# INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

## 1 INTRODUCTION

### Project overview

Inventory management is essentially a variety of techniques, tools and technologies that a business uses to manage and control their inventory. The way that it's utilized and implemented ranges from simple right through to complex. It depends on the needs and scope of the business and the capabilities and functionality of the management software used.

The following provides information about how the Inventory Management system integrates with general accounting and other distribution systems.

- General Accounting
- Inventory Management
- Bulk Stock Control
- Procurement
- Sales Order Management
- Address Book

The Inventory Management system stores item information for the Sales Order Management, Procurement, and manufacturing systems. It also stores sales and purchasing costs and quantities available by location and places holds on locations from which you do not sell items.

You update the general ledger inventory account balances with any change in inventory valuation, count variances, or movement.

### Purpose

The main purpose of inventory management is to help businesses easily and Efficiently manage the ordering, stocking, storing, and using of inventory. By effectively managing your inventory ,you'll always know what items are in stock, how many of them there are, and where they are located.

Plus, practicing strong inventory management allows you to understand how you use your inventory–and how demand changes for it–over time. You can zero in on exactly what you need, what's not so important, and what's just a waste of money. That's using inventory management to practice inventory control. By the way, inventory control is the balancing act of always having enough stock to meet demand, while spending as little as possible on ordering and carrying inventory.

# 2 LITERATURE SURVEY

## Existing problem

When your inventory becomes hard to find, you have inventory visibility problems. Lack of visibility is one of the most common inventory management problems. Locating the correct item in the right place as quickly as possible is essential to inventory. If the hard to find inventory is part of the supply chain for manufacturing, it can impact the operations of the entire manufacturing process. If the inventory stock is being accessed for shipping and cannot be located, it leads to ncomplete or wrong shipments and severely impacts customer satisfaction. Either way inventory visibility problems have a severe impact on the performance of the business and is one of the symptoms of poor inventory management.

Not Measuring Your Business's Performance

Being able to measure various parameters, such as the amount of stock, customer satisfaction ratings, working capital, and sale cycles can tell you much about your business. Yet, you can't do that without high-powered reporting software.

Putting Inventory in the Wrong Spot

When you don't have a way to manage your inventory, items will be placed in the wrong spot. When this happens, the wrong items could be pulled for shipments. The supply chain gets disrupted. Customers are upset. Therefore, inventory needs to be put in its proper place every single time

## References


https://www.visual-paradigm.com/scrum/scrum-burndown-chart/

https://www.atlassian.com/agile/tutorials/burndown-charts

Reference: https://www.atlassian.com/agile/project-management

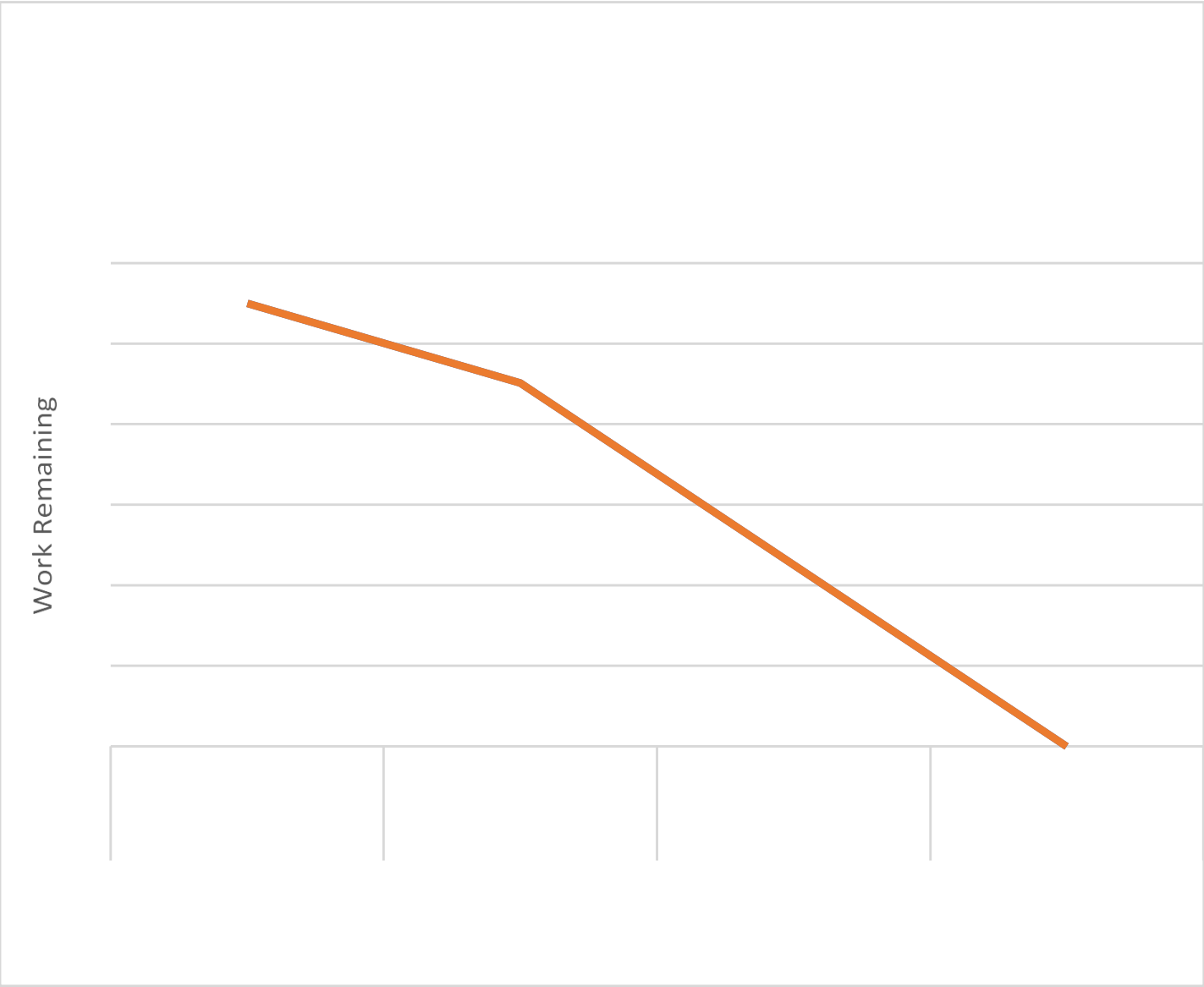
https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-_software

https://www.atlassian.com/agile/tutorials/epics_https://www.atlassian.com/agile/tutorials/sprints

https://www.atlassian.com/agile/project-management/estimation

https://www.atlassian.com/agile/tutorials/burndown-charts

https://careereducation.smartinternz.com/Student/guided_project_workspa_ce/47838



## Problem Statement Definition

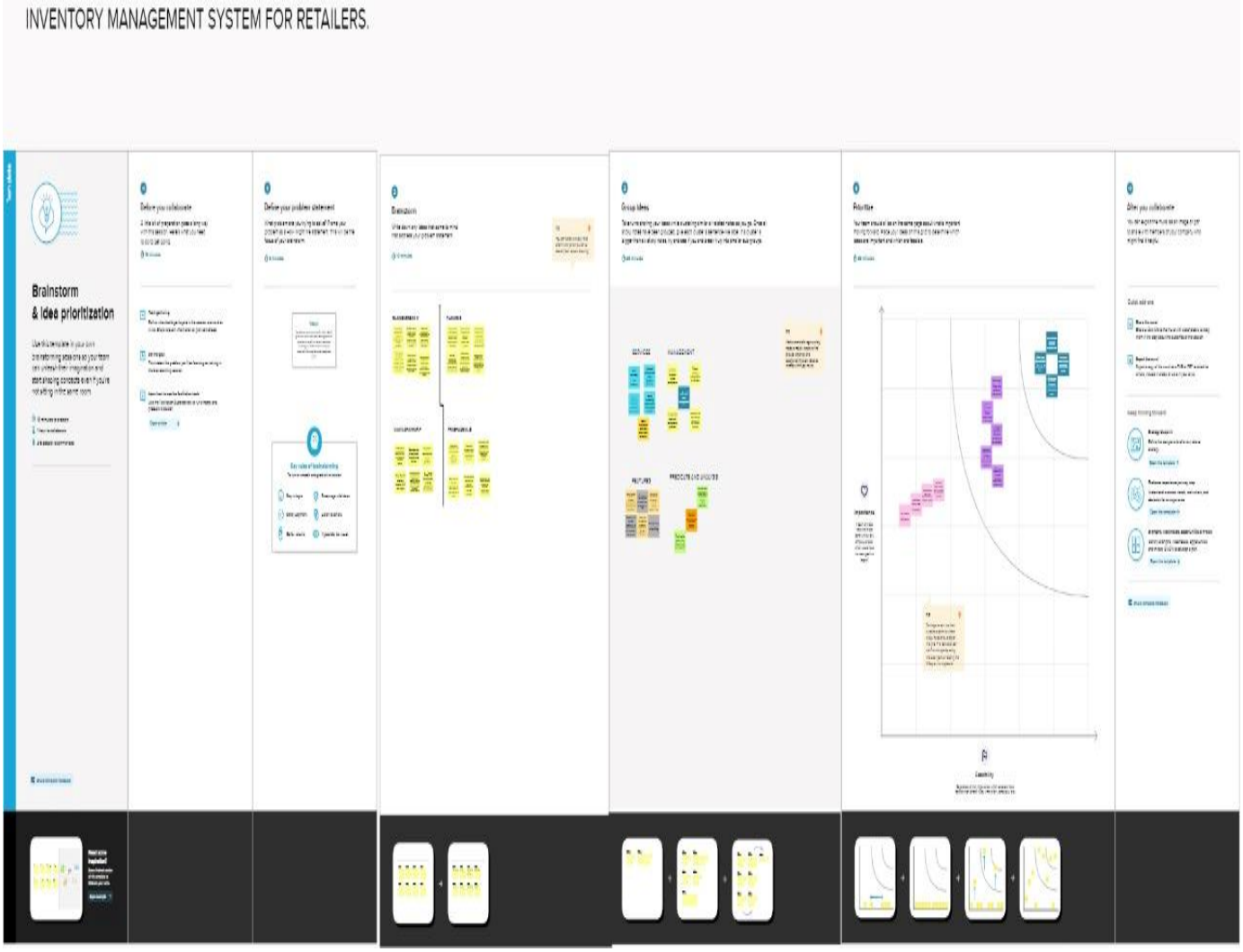
| | |
|---|---|
| Who does the problem affect? | The retailers who are all have the stocks and the size /quantity of the stock is not matter. They can do large scale or small-scale business. |
| What is the issue? | Inventory reports are essential to making decisions. An inventory department cannot summarise and report based on real-time inventory data when using a manual system. Reports on historical trends are also challenging to prepare quickly. When management cannot visualise inventory stock or trends, making informed decisions on purchase and inventory becomes tough. This directly affects the bottom line of the company. |
| When does the issue occur? | Poor planning may lead to late orders. Unaware of stocks details cause miss lot of customer and struggle to face seasonal/festival sales. Poor tracking may lead to back orders. |
| Where is the issue coming? | Human can make mistake. The perfection level is their interest and consciousness. Lack of interest and consciousness. The counts/ calculations of stock are beyond human power. |
| Why is it important that we fix the problem? | Nearly 81% of consumers experienced an "out-of-stock" situation in the past 12 months, resulting in lost sales for retailers and lots of disappointment for in-store shoppers. Globally, retailers recorded losses of a whopping $1.75 trillion due to mismanaged inventory. Therefore considering the economic crisis of the retailers and to reduce the manpower efficiently while handling data, it is very important to have a best inventory management system for retailers |

.

# 3 IDEATION & PROPOSED SOLUTION

Empathy Map Canvas

## Ideation & Brainstorming



## Proposed Solution

## Proposed Solution

| S.NO | Parameter | Description |
|---|---|---|
| 1. | Problem Statement (Problem to be solved) | Lot of stocks can handle is too difficult when we use traditional method for inventory. Sometimes calculation is wrong or take more times. |
| 2. | Idea / Solution description | Creating a Inventory Management System as a web application which maintain and manage the stock of the retailers. |
| 3. | Novelty / Uniqueness | Predict the demand of stocks and suggestion to spend less money to buy new stocks. |
| 4. | Social Impact / Customer Satisfaction | Customer feedback will be collected and rated their satisfaction. Easily identify the stocks which are most liked by customers. |
| 5. | Business Model (Revenue Model) | By the help of high demand products details retailers can order for more supply |
| 6. | Scalability of the Solution | The system can handle large scale business also .The stock information are very perfect and collaborate with multiple retailers. |

# Problem Solution fit

# 4 REQUIREMENT ANALYSIS

## Functional requirement

Following are the functional requirements of the proposed solution.

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

| | |
|---|---|
| Team Members | Silambarasan V<br>Thavasi S<br>Vijaya Shankar P<br>Prabudeva P |
| Team ID | PNT2022TMID13832 |
| Project Name | Inventory Management System for Retailers |
| Maximum Marks | 4 Marks |

**Functional Requirements:**

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|---|---|---|
| FR-1 | User Registration | Registration through Form<br>Registration through Gmail |
| FR-2 | User Confirmation | Confirmation via Email<br>Confirmation via OTP |
| FR-3 | User Login | Login with username<br>Login with Email<br>Login with password |
| FR-4 | Dashboard | View product availability, name of the product, stock keep unit, brand, retail price, product category, lot number, expiration date, vendor information, wholesale cost, etc., |
| FR-5 | Identification of the stock location | Provide number label for - shelf, rack, and boxes |
| FR-6 | Periodical stock checking | Automate the tracking of stock count |
| FR-7 | Purchase management and Forecasting | Order review and placement, avoid risk stock, review product, priorities purchases based on an item's profitability, popularity, and lead time. |
| FR-8 | Returns Management System | Examine for flaws or damage and return to the vendor if necessary. Add it to inventory counts if it is sellable. |

## Non-functional Requirements:

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

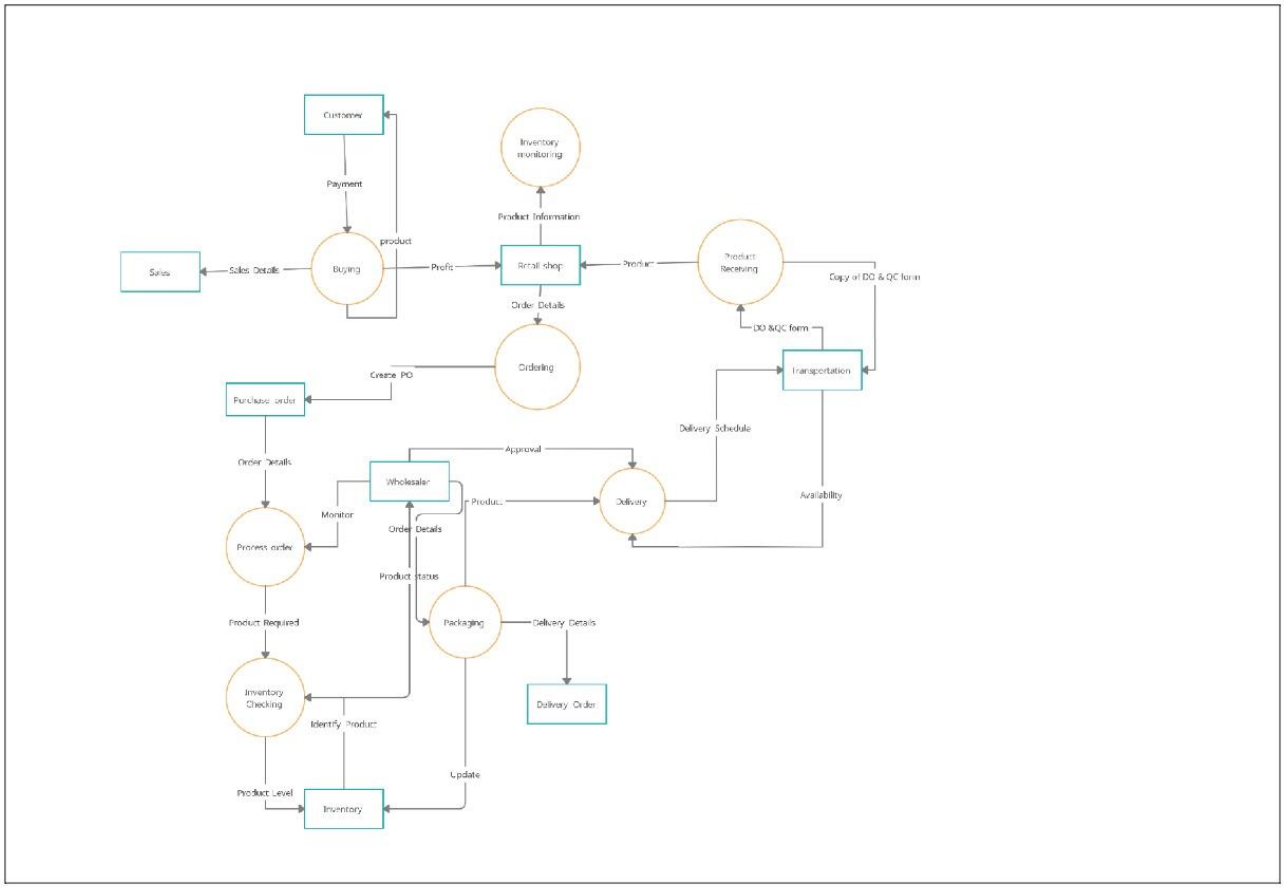
| | | |
|---|---|---|
| FR-9 | Markdown and promotion | Display product savings, keep enough inventory on hand to satisfy demand. |
| FR-10 | Calculating the death stock | Return the stock to the vendor for credits |

**Non-functional Requirements:**

| FR No. | Non-Functional Requirement | Description |
|---|---|---|
| NFR-1 | **Usability** | The success of deploying an inventory monitoring system in your company depends on usability. The system must be user-friendly and straightforward in the manner it presents all pertinent data and linkages, and its menus must include buttons that are simple to grasp. The software is not worthwhile if training takes hours for your workforce. Remember to pick a solution that makes inventory management simple. This variant is compatible with desktop browsers. |
| NFR-2 | **Security** | It is the method for making sure that kept goods are safe and under the best management control. It is crucial for effective warehouse management because a warehouse's productivity and safety determine how well a firm performs. In this case, only authorized people with their username and password can access the system. |
| NFR-3 | **Reliability** | The user must constantly receive accurate inventory status from the system. By routinely comparing the real levels to the levels shown in the system, any errors are fixed. |
| NFR-4 | **Performance** | Every time a user requests a process, the system must successfully complete the tasks like updating the stocks in the database, adding new stocks, and deleting it. Every time the system is turned on, all its features must be accessible to the customer. The system's calculations must adhere to the standards established by the customer and should not change until the customer specifically requests a change. |

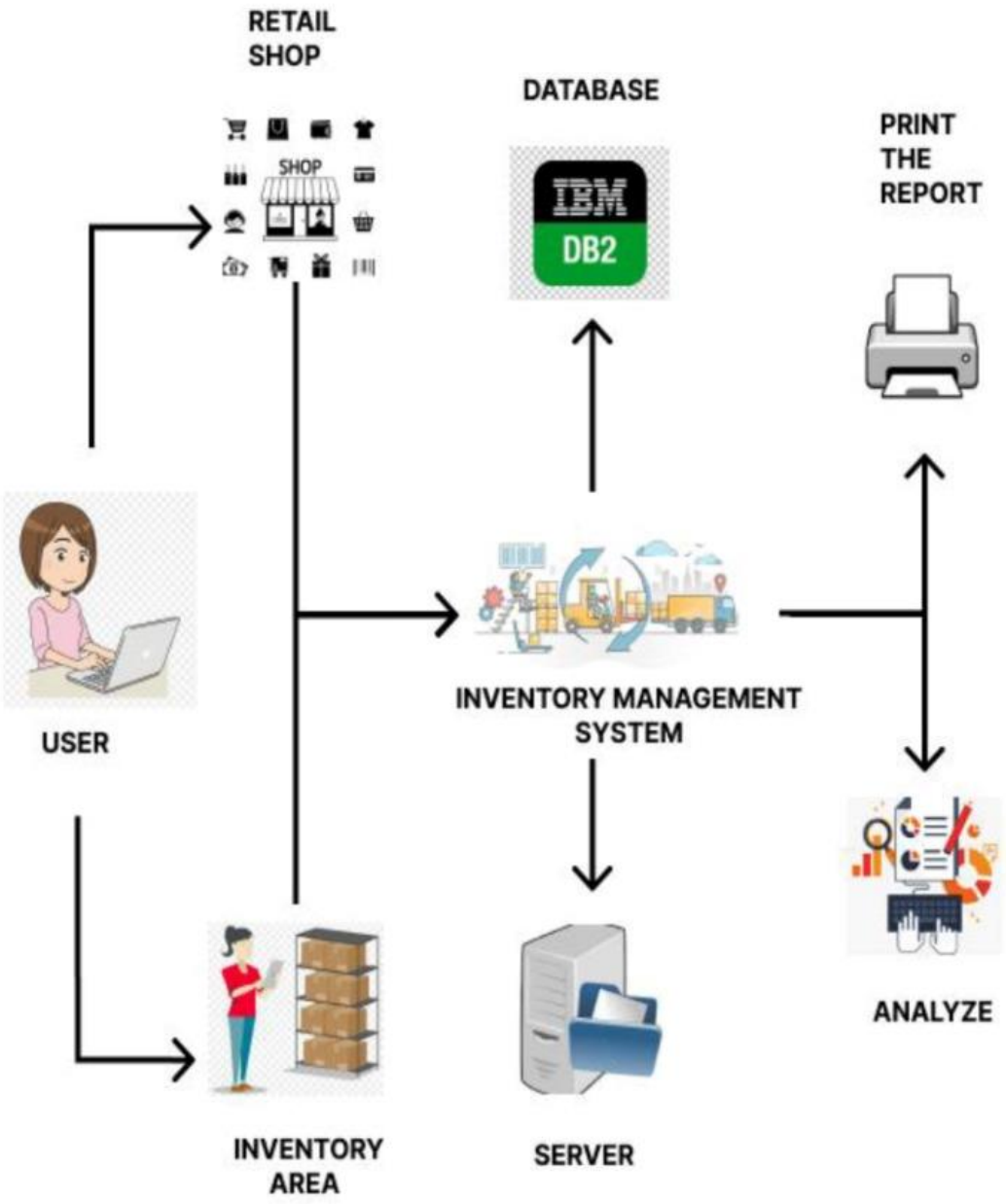
# 5 PROJECT DESIGN

## Data Flow Diagrams



## Solution & Technical Architecture

## SOLUTION ARCHITECTURE DIAGRAM



# User Stories

**User Stories**

| User Type | Functional Requirement (Epic) | User Story Number | User Story / Task | Acceptance criteria | Priority | Release |
|---|---|---|---|---|---|---|
| Customer (Mobile user) | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | I can access my account / dashboard | High | Sprint-1 |
| | | USN-2 | As a user, I will receive confirmation email once I have registered for the application | I can receive confirmation email & click confirm | High | Sprint-1 |
| | | USN-3 | As a user, I can register for the application through Facebook | I can register & access the dashboard with Facebook Login | Low | Sprint-2 |
| | | USN-4 | As a user, I can register for the application through Gmail | I can register & application Through Gmail | Medium | Sprint-1 |
| | Login | USN-5 | As a user, I can log into the application by entering email & password | I can access my account | High | Sprint-1 |
| | Dashboard | USN-6 | As a user, I can log into my account for the mobile | I can access my account /Dashboard | High | Sprint-1 |
| Customer (Web user) | Registration | USN-7 | As a user, I can register for the application by entering my email, password, and confirming my password | I can access my account/Dashboard | High | Sprint-1 |
| | | USN-8 | As a user, I will receive confirmation email once I have registered for the application | I can receive confirmation email & click confirm | High | Sprint-1 |
| | | USN-9 | As a user, I can register for the application through Facebook | I can register & access the dashboard with Facebook Login | Low | Sprint-2 |
| | | USN-10 | As a user, I can upload a Profile photo and add my name to my account | I can upload my Profile photo/Name in my account | Medium | Sprint-1 |

# 6 PROJECT PLANNING & SCHEDULING

## Sprint Planning & Estimation

| 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 | 20 | 6 Days | 24 Oct 2022 | 29 Oct 2022 | | 29 Oct 2022 |
| Sprint-2 | 20 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | | 05 Nov 2022 |
| Sprint-3 | 20 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | | 12 Nov 2022 |
| Sprint-4 | 20 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | | 19 Nov 2022 |

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

# Sprint Delivery Schedule

**Project Planning Phase**
**Project Planning Template (Product Backlog, Sprint Planning, Stories, Story Points)**

| | |
|---|---|
| Date | 23 October 2022 |
| Team ID | PNT2022TMID13832 |
| Project Name | Inventory Management for Retailers |
| Maximum Marks | 8 Marks |

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

Use the below template to create product backlog and sprint schedule

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|---|---|---|---|---|---|---|
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 3 | medium | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |
| Sprint-1 | Registration | USN-2 | As a user, I will receive a confirmation email once I have registered for the application | 2 | low | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |
| Sprint-2 | Registration | USN-3 | As a user, I can register for the application through Facebook | 2 | low | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |

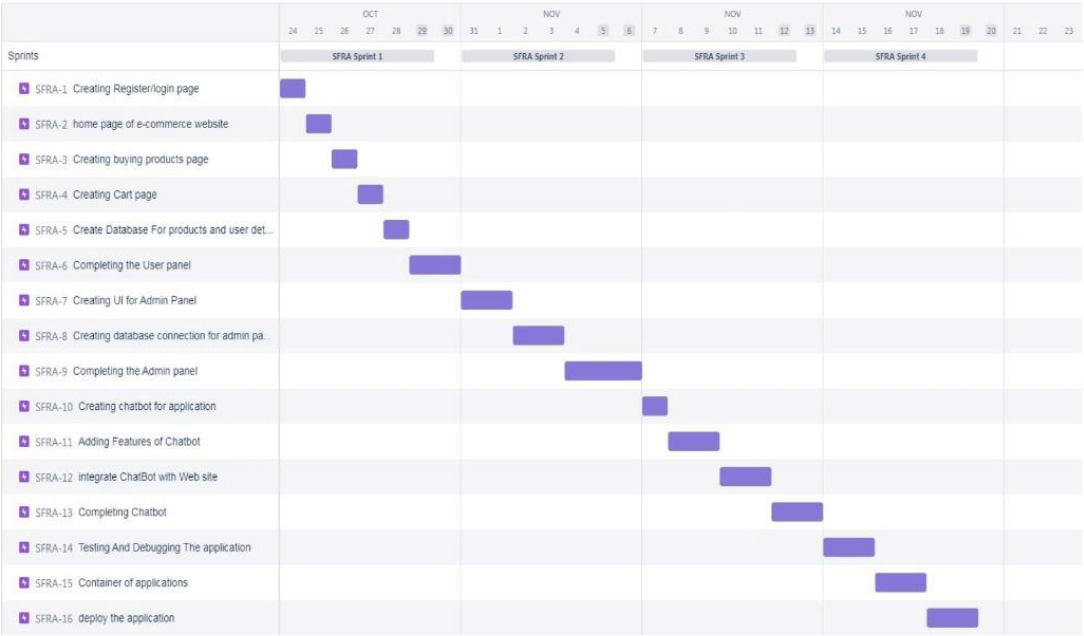
| | | | | | | |
|---|---|---|---|---|---|---|
| Sprint-1 | Registration | USN-4 | As a user, I can register for the application through Gmail | 5 | high | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |
| Sprint-1 | Login | USN-5 | As a user, I can log into the application by entering email & password | 2 | low | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |
| Sprint-1 | Dashboard | USN-6 | As a user, I must be able to see my details on the dashboard | 1 | low | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |
| Sprint-1 | Dashboard | USN-7 | As a user, I should be able to change my account settings whenever I prefer. | 3 | medium | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |
| Sprint-2 | Inventory | USN-8 | As a retailer, I should be able to alter product details | 5 | high | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |
| Sprint-2 | Inventory | USN-9 | As a retailer, I should be able to add or reduce the number of product | 2 | medium | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |
| Sprint-3 | Inventory | USN-10 | As a retailer, I should be able to get alert or notification on shortage of stock via email | 2 | low | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |

| | | | | | | |
|---|---|---|---|---|---|---|
| Sprint-3 | Communication | USN-11 | As a user, I should be able to get the needed details with the help of a chat bot | 5 | high | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |
| Sprint-4 | Maintenance | USN-12 | As an admin, I should be able to access control | 5 | high | Silambarasan V, Thavasi S, Vijaya Shankar P, Prabhudeva B |

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

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

**Burndown Chart:**




# 7 CODING & SOLUTIONING

```
import ibm_db
hostname="19af6446-6171-4641-8aba-
9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud"

uid="yrx08873"
pwd="3q2cVkfmwZwvH8UU"
driver="{IBM DB2 ODBC DRIVER}"
db="bludb"
port="30699"
protocol="TCP/IP"
cert="Certificate.crt"

dsn=(
    "DATABASE={0};"
    "HOSTNAME={1};"
    "PORT={2};"
    "UID={3};"
    "SECURITY=SSL;"
    "SSLServerCertificate={4};"
    "PWD={5};"
```

```
    ).format(db,hostname,port,uid,cert,pwd)
print(dsn)


try:
    db2=ibm_db.connect(dsn,"","")
    print("connected to data base")
except:
    print("unable to connect",ibm_db.conn_errormsg())
```

```
16.
19.
20. dsn = (
21.     "DATABASE ={0};"
22.     "HOSTNAME ={1};"
26.     "PORT ={2};"
27.     "UID ={3};"
28.     "SECURITY=SSL;"
29.     "PROTOCOL={4};"
30.     "PWD ={6};"
31. ).format(db_name, hostname, port, uid, protocol, cert, pwd)
32. connection = ibm_db.connect(dsn, "", "")
33. print()
34. # query = "SELECT username FROM users WHERE username=?"
35. # stmt = ibm_db.prepare(connection, query)
36. # ibm_db.bind_param(stmt, 1, username)
37. # ibm_db.execute(stmt)
38. # username = ibm_db.fetch_assoc(stmt)
39. # print(username)
40. app.secret_key = 'a'
41.


42. @app.route('/register', methods=['GET', 'POST'])
43. def register():
44.     msg = " "
45.     if request.method == 'POST':
46.         username = request.form['uname']
47.         email_id = request.form['email']
48.         phone_no = request.form['phone_no']
49.         password = request.form['pass']
50.         query = "SELECT * FROM users WHERE username=?;"
51.         stmt = ibm_db.prepare(connection, query)
52.         ibm_db.bind_param(stmt, 1, username)
53.         ibm_db.execute(stmt)
54.         account = ibm_db.fetch_assoc(stmt)
55.         if (account):
56.
57.             msg = "Account already exists!"
```

```
58.                 return render_template('register.html',
  msg=msg)
59.             # elif not re.match(r'[^@]+@[^@]+\.[^@]+',
  email_id):
60.             #     msg = "Invalid email addres"
61.             # elif not re.match(r'[A-Za-z0-9+', username):
62.             #     msg = "Name must contain only characters and
  numbers"
63.             else:
64.                 query = "INSERT INTO users values(?,?,?,?)"
65.                 stmt = ibm_db.prepare(connection, query)
66.                 ibm_db.bind_param(stmt, 1, username)
67.                 ibm_db.bind_param(stmt, 2, email_id)
68.                 ibm_db.bind_param(stmt, 3, phone_no)
69.                 ibm_db.bind_param(stmt, 4, password)
70.                 ibm_db.execute(stmt)
71.                 msg = 'You have successfully Logged In!!'
72.                 return render_template('login.html', msg=msg)
73.         else:
74.             msg = 'PLEASE FILL OUT OF THE FORM'
75.             return render_template('register.html', msg=msg)
76.
77.  @app.route('/', methods=['GET', 'POST'])
78.  @app.route('/login', methods=['GET', 'POST'])
79.  def login():
80.      global userid
81.      msg = ''
82.      if request.method == "POST":
83.          username = request.form['uname']
84.          password = request.form['pass']
85.          query = "select * from users where username=? and
  password=?"
86.          stmt = ibm_db.prepare(connection, query)
87.          ibm_db.bind_param(stmt, 1, username)
88.          ibm_db.bind_param(stmt, 2, password)
89.          ibm_db.execute(stmt)
90.          account = ibm_db.fetch_assoc(stmt)
91.          print(account)
92.          if account:
93.              session['Loggedin'] = True
94.              session['id'] = account['USERNAME']
95.              session['username'] = account['USERNAME']
96.              msg = 'Logged in Successfully'
97.              return redirect(url_for("dashboard"))
```

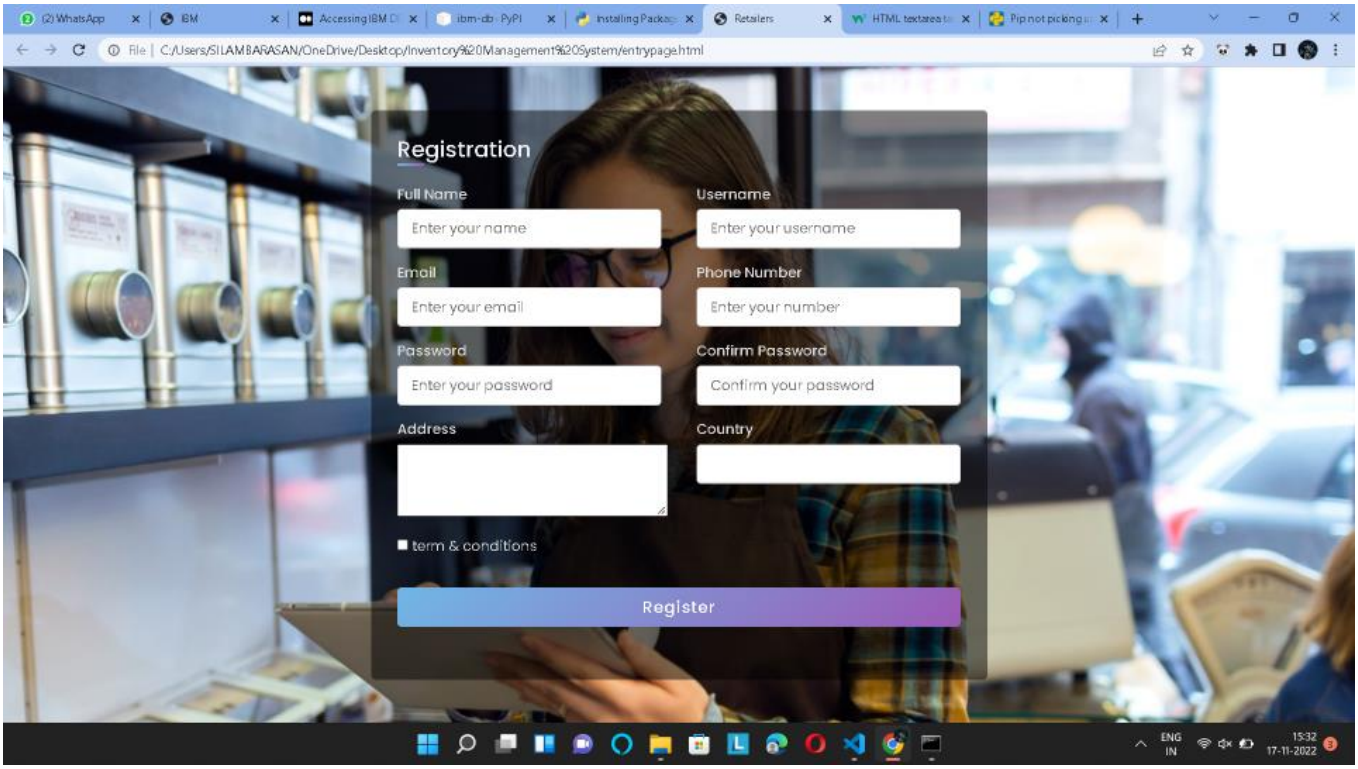
```
98.         else:
99.             msg = 'Incorrect Username or Password'
100.            return render_template('login.html', msg=msg)
101.    else:
102.        msg = 'PLEASE FILL OUT OF THE FORM'
103.        return render_template('login.html', msg=msg)
104.

105. @app.route('/welcome', methods=['GET', 'POST'])
106. def welcome():
107.    if request.method == 'POST':
108.        username = request.form['uname']
109.        print(username)
110.        return render_template('welcome.html',
  username=username)
111.    else:
112.        return render_template('welcome.html',
  username=username)
113.
114. @app.route('/about')
115. def about():
116.    return render_template('about.html')
117.
118. @app.route('/product', methods=['GET', 'POST'])
119. def product():
120.    msg = " "
121.    if request.method == 'POST':
122.        pid = request.form['pid']
123.        pname = request.form['pname']
124.        rate = request.form['rate']
125.        quantity = request.form['quantity']
126.        brand = request.form['brand']
127.        category = request.form['category']
128.        img = request.form['img']
129.
130.        query = "SELECT * FROM INVENTORYITEMS WHERE
  productID=?;"
131.        stmt = ibm_db.prepare(connection, query)
132.        ibm_db.bind_param(stmt, 1, int(pid))
133.        ibm_db.execute(stmt)
134.        account = ibm_db.fetch_assoc(stmt)
135.        if (account):
136.            msg = "Product ID already exists!"
137.            #_________________-
```

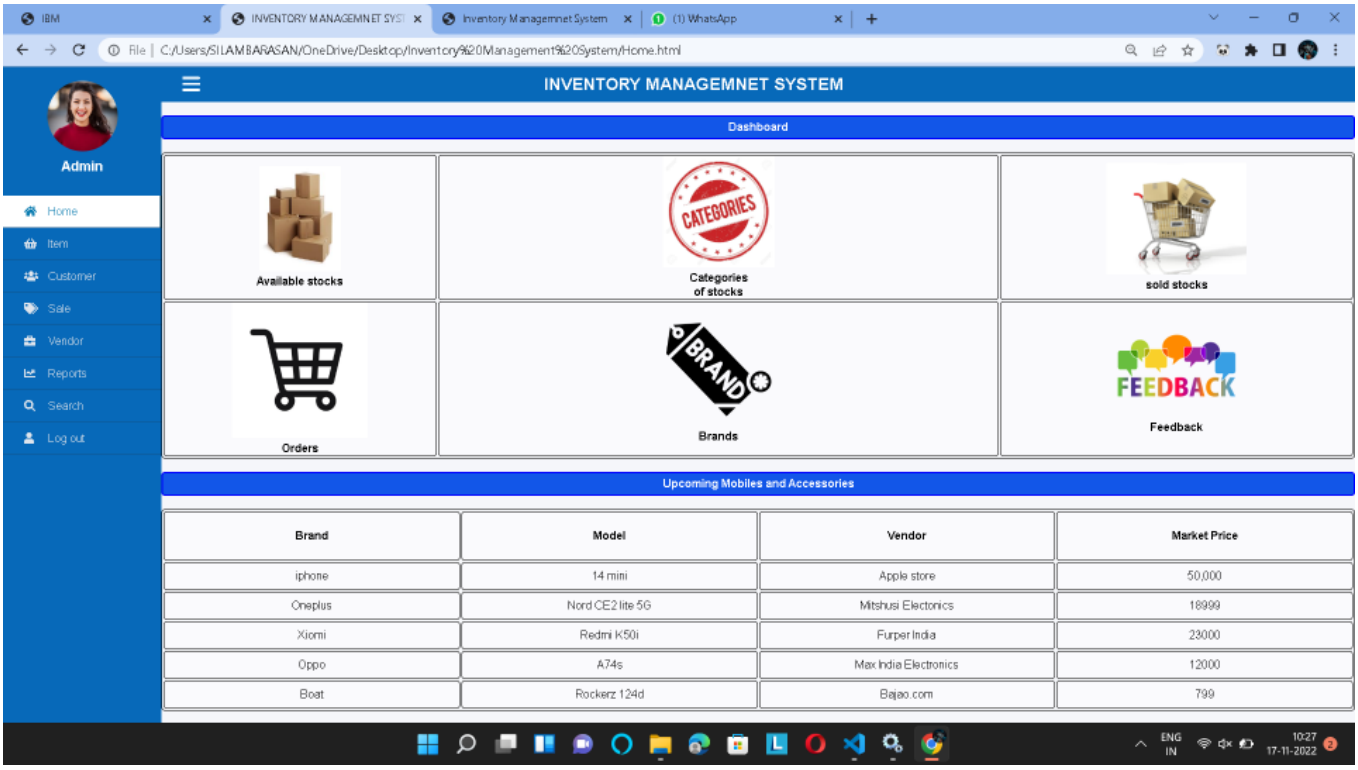
```
        else:
            query = "INSERT INTO INVENTORYITEMS values(?,?,?,?,?,?,?)"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, int(pid))
            ibm_db.bind_param(stmt, 2, pname)
            ibm_db.bind_param(stmt, 3, float(rate))
            ibm_db.bind_param(stmt, 4, int(quantity))
            ibm_db.bind_param(stmt, 5, brand)
            ibm_db.bind_param(stmt, 6, category)
            q = int(quantity)

            if(q > 0):
                ibm_db.bind_param(stmt, 7, True)
            else:
                ibm_db.bind_param(stmt, 7, False)

            ibm_db.execute(stmt)
            msg = "You have successfully Added!"
            items = GetInventoryItems()
            return render_template("product.html",items=items, msg=msg )
    else:
        msg = "PLEASE FILL OUT OF THE FORM"
        items = GetInventoryItems()
        return render_template("product.html", items=items)


@app.route("/dashboard", methods=["GET", "POST"])
def dashboard():
    items = GetInventoryItems()
    items.reverse()
    pcount = len(items)
    orderlist = GetOrderList()
    orderlist.reverse()
    ocount = len(orderlist)
    return render_template("dashboard.html", items=items, pcount=pcount,orderlist=orderlist,ocount=ocount)

@app.route("/order", methods=["GET", "POST"])
def order():
     msg = " "
     if request.method == "POST":
        oid = request.form["oid"]
```

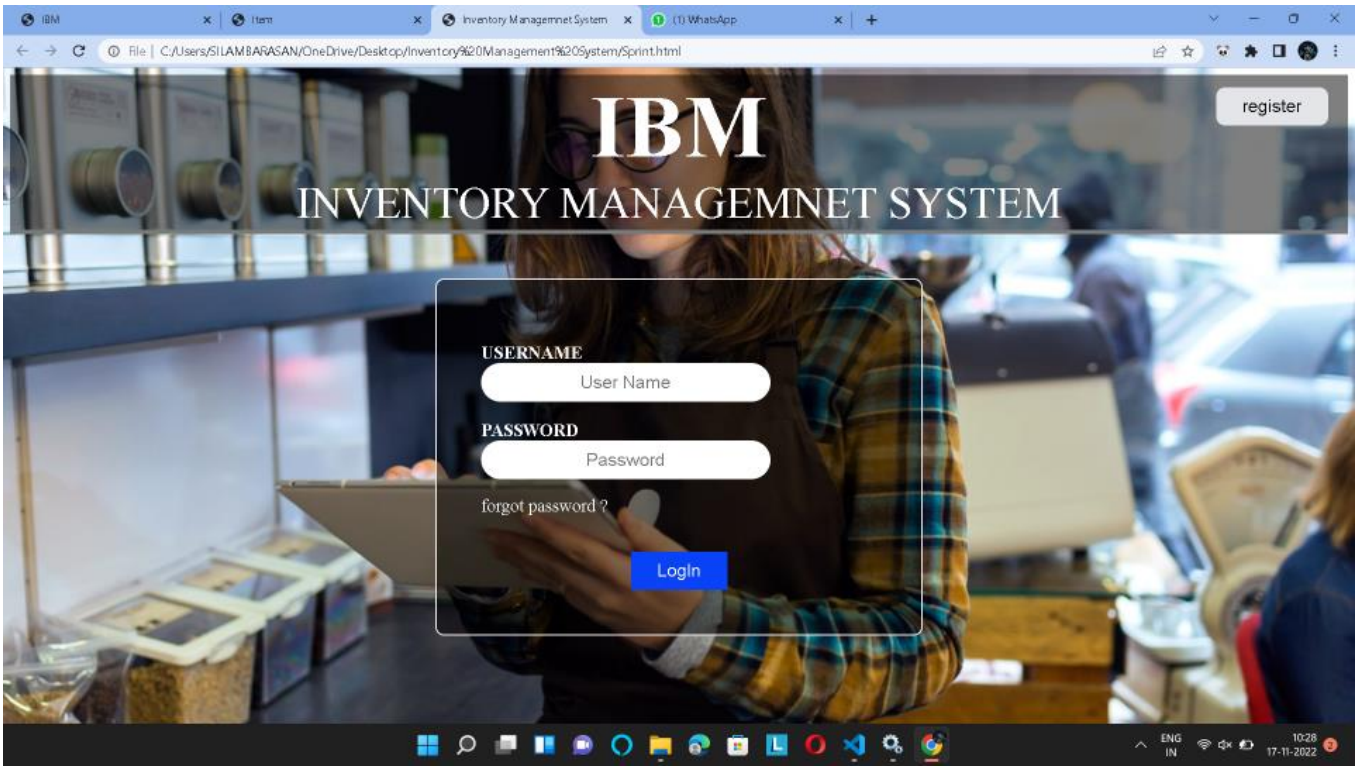
```
        cname = request.form['cname']
        cno = request.form['cno']
        odate = request.form['odate']
        pname = request.form['pname']
        nitems = request.form['items']
        discount = request.form['discount']
        status = request.form['status']
        data = GetProductAmount(pname)
        amount = abs(((float(discount) / 100) * float(data['RATE])) - float(data['RATE']))

        query = "INSERT INTO Orders values(?,?,?,?,?,?,?,?,?)"
        stmt = ibm_db.prepare(connection, query)
         ibm_db.bind_param(stmt, 1, oid)
         ibm_db.bind_param(stmt, 2, odate)
         ibm_db.bind_param(stmt, 3, cname)
         ibm_db.bind_param(stmt, 4, cno)
         ibm_db.bind_param(stmt, 5, pname)
         ibm_db.bind_param(stmt, 6, nitems)
         ibm_db.bind_param(stmt, 7, discount)
         ibm_db.bind_param(stmt, 8, amount)
         ibm_db.bind_param(stmt, 9, status)

        ibm_db.execute(stmt)
        msg = 'You have successfully Added!'
        items = GetOrderList()
        data = GetProductName()
        return render_template('order.html',items=items, data=data)
    else:
        msg = 'PLEASE FILL OUT OF THE FORM'
        items = GetOrderList()
        data = GetProductName()
        return render_template('order.html',items=items, data=data)

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

def GetInventoryItems():
    itemsdata = []
    query = "SELECT * FROM INVENTORYITEMS"
```

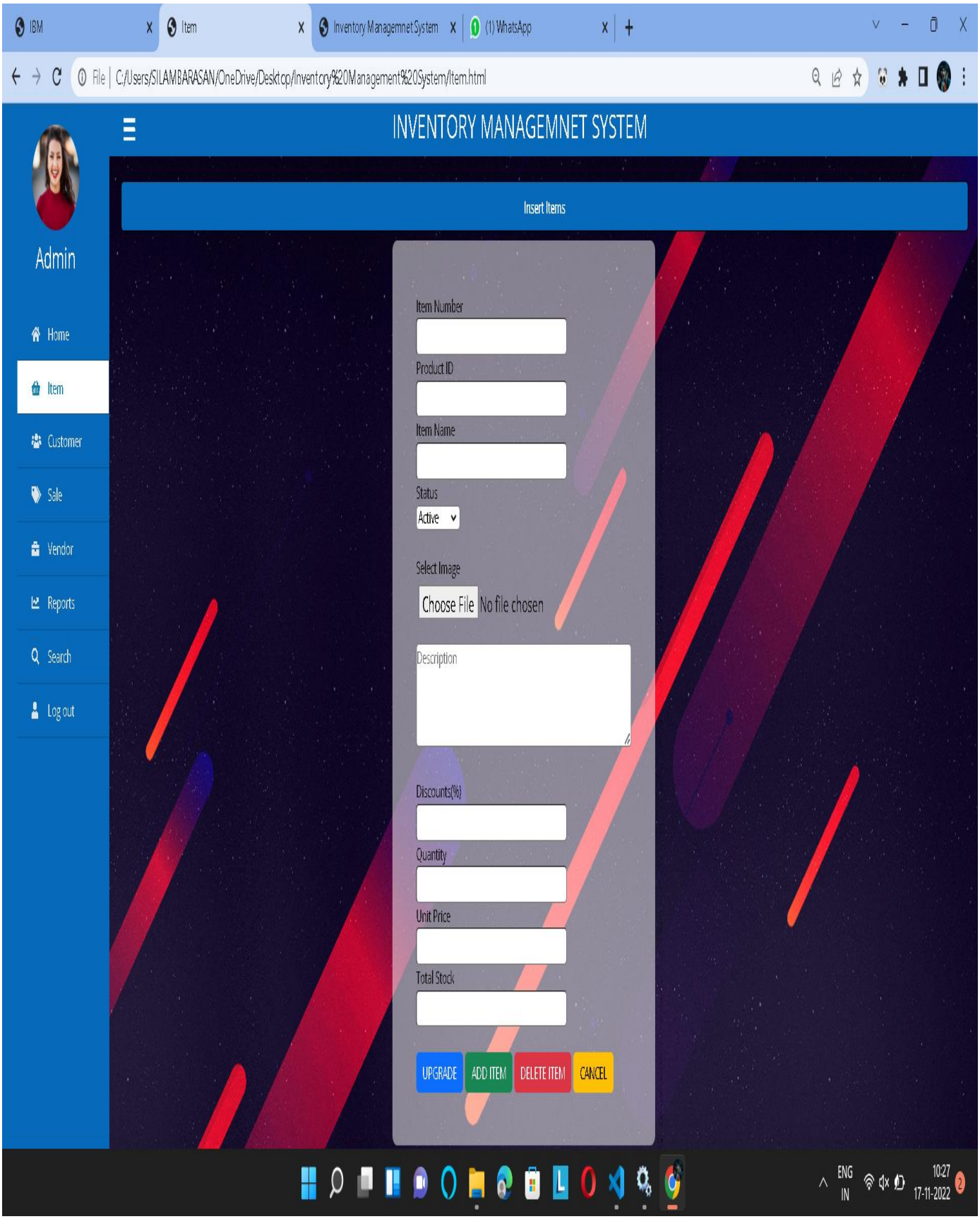
```
    stmt = ibm_db.prepare(connection, query)
    ibm_db.execute(stmt)
    items = ibm_db.fetch_assoc(stmt)
    i = 0
    while items != False:

        itemsdata.append(items)
        items = ibm_db.fetch_assoc(stmt)
        i = i+1
    return itemsdata

def GetOrderList():
    itemsdata = []
    query = "SELECT * FROM Orders"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.execute(stmt)
    items = ibm_db.fetch_assoc(stmt)
    i = 0
    while items != False:
        itemsdata.append(items)
        items = ibm_db.fetch_assoc(stmt)
        i = i+1

    return itemsdata

def GetProductAmount(pname):
    query = "select * from INVENTORYITEMS WHERE productName=?"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, pname)
    ibm_db.execute(stmt)
    return ibm_db.fetch_assoc(stmt)

def GetProductName():

    query = "SELECT productName FROM INVENTORYITEMS;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.execute(stmt)
    return ibm_db.fetch_tuple(stmt)

if __name___== "__main__":
    app.run(debug=True)
    app.run(host="0.0.0.0")
```

Output:


Registration
Full Name
Enter your name
Username
Enter your username
Email
Enter your email
Phone Number
Enter your number
Password
Enter your password
Confirm Password
Confirm your password
Address
Country
term & conditions
Register


IBM
INVENTORY MANAGEMNET SYST
Inventory Managemnet System
(1) WhatsApp
File | C:/Users/SILAMBARASAN/OneDrive/Desktop/Inventory%20Management%20System/Home.html
INVENTORY MANAGEMNET SYSTEM
Admin
Home
Item
Customer
Sale
Vendor
Reports
Search
Log out
Dashboard
Available stocks
Categories of stocks
sold stocks
Orders
Brands
Feedback
Upcoming Mobiles and Accessories
Brand | Model | Vendor | Market Price
iphone | 14 mini | Apple store | 50,000
Oneplus | Nord CE2 lite 5G | Mitshusi Electonics | 18999
Xiomi | Redmi K50i | Furper India | 23000
Oppo | A74s | Max India Electronics | 12000
Boat | Rockerz 124d | Bajao.com | 799
ENG IN
10:27 17-11-2022


IBM
Item
Inventory Managemnet System
(1) WhatsApp
File | C:/Users/SILAMBARASAN/OneDrive/Desktop/Inventory%20Management%20System/Sprint.html
register
IBM
INVENTORY MANAGEMNET SYSTEM
USERNAME
User Name
PASSWORD
Password
forgot password ?
LogIn
ENG
IN
10:28
17-11-2022


IBM
Item
Inventory Managemnet System
(1) WhatsApp
File | C:/Users/SILAMBARASAN/OneDrive/Desktop/Inventory%20Management%20System/Item.html
INVENTORY MANAGEMNET SYSTEM
Admin
Home
Item
Customer
Sale
Vendor
Reports
Search
Log out
Insert Items
Item Number
Product ID
Item Name
Status
Active
Select Image
Choose File
No file chosen
Description
Discounts(%)
Quantity
Unit Price
Total Stock
UPGRADE
ADD ITEM
DELETE ITEM
CANCEL
ENG
IN
10:27
17-11-2022

Inventory Managment System

Login Register About

## About

This is a IBM nalayathiran assignment.

This is a basic webpage where we can sign up with user details and we can login with those details.

### Technologies used

I have use Flask as a framework html and css for building the webpage and i used sqllite3 for database connectivity.

### Flask

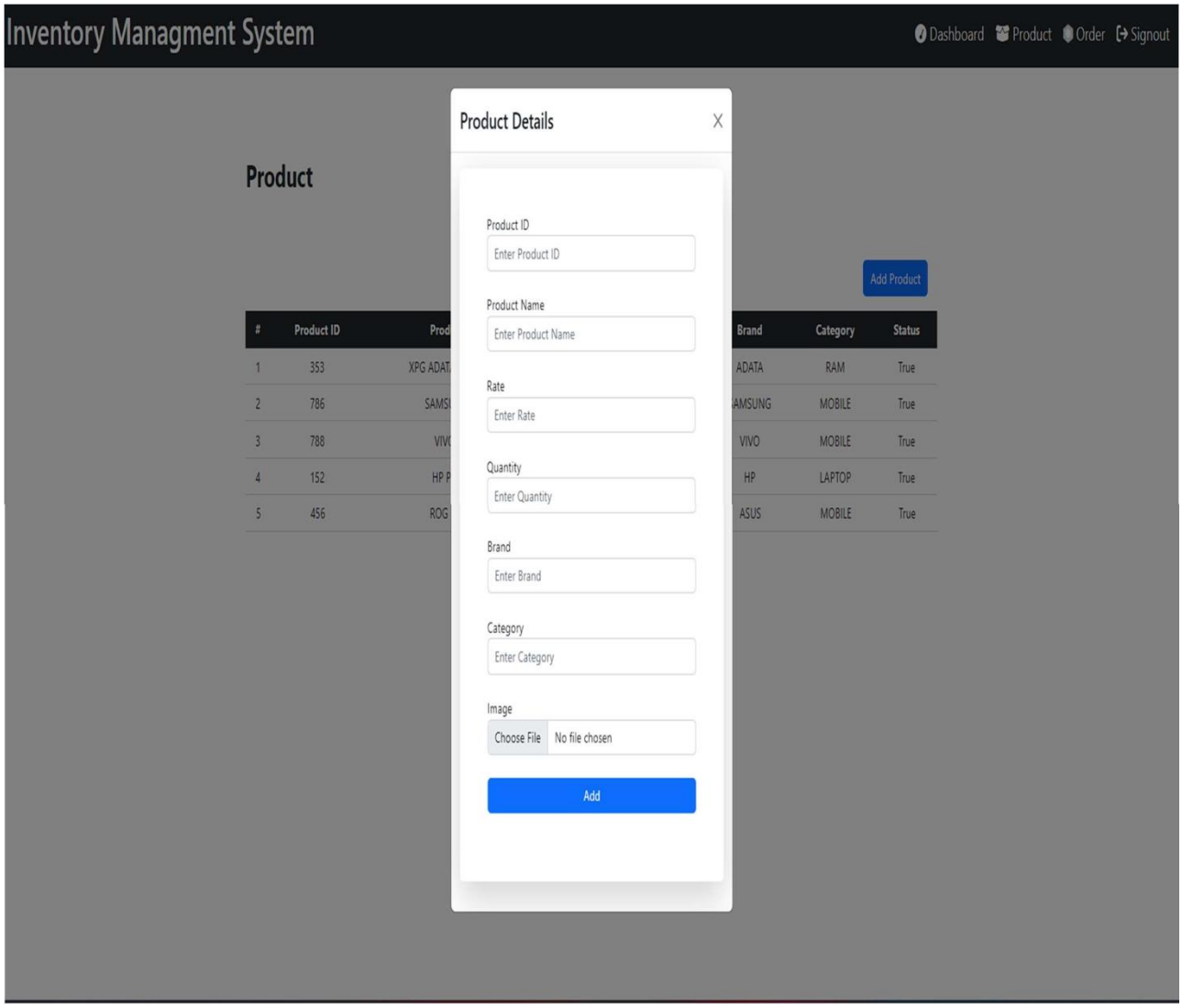
Flask is a small and lightweight Python web framework that provides useful tools and features that make creating web applications in Python easier. It gives developers flexibility and is a more accessible framework for new developers since you can build a web application quickly using only a single Python file.

### Team members

- TAMILARASAN S
- LEELARAMAN C
- JAYAPRAKASH P
- NAVEEN N

× Close

Hi! I'm a virtual assistant. How can I help you today?


Inventory Managment System
Dashboard
Product
Order
Signout
Product
Add Product
# Product ID Prod Brand Category Status
1 353 XPG ADAT ADATA RAM True
2 786 SAMS AMSUNG MOBILE True
3 788 VIV VIVO MOBILE True
4 152 HP P HP LAPTOP True
5 456 ROG ASUS MOBILE True
Product Details
X
Product ID
Enter Product ID
Product Name
Enter Product Name
Rate
Enter Rate
Quantity
Enter Quantity
Brand
Enter Brand
Category
Enter Category
Image
Choose File
No file chosen
Add

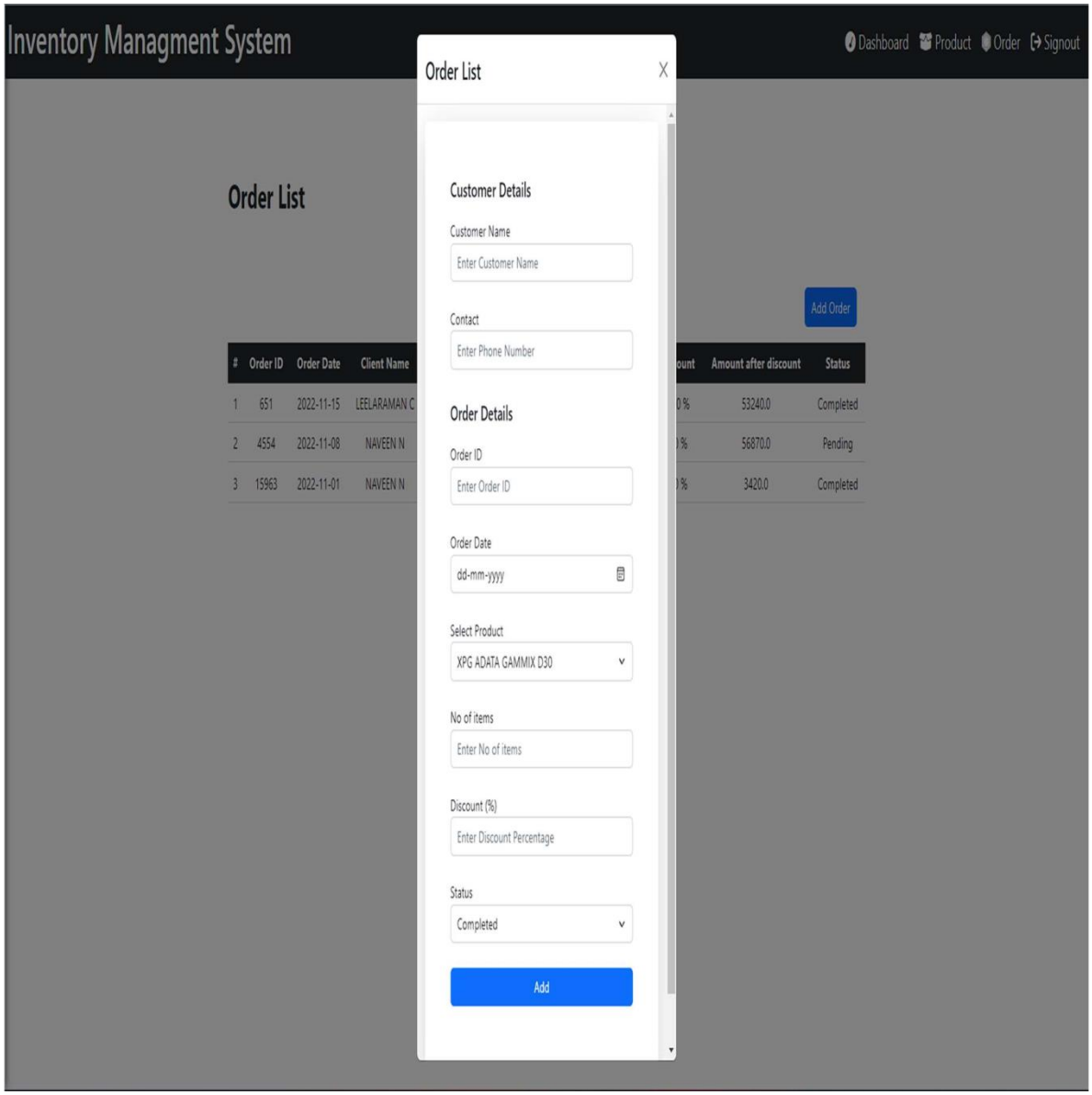
Inventory Managment System

Dashboard Product Order Signout

## Order List

Add Order

| # | Order ID | Order Date | Client Name | Contact | Product | No of items | Discount | Amount after discount | Status |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 651 | 2022-11-15 | LEELARAMAN C | 941651561 | HP 15 | 6 | 12.0 % | 53240.0 | Completed |
| 2 | 4554 | 2022-11-08 | NAVEEN N | 495436743 | HP 15 | 5 | 6.0 % | 56870.0 | Pending |
| 3 | 15963 | 2022-11-01 | NAVEEN N | 987564626 | XPG ADATA GAMMIX D30 | 5 | 5.0 % | 3420.0 | Completed |




**Result:**

Inventory management system for retailors using cloud is developed and executed at the level of completed progress .

# 8 ADVANTAGES & DISADVANTAGES

Advantages:-

- Each material can be procured in the most economical quantity.
- Purchasing and inventory control people automatically gives their attention to those items which are required only when are needed.
- Positive control can easily be handled to maintain the inventory investment at the desired level only by calculating the predetermined maximum and minimum values.

Disadvantages:-

- Sometimes, the orders are placed at the irregular time periods which may not be convenient to the producers or the suppliers of the materials.
- The items cannot be grouped and ordered at a time since the reorder points occur irregularly.
- If there is a case when the order placement time is very high, there would be two to three orders pending with the supplier each time and there is likelihood that he may supply all orders at a time.
- EOQ may give an order quantity which is much lower than the supplier minimum and there is always a probability that the order placement level for a material has been reached but not noticed in which case a stock out may occur.
- The system assumes stable usage and definite lead time. When these change significantly, a new order quantity and a new order point should be fixed, which is quite cumbersome.

# 9 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 have read above, there are different techniques that businesses can utilize to simplify and optimize stock management processes and control systems.

# 10 FUTURE SCOPE

- **Manage Inventory**: Inventory management helps to manage the stock of the company. it provides proper details of the products what kind of raw material, what are the sizes we require and etc. to the purchasing department.
- **Less Storage**: When the inventory management provides proper information to management, they buy according to them which helps the company to store fewer products.
- **Improve Productivity:** Inventory management helps to improve the productivity of the machines and manpower. Employees are aware of stocks and the quantity that require to produce.
- **Increase Profits:** Inventory management helps to improve the profits of the company. it helps to provide proper information about stocks, that saves the unnecessary expenses on stocks.

# 11 APPENDIX

**Source Code**

## App.py

```
 import re
import ibm_db
from flask import Flask, redirect, render_template, request,
session, url_for

app = Flask(__name__)

hostname = ' 19af6446-6171-4641-8aba-
9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud ' uid = '
yrx08873 '
pwd = ' 3q2cVkfmwZwvH8UU '
driver = "{IBM DB2 ODBC DRIVER}"
db_name = 'bludb'
port = '30699'
protocol = 'TCPIP'
cert = " TCP/IP" cert="Certificate.crt "

dsn = (
    "DATABASE ={0};"
    "HOSTNAME ={1};"
    "PORT ={2};"
    "UID ={3};"
    "SECURITY=SSL;"
    "PROTOCOL={4};"
```

```
    "PWD ={6};"
).format(db_name, hostname, port, uid, protocol, cert, pwd)
connection = ibm_db.connect(dsn, "", "")
print()
# query = "SELECT username FROM users WHERE username=?"
# stmt = ibm_db.prepare(connection, query)
# ibm_db.bind_param(stmt, 1, username)
# ibm_db.execute(stmt)
# username = ibm_db.fetch_assoc(stmt)
# print(username)
app.secret_key = 'a'



@app.route('/register', methods=['GET', 'POST'])
def register():
    msg = " "
    if request.method == 'POST':
        username = request.form['uname']
        email_id = request.form['email']
        phone_no = request.form['phone_no']
        password = request.form['pass']
        query = "SELECT * FROM users WHERE username=?;"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if (account):

            msg = "Account already exists!"
            return render_template('register.html', msg=msg)
        # elif not re.match(r'[^@]+@[^@]+\.[^@]+', email_id):
        #     msg = "Invalid email addres"
        # elif not re.match(r'[A-Za-z0-9+', username):
        #     msg = "Name must contain only characters and
numbers"
        else:
            query = "INSERT INTO users values(?,?,?,?)"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, username)
            ibm_db.bind_param(stmt, 2, email_id)
            ibm_db.bind_param(stmt, 3, phone_no)
            ibm_db.bind_param(stmt, 4, password)
            ibm_db.execute(stmt)
            msg = 'You have successfully Logged In!!'
            return render_template('login.html', msg=msg)
```

```
    else:
        msg = "PLEASE FILL OUT OF THE FORM"
        return render_template("register.html", msg=msg)

@app.route("/", methods=["GET", "POST"])
@app.route("/login", methods=["GET", "POST"])
def login():
    global userid
    msg = ''
    if request.method == "POST":
        username = request.form["uname"]
        password = request.form["pass"]
        query = "select * from users where username=? and
password=?"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session["Loggedin"] = True
            session["id"] = account["USERNAME"]
            session["username"] = account["USERNAME"]
            msg = "Logged in Successfully"
            return redirect(url_for("dashboard"))
        else:
            msg = "Incorrect Username or Password"
            return render_template("login.html", msg=msg)
    else:
        msg = "PLEASE FILL OUT OF THE FORM"
        return render_template("login.html", msg=msg)


@app.route("/welcome", methods=["GET", "POST"])
def welcome():
    if request.method == "POST":
        username = request.form["uname"]
        print(username)
        return render_template("welcome.html",
username=username)
    else:
        return render_template("welcome.html",
username=username)

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

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

@app.route('/product', methods=['GET', 'POST'])
def product():
    msg = ""
    if request.method == 'POST':
        pid = request.form['pid']
        pname = request.form['pname']
        rate = request.form['rate']
        quantity = request.form['quantity']
        brand = request.form['brand']
        category = request.form['category']
        img = request.form['img']

        query = "SELECT * FROM INVENTORYITEMS WHERE
productID=?;"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, int(pid))
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if (account):
            msg = "Product ID already exists!"
            #_________________-
        else:
            query = "INSERT INTO INVENTORYITEMS
values(?,?,?,?,?,?,?)"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, int(pid))
            ibm_db.bind_param(stmt, 2, pname)
            ibm_db.bind_param(stmt, 3, float(rate))
            ibm_db.bind_param(stmt, 4, int(quantity))
            ibm_db.bind_param(stmt, 5, brand)
            ibm_db.bind_param(stmt, 6, category)
            q = int(quantity)

            if(q > 0):
                ibm_db.bind_param(stmt, 7, True)
            else:
                ibm_db.bind_param(stmt, 7, False)

            ibm_db.execute(stmt)
            msg = "You have successfully Added!"
            items = GetInventoryItems()
            return render_template('product.html',items=items,
msg=msg)
```

```
    else:
        msg = "PLEASE FILL OUT OF THE FORM"
        items = GetInventoryItems()
        return render_template('product.html', items=items)



@app.route('/dashboard', methods=['GET', 'POST'])
def dashboard():
    items = GetInventoryItems()
    items.reverse()
    pcount = len(items)
    orderlist = GetOrderList()
    orderlist.reverse()
    ocount = len(orderlist)
    return render_template('dashboard.html', items=items,
pcount=pcount,orderlist=orderlist,ocount=ocount)

@app.route('/order', methods=['GET', 'POST'])
def order():
     msg = " "
     if request.method == 'POST':
        oid = request.form['oid']
        cname = request.form['cname']
        cno = request.form['cno']
        odate = request.form['odate']
        pname = request.form['pname']
        nitems = request.form['items']
        discount = request.form['discount']
        status = request.form['status']
        data = GetProductAmount(pname)
        amount = abs(((float(discount) / 100) *
float(data['RATE'])) - float(data['RATE']))

        query = "INSERT INTO Orders values(?,?,?,?,?,?,?,?,?)"
        stmt = ibm_db.prepare(connection, query)
         ibm_db.bind_param(stmt, 1, oid)
         ibm_db.bind_param(stmt, 2, odate)
         ibm_db.bind_param(stmt, 3, cname)
         ibm_db.bind_param(stmt, 4, cno)
         ibm_db.bind_param(stmt, 5, pname)
         ibm_db.bind_param(stmt, 6, nitems)
         ibm_db.bind_param(stmt, 7, discount)
         ibm_db.bind_param(stmt, 8, amount)
         ibm_db.bind_param(stmt, 9, status)
```

```
        ibm_db.execute(stmt)
        msg = "You have successfully Added!"
        items = GetOrderList()
        data = GetProductName()
        return render_template('order.html',items=items,
data=data)
    else:
        msg = "PLEASE FILL OUT OF THE FORM"
        items = GetOrderList()
        data = GetProductName()
        return render_template('order.html',items=items,
data=data)

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

def GetInventoryItems():
    itemsdata = []
    query = "SELECT * FROM INVENTORYITEMS"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.execute(stmt)
    items = ibm_db.fetch_assoc(stmt)
    i = 0
    while items != False:

        itemsdata.append(items)
        items = ibm_db.fetch_assoc(stmt)
        i = i+1
    return itemsdata

def GetOrderList():
    itemsdata = []
    query = "SELECT * FROM Orders"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.execute(stmt)
    items = ibm_db.fetch_assoc(stmt)
    i = 0
    while items != False:
        itemsdata.append(items)
        items = ibm_db.fetch_assoc(stmt)
        i = i+1

    return itemsdata

def GetProductAmount(pname):
```

```
    query = "select * from INVENTORYITEMS WHERE productName=?"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.bind_param(stmt, 1, pname)
    ibm_db.execute(stmt)
    return ibm_db.fetch_assoc(stmt)

def GetProductName():

    query = "SELECT productName FROM INVENTORYITEMS;"
    stmt = ibm_db.prepare(connection, query)
    ibm_db.execute(stmt)
    return ibm_db.fetch_tuple(stmt)

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

**Templates**

**About.html:**

```
{% extends "_Layout.html" %}
{% block body %}
<div class="container py-5">
    <div class="row">
        <div class="col-md-3 col-sm-6 col-xs-12">
            <div class="aboutus">
                <h2 class="aboutus-title">About</h2>
                <p class="aboutus-text">This is a IBM
nalayathiran assignment.</p>
                <p class="aboutus-text">This is a basic webpage
where we can sign up with user details and we can login with
those details.
                </p>
            </div>
        </div>

        <div class="col-md-5 col-sm-6 col-xs-12">
            <div class="feature">
                <div class="feature-box">
                    <div class="clearfix">
                        <div class="iconset">
                            <span class="glyphicon glyphicon-
cog icon"></span>
                        </div>
                        <div class="feature-content">
                            <h4>Technologies used</h4>
                            <p>I have use Flask as a framework
html and css for building the webpage and i used sqllite3 for
database connectivity.</p>
                        </div>
                    </div>
                </div>
                <div class="feature-box">
                    <div class="clearfix">
                        <div class="iconset">
                            <span class="glyphicon glyphicon-
cog icon"></span>
                        </div>
                        <div class="feature-content">
                            <h4>Flask</h4>
                            <p>Flask is a small and lightweight
Python web framework that provides useful tools and features
```

```
that make creating web applications in Python easier. It gives
developers flexibility and is a more accessible framework for
new developers since you can build a web application quickly
using only a single Python file.</p>
                        </div>
                    </div>
                </div>
                <div class="feature-box">
                    <div class="clearfix">
                        <div class="iconset">
                            <span class="glyphicon glyphicon-
cog icon"></span>
                        </div>
                        <div class="feature-content">
                            <h4>Team members</h4>
                            <li>silambarasan v</li>
                            <li>thavasi s</li>
                            <li>vijaya Shankar p</li>
                            <li>prabhu deva </li>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

**Base.html:**

**DashboardLayout.html:**

```
<html>
<head>
<title>Inventory Managment System for Retailers</title>
<link href="css/Bootstrap/css/bootstrap.min.css"
rel="stylesheet">
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/boo
tstrap.min.css" rel="stylesheet">
<link href="css/style.css" rel="stylesheet">
<meta name="viewport" content="width=device-width, initial-
scale=1">
```

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">
<link href="css/Bootstrap/css/bootstrap.min.css"
rel="stylesheet">
<link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols
+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/boo
tstrap.min.css" rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTR
i" crossorigin="anonymous"/>

</head>

<body class="body p-5">

  <!-- Fixed Nav Bar-->
  <div class="navbar fixed-top row bg-dark">
    <div class="px-3 col-sm-5  text-white">
      <h1 class="head">Inventory Managment System</h1>
    </div>
   <div class="col-sm-7 m-auto align-items-center text-end
text-white">
    <a class="px-0 navbar-brand text-white" href="/dashboard">
      <i class="fa fa-tachometer"></i> Dashboard</a>

      <a class="px-0 navbar-brand text-white" href="/product">
        <i class="fas fa-box-open"></i> Product</a>

    <a class="px-0 navbar-brand text-white" href="/index">
      <i class="fas fa-sign-out"></i> Signout</a>
   </div>


  </div>

  <!-- Render Body -->
<div class="container p-5">
    {% block body %}

    {% endblock %}
```

```
</div>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd
/popper.min.js" integrity="sha384-
lQsoLXl5PlLFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8
p" crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/boots
trap.min.js" integrity="sha384-
cVKlPhGWiC2Al4u+LWgxfKTRlcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQ
F" crossorigin="anonymous"></script>
<script src="JS/Script.js"></script>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.
min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstr
ap.min.js"></script>
  </body>

</html>
```

**Layout.html:**

```
<html>
<head>
<title>Inventory Managment System for Retailers</title>
<link href="css/Bootstrap/css/bootstrap.min.css"
rel="stylesheet">
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/boo
tstrap.min.css" rel="stylesheet">
<link href="css/style.css" rel="stylesheet">
<meta name="viewport" content="width=device-width, initial-
scale=1">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">
<link href="css/Bootstrap/css/bootstrap.min.css"
rel="stylesheet">
<link rel="stylesheet"
href="https://fonts.googleapis.com/css2?family=Material+Symbols
+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />
```

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css">
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/boo
tstrap.min.css" rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTR
i" crossorigin="anonymous"/>

</head>

<body class="body p-5">

  <!-- Fixed Nav Bar-->
  <div class="navbar fixed-top row bg-dark">
    <div class="px-3 col-sm-5  text-white">
      <h1 class="head">Inventory Managment System</h1>
    </div>
   <div class="col-sm-7 m-auto align-items-center text-end
text-white">

    <a class="px-0 navbar-brand text-white" href="/login">
      <i class="fa-solid fa-sign-in "></i>
      Login</a>
    <a class="px-0 navbar-brand text-white" href="/register">
      <i class="fa-solid  fa-user-plus"></i>
      Register</a>
    <a class="px-0 navbar-brand text-white" href="/about">
      <i class="fa-solid fa-circle-info px-1"></i>About</a>
   </div>


  </div>

  <!-- Render Body -->
<div class="container p-5">
    {% block body %}

    {% endblock %}
</div>
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "2d044b92-023d-4987-95b1-17d700546d4a", //
The ID of this integration.
    region: "jp-tok", // The region your integration is hosted
in.
```

```
    serviceInstanceID: "1e45b316-4478-47ea-954e-6548b7dc66ae",
// The ID of your service instance.
    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-
chat.global.assistant.watson.appdomain.cloud/versions/" +
(window.watsonAssistantChatOptions.clientVersion || 'latest') +
"/WatsonAssistantChatEntry.js";
    document.head.appendChild(t);
  });
</script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd
/popper.min.js" integrity="sha384-
lQsoLXl5PlLFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8
p" crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/boots
trap.min.js" integrity="sha384-
cVKlPhGWiC2Al4u+LWgxfKTRlcfu0JTxR+EQDz/bglDoEyl4H0zUF0QKbrJ0EcQ
F" crossorigin="anonymous"></script>
<script src="JS/Script.js"></script>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.
min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstr
ap.min.js"></script>
  </body>

</html>
```

**dashboard.html:**

```
{% extends "_DashboardLayout.html" %}
{% block body %}

<div class="py-5" style="background-image:
url('../static/Images/dashboardbg.png');" >
  <h2  style="font-weight: 700;">Dashboard</h2>
    <center class="py-5">
```

```
<div class="p-1 container text-start">
  <div class="row">
    <div class="col-sm-5">
      <h5>TOTAL PRODUCTS : {{pcount}}</h5>

    </div>
    <div class="col-sm-5">
      <h5>TOTAL ORDERS : {{ocount}}</h5>
    </div>
  </div>

</div>

  <div class="p-1 text-start">
   <h4>Recent Products</h4>
  </div>


      <table class="table " style="opacity:0.8;">
          <thead class="table-dark">
              <tr class="text-center">
                  <th scope="col">#</th>
                  <th scope="col">Product ID</th>
                  <th scope="col">Product Name</th>
                  <th scope="col">Rate</th>
                  <th scope="col">Quantity</th>
                  <th scope="col">Brand</th>
                  <th scope="col">Category</th>
                  <th scope="col">Status</th>
              </tr>
          </thead>

         <tbody class="text-center bg-white">
            {% for item in items %}
              <tr>
                <!-- # -->
                <td> {{items.index(item) + 1}} </td>
                <!-- Product ID -->
                <td> {{item["PRODUCTID"]}} </td>
                <!-- Product Name -->
                <td> {{item["PRODUCTNAME"]}} </td>
                <!-- Rate -->
                <td> {{item["RATE"]}} </td>
                <!-- Quantity -->
                <td> {{item["QUANTITY"]}} </td>
                <!-- Brand -->
```

```
                        <td> {{item["BRAND"]}} </td>
                        <!-- Category -->
                        <td> {{item["CATEGORY"]}} </td>
                        <!-- Status -->
                        <td> {{item["STATUS"]}} </td>
                      </tr>
                    {% endfor %}
                  </tbody>
                </table>
<br>
                <div class="p-1 text-start">
                  <h4>Recent Orders</h4>
                 </div>

                 <table class="table" style="opacity:0.8;">
                  <thead class="table-dark">
                      <tr class="text-center">
                        <th scope="col">#</th>
                        <th scope="col">Order ID</th>
                        <th scope="col">Order Date</th>
                        <th scope="col">Client Name</th>
                        <th scope="col">Contact</th>
                        <th scope="col">Product</th>
                        <th scope="col">No of items</th>
                        <th scope="col">Discount</th>
                        <th scope="col">Amount after
discount</th>
                        <th scope="col">Status</th>
                      </tr>
                  </thead>
                  <tbody class="text-center bg-white">
                    {% for item in orderlist %}
                      <tr>
                        <!-- # -->
                        <td> {{orderlist.index(item) + 1}} </td>
                        <!-- ORDER_ID -->
                        <td> {{item["ORDER_ID"]}} </td>
                        <!-- ORDER_DATE -->
                        <td> {{item["ORDER_DATE"]}} </td>
                        <!-- CLIENT_NAME -->
                        <td> {{item["CLIENT_NAME"]}} </td>
                        <!-- CONTACT_NO -->
                        <td> {{item["CONTACT_NO"]}} </td>
                        <!-- PRODUCT -->
                        <td> {{item["PRODUCT"]}} </td>
                        <!-- NO_OF_ITEMS -->
```

```
                    <td> {{item['NO_OF_ITEMS']}} </td>
                    <!-- DISCOUNT -->
                    <td> {{item['DISCOUNT']}} %</td>
                    <!-- AMOUNT -->
                    <td> {{item['AMOUNT']}}</td>
                    <!-- Status -->
                    <td> {{item['STATUS']}}</td>
                  </tr>
                 {% endfor %}
               </tbody>

          </table>
    </center>
</div>


{% endblock %}
```

**product.html**

```
{% extends "_DashboardLayout.html" %}
{% block body %}

<div class="py-5">
  <h2  style="font-weight: 700;">Product</h2>
    <center class="py-5">

        <div class="p-3 text-end">
         <button type="button" class="btn btn-primary" data-bs-
toggle="modal" data-bs-target="#myModal">Add Product</button>
        </div>

        <div class="modal fade" id="myModal">
          <div class="modal-dialog modal-dialog-centered modal-
dialog-scrollable">
            <div class="modal-content">

              <div class="modal-header">
                <h4 class="modal-title">Product Details</h4>
                <button type="button" class="btn-close" data-
bs-dismiss="modal"></button>
```

```
                </div>

                <div class="modal-body">
                  <form class="center"
action="{{url_for("dashboard")}}" method="post">
                    <div class="shadow-lg p-5 bg-white rounded">
                    <div class="form-group text-start">
                    <label >Product ID</label>
                    <br>
                    <input class="form-control" type="number"
name="pid" placeholder="Enter Product ID"/>
                    <br>
                    <label>Product Name</label>
                    <br>
                    <input class="form-control" type="text"
name="pname" placeholder="Enter Product Name"/>
                    <br>
                    <label>Rate</label>
                    <br>
                    <input class="form-control" type="text"
name="rate" placeholder="Enter Rate"/>
                    <br>
                    <label>Quantity</label>
                    <br>
                    <input class="form-control" type="number"
name="quantity" placeholder="Enter Quantity"/>
                    <br>
                    <label>Brand</label>
                    <br>
                    <input class="form-control" type="text"
name="brand" placeholder="Enter Brand"/>
                    <br>
                    <label>Category</label>
                    <br>
                    <input class="form-control" type="text"
name="category" placeholder="Enter Category"/>
                    <br>
                    <label>Image</label>
                    <br>
                    <input class="form-control" type="file"
name="img"/>
                    <br>

                    <input type="submit" value="Add"  class="btn
btn-primary mb-4" style="width:100%"/>
                  </div>
```

```
            </div>

        </form>
          </div>

          </div>
          </div>
          </div>





        <table class="table">
            <thead class="table-dark">
                <tr class="text-center">
                    <th scope="col">#</th>
                    <th scope="col">Product ID</th>
                    <th scope="col">Product Name</th>
                    <th scope="col">Rate</th>
                    <th scope="col">Quantity</th>
                    <th scope="col">Brand</th>
                    <th scope="col">Category</th>
                    <th scope="col">Status</th>
                </tr>
            </thead>
            <tbody class="text-center">
              {% for item in items %}
                <tr>
                  <!-- # -->
                  <td> {{items.index(item) + 1}} </td>
                  <!-- Product ID -->
                  <td> {{item['PRODUCTID']}} </td>
                  <!-- Product Name -->
                  <td> {{item['PRODUCTNAME']}} </td>
                  <!-- Rate -->
                  <td> {{item['RATE']}} </td>
                  <!-- Quantity -->
                  <td> {{item['QUANTITY']}} </td>
                  <!-- Brand -->
                  <td> {{item['BRAND']}} </td>
                  <!-- Category -->
                  <td> {{item['CATEGORY']}} </td>
                  <!-- Status -->
                  <td> {{item['STATUS']}} </td>
                </tr>
```

```
                    {% endfor %}
                </tbody>

            </table>

    </center>
</div>


{% endblock %}
```

**register.html**

```
{% extends "_Layout.html" %}
{% block body %}

<div class="p-5">
    <form class="center"
action="http://localhost:5000/register" method="post">
  <div class="shadow-lg p-5 bg-white rounded">
    <label class="py-2" style="font-weight:500;font-size:xx-
large;">Sign up</label>
  <div class="form-group">
  <label >Email</label>
  <br>
  <input class="form-control" type="text" name="email"
placeholder="Enter email"/>
  <br>
  <label >Username</label>
  <br>
  <input class="form-control" type="text" name="uname"
placeholder="Enter username"/>
  <br>
  <label>Password</label>
  <br>
  <input class="form-control" type="password" name="pass"
placeholder="Enter password"/>
  <br>
  <label>Phone no</label>
  <br>
  <input class="form-control" type="number" name="phone_no"
placeholder="Enter phone no"/>
  <br>
```

```
  <input type="submit" value="Sign up"   class="btn btn-primary
mb-4" style="width:100%"/>
 <span class="form-control">
  <center>
    Already have an account?  
    <a href="/login">login</a>
  </center>
 </span>

</div>

  </form>
  </div>

{% endblock %}
```

**login.html:**

```
{% extends "_Layout.html" %}
{% block body %}

<div class="p-5 d-flex align-items-center justify-content-
center">
    <form class="center" action="{{url_for("login")}}"
method="post">
  <div class="shadow-lg p-5 bg-white rounded">
    <label class="py-2" style="font-weight:500;font-size:xx-
large;">Login</label>
  <div class="form-group">
  <label >UserName</label>
  <br>
  <input class="form-control" type="text" name="uname"
placeholder="Enter username"/>
  <br>
  <label>Password</label>
  <br>
  <input class="form-control" type="password" name="pass"
placeholder="Enter password"/>
  <br>
```

```
  <input type="submit" value="Login"   class="btn btn-primary
mb-4" style="width:100%"/>
  <span class="form-control">
    <center>
     Don't have an account?  
      <a href="/register">register</a>
    </center>
   </span>
</div>



  </form>
  </div>

  {% endblock %}
```

**Order.html:**

```
{% extends "_DashboardLayout.html" %}
{% block body %}

<div class="py-5">
  <h2  style="font-weight: 700;">Order List</h2>
    <center class="py-5">

        <div class="p-3 text-end">
         <button type="button" class="btn btn-primary" data-bs-
toggle="modal" data-bs-target="#myModal">Add Order</button>
        </div>

        <div class="modal fade" id="myModal">
          <div class="modal-dialog modal-dialog-centered modal-
dialog-scrollable">
            <div class="modal-content">

              <div class="modal-header">
                <h4 class="modal-title">Order List</h4>
                <button type="button" class="btn-close" data-
bs-dismiss="modal"></button>
              </div>

              <div class="modal-body">
                <form class="center"
action="{{url_for("order")}}" method="post">
                  <div class="shadow-lg p-5 bg-white rounded">
                  <div class="form-group text-start">
                    <h5 class="py-2">Customer Details</h5>
```

```
                    <label >Customer Name</label>
                    <br>
                    <input class="form-control" type="text"
name="cname" placeholder="Enter Customer Name"/>
                    <br>
                    <label>Contact</label>
                    <br>
                    <input class="form-control" type="number"
name="cno" placeholder="Enter Phone Number"/>
                    <br>
                    <h5 class="py-2">Order Details</h5>
                    <label>Order ID</label>
                    <br>
                    <input class="form-control" type="number"
name="oid" placeholder="Enter Order ID"/>
                    <br>
                    <label>Order Date</label>
                    <br>
                    <input class="form-control" type="date"
name="odate" placeholder="Enter Order Date"/>
                    <br>
                    <label>Select Product</label>
                    <br>

                    <select class="form-select" aria-
label="Select Product Name" name="pname">
                      {% for item in data %}
                      <option value="{{item}}">{{item}}</option>
                      {% endfor %}
                    </select>

                    <br>
                    <label>No of items</label>
                    <br>
                    <input class="form-control" type="number"
name="items" placeholder="Enter No of items"/>
                    <br>
                    <label>Discount (%) </label>
                    <br>
                    <input class="form-control" type="text"
name="discount" placeholder="Enter Discount Percentage"/>
                    <br>
                    <label>Status</label>
                    <br>
```

```
                    <select class="form-select" aria-
label="status" name="status">
                      <option selected
value="Completed">Completed</option>
                      <option value="Pending">Pending</option>
                    </select>
                    <br>

                    <input type="submit" value="Add"   class="btn
btn-primary mb-4" style="width:100%"/>
                  </div>
                  </div>

            </form>
              </div>

              </div>
              </div>
              </div>





            <table class="table">
                <thead class="table-dark">
                    <tr class="text-center">
                        <th scope="col">#</th>
                        <th scope="col">Order ID</th>
                        <th scope="col">Order Date</th>
                        <th scope="col">Client Name</th>
                        <th scope="col">Contact</th>
                        <th scope="col">Product</th>
                        <th scope="col">No of items</th>
                        <th scope="col">Discount</th>
                        <th scope="col">Amount after
discount</th>
                        <th scope="col">Status</th>
                    </tr>
                </thead>
                <tbody class="text-center">
                  {% for item in items %}
                    <tr>
                      <!-- # -->
                      <td> {{items.index(item) + 1}} </td>
                      <!-- ORDER_ID -->
```

```
                        <td> {{item["ORDER_ID"]}} </td>
                        <!-- ORDER_DATE -->
                        <td> {{item["ORDER_DATE"]}} </td>
                        <!-- CLIENT_NAME -->
                        <td> {{item["CLIENT_NAME"]}} </td>
                        <!-- CONTACT_NO -->
                        <td> {{item["CONTACT_NO"]}} </td>
                        <!-- PRODUCT -->
                        <td> {{item["PRODUCT"]}} </td>
                        <!-- NO_OF_ITEMS -->
                        <td> {{item["NO_OF_ITEMS"]}} </td>
                        <!-- DISCOUNT -->
                        <td> {{item["DISCOUNT"]}} %</td>
                        <!-- AMOUNT -->
                        <td> {{item["AMOUNT"]}}</td>
                        <!-- Status -->
                        <td> {{item["STATUS"]}}</td>
                      </tr>
                    {% endfor %}
                  </tbody>

                </table>

    </center>
    <div class="alert alert-white text-center" role="alert">
      {{msg}}
    </div>
</div>


{% endblock %}
```

**Welcome.html:**

```
{% extends "_Layout.html" %}
{% block body %}
<h1>Welcome {{username}}</h1>
{% endblock %}
```

**Style.css:**

```
@media (max-width: 768px) {
    .carousel-inner .carousel-item > div {
      display: none;
    }
    .carousel-inner .carousel-item > div:first-child {
      display: block;
    }
```

```
  }

  .carousel-inner .carousel-item.active,
  .carousel-inner .carousel-item-start,
  .carousel-inner .carousel-item-next,
  .carousel-inner .carousel-item-prev {
    display: flex;

  }

  @media (min-width: 768px) {
    .carousel-inner .carousel-item-right.active,
    .carousel-inner .carousel-item-next,
    .carousel-item-next:not(.carousel-item-start) {
      transform: translateX(25%) !important;
    }

    .carousel-inner .carousel-item-left.active,
    .carousel-item-prev:not(.carousel-item-end),
    .active.carousel-item-start,
    .carousel-item-prev:not(.carousel-item-end) {
      transform: translateX(-25%) !important;
    }

    .carousel-item-next.carousel-item-start, .active.carousel-
item-end {
      transform: translateX(0) !important;
    }

    .carousel-inner .carousel-item-prev,
    .carousel-item-prev:not(.carousel-item-end) {
      transform: translateX(-25%) !important;
    }
  }

  .text-center
  {
    text-align: center;
  }

  .icon
  {
  margin-left:15%;
  }
.center
{
```

```
position: relative;
left:40%;
margin-top:10%;
width: 25%;
}
@media only screen and (max-width: 800px) {
.center {
  left:50%;
  margin-top:10%;
  width:50%;

}
}
body{
  background-color: #7E57C2;
}

.mt-100{
  margin-top: 200px;
}
.progress {
  width: 150px;
  height: 150px !important;
  float: left;
  line-height: 150px;
  background: none;
  margin: 20px;
  box-shadow: none;
  position: relative;
}
.progress:after {
  content: "";
  width: 100%;
  height: 100%;
  border-radius: 50%;
  border: 12px solid #fff;
  position: absolute;
  top: 0;
  left: 0;
}
.progress>span {
  width: 50%;
  height: 100%;
  overflow: hidden;
  position: absolute;
  top: 0;
```

```
  z-index: 1;
}
.progress .progress-left {
  left: 0;
}
.progress .progress-bar {
  width: 100%;
  height: 100%;
  background: none;
  border-width: 12px;
  border-style: solid;
  position: absolute;
  top: 0;
}
.progress .progress-left .progress-bar {
  left: 100%;
  border-top-right-radius: 80px;
  border-bottom-right-radius: 80px;
  border-left: 0;
  -webkit-transform-origin: center left;
  transform-origin: center left;
}
.progress .progress-right {
  right: 0;
}
.progress .progress-right .progress-bar {
  left: -100%;
  border-top-left-radius: 80px;
  border-bottom-left-radius: 80px;
  border-right: 0;
  -webkit-transform-origin: center right;
  transform-origin: center right;
  animation: loading-1 1.8s linear forwards;
}
.progress .progress-value {
  width: 90%;
  height: 90%;
  border-radius: 50%;
  background: #000;
  font-size: 24px;
  color: #fff;
  line-height: 135px;
  text-align: center;
  position: absolute;
  top: 5%;
  left: 5%;
```

```
}
.progress.blue .progress-bar {
  border-color: #049dff;
}
.progress.blue .progress-left .progress-bar {
  animation: loading-2 1.5s linear forwards 1.8s;
}
.progress.yellow .progress-bar {
  border-color: #fdba04;
}
.progress.yellow .progress-right .progress-bar {
  animation: loading-3 1.8s linear forwards;
}
.progress.yellow .progress-left .progress-bar {
  animation: none;
}
@keyframes loading-1 {
  0% {
    -webkit-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(180deg);
    transform: rotate(180deg);
  }
}
@keyframes loading-2 {
  0% {
    -webkit-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(144deg);
    transform: rotate(144deg);
  }
}
@keyframes loading-3 {
  0% {
    -webkit-transform: rotate(0deg);
    transform: rotate(0deg);
  }
  100% {
    -webkit-transform: rotate(135deg);
    transform: rotate(135deg);
  }
}
```

**GitHub**

**Project IBM-Project-33281-1660217783**

As of now we couldn’t able to complete the project on due time ,

Lots of works are in under process ,so we are not able provide the demo link.