IBM – NALAIYA THIRAN PROJECT

INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

A PROJECT REPORT

Submitted by

INDUSTRY MENTOR : VASUDEVA HANUSH

FACULTY MENTOR : DHANASEKARAN GURUSAMY

TEAM ID : PNT2022TMID40335

TEAM LEADER : SHOFICA L
TEAM MEMBER : RAMYA S
TEAM MEMBER : KAMALI V
TEAM MEMBER : NISHA R

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2022

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our **Faculty Mentor** and **Industry Mentor** for their support and guidance in completing our project on Inventory Management System for Retailers.

We would like to extend our gratitude to the **IBM** for **Nalaiya Thiran** project for providing us with all the facility that was required.

It was a great learning experience. We would like to take this opportunity to express our gratitude.

DATE:

19/11/2022

TEAM MEMBERS

KAMALI V

NISHA R

RAMYA S

SHOFICA L

PROJECT REPORT FORMAT

1. INTRODUCTION

1.1 **Project Overview**

Inventory management information system is high performance software, which speed up the business operation of the organization. Every organization, which deals with the raw materials, put its great effort in the efficient utilization of its raw, material according to its need and requirement. The organization has to perform number of tasks and operations in order to run its business in manual system. For example From NaavebUROM Estimation of new raw material required. Preparation of purchase order.

Preparation of inward sale invoice This Software “Inventory Management System” is used for recording the information about the day to day transaction of stock of an organization. It stores purchase information of the products with credit/debit information form the supplier. Similarly, it stores sales information with credit/debit about the customer. If a product is purchased, then the related information is stored in stocks, that is, stocks are up to date. Another part I it prepare sales report after product it sold. In the sales information, the information about who sold the product is also kept, so there is no problem for misunderstandings in future.

1.2 Purpose

The project is remarkable chance to experience the real word working environment and culture where the knowledge learn during the IBM course can be implemented. This project not only marks us familiar with real working environment but also make us more mature in the way we deal with real word problem and try to solve problem in the best way possible by applying the knowledge we have acquired throughout the IBM course.

The main objective of the project is to analyse the existing system under study and give necessary suggestions or solution to improve it. To implement the theoretical knowledge acquired from college in real working environment.

To enable us to understand how theory knowledge differs from practical life thus helping us to understand the complexity and unforeseen nature of problem and opportunity that exist in the country as it name implies, the main objective of this software is to record the information about the stocks of an organization and perform basic operations.

2. LITERATURE SURVEY

2.1 Existing Problem

There is a number of inventory management system available in the market. After doing my research, i have come to know that most of them are limited to few products. Some others are lacking in good ui. Marketing points are not much focused on increasing sales.

Customer management system and inventory management system can't be linked due to different organization which leads to compromising the client satisfaction level. Most of them are not using the cloud computing concept but we are trying to develop such a system that is for everyone rather than for only big companies or for a small organization.

Most of them are expensive to use and their maintenance is generally not cheap. Our system is pay-as-per-use.

2.2 References

1. https://www.camcode.com/asset-tags/what-is-an-inventory-management-system/

2. Jimmy wales,online encyclopedia Wikipedia,http://www.wikipeda.org

3. James Gosling.java(programming language),**https://www.java.com**

4. Names Allaire,Netbeans-fully-featured java IDE, **https://www.netbeans.org**

5. James Goslings,welcome to java world.com:how-tofeature and columns by java experts news;java appletst;sample code;tips, **https://www.javaworld.com**

2.3 Problem Statement Definition

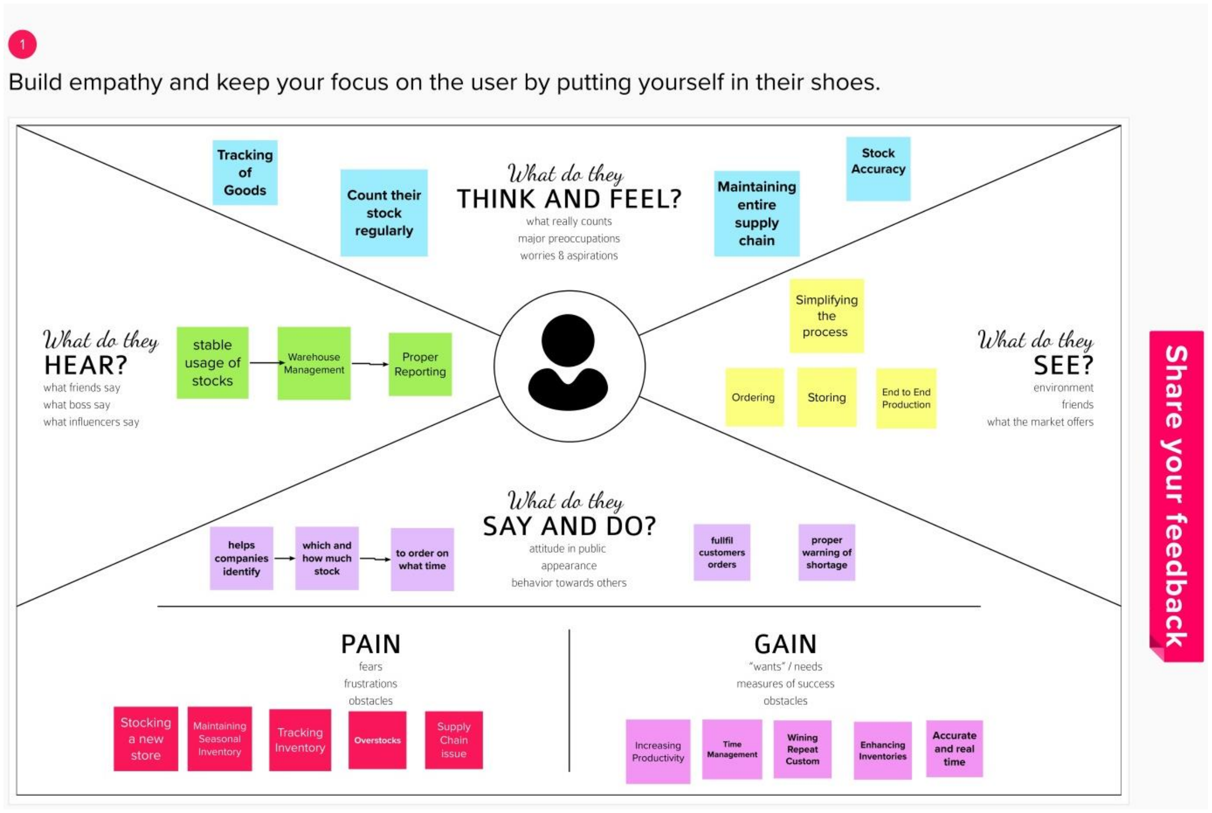
After analyzing many existing IMS we have now the obvious vision of the project to be developed. Before we started to build the application team had many challenges. We defined our problem statement as:

- To make desktop based application of IMS for small organization.
- To make the system easily managed and can be secured.
- To cover all the areas of IMS like purchase details, sales details and stock management.

3. IDEATION & PROPOSED SOLUTION

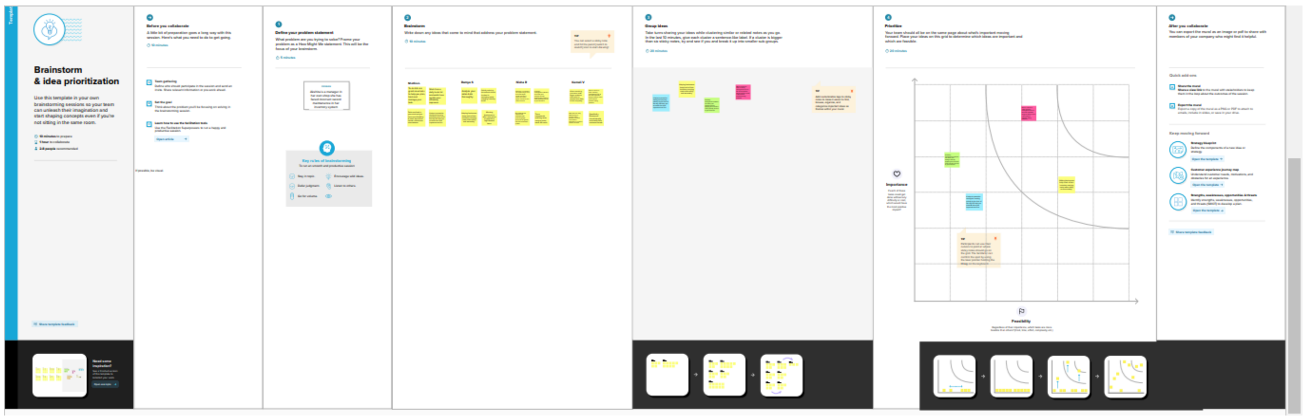
3.1. Empathy Map Canvas

An empathy map canvas helps brands provide a better experience for users by helping teams understand the perspectives and mindset of their customers. Using a template to create an empathy map canvas reduces the preparation time and standardizes the process so you create empathy map canvases of similar quality.



3.2. Ideation & Brainstorming

Ideation is often closely related to the practice of brainstorming, a specific technique that is utilized to generate new ideas. A principal difference between ideation and brainstorming is that ideation is commonly more thought of as being an individual pursuit, while brainstorming is almost always a group activity.



3.3 Proposed Solution

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Nowadays, in an era that has advanced technology and a place in the world. Everything can be linked only at your fingertips in the times of rapidly developing with the sophisticated technology of today. Therefore, an inventory system is also not lagging behind in introducing a method of keeping an inventory data systematically and safely. The system plays a very important role in improving the competitiveness of a business. Usually, organizations today face too many challenges to achieve the cost, speed and reliability. Efficient inventory system really helps in order to make sure the store’s performance and data record is always in good condition and secured from

		abusers. The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.
2.	Idea / Solution description	Manual processing is error-prone, especially in complicated processes like inventory management. Retail inventory management software can ease the pain of the process. It also improves overall accuracy and business productivity. Specific inventory management software for retail can streamline your core activities. As a result, this software would promote customer satisfaction and business growth. To-Increase Anywhere for Retail can help you shorten the process cycles of tedious inventory processes. As our software can efficiently handle critical aspects of your inventory, it can be an asset to your inventory management.

3.	Novelty / Uniqueness	**Real time inventory tracking system** • Sales order are reflected in your stock positions. • Warehouse effects. **Purchase management and supplier management** • Generates and auto fill your orders • Centrally stores all your supplier details • View your transaction history with each supplier • Dynamically generates any quantity discounts your suppliers give you to make optimal purchasing easy. **Real time Inventory values** • Accurate inventory values that account for variations in the price or volume of your purchases. • A live view of inventory value by warehouse, region and country. • Multi-currency support if you buy and / or sell in multiple countries. • The ability to accurately track variable inventory costs like courier fees or production wastage. • The ability to group products to give a

		more granular view of Cost of Goods Sold (Cogs). • A live view of the profit margin on your products according to the sales channel and location and that accounts for any variable costs.
4.	Social Impact / Customer Satisfaction	The results indicate that higher levels of inventory management practice can lead to an enhanced competitive advantage and improved organizational performance. Also, competitive advantage can have a direct, positive impact on organizational performance. Inventory management helps you maintain customer satisfaction when it comes to product returns. When product is returned because it is damaged or dead on arrival, and it is still under warranty, you can arrange with the manufacturer to do an instant swap of the product to keep the customer happy.
5.	Business Model (Revenue Model)	By providing service to the small and large scale retailers.
6.	Scalability of the Solution	To increase the scalability of your business, you should use an automated inventory management system for inventory tracking. This will make your business much more scalable so that you can continue building consistent growth and

		take advantage of increased sales. An automated inventory management system will give your business the structure and real-time metrics it needs to remain competitive and achieve growth goals.

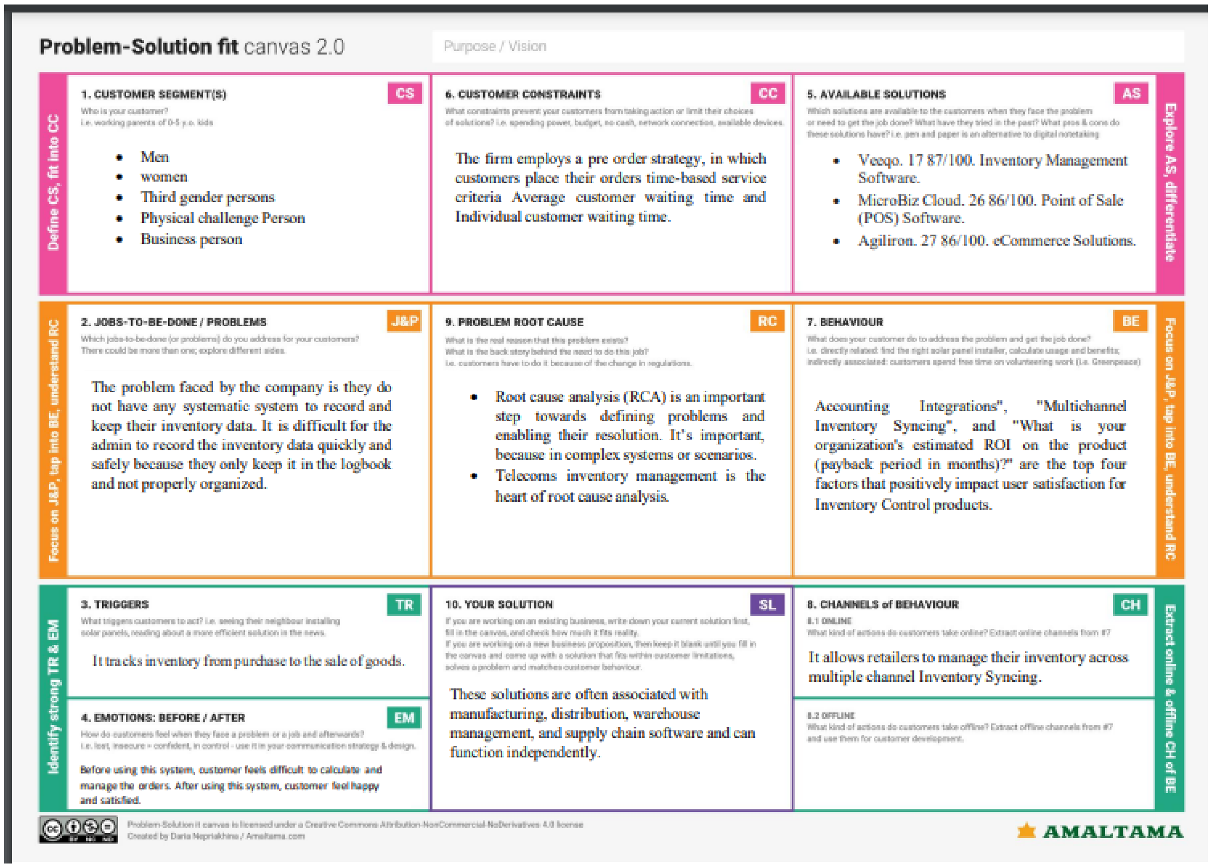
3.4 Problem Solution fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer’s problem. It helps entrepreneurs, marketers and corporate innovators identify behavioural patterns and recognize what would work and why

Purpose:

- Solve complex problems in a way that fits the state of your customers.
- Succeed faster and increase your solution adoption by tapping into existing mediums and channels of behaviour.
- Sharpen your communication and marketing strategy with the right triggers and messaging.
- Increase touch-points with your company by finding the right problem-behaviour fit and building trust by solving frequent annoyances, or urgent or costly problems.
- Understand the existing situation in order to improve it for your target group.

TEMPLATE



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

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

- Men
- women
- Third gender persons
- Physical challenge Person
- Business person

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

The firm employs a pre order strategy, in which customers place their orders time-based service criteria Average customer waiting time and Individual customer waiting time.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

- Veeqo. 17 87/100. Inventory Management Software.
- MicroBiz Cloud. 26 86/100. Point of Sale (POS) Software.
- Agiliron. 27 86/100. eCommerce Solutions.

Explore AS, differentiate

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

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

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

The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

- Root cause analysis (RCA) is an important step towards defining problems and enabling their resolution. It's important, because in complex systems or scenarios.
- Telecoms inventory management is the heart of root cause analysis.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

Accounting Integrations", "Multichannel Inventory Syncing", and "What is your organization's estimated ROI on the product (payback period in months)?" are the top four factors that positively impact user satisfaction for Inventory Control products.

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

Identify strong TR & EM

3. TRIGGERS TR

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

It tracks inventory from purchase to the sale of goods.

4. EMOTIONS: BEFORE / AFTER EM

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

Before using this system, customer feels difficult to calculate and manage the orders. After using this system, customer feel happy and satisfied.

10. YOUR SOLUTION SL

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

These solutions are often associated with manufacturing, distribution, warehouse management, and supply chain software and can function independently.

8. CHANNELS of BEHAVIOUR CH

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

It allows retailers to manage their inventory across multiple channel Inventory Syncing.

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

Extract online & offline CH of BE

AMALTAMA

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)

FR-1	User Registration	Registration through Form Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login with username Login with password
FR-4	Product record	Product name Stock count Product category Vendor details
FR-5	Email Notification	Email through SendGrid Reduced stock quantity Email to both retailer and seller
FR-6	Audit Monitoring	Monitor incoming and outgoing stock
FR-7	Database	Usage of Standard database for storing of data.

4.2 Non Functional requirements

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

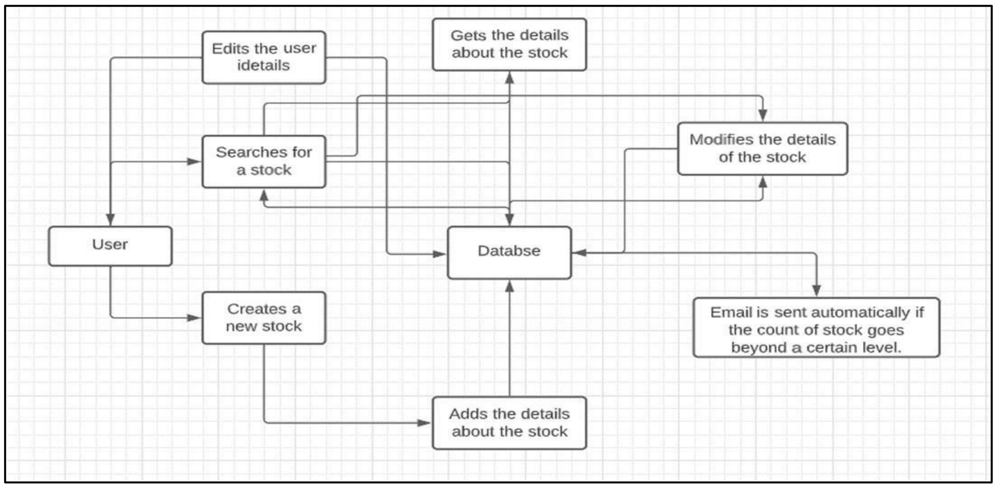
NFR No.	Non-Functional Requirement	Description

NFR-1	**Usability**	• Highly portable • User-friendly • Highly responsive • Easy to use • Not complex
NFR-2	**Security**	• Access Control • Password management features • User privileges • Provides authentication
NFR-3	**Reliability**	• Secure server for reliable and fault tolerant connection. • It will be reliable that it can update with very time period so that the accuracy will be good.
NFR-4	**Performance**	• Reliable performance with high-end servers. • User can track the record of goods available using the application.
NFR-5	**Availability**	• Service hosting server downtime should be negligible during upgradation. • User can track the record of goods available using the application.
NFR-6	**Scalability**	• The resources and service provided by the software should be scalable. • It is scalable that we are going to use data in kilobytes so that the quite amount of storage is satisfied.

5. PROJECT DESIGN

5.1 Data Flow Diagrams:

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



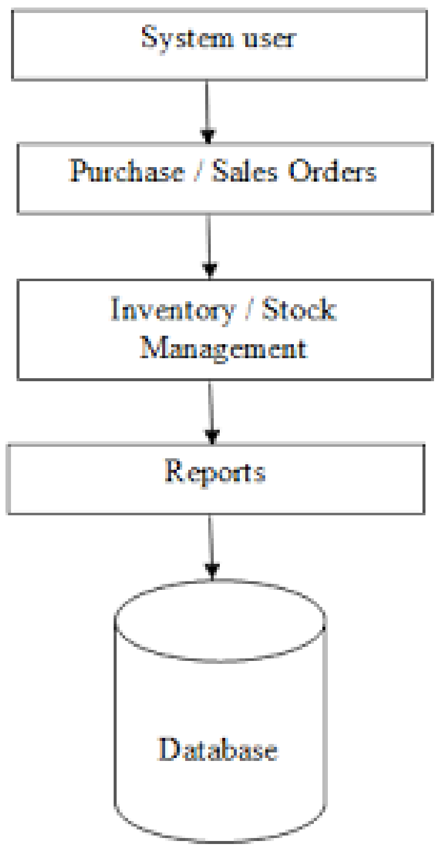
Edits the user idetails
Gets the details about the stock
Searches for a stock
Modifies the details of the stock
User
Databse
Creates a new stock
Email is sent automatically if the count of stock goes beyond a certain level.
Adds the details about the stock

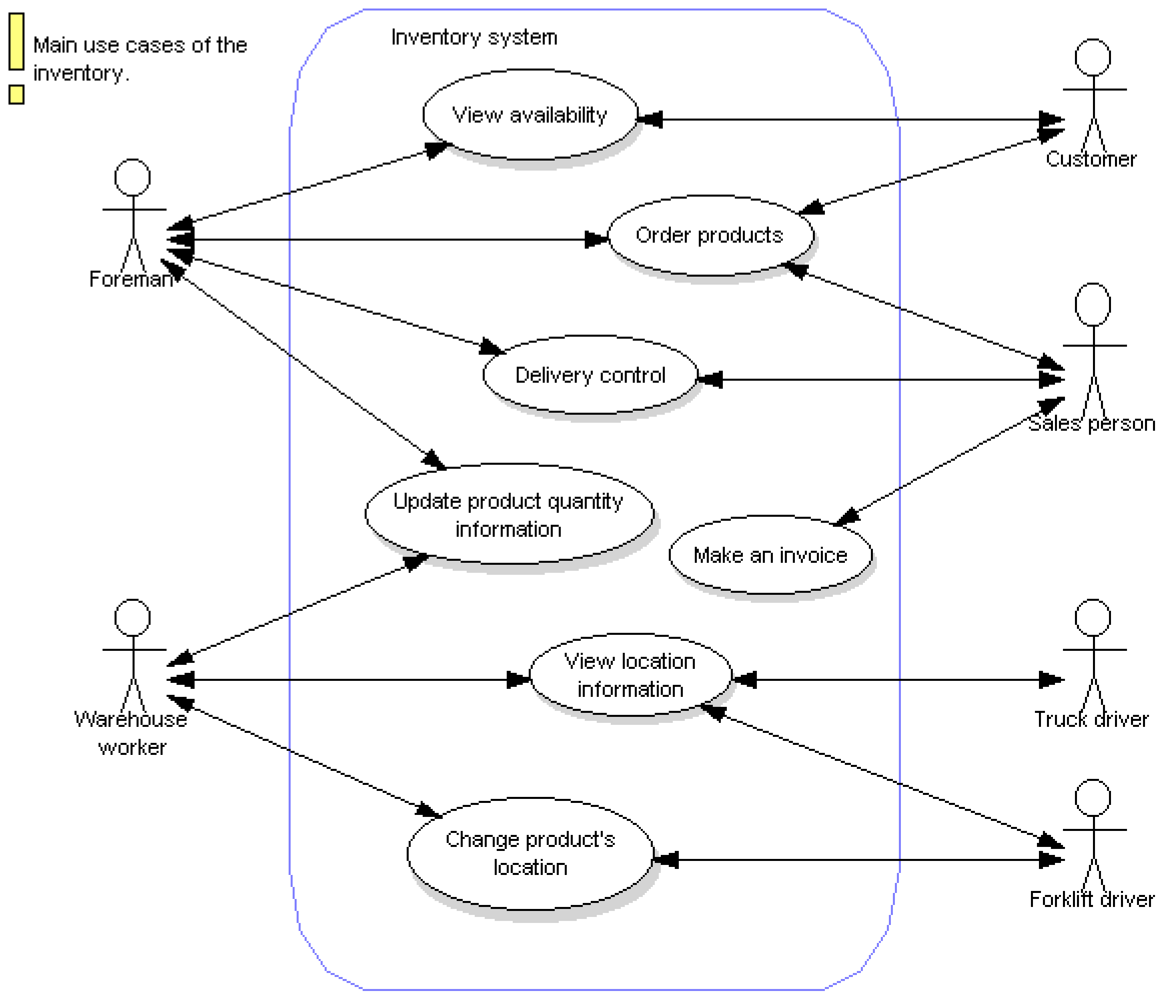
5.2 Solution &Technical Architecture

Solution Architecture:

- There was no an efficient solution available in the many companies during these days. Every process was based on paperwork, human fault rate was high, the process and the tracing the inventory losses were not possible, and there were no efficient logging systems.
- After the computer age, every process is started to be integrated into electronic environment. And now we have qualified technology to implement new solutions to these problems.
- Software based systems bring the advantages of having the most efficient control with less effort and employees. These developments provide new solutions for also inventory management systems.

Example - Solution Architecture Diagram:


System user
Purchase / Sales Orders
Inventory / Stock Management
Reports
Database



Technical Architecture:

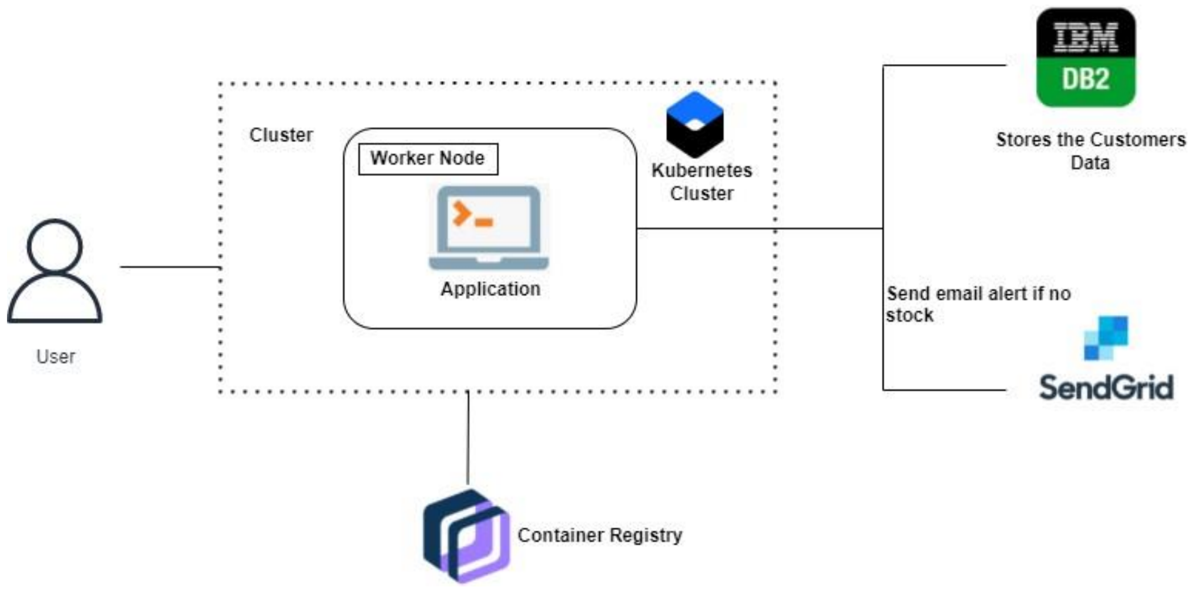


Table-1 : Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Bootstrap etc.
2.	Application Logic-1	Logic for a process in the application	Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	SqlAlchemy, Sqlite etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local File system
8.	External API-1	Purpose of External API used in the application	JOB API, etc.
9.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Flask, Bootstrap, Kubernetes
2.	Security Implementations	List all the security / access controls implemented, use offirewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Kubernetes, docker
4.	Availability	Justify the availability of application (e.g. use offload balancers, distributed servers etc.)	Distributed Servers
5.	Performance	Design consideration for theperformance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Use of CDN

5.3 User Stories

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority
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
Sprint-1	Registration	USN-2	As a user, I can register for the application through E-mail	1	High
Sprint-2	Registration	USN-3	As a user, I will receive confirmation email once I have registered for the application	2	Low
Sprint-2	Registration	USN-4	As a user, I can log into the application by entering email & password	2	Low

Sprint-1	Registration	USN-5	As a user, I can view the products which are available	2	Medium
Sprint-1	Login	USN-6	As a user, I can add the products I wish to buy to the carts	1	High
Sprint-3	Dashboard	USN-7	As a user, I can add products which are not available in the dashboard to the stock list	1	Low
Sprint-3	Dashboard	USN-8	As a user, I can contact the Customer Care Executive and request any services I want from the customer care	1	High
Sprint-3	Dashboard	USN-9	I can be able to report any difficulties I experience as a report	2	High
Sprint-4	Dashboard	USN-10	As a user, I can able to see the Nearby cheap, and Quality products	1	Low
Sprint-4	Management	USN-11	As a Administrator, I will update our web application with additional features.	2	Low
Sprint-4	Management	USN-12	As a Administrator, I can maintain third party Services	1	High

6. PROJECT PLANNING & SCHEDULING

6.1. Sprint Planning & Estimation

Project Tracker, Velocity& Burn down Chart:

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	6	6 Days	24 Oct 2022	29 Oct 2022	6	29 Oct 2022
Sprint-2	6	6 Days	31 Oct 2022	05 Nov 2022	6	05 Nov 2022
Sprint-3	6	6 Days	07 Nov 2022	12 Nov 2022	5	12 Nov 2022
Sprint-4	6	6 Days	14 Nov 2022	19 Nov 2022	6	19 Nov 2022

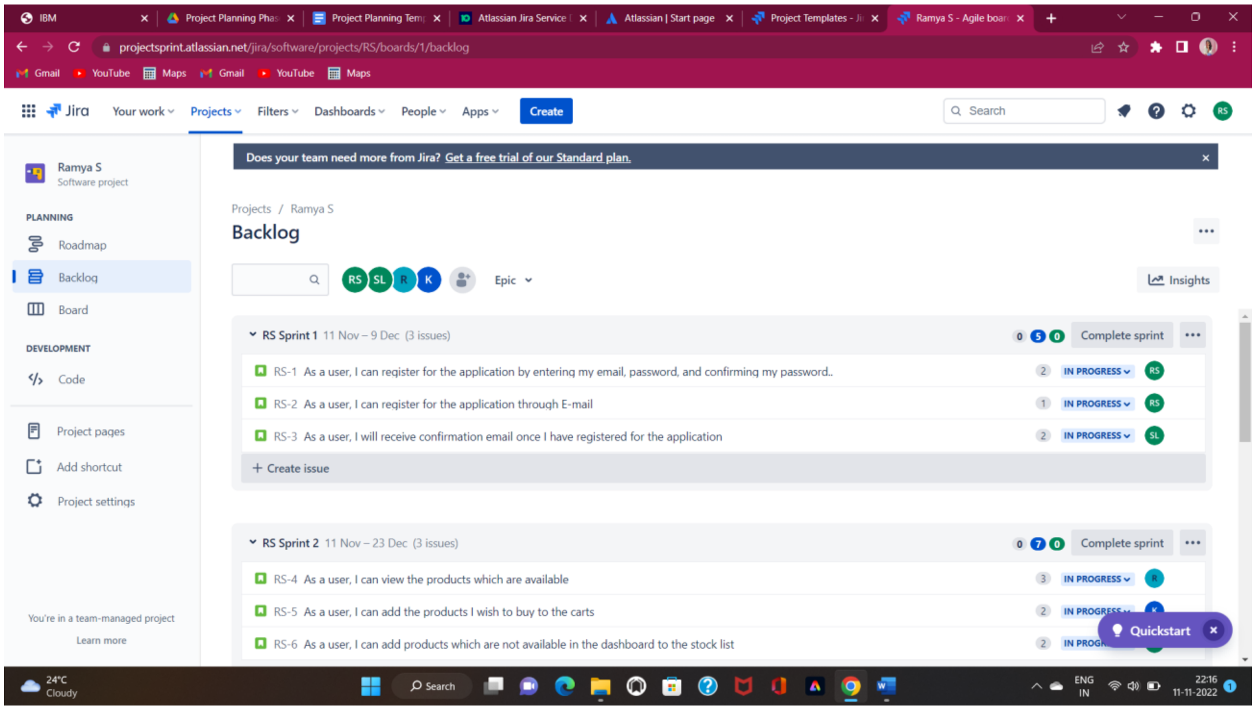
6.2. Sprint Delivery Schedule

Use the below template to create product backlog and sprint schedule:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password..	2	High	Ramya S Shofica L
Sprint-1	Registration	USN-2	As a user, I can register for the application through E-mail	1	High	Kamali V
Sprint-2	Registration	USN-3	As a user, I will receive confirmation email once I have registered for the application	2	Low	Nisha R Kamali V
Sprint-2	Registration	USN-4	As a user, I can log into the application by entering email & password	2	Low	Shofica L Ramya S
Sprint-1	Registration	USN-5	As a user, I can view the products which are available	2	Medium	Nisha R Kamali V
Sprint-1	Login	USN-6	As a user, I can add the products I wish to buy to the carts	1	High	Ramya S
Sprint-3	Dashboard	USN-7	As a user, I can add products which are not available in the dashboard to the stock list	1	Low	Nisha R
Sprint-3	Dashboard	USN-8	As a user, I can contact the Customer Care Executive and request any services I want from the customer care	1	High	Shofica L
Sprint-3	Dashboard	USN-9	I can be able to report any difficulties I experience	2	High	Ramya S Shofica L

			as a report			
Sprint-4	Dashboard	USN-10	As a user, I can able to see the Nearby cheap, and Quality products	1	Low	Nisha R
Sprint-4	Management	USN-11	As a Administrator, I will update our web application with additional features.	2	Low	Shofica L Nisha R
Sprint-4	Management	USN-12	As a Administrator, I can maintain third party Services	1	High	Ramya S

6.3. Report from JIRA



projectsprint.atlassian.net/jira/software/projects/RS/boards/1/backlog

Does your team need more from Jira? Get a free trial of our Standard plan.

Ramya S
Software project

Projects / Ramya S

Backlog

RS Sprint 1 11 Nov – 9 Dec (3 issues) 0 5 0 Complete sprint

RS-1	As a user, I can register for the application by entering my email, password, and confirming my password..	2	IN PROGRESS	RS
RS-2	As a user, I can register for the application through E-mail	1	IN PROGRESS	RS
RS-3	As a user, I will receive confirmation email once I have registered for the application	2	IN PROGRESS	SL

RS Sprint 2 11 Nov – 23 Dec (3 issues) 0 7 0 Complete sprint

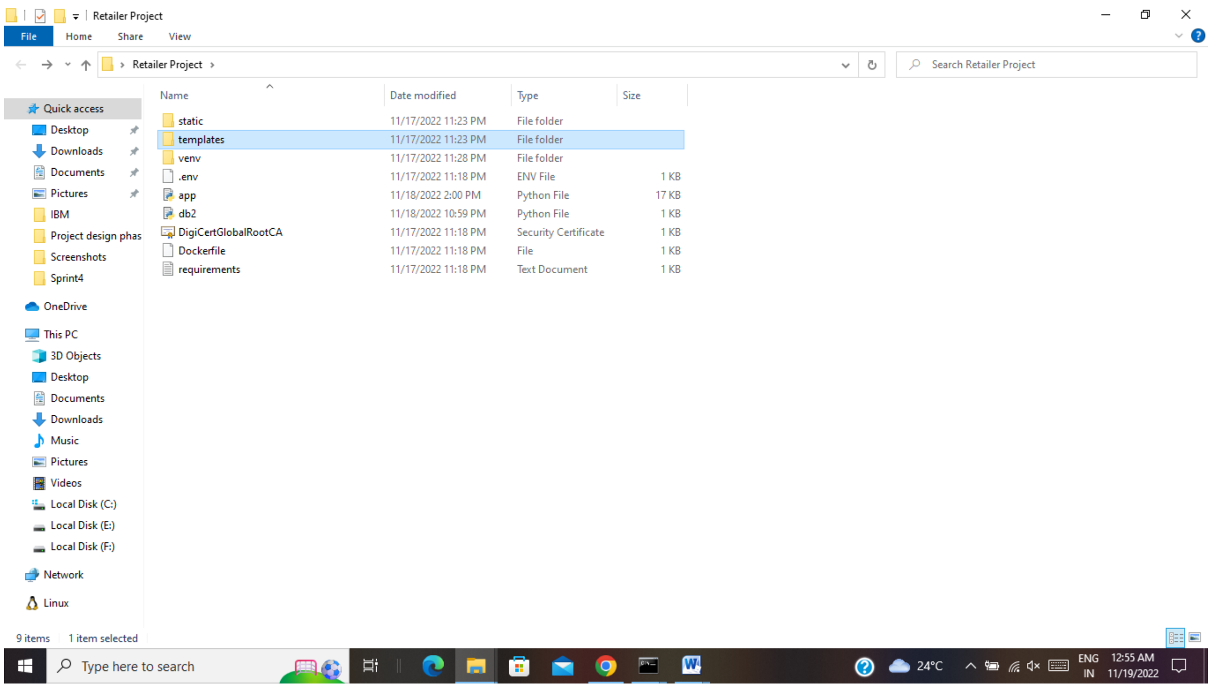
RS-4	As a user, I can view the products which are available	3	IN PROGRESS	R
RS-5	As a user, I can add the products I wish to buy to the carts	2	IN PROGRESS	K
RS-6	As a user, I can add products which are not available in the dashboard to the stock list	2	IN PROG	

You're in a team-managed project
Learn more

7. CODING & SOLUTIONING

7.1 Feature 1

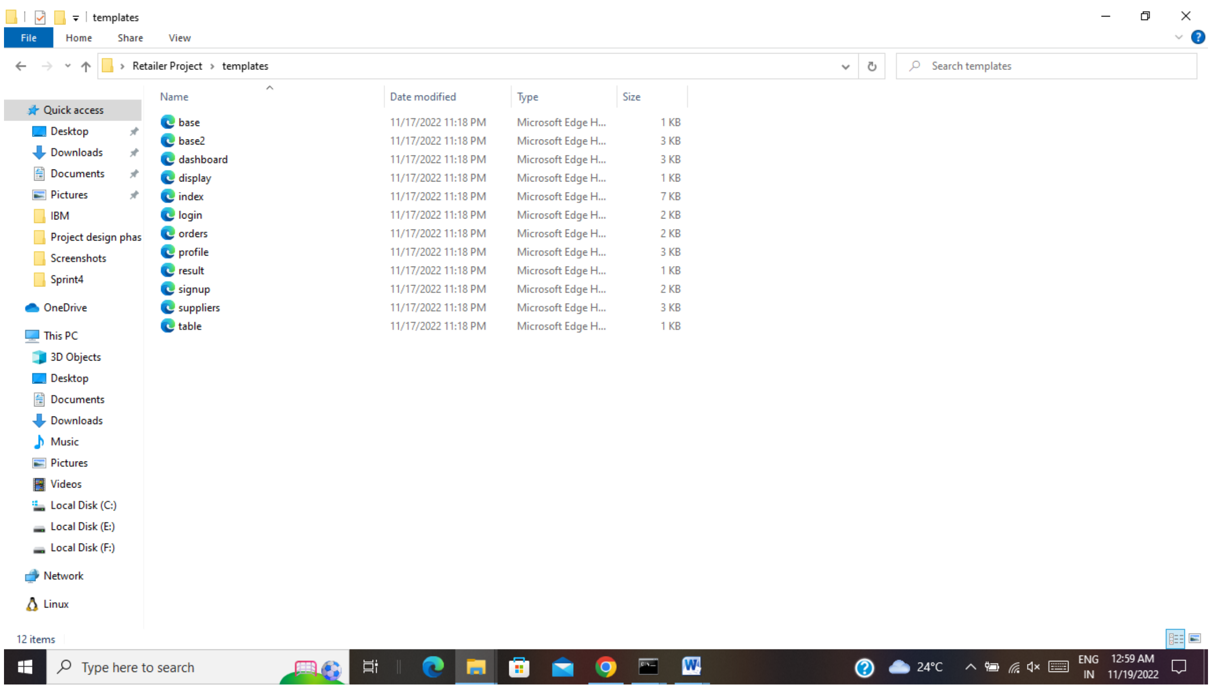
This project consists of HTML, PYTHON, CSS, FLASK, DATABASE, JAVASCRIPT, DOCKER, KUBERNATES, WATSAN ASSISTANT.



Name	Date modified	Type	Size
static	11/17/2022 11:23 PM	File folder	
templates	11/17/2022 11:23 PM	File folder	
venv	11/17/2022 11:28 PM	File folder	
.env	11/17/2022 11:18 PM	ENV File	1 KB
app	11/18/2022 2:00 PM	Python File	17 KB
db2	11/18/2022 10:59 PM	Python File	1 KB
DigiCertGlobalRootCA	11/17/2022 11:18 PM	Security Certificate	1 KB
Dockerfile	11/17/2022 11:18 PM	File	1 KB
requirements	11/17/2022 11:18 PM	Text Document	1 KB

HTML Pages are designed and linked in python flask code and runs as a complete program.

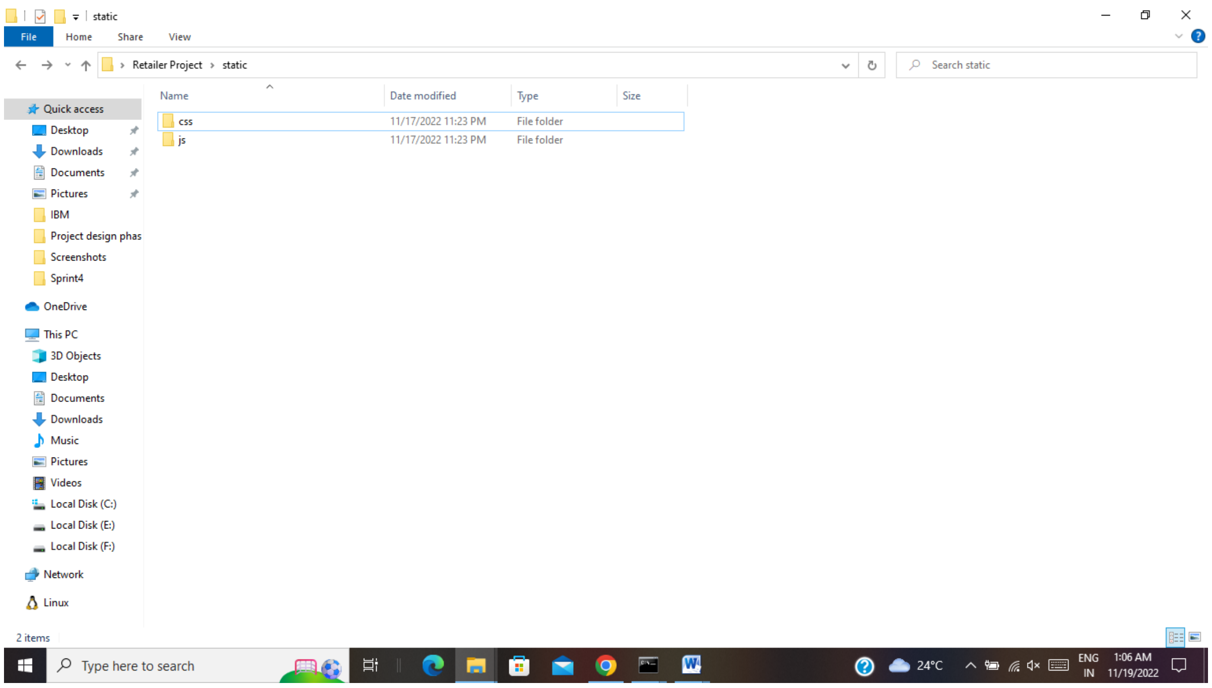
- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.


Retailer Project > templates

Name	Date modified	Type	Size
base	11/17/2022 11:18 PM	Microsoft Edge H...	1 KB
base2	11/17/2022 11:18 PM	Microsoft Edge H...	3 KB
dashboard	11/17/2022 11:18 PM	Microsoft Edge H...	3 KB
display	11/17/2022 11:18 PM	Microsoft Edge H...	1 KB
index	11/17/2022 11:18 PM	Microsoft Edge H...	7 KB
login	11/17/2022 11:18 PM	Microsoft Edge H...	2 KB
orders	11/17/2022 11:18 PM	Microsoft Edge H...	2 KB
profile	11/17/2022 11:18 PM	Microsoft Edge H...	3 KB
result	11/17/2022 11:18 PM	Microsoft Edge H...	1 KB
signup	11/17/2022 11:18 PM	Microsoft Edge H...	2 KB
suppliers	11/17/2022 11:18 PM	Microsoft Edge H...	3 KB
table	11/17/2022 11:18 PM	Microsoft Edge H...	1 KB

12 items

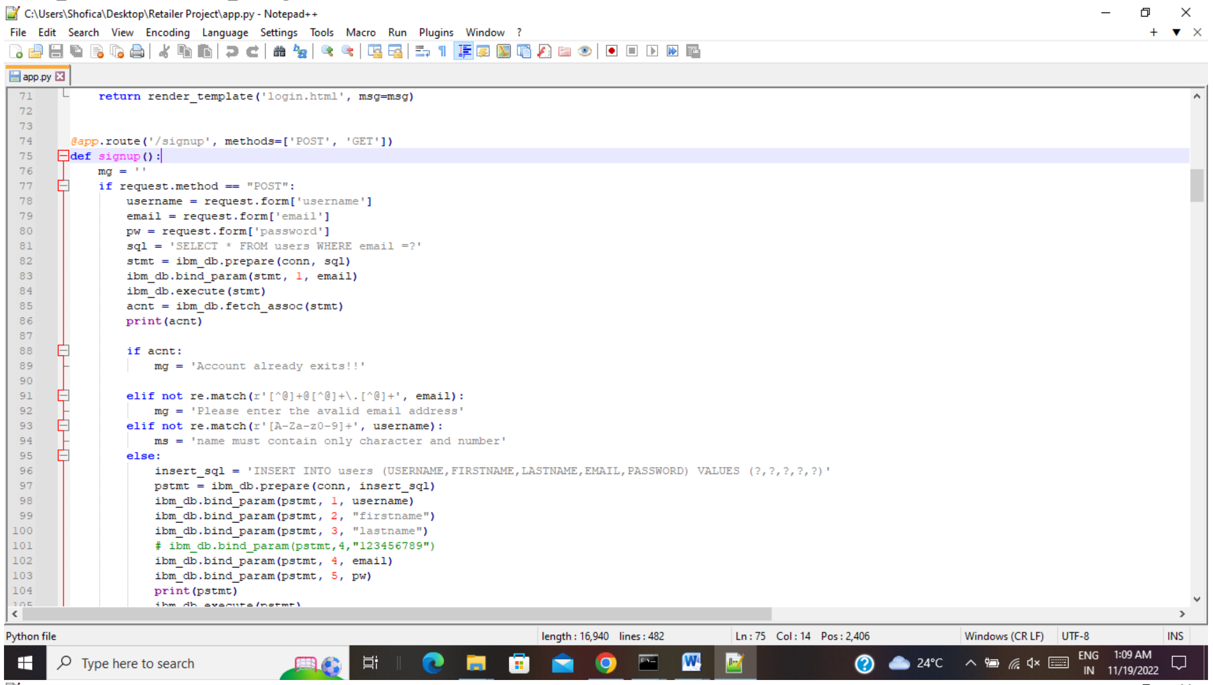
Following are the css and js page designed in this project.


Retailer Project > static

Name	Date modified	Type	Size
css	11/17/2022 11:23 PM	File folder	
js	11/17/2022 11:23 PM	File folder	

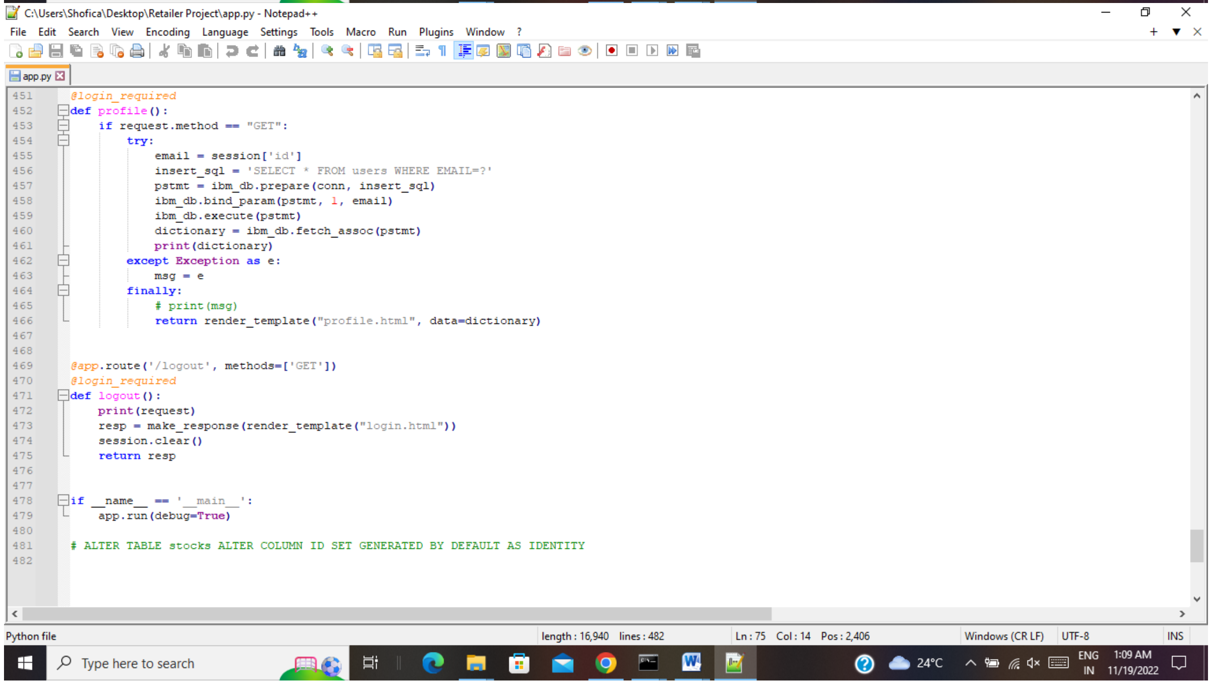
2 items

The Following python program is the main program runs and executes the entire project in which the html code and database connections are made. Thus the successful execution of the program app.py in terminal will get output of this project.


C:\Users\Shofica\Desktop\Retailer Project\app.py - Notepad++

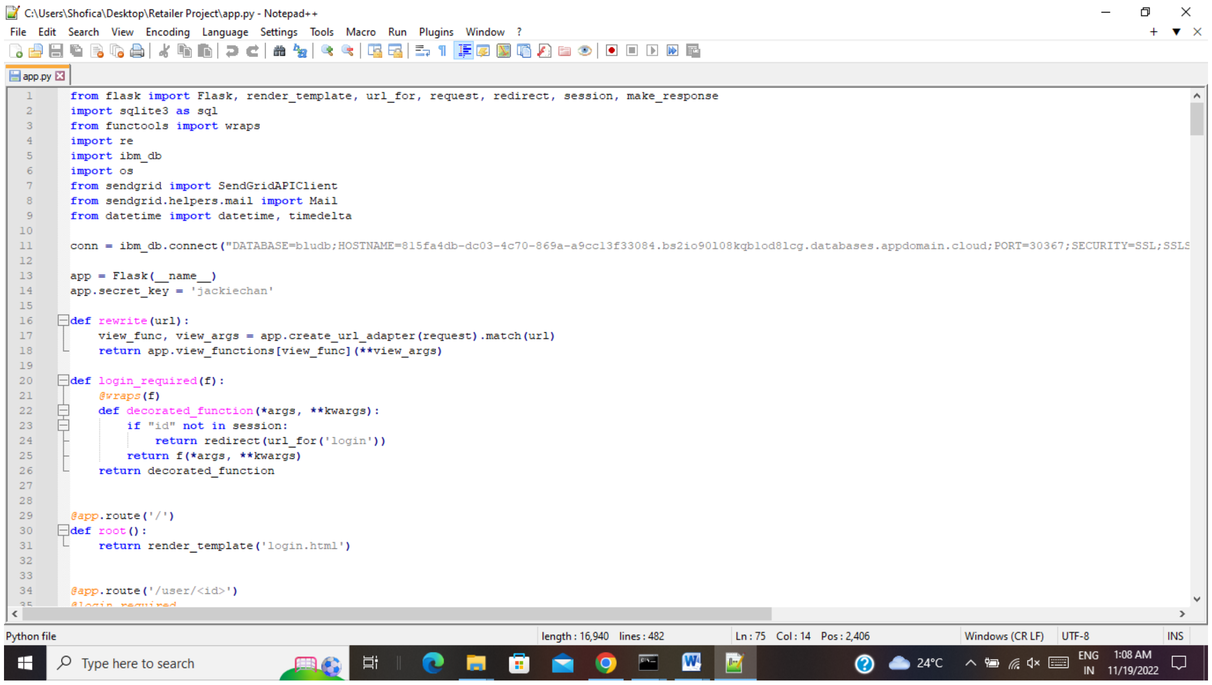
```python
        return render_template('login.html', msg=msg)


@app.route('/signup', methods=['POST', 'GET'])
def signup():
    mg = ''
    if request.method == "POST":
        username = request.form['username']
        email = request.form['email']
        pw = request.form['password']
        sql = 'SELECT * FROM users WHERE email =?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        acnt = ibm_db.fetch_assoc(stmt)
        print(acnt)

        if acnt:
            mg = 'Account already exits!!'

        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            mg = 'Please enter the avalid email address'
        elif not re.match(r'[A-Za-z0-9]+', username):
            ms = 'name must contain only character and number'
        else:
            insert_sql = 'INSERT INTO users (USERNAME,FIRSTNAME,LASTNAME,EMAIL,PASSWORD) VALUES (?,?,?,?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, username)
            ibm_db.bind_param(pstmt, 2, "firstname")
            ibm_db.bind_param(pstmt, 3, "lastname")
            # ibm_db.bind_param(pstmt,4,"123456789")
            ibm_db.bind_param(pstmt, 4, email)
            ibm_db.bind_param(pstmt, 5, pw)
            print(pstmt)
            ibm_db.execute(pstmt)
```

Python file length : 16,940 lines : 482 Ln : 75 Col : 14 Pos : 2,406 Windows (CR LF) UTF-8 INS


C:\Users\Shofica\Desktop\Retailer Project\app.py - Notepad++

```python
@login_required
def profile():
    if request.method == "GET":
        try:
            email = session['id']
            insert_sql = 'SELECT * FROM users WHERE EMAIL=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, email)
            ibm_db.execute(pstmt)
            dictionary = ibm_db.fetch_assoc(pstmt)
            print(dictionary)
        except Exception as e:
            msg = e
        finally:
            # print(msg)
            return render_template("profile.html", data=dictionary)


@app.route('/logout', methods=['GET'])
@login_required
def logout():
    print(request)
    resp = make_response(render_template("login.html"))
    session.clear()
    return resp


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

# ALTER TABLE stocks ALTER COLUMN ID SET GENERATED BY DEFAULT AS IDENTITY

```

Python file length : 16,940 lines : 482 Ln : 75 Col : 14 Pos : 2,406 Windows (CR LF) UTF-8 INS

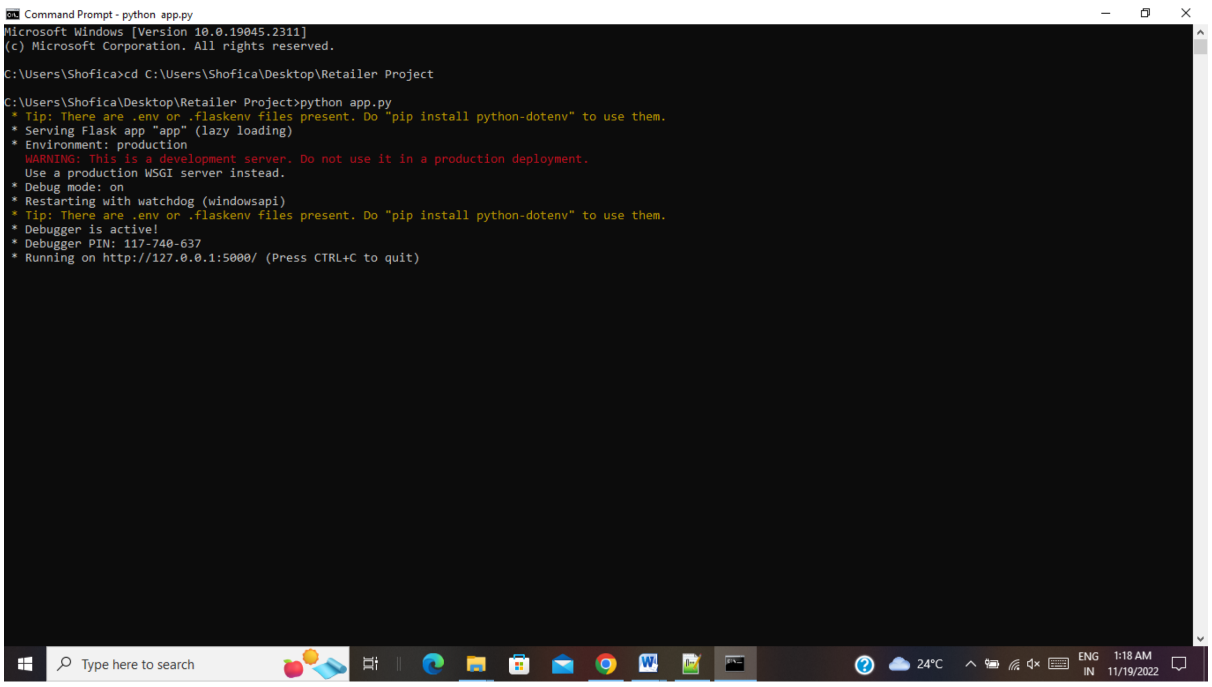

C:\Users\Shofica\Desktop\Retailer Project\app.py - Notepad++

```python
from flask import Flask, render_template, url_for, request, redirect, session, make_response
import sqlite3 as sql
from functools import wraps
import re
import ibm_db
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
from datetime import datetime, timedelta

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=815fa4db-dc03-4c70-869a-a9cc13f33084.bs2io90108kqb1od8lcg.databases.appdomain.cloud;PORT=30367;SECURITY=SSL;SSLS

app = Flask(__name__)
app.secret_key = 'jackiechan'

def rewrite(url):
    view_func, view_args = app.create_url_adapter(request).match(url)
    return app.view_functions[view_func](**view_args)

def login_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        if "id" not in session:
            return redirect(url_for('login'))
        return f(*args, **kwargs)
    return decorated_function


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


@app.route('/user/<id>')
```

Python file length : 16,940 lines : 482 Ln : 75 Col : 14 Pos : 2,406 Windows (CR LF) UTF-8 INS

7.2 Feature 2

Set the path to the command prompt and run the program.

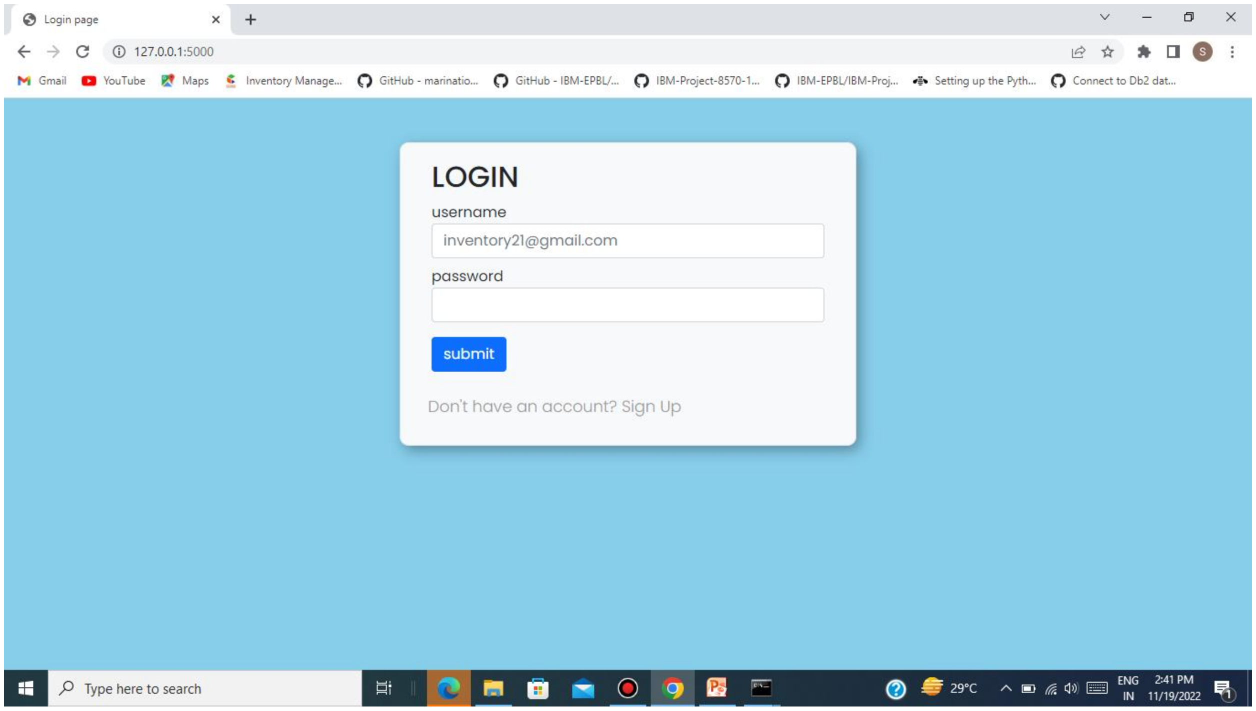


```
Microsoft Windows [Version 10.0.19045.2311]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shofica>cd C:\Users\Shofica\Desktop\Retailer Project

C:\Users\Shofica\Desktop\Retailer Project>python app.py
 * Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with watchdog (windowsapi)
 * Tip: There are .env or .flaskenv files present. Do "pip install python-dotenv" to use them.
 * Debugger is active!
 * Debugger PIN: 117-740-637
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Copy the last http://127.0.0.1:5000/ and paste it on browser the following executions will be displayed.

LOGIN PAGE

HERE USER CAN LOGIN IF ALREADY REGISTERED



LOGIN

username: inventory21@gmail.com

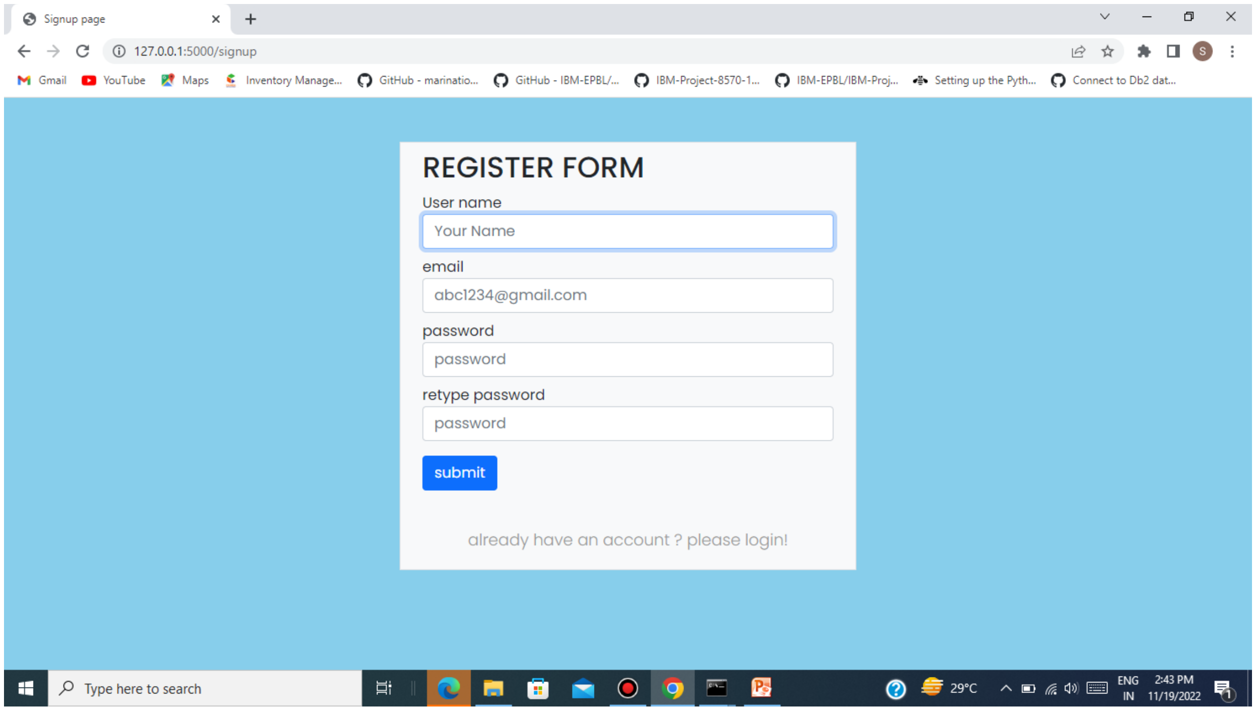
password:

submit

Don't have an account? Sign Up

REGISTION PAGE

This page is for the new user



REGISTER FORM

User name

Your Name

email

abc1234@gmail.com

password

password

retype password

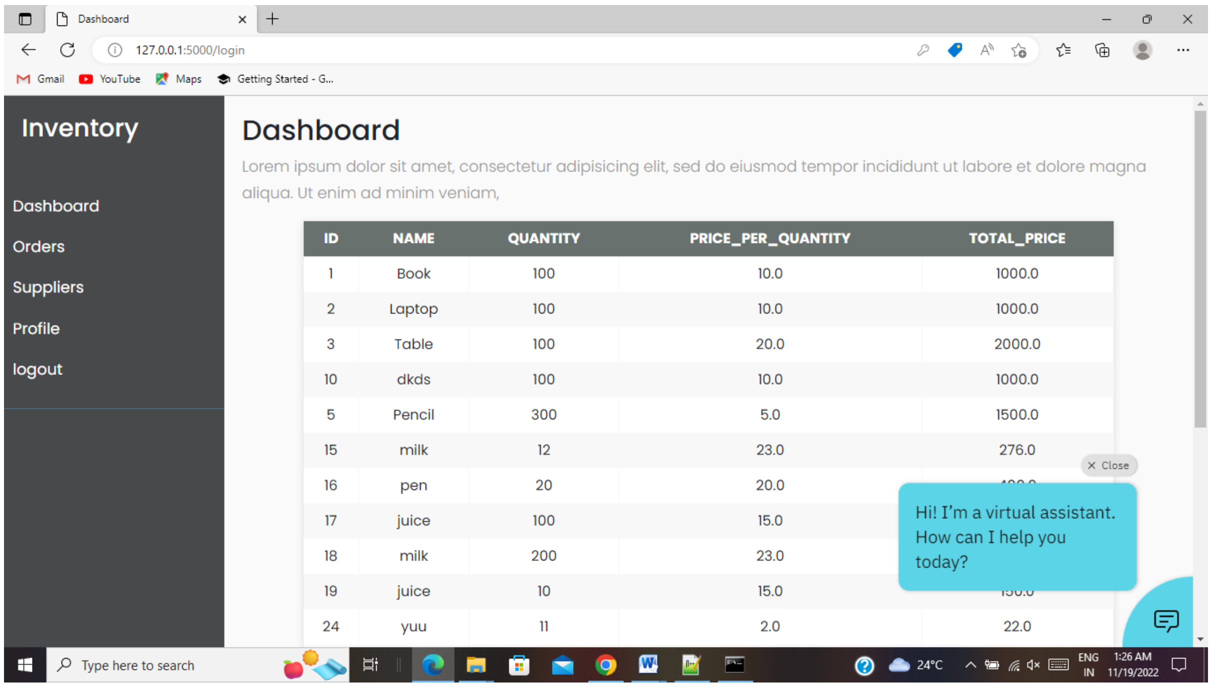
password

submit

already have an account ? please login!

DASHBOARD

Dashboard contains Watson assistant, orders, suppliers, profile and logout.



Inventory

Dashboard
Orders
Suppliers
Profile
logout

Dashboard

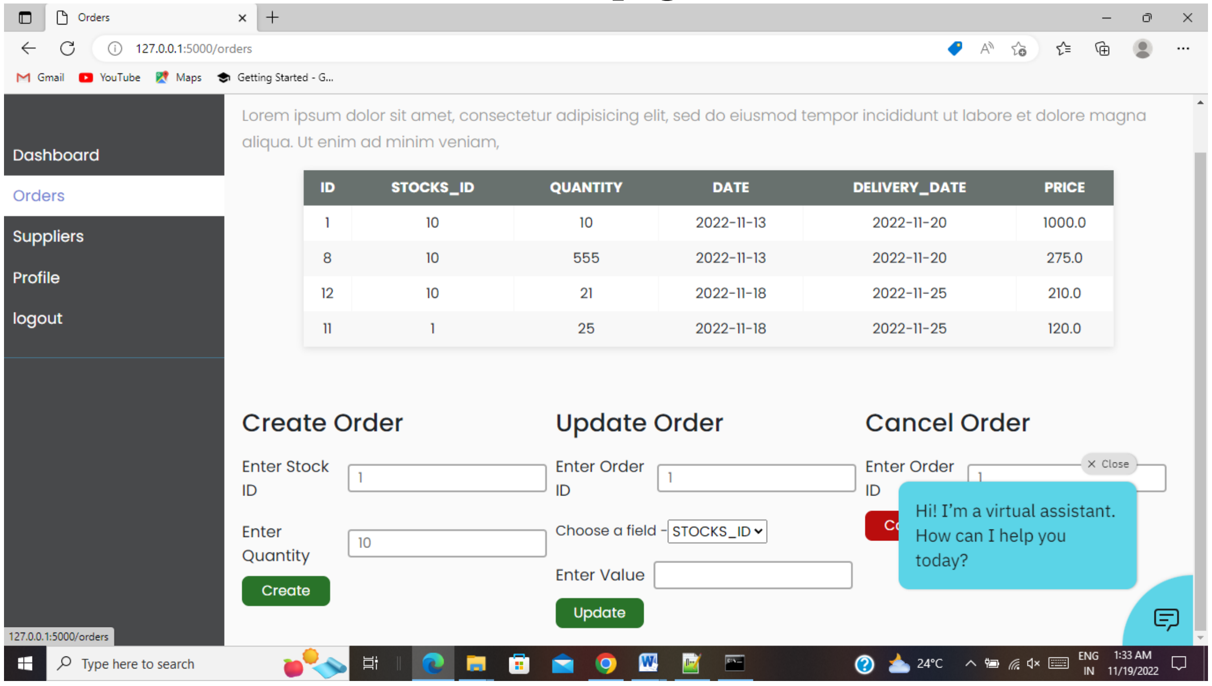
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

ID	NAME	QUANTITY	PRICE_PER_QUANTITY	TOTAL_PRICE
1	Book	100	10.0	1000.0
2	Laptop	100	10.0	1000.0
3	Table	100	20.0	2000.0
10	dkds	100	10.0	1000.0
5	Pencil	300	5.0	1500.0
15	milk	12	23.0	276.0
16	pen	20	20.0	[illegible]
17	juice	100	15.0	[illegible]
18	milk	200	23.0	[illegible]
19	juice	10	15.0	150.0
24	yuu	11	2.0	22.0

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

ORDERS

The user can make orders in this page.



Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

ID	STOCKS_ID	QUANTITY	DATE	DELIVERY_DATE	PRICE
1	10	10	2022-11-13	2022-11-20	1000.0
8	10	555	2022-11-13	2022-11-20	275.0
12	10	21	2022-11-18	2022-11-25	210.0
11	1	25	2022-11-18	2022-11-25	120.0

Create Order
Enter Stock ID: 1
Enter Quantity: 10
Create

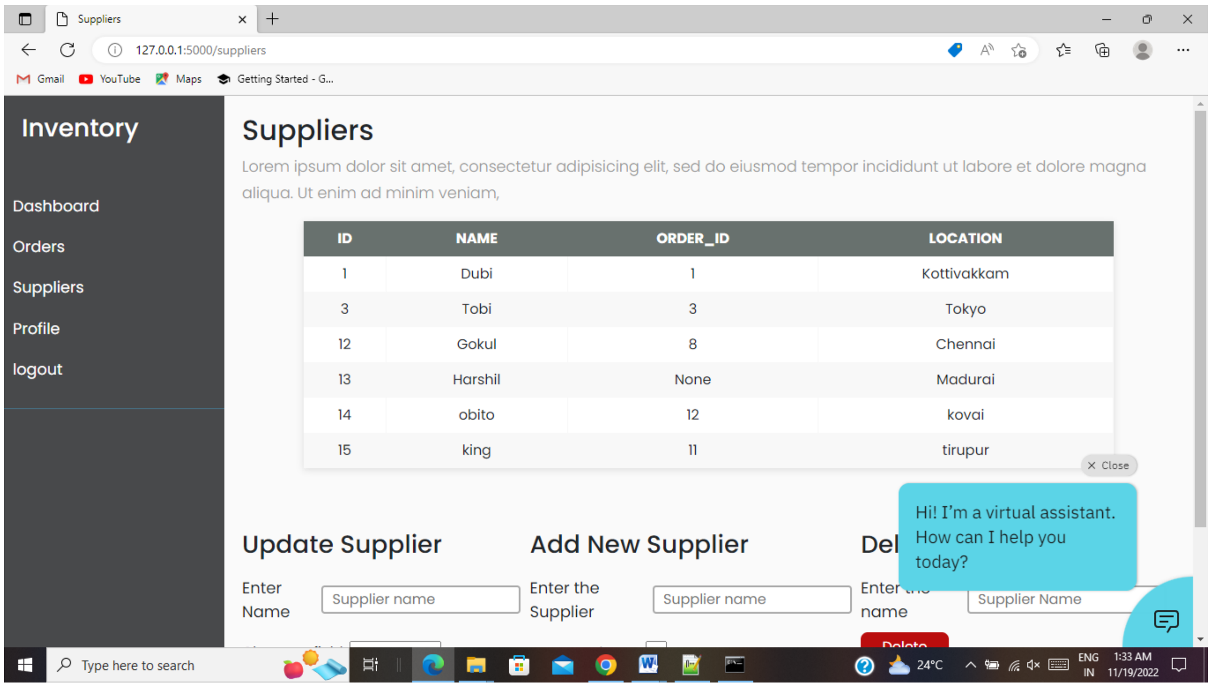
Update Order
Enter Order ID: 1
Choose a field - STOCKS_ID
Enter Value:
Update

Cancel Order
Enter Order ID: 1

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

SUPPLIERS

This page gives the details about the suppliers.



Suppliers

Inventory

Dashboard
Orders
Suppliers
Profile
logout

Suppliers

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

ID	NAME	ORDER_ID	LOCATION
1	Dubi	1	Kottivakkam
3	Tobi	3	Tokyo
12	Gokul	8	Chennai
13	Harshil	None	Madurai
14	obito	12	kovai
15	king	11	tirupur

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

Update Supplier

Enter Name [Supplier name]

Add New Supplier

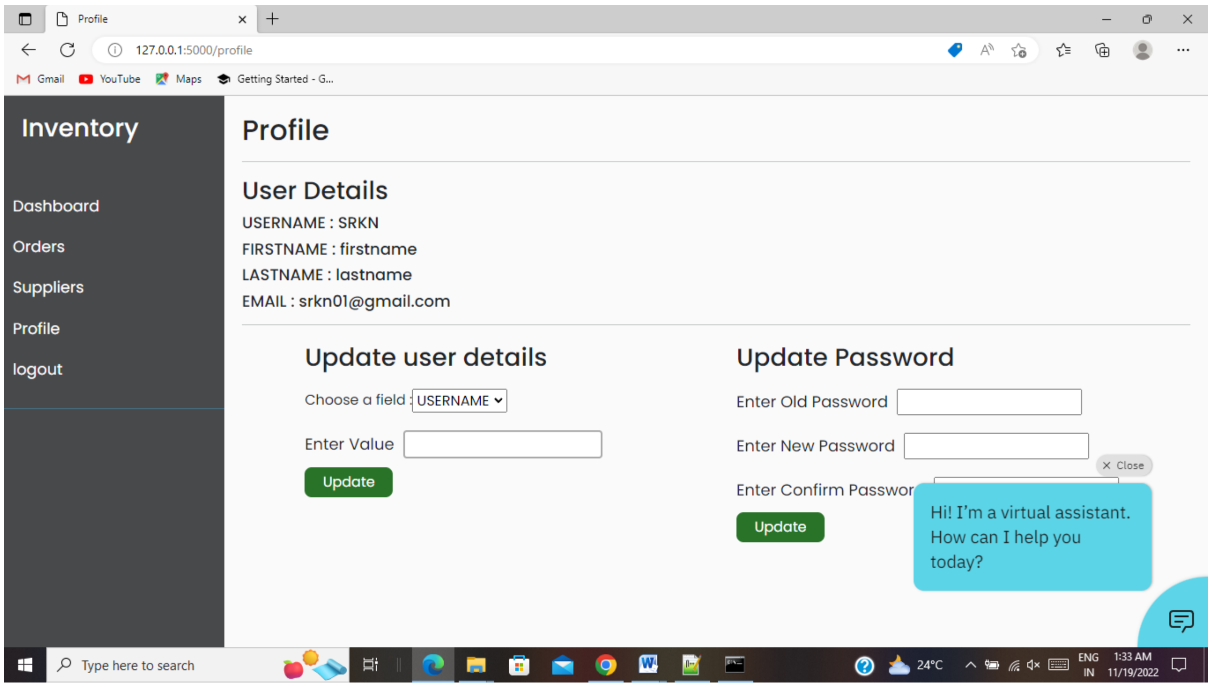
Enter the Supplier [Supplier name]

Del

Enter the name [Supplier Name]

PROFILE

In this page the user can check and update their profile.



Profile

Inventory

Dashboard
Orders
Suppliers
Profile
logout

Profile

User Details

USERNAME : SRKN
FIRSTNAME : firstname
LASTNAME : lastname
EMAIL : srkn01@gmail.com

Update user details

Choose a field : USERNAME

Enter Value

Update

Update Password

Enter Old Password

Enter New Password

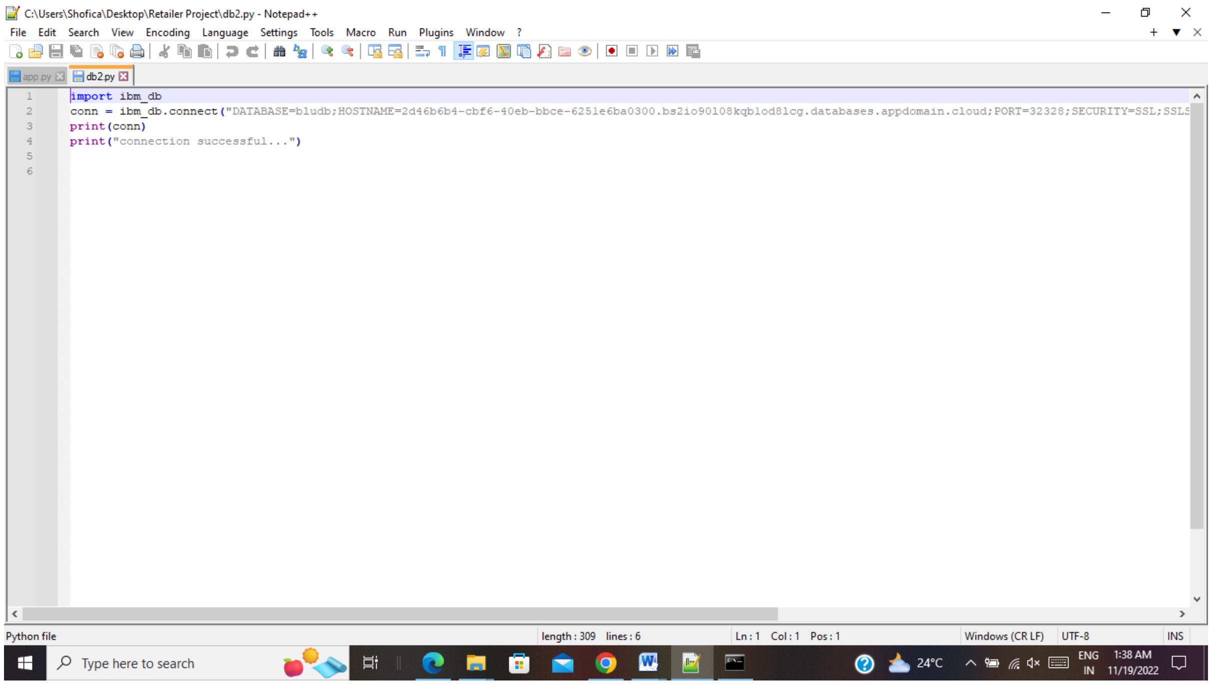
Enter Confirm Passwor

Update

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

7.3 DATABASE SCHEMA

The database used in this project is shown below.


C:\Users\Shofica\Desktop\Retailer Project\db2.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
app.py
db2.py

```
import ibm_db
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90108kqblod8lcg.databases.appdomain.cloud;PORT=32328;SECURITY=SSL;SSLS
print(conn)
print("connection successful...")
```

Python file
length : 309 lines : 6
Ln : 1 Col : 1 Pos : 1
Windows (CR LF)
UTF-8
INS
Type here to search
24°C
ENG IN
1:38 AM 11/19/2022

8. TESTING

8.1 Test Cases

Test case ID	Feature Type	Component	Test Scenario
LoginPage_TC_OO 1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup
LoginPage_TC_OO 3	Functional	Home page	Verify user is able to log into application with Valid credentials
LoginPage_TC_OO 4	Functional	Login page	Verify user is able to log into application with InValid credentials

LoginPage_TC_OO 4	Functional	Login page	Verify user is able to log into application with InValid credentials
LoginPage_TC_OO 5	Functional	Login page	Verify user is able to log into application with InValid credentials

8.2 User Acceptance Testing

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	19
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	6	0	0	6
Client Application	25	0	0	20
Security	2	0	0	2
Outsource Shipping	3	0	0	3

Exception Reporting	7	0	0	7
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. RESULTS

9.1 Performance Metrices

- Retailers today have more access to metrics than those in the past. As always, with metrics and business, if we can measure it, we can improve it — and retailers can improve their performance in a variety of ways. In this guide, we have compiled several of the most useful KPIs for tracking growth and performance in a retail business.
- The most common indicator of growth in retail is the sales volume. If you're selling more, then you're growing. However, growth encompasses more than just the number of sales, it also involves improving your processes. Improved processes can mean becoming efficient in reaching more customers, improving employee morale, and cost-effectively expanding or shrinking your inventory.
- In the end, those will translate to more sales and better business growth. Below are some of the most common retail KPIs to measure success.

10. ADVANTAGES & DISADVANTAGES

Advantages:

- Provides protection against fluctuations in demand and supply by monitoring the trends in demand and supply.
- Ensures a better service to the customers by avoiding the out of stock situations by keeping a check on the minimum stock levels.

- Helps to reduce risk of loss on account of obsolescence or deterioration of items.
- Helps to reduce administrative work load in respect of purchasing, inspection, store-keeping, etc. thus in turn reducing manpower requirements, and consequently costs.5. Helps to make effective utilization of working capital by avoiding its blockage in excess inventory
- It helps to maintain the right amount of stocks. The goal is to find that zone where you are never losing money in your inventory in either direction. With the aid of an efficient inventory management strategy, it is easy to improve the accuracy of inventory order.
- It leads to a more organized warehouse: with the aid of a good inventory management system, you can easily organize your warehouse.
- Increased information transparency: a good inventory management helps to keep the flow of information transparent.
- **A well-structured inventory management system leads to improved customer retention:** for customers to keep patronizing you, you will need to always have the goods they want, at the amount they want, and at the time they want it.

Disadvantages:

- Some inventory management systems such as the fixed order period system compels a periodic review of all items. This itself makes the system a bit inefficient.
- Even with an efficient inventory management method, you can control but not eliminate business risk.
- The control of inventory is complex because of the many functions it performs. It should thus be viewed as a shared responsibility.

- Holding inventory can result to a greater risk of loss to devaluation (changes in price).
- in order to hold inventory, you will need to have space so unless the goods you deal in are really small in size, then you will need a warehouse to store it. In addition, you will also need to buy shelves and racks to store your goods, forklifts to move around the stock and of course staff.

11. CONCLUSION

Inventory management is a useful method for simplifying all the warehousing activities of the organization. With this technique, the company can now access and determine its stock and inventory with efficiency to smoothen all the business operations.

It has also proved to be a valuable tool for maintaining the working capital requirement.

12. FUTURE SCOPE

- The Fourth Industrial Revolution will continue to drive technological change that will impact the way that we manage inventories.
- Successful companies will view inventory as a strategic asset, rather than an aggravating expense or an evil to be tolerated.
- Collaboration with supply chain partners, coupled with a holistic approach to supply chain management, will be key to effective inventory management.
- The nature of globalization will change, impacting inventory deployment decisions dramatically.

- Increased focus on supply chain security, and concerns about the quality of inventory itself, will be primary motivators to changing supply chain and inventory strategy.

13. APPENDIX

SOURCE CODE:

```
from flask import Flask, render_template, url_for, request, redirect, session, make_response

import sqlite3 as sql

from functools import wraps

import re

import ibm_db

import os

from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail

from datetime import datetime, timedelta

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=815fa4db-dc03-4c70-869a-
a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30367;SECURITY=SSL;S
SLServerCertificate=DigiCertGlobalRootCA.crt;UID=gkx49901;PWD=kvWCsySl7vApfsy2", '', '')

app = Flask(__name__)

app.secret_key = 'jackiechan'

def rewrite(url):

    view_func, view_args = app.create_url_adapter(request).match(url)

    return app.view_functions[view_func](**view_args)

def login_required(f):

    @wraps(f)

    def decorated_function(*args, **kwargs):

        if "id" not in session:

            return redirect(url_for('login'))

        return f(*args, **kwargs)
    return decorated_function
@app.route('/')
def root():
    return render_template('login.html')
@app.route('/user/<id>')
@login_required
def user_info(id):
    with sql.connect('inventorymanagement.db') as con:
        con.row_factory = sql.Row
        cur = con.cursor()
        cur.execute(f'SELECT * FROM users WHERE email="{id}"')
  user = cur.fetchall()
    return render_template("user_info.html", user=user[0])
@app.route('/login', methods=['GET', 'POST'])
def login():
    global userid
    msg = ''
    if request.method == 'POST':
        un = request.form['username']
        pd = request.form['password_1']
        print(un, pd)
        sql = "SELECT * FROM users WHERE email =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, un)
        ibm_db.bind_param(stmt, 2, pd)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account        if account:
            session['loggedin'] = True
            session['id'] = account['EMAIL']
            userid = account['EMAIL']
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'
        return rewrite('/dashboard')
        else:
msg = 'Incorrect username / password !'
    return render_template('login.html', msg=msg)
@app.route('/signup', methods=['POST', 'GET'])
def signup():
    mg = ''
    if request.method == "POST":
        username = request.form['username']
        email = request.form['email']
        pw = request.form['password']
        sql = 'SELECT * FROM users WHERE email =?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        acnt = ibm_db.fetch_assoc(stmt)
        print(acnt)
      if acnt:
            mg = 'Account already exits!!'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            mg = 'Please enter the avalid email address'
        elif not re.match(r'[A-Za-z0-9]+', username):
            ms = 'name must contain only character and number'
        else:
```

```
 insert_sql = 'INSERT INTO users
(USERNAME,FIRSTNAME,LASTNAME,EMAIL,PASSWORD) VALUES (?,?,?,?,?)'

            pstmt = ibm_db.prepare(conn, insert_sql)

            ibm_db.bind_param(pstmt, 1, username)

            ibm_db.bind_param(pstmt, 2, "firstname")

            ibm_db.bind_param(pstmt, 3, "lastname")

            # ibm_db.bind_param(pstmt,4,"123456789")

            ibm_db.bind_param(pstmt, 4, email)

            ibm_db.bind_param(pstmt, 5, pw)

            print(pstmt)

            ibm_db.execute(pstmt)

            mg = 'You have successfully registered click login!'

            message = Mail(

                from_email=os.environ.get('MAIL_DEFAULT_SENDER'),

                to_emails=email,

                subject='New SignUp',

                html_content='<p>Hello, Your Registration was successfull. <br><br> Thank you for
choosing us.</p>')

  sg = SendGridAPIClient(

                api_key=os.environ.get('SENDGRID_API_KEY'))

response = sg.send(message)

            print(response.status_code, response.body)

            return render_template("login.html", meg=mg)

    elif request.method == 'POST':

        msg = "fill out the form first!"

    return render_template("signup.html", meg=mg)

@app.route('/dashboard', methods=['POST', 'GET'])

@login_required

def dashBoard():
    sql = "SELECT * FROM stocks"

    stmt = ibm_db.exec_immediate(conn, sql)

    dictionary = ibm_db.fetch_assoc(stmt)

    stocks = []

    headings = [*dictionary]

    while dictionary != False:

        stocks.append(dictionary)

        # print(f"The ID is : ", dictionary["NAME"])

        # print(f"The name is : ", dictionary["QUANTITY"])

        dictionary = ibm_db.fetch_assoc(stmt)

 return render_template("dashboard.html", headings=headings, data=stocks)

@app.route('/addstocks', methods=['POST'])

@login_required

def addStocks():

    if request.method == "POST":

        print(request.form['item'])

        try:

            item = request.form['item']

  quantity = request.form['quantity']

            price = request.form['price']

            total = int(price) * int(quantity)

            insert_sql = 'INSERT INTO stocks
(NAME,QUANTITY,PRICE_PER_QUANTITY,TOTAL_PRICE) VALUES (?,?,?,?)'

            pstmt = ibm_db.prepare(conn, insert_sql)

            ibm_db.bind_param(pstmt, 1, item)

            ibm_db.bind_param(pstmt, 2, quantity)

            ibm_db.bind_param(pstmt, 3, price)

            ibm_db.bind_param(pstmt, 4, total)

            ibm_db.execute(pstmt)
```

```
   except Exception as e:
           msg = e
 finally:
           # print(msg)
           return redirect(url_for('dashBoard'))
@app.route('/updatestocks', methods=['POST'])
@login_required
def UpdateStocks():
    if request.method == "POST":
        try:
            item = request.form['item']
            print("hello")
            field = request.form['input-field'] value = request.form['input-value']
            print(item, field, value)
            insert_sql = 'UPDATE stocks SET ' + field + "= ?" + " WHERE NAME=?"
            print(insert_sql)
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, item)
            ibm_db.execute(pstmt)
            if field == 'PRICE_PER_QUANTITY' or field == 'QUANTITY':
                insert_sql = 'SELECT * FROM stocks WHERE NAME= ?'
                pstmt = ibm_db.prepare(conn, insert_sql)
                ibm_db.bind_param(pstmt, 1, item)
                ibm_db.execute(pstmt)
                dictonary = ibm_db.fetch_assoc(pstmt)
                print(dictonary)
                total = dictonary['QUANTITY'] * dictonary['PRICE_PER_QUANTITY']
                insert_sql = 'UPDATE stocks SET TOTAL_PRICE=? WHERE NAME=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, total)
            ibm_db.bind_param(pstmt, 2, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e
  finally:
            # print(msg)
            return redirect(url_for('dashBoard'))
@app.route('/deletestocks', methods=['POST'])
@login_required
def deleteStocks():
    if request.method == "POST":
        print(request.form['item'])
        try:
            item = request.form['item']
            insert_sql = 'DELETE FROM stocks WHERE NAME=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e
        finally:
            # print(msg)
            return redirect(url_for('dashBoard'))
@app.route('/update-user', methods=['POST', 'GET'])
@login_required
def updateUser():
    if request.method == "POST":
        try:
            email = session['id']
            field = request.form['input-field']
            value = request.form['input-value']
            insert_sql = 'UPDATE users SET ' + field + '= ? WHERE EMAIL=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, email)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e
        finally:
            # print(msg)
            return redirect(url_for('profile'))
@app.route('/update-password', methods=['POST', 'GET'])
@login_required
def updatePassword():
    if request.method == "POST":
        try:
            email = session['id']
            password = request.form['prev-password']
            curPassword = request.form['cur-password']
            confirmPassword = request.form['confirm-password']
            insert_sql = 'SELECT * FROM  users WHERE EMAIL=? AND PASSWORD=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, email)
            ibm_db.bind_param(pstmt, 2, password)
            ibm_db.execute(pstmt)
            dictionary = ibm_db.fetch_assoc(pstmt)
            print(dictionary)

            if curPassword == confirmPassword:

                insert_sql = 'UPDATE users SET PASSWORD=? WHERE EMAIL=?'

                pstmt = ibm_db.prepare(conn, insert_sql)

                ibm_db.bind_param(pstmt, 1, confirmPassword)

                ibm_db.bind_param(pstmt, 2, email)

                ibm_db.execute(pstmt)

        except Exception as e:

            msg = e

        finally:

            # print(msg)

            return render_template('result.html')

@app.route('/orders', methods=['POST', 'GET'])

@login_required

def orders():

    query = "SELECT * FROM orders"

    stmt = ibm_db.exec_immediate(conn, query)

    dictionary = ibm_db.fetch_assoc(stmt)

    orders = []

    headings = [*dictionary]

    while dictionary != False:

        orders.append(dictionary)

        dictionary = ibm_db.fetch_assoc(stmt)

    return render_template("orders.html", headings=headings, data=orders)

@app.route('/createOrder', methods=['POST'])

@login_required

def createOrder():

    if request.method == "POST":

        try:
            stock_id = request.form['stock_id']

            query = 'SELECT PRICE_PER_QUANTITY FROM stocks WHERE ID= ?'

            stmt = ibm_db.prepare(conn, query)

            ibm_db.bind_param(stmt, 1, stock_id)

            ibm_db.execute(stmt)

            dictionary = ibm_db.fetch_assoc(stmt)

            if dictionary:

                quantity = request.form['quantity']

                date = str(datetime.now().year) + "-" + str(

                    datetime.now().month) + "-" + str(datetime.now().day)

                delivery = datetime.now() + timedelta(days=7)

                delivery_date = str(delivery.year) + "-" + str(

                    delivery.month) + "-" + str(delivery.day)

                price = float(quantity) * \

                    float(dictionary['PRICE_PER_QUANTITY'])

                query = 'INSERT INTO orders
(STOCKS_ID,QUANTITY,DATE,DELIVERY_DATE,PRICE) VALUES (?,?,?,?,?)'

                pstmt = ibm_db.prepare(conn, query)

                ibm_db.bind_param(pstmt, 1, stock_id)

                ibm_db.bind_param(pstmt, 2, quantity)

                ibm_db.bind_param(pstmt, 3, date)

                ibm_db.bind_param(pstmt, 4, delivery_date)

                ibm_db.bind_param(pstmt, 5, price)

                ibm_db.execute(pstmt)

        except Exception as e:

            print(e)

        finally:

            return redirect(url_for('orders'))

@app.route('/updateOrder', methods=['POST'])
```

```
@login_required
def updateOrder():
    if request.method == "POST":
        try:
            item = request.form['item']
            field = request.form['input-field']
            value = request.form['input-value']
            query = 'UPDATE orders SET ' + field + "= ?" + " WHERE ID=?"
            pstmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            print(e)
        finally:
            return redirect(url_for('orders'))
@app.route('/cancelOrder', methods=['POST'])
@login_required
def cancelOrder():
    if request.method == "POST":
        try:
            order_id = request.form['order_id']
            query = 'DELETE FROM orders WHERE ID=?'
            pstmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(pstmt, 1, order_id)
            ibm_db.execute(pstmt)
        except Exception as e:
            print(e)
        finally:
            return redirect(url_for('orders'))

@app.route('/suppliers', methods=['POST', 'GET'])

@login_required

def suppliers():

    sql = "SELECT * FROM suppliers"

    stmt = ibm_db.exec_immediate(conn, sql)

    dictionary = ibm_db.fetch_assoc(stmt)

    suppliers = []

    orders_assigned = []

    headings = [*dictionary]

    while dictionary != False:

        suppliers.append(dictionary)

        orders_assigned.append(dictionary['ORDER_ID'])

        dictionary = ibm_db.fetch_assoc(stmt)

# get order ids from orders table and identify unassigned order ids

    sql = "SELECT ID FROM orders"

    stmt = ibm_db.exec_immediate(conn, sql)

    dictionary = ibm_db.fetch_assoc(stmt)

    order_ids = []

    while dictionary != False:

        order_ids.append(dictionary['ID'])

        dictionary = ibm_db.fetch_assoc(stmt)


    unassigned_order_ids = set(order_ids) - set(orders_assigned)

    return render_template("suppliers.html", headings=headings, data=suppliers,
order_ids=unassigned_order_ids)

@app.route('/updatesupplier', methods=['POST'])

@login_required

def UpdateSupplier():
    if request.method == "POST":
        try:
            item = request.form['name']
            field = request.form['input-field']
            value = request.form['input-value']
            print(item, field, value)
            insert_sql = 'UPDATE suppliers SET ' + field + "= ?" + " WHERE NAME=?"
            print(insert_sql)
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e
        finally:
            return redirect(url_for('suppliers'))
@app.route('/addsupplier', methods=['POST'])
@login_required
def addSupplier():
    if request.method == "POST":
        try:
            name = request.form['name']
            order_id = request.form.get('order-id-select')
            print(order_id)
            print("Hello world")
            location = request.form['location']
            insert_sql = 'INSERT INTO suppliers (NAME,ORDER_ID,LOCATION) VALUES (?,?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, name)
            ibm_db.bind_param(pstmt, 2, order_id)
            ibm_db.bind_param(pstmt, 3, location)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e
        finally:
            return redirect(url_for('suppliers')
@app.route('/deletesupplier', methods=['POST'])
@login_required
def deleteSupplier():
    if request.method == "POST":
        try:
            item = request.form['name']
            insert_sql = 'DELETE FROM suppliers WHERE NAME=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            return redirect(url_for('suppliers'))
@app.route('/profile', methods=['POST', 'GET'])
@login_required
def profile():
    if request.method == "GET":
        try:
            email = session['id']
            insert_sql = 'SELECT * FROM users WHERE EMAIL=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, email)
            ibm_db.execute(pstmt)
            dictionary = ibm_db.fetch_assoc(pstmt)
            print(dictionary)
        except Exception as e:
            msg = e
        finally:
            # print(msg)
            return render_template("profile.html", data=dictionary)
@app.route('/logout', methods=['GET'])
@login_required
def logout():
    print(request)
    resp = make_response(render_template("login.html"))
    session.clear()
    return resp
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Github & Demo link

Github Link

https://github.com/IBM-EPBL/IBM-Project-45119-1660728353

Demo Link

https://youtu.be/FN-L5htsQK8