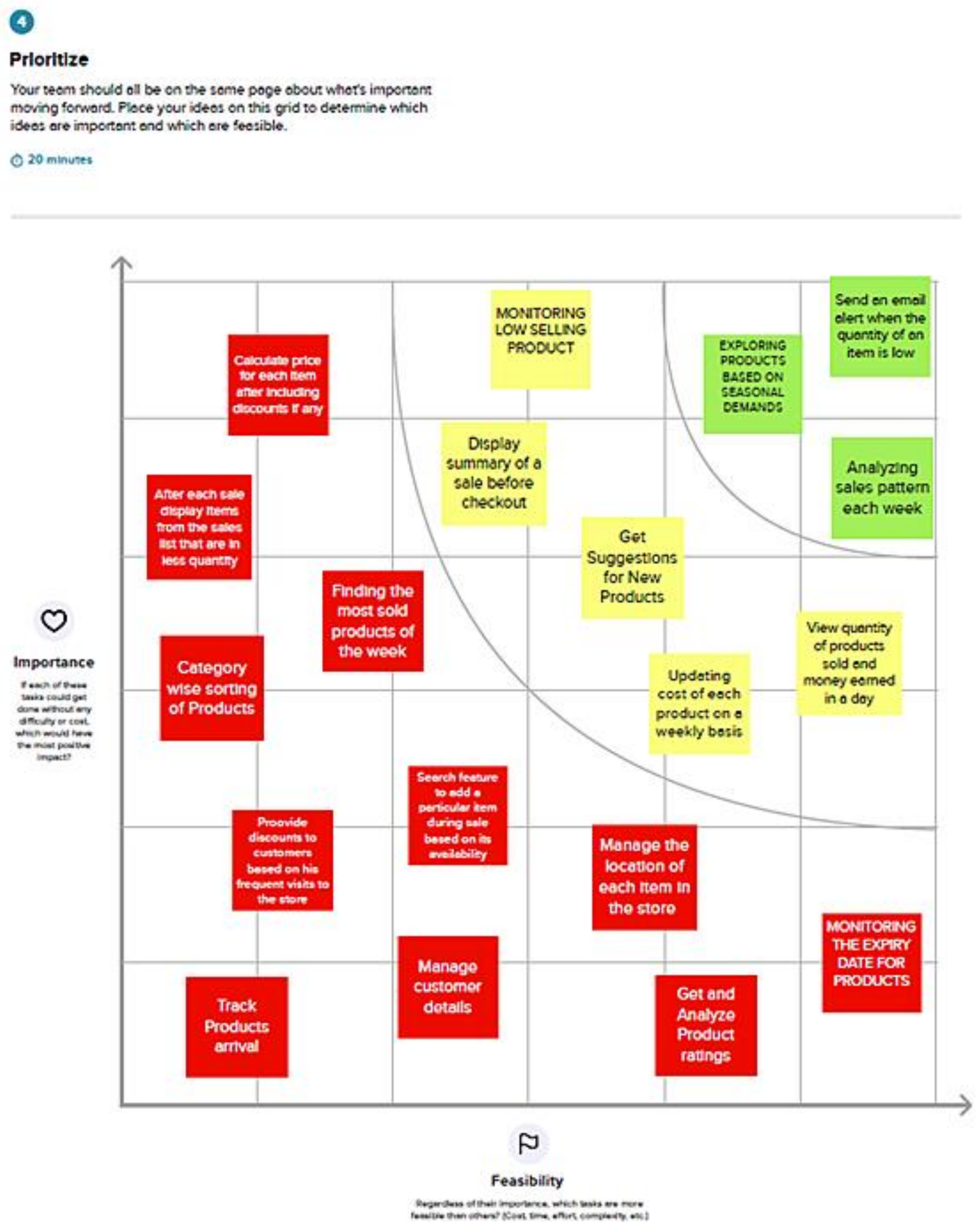
- Category wise sorting of Products
- Search feature to add a particular item during sale based on its availability

Managing

- Calculate price for each item after including discounts if any
- Manage customer details
- Updating cost of each product on a weekly basis

Add ons

- Provide discounts to customers based on his frequent visits to the store
- Manage the location of each item in the store
- Get Suggestions for New Products
- Send an email alert when the quantity of an item is low


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

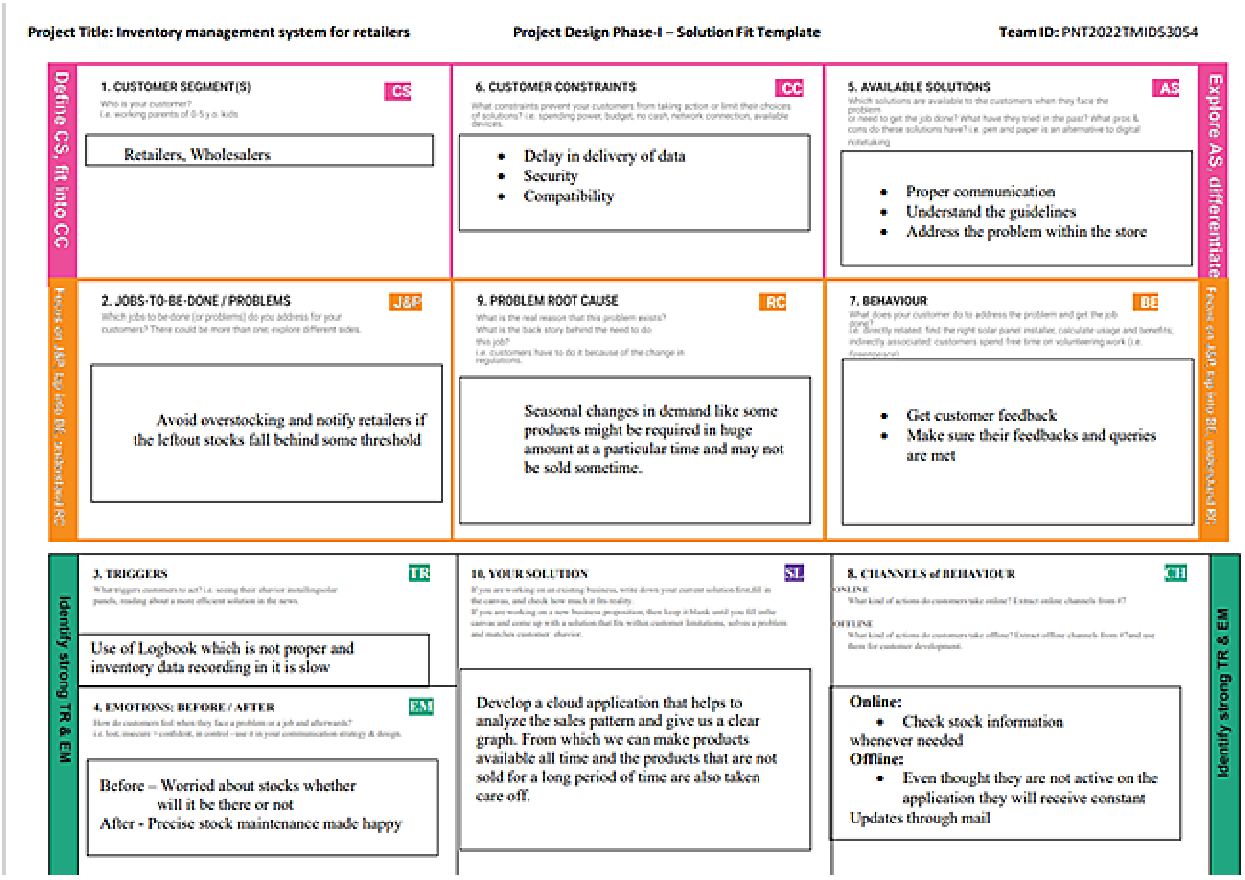
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

Calculate price for each item after including discounts if any
MONITORING LOW SELLING PRODUCT
EXPLORING PRODUCTS BASED ON SEASONAL DEMANDS
Send an email alert when the quantity of an item is low
Display summary of a sale before checkout
Analyzing sales pattern each week
After each sale display items from the sales list that are in less quantity
Get Suggestions for New Products
Finding the most sold products of the week
Category wise sorting of Products
View quantity of products sold and money earned in a day
Updating cost of each product on a weekly basis
Search feature to add a particular item during sale based on its availability
Provide discounts to customers based on his frequent visits to the store
Manage the location of each item in the store
MONITORING THE EXPIRY DATE FOR PRODUCTS
Track Products arrival
Manage customer details
Get and Analyze Product ratings

C. **Proposed Solution**

S.No	Parameter	Description
1	Problem Statement	• Retailers do not have any systematic system to record the inventory data quickly and safely as they only keep a note of it in a logbook which is not properly organized.
2	Idea/Solution Description	• Developing a cloud-based Inventory Management System to maintain and manage the stocks for retailers in an efficient manner.
3	Novelty/Uniqueness	• Trending stocks of the week gets displayed in the dashboard • Easy analysis of the sale through a Graphical representation so we can order and maintain stocks accordingly
4	Social Impact/Customer Satisfaction	• Customers will get more varieties due to High availability of Products • Customer Feedback will be collected and reviewed by the retailers for improvements if any.
5	Business Model (Financial Benefit)	• Retailers can understand the deepest customer needs and order accordingly so wastage of stocks could be avoided
6	Scalability of Solution	• It enables multiple retailers to collaborate and maintain stocks very easily at one single place.

d. **Problem Solution Fit**



Project Title: Inventory management system for retailers **Project Design Phase-I – Solution Fit Template** **Team ID: PNT2022TMID53054**

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

Retailers, Wholesalers

6. CUSTOMER CONSTRAINTS CC

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.

- Delay in delivery of data
- Security
- Compatibility

5. AVAILABLE SOLUTIONS AS

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

- Proper communication
- Understand the guidelines
- Address the problem within the store

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

Avoid overstocking and notify retailers if the leftout stocks fall behind some threshold

9. PROBLEM ROOT CAUSE RC

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.

Seasonal changes in demand like some products might be required in huge amount at a particular time and may not be sold sometime.

7. BEHAVIOUR BE

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)

- Get customer feedback
- Make sure their feedbacks and queries are met

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

Use of Logbook which is not proper and inventory data recording in it is slow

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Before – Worried about stocks whether will it be there or not
After - Precise stock maintenance made happy

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

Develop a cloud application that helps to analyze the sales pattern and give us a clear graph. From which we can make products available all time and the products that are not sold for a long period of time are also taken care off.

8. CHANNELS of BEHAVIOUR CH

ONLINE
What kind of actions do customers take online? Extract online channels from #7

OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

Online:
- Check stock information whenever needed

Offline:
- Even thought they are not active on the application they will receive constant Updates through mail

Identify strong TR & EM

4. REQUIREMENTS ANALYSIS

a. Functional Requirements

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Signup using the embedded form/through Gmail
FR-2	User Confirmation	Confirmation email on registering using form
FR-3	User Login	Login using the credentials provided during the time of registration/using OAuth
FR-4	Product Management	Manage different products sold at a store by entering a name, price, description and discount offered
FR-5	Stock Management	Add new stock to the store by entering the desired item name and the quantity acquired. Stock gets removed automatically after each sale

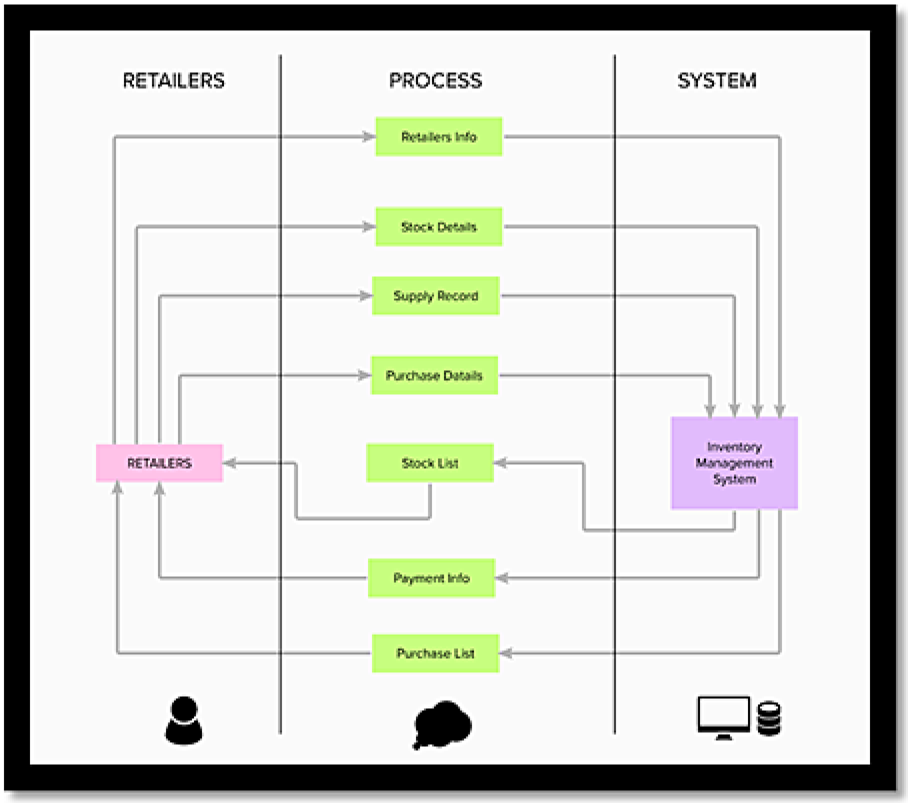
FR-6	Customers	Add a customer if he/she is new to the store. Use the details to map each sale with a particular customer
FR-7	Sale	Add different items and the quantity bought by a customer to calculate the total billing amount and register the sale
FR-8	Stock Shortage Alert	Alert the user via email when an item goes below the threshold value
FR-9	Historical Data Analysis	Visualizations for the list of items sold along with the quantity during a particular time period

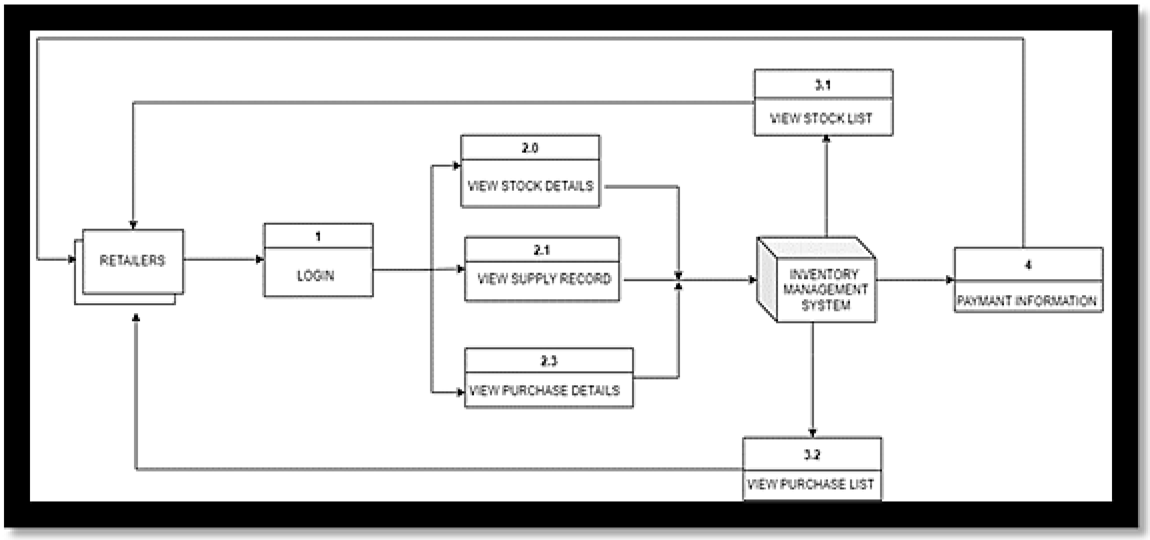
b. **Non-Functional Requirements**

NFR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	The system uses a web browser as an interface, which all users are familiar about and no specific training is required
NFR-2	**Security**	Every data specific to a user could be accessed only by the respective user as every login activity is authenticated and authorize
NFR-3	**Reliability**	The users should be able to access the correct data at all times
NFR-4	**Performance**	The system should not take a longer time to send a response to the user that he is in need of and the resources should be allocated accordingly for different tasks such as the visualisations can take more time but whereas registering a sale/updating the inventory shouldn't
NFR-5	**Availability**	The system should be accessible at all times - 24/7 when the users aren't notified about the server maintenance
NFR-6	**Scalability**	The system should be able to accept any kind of new changes in the near future such as increase in the user count, throughput of data, extending it to hand-held devices …

5. **PROJECT DESIGN**

a. **Data Flow Diagrams**





b. **Solution and Technical Architecture**

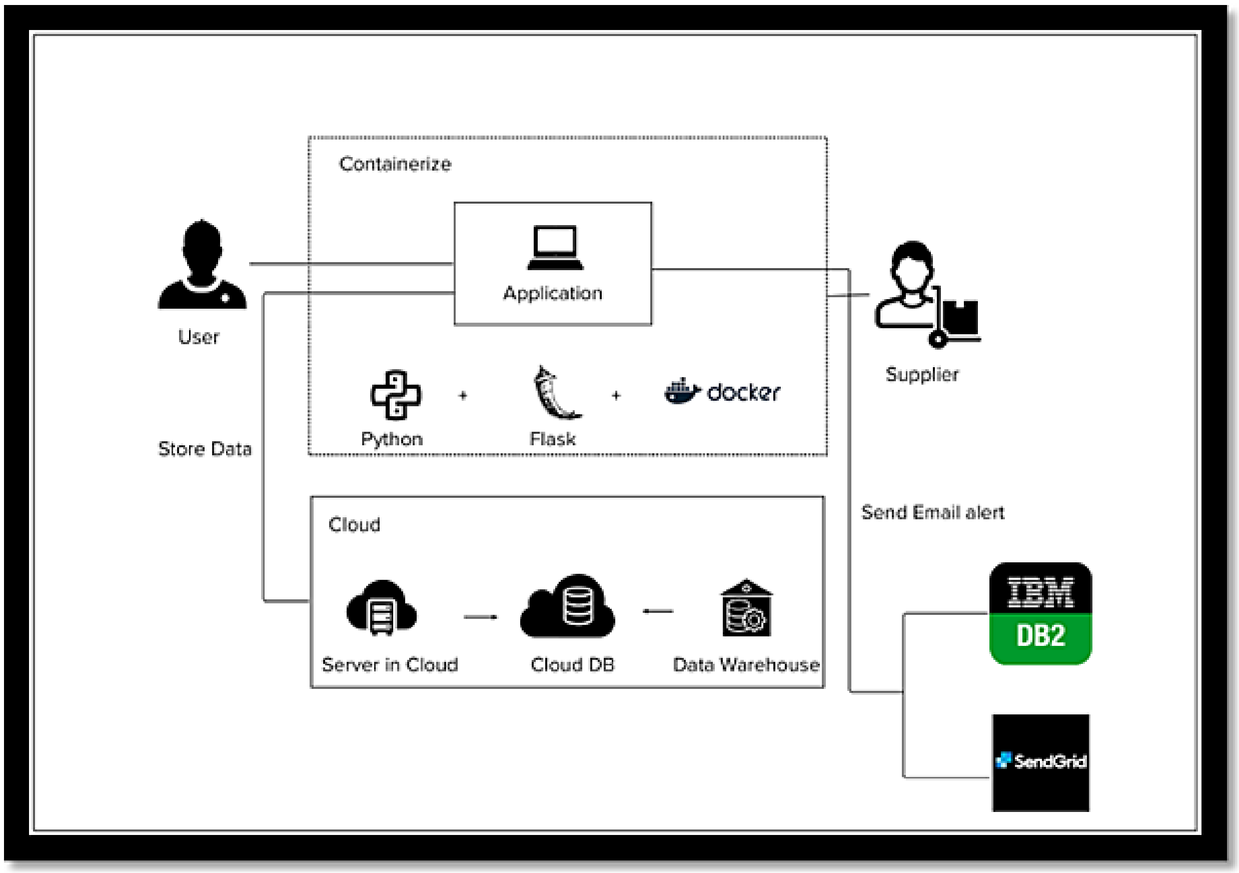


Figure: Solution Architecture

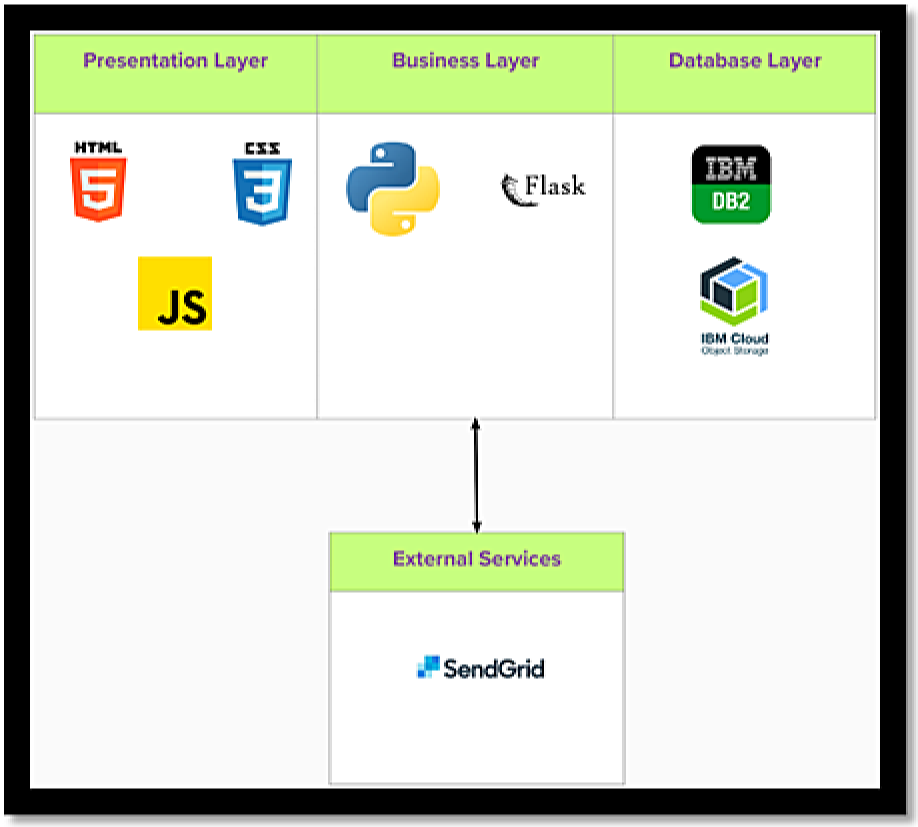


Figure: Technical Architecture

c. User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priorit y	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 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 Gmail	I can register & access dashboard	Low	Sprint-1

				with Gmail Login		
	Login	USN-4	As a user, I can log into the application by entering email & password	I can log into the application by entering email and password	High	Sprint-1
	Dashboard	USN-5	As a user, I can login into my application	I can access my dashboard	High	Sprint-1
	Add new items	USN-6	As a user, I can add new items to the inventory along with the quantity bought	I can add new items to the inventory	High	Sprint-2
	Search	USN-7	As a user, I can search for a particular item for its availability	I can find an item with item id or name	High	Sprint-2
	Customer		As a user, I add a new customer / manage their details before performing a sale	I can add/view a customer	High	Sprint-2
	Sale	USN-8	As a user, I can perform sale for a customer by entering the items and the quantity bought by him	I can add items to the customer list and sum up each value to calculate total sale value	High	Sprint-2
	Visualizations	USN-9	As a user, I can view the list of items along with the quantity bought in a particular time period	I can view the items that were ordered the most and the least	High	Sprint-3
Customer Care Executive	Feedback	USN-10	As a user, I can record the feedbacks from different customers about the products and services	User friendly customer support	High	Sprint-3
Administrator	Responsibility	USN-11	As an administrator, I can only add and maintain users to this application	I can add and maintain users	High	Sprint-1

6. **PROJECT PLANNING AND SCHEDULING**

a. **Sprint Planning and Estimation**

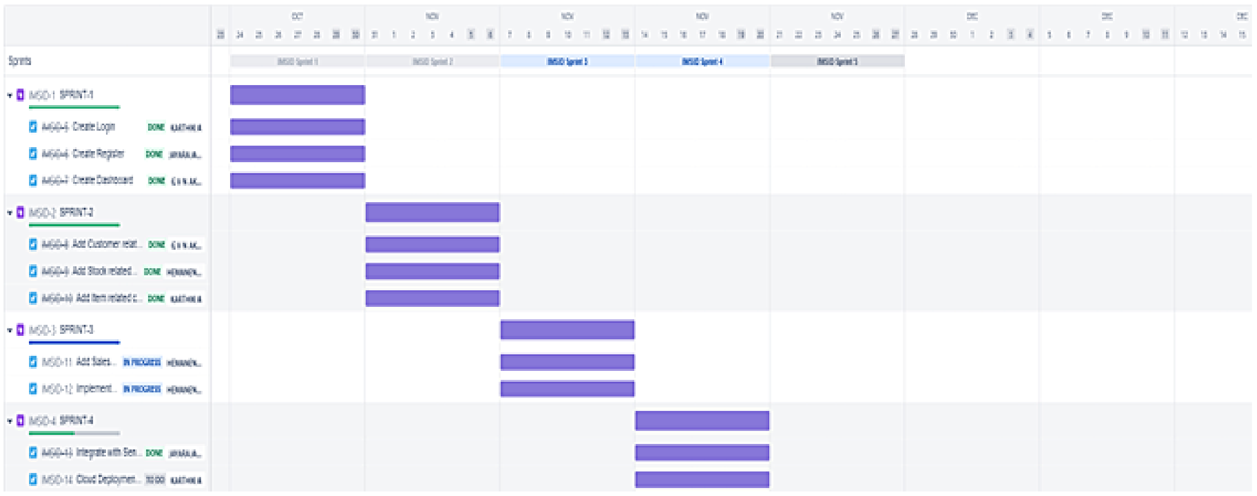
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Login	USN-1	As a user, I can log into the application by entering email & password	2	High	Karthik A
Sprint-1	Registration	USN-2	As a user, I can register for the application by entering my email, password, and confirming my password. As a user, I will receive confirmation email once I have registered for the application	2	High	Jayaraja S.K
Sprint-1	Dashboard	USN-3	As a user, I can login into my application	3	High	Akshay Varma
Sprint-2	Add Customer related components	USN-4	As a user, I add a new customer / manage their details before performing a sale	3	High	Akshay Varma
Sprint-2	Add Stock related components	USN-5	As a user, I can search for a particular item for its availability	3	High	Hemanent S
Sprint-2	Add item related components	USN-6	As a user, I can add new items to the inventory along with the quantity bought	3	High	Karthik A

Sprint-3	Add Sales related components	USN-7	As a user, I can perform sale for a customer by entering the items and the quantity bought by him	3	High	Hemanent S
Sprint-3	Implement Data Visualization	USN-8	As a user, I can view the list of items along with the quantity bought in a particular time period	2	High	Hemanent S
Sprint-4	Integrate with SendGrid Service	USN-9	As a user, I will receive email notifications	2	High	Jayaraja S.K
Sprint-4	Cloud Deployment	USN-10	As a user, I can view the application anywhere with internet access	3	High	Karthik A

b. **Sprint Delivery Schedule**

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	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)

c. **Reports from JIRA**



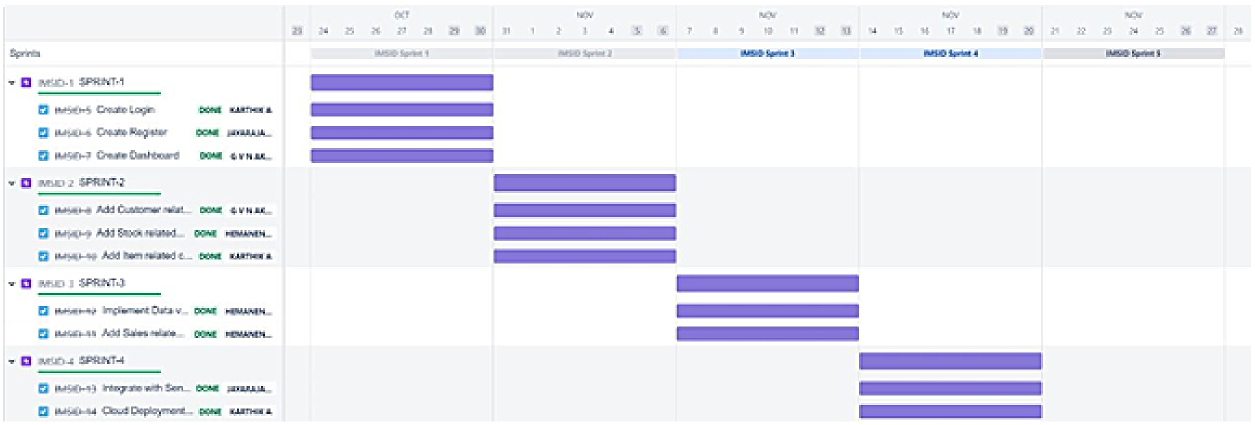
Sprints: IMSID Sprint 1 | IMSID Sprint 2 | IMSID Sprint 3 | IMSID Sprint 4 | IMSID Sprint 5

IMSID-1 SPRINT-1
- IMSID-5 Create Login DONE KARTHIK A
- IMSID-6 Create Register DONE JAYARAJA...
- IMSID-7 Create Dashboard DONE G V N AK...

IMSID-2 SPRINT-2
- IMSID-8 Add Customer relat... DONE G V N AK...
- IMSID-9 Add Stock related... DONE HEMANEN...
- IMSID-10 Add Item related c... DONE KARTHIK A

IMSID-3 SPRINT-3
- IMSID-11 Add Sales... IN PROGRESS HEMANEN...
- IMSID-12 Implement... IN PROGRESS HEMANEN...

IMSID-4 SPRINT-4
- IMSID-13 Integrate with Sen... DONE JAYARAJA...
- IMSID-14 Cloud Deploymen... TO DO KARTHIK A



Sprints: IMSID Sprint 1 | IMSID Sprint 2 | IMSID Sprint 3 | IMSID Sprint 4 | IMSID Sprint 5

IMSID-1 SPRINT-1
- IMSID-5 Create Login DONE KARTHIK A
- IMSID-6 Create Register DONE JAYARAJA...
- IMSID-7 Create Dashboard DONE G V N AK...

IMSID-2 SPRINT-2
- IMSID-8 Add Customer relat... DONE G V N AK...
- IMSID-9 Add Stock related... DONE HEMANEN...
- IMSID-10 Add Item related c... DONE KARTHIK A

IMSID-3 SPRINT-3
- IMSID-12 Implement Data v... DONE HEMANEN...
- IMSID-11 Add Sales relate... DONE HEMANEN...

IMSID-4 SPRINT-4
- IMSID-13 Integrate with Sen... DONE JAYARAJA...
- IMSID-14 Cloud Deployment... DONE KARTHIK A

7. CODING AND SOLUTIONING

a. Feature – 1

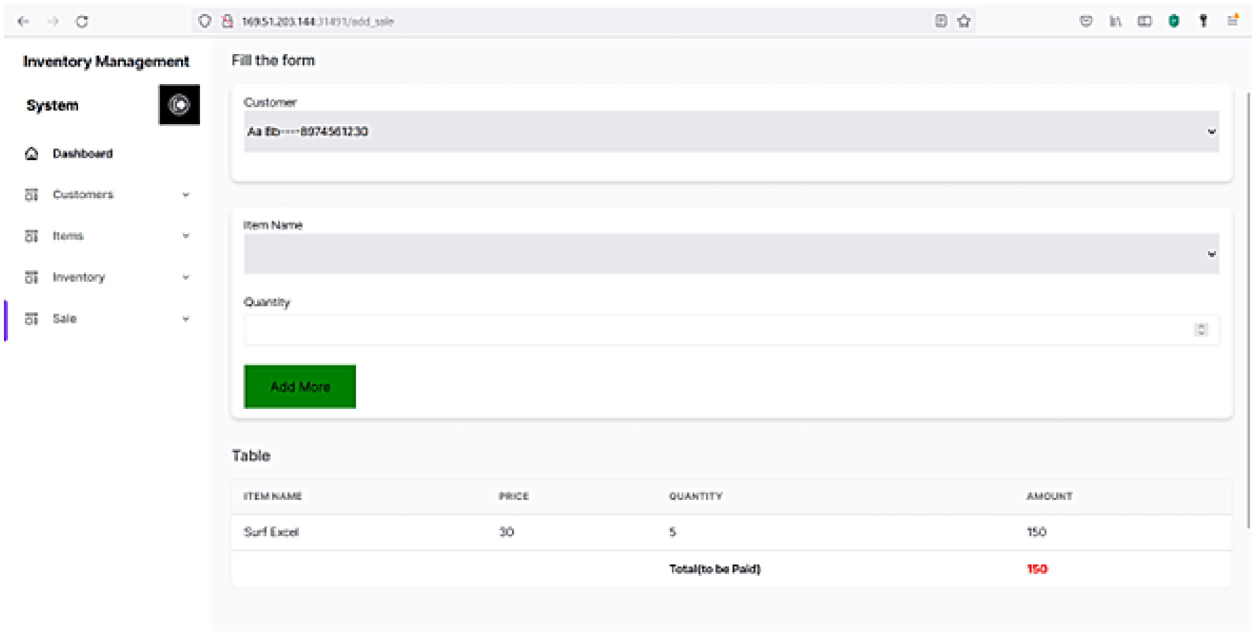
i. Description

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

ii. Source Code

```
@app.route("/add_sale", methods=['GET','POST'])
def add_sale():
    # ITEMS
    items=list()
    ret_id=request.cookies.get('userID')
    query = '''select * from items where retailer_id=\'{}\''''.format(ret_id)
    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=list()
    ret_id=request.cookies.get('userID')
    query = '''select * from customer where retailer_id=\'{}\''''.format(ret_id)
    exec_query = ibm_db.exec_immediate(conn, query)
    row = ibm_db.fetch_both(exec_query)
    while(row):
        custname.append({"name":row["CUSTOMER_NAME"],"id":row["CUSTOMER_ID"]})
        row = ibm_db.fetch_both(exec_query)
    if request.method=="GET" and request.cookies.get('userID')!=None:
        return render_template("Dashboard/add_sale.html",items=items,cname=custname,status="")
    elif request.method=="POST" and request.cookies.get('userID')!=None:
        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)
        print(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)
        print(item_list,n)
        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)
            #MAIL
            products=list()
            ret_id=request.cookies.get('userID')
            query = '''select item_name,left_out from items where left_out<5 and 
retailer_id=\'{}\''''.format(ret_id)
            exec_query = ibm_db.exec_immediate(conn, query)
            row = ibm_db.fetch_both(exec_query)
            while(row):
                print(row)
                products.append({"name":row["ITEM_NAME"],"quantity":row["LEFT_OUT"]})
                row = ibm_db.fetch_both(exec_query)
            retailer_id=request.cookies.get('userID')
            query = '''select email from retailers where retailer_id=\'{}\''''.format(retailer_id)
            exec_query = ibm_db.exec_immediate(conn, query)
            row = ibm_db.fetch_both(exec_query)
            retailer_email=row["EMAIL"]
            print(retailer_email)
            stock_alert_mail(retailer_email, products)
            # 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,status="Sale Success")
```

iii. **Screenshots**


169.51.203.144:31491/add_sale

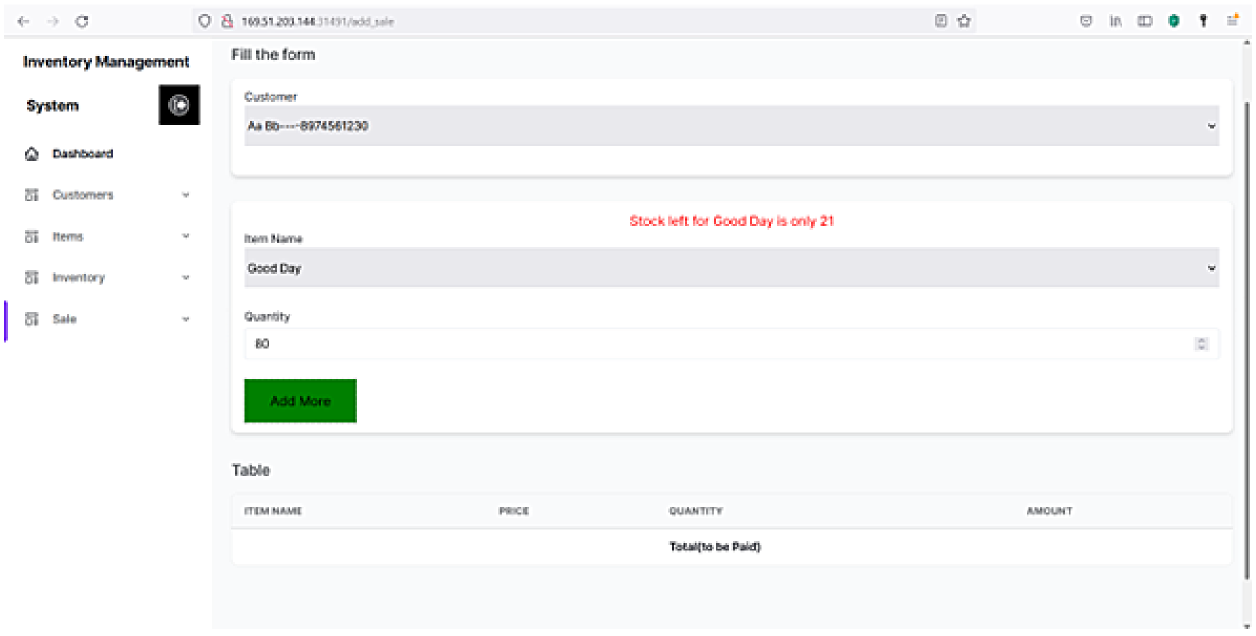
Inventory Management System

Fill the form

Customer	Aa Bb----8974561230
Item Name	
Quantity	

Table

ITEM NAME	PRICE	QUANTITY	AMOUNT
Surf Excel	30	5	150
		Total(to be Paid)	150


169.51.203.144:31491/add_sale

Inventory Management System

Fill the form

Stock left for Good Day is only 21

Customer	Aa Bb----8974561230
Item Name	Good Day
Quantity	80

Table

ITEM NAME	PRICE	QUANTITY	AMOUNT
		Total(to be Paid)	

b. **Feature – 2**

i. **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

ii. Source Code

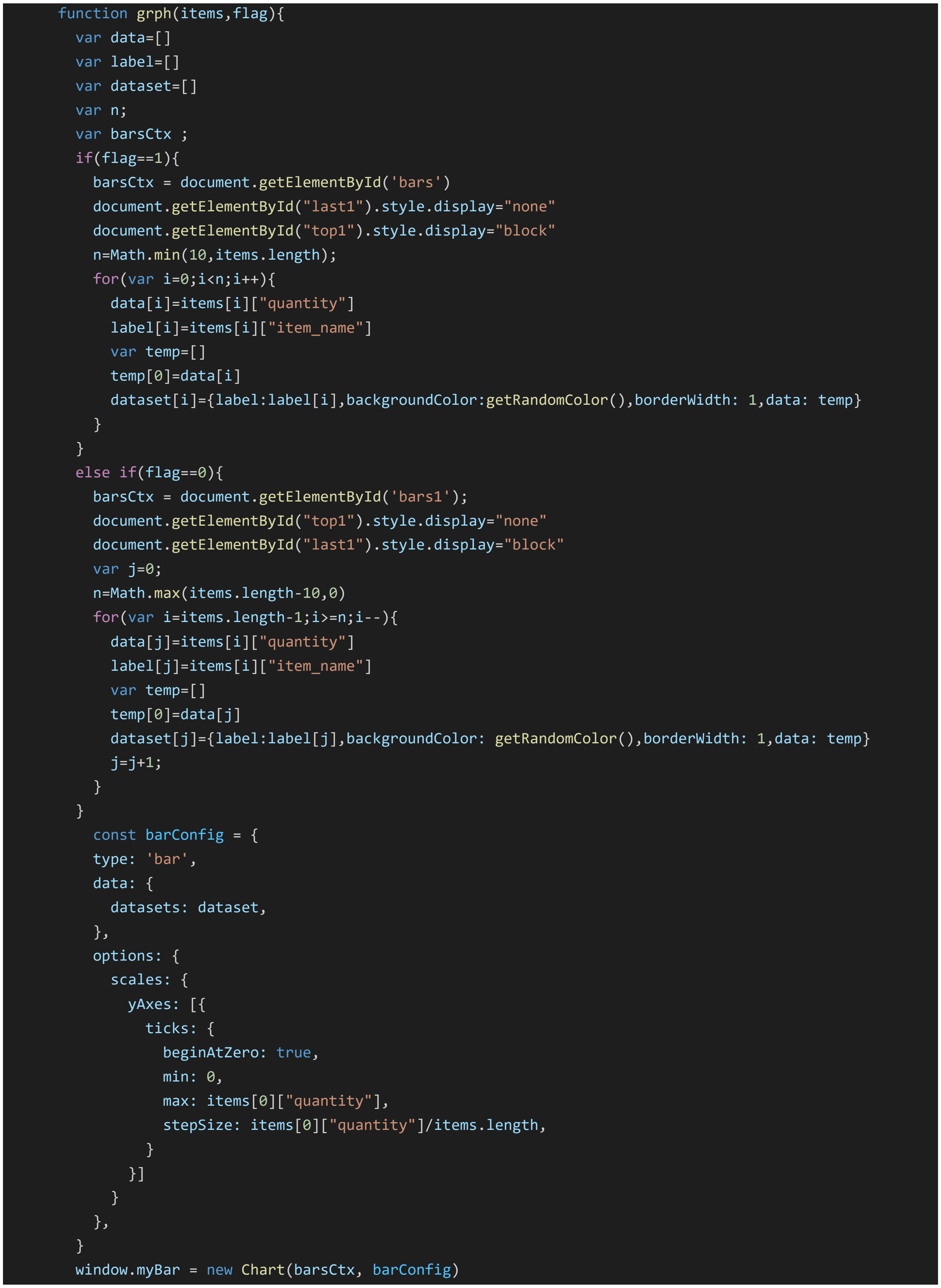
i. app.py

```
@app.route("/view_sale", methods=['GET','POST'])
def view_sale():
    items=list()
    if request.method=="GET" and request.cookies.get('userID')!=None:
        return render_template("Dashboard/view_sale.html",items=items)
    elif request.method=="POST" and request.cookies.get('userID')!=None:
        start=request.form["start_date"]
        end=request.form["end_date"]
        query = '''select item_id,sum(quantity) from sale_items where sale_id in (select sale_id from sale 
where sale_date<= \'{}\' and sale_date>=\'{}\') group by(item_id) order by sum(quantity) 
desc'''.format(end,start)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        while(row):
            ret_id=request.cookies.get('userID')
            query1 = '''select item_name,price from items where item_id=\'{}\' and 
retailer_id=\'{}\'''.format(row[0],ret_id)
            exec_query1 = ibm_db.exec_immediate(conn, query1)
            row1 = ibm_db.fetch_both(exec_query1)
            if(row1):

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)
```

ii. view_sale.html

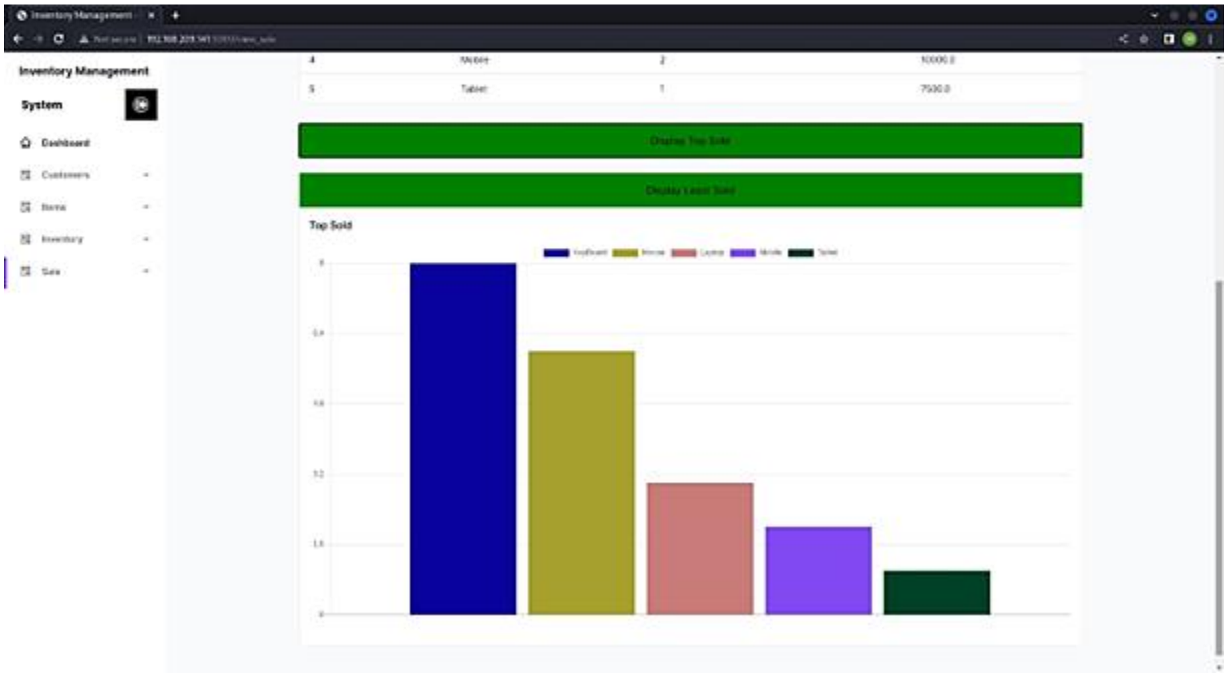
```
<script>
      function getRandomColor() {
        var letters = '0123456789ABCDEF';
        var color = '#';
        for (var i = 0; i < 6; i++) {
          color += letters[Math.floor(Math.random() * 16)];
        }
        return color;
      }
```



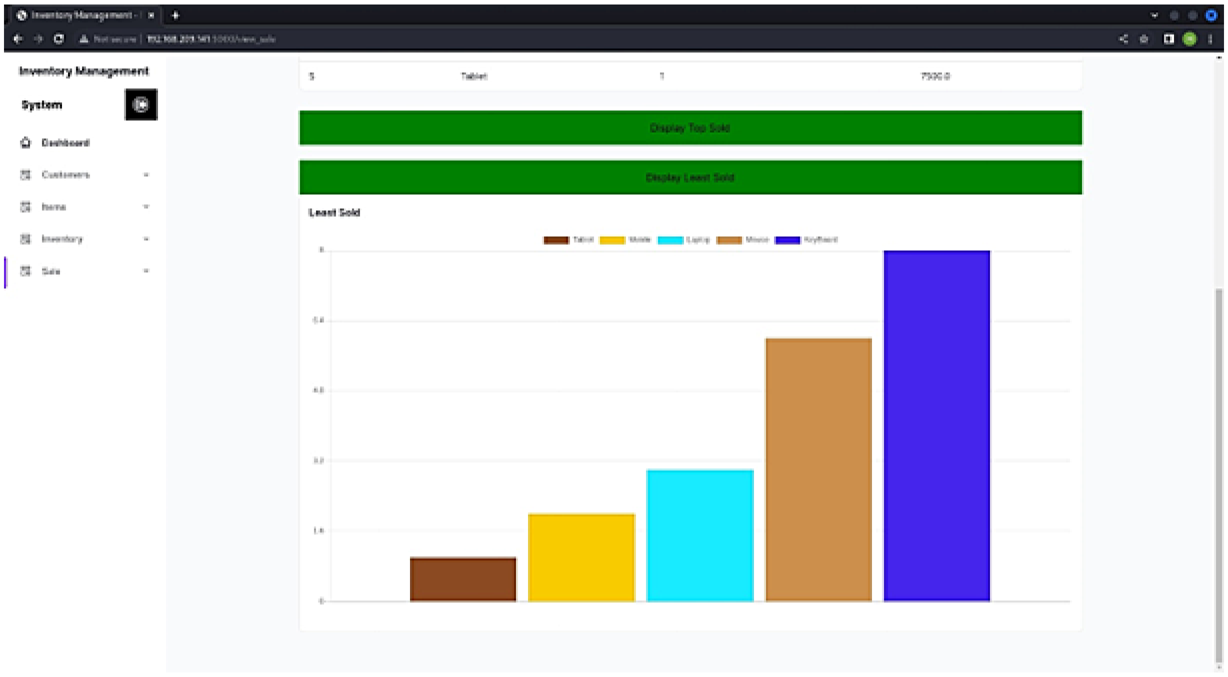
```
function grph(items,flag){
  var data=[]
  var label=[]
  var dataset=[]
  var n;
  var barsCtx ;
  if(flag==1){
    barsCtx = document.getElementById('bars')
    document.getElementById("last1").style.display="none"
    document.getElementById("top1").style.display="block"
    n=Math.min(10,items.length);
    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;
    }
  }
    const barConfig = {
    type: 'bar',
    data: {
      datasets: dataset,
    },
    options: {
      scales: {
        yAxes: [{
          ticks: {
            beginAtZero: true,
            min: 0,
            max: items[0]["quantity"],
            stepSize: items[0]["quantity"]/items.length,
          }
        }]
      }
    },
  }
  window.myBar = new Chart(barsCtx, barConfig)
```

```
    }
</script>
```

iii. **Screenshots**



4	Mobile	2	10000.0
5	Tablet	1	7500.0



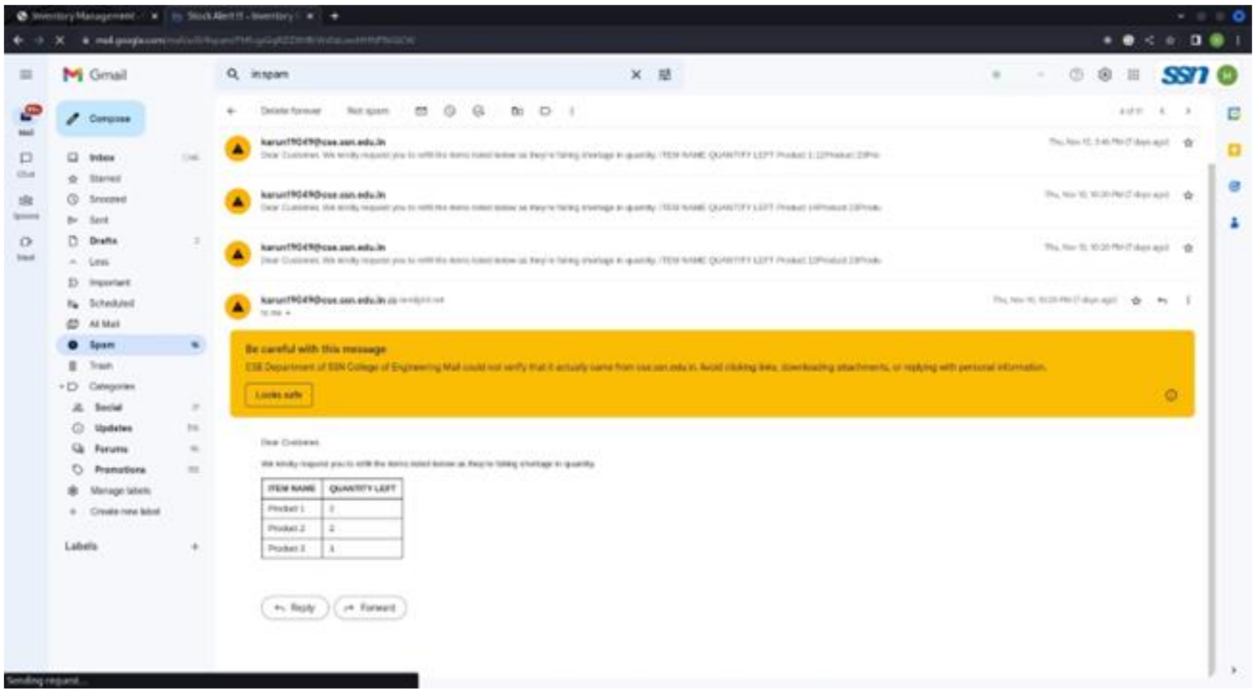
5	Tablet	1	7500.0

c. **Feature – 3**

i. **Description**

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

ii. **Screenshots**



karun19049@cse.ssn.edu.in

Dear Customer, We kindly request you to refill the items listed below as they're facing shortage in quantity. ITEM NAME QUANTITY LEFT Product 1 ...

karun19049@cse.ssn.edu.in

Dear Customer, We kindly request you to refill the items listed below as they're facing shortage in quantity. ITEM NAME QUANTITY LEFT Product ...

karun19049@cse.ssn.edu.in

Dear Customer, We kindly request you to refill the items listed below as they're facing shortage in quantity. ITEM NAME QUANTITY LEFT Product ...

karun19049@cse.ssn.edu.in

to me

Be careful with this message

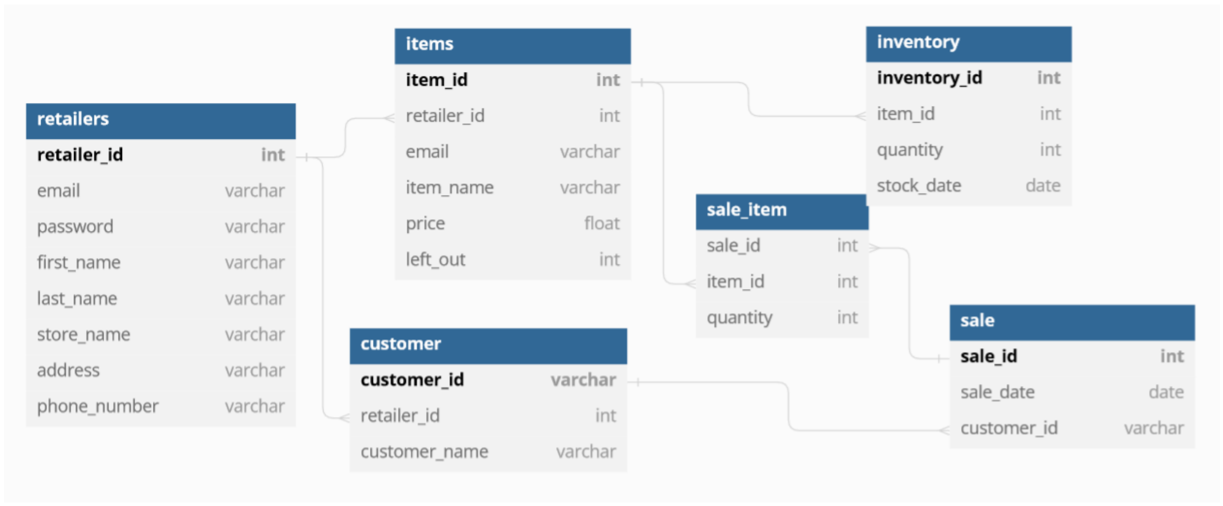
CSE Department of SSN College of Engineering Mail could not verify that it actually came from cse.ssn.edu.in. Avoid clicking links, downloading attachments, or replying with personal information.

Dear Customer,

We kindly request you to refill the items listed below as they're facing shortage in quantity.

ITEM NAME	QUANTITY LEFT
Product 1	[illegible]
Product 2	2
Product 3	[illegible]

d. **Database Schema**



8. TESTING

a. Test Cases

i. Sprint – 1

Feature Type	Componen t	Test Scenario	Steps To Execute	Expected Result	Status
Functional	Login	Verify whether User is able to login into account	1. Enter email and password 2. Click Login button	User should be able to login into account	Pass
Functional	Login	Verify whether a notification is displayed if the credentials are invalid	1. Enter email and password 2. Click Login button	Message for incorrect credentials should be displayed	Pass
UI	Login	Verify whether the submit button is activated only when all the fields with desired type are entered		If all the fields are not entered, then a message has to be displayed that all the fields are not entered	Pass
Functional	Signup	Verify whether User is able to signup	1. Enter name, storename, mobile number, email and password 2. Click Signup button	User should be able to signup	Pass
Functional	Signup	Verify whether a notification is displayed if the email is already used	1. Enter name, storename, mobile number, email and password 2. Click Signup button	Notify the user that the email is already used	Pass
UI	Signup	Verify whether the submit button is activated only when all the fields with desired type are entered		If all the fields are not entered, then a message has to be displayed that all the fields are not entered	Pass

ii. **Sprint – 2**

Feature Type	Componen t	Test Scenario	Steps To Execute	Expected Result	Status
Functional	Customer	Verify retailer is able to add a customer	Enter customer id and name and click on add	Notify - Customer details successfully addded	Pass
Functional	Customer	Verify whether the retailer gets to know that the customer details already exists	Enter customer id and name and click on add	Notify - Customer already exists	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	Application should display that the name is missing	Pass
Functional	Customer	View the list of Customers	Click on view customers from the dashboard	All the customer details should be displayed as a table	Pass
Functional	Item	Verify whether the retailer is able to add a new item to the store along with the price	1. Enter the item name with some unique identifier 2. Click submit	Notify that the item has been added successfully	Pass
Functional	Item	View the list of Items	Click on view items from the dashboard	All the items should be displayed as a table	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	Application should display that price is missing	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	Application should display that price will accept only numbers	Pass
Functional	Inventory	Verify whether the retailer is able to add stock details into the application		Notify the user that the details has been added	Pass

Functional	Inventory	Verify whether the stock date is recorded automatically	Go to view stock details and select the desired item and today's date to see if the current date has been noted automatically	The current date on which the stock has been added should be displayed	Pass
UI	Inventory	Verify whether the items available in the store are displayed in the drop-down	Click the drop-down to see the list of items	All the items available in the store should be listed	Pass
UI	Inventory	Verify whether the submit button gets activated only when all the required fields are entered	1. Select the item and enter the quantity 2. Click submit	Application should display that the name is missing	Pass

iii. **Sprint – 3**

Feature Type	Component	Test Scenario	Steps To Execute	Expected Result	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	Multiple items should be displayed along with its quantity	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	Total price should be calcuated and displayed automatically	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	Notify the user that the quantity entered is above the available stock	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	Date and the quantity sold for the item in the selected range should be displayed as a table	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	List of item name and their total quantity sold should be displayed	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	User should be notified that the date's are mis-matching	Pass
UI	Historical Analysis	Verify whether a graph is displayed for the top 5 items that were sold the most in the specified range		Bar chart should be displayed for top 5 items where the x-axis should correspond to the item name and the y-axis to the count	Pass
UI	Historical Analysis	Verify whether a graph is displayed for the top 5 items that were sold the least in the specified range		Bar chart should be displayed for least 5 items where the x-axis should correspond to the item name and the y-axis to the count	Pass

iv. Sprint – 4

Feature Type	Component	Test Scenario	Steps To Execute	Expected Result	Status
Functional	Stock Shortage Alert	Verify whether the retailer receives an email when some items fall shortage in quantity	Perform a sale	Email should have the list of item names and its respective quantity left out	Pass

b. User Acceptance Testing

i. **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

ii. **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

iii. **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

iv. **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	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	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. ADVANTAGES AND DIS-ADVANTAGES

a. 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

b. **Dis-Advantages**

6. Could not be accessed using handheld devices

7. 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

8. 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

10. **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

11. **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

12. APPENDIX

a. Source Code

Directory Structure

- app.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

app.py

```
from flask import *

from datetime import date
import ibm_db
import sendgrid
import os
from sendgrid.helpers.mail import Mail, Email, To, Content
app=Flask(__name__)
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=2f3279a5-73d1-4859-88f0-
a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=30756;SECURITY=SSL;SSLServerCertificate:D
igiCertGlobalRootCA;PROTOCOL=TCPIP;UID=rch34173;PWD=MwsEJWWZqnmoeUkt;", "", "")

@app.route("/", methods=['GET'])
def login():
    if request.method=='GET'  :
        resp=make_response(render_template('Login/index.html'))
        resp.set_cookie('userID',"",expires=0)
        return resp
    return render_template("Login/index.html",status="",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':
        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("/", methods=['POST'])
def setcookie():
    if request.method=='POST':
        email=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:
                resp=make_response(render_template('Dashboard/index.html'))
                resp.set_cookie('userID',str(row['RETAILER_ID']))
                return resp


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



@app.route("/dashboard", methods=['GET','POST'])
def dashboard():
    if request.method=="GET" and request.cookies.get('userID')!=None:
        return render_template("Dashboard/index.html")


@app.route("/add_customer", methods=['GET','POST'])
def add_customer():
    if request.method=="GET" and request.cookies.get('userID')!=None:
        return render_template("Dashboard/add_customer.html")
    elif request.method=="POST" and request.cookies.get('userID')!=None:
        name=request.form["name"]
        retailer_id=request.cookies.get('userID')
        id=int(request.form["id"])
        query = '''select * from customer where customer_id = \'{}\' and
retailer_id=\'{}\''''.format(id,retailer_id)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        retailer_id=request.cookies.get('userID')
        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" and request.cookies.get('userID')!=None:


        ret_id=request.cookies.get('userID')
        query = '''select * from customer where retailer_id=\'{}\''''.format(ret_id)


        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" and request.cookies.get('userID')!=None:
        return render_template("Dashboard/add_item.html")
    elif request.method=="POST" and request.cookies.get('userID')!=None:
        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)
        retailer_id=request.cookies.get('userID')
        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")
```

```python
@app.route("/view_item", methods=['GET','POST'])
def view_item():
    if request.method=="GET" and request.cookies.get('userID')!=None:
        ret_id=request.cookies.get('userID')
        query = '''select * from items where retailer_id=\'{}\''''.format(ret_id)
        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=[]
    retailer_id=request.cookies.get('userID')
    query = '''select * from items where retailer_id=\'{}\''''.format(retailer_id)
    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" and request.cookies.get('userID')!=None:
        return render_template("Dashboard/add_inventory.html",name=name,len=len(name),status=" ")
    elif request.method=="POST" and request.cookies.get('userID')!=None:
        fname=request.form["name"]
        quantity=request.form["quantity"]
        stock_date=date.today()
        # Finding ITEM ID
        ret_id=request.cookies.get('userID')
        query = '''select item_id from items where item_name = \'{}\' and
retailer_id=\'{}\''''.format(fname,ret_id)
        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,stock_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")
```

```python
@app.route("/view_inventory", methods=['GET','POST'])
def view_inventory():
    name=[]
    retailer_id=request.cookies.get('userID')
    query = '''select * from items where retailer_id=\'{}\''''.format(retailer_id)
    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" and request.cookies.get('userID')!=None:
        return render_template("Dashboard/view_inventory.html",name=name,items=items)
    elif request.method=="POST" and request.cookies.get('userID')!=None:
        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()
    ret_id=request.cookies.get('userID')
    query = '''select * from items where retailer_id=\'{}\''''.format(ret_id)
    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=list()
    ret_id=request.cookies.get('userID')
    query = '''select * from customer where retailer_id=\'{}\''''.format(ret_id)
    exec_query = ibm_db.exec_immediate(conn, query)
    row = ibm_db.fetch_both(exec_query)
    while(row):
        custname.append({"name":row["CUSTOMER_NAME"],"id":row["CUSTOMER_ID"]})
        row = ibm_db.fetch_both(exec_query)
    if request.method=="GET" and request.cookies.get('userID')!=None:
        return render_template("Dashboard/add_sale.html",items=items,cname=custname,status="")
    elif request.method=="POST" and request.cookies.get('userID')!=None:
        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)
        print(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)
        print(item_list,n)
        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)
            #MAIL
            products=list()
            ret_id=request.cookies.get('userID')
            query = '''select item_name,left_out from items where left_out<5 and 
retailer_id=\'{}\''''.format(ret_id)
            exec_query = ibm_db.exec_immediate(conn, query)
            row = ibm_db.fetch_both(exec_query)
            while(row):
                print(row)
                products.append({"name":row["ITEM_NAME"],"quantity":row["LEFT_OUT"]})
                row = ibm_db.fetch_both(exec_query)
            retailer_id=request.cookies.get('userID')
            query = '''select email from retailers where retailer_id=\'{}\''''.format(retailer_id)
            exec_query = ibm_db.exec_immediate(conn, query)
            row = ibm_db.fetch_both(exec_query)
            retailer_email=row["EMAIL"]
            print(retailer_email)
            stock_alert_mail(retailer_email, products)
            # 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,status="Sale Success")



@app.route("/view_sale", methods=['GET','POST'])
def view_sale():
    items=list()
    if request.method=="GET" and request.cookies.get('userID')!=None:
        return render_template("Dashboard/view_sale.html",items=items)
    elif request.method=="POST" and request.cookies.get('userID')!=None:
        start=request.form["start_date"]
        end=request.form["end_date"]
        query = '''select item_id,sum(quantity) from sale_items where sale_id in (select sale_id from sale
where sale_date<= \'{}\' and sale_date>=\'{}\') group by(item_id) order by sum(quantity)
desc'''.format(end,start)
        exec_query = ibm_db.exec_immediate(conn, query)
        row = ibm_db.fetch_both(exec_query)
        while(row):
            ret_id=request.cookies.get('userID')
            query1 = '''select item_name,price from items where item_id=\'{}\' and
retailer_id=\'{}\''''.format(row[0],ret_id)
            exec_query1 = ibm_db.exec_immediate(conn, query1)
            row1 = ibm_db.fetch_both(exec_query1)
            if(row1):

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)


def stock_alert_mail(to_email, products):
    sg =
sendgrid.SendGridAPIClient(api_key='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'
)
    from_email = Email("karthik19046@cse.ssn.edu.in")


    to_email = To(to_email)


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


    msg = '''
    <html>


    <body>
        <p>Dear Customer,</p>
        <p>We kindly request you to refill the items listed below as they're falling shortage in
quantity.</p>

        <table style='border: 1px solid black; border-collapse: collapse;'>

        <tr>
            <td style="font-weight: bold; padding: 5px 10px 5px 10px; border: 1px solid black;">ITEM
NAME</td>
            <td style="font-weight: bold; padding: 5px 10px 5px 10px; border: 1px solid black;">QUANTITY
LEFT</td>
        </tr>
    '''

    for i in products:
        msg += "<tr>"

        msg += f'<td style="padding: 5px 10px 5px 10px; border: 1px solid black;">{i["name"]}</td>'
        msg += f'<td style="padding: 5px 10px 5px 10px; border: 1px solid black;">{i["quantity"]}</td>'



        msg += "</tr>"

    msg += '''
    </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__":
    port = int(os.environ.get('PORT', 5000))
    app.run(debug=True, host='0.0.0.0', port=port)
```

signup.html

```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   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 <div class="navigation">
              <a class="button" href="/ ">
                <img
src="https://as2.ftcdn.net/v2/jpg/01/76/95/97/1000_F_176959719_RuOVaYFGouYlg9o72Wu7PEmHaBrKuE1C.jpg">
```

```
              <div class="logout">LOGOUT</div>
              </a>
            </div>
            <b>&emsp;&ensp;&nbsp;<span style="color:black;font-size: 108%;">System</span></b>
          </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">
          <div class="container px-6 mx-auto grid">
            <h2
              class="my-6 text-2xl font-semibold text-gray-700 dark:text-gray-200"
            >
              Add Sale
            </h2>


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


            </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 cname%}
                  <option style="min-height: 50px;" value='{{i["name"]}}'>{{i["name"]}}----
{{i["id"]}}</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: green;padding: 15px
32px;">Add More</button>
            </div>
            <h4
            class="mb-4 text-lg font-semibold text-gray-600 dark:text-gray-300"
          >
            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">Amount</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">
                              <b>Total(to be Paid)</b>
                            </td>
                            <td class="px-4 py-3 text-sm" style="color:red">
                              <p id="total_amt"></p>
                            </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>
        </br>
          <button type="submit" style="background-color: green;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_amt").innerHTML="<b>"+tot+"</b>";
    }
    </script>
 </body>
</html>
```

b. **GitHub and Project Demo Link**

i. **GitHub Link**

https://github.com/IBM-EPBL/IBM-Project-16257-1659610297

ii. **Video Demo Link**

https://drive.google.com/file/d/1MCwMc_tjI-T94dIOviHtNAb6YQeepOxx/view?usp=sharing