1.INTRODUCTION

1.1 Project Overview

In today's generation, we are engaged in highly computerized technology aiming to make individual lifestyle more comfortable and easier most especially in the world of business. The manual system is now considered the first process after the birth of the computerized system. Online transaction is now very common to widen the target market of the company. It becomes more efficient to the customer considering it can save time and considered hassle-free. The most commonly used system by several companies is the sales system and inventory system creating a web-based system. Advanced system on sale provides more reliable recording of sales of the company with comparison to its actual cost. In addition to the data information needed by the company to decide matters in relation to inventory can be easily generated. Moreover, the inventory control which ensures stocking the in-demand and correct items in the correct quantities. This sales and inventory system can help the company to avoid overstocking. When the company overstock, the fund will not back easy because the cycle of the stock is a stop for unneeded items required time, space, and fund which could have been used on more critical assets. The computer is a useful technology in our society that makes our work easy to wrap-up. Nowadays, computer technology continues to evolve and grow fast. Then every business in our society started shifting from using the manual system to an automated one that makes our work easier and faster. Point of Sales and Inventory System is a computer is an easy way of checking and listing of the sales of the company, it is faster and more reliable rather than doing manually. The system can minimize human errors in editing and be easily access anytime by the company. Sales and inventory system make the company more effective, more productive and is convenient for the company and its customers. The system is making to help the establishment show more relevant items to the customers.

1.1 Purpose

The purpose of the Point of Sales and Inventory system is to serve customers efficiently. This study aims to help the staff/employees of TH Garment Center make their work faster by using the system where they can monitor the remaining products in the records where they can see in the database. The system will provide a good service to the company like better transaction process that brings bigger profit.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

1.Lack of Expertise:

It can be tough to find skilled inventory managers who are adept at the latest technology and can improve inventory strategy. Simply upgrading your inventory management platform with a host of features isn't enough. You need capable management.

2.Poor Communication:

Communication and collaboration are key. When departments are apathetic about sharing information, it makes identifying inventory trends and finding ways to improve much more difficult.

3.Inefficient Processes:

Low-tech, manual inventory management procedures don't seem like a daunting challenge when inventory is small and there's only one warehouse location to manage. But as sales volume increases and inventory expands, inefficient, labor-intensive and low-tech standard operating procedures are difficult to scale.

4.Inadequate Software:

To scale inventory management software to support complex logistics, it needs to integrate with your existing business process platforms. The difficult task is choosing of thousand inventory management system solutions and mastering a host of features that require training and ongoing support.

2.2 References

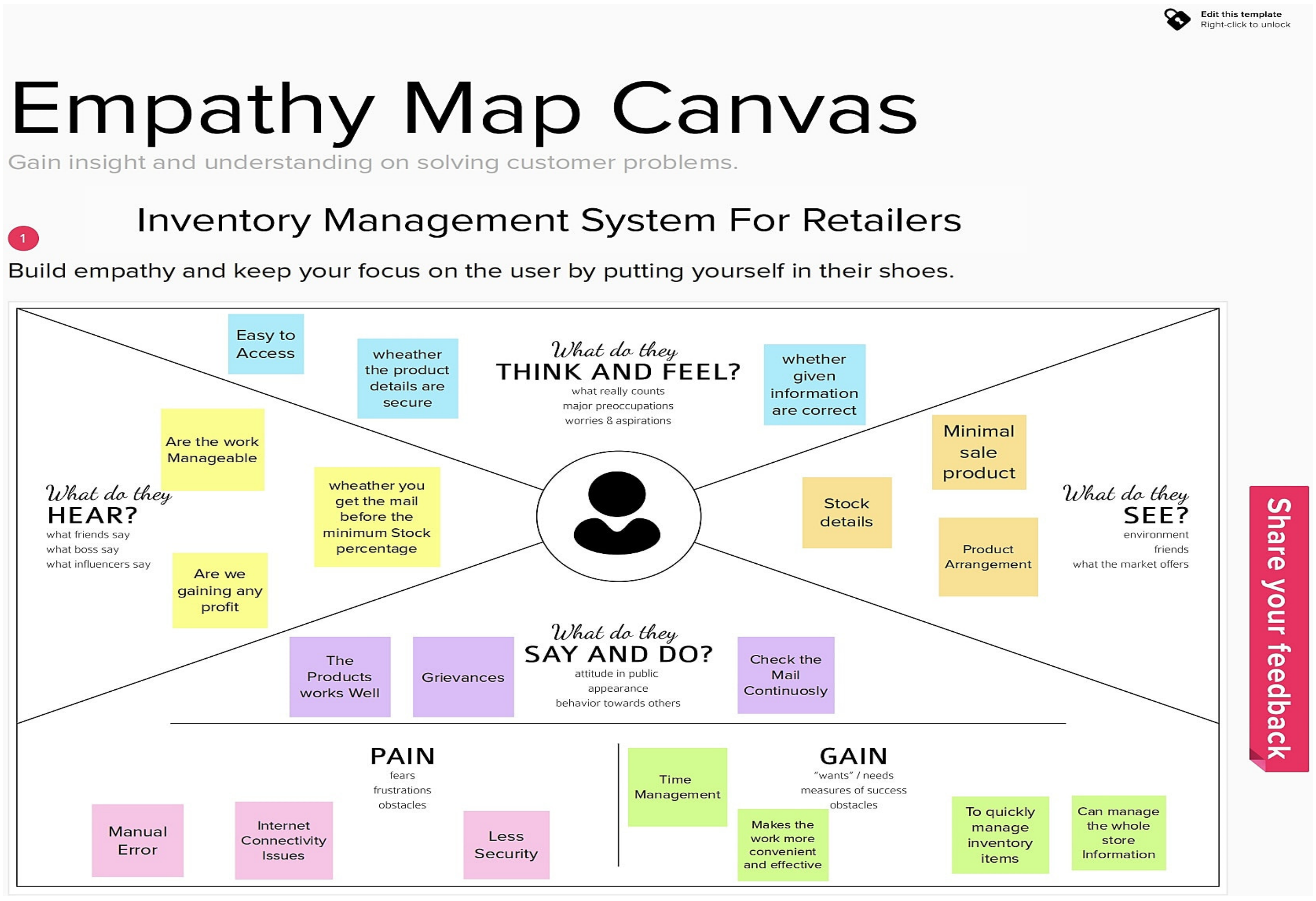
- https://www.netsuite.com/portal/resource/articles/inventory-management/inventory-management-challenges.shtml
- https://www.kapturecrm.com/blog/20-top-challenges-solutions-of-inventory-management

2.3 Problem Statement Definition

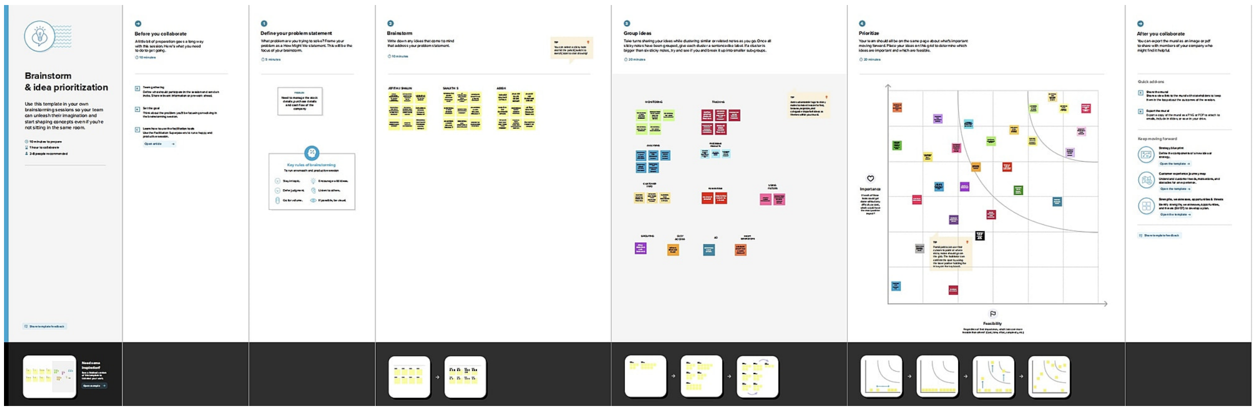
Demand is frequently unpredictable in inventory systems, and lead times can often vary. Managers frequently keep a safety supply to minimize shortages. In such cases, it's difficult to say what order amounts and reorder points will result in the lowest total inventory cost.The inventory issue refers to the general issue of deciding how much inventory to keep on hand in expectation of possible demand. Loss occurs when a business is unable to meet demand (for example, when a store loses sales or when soldiers in a war run out of ammunition) or when commodities are stocked for which there is no demand.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



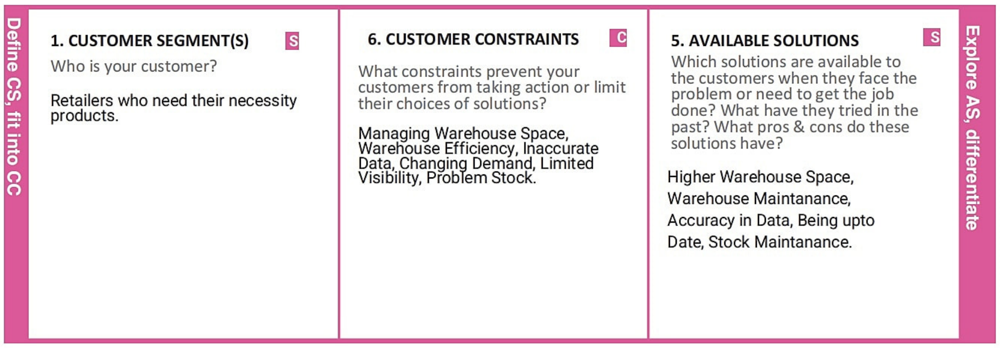
3.3 Proposed Solution

Proposed Solution Template:

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

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Low-tech, manual inventory management procedures don't seem like a daunting challenge when inventory is small and there's only one warehouse location to manage. But as sales volume increases and inventory expands, inefficient, labor-intensive and low-tech standard operating procedures are difficult to scale.
2.	Idea / Solution description	All the information you need about your inventory can be in your pocket. With mobile solutions and cloud-based software, you can control inventory and improve your warehouse productivity from anywhere in the world.
3.	Novelty / Uniqueness	Introduce dashboards with simple interfaces that show real-time inventory data. Having everything on one screen helps remove communication barriers across accounting, sales and warehouse operations.
4.	Social Impact / Customer Satisfaction	Inventory management helps us to maintain customer satisfaction when it comes to product returns.When product is returned because it is damaged or dead on arrival,and it is still under warranty,we can arrange with the manufacturer to do an instant swap of the product to keep the customer happy ,If we are the manufacturer,the we would maintain extra inventory levels that mirror customer return rates to help maintain customer satisfaction.
5.	Business Model (Revenue Model)	With proper inventory management, we spend money on inventory that sells, so cash is always moving through our business.
6.	Scalability of the Solution	To increase the scalability of our business, we uses an automated inventory management system for inventory tracking. This will make our business much more scalable so that we can continue building consistent growth and take advantage of increased sales. And automated inventory management system will give our

3.4 Problem Solution Fit



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) S

Who is your customer?

Retailers who need their necessity products.

6. CUSTOMER CONSTRAINTS C

What constraints prevent your customers from taking action or limit their choices of solutions?

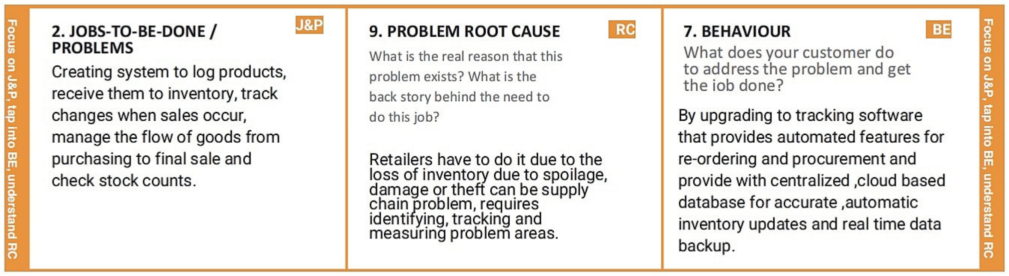
Managing Warehouse Space, Warehouse Efficiency, Inaccurate Data, Changing Demand, Limited Visibility, Problem Stock.

5. AVAILABLE SOLUTIONS S

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?

Higher Warehouse Space, Warehouse Maintanance, Accuracy in Data, Being upto Date, Stock Maintanance.

Explore AS, differentiate



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

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

Creating system to log products, receive them to inventory, track changes when sales occur, manage the flow of goods from purchasing to final sale and check stock counts.

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?

Retailers have to do it due to the loss of inventory due to spoilage, damage or theft can be supply chain problem, requires identifying, tracking and measuring problem areas.

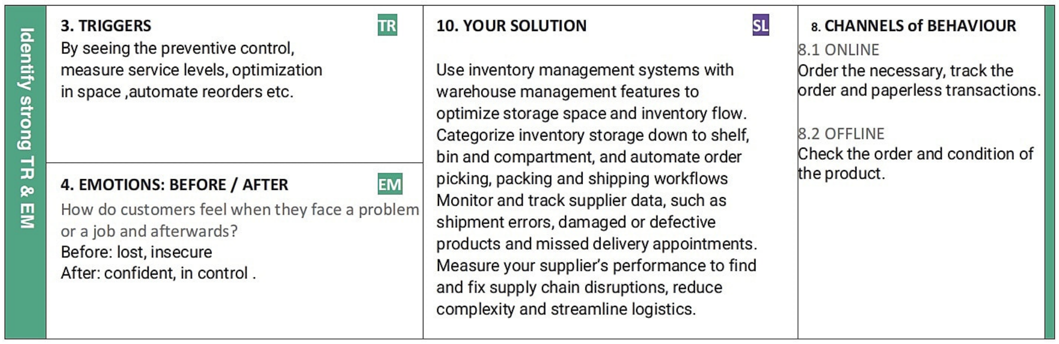
7. BEHAVIOUR BE

What does your customer do to address the problem and get the iob done?

By upgrading to tracking software that provides automated features for re-ordering and procurement and provide with centralized ,cloud based database for accurate ,automatic inventory updates and real time data backup.

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

Identify strong TR & EM



Identify strong TR & EM

3. TRIGGERS TR

By seeing the preventive control, measure service levels, optimization in space ,automate reorders etc.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?

Before: lost, insecure

After: confident, in control .

10. YOUR SOLUTION SL

Use inventory management systems with warehouse management features to optimize storage space and inventory flow. Categorize inventory storage down to shelf, bin and compartment, and automate order picking, packing and shipping workflows Monitor and track supplier data, such as shipment errors, damaged or defective products and missed delivery appointments. Measure your supplier's performance to find and fix supply chain disruptions, reduce complexity and streamline logistics.

8. CHANNELS of BEHAVIOUR

8.1 ONLINE

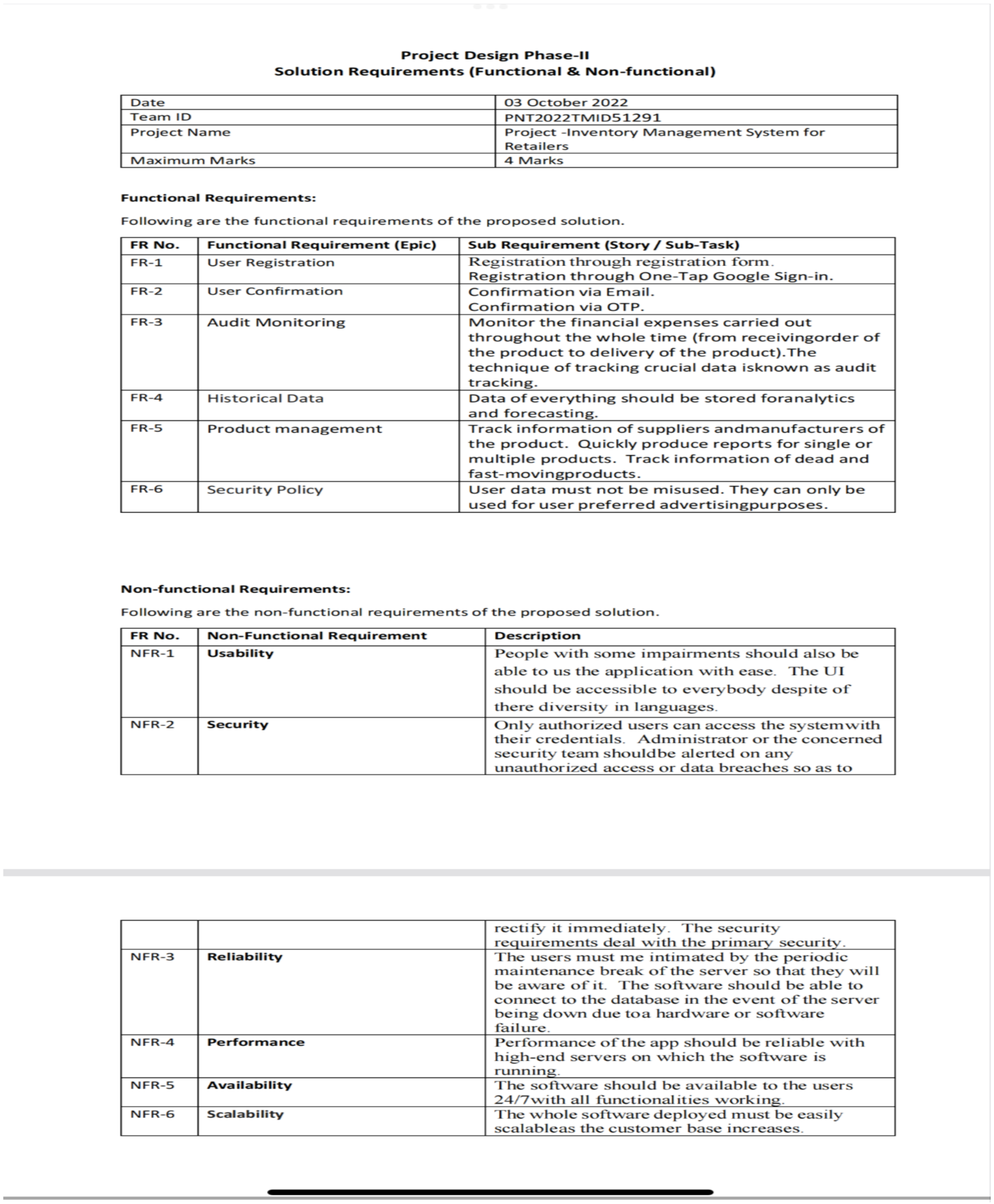
Order the necessary, track the order and paperless transactions.

8.2 OFFLINE

Check the order and condition of the product.

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS



Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	03 October 2022
Team ID	PNT2022TMID51291
Project Name	Project -Inventory Management System for Retailers
Maximum Marks	4 Marks

Functional Requirements:

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 registration form. Registration through One-Tap Google Sign-in.
FR-2	User Confirmation	Confirmation via Email. Confirmation via OTP.
FR-3	Audit Monitoring	Monitor the financial expenses carried out throughout the whole time (from receivingorder of the product to delivery of the product).The technique of tracking crucial data isknown as audit tracking.
FR-4	Historical Data	Data of everything should be stored foranalytics and forecasting.
FR-5	Product management	Track information of suppliers andmanufacturers of the product. Quickly produce reports for single or multiple products. Track information of dead and fast-movingproducts.
FR-6	Security Policy	User data must not be misused. They can only be used for user preferred advertisingpurposes.

Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	People with some impairments should also be able to us the application with ease. The UI should be accessible to everybody despite of there diversity in languages.
NFR-2	**Security**	Only authorized users can access the systemwith their credentials. Administrator or the concerned security team shouldbe alerted on any unauthorized access or data breaches so as to rectify it immediately. The security requirements deal with the primary security.
NFR-3	**Reliability**	The users must me intimated by the periodic maintenance break of the server so that they will be aware of it. The software should be able to connect to the database in the event of the server being down due toa hardware or software failure.
NFR-4	**Performance**	Performance of the app should be reliable with high-end servers on which the software is running.
NFR-5	**Availability**	The software should be available to the users 24/7with all functionalities working.
NFR-6	**Scalability**	The whole software deployed must be easily scalableas the customer base increases.

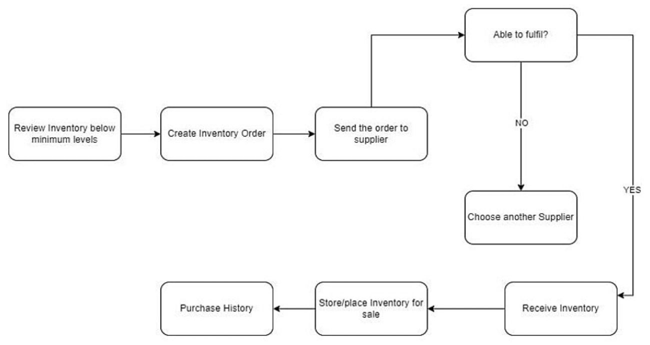
5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

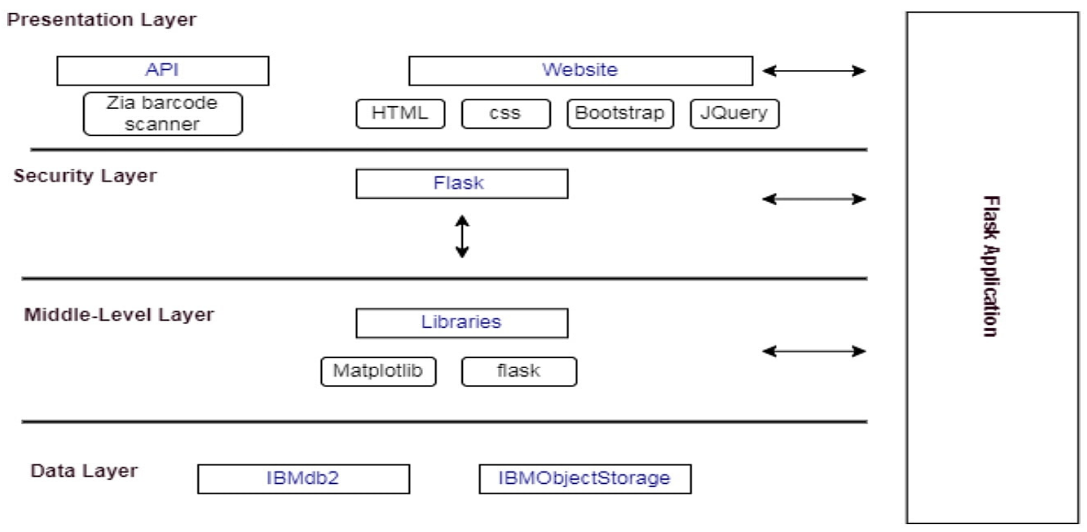
Data Flow Diagrams:

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

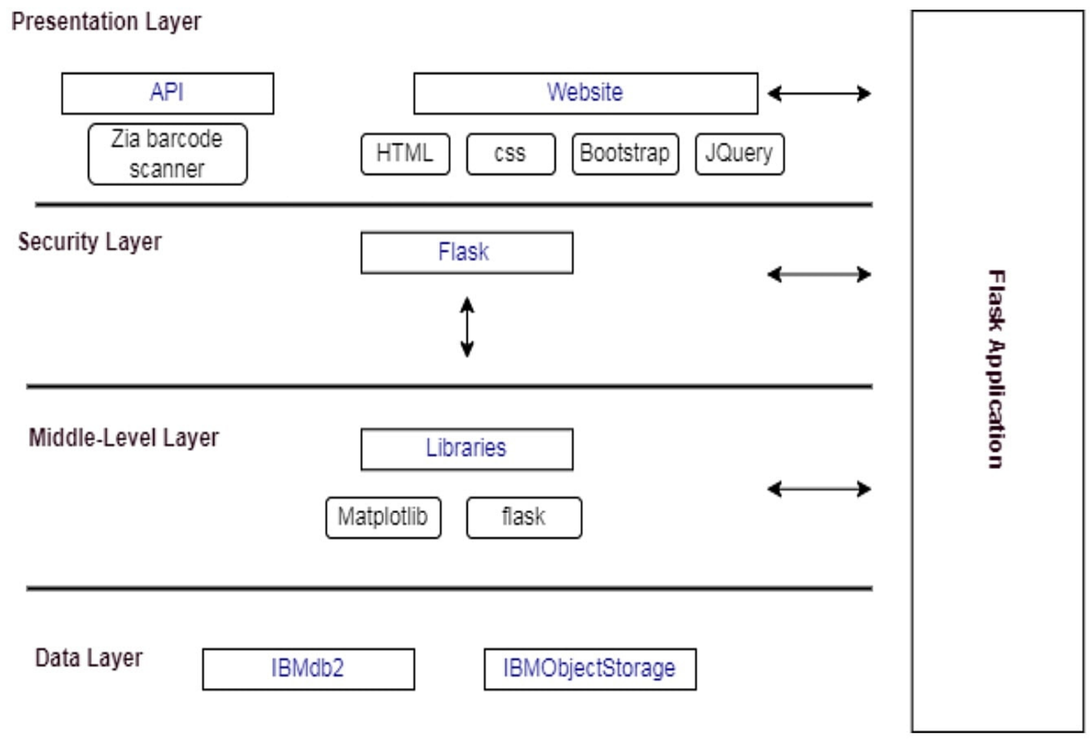
DATA FLOW DIAGRAM



5.2 SOLUTION & TECHNICAL ARCHITECTURE

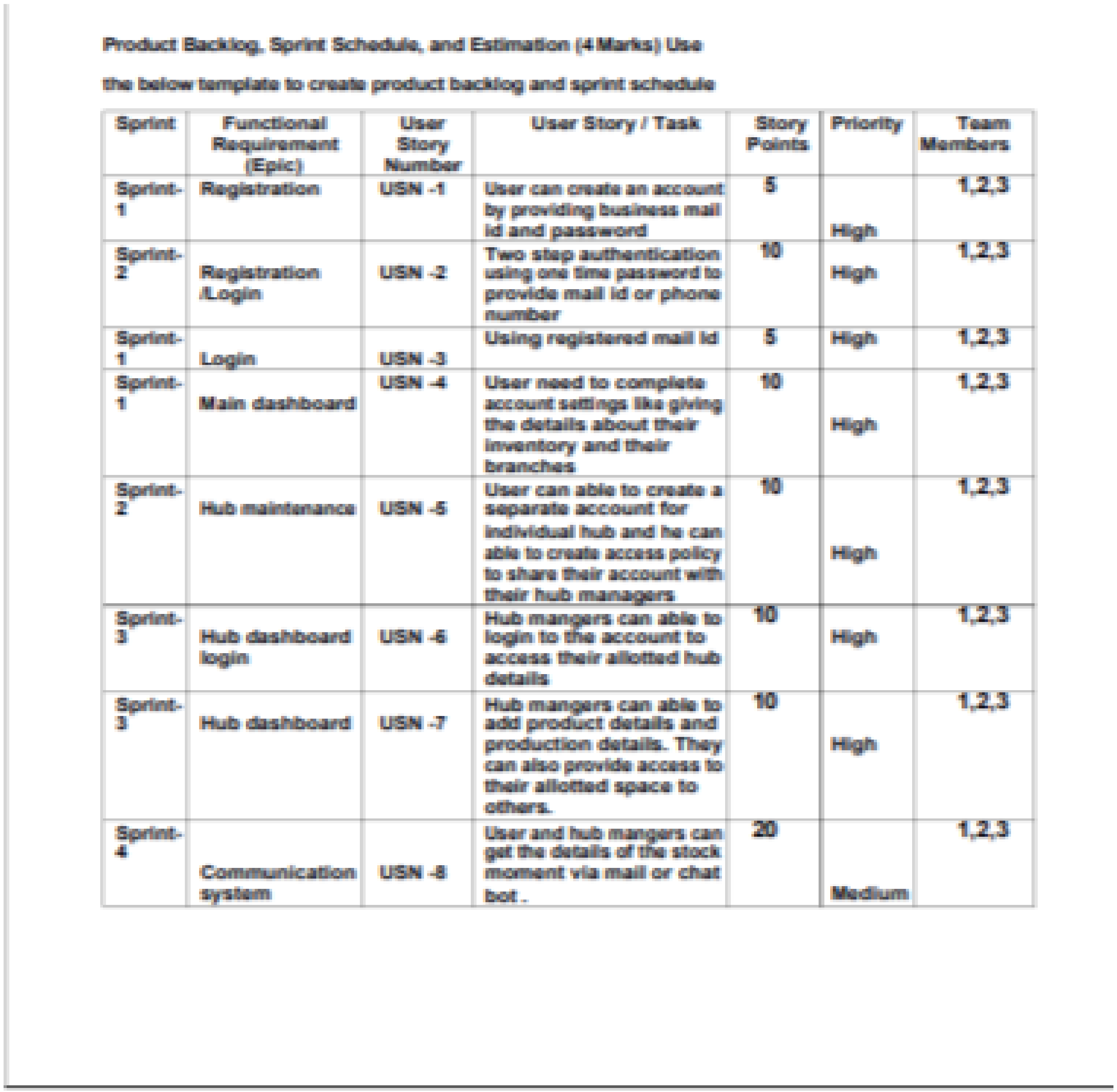


5.3 USER STORIES



6. PROJECT PLANNING AND SCHEDULE

6.1.1 Sprint Planning & Estimation



Product Backlog, Sprint Schedule, and Estimation (4 Marks) Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN -1	User can create an account by providing business mail id and password	5	High	1,2,3
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
Sprint-1	Login	USN -3	Using registered mail id	5	High	1,2,3
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
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
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
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
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

7.1 CODING & SOLUTIONING

```
from flask import Flask, render_template, request, redirect, url_for
import sqlite3 as sql
app = Flask(__name__)

#db connection
con = sql.connect("database.db",check_same_thread=False)
con.row_factory = sql.Row
cur = con.cursor()

# Dynamic Functions
def getData(tableName,locationName,productName):
    if locationName is None and productName is None:
        queryData = cur.execute("SELECT * FROM
{}".format(tableName)).fetchall()
    else:
        queryData = cur.execute("SELECT * FROM {} WHERE locationName = ? AND
productName = ? ".format(tableName),(locationName,productName)).fetchall()
    return queryData

def insertData(tableName,newData):
    try:
        if tableName == "products":
            cur.execute("INSERT INTO products (productName)VALUES
(?)",(newData['productName'],) )
            con.commit()
            msg = "Product Added Successfully"
        if tableName == "location":
            cur.execute("INSERT INTO location (locationName)VALUES
(?)",(newData['locationName'],) )
            con.commit()
            msg = "Location Added Successfully"
        if tableName == "productmovement":
            cur.execute("INSERT INTO productmovement (atTime, from_location,
to_location, productName, qty)VALUES
(?,?,?,?,?)",(newData['atTime'],newData['from_location'],newData['to_locat
ion'],newData['productName'],newData['qty']))
            con.commit()
            msg = "Movement Added Successfully"
        if tableName == "balance":
            cur.execute("INSERT INTO balance (locationName, productName,
qty)VALUES (?,?,?)",(newData["locationName"],newData["productName"],0))
            con.commit()
            msg = "Entries of balance updated"
        return msg
    except:
        con.rollback()
        msg = "Error in insert operation"
        return msg

def updateData(tableName,newData):
    try:
        if tableName == "products":
            cur.execute("UPDATE products SET productName = ? WHERE productID
= ?",(newData['NEWProductName'],newData['ProductID']) )
            con.commit()
            msg = "Product Edited Successfully"
        if tableName == "location":
            cur.execute("UPDATE location SET locationName = ? WHERE
locationID = ?",(newData['NEWLocationName'],newData['LocationID']) )
            con.commit()
            msg = "Location Edited Successfully"
        if tableName == "productmovement":
            cur.execute("UPDATE productmovement SET qty = ? WHERE movementID
= ?",(newData['editedqty'], newData['movementID']))
            con.commit()
            msg = "Movement Edited Successfully"
        if tableName == "balance":
            cur.execute("UPDATE balance SET qty = ?  WHERE locationName = ?
AND productName =
?",(newData['qty'],newData['locationName'],newData['productName']) )
```

```
         con.commit()
         msg = "Balance Edited Successfully"
      return msg

   except:
      con.rollback()
      msg = "Error in update operation"
      return msg

def deleteData(tableName,id):
   try:
      if tableName == "products":
         cur.execute("DELETE FROM products WHERE productID = ?",(id))
         con.commit()
         msg = "Product Deleted Successfully"
      if tableName == "location":
         cur.execute("DELETE FROM location WHERE locationID = ?",(id))
         con.commit()
         msg = "Location Deleted Successfully"
      if tableName == "productmovement":
         cur.execute("DELETE FROM productmovement WHERE movementID =
?",(id))
         con.commit()
         msg = "Movement Deleted Successfully"
      return msg
   except:
      con.rollback()
      msg = "Error in delete operation"
      return msg

#function that updates balance table
def movementManager(locationName,productName,qty):
   oldQuantity = 0
   newQuantity = 0

   if locationName !="-":
        balanceRows = getData("balance",locationName,productName)

        #update in balance table if entry exists
        if len(balanceRows) != 0:
            for br in balanceRows:
                oldQuantity = int(br["qty"])

            newQuantity = oldQuantity + qty
            if(newQuantity < 0):
                status = "Insuffecient Quantity In "+locationName

            else:
                newData = {}
                newData['qty'] = newQuantity
                newData['locationName'] = locationName
                newData['productName'] = productName
                updateData("balance",newData)
                status = "Balance Operation Successfull"
    else:
        status = "Balance Operation Successfull"

    return status

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

#homepage
@app.route('/home')
def home():
    #initalize empty list
    data = []

    productRows = getData("products",locationName = None, productName =
None)
```

```
    locationRows = getData("location",locationName = None, productName =
None)

    for lr in locationRows:
        for pr in productRows:
            balRow = getData("balance",lr["locationName"],pr["productName"])

            if len(balRow) != 0:
                for br in balRow:
                    qty = int(br["qty"])

                innerData = {}
                innerData['locationName'] = lr["locationName"]
                innerData['productName'] = pr["productName"]
                innerData['qty'] = qty

                #push dict to array
                data.append(innerData.copy())
    return render_template('home.html', data = data, locationRows =
locationRows)

#________________________________Product
Functions________________________________

#product management page
@app.route('/productM')
def productM():
    rows = getData("products",locationName = None, productName = None)
    return render_template('productM.html',rows = rows)

#add new product
@app.route('/addProduct',methods = ['POST'])
def addProduct():
    if request.method == 'POST':
        newData = {}
        newData['productName'] = request.form['pm']
```

```
      msg = insertData("products",newData)
      return redirect(url_for('productM')+"?msg="+msg)

#edit product
@app.route('/editProduct',methods = ['POST'])
def editProduct():
   if request.method == 'POST':
      newData = {}
      newData['ProductID'] = request.form['ProductID']
      newData['NEWProductName'] = request.form['NEWProductName']
      msg = updateData("products",newData)

      return redirect(url_for('productM')+"?msg="+msg)

#delete product
@app.route('/deleteProduct/<productID>')
def deleteProduct(productID):
   msg = deleteData("products",productID)
   return redirect(url_for('productM')+"?msg="+msg)


#_______________________________Location
Functions_______________________________________

#location management page
@app.route('/locationM')
def locationM():
   rows = getData("location",locationName = None, productName = None)
   return render_template('locationM.html',rows = rows)

#add new location
@app.route('/addLocation',methods = ['POST'])
def addLocation():
   if request.method == 'POST':
      newData = {}
      newData['locationName'] = request.form['lm']
```

```
        msg = insertData("location",newData)
        return redirect(url_for('locationM')+"?msg="+msg)

#edit location
@app.route('/editLocation',methods = ['POST'])
def editLocation():
    if request.method == 'POST':
        newData = {}
        newData['LocationID'] = request.form['LocationID']
        newData['NEWLocationName'] = request.form['NEWLocationName']
        msg = updateData("location",newData)

        return redirect(url_for('locationM')+"?msg="+msg)

#delete location
@app.route('/deleteLocation/<locationID>')
def deleteLocation(locationID):
    msg = deleteData("location",locationID)
    return redirect(url_for('locationM')+"?msg="+msg)

#_______________________________Movement
Functions___________________________________________

#movement management page
@app.route('/movementM')
def movementM():
    rows = getData("productmovement",locationName = None, productName =
None)
    productRows = getData("products",locationName = None, productName =
None)
    locationRows = getData("location",locationName = None, productName =
None)

    #check if all posiblites of entries exists in balance products
    for pr in productRows:
        for lr in locationRows:
         data =
getData("balance",locationName=lr["locationName"],productName =
pr["productName"])

         if len(data) == 0:
            newData = {}
            newData['locationName'] = lr["locationName"]
            newData['productName'] = pr["productName"]
            newData['qty'] = 0
            insertData("balance",newData)

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

#add new movement
@app.route('/addMovement',methods = ['POST'])
def addMovement():
   if request.method == 'POST':
      newData = {}

      newData['atTime'] = request.form['atTime']
      newData['from_location'] = request.form['from_location']
      newData['to_location'] = request.form['to_location']
      newData['productName'] = request.form['productName']
      newData['qty'] = int(request.form['qty'])

      #reduce from_location
      status1 = movementManager(newData['from_location'],
newData['productName'] ,-(newData['qty']))
      #add to_location
      status2 = movementManager(newData['to_location'],
newData['productName'] ,(newData['qty']))

      #only add movement if it has quantity left
      if status1 == "Balance Operation Successfull" and status2 ==
"Balance Operation Successfull" :
```

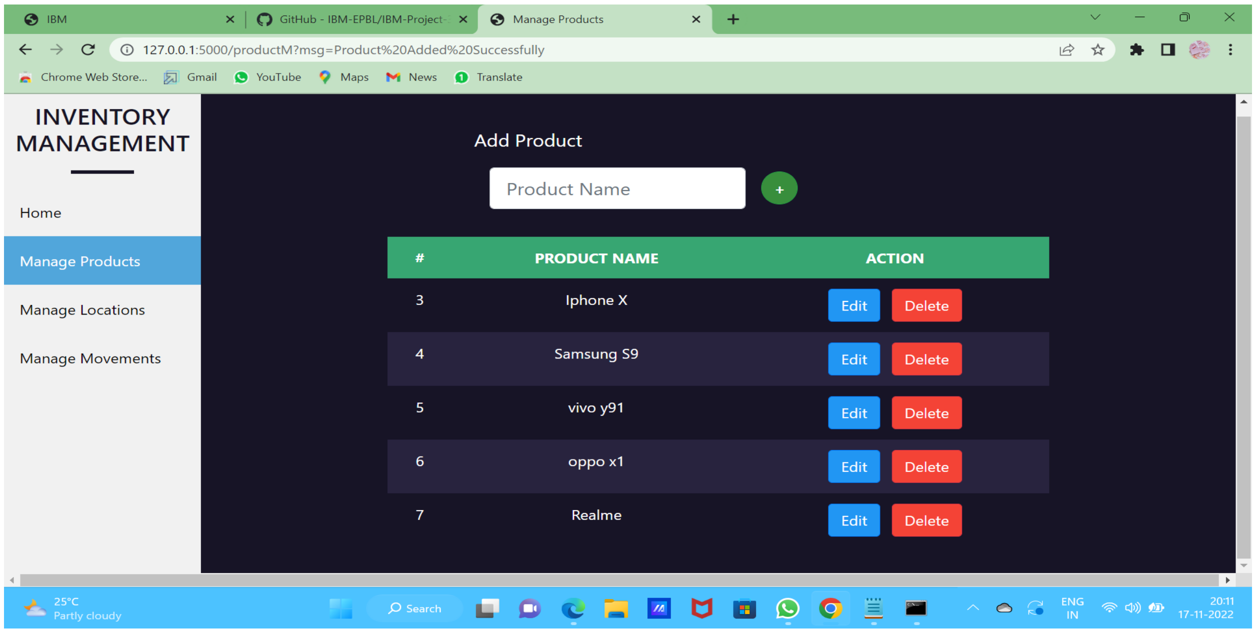
```
            msg = insertData("productmovement",newData)
        else:
            msg = "Insuffecient Quantity In "+newData['from_location']

    return redirect(url_for('movementM')+"?msg="+msg)

#add new movement
@app.route('/editMovement',methods = ['POST'])
def editMovement():

    if request.method == 'POST':
        from_location = request.form['from_location']
        to_location = request.form['to_location']
        productName = request.form['productName']
        qty = int(request.form['qty'])
        editedqty = int(request.form['editedqty'])

        newData = {}

        newData['movementID'] = request.form['movementID']
        newData['editedqty'] = request.form['editedqty']


        #reduce old quantity
        status1 = movementManager(from_location, productName ,qty)
        status2 = movementManager(to_location, productName ,-qty)

        #add new quantity
        status1 = movementManager(from_location, productName ,-editedqty)
        status2 = movementManager(to_location, productName ,editedqty)

        if status1 == "Balance Operation Successfull" and status2 ==
"Balance Operation Successfull" :
            msg = updateData("productmovement",newData)
        else:

            #reduce new quantity
            status1 = movementManager(from_location, productName ,-editedqty)
            status2 = movementManager(to_location, productName ,-editedqty)

            #add old quantity
            status1 = movementManager(from_location, productName ,-qty)
            status2 = movementManager(to_location, productName ,qty)

            msg = "Insuffecient Quantity In "+from_location

    return redirect(url_for('movementM')+"?msg="+msg)

#delete movement
@app.route('/deleteMovement',methods = ['POST'])
def deleteMovement():
    if request.method == 'POST':
        movementID = request.form['movementID']
        from_location = request.form['from_location']
        to_location = request.form['to_location']
        productName = request.form['productName']
        qty = int(request.form['qty'])

        #add back to from_location
        status1 = movementManager(from_location, productName ,qty)
        #sub from to_location
        status2 = movementManager(to_location, productName ,-qty)

        #only add movement if it has quantity left
        if status1 == "Balance Operation Successfull" and status2 ==
"Balance Operation Successfull" :
            msg = deleteData("productmovement",movementID)
        else:
            msg = "Insuffecient Quantity In "+from_location
        return redirect(url_for('movementM')+"?msg="+msg)

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

8. TESTING

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

9. RESULTS

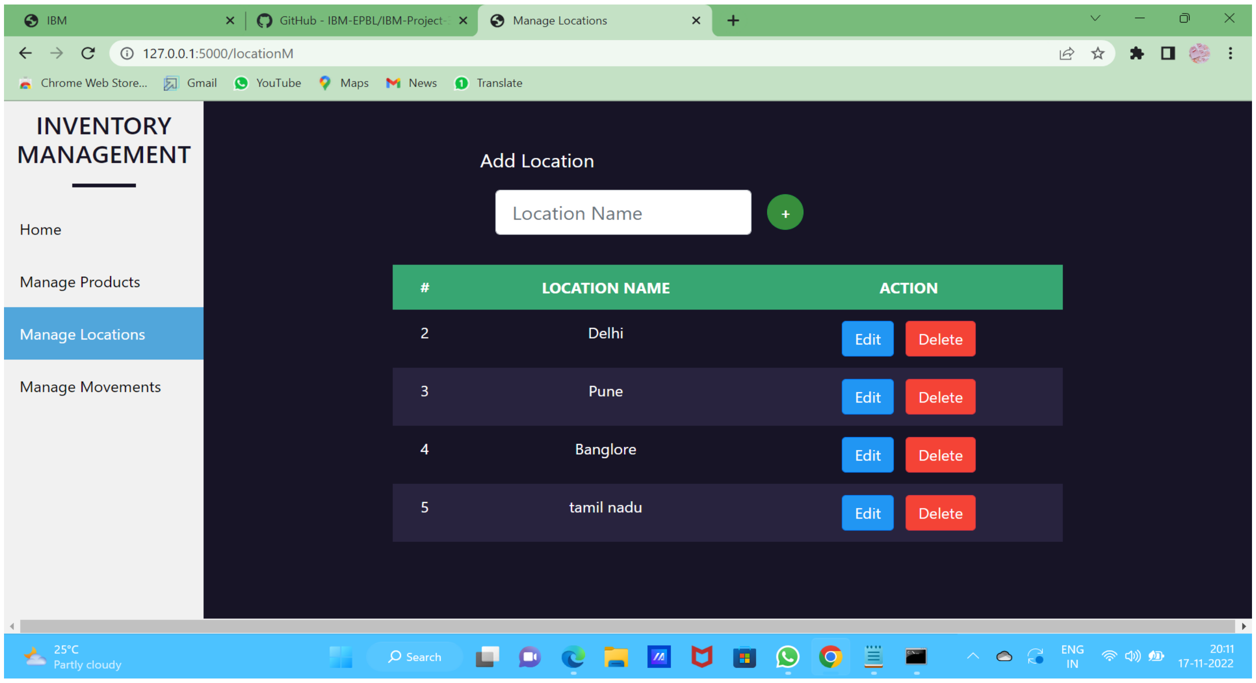


INVENTORY MANAGEMENT

Add Product

#	PRODUCT NAME	ACTION
3	Iphone X	Edit Delete
4	Samsung S9	Edit Delete
5	vivo y91	Edit Delete
6	oppo x1	Edit Delete
7	Realme	Edit Delete

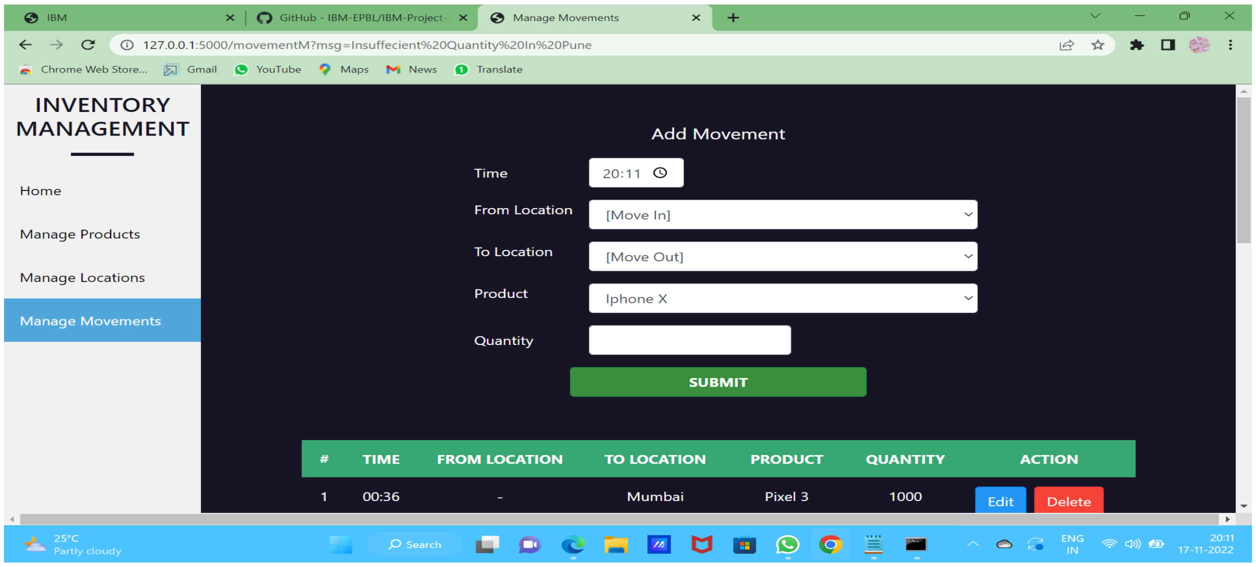
IBM
GitHub - IBM-EPBL/IBM-Project-
Manage Products
127.0.0.1:5000/productM?msg=Product%20Added%20Successfully
Home
Manage Products
Manage Locations
Manage Movements
Product Name
+



INVENTORY MANAGEMENT

Add Location

#	LOCATION NAME	ACTION
2	Delhi	Edit Delete
3	Pune	Edit Delete
4	Banglore	Edit Delete
5	tamil nadu	Edit Delete


127.0.0.1:5000/movementM?msg=Insuffecient%20Quantity%20In%20Pune

INVENTORY MANAGEMENT

Home

Manage Products

Manage Locations

Manage Movements

Add Movement

Time	20:11
From Location	[Move In]
To Location	[Move Out]
Product	Iphone X
Quantity	

SUBMIT

#	TIME	FROM LOCATION	TO LOCATION	PRODUCT	QUANTITY	ACTION
1	00:36	-	Mumbai	Pixel 3	1000	Edit Delete

10. ADVANTAGES AND DISADVANTAGES

Advantages of Inventory Management System Project

There are many advantages of the inventory management system. Thus, summarized below which can avoid the company from suffering from big economical losses and other problems that may occur during the everyday operations of the firm that can be observed as the materials being out of stock or machine failures and many other operations happenings on a day-to-day basis.

Disadvantage: System Crash

One of the biggest problems with any computerized system is the potential for a system crash. A corrupt hard drive, power outages and other technical issues can result in the loss of needed data. At the least, businesses are interrupted when they are unable to access data they need. Business owners should back up data regularly to protect against data loss.

11. CONCLUSION

Inventory management is a very complex but essential part of the supply chain. An effective inventory management system helps to reduce stock-related costs such as warehousing, carrying, and ordering costs.As you can see the importance of inventory management is very serious, it is one of the most important aspects of any business. The aspect of this part of the business is whether or not you can satisfy the demand of your customers if you aren't sure if you have all the materials availableto make the final product (Thibodeaux, 2014). Without Wheeled Coach©having the proper inventory management they would not be able to supply their customers with their ordered ambulance.

12. FUTURE SCOPE

The scope of an inventory system can cover many needs, including valuing the inventory, measuring the change in inventory and planning for future inventory levels. The value of the inventory at the end of each period provides a basis for financial reporting on the balance sheet. Measuring the change in inventory allows the company to determine the cost of inventory sold during the period. This allows the company to plan for future inventory needs.

13. APPENDIX

Github link-https://github.com/IBM-EPBL/IBM-Project-40109-1660623512

Project Demo Link-

https://drive.google.com/file/d/1d3Wl5ywNnTx73NHLFgMXl7VqZCKgfPaA/view?usp=drivesdk