INVENTORY MANAGEMENT SYSTEM FOR RETAILERS

1. INTRODUCTION

1.1 Project Overview

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.

1.2 Purpose

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.

2. LITERATURE SURVEY

2.1 Existing problem

- Existing system contains many videos in the dataset which cause data storage issues.
- Use of raspberry Pi prevents reachability to people due to excessive cost.
- Sensors may become faulty over time which affects the reliability of the system.
- There are some models which can only predict alphabets with good accuracy.
- Some model fails on low light intensity and uncontrolled background.
- The hand key points estimation for more than two hands cannot be properly estimated.

2.2 References

[1] Ahmed, and I. Sultana, “A literature review on inventory modeling with reliability considerations,” International Journal of Industrial Engineering Computations, vol.5, pp.169-178, 2014.

[2] N. A. Anichebe, and A. O. Agu, “Effect of Inventory Management on Organizational Effectiveness,” Information and Knowledge Management, vol.3, No.8, 2013.

[3] B. Plossl, “Management,” New York, Prentice Hall Inc, 2005.

[4] S. Ziukov, “A literature review on models of inventory management under uncertainty,” Business systems and Economics, vol.5, No.1, 2015.

[5] G. J. Liu, R. Shah and R. G. Schroeder, "Managing demand and supply uncertainties to achieve mass customization ability," Journal of Manufacturing Technology Management, vol. 21, no. 8, pp. 990-1012, 2010.

[6] R. Pillai, “Inventory management performance in machine tool SMEs: What factors do influence them?” Journal of Industrial Engineering and Management, vol.3, No.3, pp.542-560, 2010.

[7] H. Ullah, and S. Parveen, “A Liturature Review on Inventory Lot Sizing Problems,” Global Journal of Research in Engineering, vol.10, Iss.5, (ver.1.0), 2010.

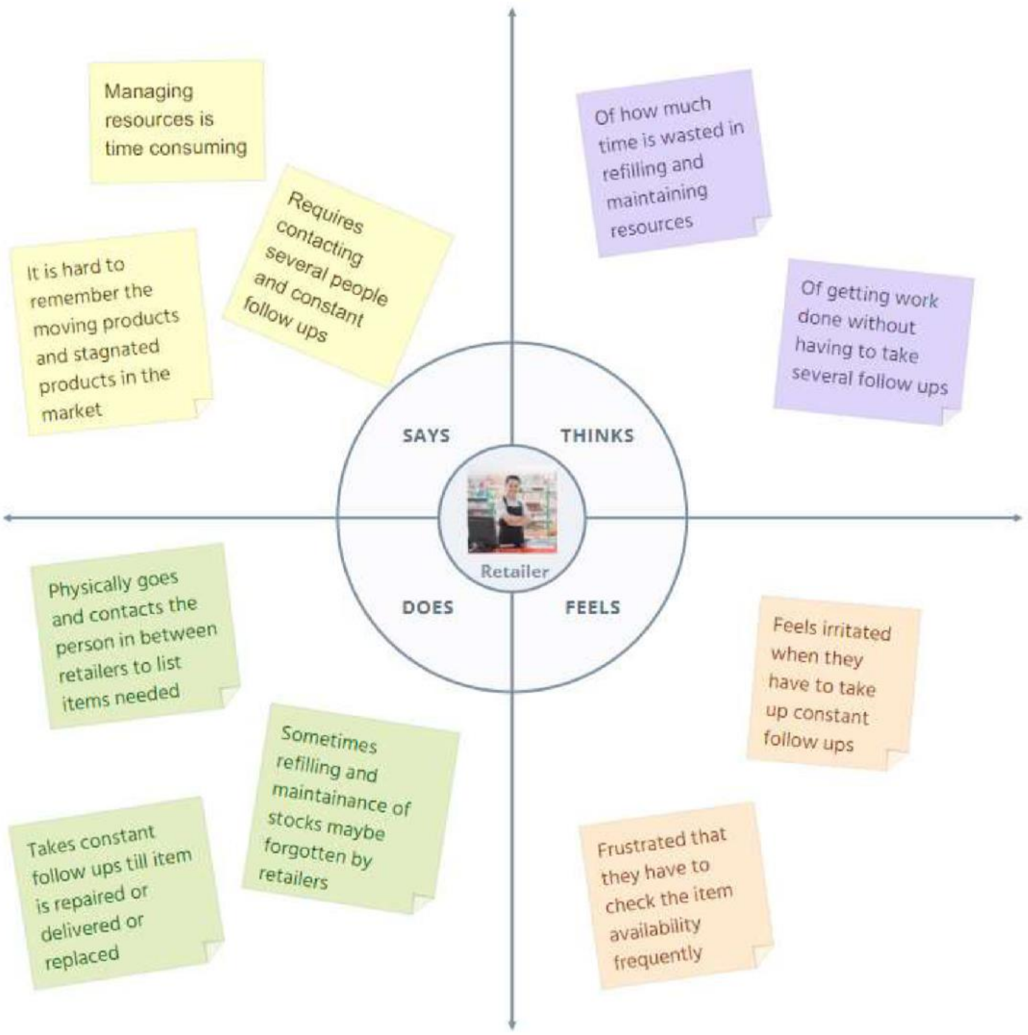
2.3 Problem Statement Definition

Retailers do not have any automated system to record orders and keep track of their inventory data. It is difficult for the retailers to constantly monitor stocks of all items and place orders accordingly every time they are running low because they only keep records in logbooks that are not properly organized.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

Empathy Map - Inventory Management System For Retailers



Managing resources is time consuming

Requires contacting several people and constant follow ups

It is hard to remember the moving products and stagnated products in the market

Of how much time is wasted in refilling and maintaining resources

Of getting work done without having to take several follow ups

Physically goes and contacts the person in between retailers to list items needed

Sometimes refilling and maintainance of stocks maybe forgotten by retailers

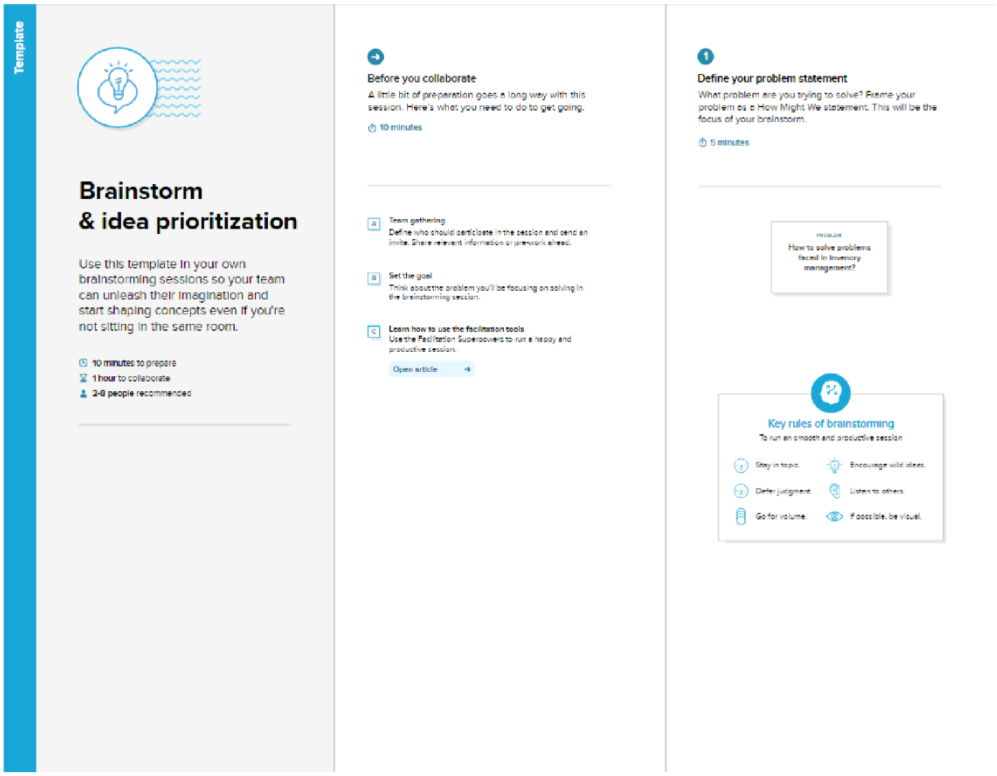
Takes constant follow ups till item is repaired or delivered or replaced

Feels irritated when they have to take up constant follow ups

Frustrated that they have to check the item availability frequently

3.2 Ideation & Brainstorming

Step 1



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

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 relevent 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.

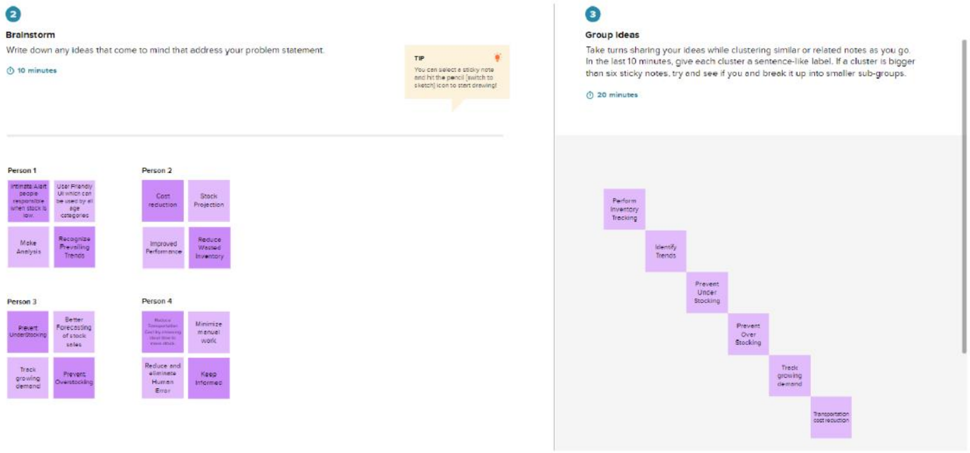
5 minutes

PROBLEM
How to solve problems faced in Invenory management?

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.

Step 2



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!

Person 1
- Intimate Alert people responsible when stock is low.
- User Friendly UI which can be used by all age categories
- Make Analysis
- Recognize Prevailing Trends

Person 2
- Cost reduction
- Stock Projection
- Improved Performance
- Reduce Wasted Inventory

Person 3
- Prevent UnderStocking
- Better Forecasting of stock sales
- Track growing demand
- Prevent Overstocking

Person 4
- Reduce Transportation Cost by choosing ideal time to move stock
- Minimize manuel work
- Reduce and eliminate Human Error
- Keep Informed

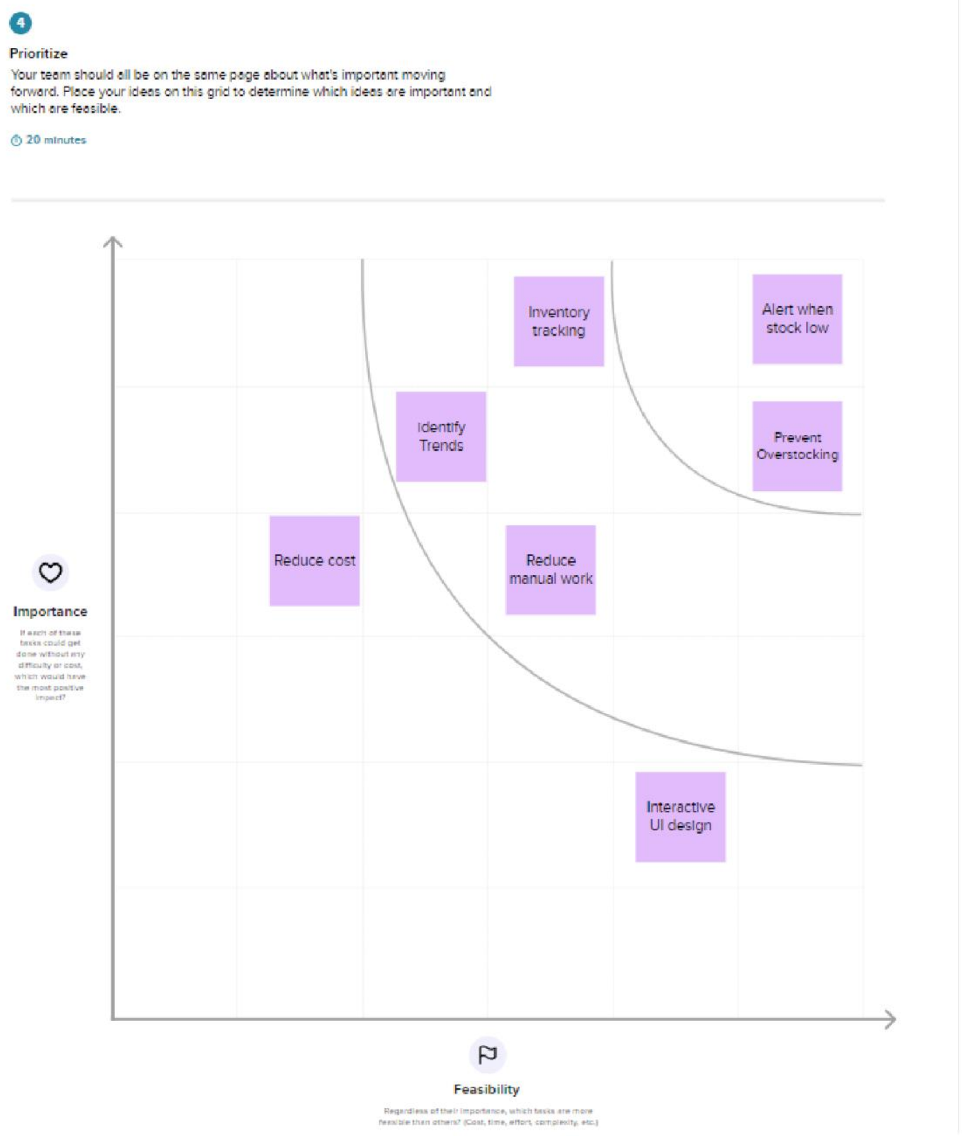
3 Group Ideas

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

- Perform Inventory Tracking
- Identify Trends
- Prevent Under Stocking
- Prevent Over Stocking
- Track growing demand
- Transportation cost reduction

Step 3



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

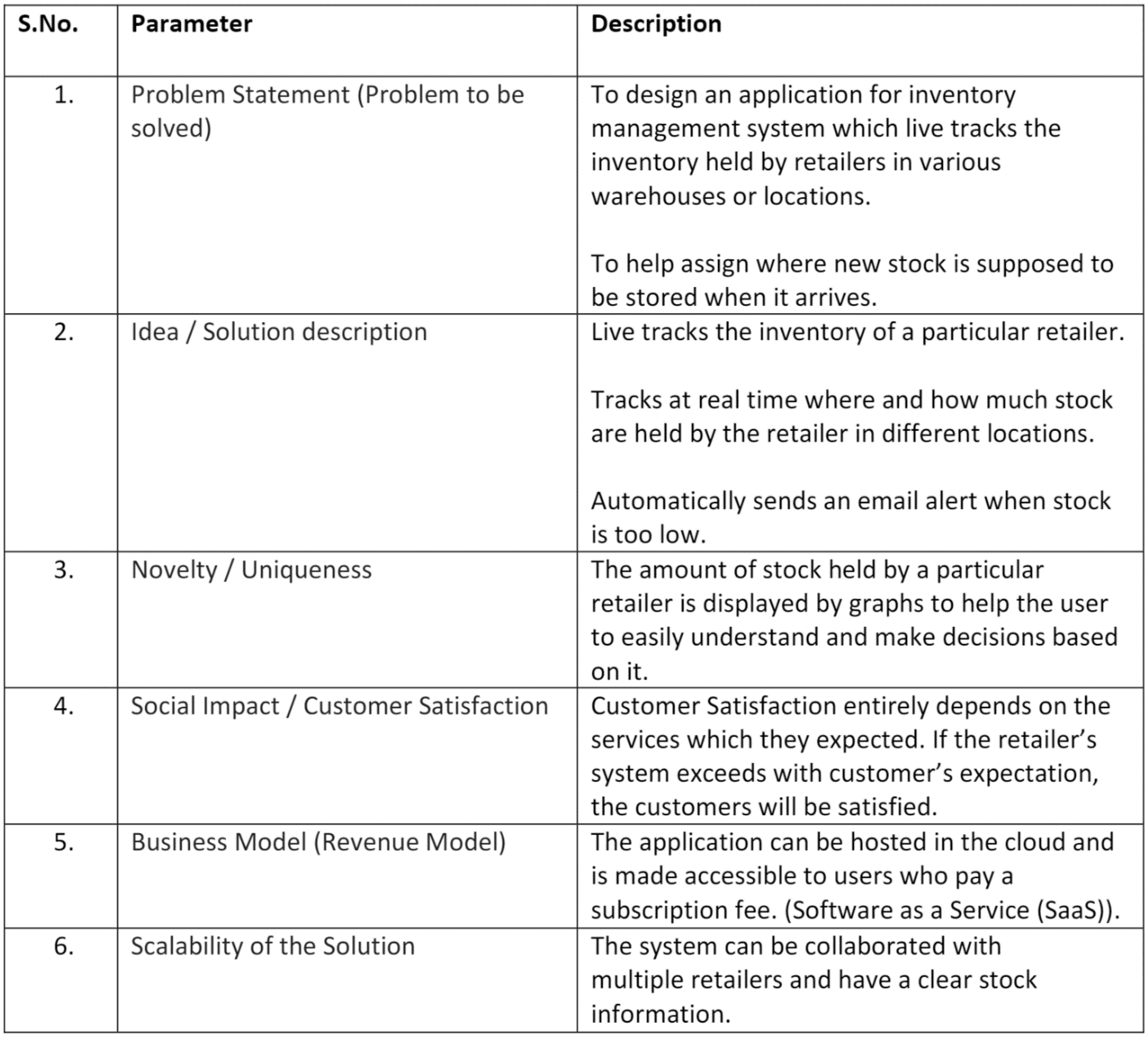
Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?

Feasibility

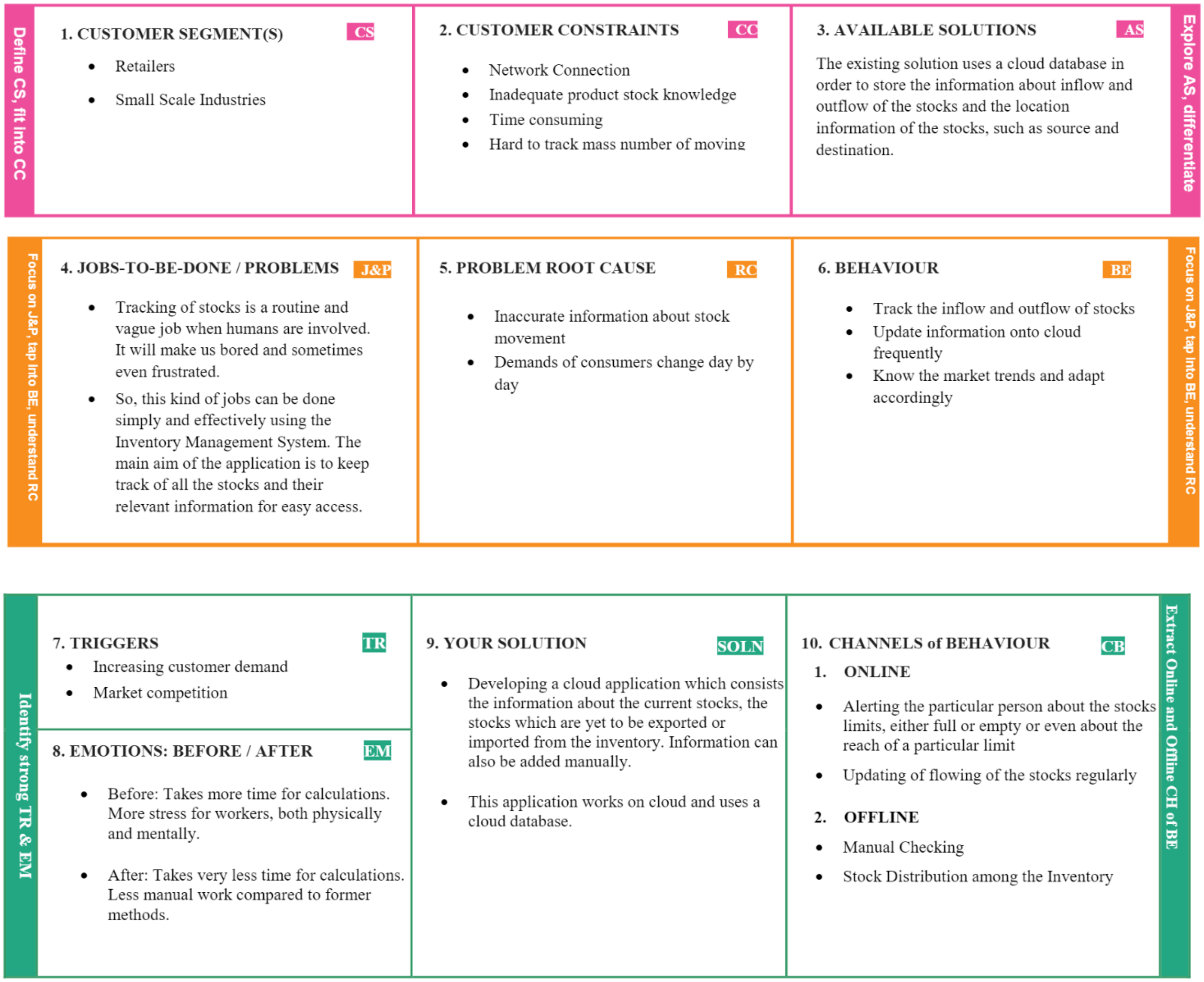
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 Proposed Solution



S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To design an application for inventory management system which live tracks the inventory held by retailers in various warehouses or locations. To help assign where new stock is supposed to be stored when it arrives.
2.	Idea / Solution description	Live tracks the inventory of a particular retailer. Tracks at real time where and how much stock are held by the retailer in different locations. Automatically sends an email alert when stock is too low.
3.	Novelty / Uniqueness	The amount of stock held by a particular retailer is displayed by graphs to help the user to easily understand and make decisions based on it.
4.	Social Impact / Customer Satisfaction	Customer Satisfaction entirely depends on the services which they expected. If the retailer's system exceeds with customer's expectation, the customers will be satisfied.
5.	Business Model (Revenue Model)	The application can be hosted in the cloud and is made accessible to users who pay a subscription fee. (Software as a Service (SaaS)).
6.	Scalability of the Solution	The system can be collaborated with multiple retailers and have a clear stock information.

3.4 Problem Solution fit



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- Retailers
- Small Scale Industries

2. CUSTOMER CONSTRAINTS CC

- Network Connection
- Inadequate product stock knowledge
- Time consuming
- Hard to track mass number of moving

3. AVAILABLE SOLUTIONS AS

The existing solution uses a cloud database in order to store the information about inflow and outflow of the stocks and the location information of the stocks, such as source and destination.

Explore AS, differentiate

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

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

- Tracking of stocks is a routine and vague job when humans are involved. It will make us bored and sometimes even frustrated.
- So, this kind of jobs can be done simply and effectively using the Inventory Management System. The main aim of the application is to keep track of all the stocks and their relevant information for easy access.

5. PROBLEM ROOT CAUSE RC

- Inaccurate information about stock movement
- Demands of consumers change day by day

6. BEHAVIOUR BE

- Track the inflow and outflow of stocks
- Update information onto cloud frequently
- Know the market trends and adapt accordingly

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

Identify strong TR & EM

7. TRIGGERS TR

- Increasing customer demand
- Market competition

8. EMOTIONS: BEFORE / AFTER EM

- Before: Takes more time for calculations. More stress for workers, both physically and mentally.
- After: Takes very less time for calculations. Less manual work compared to former methods.

9. YOUR SOLUTION SOLN

- Developing a cloud application which consists the information about the current stocks, the stocks which are yet to be exported or imported from the inventory. Information can also be added manually.
- This application works on cloud and uses a cloud database.

10. CHANNELS of BEHAVIOUR CB

1. **ONLINE**

- Alerting the particular person about the stocks limits, either full or empty or even about the reach of a particular limit
- Updating of flowing of the stocks regularly

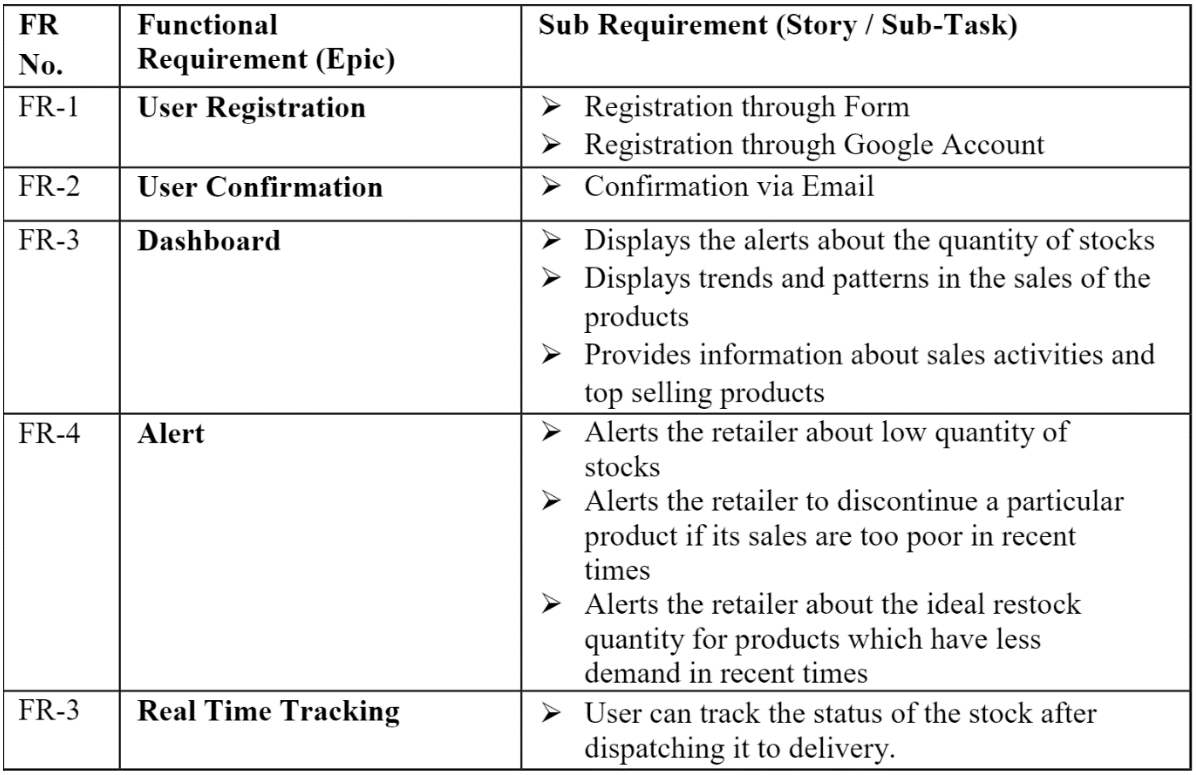
2. **OFFLINE**

- Manual Checking
- Stock Distribution among the Inventory

Extract Online and Offline CH of BE

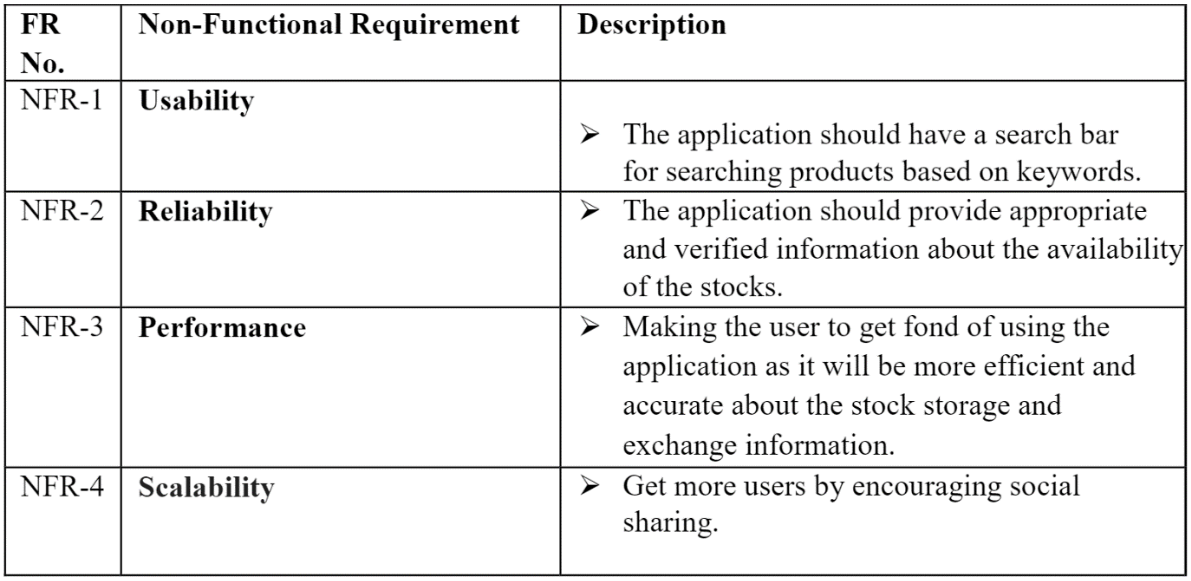
4. REQUIREMENT ANALYSIS

4.1 Functional requirement



FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Registration**	➢ Registration through Form ➢ Registration through Google Account
FR-2	**User Confirmation**	➢ Confirmation via Email
FR-3	**Dashboard**	➢ Displays the alerts about the quantity of stocks ➢ Displays trends and patterns in the sales of the products ➢ Provides information about sales activities and top selling products
FR-4	**Alert**	➢ Alerts the retailer about low quantity of stocks ➢ Alerts the retailer to discontinue a particular product if its sales are too poor in recent times ➢ Alerts the retailer about the ideal restock quantity for products which have less demand in recent times
FR-3	**Real Time Tracking**	➢ User can track the status of the stock after dispatching it to delivery.

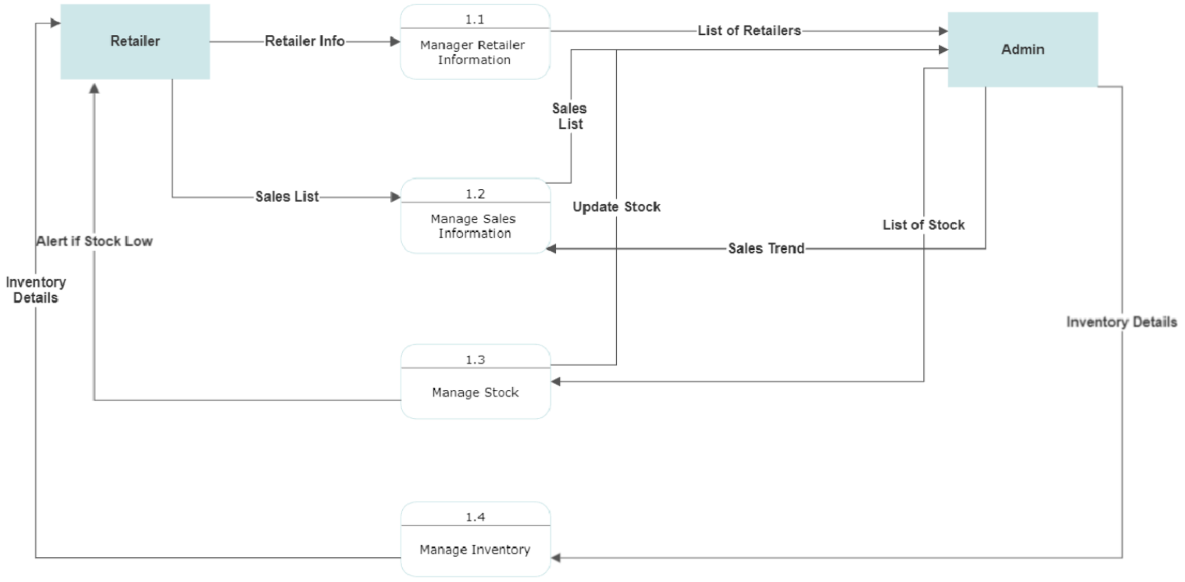
4.2 Non-Functional requirements



FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	➢ The application should have a search bar for searching products based on keywords.
NFR-2	**Reliability**	➢ The application should provide appropriate and verified information about the availability of the stocks.
NFR-3	**Performance**	➢ Making the user to get fond of using the application as it will be more efficient and accurate about the stock storage and exchange information.
NFR-4	**Scalability**	➢ Get more users by encouraging social sharing.

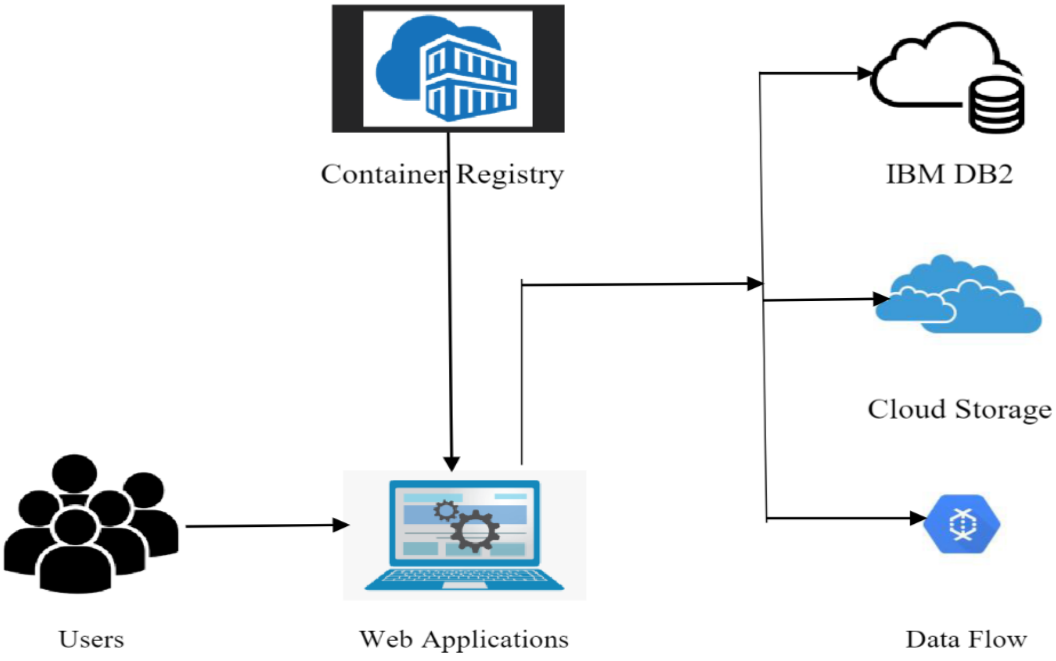
5. PROJECT DESIGN

5.1 Data Flow Diagrams

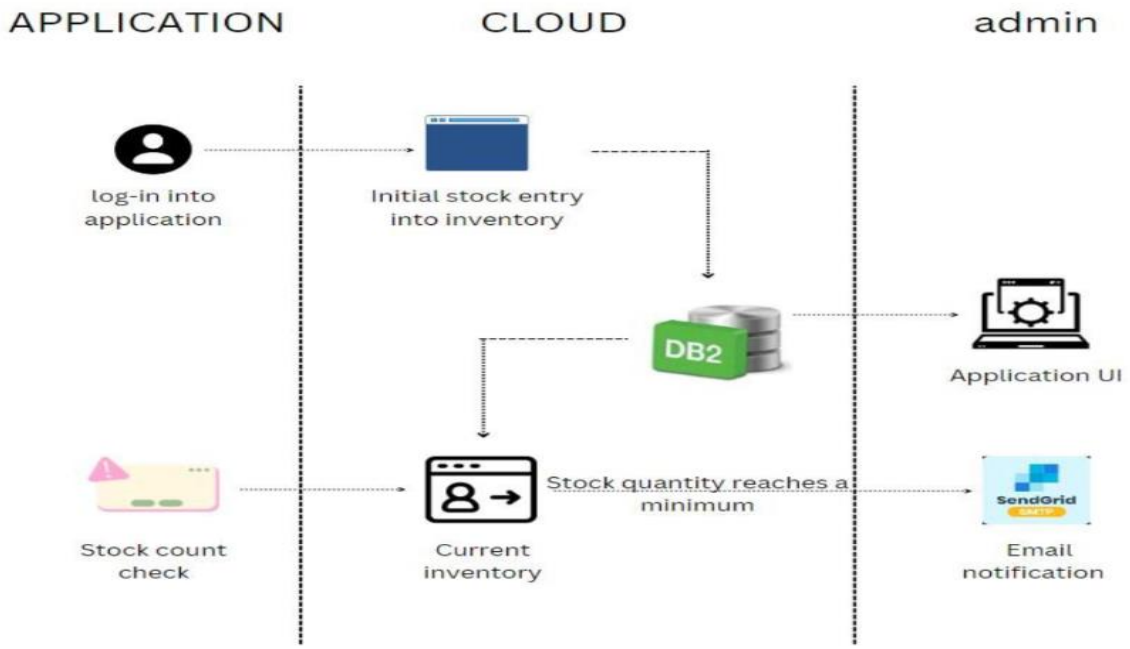


5.2 Solution & Technical Architecture

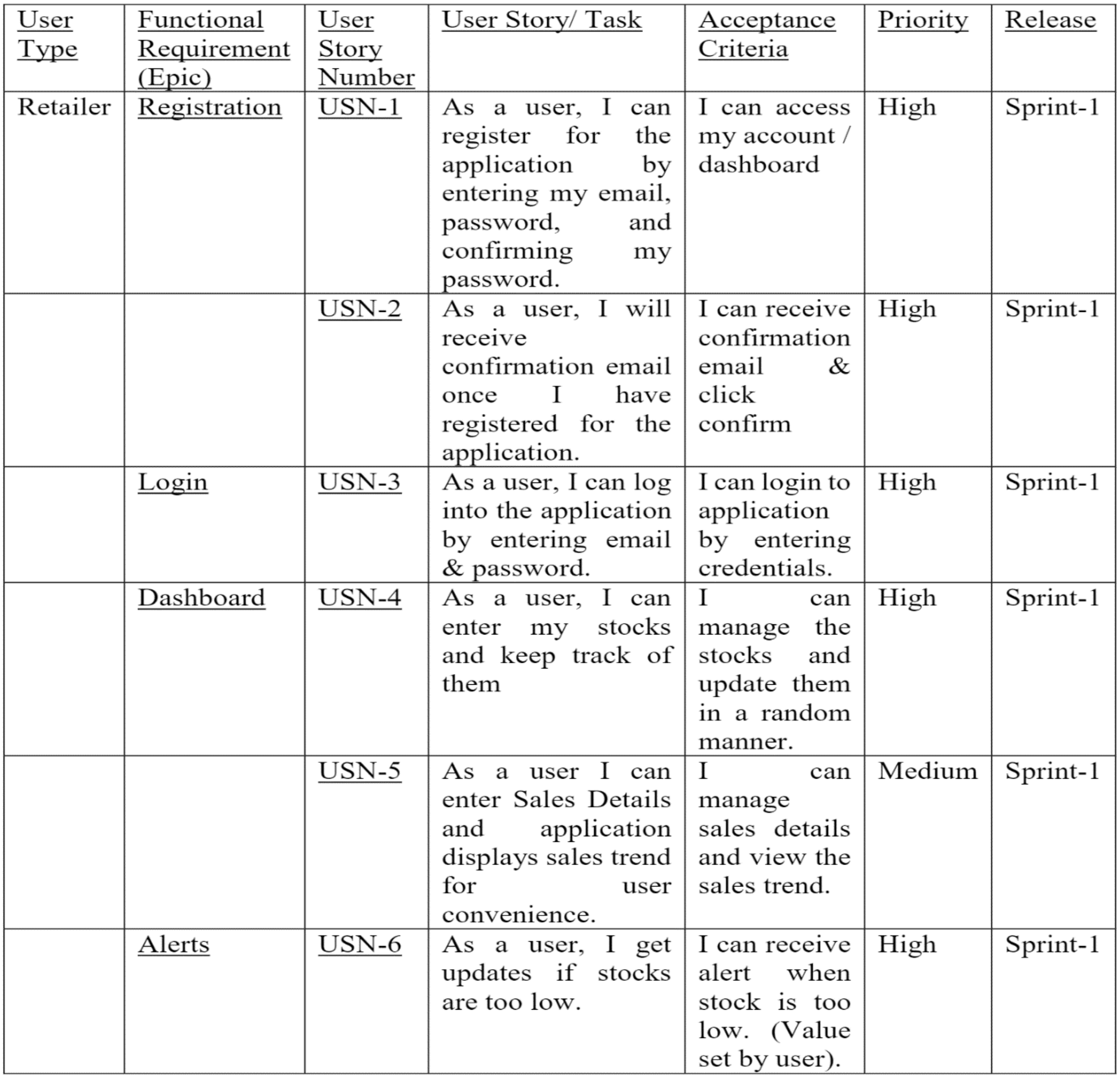
5.2.1 Solution Architecture



5.2.2 Technology Architecture



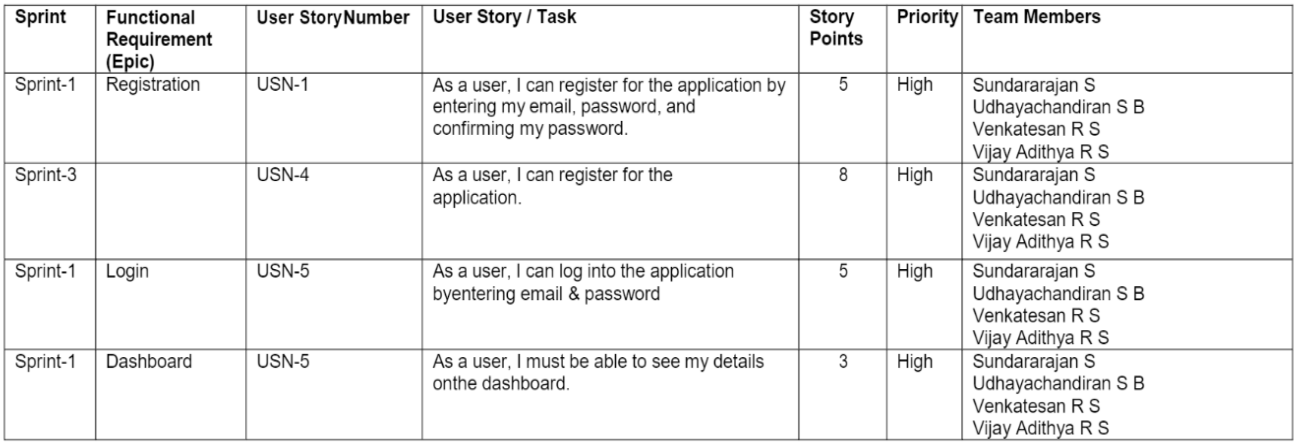
5.3 User Stories

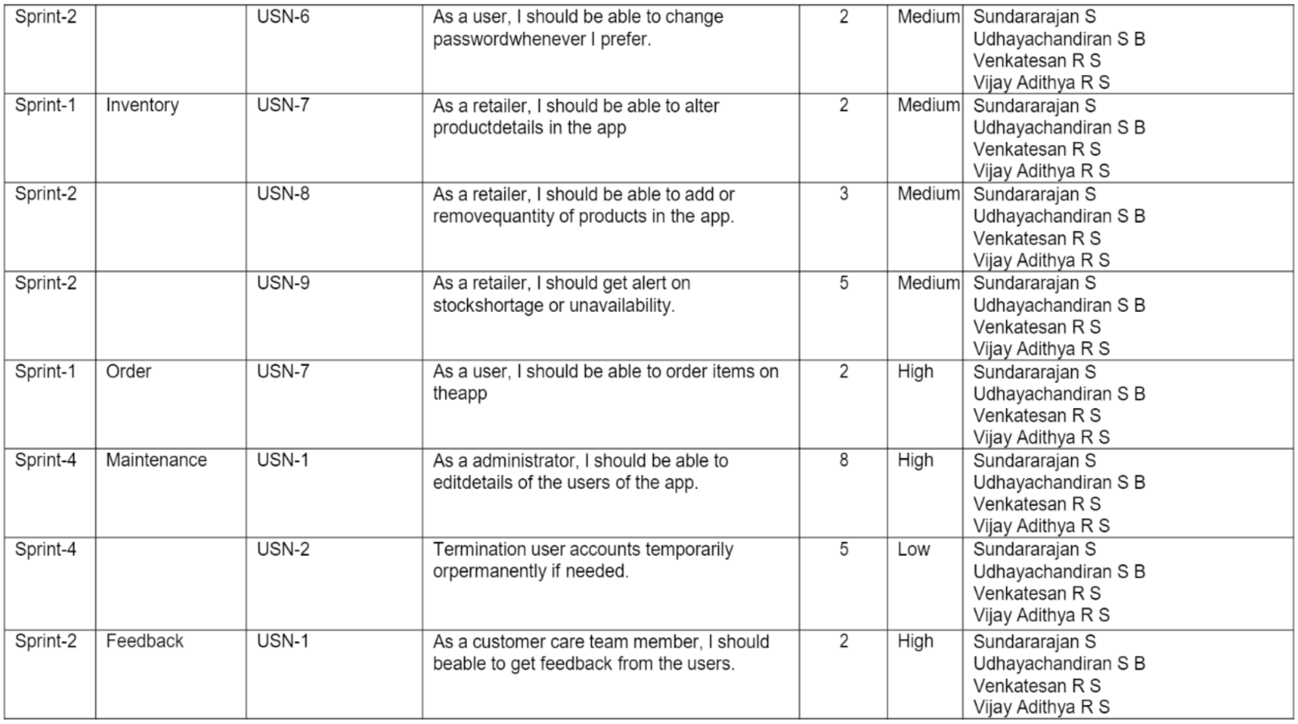


User Type	Functional Requirement (Epic)	User Story Number	User Story/ Task	Acceptance Criteria	Priority	Release
Retailer	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application.	I can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-3	As a user, I can log into the application by entering email & password.	I can login to application by entering credentials.	High	Sprint-1
	Dashboard	USN-4	As a user, I can enter my stocks and keep track of them	I can manage the stocks and update them in a random manner.	High	Sprint-1
		USN-5	As a user I can enter Sales Details and application displays sales trend for user convenience.	I can manage sales details and view the sales trend.	Medium	Sprint-1
	Alerts	USN-6	As a user, I get updates if stocks are too low.	I can receive alert when stock is too low. (Value set by user).	High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation



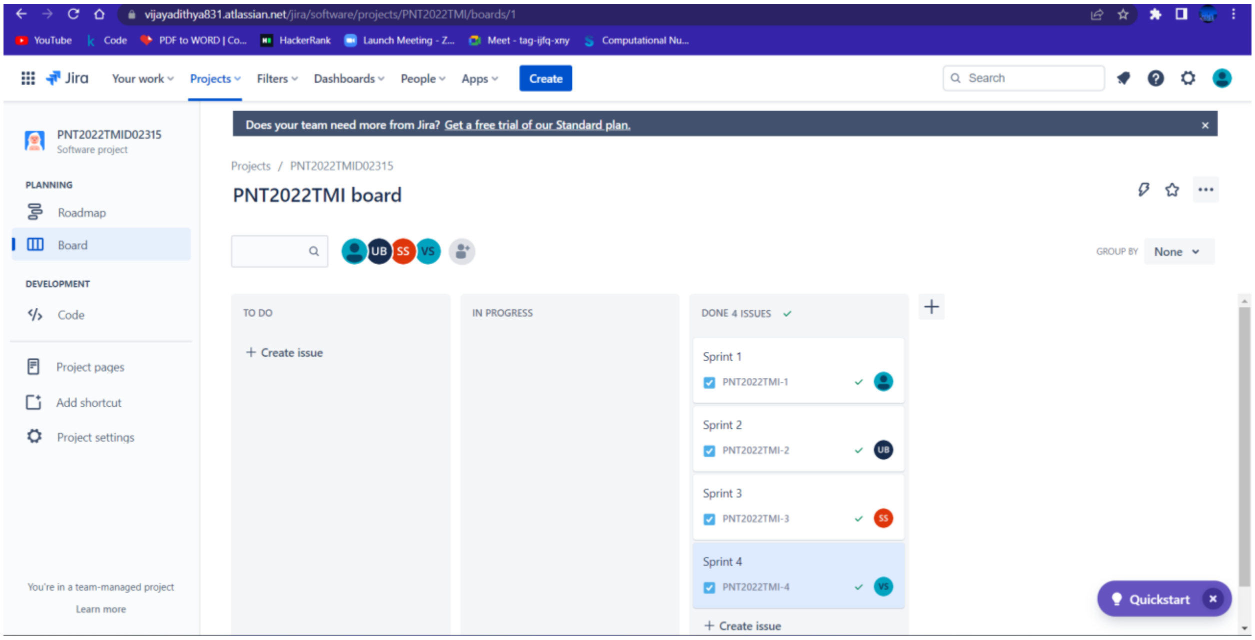


Sprint	Functional Requirement (Epic)	User StoryNumber	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	5	High	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-3		USN-4	As a user, I can register for the application.	8	High	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-1	Login	USN-5	As a user, I can log into the application byentering email & password	5	High	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-1	Dashboard	USN-5	As a user, I must be able to see my details onthe dashboard.	3	High	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-2		USN-6	As a user, I should be able to change passwordwhenever I prefer.	2	Medium	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-1	Inventory	USN-7	As a retailer, I should be able to alter productdetails in the app	2	Medium	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-2		USN-8	As a retailer, I should be able to add or removequantity of products in the app.	3	Medium	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-2		USN-9	As a retailer, I should get alert on stockshortage or unavailability.	5	Medium	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-1	Order	USN-7	As a user, I should be able to order items on theapp	2	High	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-4	Maintenance	USN-1	As a administrator, I should be able to editdetails of the users of the app.	8	High	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-4		USN-2	Termination user accounts temporarily orpermanently if needed.	5	Low	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S
Sprint-2	Feedback	USN-1	As a customer care team member, I should beable to get feedback from the users.	2	High	Sundararajan S Udhayachandiran S B Venkatesan R S Vijay Adithya R S

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	5 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA



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

Projects / PNT2022TMID02315

PNT2022TMI board

TO DO

+ Create issue

IN PROGRESS

DONE 4 ISSUES ✓

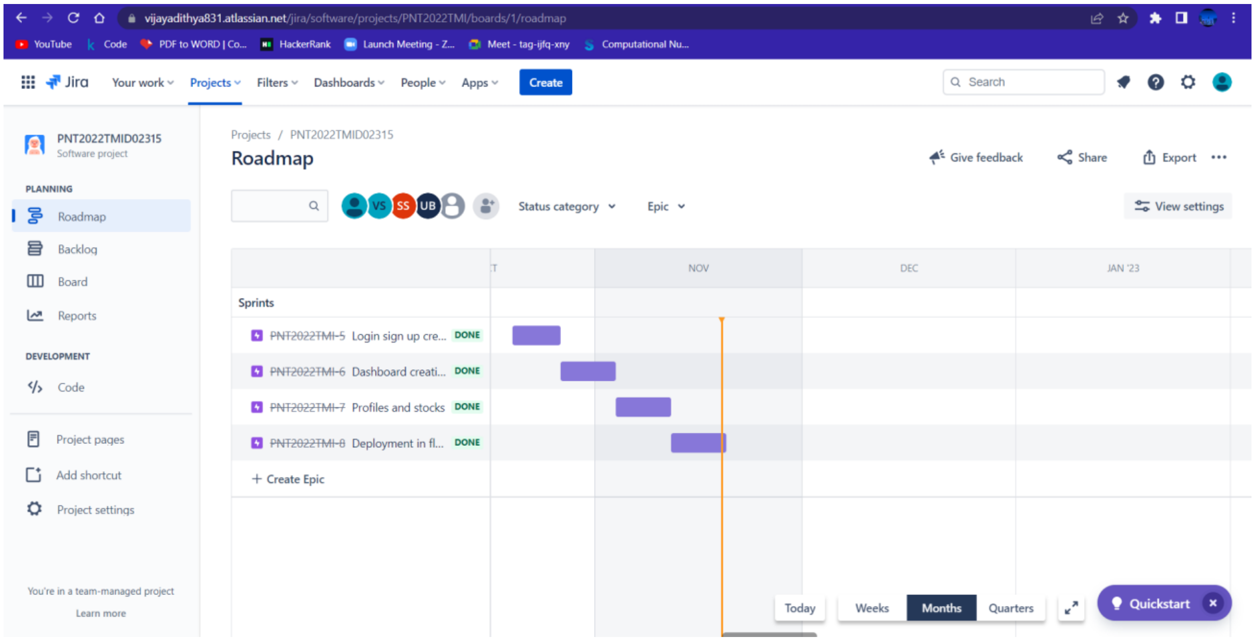
Sprint 1 ☑ PNT2022TMI-1 ✓

Sprint 2 ☑ PNT2022TMI-2 ✓ UB

Sprint 3 ☑ PNT2022TMI-3 ✓ SS

Sprint 4 ☑ PNT2022TMI-4 ✓ VS

+ Create issue



Projects / PNT2022TMID02315

Roadmap

Sprints

~~PNT2022TMI-5~~ Login sign up cre... DONE

~~PNT2022TMI-6~~ Dashboard creati... DONE

~~PNT2022TMI-7~~ Profiles and stocks DONE

~~PNT2022TMI-8~~ Deployment in fl... DONE

+ Create Epic

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

7.1 Dashboard

```
{% extends "layout.html" %}

{% block content %}

  <div class="wrapper">

   <!-- Sidebar  -->

   {% include 'sidebar.html' %}

   <!-- Page Content  -->

   <div id="content">

    <main>

     <h1>Dashboard</h1>

     <div class="date">

       <input type="date">

     </div>

     <div class="insights">

       <div class="sales">

         <span class="material-icons-sharp">analytics</span>

         <div class="middle">

           <div class="left">

             <h3>Total Sales</h3>
```

```
            <h1>Rs.25000</h1>

        </div>

        <div class="progress">

            <svg>

                <circle cx='38' cy='38' r='36'></circle>

            </svg>

            <div class="number">

                <p> 81%</p>

            </div>

        </div>

    </div>

    <small class="text-muted">Last 24 Hours</small>

</div>

<!--------------------------------END OF SALES---------------------------------->

<div class="expenses">

    <span class="material-icons-sharp">bar_chart</span>

    <div class="middle">

        <div class="left">

            <h3>Total Expenses</h3>

            <h1>Rs.14000</h1>
        </div>

        <div class="progress">

            <svg>

                <circle cx='38' cy='38' r='36'></circle>

            </svg>

            <div class="number">

                <p> 62%</p>

            </div>

        </div>

    </div>

    <small class="text-muted">Last 24 Hours</small>

</div>

<!-------------------------------END OF EXPENSES--------------------------------->


<div class="income">

    <span class="material-icons-sharp">stacked_line_chart</span>

    <div class="middle">

        <div class="left">

            <h3>Total Income</h3>

            <h1>Rs.10000</h1>
```

```
                      </div>
                      <div class="progress">
                          <svg>
                              <circle cx='38' cy='38' r='36'></circle>
                          </svg>
                          <div class="number">
                              <p> 44%</p>
                          </div>
                      </div>
                  </div>
                  <small class="text-muted">Last 24 Hours</small>
              </div>
              <!-------------------------------END OF INCOME--------------------------------->
          </div>
      </main>
      {% include 'table.html' %}
      </div>
    </div>
{% endblock content %}
```

7.2 Edit Stock

```
{% extends "layout.html" %}
{% block content %}
<div class="wrapper">
    <!-- Sidebar  -->
    {% include 'sidebar.html' %}
    <!-- Page Content  -->
    <div id="content">
        <h2>Inventory</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="" method="post">
                <h3>Create Item</h3>
                <div class="field">
                    <label class="custom-label" for="item"
                        >Enter Stock ID</label>
                    <input
                        class="text-inputs"
                        type="text"
                        name="stock_id"
                        placeholder="1"
                    />
                </div>
                <div class="field">
                    <label class="custom-label" for="item"
                        >Enter Quantity</label
                    >
                    <input
                        class="text-inputs"
                        type="number"
                        name="quantity"
                        placeholder="10"
                    />
                </div>
                <button class="submit-button">Create</button>
            </form>
```

```
<form action="" method="post">
  <h3>Update Order</h3>
  <div class="field">
    <label class="custom-label" for="item"
      >Enter Order ID</label
    >
    <input
      class="text-inputs"
      name="item"
      type="number"
      placeholder="1"
    />
  </div>
  <div class="field">
    <label for="custom-label" for="input-field"
      >Choose a field -
    </label>
    <select name="input-field" id="field">
      <option value="STOCKS_ID">STOCKS_ID</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" />
  </div>
  <button class="submit-button">Update</button>
</form>
<form action="" method="post">
  <h3>Cancel Order</h3>
  <div class="field">
    <label class="custom-label" for="item"
      >Enter Order ID</label
    >
    <input
      class="text-inputs"
      name="order_id"
```

```
            type="number"
            placeholder="1"
          />
        </div>
        <button class="submit-button red-button">Cancel</button>
      </form>
    </div>
  </div>
</div>
{% endblock content %}
```

8. TESTING

8.1 Test Cases

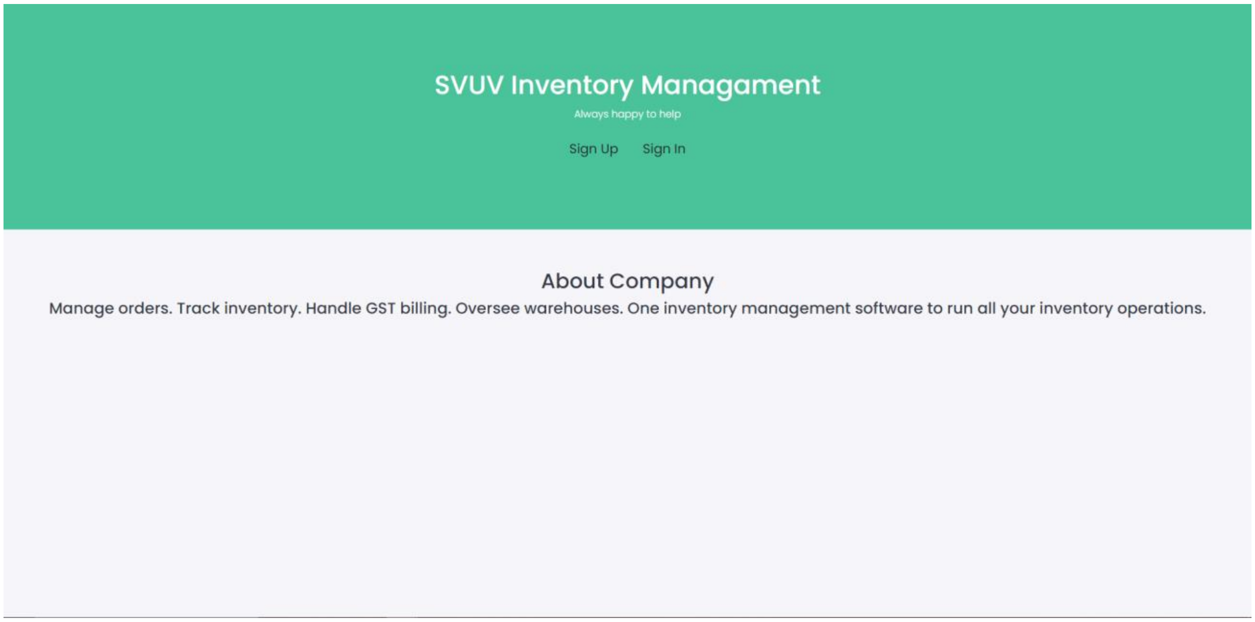
Test Case ID	Feature Type	Component	Test Scenario	Steps to Execute	Result	Status
HomePage_TC_001	Functional	Home page	Verify user is able to see the Login/Signup popup when user clicked on Login Button in the Homepage	1.Enter URL and click go 2.Click on Sign in Button 3.Verify Sign in/Sign up popup displayed or not	Login page should pop up as soon as the Login button is clicked.	Pass
LoginPage_TC_002	UI	Login Page	Verify the UI elements in Login/Signup popup	1. Enter URL and click GO 2. Click on Sign up button	Application should display the following: a. Name b. Email	Pass

				3. Verify Sign up page displays the following: a. Name b. Email c. Password d. Conform password e. Register button	c. Password d. Conform password e. Sign in/Sign up button	
LoginPage_ TC_003	Functional	Login Page	Verify user is able to log into application with Valid credentials	1. Enter URL and click GO 2. Click on Sign in link 3. Enter valid email and password 4. Click on Sign in button	The user should navigate to user account dashboard page.	Pass
LoginPage_ TC_004	Functional	Dashboard Page	Verify the user is able to view the dashboard and see the charts	1. Enter URL (dashboard.html) and click GO 2. View the stocks in tabular as well as graphical form 3. The table should consist of a list containing item_id, item, quantity_left, price_per_quantity, etc.	The application should display the quantities of stocks present currently in the inventory.	Pass

9. RESULTS

9.1 Performance Metrics

Home Page



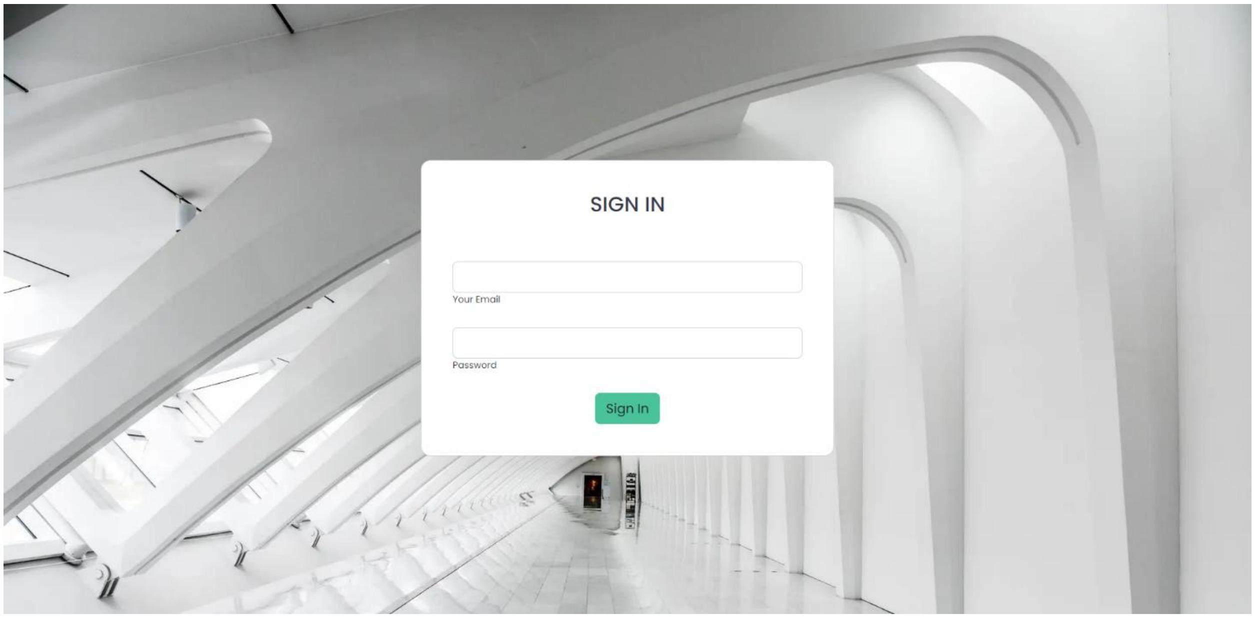
SVUV Inventory Managament

Always happy to help

About Company

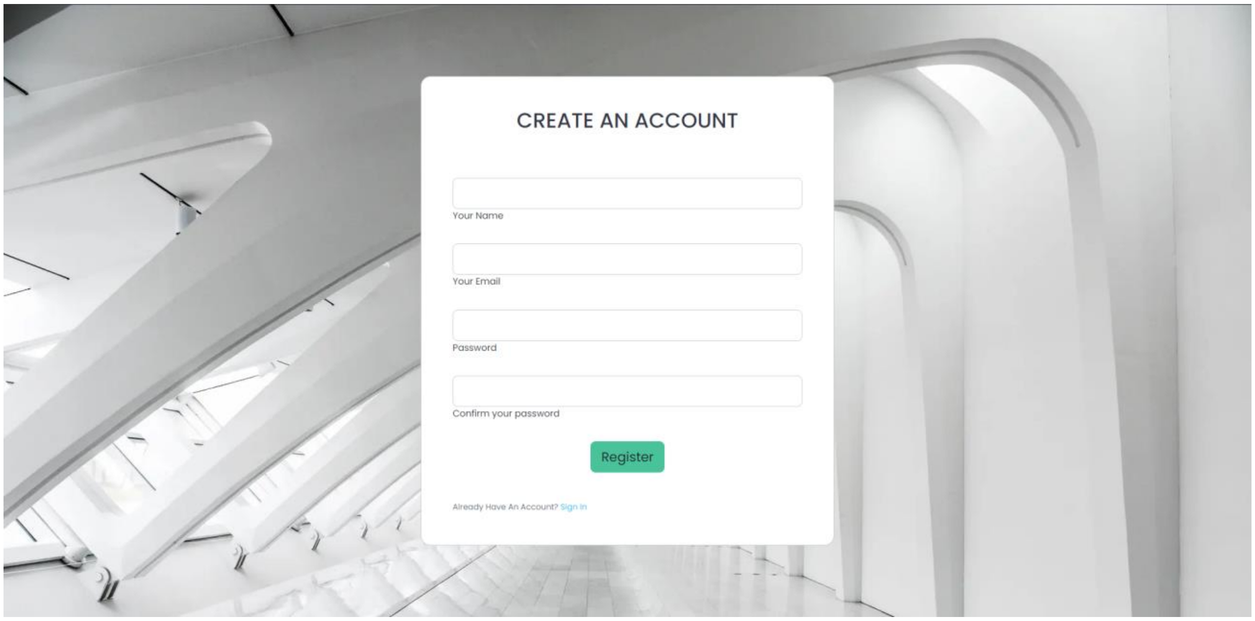
Manage orders. Track inventory. Handle GST billing. Oversee warehouses. One inventory management software to run all your inventory operations.

Sign In Page



SIGN IN

Sign Up Page



CREATE AN ACCOUNT

Your Name

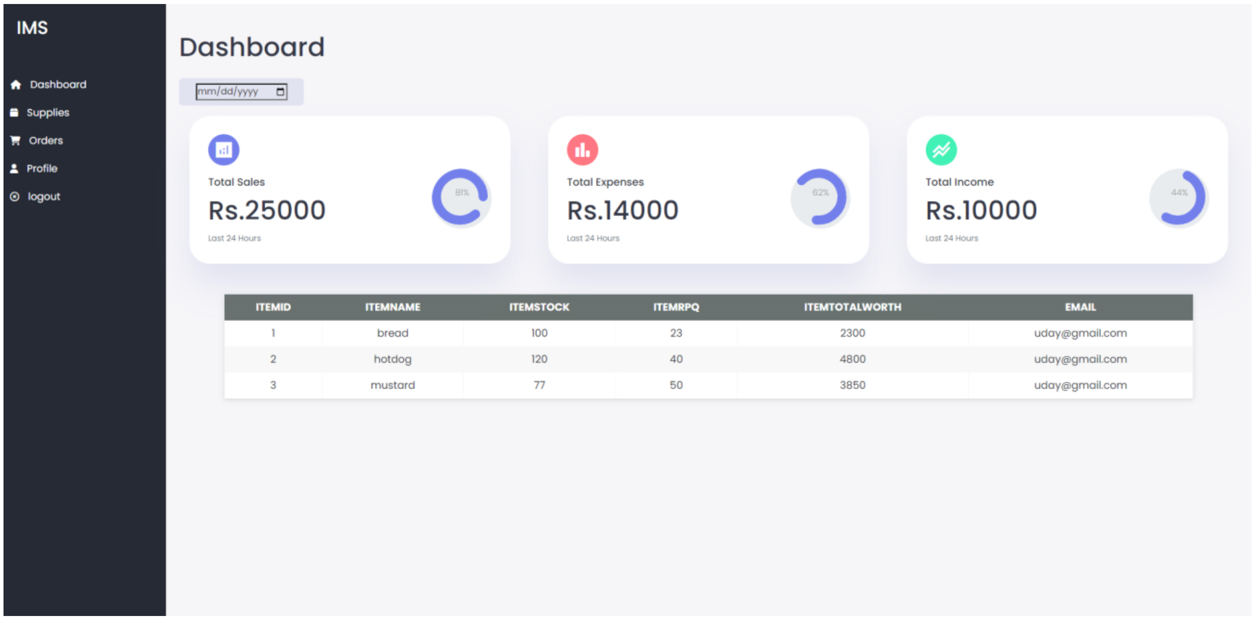
Your Email

Password

Confirm your password

Already Have An Account? Sign In

Dashboard



Dashboard

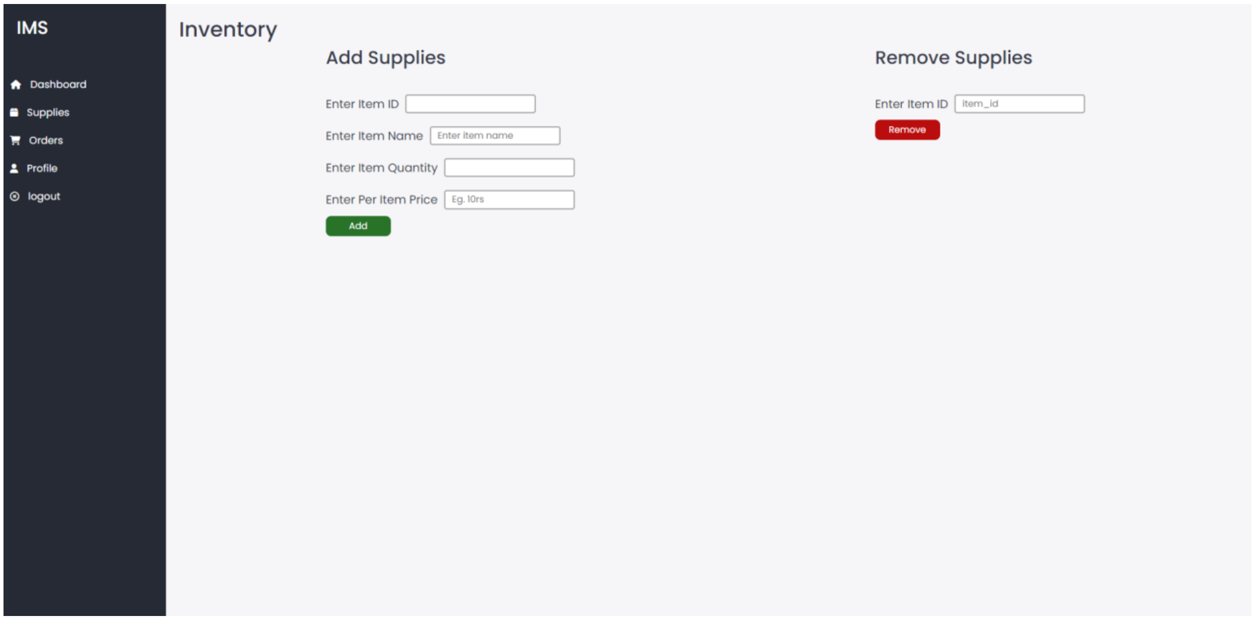
Total Sales
Rs.25000
Last 24 Hours

Total Expenses
Rs.14000
Last 24 Hours

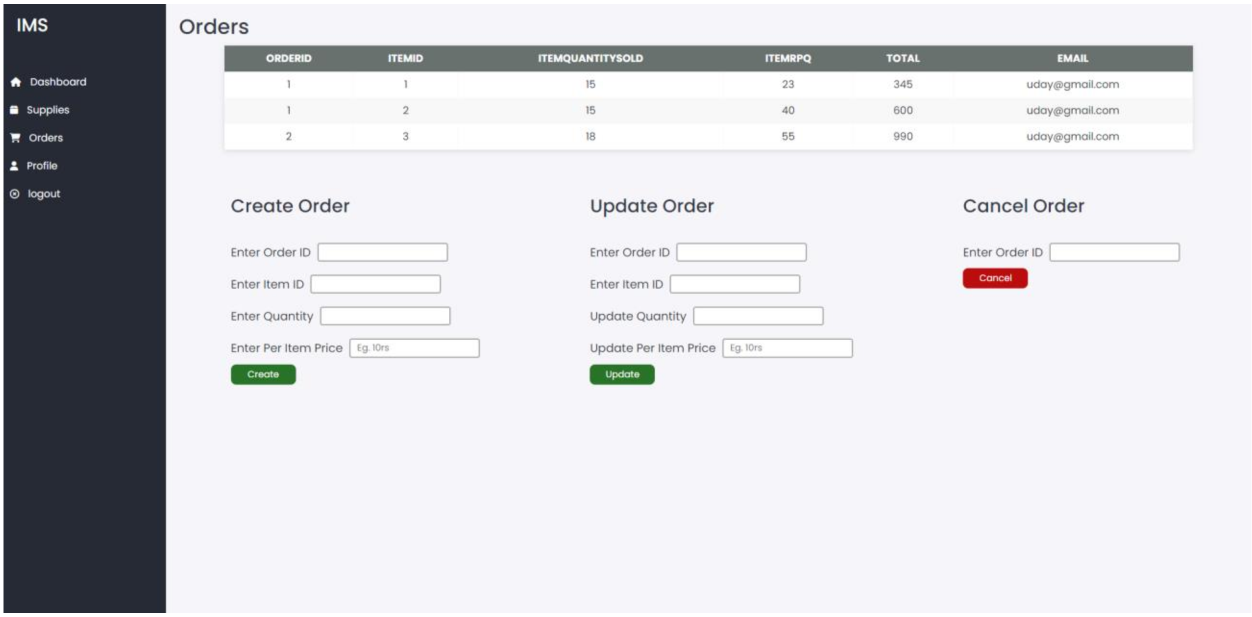
Total Income
Rs.10000
Last 24 Hours

ITEMID	ITEMNAME	ITEMSTOCK	ITEMRPQ	ITEMTOTALWORTH	EMAIL
1	bread	100	23	2300	uday@gmail.com
2	hotdog	120	40	4800	uday@gmail.com
3	mustard	77	50	3850	uday@gmail.com

Supplies

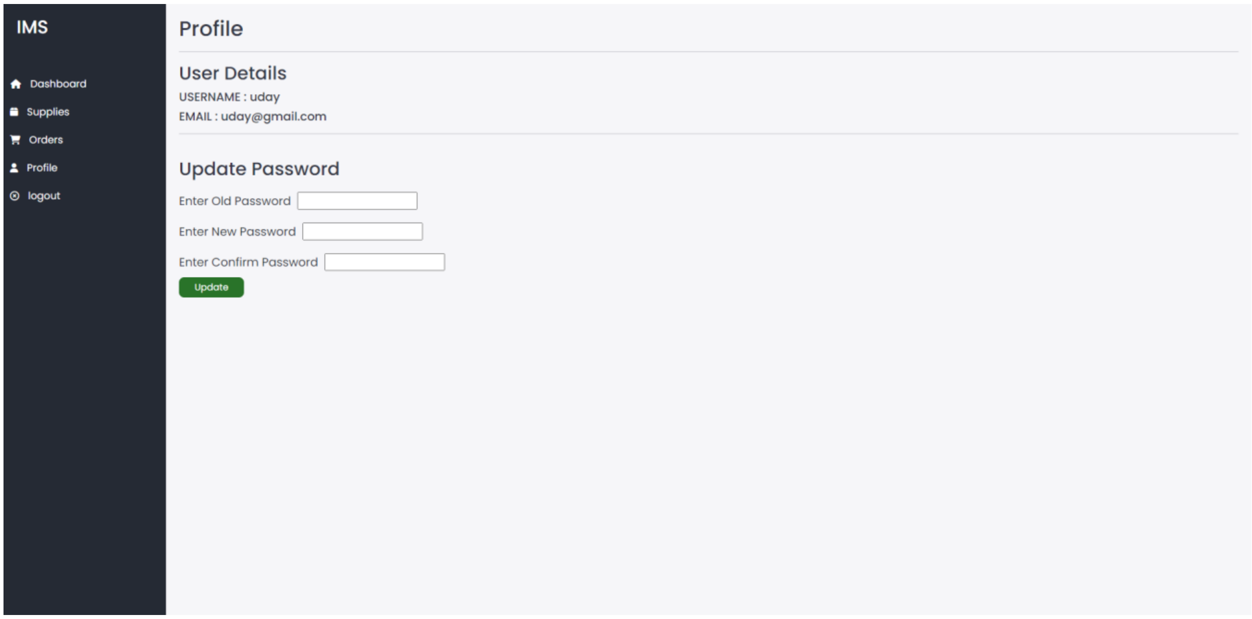


Orders



ORDERID	ITEMID	ITEMQUANTITYSOLD	ITEMRPQ	TOTAL	EMAIL
1	1	15	23	345	uday@gmail.com
1	2	15	40	600	uday@gmail.com
2	3	18	55	990	uday@gmail.com

Profile



USERNAME : uday
EMAIL : uday@gmail.com

10. ADVANTAGES & DISADVANTAGES

The major advantage in using the Inventory Management System for Retailers application is saving the time and human errors in recording the information about the stocks present in a particular inventory/warehouse by minimizing manual work. The application also consists of graphs which depict the expenses spent by the retailers. Since a database is available to manage stocks received and sold to customer, retailers can keep track of what is currently available in inventory.

The disadvantage in the proposed system is that even though this project proves to reduce the amount of manual labor and reduce human errors, major sections of the system are still dependent on an admin or a person on the retailer side feeding details constantly in a database which can prove to be a meticulous task. Also, the main data regarding the inventory is only available in tabular form, which arises a demand for pictorial representation.

11. CONCLUSION

To conclude, Inventory Management System is a simple desktop based application basically suitable for small organization. It has every basic items which are used for the small organization. Our team is successful in making the application where we can update, insert and delete the item as per the requirement. This application also provides an alert when the quantity of the stocks falls below the threshold value, so that the retailer will be taking necessary steps to refill the particular stock.

Through it has some limitations, our team strongly believes that the implementation of this system will surely benefit the organization.

12. FUTURE SCOPE

A system can be developed in such a way that it can completely eliminate the necessity of an admin to update values in the database. This can be done using technology like RFID tags to automatically update stock quantities in the database whenever the stock enters and leaves the inventory.

A certain analysis part can be added to this system such as a module to identify the sales trend of a particular product and notify the retailer to maybe avoid buying a product if it does not sell good in the market.

13. APPENDIX

SOURCE CODE

Landing Page

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

  <style>

  .jumbotron {

    background-color: rgba(74,194,154,255);

    color: #fff;

    padding: 100px 25px;

  }

  .container-fluid {

    padding: 60px 50px;

  }
  .bg-grey {
    background-color: #f6f6f6;
  }
  .logo {
    font-size: 200px;
  }
  @media screen and (max-width: 768px) {
    .col-sm-4 {
      text-align: center;
      margin: 25px 0;
    }
  }
  </style>
</head>
<body>

<div class="jumbotron text-center">
  <h1>SVUV Inventory Managament</h1>
  <p>Always happy to help</p>
  <center>
```

```
 <div>
  <button class="btn btn-default btn-lg" ><a href = "/signup">Sign Up</a></button>
  <button class="btn btn-default btn-lg" ><a href = "/signin">Sign In</a></button>
 </div>
</center>
</div>

<!-- Container (About Section) -->
<div class="container-fluid">
 <div class="row">

  <div class="col-sm-12">
    <center>
   <h2>About Company</h2>
    <h4>Manage orders. Track inventory. Handle GST billing. Oversee warehouses. One inventory management software to run all your inventory operations.</h4>
  </center>
  </div>
 </div>
</div>
```

```
</body>
</html>
```

Sign In Page

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://getbootstrap.com/docs/5.0/assets/css/docs.css" rel="stylesheet">

    <!-- Bootstrap core CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+fzT" crossorigin="anonymous">
    <title>Document</title>
</head>
<body>
    <section class="vh-100 bg-image"
```

```
  style="background-image: url('https://mdbcdn.b-cdn.net/img/Photos/new-templates/search-box/img4.webp');">
  <div class="mask d-flex align-items-center h-100 gradient-custom-3">
    <div class="container h-100">
      <div class="row d-flex justify-content-center align-items-center h-100">
        <div class="col-12 col-md-9 col-lg-7 col-xl-6">
          <div class="card" style="border-radius: 15px;">
            <div class="card-body p-5">
              <h2 class="text-uppercase text-center mb-5">Sign In</h2>

              <form>

                <div class="form-outline mb-4">
                  <input type="email" id="form3Example3cg" class="form-control form-control-lg" />
                  <label class="form-label" for="form3Example3cg">Your Email</label>
                </div>

                <div class="form-outline mb-4">
                  <input type="password" id="form3Example4cg" class="form-control form-control-lg" />
                  <label class="form-label" for="form3Example4cg">Password</label>
                </div>


                <div class="d-flex justify-content-center">
                  <button type="button"
                    class="btn btn-block btn-lg gradient-custom-4 text-body" style="background-
color:rgba(74,194,154,255);">Sign In</button>
                </div>


              </form>


            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
</body>
</html>
```

Sign Up page

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://getbootstrap.com/docs/5.0/assets/css/docs.css" rel="stylesheet">

    <!-- Bootstrap core CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-iYQeCzEYFbKjA/T2uDLTpkwGzCiq6soy8tYaI1GyVh/UjpbCx/TYkiZhlZB6+fzT" crossorigin="anonymous">
    <title>Document</title>
</head>
<body>
    <section class="vh-100 bg-image"
  style="background-image: url('https://mdbcdn.b-cdn.net/img/Photos/new-templates/search-box/img4.webp');">
  <div class="mask d-flex align-items-center h-100 gradient-custom-3">
    <div class="container h-100">
      <div class="row d-flex justify-content-center align-items-center h-100">
```

```
        <div class="col-12 col-md-9 col-lg-7 col-xl-6">
          <div class="card" style="border-radius: 15px;">
            <div class="card-body p-5">
              <h2 class="text-uppercase text-center mb-5">Create an account</h2>

              <form>

                <div class="form-outline mb-4">
                  <input type="text" id="form3Example1cg" class="form-control form-control-lg" />
                  <label class="form-label" for="form3Example1cg">Your Name</label>
                </div>

                <div class="form-outline mb-4">
                  <input type="email" id="form3Example3cg" class="form-control form-control-lg" />
                  <label class="form-label" for="form3Example3cg">Your Email</label>
                </div>

                <div class="form-outline mb-4">
                <input type="password" id="form3Example4cg" class="form-control form-control-lg" />

                <label class="form-label" for="form3Example4cg">Password</label>

              </div>



              <div class="form-outline mb-4">

                <input type="password" id="form3Example4cdg" class="form-control form-control-lg" />

                <label class="form-label" for="form3Example4cdg">Repeat your password</label>

              </div>



              <div class="d-flex justify-content-center">

                <button type="button"

                  class="btn btn-block btn-lg gradient-custom-4 text-body" style="background-color: rgba(74,194,154,255);">Register</button>

              </div>



            </form>



          </div>
          </div>

        </div>

      </div>

    </div>

  </div>

</section>

</body>

</html>
```

Index

```
<!DOCTYPE html>

<html lang="en">



<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>IMS</title>

    <link rel="stylesheet" href="/static/css/style.css">

    <link href="https://fonts.googleapis.com/icon?family=Material+Icons+Sharp" rel="stylesheet">

</head>
```

```
<body>
    <div class="container">
        <aside>
            <div class="top">
                <div class="logo"> <img src="/static/ims.png">
                    <h2>I<span class="danger">M</span>S</h2>
                </div>
                <div class="close" id="close-btn"><span class="material-icons-sharp">close</span></div>
            </div>
            <div class="sidebar">
                <a href="#" class="active">
                    <span class="material-icons-sharp">grid_view</span>
                    <h3>Dashboard</h3>
                </a>
                <a href="#">
                    <span class="material-icons-sharp">person_outline</span>
                    <h3>Customers</h3>
                </a>
```

```
<a href="#">
    <span class="material-icons-sharp">receipt_long</span>
    <h3>Orders</h3>
</a>
<a href="#">
    <span class="material-icons-sharp">insights</span>
    <h3>Analytics</h3>
</a>
<a href="#">
    <span class="material-icons-sharp">mail_outline</span>
    <h3>Messages</h3>

</a>
<a href="#">
    <span class="material-icons-sharp">inventory</span>
    <h3>Products</h3>
</a>
<a href="#">
    <span class="material-icons-sharp">description</span>
    <h3>Reports</h3>
```

```
                </a>

                <a href="#">

                    <span class="material-icons-sharp">settings</span>

                    <h3>Settings</h3>

                </a>

                <a href="#">

                    <span class="material-icons-sharp">add</span>

                    <h3>Add Product</h3>

                </a>

                <a href="#">

                    <span class="material-icons-sharp">logout</span>

                    <h3>Logout</h3>

                </a>

            </div>

        </aside>

         <!------------------------------------------END-OF-ASIDE------------------------------------------------
>

        <main>

            <h1>Dashboard</h1>
```

```
<div class="date">
    <input type="date">
</div>

<div class="insights">
    <div class="sales">
        <span class="material-icons-sharp">analytics</span>
        <div class="middle">
            <div class="left">
                <h3>Total Sales</h3>
                <h1>Rs.25000</h1>
            </div>
            <div class="progress">
                <svg>
                    <circle cx='38' cy='38' r='36'></circle>
                </svg>
                <div class="number">
                    <p> 81%</p>
                </div>
            </div>
    </div>

    <small class="text-muted">Last 24 Hours</small>

</div>

<!-------------------------------END OF SALES--------------------------------->



<div class="expenses">

    <span class="material-icons-sharp">bar_chart</span>

    <div class="middle">

        <div class="left">

            <h3>Total Expenses</h3>

            <h1>Rs.14000</h1>

        </div>

        <div class="progress">

            <svg>

                <circle cx='38' cy='38' r='36'></circle>

            </svg>

            <div class="number">

                <p> 62%</p>

            </div>

        </div>
    </div>
    <small class="text-muted">Last 24 Hours</small>
</div>
<!-----------------------------END OF EXPENSES------------------------------->

<div class="income">
    <span class="material-icons-sharp">stacked_line_chart</span>
    <div class="middle">
        <div class="left">
            <h3>Total Income</h3>
            <h1>Rs.10000</h1>
        </div>
        <div class="progress">
            <svg>
                <circle cx='38' cy='38' r='36'></circle>
            </svg>
            <div class="number">
                <p> 44%</p>
            </div>
        </div>
```

```
                </div>

                <small class="text-muted">Last 24 Hours</small>

              </div>

              <!-------------------------------END OF INCOME--------------------------------->

            </div>

          </main>

        </div>

      </body>



      </html>
```

Orders

```
      {% extends "layout.html" %}

      {% block content %}

      <div class="wrapper">

        <!-- Sidebar  -->

        {% include 'sidebar.html' %}

        <!-- Page Content  -->

        <div id="content">

          <h2>Orders</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="" method="post">
    <h3>Create Order</h3>
    <div class="field">
      <label class="custom-label" for="item"
        >Enter Stock ID</label
      >
      <input
        class="text-inputs"
        type="text"
        name="stock_id"
        placeholder="1"
      />
    </div>
    <div class="field">
    <label class="custom-label" for="item"
      >Enter Quantity</label
    >
    <input
      class="text-inputs"
      type="number"
      name="quantity"
      placeholder="10"
    />
  </div>
  <button class="submit-button">Create</button>
</form>

<form action="" method="post">
  <h3>Update Order</h3>
  <div class="field">
    <label class="custom-label" for="item"
      >Enter Order ID</label
    >
    <input
    class="text-inputs"

    name="item"

    type="number"

    placeholder="1"

  />

</div>

<div class="field">

  <label for="custom-label" for="input-field"

    >Choose a field -

  </label>

  <select name="input-field" id="field">

    <option value="STOCKS_ID">STOCKS_ID</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" />
```

```html
    </div>
    <button class="submit-button">Update</button>
  </form>
  <form action="" method="post">
    <h3>Cancel Order</h3>
    <div class="field">
      <label class="custom-label" for="item"
        >Enter Order ID</label
      >
      <input
        class="text-inputs"
        name="order_id"
        type="number"
        placeholder="1"
      />
    </div>
    <button class="submit-button red-button">Cancel</button>
  </form>
</div>
```

```
        jbbhjbhkjvkjhv
    </div>
</div>
{% endblock content %}
```

Profile

```
{% extends "layout.html" %} {% block content %}
<div class="wrapper">
    <!-- Sidebar  -->
    {% include 'sidebar.html' %}
    <!-- Page Content  -->
    <div id="content">
        jbbhjbhkjvkjhv
        <h2>Profile</h2>
        <hr />
        <div class="user-deatils">
            <h3>User Details</h3>
            <h4>USERNAME :</h4>
            <h4>FIRSTNAME :</h4>
            <h4>LASTNAME :</h4>
            <h4>EMAIL :</h4>
```

```
</div>
<hr />
<div class="forms-wrapper mg-20">
    <form action="" 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="" 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>
  </div>
</div>
{% endblock content %}
```

Flask.py

```
from flask import Flask, render_template, url_for, request, redirect, session, make_response , g ,flash
#from flask_login import login_required, login_user, logout_user, current_user
import sqlite3 as sql
from functools import wraps
import re
import ibm_db
import os

from datetime import datetime, timedelta
```

```
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=125f9f61-9715-46f9-9399-
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30426;SECURITY
=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=ktx29071;PWD=zl0kavBhK4
VQc9KJ", ", ")


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)


#----------------------------home-------------------------------------


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

def home():

   if session.get('loggedin')== True:

         return redirect(url_for("dashboard"))

   else:

      return render_template('home.html')
```

```
#--------------------------logout-------------------------------------


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

def logout():

   session.pop("user",None)

   session.pop("loggedin",None)

   session.pop("registered",None)

   session.pop("uname",None)

   session.pop("ord_id",None)

   session.pop("ord_item_id",None)

   session.pop("ord_quantity",None)

   session.pop("ord_item_ppq",None)

   return redirect(url_for("home"))


#--------------------------login-------------------------------------


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

def signin():

   lmsg = ''

   if session.get('loggedin')== True:
    return redirect(url_for('dashboard'))

else:

  if request.method == 'POST':

    session.pop("user",None)

    un = request.form['username']

    pd = request.form['password_1']

    print(un, pd)

    sql = "SELECT * FROM CUSTOMERS 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['user'] = account['EMAIL']

      session['uname']=account['NAME']

      lmsg = 'Logged in successfully !'

      return redirect(url_for('dashboard'))
```

```
            else:

                lmsg = 'Incorrect username / password !'

                return render_template('signin.html',title="Signin",lmsg=lmsg)

        else:

            return render_template('signin.html',title="Signin")



#----------------------------dashboard---------------------------------------



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

def dashboard():

    if session.get('loggedin')== True:

        uname = session['uname']

        user = session['user']

        sql = 'SELECT * FROM ITEMS WHERE EMAIL =?'

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, user)

        ibm_db.execute(stmt)

        dictionary=ibm_db.fetch_assoc(stmt)

        # if session.get('ord_id')== True:

        #    ord_id = session['ord_id']
        #    ord_item_id = session['ord_item_id']

        #    oiqty=dictionary["ITEMSTOCK"]

        #    niqty=session['ord_quantity']

        #    new_qty= (int)(oiqty)- (int)(niqty)

        #    orpq=dictionary["ITEMRPQ"]

        #    nrpq=session['ord_item_ppq']

        #    new_rqp= (int)(orpq)- (int)(nrpq)

        #    newTotal = new_qty * new_rqp


        items=[]

        headings = [*dictionary]

        while dictionary != False:

            items.append(dictionary)

            dictionary = ibm_db.fetch_assoc(stmt)



        return render_template('dashboard.html',title="Welcome,
"+uname,headings=headings,data=items)

    else:

        return redirect(url_for("signin"))
```

```
#-----------------------------register---------------------------------------

@app.route('/signup', methods=['GET', 'POST'])
def signup():
  rmsg = ''
  if session.get('registered')==True or session.get('loggedin')==True:
    return redirect(url_for("signin"))
  else:
    if request.method == 'POST':
      session.pop('loggedin',None)
      fname = request.form['name']
      mail = request.form['mail']
      npwd = request.form['npwd']
      cpwd = request.form['cpwd']
      check_sql = 'SELECT * FROM customers WHERE Email =?'
      stmt = ibm_db.prepare(conn, check_sql)
      ibm_db.bind_param(stmt, 1, mail)
      ibm_db.execute(stmt)
      acc_check = ibm_db.fetch_assoc(stmt)
      print(acc_check)
      if acc_check and acc_check['EMAIL']==mail:

        rmsg = 'Account already exits!!'

        return render_template('signup.html',title="Signup",rmsg=rmsg)

      elif not bool(re.match('[a-zA-Z\s]+$', fname)):

        rmsg = 'Name must contain only alphabets'

        return render_template('signup.html',title="Signup",rmsg=rmsg)

      elif not re.match(r'[^@]+@[^@]+\.[^@]+', mail):

        rmsg = 'Enter valid email address'

        return render_template('signup.html',title="Signup",rmsg=rmsg)

      elif npwd != cpwd:

        rmsg = 'Password does not match'

        return render_template('signup.html',title="Signup",rmsg=rmsg)

      else:

        sql = "INSERT INTO customers (Name,Email,Password,Cpassword) VALUES(?,?,?,?);"

        istmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(istmt, 1, fname)

        ibm_db.bind_param(istmt, 2, mail)

        ibm_db.bind_param(istmt, 3, npwd)

        ibm_db.bind_param(istmt, 4, cpwd)
```

```
            ibm_db.execute(istmt)

            rmsg='Registeration Successful'

            session['registered'] = True

            return render_template('signin.html',title="Signin",lmsg=rmsg)

      else:

         return render_template('signup.html',title="Signup",rmsg=rmsg)



#----------------------orders-------------------------------------



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

def orders():

    if session.get('loggedin')== True:

        uname = session['uname']

        user = session['user']

        sql = 'SELECT * FROM ORDERS WHERE EMAIL =?'

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, user)

        ibm_db.execute(stmt)

        dictionary=ibm_db.fetch_assoc(stmt)

        items=[]
        headings = [*dictionary]

        while dictionary != False:

            items.append(dictionary)

            dictionary = ibm_db.fetch_assoc(stmt)


        return render_template('orders.html',title=uname+'\'s
orders',headings=headings,data=items)

    else:

        return redirect(url_for("signin"))


#-----------------------------supplies------------------------------


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

def supplies():

   if session.get('loggedin')!= True:

         return redirect(url_for("signin"))

   else:

      return render_template('supplies.html')


#-----------------------------profile------------------------------
```

```
@app.route('/profile', methods=['GET', 'POST'])
def profile():
    if session.get('loggedin')== True:

        username=session['uname']
        email=session['user']

        return render_template('profile.html',usname=username,email=email)
    else:
        return redirect(url_for("signin"))


@app.route('/updatepassword', methods=['GET', 'POST'])
def updatepassword():
    if session.get('loggedin')!= True:
            return redirect(url_for("signin"))
    else:
        email=session['user']
        uname = session['uname']
        upmsg="Previous password not matched"
        pp = request.form['prev-password']

        cp = request.form["cur-password"]

        cop = request.form['confirm-password']

        print(pp, cp)

        sql = "SELECT * FROM CUSTOMERS WHERE email =? AND PASSWORD=?"

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt, 1, email)

        ibm_db.bind_param(stmt, 2, pp)

        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        if account:

            query = 'UPDATE CUSTOMERS SET PASSWORD=?,CPASSWORD=? WHERE EMAIL=?'

            pstmt = ibm_db.prepare(conn, query)

            ibm_db.bind_param(pstmt, 1, cp)

            ibm_db.bind_param(pstmt, 2, cop)

            ibm_db.bind_param(pstmt, 3, email)

            ibm_db.execute(pstmt)

            upmsg="Profile password updated"

        return render_template('profile.html',usname=uname,email=email,upmsg=upmsg)
```

```
#----------------------------editStock-----------------------------

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

def editstock():

   if session.get('loggedin')!= True:

         return redirect(url_for("signin"))

   else:

      return render_template('edit_stock.html')


#--------------------------addItem---------------------------------

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

def createitem():

    if request.method == 'POST':

        cimsg="

        item_id = request.form['item_id']

        email = session['user']

        query = 'SELECT * FROM ITEMS WHERE ITEMID = ? AND EMAIL = ?'
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt, 1, item_id)
        ibm_db.bind_param(stmt, 2, email)
        ibm_db.execute(stmt)
        dictionary = ibm_db.fetch_assoc(stmt)
        if bool(dictionary)==False:
            item_name=request.form['item_name']
            quantity = request.form['quantity']
            item_ppq=request.form['item_ppq']
            iprice=(int)(quantity)* (int)(item_ppq)
            query = 'INSERT INTO ITEMS
(ITEMID,ITEMNAME,ITEMSTOCK,ITEMRPQ,ITEMTOTALWORTH,EMAIL)
VALUES (?,?,?,?,?,?)'
            pstmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(pstmt, 1, item_id)
            ibm_db.bind_param(pstmt, 2, item_name)
            ibm_db.bind_param(pstmt, 3, quantity)
            ibm_db.bind_param(pstmt, 4, item_ppq)
            ibm_db.bind_param(pstmt, 5, iprice)
            ibm_db.bind_param(pstmt, 6, email)
            ibm_db.execute(pstmt)
            cimsg="Item added successfully"
            return render_template('edit_stock.html',cimsg=cimsg)

        else:
            cimsg="Item already Exists!!"
            return render_template('edit_stock.html',cimsg=cimsg)

    else:
        return redirect(url_for('dashboard'))

#-----------------------------removeITEM-----------------------------


@app.route('/removeitem', methods=['GET','POST'])
def removeitem():
    if request.method == 'POST':
        delimsg=''
        ritem = request.form['ritem']
        email = session['user']
        query = 'SELECT * FROM ITEMS WHERE ITEMID = ?'
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt, 1, ritem)

        ibm_db.execute(stmt)

        dictionary = ibm_db.fetch_assoc(stmt)

        if bool(dictionary)==True:

            query = 'DELETE FROM ITEMS WHERE ITEMID = ? AND EMAIL = ?'

            pstmt = ibm_db.prepare(conn, query)

            ibm_db.bind_param(pstmt, 1, ritem)

            ibm_db.bind_param(pstmt, 2, email)

            ibm_db.execute(pstmt)

            delimsg="Item Removed successfully"

            return render_template('edit_stock.html',delimsg=delimsg)



        else:

            delimsg="Item Does Not Exists!!"

            return render_template('edit_stock.html',delimsg=delimsg)



    else:

        return redirect(url_for('dashboard'))



#---------------------------addOrder----------------------------------
```

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

def createorder():

    if request.method == 'POST':

        comsg=''

        ord_id = request.form['ord_id']

        oitem_id =request.form['oitem_id']

        email = session['user']

        query = 'SELECT * FROM ORDERS WHERE ORDERID = ? AND ITEMID = ? AND EMAIL = ?'

        stmt = ibm_db.prepare(conn, query)

        ibm_db.bind_param(stmt, 1, ord_id)

        ibm_db.bind_param(stmt, 2, oitem_id)

        ibm_db.bind_param(stmt, 3, email)

        ibm_db.execute(stmt)

        dictionary = ibm_db.fetch_assoc(stmt)

        if bool(dictionary)==False:

            ord_quantity = request.form['ord_quantity']

            oitem_ppq=request.form['oitem_ppq']

            oprice=(int)(ord_quantity)* (int)(oitem_ppq)
            query = 'INSERT INTO ORDERS
(ORDERID,ITEMID,ITEMQUANTITYSOLD,ITEMRPQ,TOTAL,EMAIL) VALUES
(?,?,?,?,?,?)'
            pstmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(pstmt, 1, ord_id)
            ibm_db.bind_param(pstmt, 2, oitem_id)
            ibm_db.bind_param(pstmt, 3, ord_quantity)
            ibm_db.bind_param(pstmt, 4, oitem_ppq)
            ibm_db.bind_param(pstmt, 5, oprice)
            ibm_db.bind_param(pstmt, 6, email)
            ibm_db.execute(pstmt)
            session['ord_id'] = ord_id
            session['ord_item_id'] = oitem_id
            session['ord_quantity'] = ord_quantity
            session['ord_item_ppq'] = oitem_ppq
            comsg="Order created successfully"

            sql = 'SELECT * FROM ITEMS WHERE EMAIL =?'
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, email)
            ibm_db.execute(stmt)
            dictionary=ibm_db.fetch_assoc(stmt)

            oiqty=dictionary["ITEMSTOCK"]

            niqty=ord_quantity

            new_qty = int(oiqty) - int(niqty)

            new_total = int(new_qty)*int(oitem_ppq)



            query = 'UPDATE ITEMS SET ITEMSTOCK=?,ITEMRPQ=?,ITEMTOTALWORTH=? WHERE ITEMID=? AND EMAIL=?'

            pstmt = ibm_db.prepare(conn, query)

            ibm_db.bind_param(pstmt, 1, new_qty)

            ibm_db.bind_param(pstmt, 2, oitem_ppq)

            ibm_db.bind_param(pstmt, 3, new_total)

            ibm_db.bind_param(pstmt, 4, oitem_id)

            ibm_db.bind_param(pstmt, 5, email)

            ibm_db.execute(pstmt)

            return render_template('orders.html',comsg=comsg)



        else:

            comsg="Order already Exists!!"

            return render_template('orders.html',comsg=comsg)
    else:
        return redirect(url_for('dashboard'))


#---------------------------updateOrder-----------------------------------


@app.route('/updateorder', methods=['GET','POST'])
def updateorder():
    if request.method == 'POST':
        upomsg=''
        up_ord_id = request.form['up_ord_id']
        up_ord_item =request.form['up_ord_item']
        email = session['user']
        query = 'SELECT * FROM ORDERS WHERE ORDERID = ? AND ITEMID = ? AND EMAIL = ?'
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt, 1, up_ord_id)
        ibm_db.bind_param(stmt, 2, up_ord_item)
        ibm_db.bind_param(stmt, 3, email)
        ibm_db.execute(stmt)
        dictionary = ibm_db.fetch_assoc(stmt)
        if bool(dictionary)==True:

            ord_up_quantity = request.form['ord_up_quantity']

            upd_item_ppq = request.form['upd_item_ppq']

            uoprice=(int)(ord_up_quantity)* (int)(upd_item_ppq)

            query = 'UPDATE ORDERS SET
ITEMQUANTITYSOLD=?,ITEMRPQ=?,TOTAL=? WHERE ORDERID=? AND
ITEMID=? AND EMAIL=?'

            pstmt = ibm_db.prepare(conn, query)

            ibm_db.bind_param(pstmt, 1, ord_up_quantity)

            ibm_db.bind_param(pstmt, 2, upd_item_ppq)

            ibm_db.bind_param(pstmt, 3, uoprice)

            ibm_db.bind_param(pstmt, 4, up_ord_id)

            ibm_db.bind_param(pstmt, 5, up_ord_item)

            ibm_db.bind_param(pstmt, 6, email)

            ibm_db.execute(pstmt)



            upomsg="Order updated successfully"

            return render_template('orders.html',upomsg=upomsg)



        else:

            upomsg="Order Does Not Exists!!"
            return render_template('orders.html',upomsg=upomsg)
    else:
        return redirect(url_for('dashboard'))


#-----------------------------removeORDER-----------------------------


@app.route('/removeorder', methods=['GET','POST'])
def removeorder():
    if request.method == 'POST':
        delomsg=''
        cancel_ord = request.form['cancel_ord']
        email = session['user']
        query = 'SELECT * FROM ORDERS WHERE ORDERID = ?'
        stmt = ibm_db.prepare(conn, query)
        ibm_db.bind_param(stmt, 1, cancel_ord)
        ibm_db.execute(stmt)
        dictionary = ibm_db.fetch_assoc(stmt)
        if bool(dictionary)==True:
            query = 'DELETE FROM ORDERS WHERE ORDERID = ? AND EMAIL = ?'
            pstmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(pstmt, 1, cancel_ord)

            ibm_db.bind_param(pstmt, 2, email)

            ibm_db.execute(pstmt)

            delomsg="Order Cancelled successfully"

            return render_template('orders.html',delomsg=delomsg)



        else:

            delomsg="Order Does Not Exists!!"

            return render_template('orders.html',delomsg=delomsg)



    else:

        return redirect(url_for('dashboard'))




if __name__ == '__main__':

   app.run(debug = True)
```

GitHub:

https://github.com/IBM-EPBL/IBM-Project-19340-1659696489

Project Demo Link:

https://drive.google.com/drive/folders/16nxxjnuDNBQHuqd35HWlsE0D5O_L1uQC