Inventory Management System For Retailers

Team ID: PNT2022TMID22388

Bachelor of Engineering

Computer Science and Engineering

Vel Tech Multi Tech Dr.Rangarajan Dr.Sakunthala Engineering College

Industry Mentor: Dr. VASUDEVA HANUSH

Faculty Mentor: Dr. SATHISH KUMAR P

Team Members :-

113119UG03117 - Vinothkumar J

113119UG03103 - Srijith R

113119UG03030 - Goutham G

113119UG03066 - Pavan S

1.INTRODUCTION

1.1 Project Overview

Retail inventory management involves making sure you have the right amount of inventory, neither too little nor too much of the goods that customers want to buy. Retailers may meet client demand without running out of product by properly managing their inventory.

Effective retail inventory management reduces costs and improves knowledge of sales patterns in practice. Tools and techniques for retail inventory management provide merchants with more data on which to run their businesses. Applications have been created to assist shops in keeping track of and managing the supply of their own products. Retailers will be prompted by the System to register their accounts by providing necessary information. Retailers can log into the programme to access their accounts.Once retailers have successfully logged in to the programme, they can update the information on their inventory. Users can also add new goods by providing the necessary information regarding the item. They have access to the current inventory's specifics. If there is no stock found in the accounts of the retailers, the System will automatically send an email alert to them. In order for them to order new stock.

1.2 Purpose

The goal of retail inventory management is to maintain the right amount of desired product in stock. Retailers may meet client demand without running out of product by properly managing their inventory.

2. LITERATURE SURVEY

2.1 Existing problem

At an Engineering College, there is a student-run store called stationery shop that sells a range of items that students need, including safety boots, workshop jackets, stationery, and textbooks. It requires ongoing monitoring of stock-in and stock-out because it is a well-established store. The shop's inventory management, however, is not practical or effective enough to handle all the stock in the inventory because they continue to use a manual inventory system. This causes a number of issues, one of which is that the members of the shop always have to manually gather and calculate their stock at the start of each semester.

2.2 References

1. A. Dahbi and H. T. Mouftah, "Supply chain efficient inventory management as a service offered by a cloud-based platform," 2016 IEEE International Conference on Communications (ICC), 2016, pp. 1-7, doi: 10.1109/ICC.2016.7510722.

2. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0259284

3. W. Intayoad and P. Temdee, "Inventory cloud service for local SME: A scenario study for Ice cream factory," 2013 13th International Symposium on Communications and Information Technologies (ISCIT), 2013, pp. 741-745, doi: 10.1109/ISCIT.2013.6645952.

4. R. F. Olanrewaju, A. Irham Dollah and B. A. Ajayi, "Cloud-Based Inventory System for Effective Management of Under and Over-stock Hazards," 2021 8th International Conference on Computer and Communication Engineering (ICCCE), 2021, pp. 274-278, doi: 10.1109/ICCCE50029.2021.9467138

2.3 Problem Statement Definition

Retail inventory management involves making sure you have the right amount of inventory that customers want to buy. Retailers may meet client demand without running out of product by properly managing their inventory.

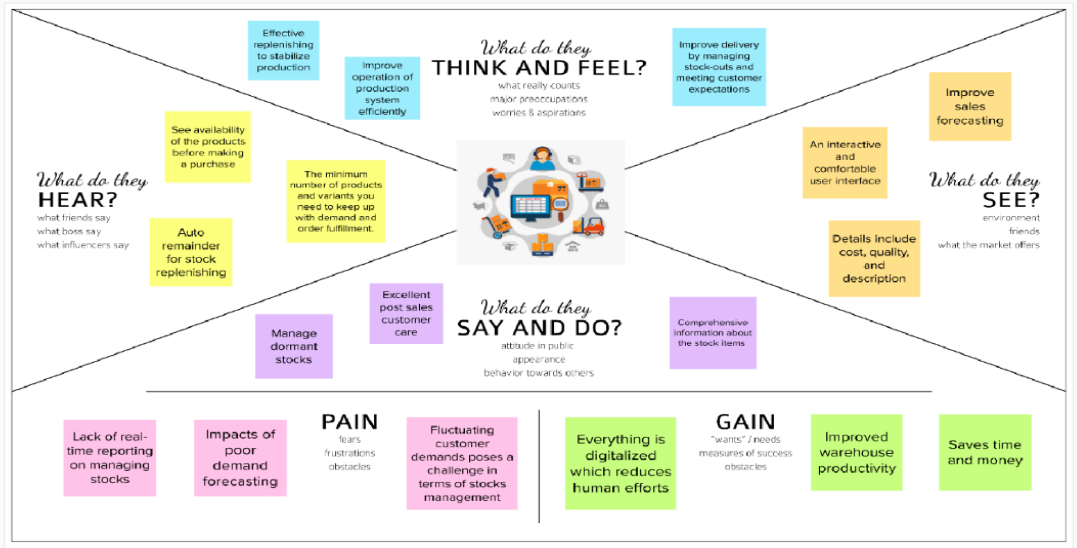
3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

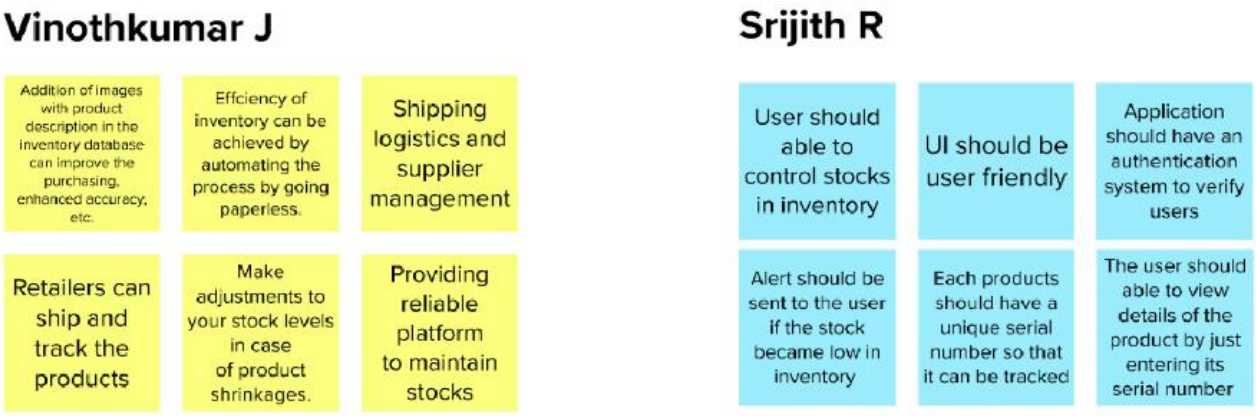
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes.

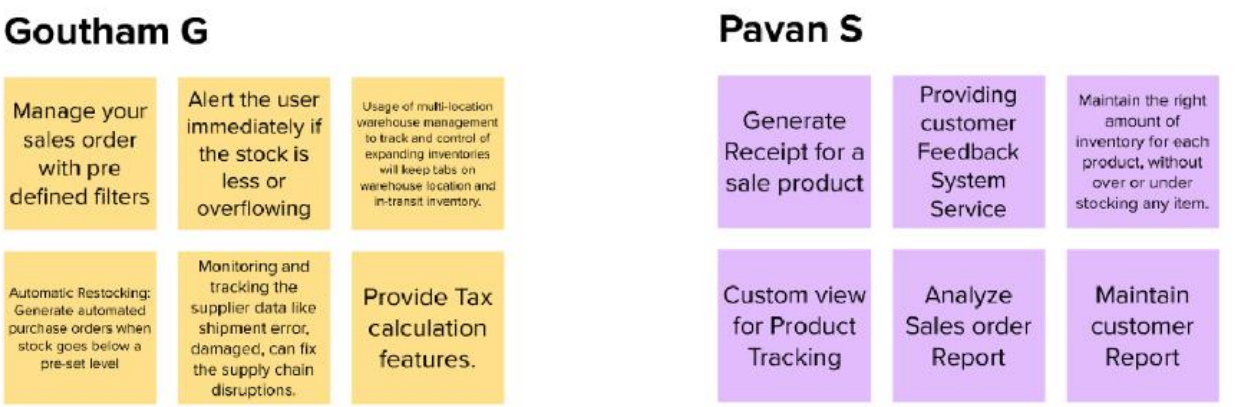
It is a useful tool to helps teams better understand their users.
Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

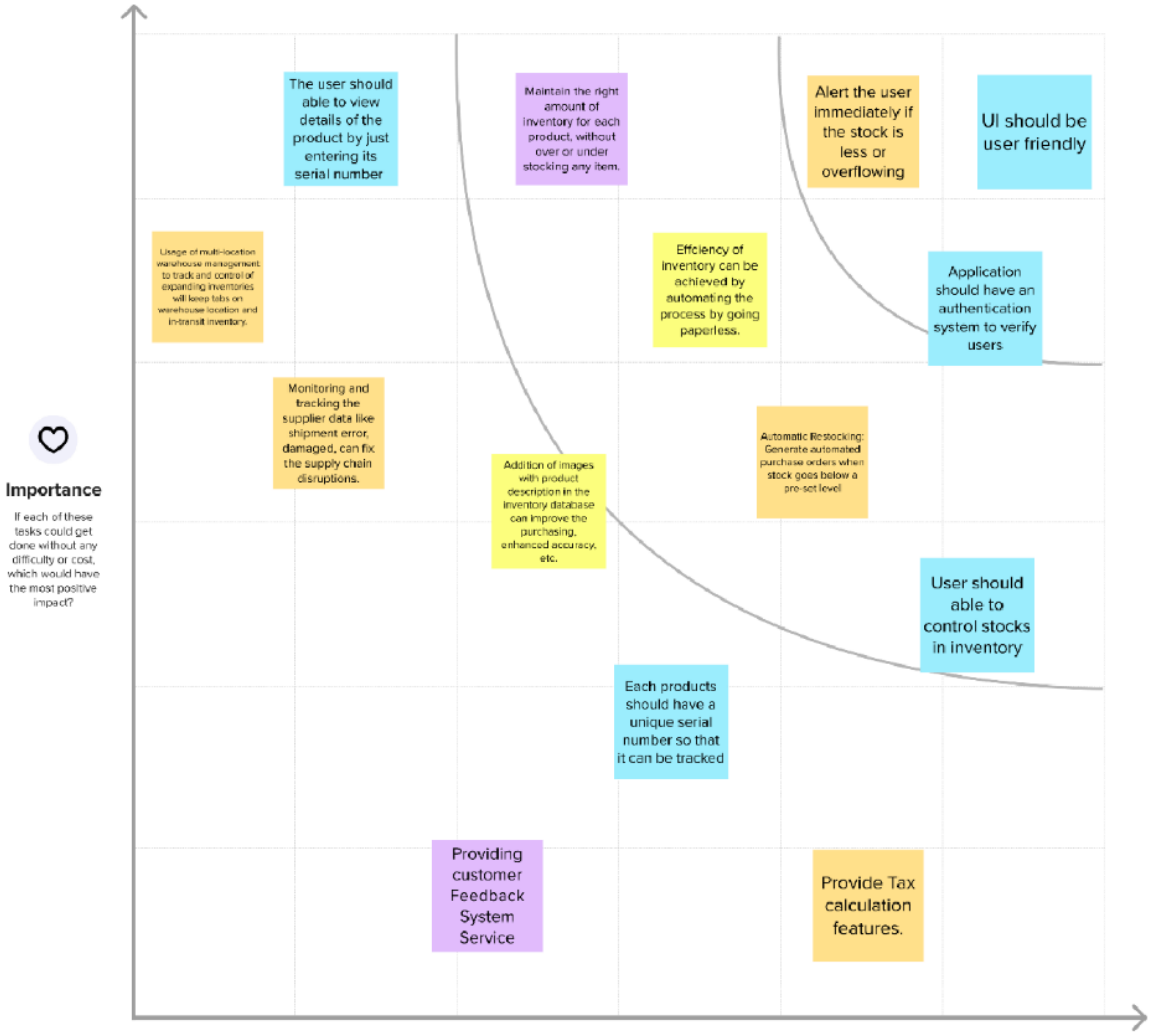
Inventory Managment System for Retailers Empathy Map



3.2 Ideation & Brainstorming





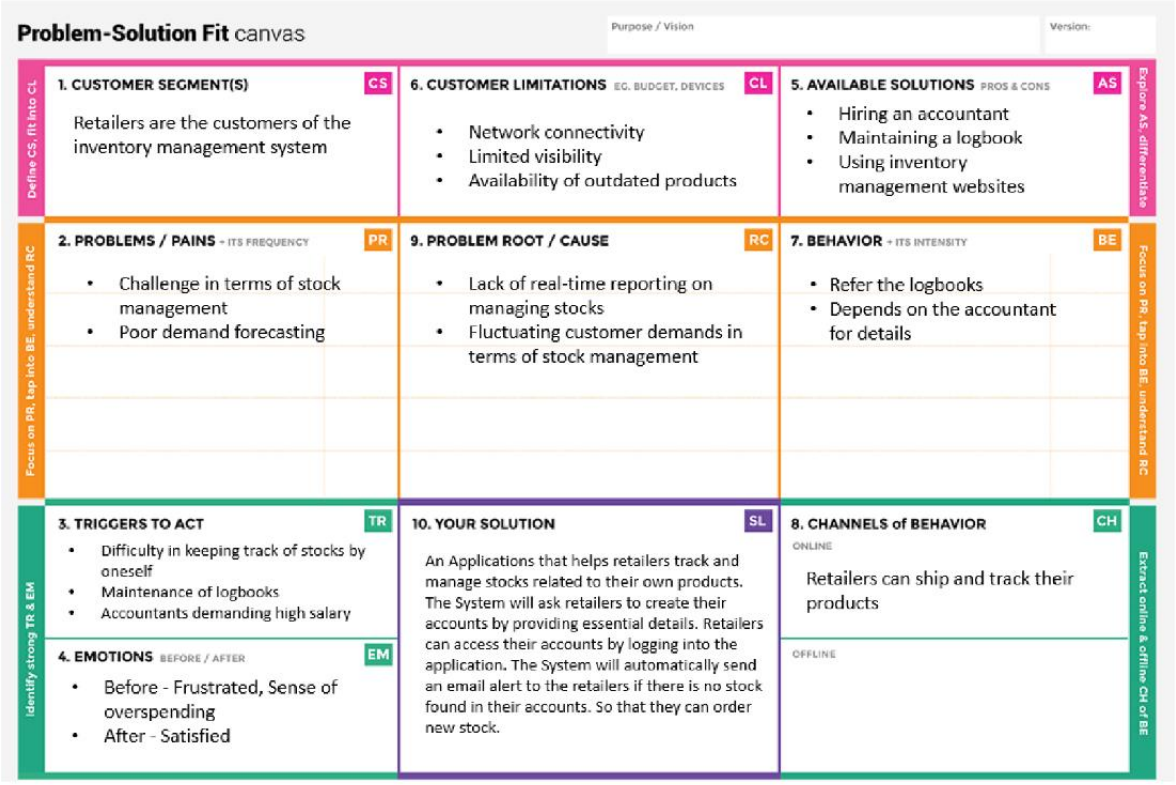

The user should able to view details of the product by just entering its serial number
Maintain the right amount of inventory for each product, without over or under stocking any item.
Alert the user immediately if the stock is less or overflowing
UI should be user friendly
Usage of multi-location warehouse management to track and control of expanding inventories will keep tabs on warehouse location and in-transit inventory.
Efficiency of inventory can be achieved by automating the process by going paperless.
Application should have an authentication system to verify users
Monitoring and tracking the supplier data like shipment error, damaged, can fix the supply chain disruptions.
Automatic Restocking: Generate automated purchase orders when stock goes below a pre-set level
Addition of images with product description in the inventory database can improve the purchasing, enhanced accuracy, etc.
User should able to control stocks in inventory
Each products should have a unique serial number so that it can be tracked
Providing customer Feedback System Service
Provide Tax calculation features.
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 Proposed Solution

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	Retailers want to carry out merchandise with sufficient stock in hand which is neither too little nor too much, so an end-to-end web application can be created which is capable of displaying the current amount of stock present in the warehouse, which helps inventory managers to keep track of products all the time.
2.	Idea / Solution description	Applications have been developed to help retailers track and manage stocks related to their own products. The System will ask retailers to create their accounts by providing essential details. Retailers can access their accounts by logging into the application. Once retailers successfully log in to the application they can update their inventory details, also users will be able to add new stock by submitting essential details related to the stock. They can view details of the current inventory. The System will automatically send an email alert to the retailers if there is no stock found in their accounts. So that they can order new stock.
3.	Novelty / Uniqueness	Though we have a lot of inventory management applications, this one is unique with a feature that when any stock count gets reduced to minimum, an alert email with stock details will be automatically generated for the retailer and the appropriate seller from whom the retailer purchases goods. This helps to reduce manpower, cost and saving time.
4.	Social Impact / Customer Satisfaction	Customer satisfaction is the key for success of a business. The availability of product is just one way in which an inventory management system creates customer satisfaction. Inventory management systems are designed to monitor product availability, determine purchasing schedules for better customer interaction.

5.	Business Model (Revenue Model)	We can provide the application for retailers on a subscription basis with which revenue can be generated.
6.	Scalability of the Solution	This proposed system for inventory management system can accommodate expansion without restricting the existing workflow and ensure an increase in the output or efficiency of the process. Inventory data can be scaled up and scaled down based on the number of available inventories in the warehouse.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

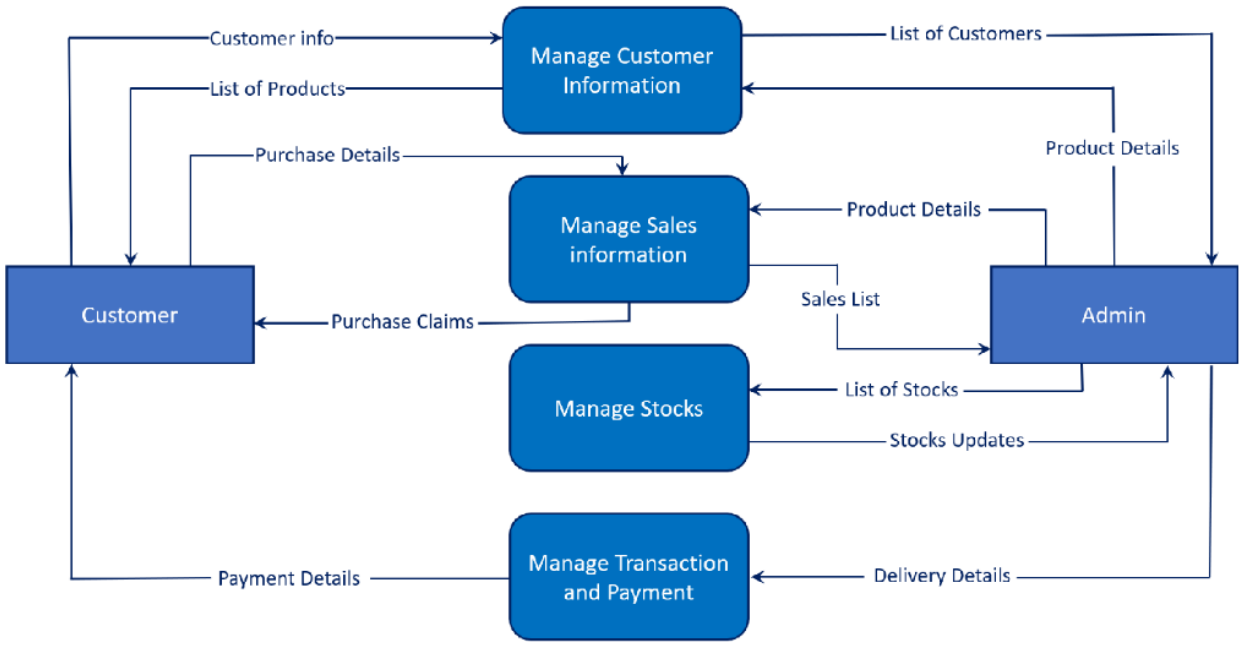
FR No.	Functional Requirement (EPIC)	Sub Requirement (Story / Sub-Talk)
FR-1	User Registration	Registration through Gmail Registration through User ID
FR-2	User Confirmation	Confirmation via Email

4.2 Non-Functional requirements

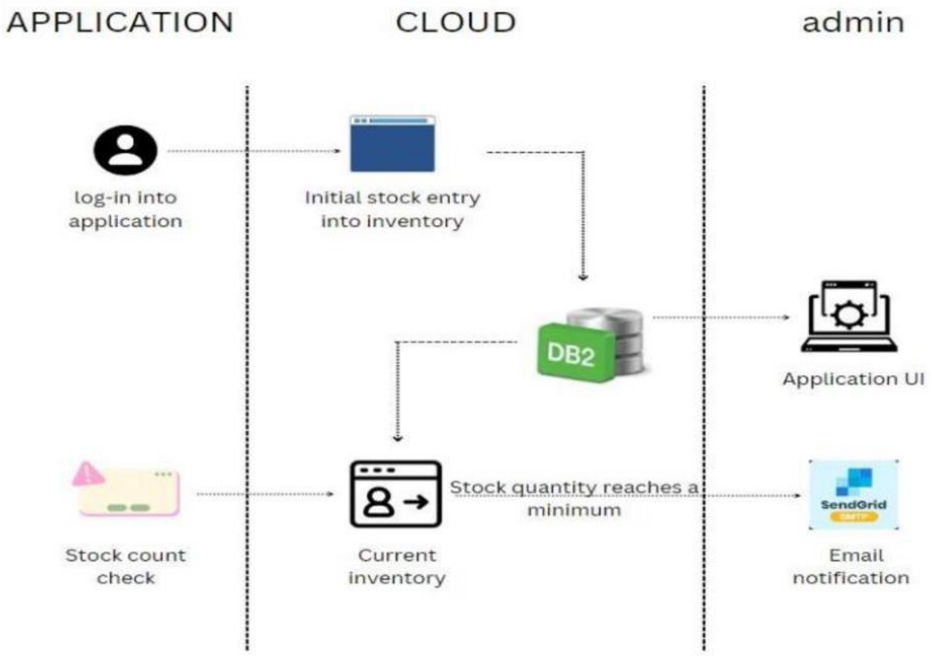
FR NO.	Non-Functional Requirement	Description
NFR-1	Usability	Always companies to identify which and how much stock to order at what time.
NFR-2	Security	Web App Sec.
NFR-3	Reliability	• Accurate reporting for better fulfilment, under or over shipping and selling. • Real-time tracking and view of stock levels, allows retailers to monitor and immediately react if level ever go amiss.
NFR-4	**Performance**	**Best User interface and accurate**
NFR-5	Availability	Readily available irrespective of time and place when accessible to internet
NFR-6	Scalability	Inventory management system can accomodate expansion without restricting the existing workflow and ensure an increase in the output or efficiency of the process.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through 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 my application through Gmail	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can log into the application by entering email & password	High	Sprint-1

	Dashboard	USN-6	As a User, | can view the stock availability status	I can view stock availability status	High	Sprint-2
	Search	USN-7	Search an item from the inventory	Easy to find any item from the inventory	High	Sprint-2
	Orders	USN-8	As a user can place the order	User can place an order and can receive confirmation message	High	Sprint-3
	Alert	USN-9	As a user, I Should receive alert on stock availability if it drops below the set threshold	I should receive an email alert on stock availability if it drops below the set threshold level	High	Sprint-3
Customer Care Executive	Customer Support	USN10	As a customer care executive, I can view the complaints from the customers on the chat box	Customer care executive can view customer complaints on the chat box	Medium	Sprint-4
Administrator	Responsibility	USN-11	As a system administrator, | want to be able to add new user when required	I can add new users	High	Sprint-3

6. PROJECT PLANNING & SCHEDULING

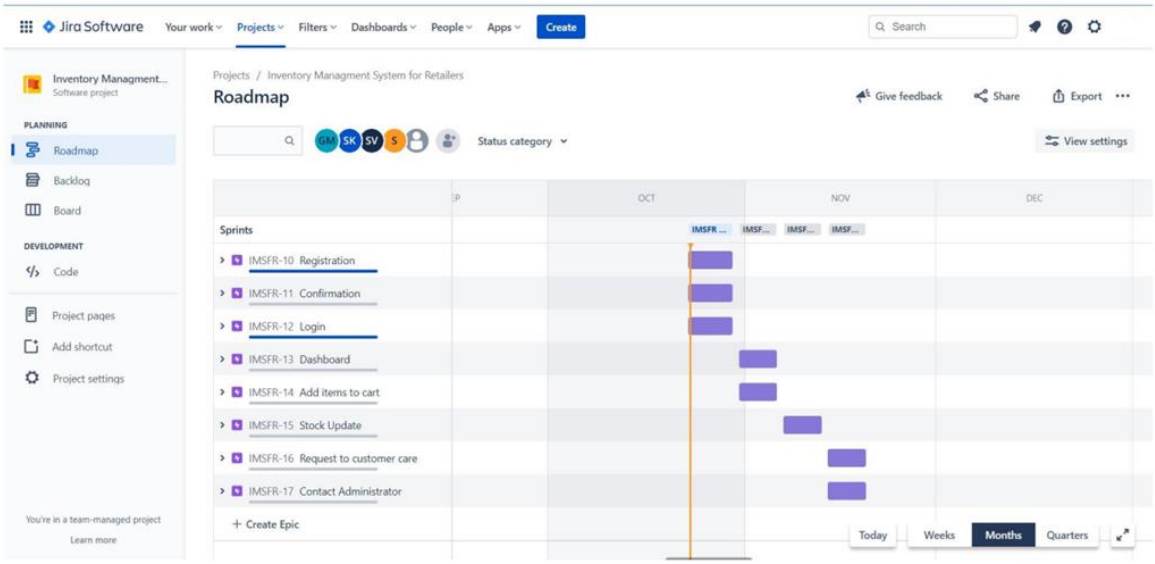
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Vinothkumar, Srijith
Sprint-1		USN-2	As a user, I can register for the application through E-mail	1	Low	Pavan
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	2	High	Srijith, Goutham
Sprint-1		USN-4	As a user, If I forgot password, I receive email and can reset my password	2	Medium	Vinothkumar
Sprint-2	Dashboard	USN-5	As a User, I can view products available on inventory	4	High	Srijith, Goutham
Sprint-2	Add items	USN-6	As a User, I can add products to the inventory	3	High	Pavan, Vinothkumar
Sprint-3	Sell items	USN-7	As a User, I can sell the products	3	High	Srijith, Goutham
Sprint-3	Stock update	USN-8	As a User, I can Update the stock details	2	Medium	Srijith
Sprint-4	Generate Email	USN-9	As a User, I receive alert mail on low stocks	4	High	Vinothkumar, Goutham

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	7	6 Days	24 Oct 2022	29 Oct 2022	7	29 Oct 2022
Sprint-2	9	6 Days	31 Oct 2022	05 Nov 2022	9	05 Nov 2022
Sprint-3	5	6 Days	07 Nov 2022	12 Nov 2022	5	12 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022	10	19 Nov 2022

6.3 Reports from JIRA



7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Login

Users are able to login to the system by filling in the details like their username and password, if the entered details are correct it will leave the message like 'Logged in successfully' ; if not it will leave the message like ' incorrect username / password'.

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


#login_validator
@app.route('/login_validation',methods=['GET','POST'])
def login_validation():
    email=request.form.get('email').lower()
    password=request.form.get('password')
    username = check_query_data.check_signle_column('user','email',email)
    session["username"] = username[1]
    session["email"] = email

    #checking for email and password in db
    if (check_query_data.check_two_column(t='user',c1='email',d1=email,c2='password',d2=password)):
        email = session.get("email")

        total_item = dashboard_view.total_items(email)
        profit = dashboard_view.dashboard_profit(email)
        low_stock = dashboard_view.low_stock(email)
        stock_cost = dashboard_view.stock_cost(email)
        username = check_query_data.check_signle_column('user','email',email)
        data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{username[0]} WHERE stock <=10 ")
        if data:

            return render_template('dashboard.html',user_name_nav=username[1],total_item=total_item,profit=profit,
                                   low_stock=low_stock,stock_cost=stock_cost,users=data)
        return render_template('dashboard.html',user_name_nav=username[1],total_item=total_item,profit=profit,
                               low_stock=low_stock,stock_cost=stock_cost)
    else:
        return render_template('login.html',invalid_msg='Invalid login credentials')
```

7.2 Registration

Users can sign up into the system by entering their details. If the account already exists it will leave the message like ‘Account already exists’ / ‘ Enter the valid email address’ / ‘name must contain only characters and numbers’. If users entered the correct details then it will send the message like ‘ You are successfully registered click signin !’.

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


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

    name=request.form.get('name')
    company_name=request.form.get('company').lower()
    email=request.form.get('email').lower()
    password=request.form.get('password')
    session["email"] = email
    #checks already email and company_name exist in db
    company_exist = check_query_data.check_signle_column(t='user',c1='company_name',d1=company_name)
    email_exist = check_query_data.check_signle_column(t='user',c1='email',d1=email)

    company_msg = "Company already exist"
    email_msg = "Email already exists"

    #checks empty field and validates company_name and email
    if(len(name)==0 or len(company_name)==0 or len(email)==0 or len(password)==0):
        return render_template("register.html",field_empty="Please enter all fields")
    elif(company_exist and email_exist):
        return render_template("register.html",company_msg=company_msg,email_msg=email_msg)
    elif(company_exist):
        return render_template("register.html",company_msg=company_msg)
    elif(email_exist):
        return render_template("register.html",email_msg=email_msg)

    #inserts user data into db user table
    insert_data_database.insert_user_table(name,email,password,company_name)
    create_table.item_table(email)
    create_table.sales_table(email)

    total_item = dashboard_view.total_items(email)
    profit = dashboard_view.dashboard_profit(email)
    low_stock = dashboard_view.low_stock(email)
    stock_cost = dashboard_view.stock_cost(email)
    username = check_query_data.check_signle_column('user','email',email)
    session["username"] = username[1]
    session["email"] = email
    data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{username[0]} WHERE stock <=10 ")
    if data:

        return render_template('dashboard.html',user_name_nav=username[1],total_item=total_item,profit=profit,
                                                low_stock=low_stock,stock_cost=stock_cost,users=data)
    return render_template("dashboard.html",user_name_nav=username[1],total_item=total_item,profit=profit,
                                                low stock=low stock,stock cost=stock cost)
```

7.3 Forgot Password

If user forgot their password they can reset their password but clicking on forgot password in login page and they will be redirected to forgot password email verification page. Here the user have to enter their email id if a account exist with the email id provided by user then a Reset password mail will be sent to user with link to change password.

```
#checks email(account) exists or not
@app.route('/verify_email',methods=['POST','GET'])
def verify_email():
    if request.method == 'POST':
        email=request.form.get('email')
        email_exist = check_query_data.check_signle_column('user','email',email)

        #puts data into session
        session["email"] = email

        if (email_exist):
            send_mail.mail(email)
            return render_template('for_email_verify.html',field_empty='We have e-mailed your password reset link!')
        return render_template('for_email_verify.html',field_empty='We cannot find your email')

    return render_template('for_email_verify.html')


#forgot password
@app.route('/forgot_password_verify/<email>',methods=['POST','GET'])
def forgot_password_verify(email):
    if request.method == 'POST':
        decode_email = session.get("decode_email")
        new_password=request.form.get('password')
        cnf_password=request.form.get('cnf_password')
        if(len(new_password) == 0 and len(cnf_password) == 0):
            return render_template('forgot_password.html',invalid_msg='Please enter password')
        elif(new_password==cnf_password):
            update_data_database.update_singel_data(cnf_password,decode_email)
            return render_template('login.html')
        else:
            return render_template('forgot_password.html',invalid_msg='New password and confirm new password do not match')

    decode_email = code.decode(f"{email}")
    session["decode_email"] = decode_email
    return render_template("forgot_password.html")
```

7.4 Dashboard

In dashboard analytics section the user will able to see number see overall stock count , low stock items count, sales profit and total cost of inventory and in low stock items section items which are under low stock threshold will be listed.

```
#dashboard page
@app.route('/dashboard')
def dashboard():
    email = session.get("email")
    total_item = dashboard_view.total_items(email)
    profit = dashboard_view.dashboard_profit(email)
    low_stock = dashboard_view.low_stock(email)
    stock_cost = dashboard_view.stock_cost(email)
    user_id=check_query_data.check_signle_column('user','email',email)
    data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{user_id[0]} WHERE stock <=10 ")
    if data:

        return render_template('dashboard.html',user_name_nav=session.get("username"),total_item=total_item,
                                        profit=profit,low_stock=low_stock,stock_cost=stock_cost,users=data)
    return render_template('dashboard.html',user_name_nav=session.get("username"),total_item=total_item,
                                            profit=profit,low_stock=low_stock,stock_cost=stock_cost)
```

7.5 Inventory Items

```
@app.route('/items',methods=['POST','GET'])
def items():
    email = session.get("email")
    user_id=check_query_data.check_signle_column('user','email',email)
    data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{user_id[0]} ORDER BY product_id")
    if data:

        return render_template("items.html",users=data,user_name_nav=session.get("username"))
    return render_template("items.html",user_name_nav=session.get("username"))
```

7.6 Deleting Items from Inventory

Users can delete the products from the system using the Delete stock function. After deleting the products, it will send the message like 'Product deleted, Success'.

```
@app.route('/delete/<name>')
def delete(name):
    email = session.get("email")
    user_id = check_query_data.check_signle_column('user','email',email)
    print(user_id)
    sql = f"DELETE FROM product_details_{user_id[0]} WHERE product_id='{escape(name)}'"
    print(sql)
    stmt = ibm_db.exec_immediate(con, sql)
    data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{user_id[0]}")
    if data:

        return render_template("items.html",users=data,user_name_nav=session.get("username"))
    return render_template("items.html",empty="Once item is purchase it will be shown here",
                                                    user_name_nav=session.get("username"))
```

7.7 Updating Items

Users can Update the details of the products from the system using the update stock function. After updating the products it will send the message like 'You have Successfully updated the products! !'.

```
elif (quantity == 0):
    status = 'Out of stock'
    send_mail.mail_out_of_stock(email,product_name)

sql = f"UPDATE product_details_{user_id[0]} SET supplier = '{supplier_name}', date = '{date}', product_name = '{product_name}',
        purchase_price = {int(purchase_price)}, selling_price = {int(selling_price)}, stock = {int(quantity)},
        total_selling_price = {int(total_selling_price)}, status = '{status}', low_stock = {int(low_stock)} WHERE product_id={escape(name)}"
ibm_db.exec_immediate(con, sql)


email = session.get("email")
user_id=check_query_data.check_signle_column('user','email',email)
data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{user_id[0]}")
if data:

    return render_template("items.html",users=data,user_name_nav=session.get("username"))
return render_template("items.html",empty="Once item is purchase it will be shown here",user_name_nav=session.get("username"))
```

```
@app.route('/edit_table/<name>',methods=['POST','GET'])
def edit_table(name):
    session["url"] = name

    return render_template("edit_table.html",empty="Once item is purchase it will be shown here",user_name_nav=session.get("username"))


@app.route('/item',methods=['POST','GET'])
def item():
    name = session.get("url")
    if request.method == 'POST':
        email = session.get("email")
        product_name=request.form.get('product_name').lower()
        supplier_name=request.form.get('Supplier_name')
        purchase_price=request.form.get('purchase_price')
        selling_price=request.form.get('selling_price')
        quantity=request.form.get('quantity')
        low_stock=request.form.get('low_stock')
        status='Instock'

        email = session.get("email")
        user_id = check_query_data.check_signle_column('user','email',email)
        data = check_query_data.check_signle_column(f'product_details_{user_id[0]}','product_id',escape(name))

        if (product_name == "" and supplier_name == "" and purchase_price == "" and selling_price == "" and quantity == "" and low_stock == ""):
            return render_template("edit_table.html",invalid_msg="Please fill all fields",user_name_nav=session.get("username"))

        if (supplier_name==""):
            supplier_name = data[1]

        date = data[2]
        if (product_name == ""):
             product_name = data[3]
        if (purchase_price == ""):
             purchase_price = data[4]
        if (selling_price == ""):
             selling_price = data[5]
        if (quantity == ""):
           quantity = data[6]
        total_selling_price = int(selling_price) * int(quantity)
        status = data[8]
        if(low_stock == ""):
            low_stock = data[9]

        if (int(quantity) <= int(low_stock) and int(quantity) > 1):
            status = "Low stock"
            send_mail.mail_low_stock(email,product_name,quantity,low_stock)
```

7.7 Purchase / Add items into Inventory

Users are able to insert the products into the system by using the Add stock function. After adding the products it will send the message like ' You have successfully added the products'. If the product already exists it will send the message like ' Product already exists !!

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

    if request.method == 'POST':
        email = session.get("email")
        product_name=request.form.get('product_name').lower()
        supplier_name=request.form.get('Supplier_name')
        purchase_price=request.form.get('purchase_price')
        selling_price=request.form.get('selling_price')
        quantity=request.form.get('quantity')
        low_stock=request.form.get('low_stock')
        status='Instock'

        if (email=="" or product_name=="" or supplier_name=="" or selling_price=="" or quantity=="" or status==""):
             return render_template("purchase_order.html",invalid_msg="Please Fill all details",user_name_nav=session.get("username"))

        user_id = check_query_data.check_signle_column('user','email',email)
        data = check_query_data.check_signle_column(f'product_details_{user_id[0]}','product_name',product_name)
        if data:
            return render_template("purchase_order.html",invalid_msg="Product name already exist",user_name_nav=session.get("username"))

        if (int(quantity) <= int(low_stock) and int(quantity) > 1):
            status = "Low stock"
            send_mail.mail_low_stock(email,product_name,quantity,low_stock)

        elif (quantity == 0):
            status = 'Out of stock'
            send_mail.mail_out_of_stock(email,product_name)

        insert_data_database.insert_item_table(supplier_name,product_name,purchase_price,selling_price,quantity,email,status,low_stock)


        return render_template("purchase_order.html",msg="Item added to purchase list",user_name_nav=session.get("username"))
    return render_template("purchase_order.html",user_name_nav=session.get("username"))
```

7.8 Sales

```
@app.route("/sales",methods=['POST','GET'])
def sales():
    admin_email = session["email"]
    if request.method == 'POST':
        admin_email = session.get("email")
        customer_name=request.form.get('customer_name')
        customer_email=request.form.get('customer_email')
        phone_number=request.form.get('phone_number')
        product_id=request.form.get('product_id')
        quantity=request.form.get('quantity')

        if (product_id=="" or customer_email== "" or customer_name == "" or phone_number=="" or quantity=="" ):
            return render_template("sales_page.html",error = "Plese fill all details",user_name_nav=session.get("username"))
        user_id = check_query_data.check_signle_column('user','email',admin_email)

        sql = f"SELECT * FROM product_details_{user_id[0]} where product_id = {product_id}"
        stmt = ibm_db.exec_immediate(con,sql)
        data = ibm_db.fetch_tuple(stmt)
        if data==False:
            return render_template("sales_page.html",invalid_msg="Product doesn't exist",user_name_nav=session.get("username"))
        status = data[8]
        if status  == 'out of stock':
            return render_template("sales_page.html",error="item out of stock",user_name_nav=session.get("username"))

        update = update_data_database.sale(admin_email,customer_name,customer_email,int(phone_number),product_id,int(quantity),user_id)

        data = fetch_query_data.fetch_data(f"SELECT * FROM SALES_DETAILS_{user_id[0]}")
        if update == 1 and data:
            return render_template("sales_page.html",users=data,msg="Item Sold",user_name_nav=session.get("username"))
        else:
            return render_template("sales_page.html",error=f"Low stock only {update} item left",user_name_nav=session.get("username"))
    user_id = check_query_data.check_signle_column('user','email',admin_email)
    data = fetch_query_data.fetch_data(f"SELECT * FROM SALES_DETAILS_{user_id[0]}")
    if data:
        return render_template("sales_page.html",users=data,user_name_nav=session.get("username"))
    return render_template("sales_page.html",user_name_nav=session.get("username"))
```

7.9 Send grid Email code for Forgot Password, Low Stock and Out of Stock

An account was created in SendGrid and apikey was generated for smtp connection. Using the api key generated, we accessed the smtp port with the help of python code and send an alert email. An alert email will be sent if the product quantity is less than 5 to the retailers

.

```
class send_mail:

    def mail(email):
        encoded_email = code.encode(email)
        encoded_email = f"{encoded_email}"
        encoded_email= encoded_email[2:len(encoded_email)-1]
        message = Mail(from_email='inventorymanagementvsgp@gmail.com',
            to_emails=email,
            subject='Reset password',
            html_content='<h3>Hello!, <br>A request has been received to change the password for your Inventory account</h3>
                    <a href="http://127.0.0.1:5000/forgot_password_verify/{}"><button type="submit" style="background-color:
                    #0583d2; border: none; color: white; padding: 12px 32px; text-align: center; text-decoration: none;
                    display: inline-block; font-size: 16px; border-radius: 5px">Reset Password</button>'.format(encoded_email))
        sg = SendGridAPIClient("'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")
        response = sg.send(message)
        print(response.status_code, response.body)
        print(response.status_code)
        print(response.body)
        print(response.headers)

    def mail_low_stock(email,product_name,stock,limit):
        encoded_email = code.encode(email)
        encoded_email = f"{encoded_email}"
        encoded_email= encoded_email[2:len(encoded_email)-1]
        message = Mail(from_email='inventorymanagementvsgp@gmail.com',
            to_emails=email,
            subject='Low stock alert mail',
            html_content='<h3>Hello!, <br> You recieved this alert because product- {} current stock is {} lower than the
                    threashold you have set.</h3><a href="http://127.0.0.1:5000"><button type="submit" style="background-color:
                    #0583d2; border: none; color: white; padding: 12px 32px; text-align: center; text-decoration: none;
                    display: inline-block; font-size: 16px; border-radius: 5px">Login</button>'.format(product_name,stock,limit))
        sg = SendGridAPIClient("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")
        response = sg.send(message)
        print(response.status_code, response.body)
        print(response.status_code)
        print(response.body)
        print(response.headers)

    def mail_out_of_stock(email,product_name):
        encoded_email = code.encode(email)
        encoded_email = f"{encoded_email}"
        encoded_email= encoded_email[2:len(encoded_email)-1]
        message = Mail(from_email='inventorymanagementvsgp@gmail.com',
            to_emails=email,
            subject='Product out of stock',
            html_content='<h3>Hello!, <br> You recieved this alert because product :{} is out of stock now.</h3><a href="http://127.0.0.1:5000">
                    <button type="submit" style="background-color: #0583d2; border: none; color: white; padding: 12px 32px; text-align:
                    enter; text-decoration: none; display: inline-block; font-size: 16px; border-radius: 5px">Login</button>'.format(product_name))
```

7.10. Database Schema

Table 1 - User

- This user table is used to store users details while registering
- Used to check details of users while login

Table 2- Product

- This product table used to store the details of the adding to the inventory
- This table is used to know the stock amount of product present in the inventory

8. TESTING

8.1 Test Cases

	TEST Scenarios
	Login
1	Verify user is able to see login page.
2	Verify user is able to login into apllication or not.
3	Verify login page elements
4	verify email is sent to user for reserting password
	Register
1	Verify if user is able to enter all the details and register
2	Verify if user isredirected to login page once registered
	Add products
1	Verify user is able to insert purchase details in purchase page
2	Verify Whether purchased products are add correctly to the item table in item page
	View products
1	Verify whether products can be viewed in item page
2	Verify whether products can be retrieved correctly from database
	Update Products
1	Verify user is able to update product detials in item page by clicking edit icon
2	Verify whether updated product detials is reflected correctly in items page
3	Verify if an alert email has been sent to user is the product quantity drops below low stock limit
	Delete Products
1	Verify user is able to delete products in item page
	Sale product
1	Verify user able to insert product sales detials in New salse page
2	Verify user able to see sales history section in sales page
	Dashboard
1	Verify dashboard analytics section shows total stock, low stock items count, sales profit and Total stock price
2	Verify dashboard low stock items section shows items in low stock
3	Verify the user is able to log out

8.2 User Acceptance Testing

The project has been tested extensively with a number of users. The users found the interface very easy to use. The web pages were minimalistic and attractive. There were no unnecessary details in web page. It was clean and simple that any new user could master. The data input format was also simple. The user can login to the system by entering username and password and they can add stock, delete stock, update stock,view stock, and insert purchase and sale details. Various inputs have been given by the users to test the consistency of the model. The model proved itself and all the users accepted the model as reliable and convenient.

9. RESULTS

9.1 Performance Metrics

The application that we have developed has better performance in speed. An email alert will be sent to user to reset password and if stock is low or out of stock the email will be sent to user. The application developed also performed well with no glitches or lag found during the testing phase

10. ADVANTAGES & DISADVANTAGES

Advantages:

- It helps to maintain the right amount of stocks
- It leads to a more organized warehouse
- It saves time and money
- A well-structured inventory management system leads to improved customer retention

Disadvantages:

- Increased space is need to hold the inventory
- Complexity
- High implementation costs

11. CONCLUSION

It is obvious that the manual inventory system needs to be replaced with an automated one that can shorten the time required, minimize human error, and increase inventory efficiency. The software of choice is also able to meet the specified demands intended to address the issue. The system can be made better in the future by including a place to store receipts and generate receipts for each transaction. Additionally, a profile area for each user may be created to store additional information about the users. Lastly, there is a feature that can tell the user when an item is getting close to its expiration date.

12. FUTURE SCOPE

There will be even more technological advancements in the field of inventories. To boost sales and draw customers, innovations ranging from artificial intelligence to inventory-free businesses are always being produced.

13.APPENDIX

Source Code

Backend code:

app.py:

```
 om flask import *
from flask_session import Session
from sql_calls import *
import ibm_db


app = Flask(__name__)


#connection to database
con = ibm_db.connect("DATABASE=bludb; HOSTNAME=ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=31321;
SECURITY=SSL; SSLServerCertificate=DigiCertGlobalRootCA.crt; UID=qxh83236;
PWD=nNnU0FLB0HNnzAzZ","","")


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


#register page
@app.route('/register')
def register():
    return render_template('register.html')


#dashboard page
@app.route('/dashboard')
def dashboard():
    email = session.get("email")
    total_item = dashboard_view.total_items(email)
```

```
    profit = dashboard_view.dashboard_profit(email)
    low_stock = dashboard_view.low_stock(email)
    stock_cost = dashboard_view.stock_cost(email)
    user_id=check_query_data.check_signle_column('user','email',email)
    data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{user_id[0]} WHERE stock <=10 ")
    if data:

        return render_template('dashboard.html',user_name_nav=session.get("username"),total_item=total_item,profit=profit,low_stock=low_stock,stock_cost=stock_cost,users=data)
    return render_template('dashboard.html',user_name_nav=session.get("username"),total_item=total_item,profit=profit,low_stock=low_stock,stock_cost=stock_cost)


#login_validator
@app.route('/login_validation',methods=['GET','POST'])
def login_validation():
    email=request.form.get('email').lower()
    password=request.form.get('password')
    username = check_query_data.check_signle_column('user','email',email)
    session["username"] = username[1]
    session["email"] = email


    #checking for email and password in db
    if (check_query_data.check_two_column(t='user',c1='email',d1=email,c2='password',d2=password)):
        email = session.get("email")


        total_item = dashboard_view.total_items(email)
        profit = dashboard_view.dashboard_profit(email)
        low_stock = dashboard_view.low_stock(email)
        stock_cost = dashboard_view.stock_cost(email)
```

```
        username = check_query_data.check_signle_column('user','email',email)
        data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{username[0]} WHERE stock <=10 ")
        if data:

            return render_template('dashboard.html',user_name_nav=username[1],total_item=total_item,profit=profit,low_stock=low_stock,stock_cost=stock_cost,users=data)
        return render_template('dashboard.html',user_name_nav=username[1],total_item=total_item,profit=profit,low_stock=low_stock,stock_cost=stock_cost)
    else:
        return render_template('login.html',invalid_msg='Invalid login credentials')


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

    name=request.form.get('name')
    company_name=request.form.get('company').lower()
    email=request.form.get('email').lower()
    password=request.form.get('password')
    session["email"] = email
    #checks already email and company_name exist in db
    company_exist = check_query_data.check_signle_column(t='user',c1='company_name',d1=company_name)
    email_exist = check_query_data.check_signle_column(t='user',c1='email',d1=email)

    company_msg = "Company already exist"
    email_msg = "Email already exists"

    #checks empty field and validates company_name and email
    if(len(name)==0 or len(company_name)==0 or len(email)==0 or len(password)==0):
```

```
        return render_template("register.html",field_empty="Please enter all fields")
    elif(company_exist and email_exist):
        return
render_template("register.html",company_msg=company_msg,email_msg=email_msg)
    elif(company_exist):
        return render_template("register.html",company_msg=company_msg)
    elif(email_exist):
        return render_template("register.html",email_msg=email_msg)


    #inserts user data into db user table
    insert_data_database.insert_user_table(name,email,password,company_name)
    create_table.item_table(email)
    create_table.sales_table(email)


    total_item = dashboard_view.total_items(email)
    profit = dashboard_view.dashboard_profit(email)
    low_stock = dashboard_view.low_stock(email)
    stock_cost = dashboard_view.stock_cost(email)
    username = check_query_data.check_signle_column('user','email',email)
    session["username"] = username[1]
    session["email"] = email
    data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{username[0]}
WHERE stock <=10 ")
    if data:


        return
render_template('dashboard.html',user_name_nav=username[1],total_item=total_item,profit=
profit,low_stock=low_stock,stock_cost=stock_cost,users=data)
    return
render_template("dashboard.html",user_name_nav=username[1],total_item=total_item,profit
=profit,low_stock=low_stock,stock_cost=stock_cost)


#checks email(account) exists or not
```

```
@app.route('/verify_email',methods=['POST','GET'])
def verify_email():
    if request.method == 'POST':
        email=request.form.get('email')
        email_exist = check_query_data.check_signle_column('user','email',email)

        #puts data into session
        session["email"] = email

        if (email_exist):
            send_mail.mail(email)
            return render_template('for_email_verify.html',field_empty='We have e-mailed your password reset link!')
        return render_template('for_email_verify.html',field_empty='We cannot find your email')

    return render_template('for_email_verify.html')

#forgot password
@app.route('/forgot_password_verify/<email>',methods=['POST','GET'])
def forgot_password_verify(email):
    if request.method == 'POST':
        decode_email = session.get("decode_email")
        new_password=request.form.get('password')
        cnf_password=request.form.get('cnf_password')
        if(len(new_password) == 0 and len(cnf_password) == 0):
            return render_template('forgot_password.html',invalid_msg='Please enter password')
        elif(new_password==cnf_password):
            update_data_database.update_singel_data(cnf_password,decode_email)
            return render_template('login.html')
        else:
            return render_template('forgot_password.html',invalid_msg='New password and confirm new password do not match')
```

```
    decode_email = code.decode(f"{email}")
    session["decode_email"] = decode_email
    return render_template("forgot_password.html")


@app.route('/items',methods=['POST','GET'])
def items():
    email = session.get("email")
    user_id=check_query_data.check_signle_column('user','email',email)
    data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{user_id[0]} ORDER
BY product_id")
    if data:


        return
render_template("items.html",users=data,user_name_nav=session.get("username"))
    return render_template("items.html",user_name_nav=session.get("username"))


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


    if request.method == 'POST':
        email = session.get("email")
        product_name=request.form.get('product_name').lower()
        supplier_name=request.form.get('Supplier_name')
        purchase_price=request.form.get('purchase_price')
        selling_price=request.form.get('selling_price')
        quantity=request.form.get('quantity')
        low_stock=request.form.get('low_stock')
        status='Instock'


        if (email=="" or product_name=="" or supplier_name=="" or selling_price=="" or
quantity=="" or status==""):
```

```
            return render_template("purchase_order.html",invalid_msg="Please Fill all details",user_name_nav=session.get("username"))


        user_id = check_query_data.check_signle_column('user','email',email)
        data = check_query_data.check_signle_column(f'product_details_{user_id[0]}','product_name',product_name)
        if data:
            return render_template("purchase_order.html",invalid_msg="Product name already exist",user_name_nav=session.get("username"))


        if (int(quantity) <= int(low_stock) and int(quantity) > 1):
            status = "Low stock"
            send_mail.mail_low_stock(email,product_name,quantity,low_stock)


        elif (quantity == 0):
            status = 'Out of stock'
            send_mail.mail_out_of_stock(email,product_name)


        insert_data_database.insert_item_table(supplier_name,product_name,purchase_price,selling_price,quantity,email,status,low_stock)



        return render_template("purchase_order.html",msg="Item added to purchase list",user_name_nav=session.get("username"))
    return render_template("purchase_order.html",user_name_nav=session.get("username"))


@app.route("/sales",methods=['POST','GET'])
def sales():
    admin_email = session["email"]
    if request.method == 'POST':
        admin_email = session.get("email")
        customer_name=request.form.get('customer_name')
```

```
    customer_email=request.form.get('customer_email')
    phone_number=request.form.get('phone_number')
    product_id=request.form.get('product_id')
    quantity=request.form.get('quantity')

    if (product_id=="" or customer_email== "" or customer_name == "" or phone_number=="" or quantity=="" ):
        return render_template("sales_page.html",error = "Plese fill all details",user_name_nav=session.get("username"))
    user_id = check_query_data.check_signle_column('user','email',admin_email)

    sql = f"SELECT * FROM product_details_{user_id[0]} where product_id = {product_id}"
    stmt = ibm_db.exec_immediate(con,sql)
    data = ibm_db.fetch_tuple(stmt)
    if data==False:
        return render_template("sales_page.html",invalid_msg="Product doesn't exist",user_name_nav=session.get("username"))
    status = data[8]
    if status  == 'out of stock':
        return render_template("sales_page.html",error="item out of stock",user_name_nav=session.get("username"))

    update = update_data_database.sale(admin_email,customer_name,customer_email,int(phone_number),product_id,int(quantity),user_id)

    data = fetch_query_data.fetch_data(f"SELECT * FROM SALES_DETAILS_{user_id[0]}")
    if update == 1 and data:
        return render_template("sales_page.html",users=data,msg="Item Sold",user_name_nav=session.get("username"))
    else:
        return render_template("sales_page.html",error=f"Low stock only {update} item left",user_name_nav=session.get("username"))
  user_id = check_query_data.check_signle_column('user','email',admin_email)
```

```
    data = fetch_query_data.fetch_data(f"SELECT * FROM SALES_DETAILS_{user_id[0]}")
    if data:
        return
render_template("sales_page.html",users=data,user_name_nav=session.get("username"))
    return render_template("sales_page.html",user_name_nav=session.get("username"))



@app.route('/delete/<name>')
def delete(name):
    email = session.get("email")
    user_id = check_query_data.check_signle_column('user','email',email)
    print(user_id)
    sql = f"DELETE FROM product_details_{user_id[0]} WHERE product_id='{escape(name)}'"
    print(sql)
    stmt = ibm_db.exec_immediate(con, sql)
    data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{user_id[0]}")
    if data:


        return
render_template("items.html",users=data,user_name_nav=session.get("username"))
    return render_template("items.html",empty="Once item is purchase it will be shown
here",user_name_nav=session.get("username"))



@app.route('/edit_table/<name>',methods=['POST','GET'])
def edit_table(name):
    session["url"] = name


    return render_template("edit_table.html",empty="Once item is purchase it will be shown
here",user_name_nav=session.get("username"))


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

```
    name = session.get("url")
    if request.method == 'POST':
        email = session.get("email")
        product_name=request.form.get('product_name').lower()
        supplier_name=request.form.get('Supplier_name')
        purchase_price=request.form.get('purchase_price')
        selling_price=request.form.get('selling_price')
        quantity=request.form.get('quantity')
        low_stock=request.form.get('low_stock')
        status='Instock'


        email = session.get("email")
        user_id = check_query_data.check_signle_column('user','email',email)
        data =
check_query_data.check_signle_column(f'product_details_{user_id[0]}','product_id',escape(n
ame))


        if (product_name == "" and supplier_name == "" and purchase_price == "" and
selling_price == "" and quantity == "" and low_stock == ""):
            return render_template("edit_table.html",invalid_msg="Please fill all
fields",user_name_nav=session.get("username"))


        if (supplier_name==""):
            supplier_name = data[1]


        date = data[2]
        if (product_name == ""):
            product_name = data[3]
        if (purchase_price == ""):
            purchase_price = data[4]
        if (selling_price == ""):
            selling_price = data[5]
        if (quantity == ""):
```

```
        quantity = data[6]
    total_selling_price = int(selling_price) * int(quantity)
    status = data[8]
    if(low_stock == ""):
        low_stock = data[9]

    if (int(quantity) <= int(low_stock) and int(quantity) > 1):
        status = "Low stock"
        send_mail.mail_low_stock(email,product_name,quantity,low_stock)

    elif (quantity == 0):
        status = 'Out of stock'
        send_mail.mail_out_of_stock(email,product_name)

    sql = f"UPDATE product_details_{user_id[0]} SET supplier = '{supplier_name}', date =
'{date}', product_name = '{product_name}', purchase_price = {int(purchase_price)},
selling_price = {int(selling_price)}, stock = {int(quantity)}, total_selling_price =
{int(total_selling_price)}, status = '{status}', low_stock = {int(low_stock)} WHERE
product_id={escape(name)}"
    ibm_db.exec_immediate(con, sql)

    email = session.get("email")
    user_id=check_query_data.check_signle_column('user','email',email)
    data = fetch_query_data.fetch_data(f"SELECT * FROM product_details_{user_id[0]}")
    if data:

        return
render_template("items.html",users=data,user_name_nav=session.get("username"))
    return render_template("items.html",empty="Once item is purchase it will be shown
here",user_name_nav=session.get("username"))

if __name__ == "__main__":
    #secret_key for session
```

```
    app.secret_key='asdfghjkl'
    app.config['SESSION_TYPE'] ='filesystem'
    Session().init_app(app)
    app.debug = True
    app.run(host='0.0.0.0', port=5000)
```

sql_callas.py:

```
import base64
import os
from flask import *
import ibm_db
import sendgrid
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail


from flask_session import Session


con = ibm_db.connect("DATABASE=bludb; HOSTNAME=ba99a9e6-d59e-4883-8fc0-
d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud; PORT=31321;
SECURITY=SSL; SSLServerCertificate=DigiCertGlobalRootCA.crt; UID=qxh83236;
PWD=nNnU0FLB0HNnzAzZ","","")


class fetch_query_data:
    def fetch_data(sql):
        list = []
        stmt = ibm_db.exec_immediate(con, sql)
        dictionary = ibm_db.fetch_both(stmt)
        while dictionary != False:
            list.append(dictionary)
            dictionary = ibm_db.fetch_both(stmt)
        if list:
```

```
            return list
        else:
            print("Data not fetched.....")

class check_query_data:

    def check_signle_column(t,c1,d1):
        sql = f"SELECT * FROM {t} WHERE {c1} ='{d1}';"
        return check_query_data.output(sql)

    def check_two_column(t,c1,d1,c2,d2):
        sql = f"SELECT * FROM {t} WHERE {c1} ='{d1}' AND {c2}='{d2}';"
        return check_query_data.output(sql)

    def output(sql):
        stmt = ibm_db.exec_immediate(con,sql)
        return ibm_db.fetch_tuple(stmt)


class insert_data_database:
    #inserting data into user_table
    def insert_user_table(name,email,password,company_name):
        inser_sql = f"INSERT INTO user (name,email,password,company_name ) VALUES
(?,?,?,?);"
        stmt = ibm_db.prepare(con, inser_sql)
        ibm_db.bind_param(stmt, 1, name)
        ibm_db.bind_param(stmt, 2, email)
        ibm_db.bind_param(stmt, 3, password)
        ibm_db.bind_param(stmt, 4, company_name)
        ibm_db.execute(stmt)

    #inserting data into product_table
```

```
    def
insert_item_table(supplier,product_name,purchase_price,selling_price,stock,email,status,low
_stock):
        user_id=check_query_data.check_signle_column('user','email',email)
        sql = f"INSERT INTO
product_details_{user_id[0]}(supplier,product_name,purchase_price,selling_price,stock,total_
selling_price,status,low_stock)VALUES (?,?,?,?,?,?,?,?,?);"
        stmt = ibm_db.prepare(con, sql)
        ibm_db.bind_param(stmt, 1, supplier)
        ibm_db.bind_param(stmt, 2, product_name)
        ibm_db.bind_param(stmt, 3, purchase_price)
        ibm_db.bind_param(stmt, 4, selling_price)
        ibm_db.bind_param(stmt, 5, stock)
        t_selling_price = int(selling_price)*int(stock)
        ibm_db.bind_param(stmt, 6, t_selling_price)
        ibm_db.bind_param(stmt,7,status)
        ibm_db.bind_param(stmt,8,low_stock)
        ibm_db.execute(stmt)




class update_data_database:


    def update_singel_data(new_data,email):
        inser_sql = f"UPDATE user SET password='{new_data}' WHERE email='{email}';"
        ibm_db.exec_immediate(con, inser_sql)




    def
sale(admin_email,customer_name,customer_email,phone_number,product_id,quantity,user_i
d):
```

```
        data = check_query_data.check_signle_column(f'product_details_{user_id[0]}','product_id',product_id)

        product_id = data[0]

        product_name = data[3]

        purchase_price = data[4]

        selling_price= data[5]

        stock = data[6]

        total_selling_price = data[7]

        low_stock_limit = data[9]

        billing_amount = selling_price*quantity

        total_selling_price -= billing_amount


        if(quantity<=stock):

            stock -= quantity

            sql = f"INSERT INTO sales_details_{user_id[0]}(customer_name,customer_email,product_id,phone_number,product_name,quantity,billing_amount)VALUES (?,?,?,?,?,?,?);"

            stmt = ibm_db.prepare(con, sql)

            ibm_db.bind_param(stmt, 1, customer_name)

            ibm_db.bind_param(stmt, 2, customer_email)

            ibm_db.bind_param(stmt, 3, product_id)

            ibm_db.bind_param(stmt, 4, phone_number)

            ibm_db.bind_param(stmt, 5, product_name)

            ibm_db.bind_param(stmt, 6, quantity)

            ibm_db.bind_param(stmt, 7, billing_amount)

            ibm_db.execute(stmt)


            if (stock > low_stock_limit):

                status = 'Instock'

            elif (stock <= low_stock_limit and stock > 1):

                status = 'Low stock'

                send_mail.mail_low_stock(admin_email,product_name,stock,low_stock_limit)
```

```
        elif (stock == 0):
            status = 'out of stock'
            send_mail.mail_out_of_stock(admin_email,product_name)

        updata_items =f"UPDATE product_details_{user_id[0]} SET total_selling_price=
{total_selling_price}, stock = {stock}, status = '{status}' WHERE product_id={product_id}"
        ibm_db.exec_immediate(con, updata_items)
        existing_profit = user_id[5]
        if existing_profit == None:
            existing_profit = 0
        profit = (int(existing_profit)+((selling_price*quantity)-(purchase_price*quantity)))
        profit_sql = f"UPDATE user SET profit = {profit} where user_id ={user_id[0]}"
        ibm_db.exec_immediate(con, profit_sql)

        return 1
    else:
        return stock




class create_table:
    def item_table(email):
        user_id=check_query_data.check_signle_column('user','email',email)
        sql=f"CREATE TABLE product_details_{user_id[0]}(product_id BIGINT NOT NULL
GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),supplier
VARCHAR(50),date timestamp DEFAULT CURRENT_TIMESTAMP,product_name
VARCHAR(50) NOT NULL UNIQUE,purchase_price BIGINT,selling_price BIGINT,stock
BIGINT,total_selling_price BIGINT,status VARCHAR(20),low_stock BIGINT,PRIMARY KEY
(product_id));"
        ibm_db.exec_immediate(con, sql)

    def sales_table(admin_email):
```

```
        user_id=check_query_data.check_signle_column('user','email',admin_email)
        sql = f"CREATE TABLE sales_details_{user_id[0]}(sales_id BIGINT NOT NULL
GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1),customer_name
VARCHAR(50),customer_email VARCHAR(50),product_id BIGINT,phone_number BIGINT,date
timestamp DEFAULT CURRENT_TIMESTAMP,product_name VARCHAR(50),quantity
BIGINT,billing_amount BIGINT,PRIMARY KEY (sales_id));"
        ibm_db.exec_immediate(con, sql)


    def low_stock(admin_email):
        user_id=check_query_data.check_signle_column('user','email',admin_email)
        sql = f"CREATE TABLE low_stock_{user_id[0]}"


class send_mail:


    def mail(email):
        encoded_email = code.encode(email)
        encoded_email = f"{encoded_email}"
        encoded_email= encoded_email[2:len(encoded_email)-1]
        message = Mail(from_email='inventorymanagementvsgp@gmail.com',
            to_emails=email,
            subject='Reset password',
            html_content='<h3>Hello!, <br>A request has been received to change the password
for your Inventory account</h3><a
href="http://127.0.0.1:5000/forgot_password_verify/{}"><button type="submit"
style="background-color: #0583d2; border: none; color: white; padding: 12px 32px; text-align:
center; text-decoration: none; display: inline-block; font-size: 16px; border-radius: 5px">Reset
Password</button>'.format(encoded_email))
        sg =
SendGridAPIClient("'SG.OX9YmnKiQXq9Q09FVUoGjw.KyAYHMM5H8SIXUl5yr7jRXnIFLguPX
zfRAhZ8jiiqi0")
        response = sg.send(message)
        print(response.status_code, response.body)
        print(response.status_code)
        print(response.body)
        print(response.headers)
```

```
    def mail_low_stock(email,product_name,stock,limit):
        encoded_email = code.encode(email)
        encoded_email = f"{encoded_email}"
        encoded_email= encoded_email[2:len(encoded_email)-1]
        message = Mail(from_email='inventorymanagementvsgp@gmail.com',
            to_emails=email,
            subject='Low stock alert mail',
            html_content='<h3>Hello!, <br> You recieved this alert because product- {} current stock is {} lower than the threashold you have set.</h3><a href="http://127.0.0.1:5000"><button type="submit" style="background-color: #0583d2; border: none; color: white; padding: 12px 32px; text-align: center; text-decoration: none; display: inline-block; font-size: 16px; border-radius: 5px">Login</button>'.format(product_name,stock,limit))
        sg = SendGridAPIClient("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")
        response = sg.send(message)
        print(response.status_code, response.body)
        print(response.status_code)
        print(response.body)
        print(response.headers)


    def mail_out_of_stock(email,product_name):
        encoded_email = code.encode(email)
        encoded_email = f"{encoded_email}"
        encoded_email= encoded_email[2:len(encoded_email)-1]
        message = Mail(from_email='inventorymanagementvsgp@gmail.com',
            to_emails=email,
            subject='Product out of stock',
            html_content='<h3>Hello!, <br> You recieved this alert because product :{} is out of stock now.</h3><a href="http://127.0.0.1:5000"><button type="submit" style="background-color: #0583d2; border: none; color: white; padding: 12px 32px; text-align: center; text-decoration: none; display: inline-block; font-size: 16px; border-radius: 5px">Login</button>'.format(product_name))
```

```
        sg =
SendGridAPIClient("SG.hiAlDVJvT_aOaxETCIYymg.9efkWWoVZoC4ksLZQ6LhwZlDkb3nBnL
NNuWpn6x2DKc")
        response = sg.send(message)
        print(response.status_code, response.body)
        print(response.status_code)
        print(response.body)
        print(response.headers)



class code:
    def encode(data):
        encode = data.encode("utf-8")
        return base64.b16encode(encode)


    def decode(data):
        return base64.b16decode(data).decode("utf-8")


class check:
    def check(email):
        user_id=check_query_data.check_signle_column('user','email',email)
        return check_query_data.output(f"SELECT count(product_id) FROM
product_details_{user_id[0]}")




class dashboard_view:
    def total_items(email):
        count = check.check(email)
        return count[0]


    def dashboard_profit(email):
        sql = f"SELECT profit FROM user WHERE email = '{email}'"
```

```
    stmt = ibm_db.exec_immediate(con,sql)
    data = ibm_db.fetch_tuple(stmt)
    temp = 0
    if data[0] == None:
        return temp
    return data[0]



def low_stock(email):
    user_id = check_query_data.check_signle_column('user','email',email)
    sql = f"SELECT count(product_id) FROM product_details_{user_id[0]} WHERE stock <=
10"
    stmt = ibm_db.exec_immediate(con,sql)
    data = ibm_db.fetch_tuple(stmt)
    return data[0]

def stock_cost(email):
    user_id = check_query_data.check_signle_column('user','email',email)
    print(user_id)
    sql = f"SELECT sum(total_selling_price) FROM product_details_{user_id[0]}"
    stmt = ibm_db.exec_immediate(con,sql)
    data = ibm_db.fetch_tuple(stmt)
    temp = 0
    if data[0] == None:
        return temp
    return data[0]

def dashboard_details(email):
    total_item = dashboard_view.total_items(email)
    profit = dashboard_view.dashboard_profit(email)
    low_stock = dashboard_view.low_stock(email)
    stock_cost = dashboard_view.stock_cost(email)
```

```
    return
render_template('dashboard.html',total_item=total_item,profit=profit,low_stock=low_stock,st
ock_cost=stock_cost)
```

login..html:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Login</title>
    <link rel="shortcut icon" href="/static/img/icons/icon-48x48.png" />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
Zenh87qX5JnK2Jl0vWa8Ck2rdkQ2Bzep5IDxbcnCeuOxjzrPF/et3URy9Bv1WTRi"
crossorigin="anonymous">
    <link rel="stylesheet" type="text/css" href="{{url_for('static',filename='css/style.css')}}">
    <link rel="stylesheet" type="text/css" href="/static/css/app.css">
    <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&display=swap"
rel="stylesheet">
  </head>
  <body>
    <main class="d-flex w-100">
      <div class="container d-flex flex-column">
        <div class="row vh-100">
          <div class="col-sm-10 col-md-8 col-lg-6 mx-auto d-table h-100">
            <div class="d-table-cell align-middle">

              <div class="text-center mt-4">
                <h1 class="h2">Welcome back</h1>
                <p class="lead">
                  Sign in to your account to continue
```

```
    </p>
   </div>

   <div class="card">
    <div class="card-body">
     <div class="m-sm-4">
      <form method="POST" action="/login_validation">
       <p style="color: red;display: inline;">{{invalid_msg}}</p>
       <p style="color: red;display: inline;">{{email_send}}</p>
       <div class="mb-3">
        <label class="form-label">Email</label>
        <input class="form-control form-control-lg" type="email" name="email"
placeholder="Enter your email" />
       </div>
       <div class="mb-3">
        <label class="form-label">Password</label>
        <input class="form-control form-control-lg" type="password" name="password"
placeholder="Enter your password" />
        <small>
         <a href="/verify_email">Forgot password</a>
        </small>
        <small style="float: right;">
         <a href="/register">Create account</a>
        </small>
       </div>
       <div>
  </label>
       </div>
       <div class="text-center mt-3">
        <button type="submit" class="btn btn-lg btn-primary">Sign in</button>
       </div>
      </form>
```

```
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
  </main>
  <script src="/static/js/app.js"></script>
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
OERcA2EqjJCMA+/3y+gxIOqMEjwtxJY7qPCqsdltbNJuaOe923+mo//f6V8Qbsw3"
crossorigin="anonymous"></script>
  <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.6/dist/umd/popper.min.js"
integrity="sha384-
oBqDVmMz9ATKxIep9tiCxS/Z9fNfEXiDAYTujMAeBAsjFuCZSmKbSSUnQlmh/jp3"
crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.min.js"
integrity="sha384-
IDwe1+LCz02ROU9k972gdyvl+AESN10+x7tBKgc9I5HFtuNz0wWnPclzo6p9vxnk"
crossorigin="anonymous"></script>
 </body>
</html>
```

forgot_password.html:

```
<!doctype html>
<html lang="en">
```

```
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <title>Forgot password</title>
  <link rel="shortcut icon" href="/static/img/icons/icon-48x48.png" />

  <link rel="stylesheet" type="text/css" href="/static/css/app.css">
  <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&display=swap"
rel="stylesheet">
</head>
<body>
  <main class="d-flex w-100">
    <div class="container d-flex flex-column">
      <div class="row vh-100">
        <div class="col-sm-10 col-md-8 col-lg-6 mx-auto d-table h-100">
          <div class="d-table-cell align-middle">

            <div class="text-center mt-4">
              <h1 class="h2">Get started</h1>
              <p class="lead">
                Start creating the best possible user experience for you customers.
              </p>
            </div>

            <div class="card">
              <div class="card-body">
                <div class="m-sm-4">
                  <form class="form" method="POST" action="/forgot_password_verify/none">
                    <p style="color: red;display: inline;">{{invalid_msg}}</p>
                    <p style="color: red;display: inline;">{{field_empty}}</p>
```

```
                <div class="mb-3">
                  <label class="form-label">New Password</label>
                  <input class="form-control form-control-lg" type="password"
name="password" placeholder="Enter your password" />
                </div>
                <div class="mb-3">

                  <label class="form-label">Confirm New Password</p></label>
                  <input class="form-control form-control-lg" type="password"
name="cnf_password" placeholder="Enter password" />
                </div>
                <small>
                  <a href="/" >Back</a>
                </small>
                <div class="text-center mt-3">
                  <button type="submit" class="btn btn-lg btn-primary">Change
Password</button>
                </div>
              </form>
            </div>
          </div>
        </div>

      </div>
     </div>
    </div>
   </main>
   <script src="/static/js/app.js"></script>

  </body>
```

```
</html>
```

Register.html:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Register</title>
    <link rel="shortcut icon" href="/static/img/icons/icon-48x48.png" />

    <link rel="stylesheet" type="text/css" href="/static/css/app.css">
    <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&display=swap"
rel="stylesheet">
  </head>
  <body>
    <main class="d-flex w-100">
      <div class="container d-flex flex-column">
        <div class="row vh-100">
          <div class="col-sm-10 col-md-8 col-lg-6 mx-auto d-table h-100">
            <div class="d-table-cell align-middle">

              <div class="text-center mt-4">
                <h1 class="h2">Get started</h1>
                <p class="lead">
                  Start creating the best possible user experience for you customers.
                </p>
              </div>
```

```
<div class="card">
  <div class="card-body">
    <div class="m-sm-4">
      <form class="form" method="POST" action="/register_validation">
        <p style="color: red;display: inline;">{{field_empty}}</p>
        <div class="mb-3">
          <label class="form-label">Name</label>
          <input class="form-control form-control-lg" type="text" name="name"
placeholder="Enter your name" />
        </div>
        <div class="mb-3">
          <label class="form-label">Company  <p style="color: red;display:
inline;">{{company_msg}}</p></label>
          <input class="form-control form-control-lg" type="text" name="company"
placeholder="Enter your company name" />
        </div>
        <div class="mb-3">
          <label class="form-label">Email  <p style="color: red;display:
inline;">{{email_msg}}</p></label>
          <input class="form-control form-control-lg" type="email" name="email"
placeholder="Enter your email" />
        </div>
        <div class="mb-3">

          <label class="form-label">Password</p></label>
          <input class="form-control form-control-lg" type="password" name="password"
placeholder="Enter password" />
        </div>
        <small>
          <a href="/" ><p style="display: inline;color: black;">Already have an
account?</p>  LOG IN</a>
        </small>
        <div class="text-center mt-3">
```

```
                <button type="submit" class="btn btn-lg btn-primary">Sign up</button>
              </div>
            </form>
          </div>
        </div>
      </div>

    </div>
   </div>
  </div>
 </main>
 <script src="/static/js/app.js"></script>
</body>
</html>
```

Dashboard.html:

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta
   name="viewport"
   content="width=device-width, initial-scale=1, shrink-to-fit=no"
  />
  <title>Dashboard</title>
  <link rel="stylesheet" type="text/css" href="/static/css/app.css">
  <link rel="stylesheet" type="text/css" href="/static/css/items.css">
  <link rel="stylesheet" type="text/css" href="/static/css/app.css" />
  <link
   href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&display=swap
"
```

```
    rel="stylesheet"
   />
 </head>

<body>
 <div class="wrapper">
  <nav id="sidebar" class="sidebar js-sidebar">
   <div class="sidebar-content js-simplebar">
    <a class="sidebar-brand" href="/dashboard">
     <span class="align-middle">Inventory</span>
    </a>

    <ul class="sidebar-nav">
     <li class="sidebar-header">
      Pages
     </li>

     <li class="sidebar-item active">
      <a class="sidebar-link" href="/dashboard">
       <i class="align-middle" data-feather="sliders"></i> <span class="align-
middle">Dashboard</span>
      </a>
     </li>

     <li class="sidebar-item ">
      <a class="sidebar-link" href="/items">
       <i class="align-middle" data-feather="shopping-bag"></i> <span class="align-
middle">Items</span>
      </a>
      <li class="sidebar-item ">
      <a class="sidebar-link" href="/purchase_order">
```

```
          <i class="align-middle" data-feather="box"></i> <span class="align-middle">Purchase items</span>
        </a>
        <li class="sidebar-item ">
        <a class="sidebar-link" href="/sales">
          <i class="align-middle" data-feather="grid"></i> <span class="align-middle">New Sales</span>
        </a>
    </nav>

    <div class="main">
      <nav class="navbar navbar-expand navbar-light navbar-bg">
       <a class="sidebar-toggle js-sidebar-toggle">
        <i class="hamburger align-self-center"></i>
       </a>

       <div class="navbar-collapse collapse">
        <ul class="navbar-nav navbar-align">
         </li>
         <li class="nav-item dropdown">
          <a
           class="nav-icon dropdown-toggle d-inline-block d-sm-none"
           href="#"
           data-bs-toggle="dropdown">
           <i class="align-middle" data-feather="settings"></i>
          </a>

          <a
           class="nav-link dropdown-toggle d-none d-sm-inline-block"
           href="#"
           data-bs-toggle="dropdown"
          >
```

```
            <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="feather feather-user align-middle me-2"><path d="M20 21v-2a4 4 0 0 0-4-4H8a4 4 0 0 0-4 4v2"></path><circle cx="12" cy="7" r="4"></circle></svg>
            <span class="text-dark">{{user_name_nav}}</span>
          </a>
          <div class="dropdown-menu dropdown-menu-end">
            <div class="dropdown-divider"></div>
            <a class="dropdown-item" href="/">Log out</a>
          </div>
        </li>
      </ul>
    </div>
  </nav>

  <main class="content">
    <div class="container-fluid p-0">
      <h1 class="h3 mb-3"><strong>Analytics</strong> Dashboard</h1>

      <div class="row">
        <div class="col">
          <div class="w-100">
            <div class="row" >
              <div class="col-sm-6">
                <div class="card" >
                  <div class="card-body">
                    <div class="row">
                      <div class="col mt-0">
                        <h5 class="card-title">Total Items</h5>
                      </div>
                      <div class="col-auto">
                        <div class="stat text-primary">
```

```
          <i class="align-middle" data-feather="box"></i>
        </div>
      </div>
    </div>
    <h1 class="mt-1 mb-3">{{total_item}}</h1>
    <div class="mb-0">
      <span class="text-danger">
        <i class="mdi mdi-arrow-bottom-right"></i
      ></span>
      <span class="text-muted"></span>
    </div>
  </div>
</div>
<div class="card">
  <div class="card-body">
    <div class="row">
      <div class="col mt-0">
        <h5 class="card-title">Low Stock Items</h5>
      </div>

      <div class="col-auto">
        <div class="stat text-primary">
          <i class="align-middle" data-feather="box"></i>
        </div>
      </div>
    </div>
    <h1 class="mt-1 mb-3">{{low_stock}}</h1>
    <div class="mb-0">
      <span class="text-success">
        <i class="mdi mdi-arrow-bottom-right"></i
      ></span>
```

```
      <span class="text-muted"></span>
     </div>
    </div>
   </div>
  </div>
  <div class="col-sm-6">
   <div class="card">
    <div class="card-body">
     <div class="row">
      <div class="col mt-0">
       <h5 class="card-title">₹ Sales Profit</h5>
      </div>

      <div class="col-auto">
       <div class="stat text-primary">
        <i
         class="align-middle"
         data-feather="dollar-sign"
        ></i>
       </div>
      </div>
     </div>
     <h1 class="mt-1 mb-3">{{profit}}</h1>
     <div class="mb-0">
      <span class="text-success">
       <i class="mdi mdi-arrow-bottom-right"></i
      ></span>
      <span class="text-muted"></span>
     </div>
    </div>
   </div>
```

```
        <div class="card">
          <div class="card-body">
            <div class="row">
              <div class="col mt-0">
                <h5 class="card-title">₹ Stock Cost</h5>
              </div>

              <div class="col-auto">
                <div class="stat text-primary">
                  <i
                    class="align-middle"
                    data-feather="dollar-sign"
                  ></i>
                </div>
              </div>
            </div>
            <h1 class="mt-1 mb-3">{{stock_cost}}</h1>
            <div class="mb-0">
              <span class="text-danger">
                <i class="mdi mdi-arrow-bottom-right"></i
              ></span>
              <span class="text-muted"></span>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

```
<div class="row">
<div class="col-12">
  <div class="card">
    <div class="card-header">
      <h1 class="h3 mb-3"><strong>Low Stock</strong> Items</h1>
      <table>
        <thead style="background-color: #222e3c;">
        <tr class="table100-head">
        <th class="column1">Id</th>
        <th class="column2">Supplier</th>
        <th class="column3">Date</th>
        <th class="column4">Product Name</th>
        <th class="column5">Purchase Price</th>
        <th class="column6">sale price</th>
        <th class="column7">Stock</th>
        <th class="column9">Total sale Price</th>
        <th class="column10">status</th>


        </thead>
        <tbody>
        {% for row in users %}
        <tr >
        <td class="column1">{{row["PRODUCT_ID"]}}</td>
        <td class="column2">{{row["SUPPLIER"]}}</td>
        <td class="column3">{{row["DATE"]}}</td>
        <td class="column4">{{row["PRODUCT_NAME"]}}</td>
        <td class="column5">{{row["PURCHASE_PRICE"]}}</td>
        <td class="column6">{{row["SELLING_PRICE"]}}</td>
        <td class="column7">{{row["STOCK"]}}</td>
        <td class="column9">{{row["TOTAL_SELLING_PRICE"]}}</td>
```

```
                <td class="column10">{{row["STATUS"]}}</td>

                </tr>
                {% endfor %}
                </tbody>
                </table>
              </div>
              <div class="card-body">
              </div>
            </div>
          </div>
        </div>
        </div>
      </main>
    </div>
  </div>

  <script src="/static/js/app.js"></script>
 </body>
</html>
```

Item.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="Responsive Admin &amp;amp; Dashboard Template based on Bootstrap 5">
  <meta name="author" content="AdminKit">
```

```
    <meta name="keywords" content="adminkit, bootstrap, bootstrap 5, admin, dashboard, template, responsive, css, sass, html, theme, front-end, ui kit, web">

    <link rel="shortcut icon" href="img/icons/icon-48x48.png" />

    <title>Items</title>

    <link rel="stylesheet" type="text/css" href="/static/css/app.css">
    <link rel="stylesheet" type="text/css" href="/static/css/items.css">
    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&display=swap" rel="stylesheet">

</head>

<body>
    <div class="wrapper">
        <nav id="sidebar" class="sidebar js-sidebar">
            <div class="sidebar-content js-simplebar">
                <a class="sidebar-brand" href="/dashboard">
          <span class="align-middle">Inventory</span>
        </a>

                <ul class="sidebar-nav">
                    <li class="sidebar-header">
                        Pages
                    </li>

                    <li class="sidebar-item ">
                        <a class="sidebar-link" href="/dashboard">
            <i class="align-middle" data-feather="sliders"></i> <span class="align-middle">Dashboard</span>
          </a>
```

```
            </li>


            <li class="sidebar-item active">
              <a class="sidebar-link" href="/items">
        <i class="align-middle" data-feather="shopping-bag"></i> <span class="align-middle">Items</span>
        </a>
        <li class="sidebar-item ">
              <a class="sidebar-link" href="/purchase_order">
        <i class="align-middle" data-feather="box"></i> <span class="align-middle">Purchase items</span>
        </a>
        <li class="sidebar-item ">
              <a class="sidebar-link" href="/sales">
        <i class="align-middle" data-feather="grid"></i> <span class="align-middle">New Sales</span>
        </a>
      </nav>


    <div class="main">
      <nav class="navbar navbar-expand navbar-light navbar-bg">
       <a class="sidebar-toggle js-sidebar-toggle">
        <i class="hamburger align-self-center"></i>
       </a>


       <div class="navbar-collapse collapse">
        <ul class="navbar-nav navbar-align">
         </li>
         <li class="nav-item dropdown">
          <a
            class="nav-icon dropdown-toggle d-inline-block d-sm-none"
            href="#"
```

```
            data-bs-toggle="dropdown">
            <i class="align-middle" data-feather="settings"></i>
          </a>

          <a
            class="nav-link dropdown-toggle d-none d-sm-inline-block"
            href="#"
            data-bs-toggle="dropdown"
          >
            <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="feather feather-user align-middle me-2"><path d="M20 21v-2a4 4 0 0 0-4-4H8a4 4 0 0 0-4 4v2"></path><circle cx="12" cy="7" r="4"></circle></svg>
            <span class="text-dark">{{user_name_nav}}</span>
          </a>
          <div class="dropdown-menu dropdown-menu-end">
            <div class="dropdown-divider"></div>
            <a class="dropdown-item" href="/">Log out</a>
          </div>
        </li>
      </ul>
    </div>
  </nav>

    <main class="content">
      <div class="container-fluid p-0">
        <h1 class="h3 mb-3">Items In Inventory</h1>

        <div class="row">
          <div class="col-12">
            <div class="card">
              <div class="card-header">
```

```
<p>{{empty}}<p1>
<table>
   <thead style="background-color: #222e3c;">
   <tr class="table100-head">
   <th class="column1">Id</th>
   <th class="column2">Supplier</th>
   <th class="column3">Date</th>
   <th class="column4">Product Name</th>
   <th class="column5">Purchase Price</th>
   <th class="column6">sale price</th>
   <th class="column7">Stock</th>
   <th class="column9">Total sale Price</th>
   <th class="column10">status</th>
   <th class="column10">Low stock limit</th>
   <th class="column11">Actions</th>

   </thead>
   <tbody>
   {% for row in users %}
   <tr >
   <td class="column1">{{row["PRODUCT_ID"]}}</td>
   <td class="column2">{{row["SUPPLIER"]}}</td>
   <td class="column3">{{row["DATE"]}}</td>
   <td class="column4">{{row["PRODUCT_NAME"]}}</td>
   <td class="column5">{{row["PURCHASE_PRICE"]}}</td>
   <td class="column6">{{row["SELLING_PRICE"]}}</td>
   <td class="column7">{{row["STOCK"]}}</td>
   <td class="column9">{{row["TOTAL_SELLING_PRICE"]}}</td>
   <td class="column10">{{row["STATUS"]}}</td>
   <td class="column10">{{row["LOW_STOCK"]}}</td>
   <td class="column11">
```

```
                        <a class="edit" title="Edit" data-toggle="tooltip"
style="color:#277BC0;display: inline;" href="/edit_table/{{row['PRODUCT_ID']}}">


                            <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round" class="feather feather-edit align-middle me-2">

                                <path d="M11 4H4a2 2 0 0 0-2 2v14a2 2 0 0 0 2 2h14a2 2 0 0 0 2-
2v-7"></path>

                                <path d="M18.5 2.5a2.121 2.121 0 0 1 3 3L12 15l-4 1 1-4 9.5-
9.5z"></path></svg>

                                <span class="align-middle"></span></a>


                        <a class="delete" title="Delete" data-toggle="tooltip"
style="color:#FF8585; display: inline;" href="/delete/{{row['PRODUCT_ID']}}">

                            <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24"
viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-
linecap="round" stroke-linejoin="round" class="feather feather-trash-2 align-middle me-2">

                                <polyline points="3 6 5 6 21 6"></polyline><path d="M19 6v14a2
2 0 0 1-2 2H7a2 2 0 0 1-2-2V6m3 0V4a2 2 0 0 1 2-2h4a2 2 0 0 1 2 2v2"></path>

                                <line x1="10" y1="11" x2="10" y2="17"></line><line x1="14"
y1="11" x2="14" y2="17"></line></svg></a>

                        </td>

                        </tr>

                        {% endfor %}

                        </tbody>


                        </table>

                    </div>

                    <div class="card-body">

                    </div>

                </div>

            </div>

        </div>
```

```
            </div>
          </main>
        </div>
    </div>

    <script src="/Inventory management system/static/js/app.js"></script>
    <script src="/static/js/app.js"></script>
</body>
</html>
```

edit_table.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="Responsive Admin &amp;amp; Dashboard Template based on Bootstrap 5">
    <meta name="author" content="AdminKit">
    <meta name="keywords" content="adminkit, bootstrap, bootstrap 5, admin, dashboard, template, responsive, css, sass, html, theme, front-end, ui kit, web">

    <link rel="shortcut icon" href="img/icons/icon-48x48.png" />

    <title>Items</title>

    <link rel="stylesheet" type="text/css" href="/static/css/app.css">
    <link rel="stylesheet" type="text/css" href="/static/css/add_prod.css">
    <link rel="stylesheet" type="text/css" href="/static/css/items.css">
    <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&display=swap" rel="stylesheet">
```

```
</head>

<style>

   .dropdown {
     position: relative;
     display: inline-block;

   }

   .dropdown-content {
     display: none;
     position: absolute;
     background-color: #f1f1f1;
     min-width: 200px;
     box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
     z-index: 1;
   }

   .dropdown-content a {
     color: black;
     padding: 12px 16px;
     text-decoration: none;
     display: block;
   }

   .dropdown-content a:hover {background-color: #ddd;}

   .dropdown:hover .dropdown-content {display: block;}
```

```
   /* toggle btn */

    .switch {
  position: relative;
  display: inline-block;
  width: 150px;
  height: 34px;
}

.switch input {display:none;}

.slider {
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #c73c3c;
  -webkit-transition: .4s;
  transition: .4s;
}

.slider:before {
  position: absolute;
  content: "";
  height: 26px;
  width: 26px;
  left: 4px;
  bottom: 4px;
  background-color: white;
```

```
  -webkit-transition: .4s;
  transition: .4s;
}

input:checked + .slider {
  background-color: #2ab933bd;
}

input:focus + .slider {
  box-shadow: 0 0 1px #2196F3;
}

input:checked + .slider:before {
  -webkit-transform: translateX(55px);
  -ms-transform: translateX(55px);
  transform: translateX(117px);
}

/*------ ADDED CSS ---------*/
.on
{
  display: none;
}

.on, .off
{
  color: white;
  position: absolute;
  transform: translate(-50%,-50%);
  top: 50%;
  left: 50%;
```

```
  font-size: 13px;
  font-family: Verdana, sans-serif;
}

input:checked+ .slider .on
{display: block;}

input:checked + .slider .off
{display: none;}

/*--------- END --------*/

/* Rounded sliders */
.slider.round {
  border-radius: 34px;
}

.slider.round:before {
  border-radius: 50%;}



</style>

<body>
   <div class="wrapper">
      <nav id="sidebar" class="sidebar js-sidebar">
         <div class="sidebar-content js-simplebar">
            <a class="sidebar-brand" href="/dashboard">
        <span class="align-middle">Inventory</span>
      </a>
```

```
            <ul class="sidebar-nav">
                <li class="sidebar-header">
                    Pages
                </li>

                <li class="sidebar-item ">
                    <a class="sidebar-link" href="/dashboard">
          <i class="align-middle" data-feather="sliders"></i> <span class="align-middle">Dashboard</span>
        </a>
                </li>

                <li class="sidebar-item ">
                    <a class="sidebar-link" href="/items">
          <i class="align-middle" data-feather="book"></i> <span class="align-middle">Items</span>
        </a>
        <li class="sidebar-item">
                    <a class="sidebar-link" href="/purchase_order">
          <i class="align-middle" data-feather="book"></i> <span class="align-middle">Purchase items</span>
        </a>
        <li class="sidebar-item ">
                    <a class="sidebar-link" href="/sales">
          <i class="align-middle" data-feather="book"></i> <span class="align-middle">New Sales</span>
        </a>
      </nav>

    <div class="main">
      <nav class="navbar navbar-expand navbar-light navbar-bg">
        <a class="sidebar-toggle js-sidebar-toggle">
          <i class="hamburger align-self-center"></i>
```

```
        </a>

        <div class="navbar-collapse collapse">
          <ul class="navbar-nav navbar-align">
            </li>
            <li class="nav-item dropdown">
              <a
                class="nav-icon dropdown-toggle d-inline-block d-sm-none"
                href="#"
                data-bs-toggle="dropdown">
                <i class="align-middle" data-feather="settings"></i>
              </a>

              <a
                class="nav-link dropdown-toggle d-none d-sm-inline-block"
                href="#"
                data-bs-toggle="dropdown"
              >
                <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="feather feather-user align-middle me-2"><path d="M20 21v-2a4 4 0 0 0-4-4H8a4 4 0 0 0-4 4v2"></path><circle cx="12" cy="7" r="4"></circle></svg>
                <span class="text-dark">{{user_name_nav}}</span>
              </a>
              <div class="dropdown-menu dropdown-menu-end">
                <div class="dropdown-divider"></div>
                <a class="dropdown-item" href="/">Log out</a>
              </div>
            </li>
          </ul>
        </div>
      </nav>
```

```
<main class="content">
  <div class="container-fluid p-0">

    </form>
    <div class="row">
      <div class="col-12">
        <div class="card">
          <div class="card-header">

            <div>
              <form method="POST" action="/item">
                <p style="color: lightseagreen;display: inline;">{{msg}}</p>
                <Table>
                  <tr>
                    <p style="color: red;display: inline;">{{invalid_msg}}</p>
                    <td>
                      <label for="Product Name">Product Name</label>
                    </td>
                    <td>
                      <input type="text" name="product_name" id="in"
placeholder="{{place_product}}">
                    </td>
                    <td>
                      <label for="Supplier Name">Supplier Name</label>
                    </td>
                    <td>
                      <input type="text" name="Supplier_name" id="in"
placeholder="{{place_supplier}}">
                    </td>
                  </tr>
                  <tr>
```

```
                                    <td>
                                        <label for="Purchase Price">Purchase Price</label>
                                    </td>
                                    <td>
                                        <input type="text" name="purchase_price" id="in"
placeholder={{place_price}}>
                                    </td>
                                    <td>
                                        <label for="Selling Price">Selling Price</label>
                                    </td>
                                    <td>
                                        <input type="text" name="selling_price" id="in"
placeholder="{{place_s_price}}">
                                    </td>
                                </tr>
                                <tr>
                                    <td>
                                        <label for="Quantity">Quantity</label>
                                    </td>
                                    <td>
                                        <input type="text" name="quantity" id="in"
placeholder="{{place_quantity}}">
                                    </td>
                                </tr>
                                <tr>
                                    <td>
                                        <label for="Quantity">Low Stock Limit</label>
                                    </td>
                                    <td>
                                        <input type="text" name="low_stock" id="in"
placeholder="{{place_lowstock}}">
                                    </td>
```

```
                                </tr>
                                    <td>
                                        <button type="submit" class="btn btn-primary">Update</button>
                                    </td>
                                    <td></td>

                                </tr>
                               </Table>
                              </form>
                             </div>

                        </div>
                        <div class="card-body">

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

        </div>
    </main>

    <footer class="footer">
        <div class="container-fluid">
            <div class="row text-muted">
                <div class="col-6 text-start">
                    <p class="mb-0">
                        <a class="text-muted" href="https://adminkit.io/"
target="_blank"><strong>AdminKit</strong></a> - <a class="text-muted"
href="https://adminkit.io/" target="_blank"><strong>Bootstrap Admin
Template</strong></a>                        &copy;
                    </p>
```

```
                </div>
                <div class="col-6 text-end">
                  <ul class="list-inline">
                    <li class="list-inline-item">
                      <a class="text-muted" href="https://adminkit.io/" target="_blank">Support</a>
                    </li>
                    <li class="list-inline-item">
                      <a class="text-muted" href="https://adminkit.io/" target="_blank">Help
Center</a>
                    </li>
                    <li class="list-inline-item">
                      <a class="text-muted" href="https://adminkit.io/" target="_blank">Privacy</a>
                    </li>
                    <li class="list-inline-item">
                      <a class="text-muted" href="https://adminkit.io/" target="_blank">Terms</a>
                    </li>
                  </ul>
                </div>
              </div>
            </div>
          </footer>
        </div>
      </div>

      <script src="/Inventory management system/static/js/app.js"></script>
      <script src="/static/js/app.js"></script>
</body>
</html>
```

purchase_order.html:

```
<!DOCTYPE html>
```

```
<html lang="en">


<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <meta name="description" content="Responsive Admin &amp;amp; Dashboard Template based on Bootstrap 5">
  <meta name="author" content="AdminKit">
  <meta name="keywords" content="adminkit, bootstrap, bootstrap 5, admin, dashboard, template, responsive, css, sass, html, theme, front-end, ui kit, web">


  <link rel="shortcut icon" href="img/icons/icon-48x48.png" />


  <title>Items</title>


  <link rel="stylesheet" type="text/css" href="/static/css/app.css">
  <link rel="stylesheet" type="text/css" href="/static/css/add_prod.css">
  <link rel="stylesheet" type="text/css" href="/static/css/items.css">
  <link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&display=swap" rel="stylesheet">


</head>


<style>


    .dropdown {
      position: relative;
      display: inline-block;


    }
```

```
    .dropdown-content {
      display: none;
      position: absolute;
      background-color: #f1f1f1;
      min-width: 200px;
      box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
      z-index: 1;
    }

    .dropdown-content a {
      color: black;
      padding: 12px 16px;
      text-decoration: none;
      display: block;
    }

    .dropdown-content a:hover {background-color: #ddd;}

    .dropdown:hover .dropdown-content {display: block;}

  /* toggle btn */

  .switch {
  position: relative;
  display: inline-block;
  width: 150px;
  height: 34px;
}

.switch input {display:none;}
```

```
.slider {
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #c73c3c;
  -webkit-transition: .4s;
  transition: .4s;
}

.slider:before {
  position: absolute;
  content: "";
  height: 26px;
  width: 26px;
  left: 4px;
  bottom: 4px;
  background-color: white;
  -webkit-transition: .4s;
  transition: .4s;
}

input:checked + .slider {
  background-color: #2ab933bd;
}

input:focus + .slider {
  box-shadow: 0 0 1px #2196F3;
}
```

```
input:checked + .slider:before {
  -webkit-transform: translateX(55px);
  -ms-transform: translateX(55px);
  transform: translateX(117px);
}

/*------ ADDED CSS ---------*/
.on
{
  display: none;
}

.on, .off
{
  color: white;
  position: absolute;
  transform: translate(-50%,-50%);
  top: 50%;
  left: 50%;
  font-size: 13px;
  font-family: Verdana, sans-serif;
}

input:checked+ .slider .on
{display: block;}

input:checked + .slider .off
{display: none;}

/*--------- END --------*/
```

```
/* Rounded sliders */
.slider.round {
  border-radius: 34px;
}

.slider.round:before {
  border-radius: 50%;}



</style>

<body>
  <div class="wrapper">
    <nav id="sidebar" class="sidebar js-sidebar">
      <div class="sidebar-content js-simplebar">
        <a class="sidebar-brand" href="/dashboard">
          <span class="align-middle">Inventory</span>
        </a>

        <ul class="sidebar-nav">
          <li class="sidebar-header">
            Pages
          </li>

          <li class="sidebar-item ">
            <a class="sidebar-link" href="/dashboard">
              <i class="align-middle" data-feather="sliders"></i> <span class="align-
middle">Dashboard</span>
            </a>
          </li>
```

```
        <li class="sidebar-item ">
          <a class="sidebar-link" href="/items">
            <i class="align-middle" data-feather="box"></i> <span class="align-middle">Items</span>
          </a>
          <li class="sidebar-item active">
          <a class="sidebar-link" href="/purchase_order">
            <i class="align-middle" data-feather="shopping-bag"></i> <span class="align-middle">Purchase items</span>
          </a>
          <li class="sidebar-item ">
          <a class="sidebar-link" href="/sales">
            <i class="align-middle" data-feather="grid"></i> <span class="align-middle">New Sales</span>
          </a>
    </nav>

     <div class="main">
       <nav class="navbar navbar-expand navbar-light navbar-bg">
        <a class="sidebar-toggle js-sidebar-toggle">
          <i class="hamburger align-self-center"></i>
        </a>

        <div class="navbar-collapse collapse">
          <ul class="navbar-nav navbar-align">
            </li>
            <li class="nav-item dropdown">
              <a
                class="nav-icon dropdown-toggle d-inline-block d-sm-none"
                href="#"
                data-bs-toggle="dropdown">
                <i class="align-middle" data-feather="settings"></i>
```

```
          </a>


          <a
            class="nav-link dropdown-toggle d-none d-sm-inline-block"
            href="#"
            data-bs-toggle="dropdown"
          >
            <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="feather feather-user align-middle me-2"><path d="M20 21v-2a4 4 0 0 0-4-4H8a4 4 0 0 0-4 4v2"></path><circle cx="12" cy="7" r="4"></circle></svg>
            <span class="text-dark">{{user_name_nav}}</span>
          </a>
          <div class="dropdown-menu dropdown-menu-end">
            <div class="dropdown-divider"></div>
            <a class="dropdown-item" href="/">Log out</a>
          </div>
        </li>
      </ul>
    </div>
  </nav>

  <main class="content">
    <div class="container-fluid p-0">

      <h1 class="h3 mb-3">Items In Inventory</h1>

      <div class="row">
        <div class="col-12">
          <div class="card">
            <div class="card-header">
```

```
        <div>
            <form method="post" action="/purchase_order">
<p style="color: lightseagreen;display: inline;">{{msg}}</p>
<Table>
 <tr>
  <p style="color: red;display: inline;">{{invalid_msg}}</p>
  <td>
   <label for="Product Name">Product Name</label>
  </td>
  <td>
   <input type="text" name="product_name" id="in">
  </td>
  <td>
   <label for="Supplier Name">Supplier Name</label>
  </td>
  <td>
   <input type="text" name="Supplier_name" id="in">
  </td>
 </tr>
 <tr>
  <td>
   <label for="Purchase Price">Purchase Price</label>
  </td>
  <td>
   <input type="number" name="purchase_price" id="in">
  </td>
  <td>
   <label for="Selling Price">Selling Price</label>
  </td>
  <td>
   <input type="number" name="selling_price" id="in">
```

```
        </td>
       </tr>
       <tr>
        <td>
         <label for="Quantity">Quantity</label>
        </td>
        <td>
         <input type="number" name="quantity" id="in">
        </td>
       </tr>
       <tr>
        <td>
         <label for="Quantity">Low Stock Limit</label>
        </td>
        <td>
         <input type="number" name="low_stock" id="in">
        </td>
       </tr>
        <td>
         <button type="submit" class="btn btn-primary">Submit</button>
        </td>
        <td></td>

       </tr>
     </Table>
   </form>
              </div>

</div>
<div class="card-body">
</div>
```

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

      </div>
    </main>
  </div>
 </div>
 <script src="/Inventory management system/static/js/app.js"></script>
 <script src="/static/js/app.js"></script>
</body>
</html>
```

sales_page.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="utf-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
 <meta name="description" content="Responsive Admin &amp;amp; Dashboard Template based on Bootstrap 5">
 <meta name="author" content="AdminKit">
 <meta name="keywords" content="adminkit, bootstrap, bootstrap 5, admin, dashboard, template, responsive, css, sass, html, theme, front-end, ui kit, web">

 <link rel="shortcut icon" href="img/icons/icon-48x48.png" />

 <title>Items</title>
```

```
  <link rel="stylesheet" type="text/css" href="/static/css/app.css">
  <link rel="stylesheet" type="text/css" href="/static/css/add_prod.css">
  <link rel="stylesheet" type="text/css" href="/static/css/items.css">
  <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;600&display=swap"
rel="stylesheet">


</head>


<style>


    .dropdown {
      position: relative;
      display: inline-block;


    }


    .dropdown-content {
      display: none;
      position: absolute;
      background-color: #f1f1f1;
      min-width: 200px;
      box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
      z-index: 1;
    }


    .dropdown-content a {
      color: black;
      padding: 12px 16px;
      text-decoration: none;
      display: block;
    }
```

```
    .dropdown-content a:hover {background-color: #ddd;}

    .dropdown:hover .dropdown-content {display: block;}



    </style>

<body>
  <div class="wrapper">
    <nav id="sidebar" class="sidebar js-sidebar">
      <div class="sidebar-content js-simplebar">
        <a class="sidebar-brand" href="/dashboard">
          <span class="align-middle">Inventory</span>
        </a>

        <ul class="sidebar-nav">
          <li class="sidebar-header">
            Pages
          </li>

          <li class="sidebar-item">
            <a class="sidebar-link" href="/dashboard">
              <i class="align-middle" data-feather="sliders"></i> <span class="align-
middle">Dashboard</span>
            </a>
          </li>

          <li class="sidebar-item ">
            <a class="sidebar-link" href="/items">
              <i class="align-middle" data-feather="box"></i> <span class="align-
middle">Items</span>
```

```
        </a>
        <li class="sidebar-item ">
        <a class="sidebar-link" href="/purchase_order">
          <i class="align-middle" data-feather="shopping-bag"></i> <span class="align-middle">Purchase items</span>
        </a>
    <li class="sidebar-item active">
        <a class="sidebar-link" href="/sales">
          <i class="align-middle" data-feather="grid"></i> <span class="align-middle">New Sales</span>
        </a>
  </nav>

    <div class="main">
      <nav class="navbar navbar-expand navbar-light navbar-bg">
       <a class="sidebar-toggle js-sidebar-toggle">
        <i class="hamburger align-self-center"></i>
       </a>

       <div class="navbar-collapse collapse">
        <ul class="navbar-nav navbar-align">
         </li>
         <li class="nav-item dropdown">
           <a
             class="nav-icon dropdown-toggle d-inline-block d-sm-none"
             href="#"
             data-bs-toggle="dropdown">
             <i class="align-middle" data-feather="settings"></i>
           </a>

           <a
             class="nav-link dropdown-toggle d-none d-sm-inline-block"
```

```
            href="#"
            data-bs-toggle="dropdown"
           >
            <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2" stroke-linecap="round" stroke-linejoin="round" class="feather feather-user align-middle me-2"><path d="M20 21v-2a4 4 0 0 0-4-4H8a4 4 0 0 0-4 4v2"></path><circle cx="12" cy="7" r="4"></circle></svg>
            <span class="text-dark">{{user_name_nav}}</span>
           </a>
           <div class="dropdown-menu dropdown-menu-end">
            <div class="dropdown-divider"></div>
            <a class="dropdown-item" href="/">Log out</a>
           </div>
          </li>
         </ul>
        </div>
       </nav>

      <main class="content">
       <div class="container-fluid p-0">

        <h1 class="h3 mb-3">Sales Items from Inventory</h1>

        <div class="row">
         <div class="col-12">
          <div class="card">
           <div class="card-header">

                        <div>
                         <form method="post" action="/sales">
                          <p style="color: lightseagreen;display: inline;">{{msg}}</p>
                          <p style="color: red;display: inline;">{{error}}</p><br>
```

```
<Table>
  <tr>
    <td>
      <label for="Customer Name" >Customer Name</label>
    </td>
    <td>
      <input type="text" name="customer_name" id="in">
    </td>
    <td>
      <label for="Customer Email">Customer Email</label>
    </td>
    <td>
      <input type="email" name="customer_email" id="in">
    </td>
  </tr>
  <tr>

    <td>
      <label for="Phone Number">Phone number</label>
    </td>
    <td>
      <input type="number" name="phone_number" id="in">
    </td>
    <td>
     <p style="color: red;display: inline;">{{invalid_msg}}</p><br>
      <label for="Product Name">Product Id</label>
    </td>
    <td>
      <input type="number" name="product_id" id="in">
    </td>
  </tr>
```

```
                                <tr>
                                    <td>
                                        <label for="Quantity">Quantity</label>
                                    </td>
                                    <td>
                                        <input type="number" name="quantity" id="in">
                                    </td>
                                </tr>
                                    <td>
                                        <button type="submit" class="btn btn-
primary">Submit</button>
                                    </td>
                                </tr>
                            </Table>
                        </form>
                    </div>
                </div>
                <div class="row">
            <div class="col-12">
              <div class="card">
                <div class="card-header">
                  <h1 class="h3 mb-3"><strong>Sales</strong> History</h1>
                  <table>
                    <thead style="background-color: #222e3c;">
                    <tr class="table100-head">
                    <th class="column1">Id</th>
                    <th class="column2">Coustomer name</th>
                    <th class="column3">Coustomer email</th>
                    <th class="column4">coustomer phone number</th>
                    <th class="column5">Product ID</th>
                    <th class="column6">Date</th>
                    <th class="column7">Product name</th>
```

```
        <th class="column9">Quantity</th>
        <th class="column10">Billing amount</th>


        </thead>
        <tbody>
        {% for row in users %}
        <tr >
        <td class="column1">{{row["SALES_ID"]}}</td>
        <td class="column2">{{row["CUSTOMER_NAME"]}}</td>
        <td class="column3">{{row["CUSTOMER_EMAIL"]}}</td>
        <td class="column4">{{row["PHONE_NUMBER"]}}</td>
        <td class="column5">{{row["PRODUCT_ID"]}}</td>
        <td class="column6">{{row["DATE"]}}</td>
        <td class="column7">{{row["PRODUCT_NAME"]}}</td>
        <td class="column9">{{row["QUANTITY"]}}</td>
        <td class="column10">{{row["BILLING_AMOUNT"]}}</td>
        </tr>
        {% endfor %}
        </tbody>
        </table>
    </div>
    <div class="card-body">
    </div>
   </div>
  </div>
</div>
   </div>
  </div>
</div>
```

```
      </div>
    </main>
  </div>
</div>

<script src="/Inventory management system/static/js/app.js"></script>
<script src="/static/js/app.js"></script>




</body>

</html>
```

Git Hub: https://github.com/IBM-EPBL/IBM-Project-21835-1659792353

Project video link: https://youtu.be/bk7My3B3lyY