INVENTORY MANAGEMENT SYSTEM

Team id	PNT2022TMID10872
Project Name	Inventory Stock Management for Retailers.

TeamMembers	ABHIRAMI.C(811519104001)
	ABINAYA.B(811519104003)
	ABINAYA.K(811519104004),
	R.JAYASHREE(811519104045)

Table Of Contents

INTRODUCTION

PROJECT OVERVIEW:

Inventory management is the process of monitoring and controlling inventory level and ensuring

adequate replenishment in order to meet customer demand. Determining the appropriate inventory

level is crucial since inventory ties up money and affects performance. Having too much inventory

reduces the working capital and impacts the company's liquidity. On the contrary, having too little

inventory leads to stock outs and missed sales which leads to less profit. It becomes clear that

management attention should be focused on keeping inventory level somewhere in between, striving

for increased customer satisfaction and minimum stock outs while keeping inventory as possible as in low cost.

PURPOSE:

Retail inventory management is the process of ensuring you carry merchandise that shoppers want, with neither too little nor too much on hand.

By managing inventory, retailers meet customer demand without running out of stock or carrying excess supply.

In practice, effective retail inventory management results in lower costs and a better understanding of sales patterns.

Retail inventory management tools and methods give retailers more information on which to run their businesses.

Applications have been developed to help retailers track and manage stocks related to their own products.

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.

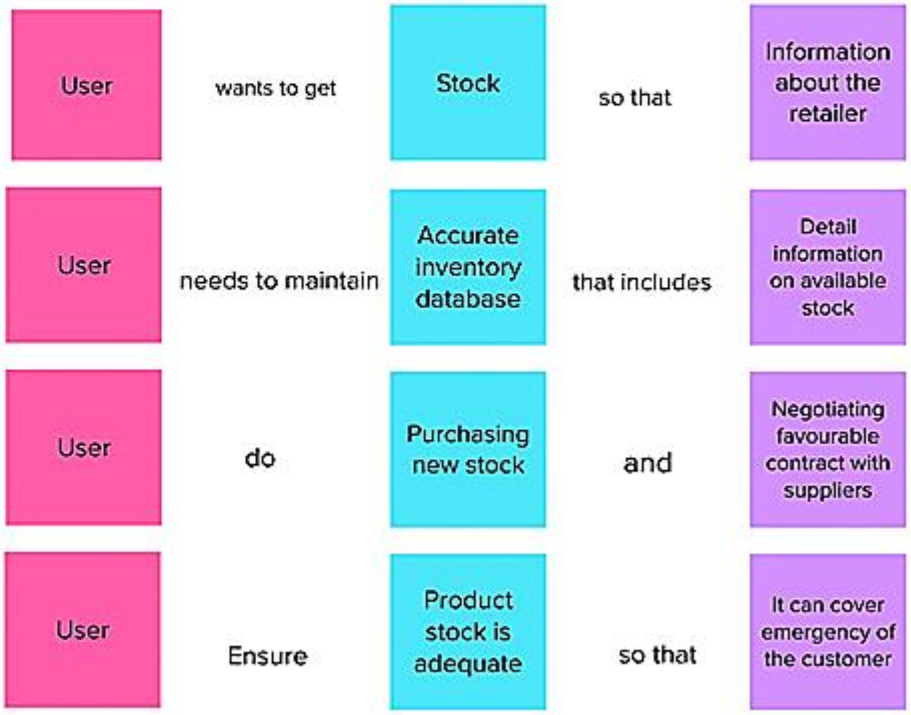
So that they can order new stock.

LITERATURE SURVEY

EXISTING PROBLEM:

1. Cannot Upload and Downloadthe latest updates.
2. No use of Web Services and Remoting.
3. Risk of mismanagement and of data when the project is under development.
4. Less Security.
5. No propercoordination between differentApplications and Users.
6. Fewer Users –Friendly

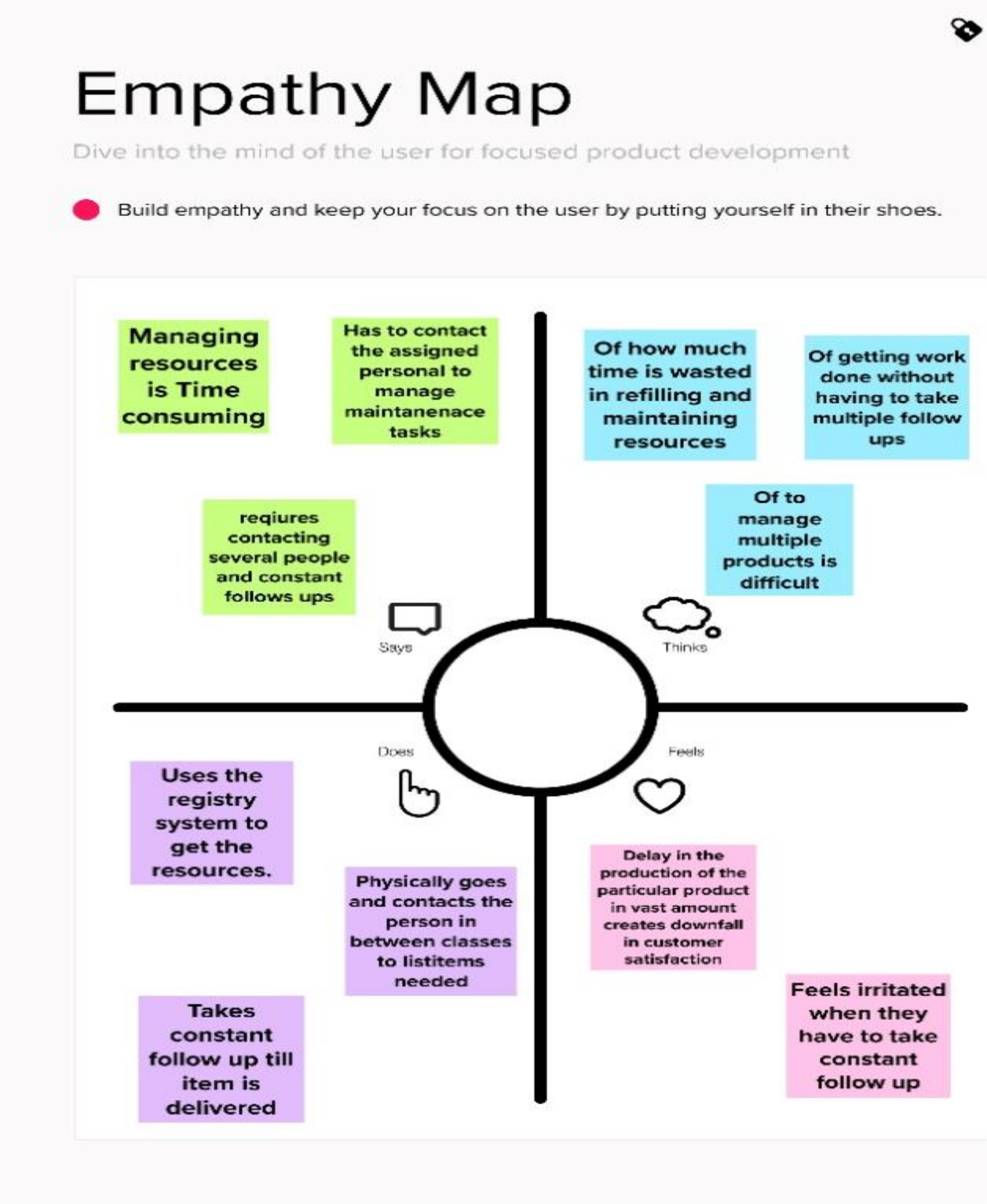
PROBLEM STATEMENT:



User	wants to get	Stock	so that	Information about the retailer
User	needs to maintain	Accurate inventory database	that includes	Detail information on available stock
User	do	Purchasing new stock	and	Negotiating favourable contract with suppliers
User	Ensure	Product stock is adequate	so that	It can cover emergency of the customer

IDEATION & PROPOSED SOLUTION

Empathy Map Canvas:



Empathy Map

Dive into the mind of the user for focused product development

● Build empathy and keep your focus on the user by putting yourself in their shoes.

Says

Managing resources is Time consuming

Has to contact the assigned personal to manage maintanenace tasks

reqiures contacting several people and constant follows ups

Thinks

Of how much time is wasted in refilling and maintaining resources

Of getting work done without having to take multiple follow ups

Of to manage multiple products is difficult

Does

Uses the registry system to get the resources.

Physically goes and contacts the person in between classes to listitems needed

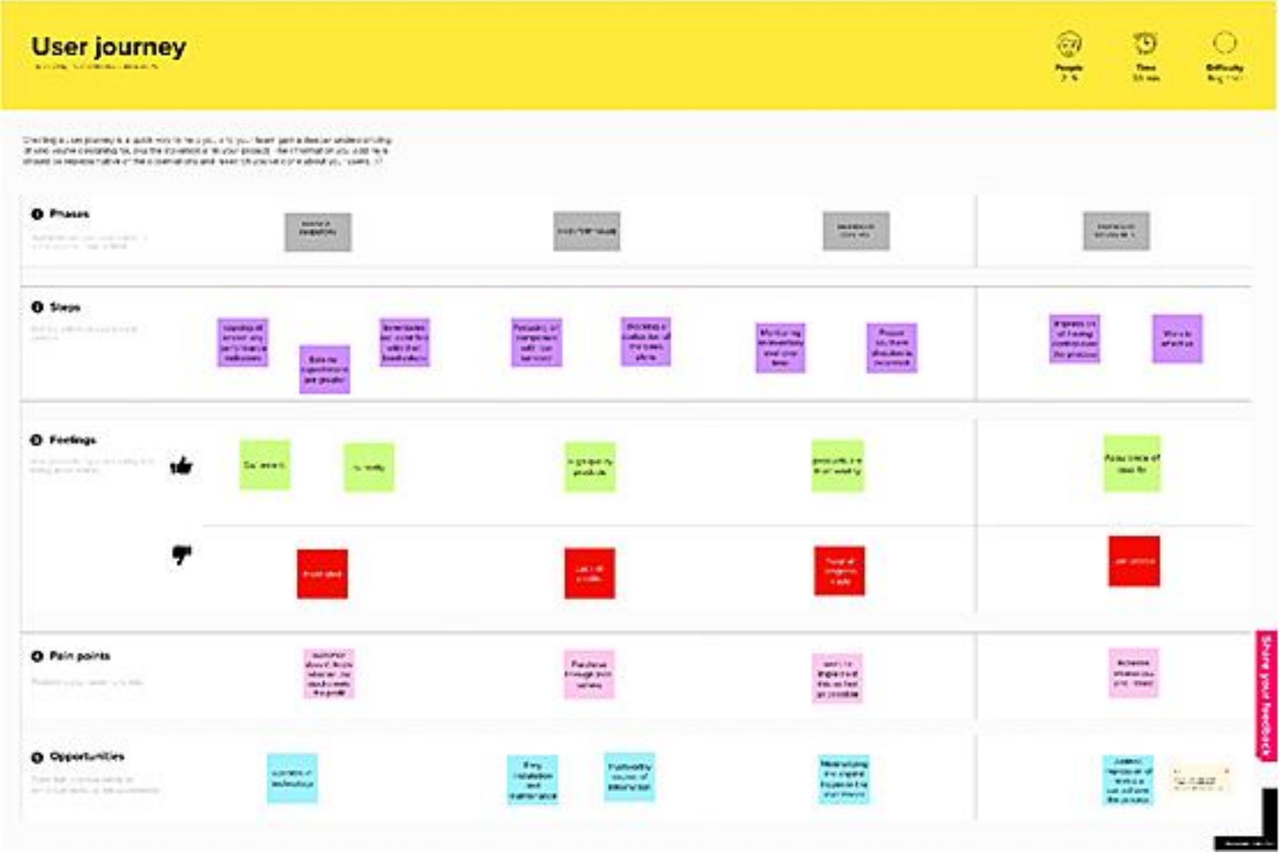
Takes constant follow up till item is delivered

Feels

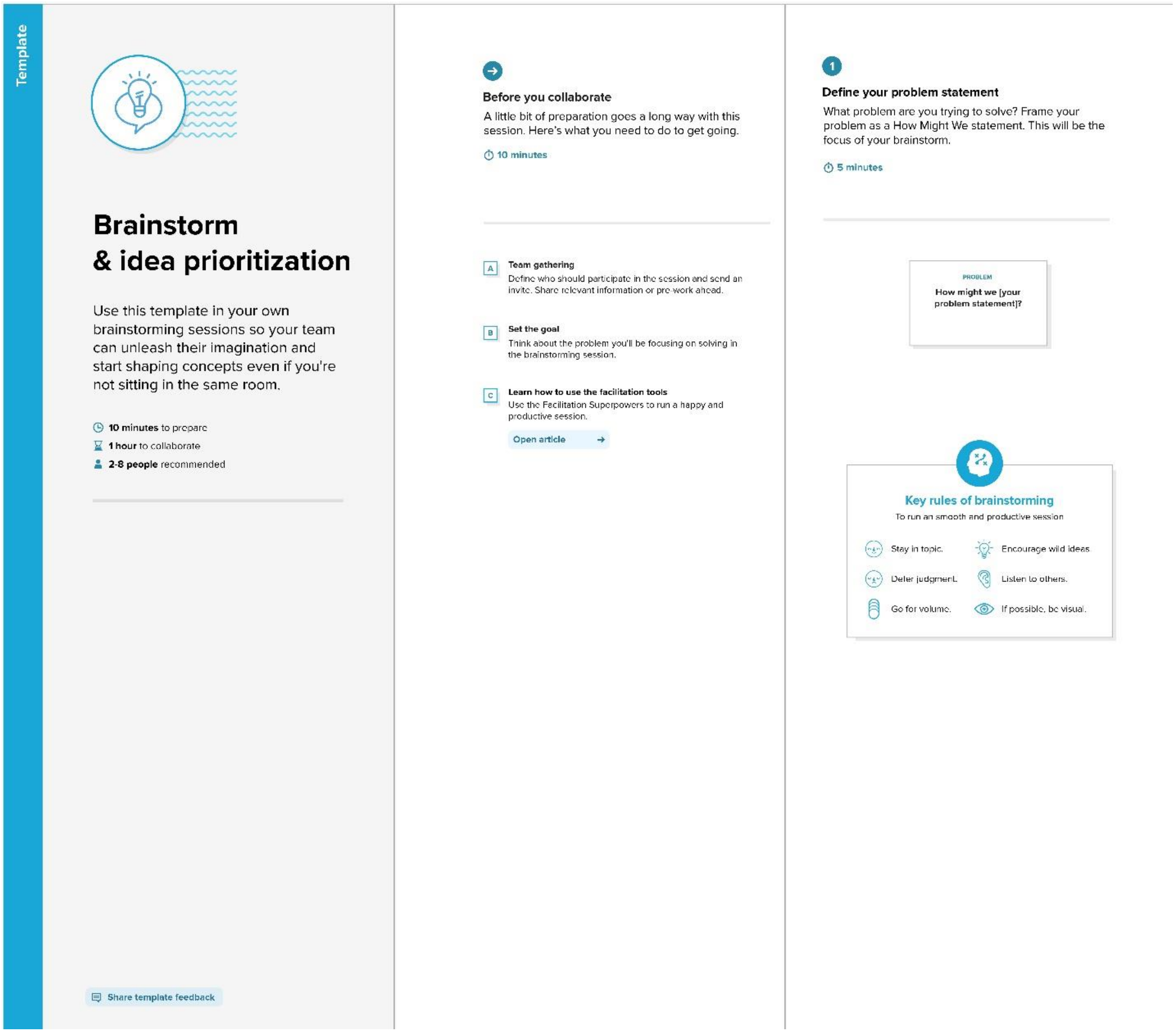
Delay in the production of the particular product in vast amount creates downfall in customer satisfaction

Feels irritated when they have to take constant follow up

USER JOURNEY


User journey
People
Time
Difficulty
1 Phases
2 Steps
3 Feelings
4 Pain points
5 Opportunities
Share your feedback

Brainstorming &idea prioritization:



Template

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
2-8 people recommended

Share template feedback

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

B **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.

C **Learn how to use the facilitation tools**
Use the Facilitation Superpowers to run a happy and productive session.

Open article →

1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM
How might we [your problem statement]?

Key rules of brainstorming
To run an smooth and productive session

Stay in topic.
Encourage wild ideas.
Defer judgment.
Listen to others.
Go for volume.
If possible, be visual.

2

Brainstorm

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

10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing.

Jayashree R

K. Abinaya

C. Abhirami

B. Abinaya

3

Group ideas

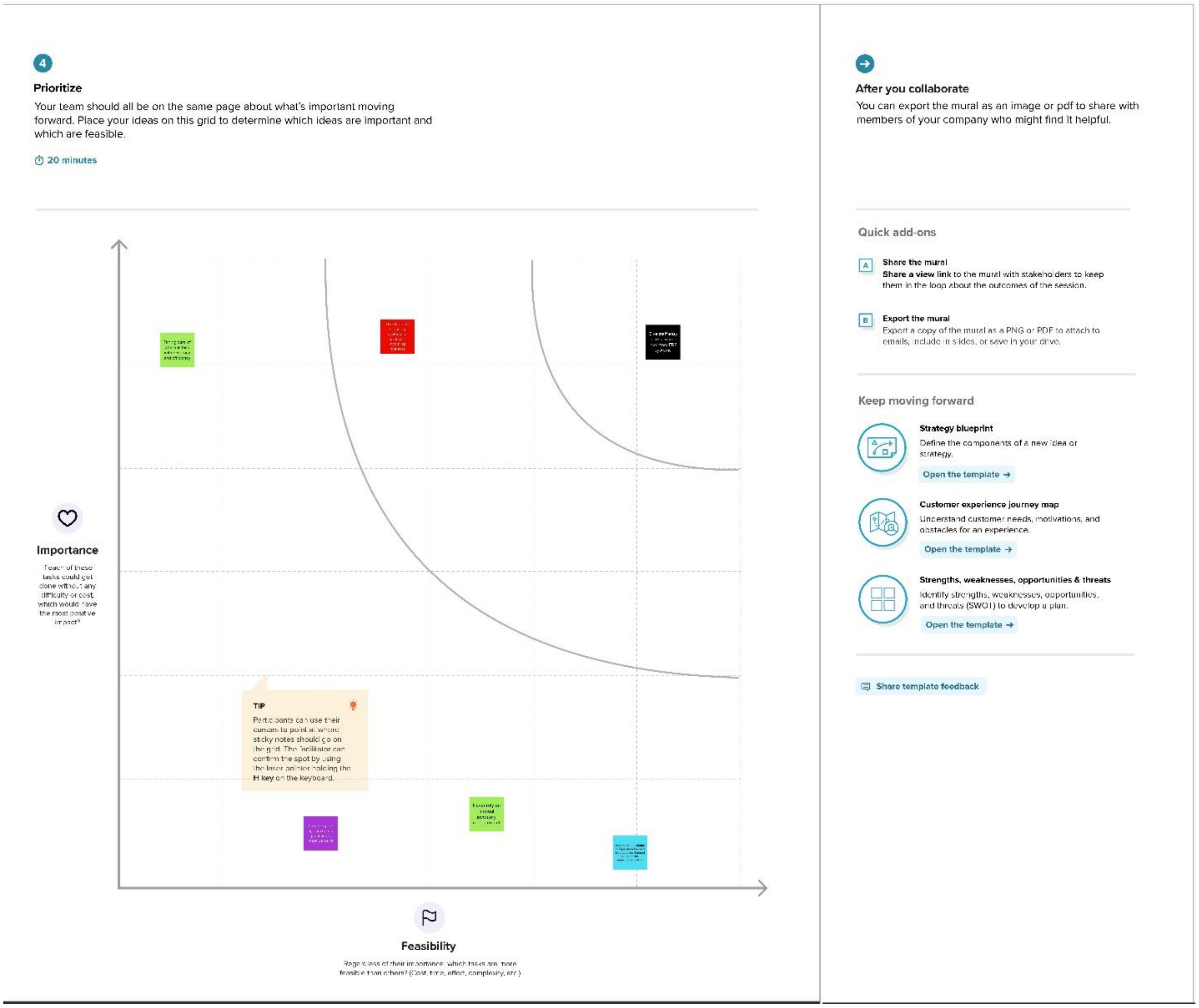
Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

Integrate mobile technology
Automate your inventory management system
Use past sales data to predict future demand



4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

⏱ 20 minutes

Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Taking care of [illegible] with [illegible] and efficiency
[illegible]
Overall Policy [illegible] new ERP system
Reasonably [illegible] manual inventory [illegible]
[illegible]
[illegible]
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H key** on the keyboard.

→

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

A **Share the mural**
Share a view link to the mural with stakeholders to keep them in the loop about the outcomes of the session.

B **Export the mural**
Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

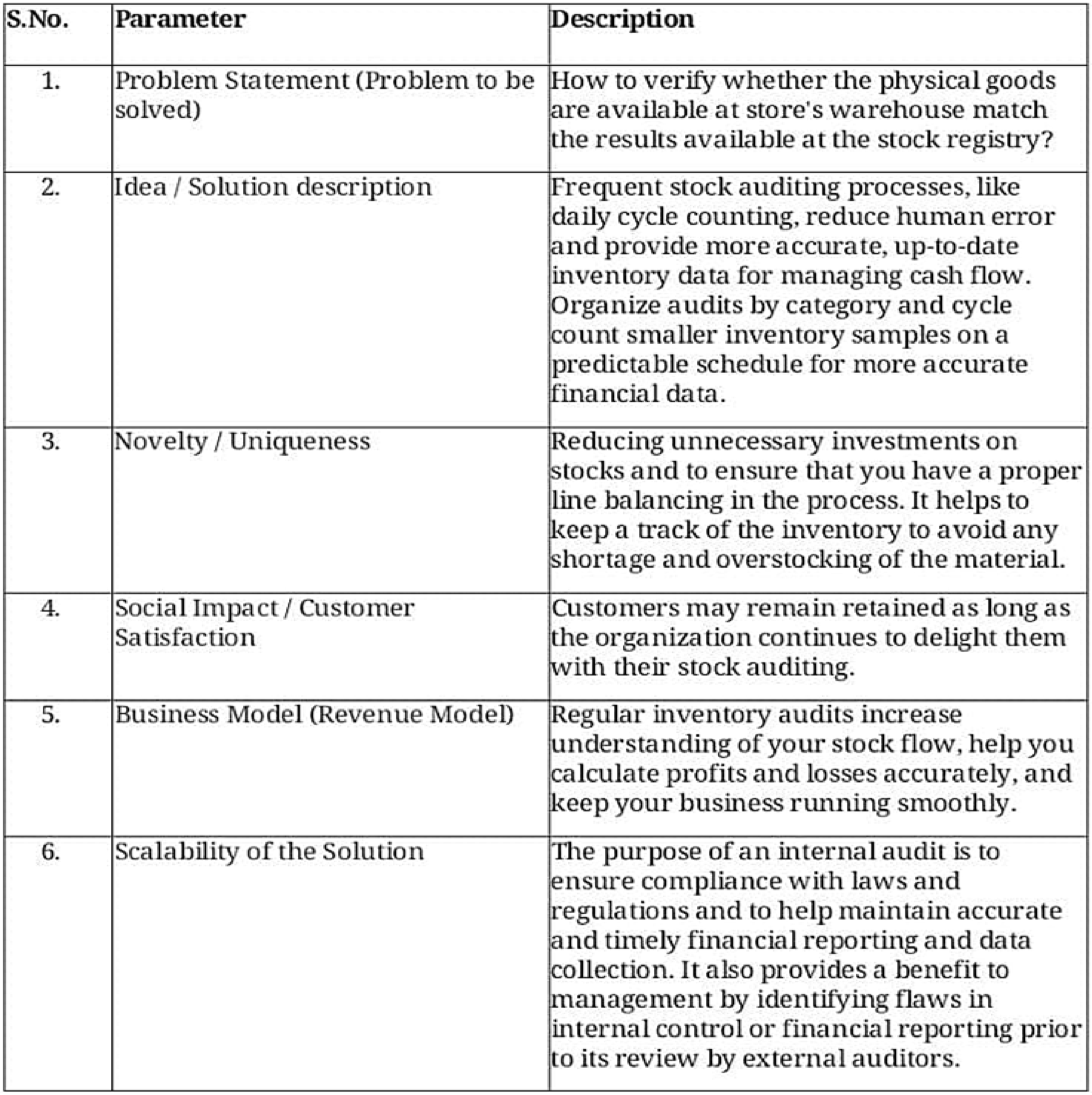
Strategy blueprint
Define the components of a new idea or strategy.
Open the template →

Customer experience journey map
Understand customer needs, motivations, and obstacles for an experience.
Open the template →

Strengths, weaknesses, opportunities & threats
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
Open the template →

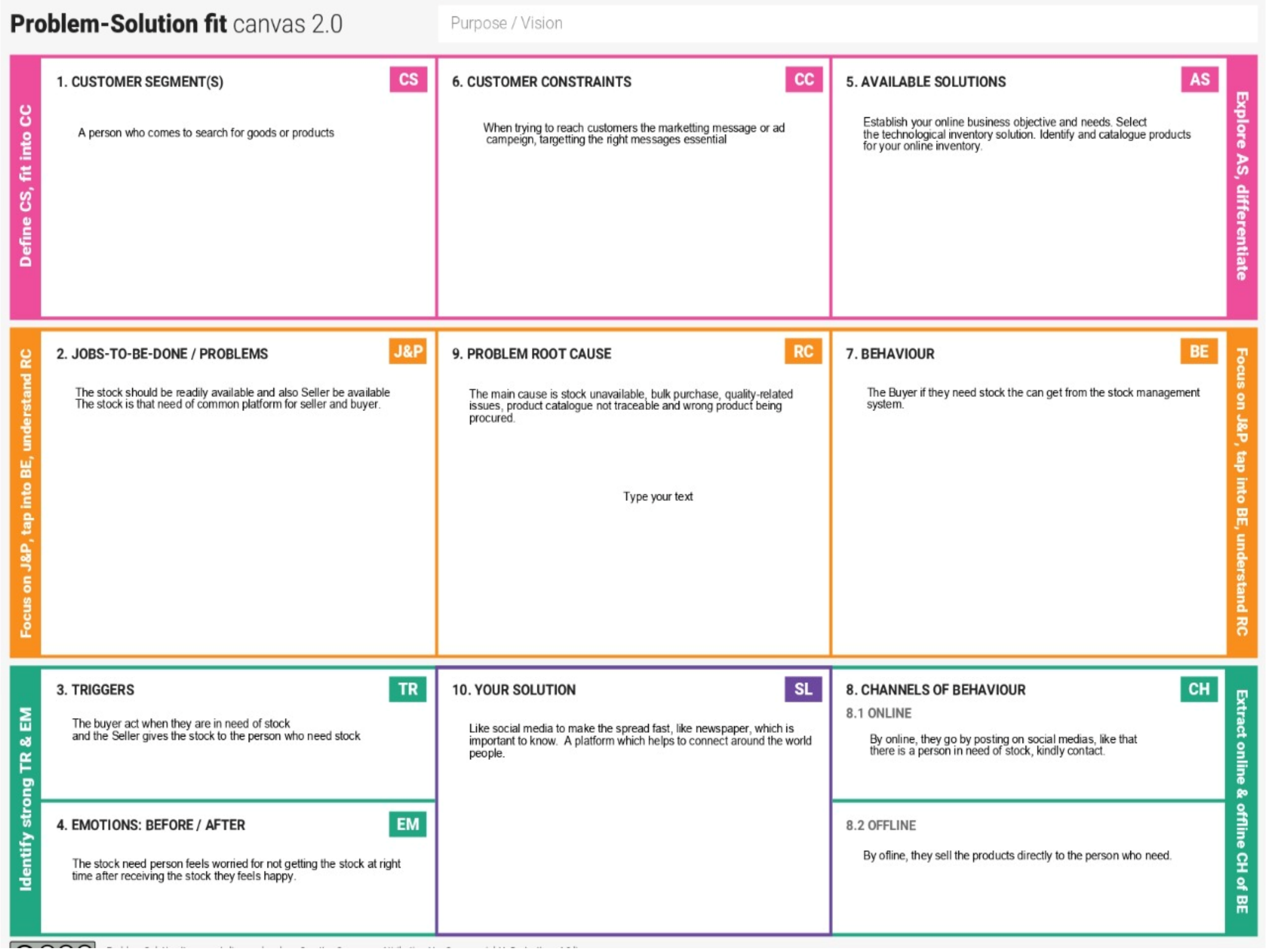
Share template feedback

Proposed SolutionTemplate:



S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	How to verify whether the physical goods are available at store's warehouse match the results available at the stock registry?
2.	Idea / Solution description	Frequent stock auditing processes, like daily cycle counting, reduce human error and provide more accurate, up-to-date inventory data for managing cash flow. Organize audits by category and cycle count smaller inventory samples on a predictable schedule for more accurate financial data.
3.	Novelty / Uniqueness	Reducing unnecessary investments on stocks and to ensure that you have a proper line balancing in the process. It helps to keep a track of the inventory to avoid any shortage and overstocking of the material.
4.	Social Impact / Customer Satisfaction	Customers may remain retained as long as the organization continues to delight them with their stock auditing.
5.	Business Model (Revenue Model)	Regular inventory audits increase understanding of your stock flow, help you calculate profits and losses accurately, and keep your business running smoothly.
6.	Scalability of the Solution	The purpose of an internal audit is to ensure compliance with laws and regulations and to help maintain accurate and timely financial reporting and data collection. It also provides a benefit to management by identifying flaws in internal control or financial reporting prior to its review by external auditors.

PROBLEM SOLUTIONFIT:



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

A person who comes to search for goods or products

6. CUSTOMER CONSTRAINTS CC

When trying to reach customers the marketting message or ad campeign, targetting the right messages essential

5. AVAILABLE SOLUTIONS AS

Establish your online business objective and needs. Select the technological inventory solution. Identify and catalogue products for your online inventory.

Explore AS, differentiate

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

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

The stock should be readily available and also Seller be available
The stock is that need of common platform for seller and buyer.

9. PROBLEM ROOT CAUSE RC

The main cause is stock unavailable, bulk purchase, quality-related issues, product catalogue not traceable and wrong product being procured.

Type your text

7. BEHAVIOUR BE

The Buyer if they need stock the can get from the stock management system.

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

Identify strong TR & EM

3. TRIGGERS TR

The buyer act when they are in need of stock
and the Seller gives the stock to the person who need stock

4. EMOTIONS: BEFORE / AFTER EM

The stock need person feels worried for not getting the stock at right time after receiving the stock they feels happy.

10. YOUR SOLUTION SL

Like social media to make the spread fast, like newspaper, which is important to know. A platform which helps to connect around the world people.

8. CHANNELS OF BEHAVIOUR CH

8.1 ONLINE

By online, they go by posting on social medias, like that there is a person in need of stock, kindly contact.

8.2 OFFLINE

By ofline, they sell the products directly to the person who need.

Extract online & offline CH of BE

REQUIREMENT ANALYSIS

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	Stock availability	List of stock available in home page
FR-4	Retailer details	Contact information about retailer
FR-5	Retailer Needs	Need status of retailer

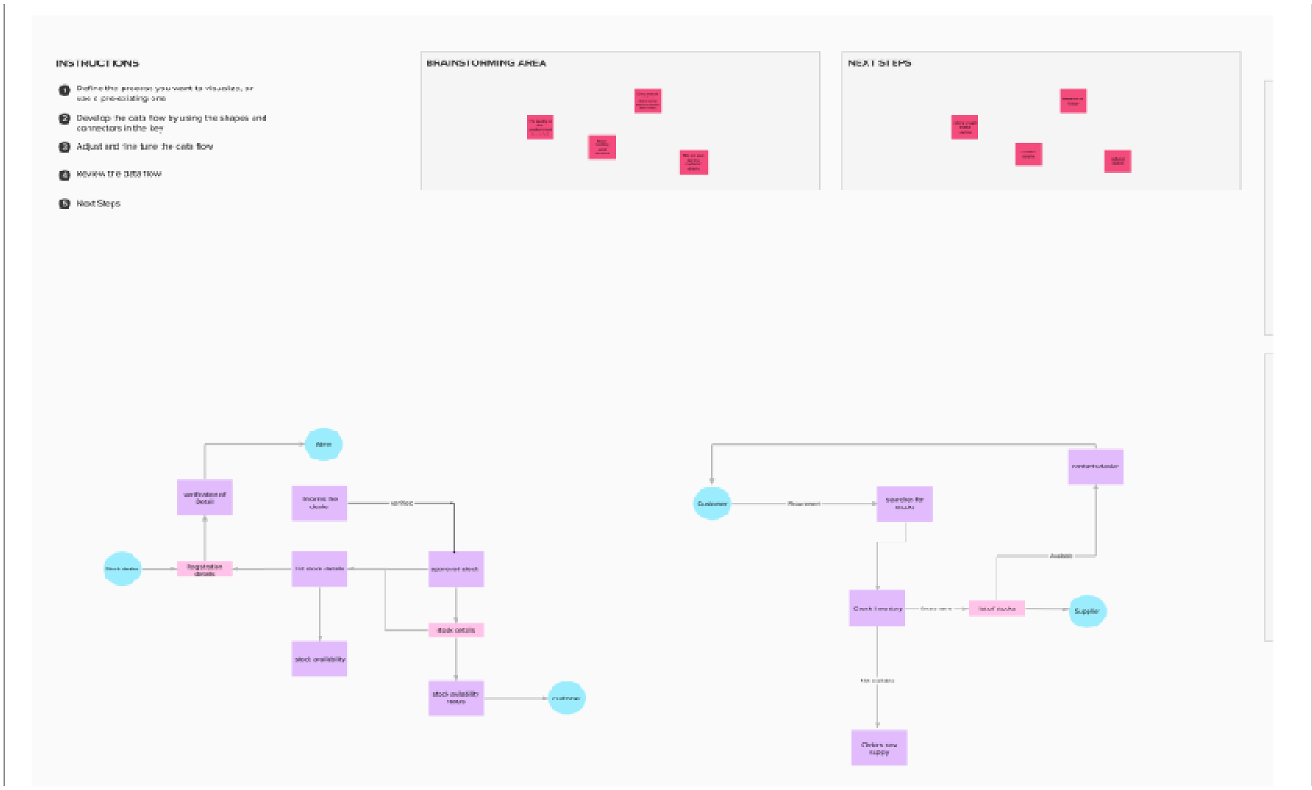
Non-functional Requirements:

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

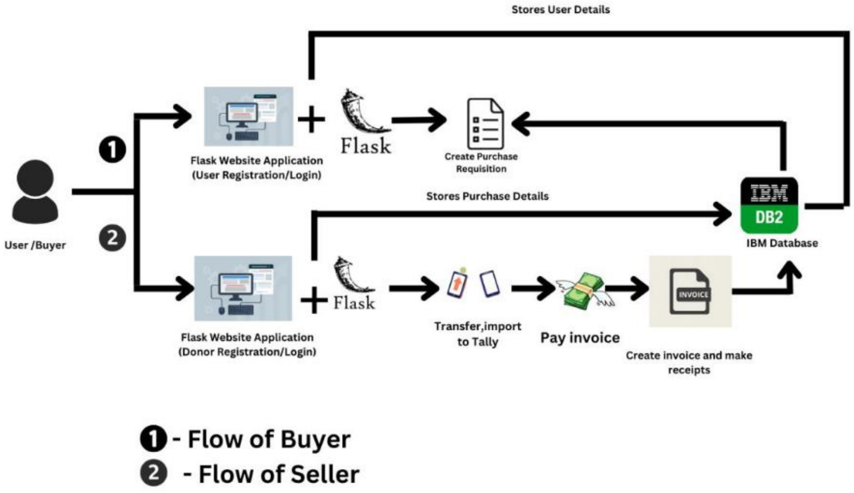
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Easy to find the seller and user-friendly website
NFR-2	**Security**	The seller are authorised so it was secured to contact the retailer
NFR-3	**Reliability**	Retailer notifications are available
NFR-4	**Performance**	There will be less network traffic
NFR-5	**Availability**	We can see the status of the stock available in seller profile
NFR-6	**Scalability**	It will be much useful for the retailer in need of Stock

PROJECT DESIGN

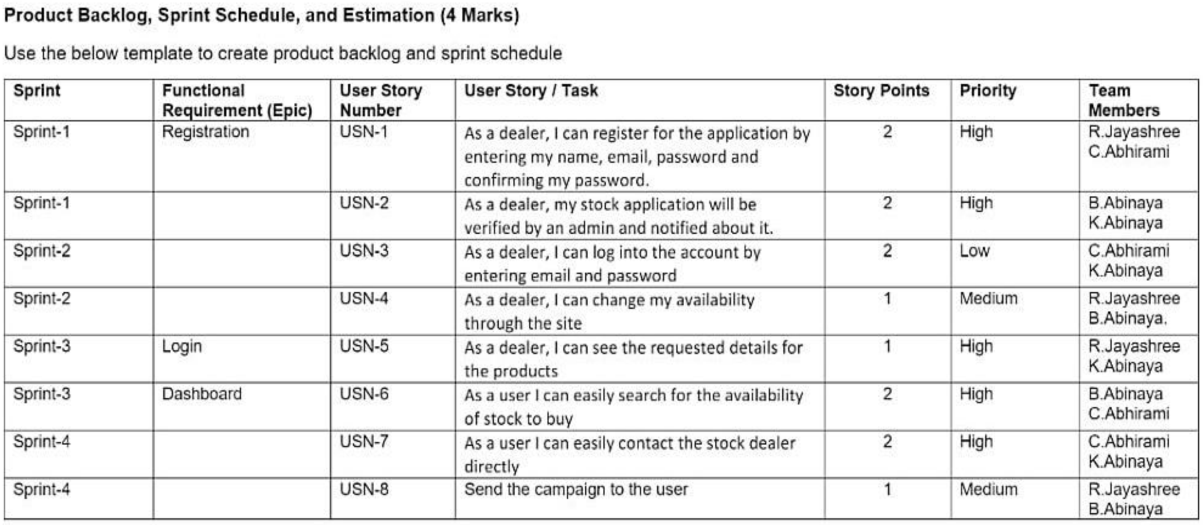
DataFlow Diagrams



Solution & Technical Architecture



PROJECT PLANNING & SCHEDULING



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

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a dealer, I can register for the application by entering my name, email, password and confirming my password.	2	High	R.Jayashree C.Abhirami
Sprint-1		USN-2	As a dealer, my stock application will be verified by an admin and notified about it.	2	High	B.Abinaya K.Abinaya
Sprint-2		USN-3	As a dealer, I can log into the account by entering email and password	2	Low	C.Abhirami K.Abinaya
Sprint-2		USN-4	As a dealer, I can change my availability through the site	1	Medium	R.Jayashree B.Abinaya.
Sprint-3	Login	USN-5	As a dealer, I can see the requested details for the products	1	High	R.Jayashree K.Abinaya
Sprint-3	Dashboard	USN-6	As a user I can easily search for the availability of stock to buy	2	High	B.Abinaya C.Abhirami
Sprint-4		USN-7	As a user I can easily contact the stock dealer directly	2	High	C.Abhirami K.Abinaya
Sprint-4		USN-8	Send the campaign to the user	1	Medium	R.Jayashree B.Abinaya

Sprint Delivery Schedule

Sprint	TotalStory Points	Duration	Sprint Start Date	Sprint End Date(Planned)	Story Points Completed (as on Planned EndDate)	Sprint Release Date (Actual)
Sprint-1	8	5 Days	27 Oct 2022	31 Nov 2022	8	03 Nov 2022
Sprint-2	13	4 Days	01 Nov 2022	06 Nov 2022	12	07 Nov 2022
Sprint-3	11	5 Days	07 Nov 2022	12 Nov 2022	11	09 Nov 2022
Sprint-4	9	5 Days	14 Nov 2022	19 Nov 2022	8	15 Nov 2022

CODING& SOLUTIONING

FEATURE 1:

Python

It is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significantindentation.[33]

Python is dynamically-typed and garbage-collected. It supports multipleprogramming paradigms, including structured (particularly procedural),object-oriented "https://en.wikipedia.org/wiki/Object-oriented_programming" and functional "https://en.wikipedia.org/wiki/Functional_programming" "https://en.wikipedia.org/wiki/Functional_programming"programming.

It is often described as a "batteries included" language due to its comprehensive standard "https://en.wikipedia.org/wiki/Standard_library" "https://en.wikipedia.org/wiki/Standard_library"library. HYPERLINK "https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-About-34"][35]

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC "https://en.wikipedia.org/wiki/ABC_(programming_language)" "https://en.wikipedia.org/wiki/ABC_(programming_language)"programming"https://en.wikipedia.org/wiki/ABC_(programming_language)" "https://en.wikipedia.org/wiki/ABC_(programming_language)"language "https://en.wikipedia.org/wiki/ABC_(programming_language)" and first released it in 1991as Python0.9.0.[36]

Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, referencecounting, and Unicode "https://en.wikipedia.org/wiki/Unicode" support. Python 3.0, released in 2008, was a major revision that is not completely backward "https://en.wikipedia.org/wiki/Backward_compatibility"- "https://en.wikipedia.org/wiki/Backward_compatibility"compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.[37]

Python consistently ranks as one of the most popular programming languages

FEATURE 2:

Flask Flask is a micro web framework "https://en.wikipedia.org/wiki/Web_framework" written in Python. It is classified as a micro "https://en.wikipedia.org/wiki/Microframework" "https://en.wikipedia.org/wiki/Microframework"framework"https://en.wikipedia.org/wiki/Microframework" because it does not require particular tools or libraries.[2]

It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide commonfunctions.

However, Flask supportsextensions that can add application features as if they were implemented in Flask itself. Extensions exist For object "https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping"- "https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping"relational "https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping" "https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping"mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Database SchemaIBM Db2-

a hybrid ANSI-compliant data virtualization tool for accessing, querying and summarizing data across the enterprise which:

Provides a massively parallel processing (MPP) architecture Exploits Hive, HBase and Apache Spark concurrently for best-in-class analytic capabilities

Requires only a single database connection or query to connect disparate sources such as HDFS, RDMS, NoSQL databases, objectstores andWeb HDFS

Provides low latency supportfor ad-hoc and complex queries,high performance, and federation capabilities

Understands dialectsfrom other vendorsand various productsfrom Oracle, IBM® Db2® and IBM Netezza®

Enables advancedrow and column security

KUBERNATES-

Kubernetes — also known as “k8s” or “kube” — is a containerorchestration platform for scheduling and automating the deployment,management, and scaling of containerized applications.

Kubernetes was first developed by engineers at Google before being open sourced in 2014. It is a descendant of Borg, a container orchestrationplatform used internallyat Google. Kubernetesis Greek
for *helmsman* or *pilot*, hence the helm in the Kubernetes logo (link resides outside IBM).

Today, Kubernetes and the broadercontainer ecosystem are maturing into a general-purpose computing platform and ecosystem that rivals — if notsurpasses — virtual machines (VMs) as the basic building blocks of modern cloud infrastructure and applications.

This ecosystem enables organizations to deliver a high- productivity Platform-as-a-Service (PaaS) that addresses multipleinfrastructure-related and operations-related tasks and issues
surrounding cloud-native development so that development teams can focus solely on coding and innovation.

TESTING

TESTING CASE:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault orweakness in a work product.

It provides a way to check the functional it your components, sub-assemblies, assemblies and/or a finishedproduct It is the process of exercising software with the intent of ensuring that the Softwaresystem meets its requirements and user expectation and does not fail in an unacceptable manner.

There are various types of test. Each test type addressesa specific testingrequirement

ACCEPTANCE TESTING

Acceptance Testing
UAT Execution & Report Submission

1. Purposeof Document

The purpose of this document is to briefly explain the test coverage and openissues of the Inventory Stock Management project at the time of the release to User AcceptanceTesting (UAT).

2 .Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
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

3. Test Case Analysis

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

Section	Total Cases	Not available	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

RESULTS

PERFORMANCE METRICS:

Project metricsare used to track the progress and performance of a project.

Monitoring parts of a project like productivity, scheduling, and scope makeit easier for team leadersto see what's on track.

As a project evolves, managers need access to changing deadlines or budgets to meet their client's expectations

ADVANTAGES & DISADVANTAGES

ADVANTAGES:

1. Speed: This website is fastand offers greataccuracy ascomparedto manual registered keeping.
2. Maintenance : Less maintenance is required
3. User Friendly: It is very easy to use and understand. It iseasilyworkable and accessible for everyone.
4. Fast Results: It would help you to provide plasma donorseasily depending upon the availability of it.

DISADVANTAGES:

1. Internet: It would require an internet connection for theworkingof the website.
2. Auto- Verification: It cannot automatically verify the genuine

   users.

   .

CONCLUSIONS

The efficient way of finding Inventory Stock Management for the people is implemented using the Inventory Stock Management website that is hosted onIBM Cloud platform.

To ensure the smooth functioning of the web site operation. I have hosted the website in IBM Db2 & Kubernates Cluster to make sure the operations are running successfully Cloud lambda function is used and to deploy the application IBM Db2 serviceis used.

FUTURE ENHANCEMENTS

Upgrading the UI that is more user-friendly which will helpmany users to access the website and also ensures that many plasma donorscan be added into the community.

Using elastic load balancer, it helps to handle multiple requests at the same time which will maintain the uptime of the websitewith negligible downtime.

APPENDIXES

SOURCE CODE:LOGIN.HTML

```
{% extends 'base.html' %} {% block head %}
<title>Signin Page</title>
{% endblock %} {% block body %}
<div class="body-full">
<div class="container body-full">
<div class="row">
<div class="col-md-8">
<h1 class="home">SignIn Page</h1></div>
<div class="col-md-4">
<br /><br />
<div class="card-body">
<form>
<label for="email">Email:</label><br />
<input type="email" class="form-control" name="email" required/><br />
<label for="password">Password:</label><br />
<input type="password" class="form-control" name="password" required/><br />
<input type="submit" value="Sign In" class="btn btn-primary" /><br /><br />
<a href="/signup" class="signup">Create a new account</a>
</form> </div> </div> </div>   </div> </div>
{% endblock %}
```

REGISTER.HTML

```
.{% extends 'base.html'%} {% block head %}
<title>Signup page</title>
{% endblock%} {%block body%}
<main class="container">
  <div class="mx-auto mt-5 border bg-light" style="width: 500px">
    <h2 class="mx-4 mt-2">REGISTER FORM</h2>
    <div class="mx-4 mt-2 text-danger">{{ meg }}</div>
    <form action="{{url_for('signup') }}" method="post">
      <div class="my-2 mx-4">
  <label for="username">User name</label>
  <input
    type="text"
    class="form-control"
    placeholder="Ram"
    name="username"
    required
  />
</div>
<div class="my-2 mx-4">
  <label for="email">email</label>
  <input
    type="text"
    class="form-control"
    placeholder="abc@gmail.com"
    name="email"
    required
  />
</div>
<div class="my-2 mx-4">
  <label for="password">password</label>
  <input
    type="password"
    class="form-control"
    placeholder="password"
    name="password"
    required
  />
</div>
<div class="my-2 mx-4">
  <label for="retype">retype password</label>
  <input
    type="password"
    class="form-control"
    placeholder="password"
    name="retype"
    required
```

```
        />
      </div>

      <input
        type="submit"
        value="submit"
        class="btn btn-primary my-4 my-2 mt-2 mx-4"
      />
      <div class="note mt-3 text-center">
        <!--Register form -->
        <p>already have an account ? please <a href="/login"> login! </a></p>
      </div>
    </form>
  </div>
</main>
{% endblock%}
```

APP.PY

```
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;SSLServerCertificate=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)
```

ABOUT PAGE:

```
{% extends 'base2.html'%} {% block head %}
<title>Profile</title>
{% endblock%} {%block body%}
<h2>Profile</h2>
<hr />
<div class="user-deatils">
  <h3>User Details</h3>
  <h4>USERNAME : {{data['USERNAME']}}</h4>
  <h4>FIRSTNAME : {{data['FIRSTNAME']}}</h4>
  <h4>LASTNAME : {{data['LASTNAME']}}</h4>
  <h4>EMAIL : {{data['EMAIL']}}</h3>
</div>
<hr>
<div class="forms-wrapper mg-20 " >
  <form action="{{url_for('updateUser')}}" method="post">
    <h3>Update user details</h3>
    <div class="field">
      <label for="input-field">Choose a field :</label>
      <select name="input-field" id="field">
        <option value="USERNAME">USERNAME</option>
        <option value="FIRSTNAME">FIRSTNAME</option>
        <option value="LASTNAME">LASTNAME</option>
      </select>
    </div>
    <div class="field">
      <label class="custom-label" for="input-value"> Enter Value</label>
      <input
        class="text-inputs"
        type="text"
        name="input-value"
        placeholder=" "
      />
    </div>
    <button class="submit-button">Update</button>
  </form>
  <form action="{{url_for('updatePassword')}}" method="post">
    <h3>Update Password</h3>
    <div class="field">
      <label class="custom-label" for="prev-password">
        Enter Old Password</label
      >
      <input
        class="text-inputs"
        type="password"
        name="prev-password"
        placeholder=" "
      />
    </div>
    <div class="field">
      <label class="custom-label" for="cur-password"> Enter New Password</label>
      <input
        class="text-inputs"
        type="password"
        name="cur-password"
        placeholder=" "
      />
    </div>
    <div class="field">
      <label class="custom-label" for="confirm-password">
        Enter Confirm Password</label
      >
      <input
        class="text-inputs"
        type="password"
        name="confirm-password"
        placeholder=" "
      />
    </div>

    <button class="submit-button">Update</button>
  </form>
</div>
{% endblock%}
```

BASE.HTML

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

<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <link        href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
    integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC
" crossorigin="anonymous" />
  <link rel="stylesheet" href="static/css/style.css" />
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
```

```
    integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
    crossorigin="anonymous"></script>

  <!-- <link rel="stylesheet" href="{{url_for('static', filename='css/style.css')}}"> -->
  {% block head%}{% endblock %}
</head>

<body>
  <div class="container mt-5">{% block body %} {% endblock %}</div>
</body>

</html>
```

REQUEST.HTML

```
{% extends 'base.html'%}

{% block head %}

<title>Login page</title>

{% endblock%}


{%block body%}


<main class="container ">

    <div class="mx-auto mt-5 border bg-light login-card " style="width:500px;">

        <h2 class='mx-4 mt-2'>LOGIN</h2>

        <form action="{{url_for('login') }}" method="post">

            <div class="mx-4 mt-2 text-danger">{{ msg }}</div>

            <div class="my-2 mx-4">

                <label for="username">username</label>

                <input type="text" class="form-control" placeholder="adc@gmail.com" name="username" required />
```

```
      </div>

      <div class="my-2 mx-4">

        <label for="password_1">password</label>

        <input type="password" class="form-control" name="password_1" required />

      </div>

      <input type="submit" value="submit" class="btn btn-primary my-4 mt-2 mx-4" />


    </form>

    <p>Don't have an account?<a href="{{ url_for('signup') }}"> Sign Up</a>

  </div>

</main>

</p>


</main>

{% endblock%}
```

INVENTORY MANAGEMENT.HTML

```
{% extends 'base2.html'%} {% block head %}
<title>Dashboard</title>
{% endblock%} {%block body%}
<h2>Dashboard</h2>
<p>
 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
 tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
</p>
{% include 'table.html' %}
<div class="forms-wrapper">
 <form action="{{url_for('UpdateStocks') }}" method="post">
  <h3>Update Stock</h3>
  <div class="field">
```

```
    <label class="custom-label" for="item"> Enter Item</label>
    <input class="text-inputs" type="text" name="item" placeholder="milk" />
  </div>
  <div class="field">
    <label for="input-field">Choose a field :</label>
    <select name="input-field" id="field">
      <option value="NAME">NAME</option>
      <option value="PRICE_PER_QUANTITY">PRICE_PER_QUANTITY</option>
      <option value="QUANTITY">QUANTITY</option>
    </select>
  </div>
  <div class="field">
    <label class="custom-label" for="input-value"> Enter Value</label>
    <input
      class="text-inputs"
      type="text"
      name="input-value"
      placeholder=" "
    />
  </div>
  <button class="submit-button">Update</button>
</form>

<form action="{{url_for('addStocks') }}" method="post">
  <h3>Add New Stock</h3>
  <div class="field">
    <label class="custom-label" for="item"> Enter the item</label>
    <input class="text-inputs" name="item" type="text" placeholder="juice" />
  </div>
  <div class="field">
    <label class="custom-label" for="quantity"> Enter quantity</label>
    <input
      class="text-inputs"
      type="number"
      name="quantity"
      placeholder="200"
    />
    </div>
    <div class="field">
      <label class="custom-label" for="price"> Enter price</label>
      <input class="text-inputs" type="number" name="price" placeholder="25" />
    </div>
    <button class="submit-button">Add Stock</button>
  </form>
  <form action="{{url_for('deleteStocks') }}" method="post">
    <h3>Remove stocks</h3>
    <div class="field">
      <label class="custom-label" for="item"> Enter the item</label>
      <input class="text-inputs" name="item" type="text" placeholder="juice" />
    </div>
    <button class="submit-button red-button">Remove</button>
  </form>
</div>

{% endblock%}
```

GITHUB LINK:

https://github.com/https://github.com/IBM-EPBL/IBM-Project-9134-1658982503

PROJECT DEMO LINK:

https://youtu.be/Mi6-JeChTtM