

IBM–NALAIYA THIRAN PROJECT

INVENTORY MANAGEMENT SYSTEM

INDUSTRY MENTOR : VASUDEVA HANUSH

FACULTY MENTOR : ARAVINDRAJ

TEAM ID : PNT2022TMID43345

TEAM LEAD : VIKHAS S G

TEAM MEMBER : ABIVISHVAS A

TEAM MEMBER : KISHORE VENKATESH S

TEAM MEMBER : VENKATESHWARAN M

TABLE OF CONTENT

| CHAPTER | CONTENTS | PAGENO |
|---------|--|--------|
| 1 | INTRODUCTION 1.1 PROJECT OVERVIEW 1.2 PURPOSE | 1 |
| 2 | LITERATURE SURVEY 2.1 EXISTING PROBLEM 2.2 REFERENCES 2.3 PROBLEM STATEMENT DEFINITION | 3 |
| 3 | IDEATION & PROPOSED SOLUTION 3.1 EMPATHY MAP CANVAS 3.2 IDEATION & BRAINSTROMING 3.3 PROPOSED SOLUTION 3.4 PROBLEM SOLUTION FIT | 6 |
| 4 | REQUIREMENT ANALYSIS 4.1 FUNCTIONAL REQUIREMENT 4.2 NON-FUNCTIONAL REQUIREMENTS | 14 |
| 5 | PROJECT DESIGN 5.1 DATA FLOW DIAGRAMS 5.2 SOLUTION & TECHNICAL ARCHITECTURE 5.3 USER STORIES | 16 |
| 6 | PROJECT PLANNING & SCHEDULING 6.1 SPRINT PLANNING & ESTIMATION 6.2 SPRINT DELIVERY SCHEDULE 6.3 REPORTS FROM JIRA | 22 |

| | | |
|-----------|--|-----------|
| 7 | CODING & SOLUTIONING 7.1 FEATURE1 7.2 FEATURE2 7.3 DATABASE SCHEMA | 28 |
| 8 | TESTING 8.1 TEST CASES 8.2 USER ACCEPTANCE TESTING | 33 |
| 9 | RESULTS 9.1PERFORMANCE METRICS | 35 |
| 10 | ADVANTAGES & DISADVANTAGES | 36 |
| 11 | CONCLUSION | 38 |
| 12 | FUTURESCOPE | 39 |
| 13 | APPENDIX SOURCE CODE GITHUB & PROJECT DEMO LINK | 40 |

1.INTRODUCTION:

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

For example From NaavebUROM

- Estimation of new raw material required.
- Preparation of purchase order.

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

1.1. PROJECT OVERVIEW:

Inventory management system is an application which is helpful for business operate. Inventory management is a challenging problem area in supply chain management. Companies need to have inventories in warehouses in order to fulfil customer demand, meanwhile these inventories have holding costs and this is frozen fund that can be lost. Therefore, the task of inventory management is to find the quantity of inventories that will fulfil the demand, avoiding overstocks. This paper presents a case study for the assembling company on inventory management. It is proposed to use inventory management in order to decrease stock levels and to apply an agents system for automation of inventory management processes. Inventory management system (IMS) use for a departmental store.

1.2. PURPOSE:

A case study at 'Guckenheimer' (an on-site corporate restaurant management and catering company) cited issues regarding a basic resources requirement list that has to be maintained manually by the staff. To keep track of their inventory levels they have to calculate a list of the groceries utilized during a course of time, calculate and analyze the requirements for the future, and place their next order to the vendors if needed. This process takes up a lot of time and human effort, and is also prone to human error. This poses a problem of a situation that the staff at 'Guckenheimer,' as well as many other restaurants face. It takes up a lot of time to manually keep track of sales and place correct orders to vendors, wasting useful labor in trivial works. A product which would assist in tackling the above mentioned problems would prove to be fruitful to clients such as 'Guckenheimer' and similar enterprises as this product would help convert the unproductive time to something more useful, by removing the unnecessary error prone complications and efforts.

2.LITERATURESURVEY:

2.1. EXISTINGPROBLEM:

Products are considered as the business resources for the organization. This includes managing the product with appropriate way to review any time as per the requirement. Therefore it is important to have a computer based IMS which has the ability to generate reports, maintain the balance of the stock, details about the purchase and sales in the organization. Before developing this application we cameupwith2InventoryManagementSystemexistinginthemarket,which helps to give the knowledge for the development of our project. These application software are only used by the large organization but so we came up with the application which can be used by the small company for the management of their stock in the production houses. After analyzing the other inventory management system we decided to include some of common and key features that should be included in every inventory management system. So we decided to include those things that help the small organization in away or other.

2.2. REFERENCES:

We have referred several documentations for the purpose of development phases.

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

[2] Jimmy Wales, online encyclopedia Wikipedia,

<http://www.wikipedia.org>

[3] James Gosling, Java (Programming Language),

<http://www.java.com>

[4] Names Allaire, Netbeans - Fully-featured Java IDE,

<http://www.netbeans.org>

[5] James Gosling, Welcome to javaworld.com:

how -

to feature and columns by Java expert; news; Java applets; sa

mple code ; tips ,

<http://www.javaworld.com>

[6] Pressman, Roger S.

“Software Engineering A Practitioner” Approach

[7] John Osborn, Java Beans:

Developing Component Software in Java

[8] Doug Lea, Concurrent Programming in Java:

Design Principles and Patterns, Addison-Wesley, November, 1996

[9] Design Report, submitted 9th November 2012.

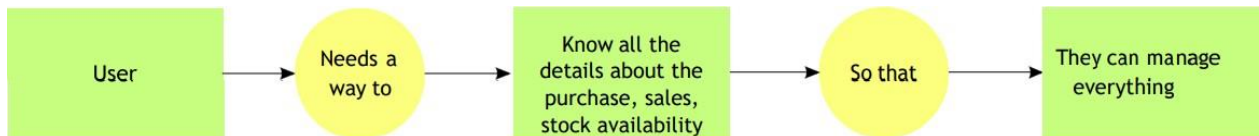
<https://skydrive.live.com/redir?resid=2CEDE9F7F5F99604!196&authkey=!AO5IghTCML6xAk8>

[10] Testing Document, submitted 26th November 2012.

<https://skydrive.live.com/redir?resid=2CEDE9F7F5F99604!161&authkey=!AC8P0Lqe9amxSeM>

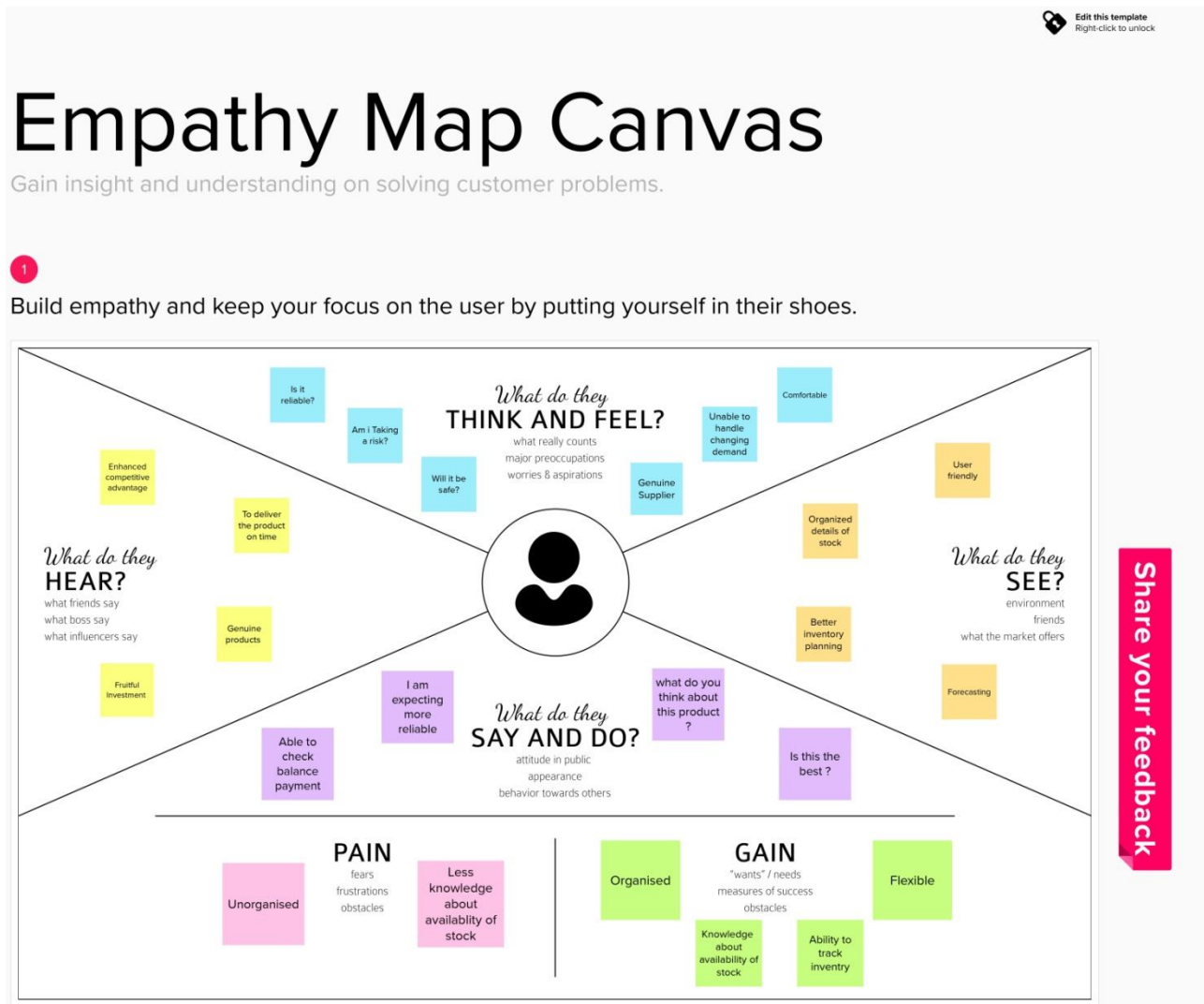
2.3. PROBLEM STATEMENT DEFINITION:

The problem statement aims to make desktop application for retailers and to track all areas of IMS like purchase details, sales details, stock management. The application helps the retailer to have complete insights about the products stored in the inventory and can manage them flexibly.



| | |
|-----------------------------------|---|
| Who does the problem affect? | Retailers and Customers |
| Why is it important to use? | Greater Insights about stocks and Increased productivity |
| What are the Benefits? | Better Inventory Accuracy and avoiding Stockouts and Excess Stock |
| How is it better than the others? | Accessible by retailer and at any time and more Organized Warehouse |
| When to use? | To get rid of obsolete and out of date inventory items. To evaluate movement of specific items. When a company wants a dynamic and systematic system to record and keep their inventory data. |

3.1 Empathy Map Canvas :



3.2 Ideation & Brainstorming :



Student Forum

Civic Engagement Workshop

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 design a
inventory management
system?**



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

ABIVISHVAS A

| | | |
|---|--|----------------------------------|
| Informing the vendors about the product details through Mail. | Analyze Low and High selling products. | Monthly sales analysis. |
| Analyzing the performance of supplier. | Avoid overstocking of products. | Warehouse batch number tracking. |

VIKHAS S G

| | | |
|--|----------------------------|--|
| Maintain the records of stock product. | Enhanced user Interface. | Proper Categorization of products. |
| Feedback and rating of products. | Data Privacy of customers. | Provide information about product weights and markdowns. |

KISHORE VENKATESH S

| | | |
|------------------------------------|------------------------------------|------------------------------|
| Have sufficient amount of product. | Enabling Multiple Payment options. | Tax calculation. |
| Providing EMI Features. | Managing Multiple orders. | Final Checkout of your cart. |

VENKATESHWARAN M

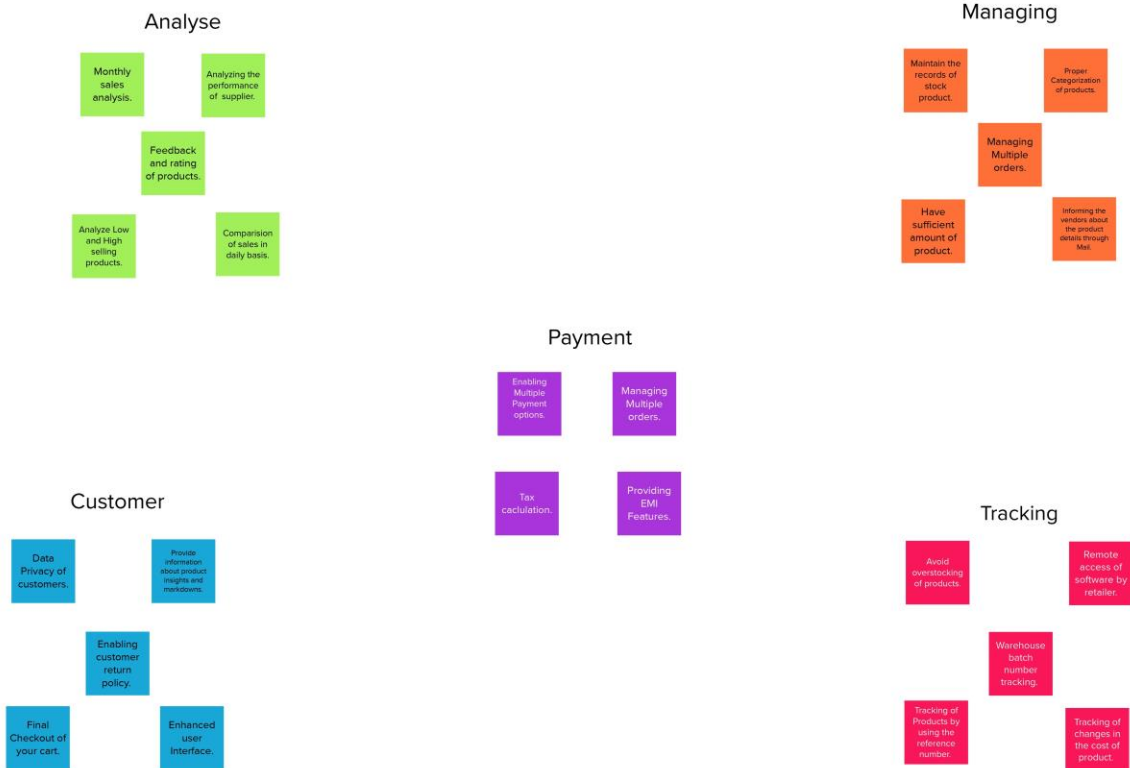
| | | |
|---|---|--|
| Tracking of changes in the cost of product. | Providing Discounts and Offer to clear unsold stocks. | Comparison of sales in daily basis. |
| Tracking of Products by using the reference number. | Enabling customer return policy. | Remote access of software by retailer. |

3

Group ideas

Use this space to group similar ideas from the brainstorm. Each group should have a title that describes what the ideas have in common. If a group is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

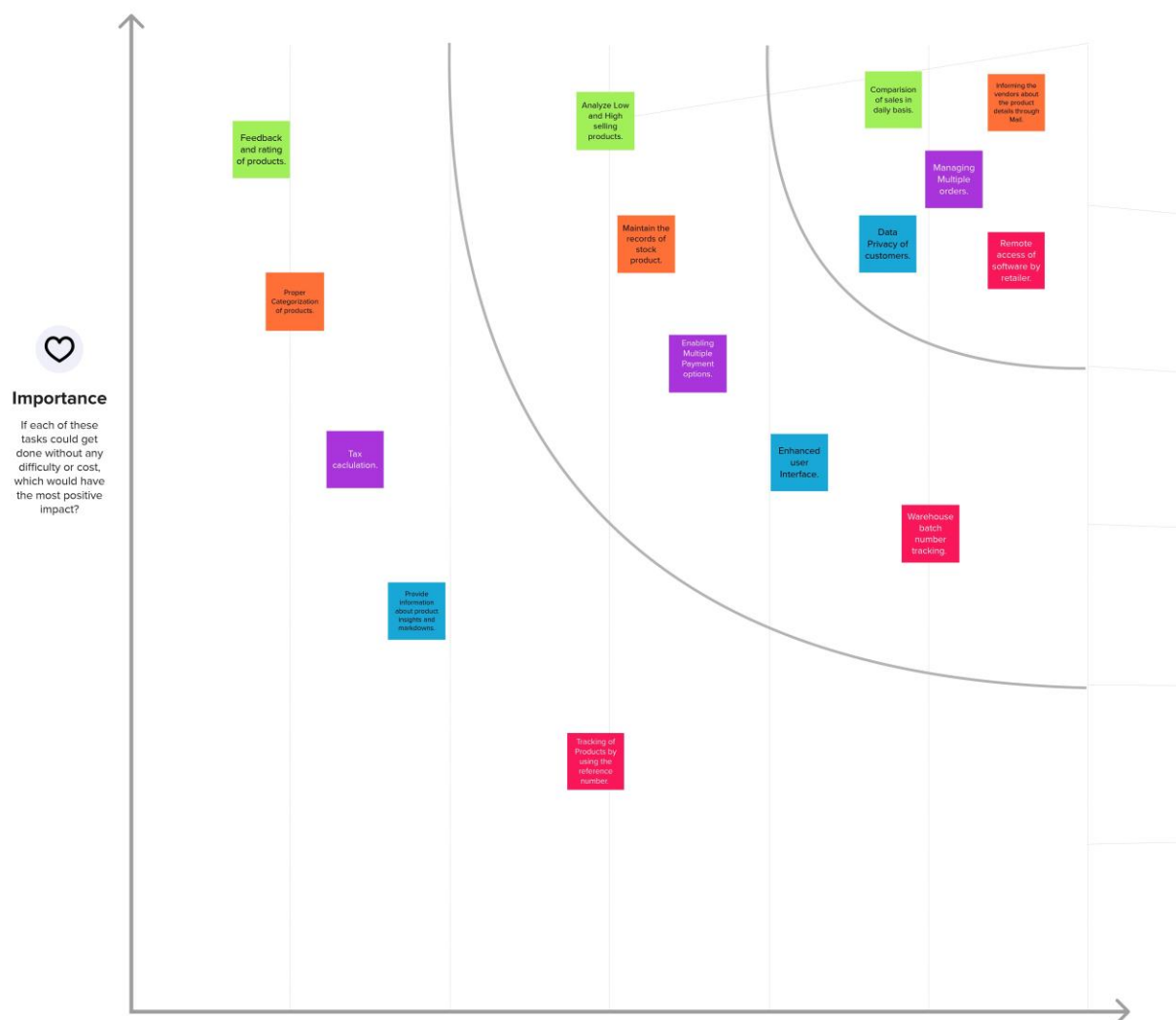


4

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



3.3 Proposed solution:

| S. No | Parameter | Description |
|-------|--|--|
| 1 | Problem statement (problem to be solved) | The problem statement aims to make desktop application for retailers and to track all areas of Inventory Management System like purchase details , sales details , stock management and other policies . |
| 2 | Idea / Solution description | Creating an app that “Tracks all areas of Inventory Management System like purchase details, sales details, stock management and notify the user. |
| 3 | Novelty / Uniqueness | It connects people virtually and give information like purchase details, sales details, stock management. Act as all-in-one place for stock retail and purchases. |
| 4 | Social Impact / Customer Satisfaction | It helps Retailers to manage the entire stock warehouse and a single platform. It also helps in managing the about demand and supply of their Inventories. Proper Inventory Management System reduces the risk of over stocking and prevents wastage of capital and stock of the Retailers. |
| 5 | Business Model (financial Benefit) | Inventory management system helps Retailers to identify which and how much stock to order at what time. Cuts off the unwanted cost for the Retailers. |
| 6 | Scalability of Solution | These solutions streamline order management as they enable business to automatically reorder stock that is running low before it runs out, ensuring no sales opportunities are missed. On the other hand, this also means that overstocking can be avoided if certain products aren't selling as expected. |

| | | | | |
|-------------------------|---|--|--|---------------------------|
| Define CS, fit into CC | 1. CUSTOMER SEGMENT(S) CS Who is your customer? i.e. working parents of 0-5 y.o. Kids <ul style="list-style-type: none"> Retailers Distributors Wholesalers Manufacturers | 6. CUSTOMER CONSTRAINTS CC What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices <ul style="list-style-type: none"> Inventory Tracking Changing demand Managing warehouse spaces Manual documentation | 5. AVAILABLE SOLUTIONS AS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking <ul style="list-style-type: none"> Centralized Tracking Demand forecasting Optimized space Add imagery Software tools to replace manual documentation | Explore AS, differentiate |
| | 2. JOBS-TO-BE-DONE / PROBLEMS J&P Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides. <ul style="list-style-type: none"> To check the efficiency of the warehouse Limited visibility Manual documentation Supply chain complexity | 9. PROBLEM ROOT CAUSE RC What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations. <ul style="list-style-type: none"> Low rate of inventory turnover High cost of storage Inaccurate information about stock movement Quick real time updations on the Quality and quantity of products | 7. BEHAVIOUR BE What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace) <ul style="list-style-type: none"> Secured data Manage, access and control through software Process will be on time FIFO approach to ensure proper transactions of good | |
| Identify strong TR & EM | 3. TRIGGERS TR What triggers customers to act? i.e. seeing their neighbor installing solar panels, reading about a more efficient solution in the news. <ul style="list-style-type: none"> Increased Productivity Easy to access and manage stocks User friendly and better user satisfaction | 10. YOUR SOLUTION SL If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solve a problem, and matches customer behavior. <ul style="list-style-type: none"> This application works on the cloud and uses a DB2 database. This application allows easy access to manage and control the stocks. Provide an option for a graphical view of sales. | 8. CHANNELS of BEHAVIOUR CH 8.1 ONLINE What kind of actions do customers take online? Extract online channels from #7 8.2 OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. online: <ul style="list-style-type: none"> Supports pre-purchase stage Updating of flowing of the stocks regularly offline: <ul style="list-style-type: none"> Manual checking Organised delivery of stocks | Identify strong TR & EM |
| | 4. EMOTIONS: BEFORE / AFTER EM How do customers feel when they face a problem or a job and afterward? i.e. lost, insecure > confident, in control - use it in your communication strategy & design. <ul style="list-style-type: none"> BEFORE <ul style="list-style-type: none"> Less accuracy of stocks Less productivity More work and stress AFTER <ul style="list-style-type: none"> High accuracy of stocks High productivity Less work and stress | | | |

4.1 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 Form Registration through Gmail |
| FR-2 | User Confirmation | Confirmation via Email Confirmation via OTP |
| FR-3 | User Login | Username/Email-ID Login with Password |
| FR-4 | Admin Login | Login with Username/Email-ID Login with Password |
| FR-5 | Inventory Management | Track quantity of products present in inventory at any instant |
| FR-6 | Tracking of stock | Notifications through Email |

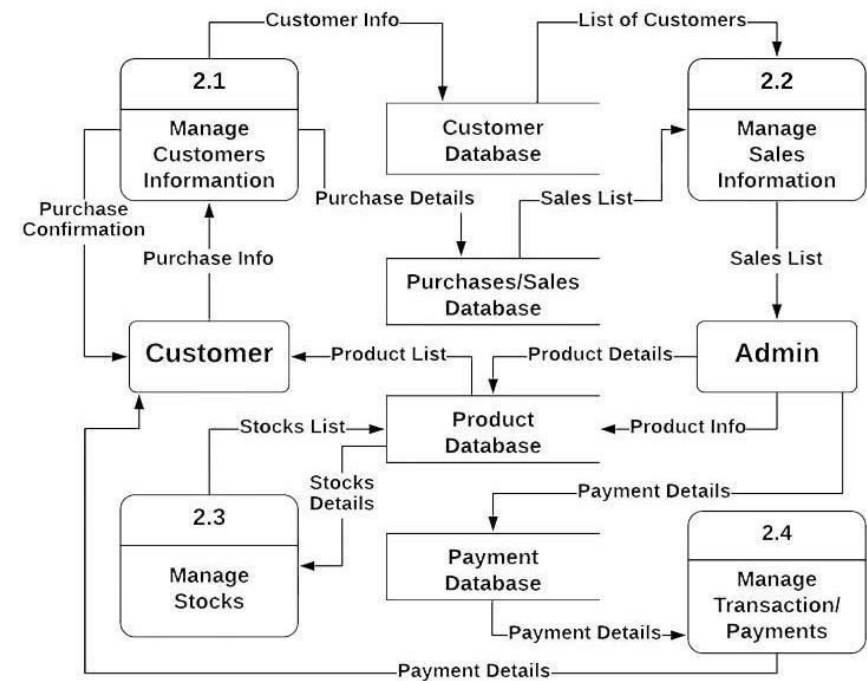
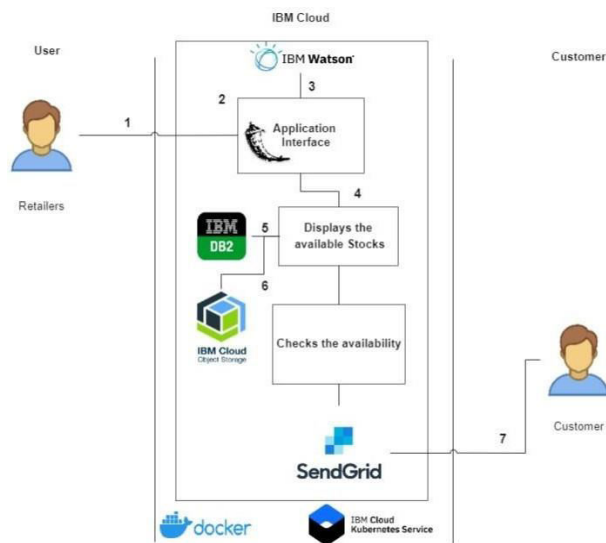
4.2 Non-functional Requirements:

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

| FR No. | Non-Functional Requirement | Description |
|--------|----------------------------|---|
| NFR-1 | Usability | This cloud web application makes the process of inventory management a lot easier which saves money and time both. This system is highly responsive to both desktop and mobile users. |
| NFR-2 | Security | Inventory security aims to prevent inventory losses – for example, due to incorrect storage, theft, or incorrect incoming goods inspection – so that the correct stock is always available. |
| NFR-3 | Reliability | The availability of products should be properly updated for customer satisfaction. The out-of-stock information should be notified. The system must give accurate inventory status to the user continuously. |
| NFR-4 | Performance | The companies have to design and operate materials management and product distribution functions effectively. Inventory control systems enable a business to determine and maintain an optimum level of investment in inventory in order to achieve the required operational performance. |
| NFR-5 | Availability | The software will be available only to the administrator of the organization and the product, as well as customer details, will be recorded by him. He can manage the inventory. |
| NFR-6 | Scalability | The System can manage large inventory and provides quick access to the inventory in no time. |

5.1 Data Flow Diagrams:

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

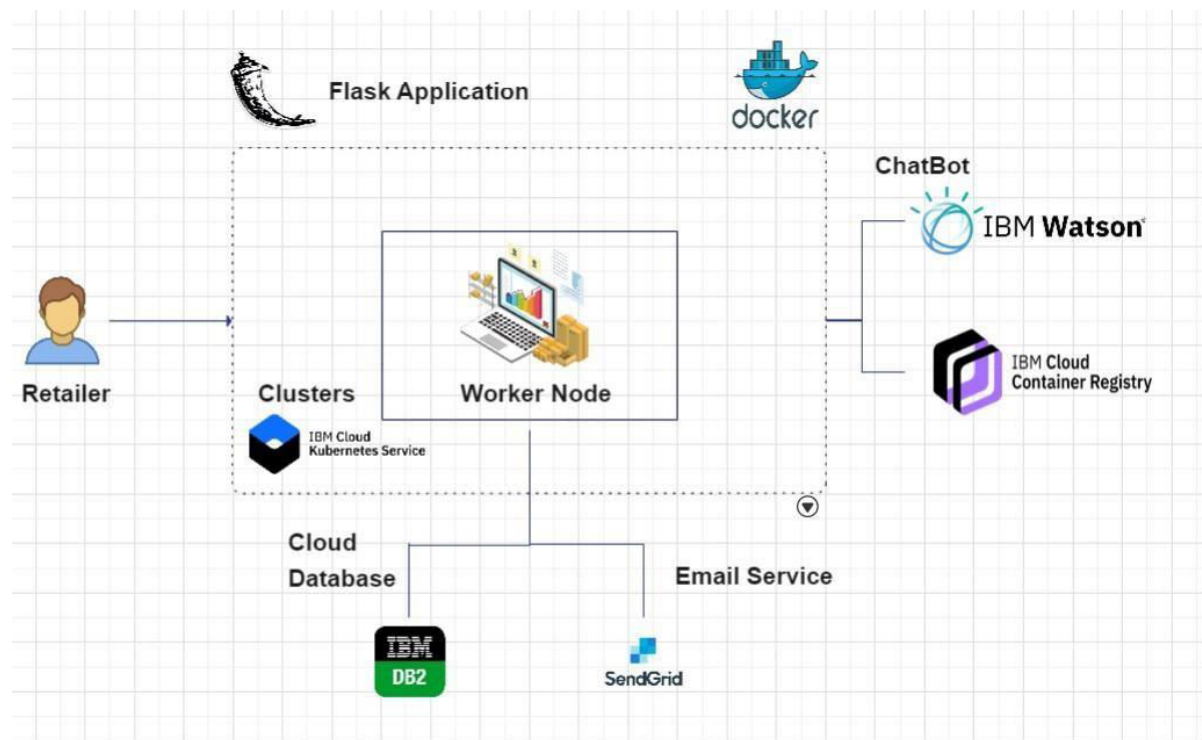


5.2 Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.

Solution Architecture Diagram:



Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

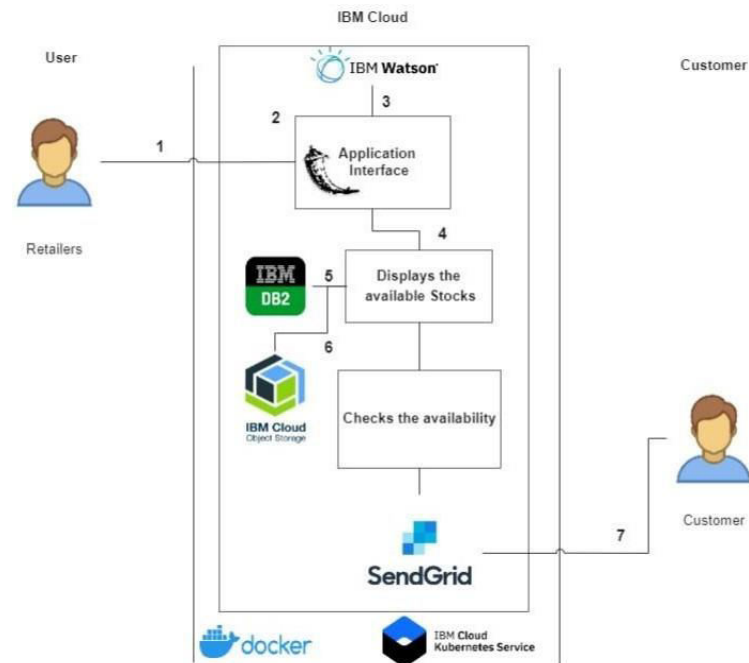


Table-1: Components & Technologies:

| S.No | Component | Description | Technology |
|------|---------------------------------|---|-----------------------|
| 1. | User Interface | User interacts with the cloud web application. | HTML, CSS, JavaScript |
| 2. | Application Logic-1 | Logic for a process in the application | Python |
| 3. | Application Logic-2 | Logic for a process in the application | IBM Watson Assistant |
| 4. | Database | Data Type, Configurations | MySQL |
| 5. | Cloud Database | Database Service on Cloud | IBM DB2 |
| 6. | File Storage | File storage requirements | IBM Object Storage |
| 7. | Infrastructure (Server / Cloud) | Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management | Kubernetes |

Table-2: Application Characteristics:

| S.No | Characteristics | Description | Technology |
|------|--------------------------|--|------------|
| 1. | Open-Source Frameworks | Micro web framework, written in Python | Flask |
| 2. | Security Implementations | List all the security/access controls implemented, use of firewalls, etc. | SHA-256 |
| 3. | Scalable Architecture | Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management. | Kubernetes |
| 4. | Availability | Docker CLI stores its configuration files in a directory called dock within your home directory. | Docker CLI |
| 5. | Performance | To send alerts to users based on their stock | Sendgrid |

5.3 User Stories

Use the below template to list all the user stories for the product.

| User Type | Functional Requirement (Epic) | User Story Number | User Story / Task | Acceptance criteria | Priority | Release |
|------------------------|-------------------------------|-------------------|---|---|----------|----------|
| Customer (Mobile user) | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | I can access my account/dashboard | High | Sprint-1 |
| | | USN-2 | As a user, I will receive a confirmation email once I have registered for the application | I can receive a confirmation email & click confirm | Medium | Sprint-1 |
| | Confirmation | USN-3 | As a user, I will confirm the registration once I have received the email from the application | I can get a confirmation for my email and password and create an authenticated account. | Medium | Sprint-1 |
| | Login | USN-4 | As a user, I can log in to the application through Gmail & Password | I can log onto the application with a verified email and password | High | Sprint-1 |
| | Dashboard | USN-5 | As a user, I can view the dashboard of the application by entering my email & password | Once I log on to the application, I can view products to buy. | High | Sprint-2 |
| | Add items to the cart | USN-6 | As a user, I can add the products I wish to buy to the carts. | As a user, I can buy any product or add it to my cart for buying later | Medium | Sprint-2 |
| | Stock Update | USN-7 | As a user, I can add products that are not available in the dashboard to the stock list. | If any of the products are not available, as a user I can update the inventory and send mail to the owner | Medium | Sprint-3 |
| Customer (Web user) | Registration | USN-8 | 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-9 | As a user, I will receive a confirmation email once I have registered for the application | I can receive a confirmation email & click confirm | Medium | Sprint-1 |
| | Confirmation | USN-10 | As a user, I will confirm the registration once I have received the email from the application | I can get a confirmation for my email and password | Medium | Sprint-1 |

| User Type | Functional Requirement (Epic) | User Story Number | User Story / Task | Acceptance criteria | Priority | Release |
|-------------------------|-------------------------------|-------------------|--|---|----------|----------|
| | | | | and create an authenticated account. | | |
| | Login | USN-11 | As a user, I can log in to the application through Gmail & Password | I can log onto the application with a verified email and password | High | Sprint-1 |
| | Dashboard | USN-12 | As a user, I can view the dashboard of the application by entering my email & password | Once I log on to the application, I can view products to buy. | High | Sprint-2 |
| | Add items to the cart | USN-13 | As a user, I can add the products I wish to buy to the carts. | As a user, I can buy any product or add it to my cart for buying later | Medium | Sprint-2 |
| | Stock Update | USN-14 | As a user, I can add products that are not available in the dashboard to the stock list. | If any of the products are not available, as a user I can update the inventory and send mail to the owner | Medium | Sprint-3 |
| Customer Care Executive | Request to Customer Care | USN-15 | As a user, I can contact the Customer Care Executive and request any services I want from customer care. | As a user, I can contact Customer Care and get support. | Low | Sprint-4 |
| Administrator | Contact Administrator | USN-16 | I can be able to report any difficulties I experience as a report | As a user, and I can give my support in possible ways to the administrator and the administration. | Medium | Sprint-4 |

6.1 Sprint Planning and Estimation

Use the below template to create product backlog and sprint schedule

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|----------|-------------------------------|-------------------|---|--------------|----------|---------------------|
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 5 | High | Vikhas S G |
| Sprint-2 | | USN-2 | As a user, I can register for the application through Gmail account | 8 | Low | Kishore Venkatesh S |
| Sprint-1 | | USN-3 | As a user, I can register for the application through mobile number | 8 | High | Abivishvas A |
| Sprint-1 | Login | USN-4 | As a user, I can log into the application by entering email or mobile number & password | 5 | High | Venkateshwaran M |
| Sprint-2 | | USN-5 | As a user, I can login into the application through OTP. | 3 | Medium | Kishore Venkatesh S |
| Sprint-3 | Dashboard | USN-6 | As a user, I must be able to see my details on the dashboard. | 3 | High | Abivishvas A |
| Sprint-4 | Inventory | USN-7 | As a retailer, I should be able to alter product details in the app | 2 | Medium | Venkateshwaran M |
| Sprint-2 | | USN-8 | As a retailer, I should get alert on stock shortage or unavailability. | 5 | Medium | Abivishvas A |
| Sprint-2 | Order | USN-9 | As a user, I should be able to order items on the app | 2 | High | Venkateshwaran M |
| Sprint-3 | | USN-10 | As a user, I should be able to verify and pay in a secure payment gateway | 3 | High | Kishore Venkatesh S |
| Sprint-3 | | USN-11 | As a user, I should be able to get the product on time. | 5 | Low | Vikhas S G |
| Sprint-4 | Administration | USN-12 | As an administrator, I should be able to edit details of the users of the app. | 8 | High | Vikhas S G |
| Sprint-4 | | USN-13 | Termination user accounts temporarily or permanently if needed. | 5 | Low | Venkateshwaran M |

Project Tracker, Velocity & Burndown Chart:

| Sprint | Total Story Points | Duration | Sprint Start Date | Sprint End Date (Planned) | Story Points Completed (as on Planned End Date) | Sprint Release Date (Actual) |
|----------|--------------------|----------|-------------------|---------------------------|---|------------------------------|
| Sprint-1 | 18 | 6 Days | 24 Oct 2022 | 29 Oct 2022 | 18 | 29 Oct 2022 |
| Sprint-2 | 15 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 15 | 05 Nov 2022 |
| Sprint-3 | 11 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 11 | 12 Nov 2022 |
| Sprint-4 | 15 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | 15 | 19 Nov 2022 |

Velocity:

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

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

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

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

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

6.2 Sprint Delivery Schedule:

| TITLE | DESCRIPTION | DATE |
|--|---|-------------------|
| Literature Survey & Information Gathering | Literature survey on the selected project & gathering information by referring to technical papers, research publications etc. | 1 SEPTEMBER 2022 |
| Prepare Empathy Map | Prepare Empathy Map Canvas to capture the user Pains & Gains, Prepare list of problem statements. | 25 SEPTEMBER 2022 |
| Ideation | List the by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance. | 8 SEPTEMBER 2022 |
| Proposed Solution | Prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc. | 19 SEPTEMBER 2022 |
| Problem Solution Fit | Prepare problem - solution fit document. | 23 SEPTEMBER 2022 |
| Solution Architecture | Prepare a solution architecture document. | 01 OCTOBER 2022 |

| | | |
|---|---|------------------|
| Customer Journey | Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit). | 07 OCTOBER 2022 |
| Functional Requirement | Prepare the functional requirement document. | 10 OCTOBER 2022 |
| Data Flow Diagrams | Draw the data flow diagrams and submit for review. | 14 OCTOBER 2022 |
| Technology Architecture | Prepare the technology architecture diagram. | 25 OCTOBER 2022 |
| Prepare Milestone & Activity List | Prepare the milestones & activity list of the project. | 3 NOVEMBER 2022 |
| Project Development - Delivery of Sprint-1, 2, 3 & 4 | Develop & submit the developed code by testing it. | WORK IN PROGRESS |

6.3 Reports from JIRA

Jira Software

Your work

Projects

Filters

Dashboards

People

Apps

Create

Q Search

🔔

?

⚙

👤

ibm-inventory manag...

Software project

PLANNING

Roadmap

Backlog

Board

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Projects / ibm-inventory management

All sprints

🔗

☆

Complete sprint

...

Q

👤

👥

Epic

Sprint

GROUP BY

None

Insights

TO DO 3 ISSUES

As a user, I should be able to get the product on time.

IIM-6

As an administrator, I should be able to edit details of the users of the app

IIM-7

Termination user accounts temporarily or permanently if needed.

IN PROGRESS 1 ISSUE

As a user, I should be able to order items on the app

IIM-5

DONE 2 ISSUES

As a user, I can register for the application by entering my email, password, and confirming my password.

IIM-3

As a user, I can log into the application by entering email or mobile number & password

IIM-4

Quickstart

×

Jira Software

Your work

Projects

Filters

Dashboards

People

Apps

Create

Q Search

🔔

?

⚙

👤

ibm-inventory manag...

Software project

PLANNING

Roadmap

Backlog

Board

DEVELOPMENT

Code

Project pages

Add shortcut

Project settings

You're in a team-managed project

Learn more

Projects / ibm-inventory management

Backlog

...

Q

👤

👥

Epic

Insights

▼ IIM Sprint 2 25 Oct – 29 Oct (1 issue)

0 0 0

Complete sprint

...

IIM-4 As a user, I can log into the application by entering email or mobile number & password

DONE

👤

+ Create issue

▼ IIM Sprint 3 1 Nov – 5 Nov (2 issues)

0 0 0

Complete sprint

...

IIM-5 As a user, I should be able to order items on the app

IN PROGRESS

👤

IIM-6 As a user, I should be able to get the product on time.

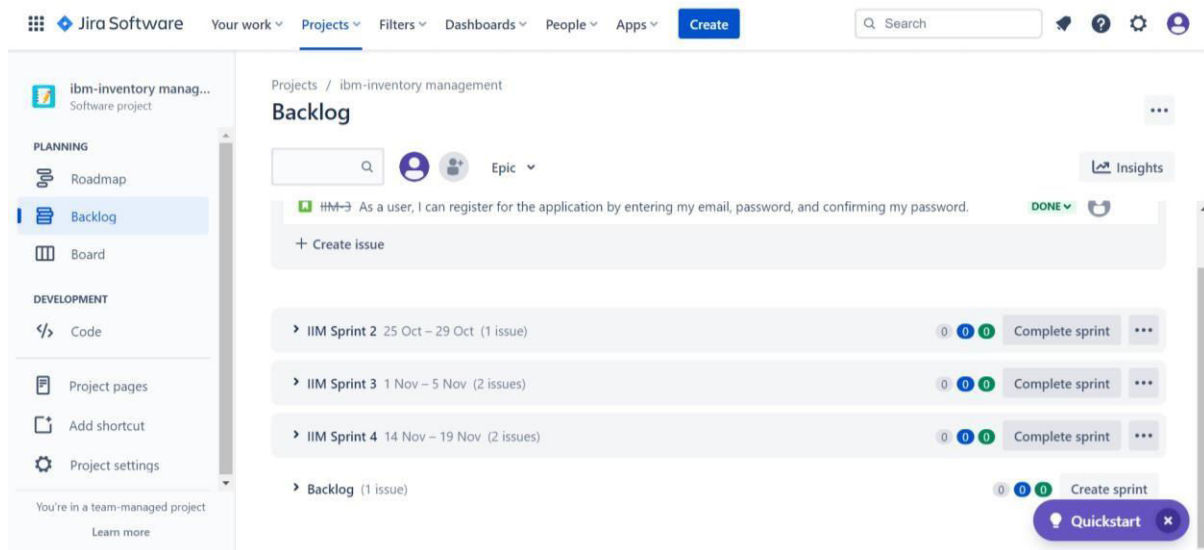
TO DO

👤

+ Create issue

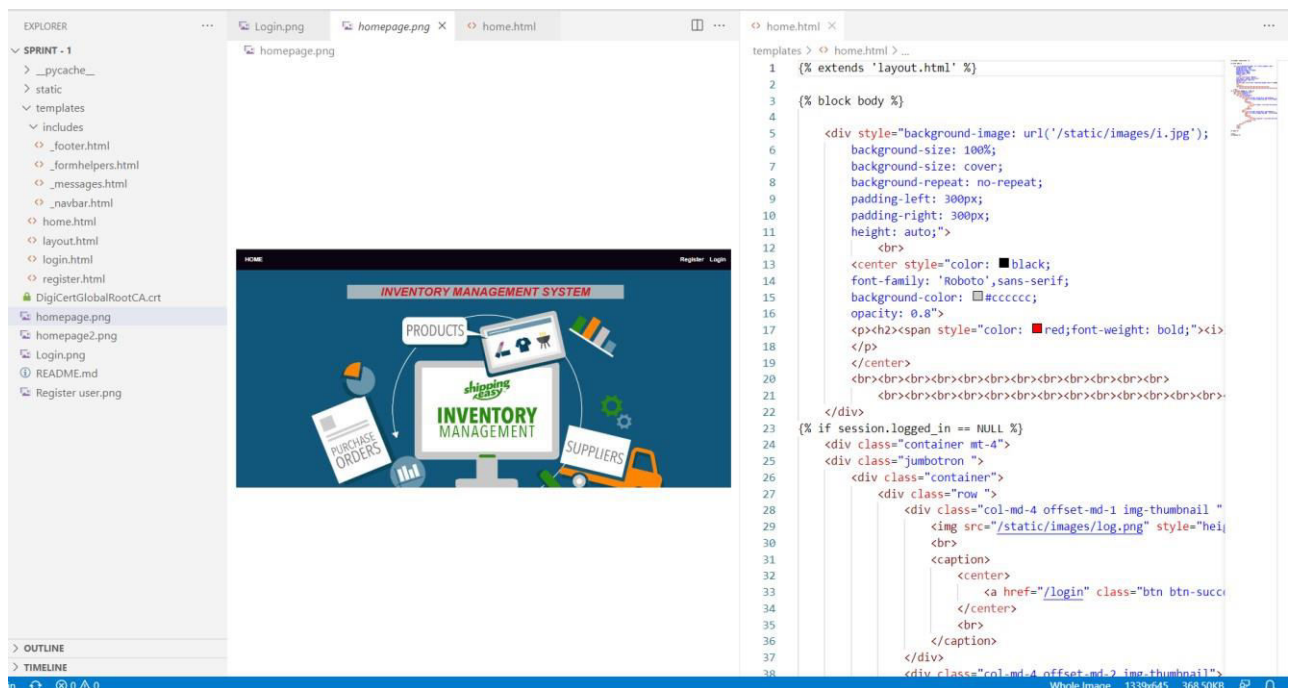
Quickstart

×

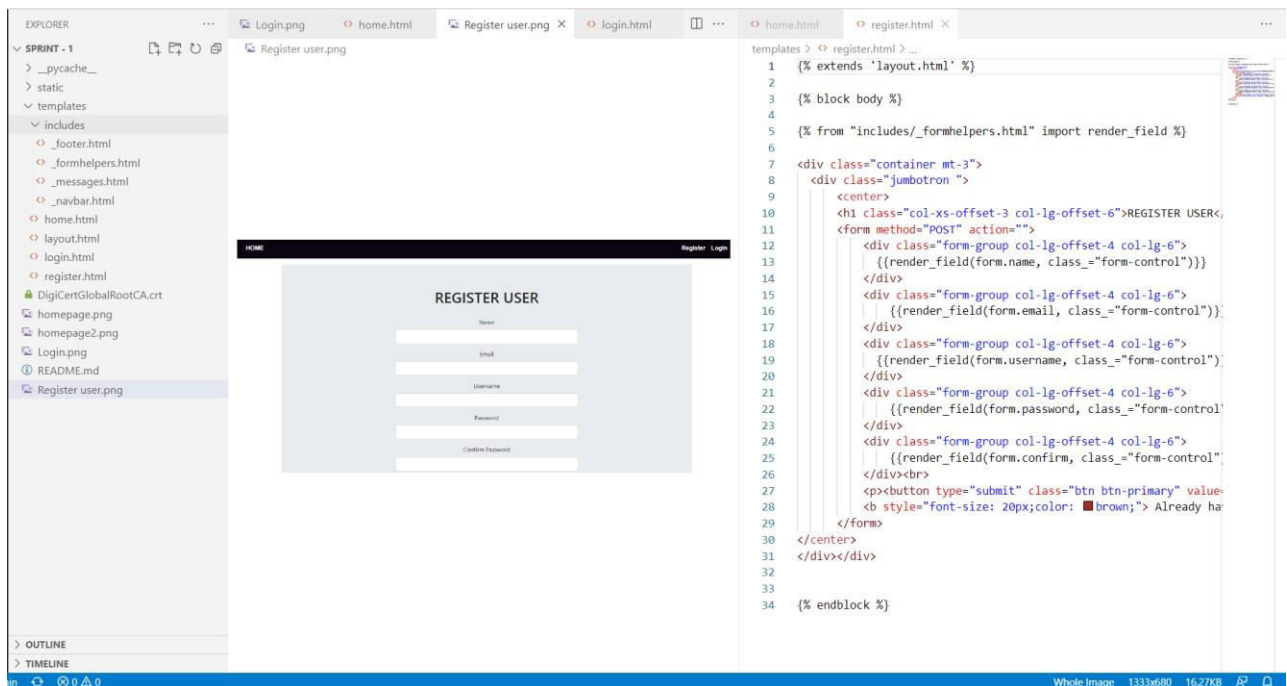


This is how to use the SCRUM framework for working with JIRA tool.

7.1 Feature 1:



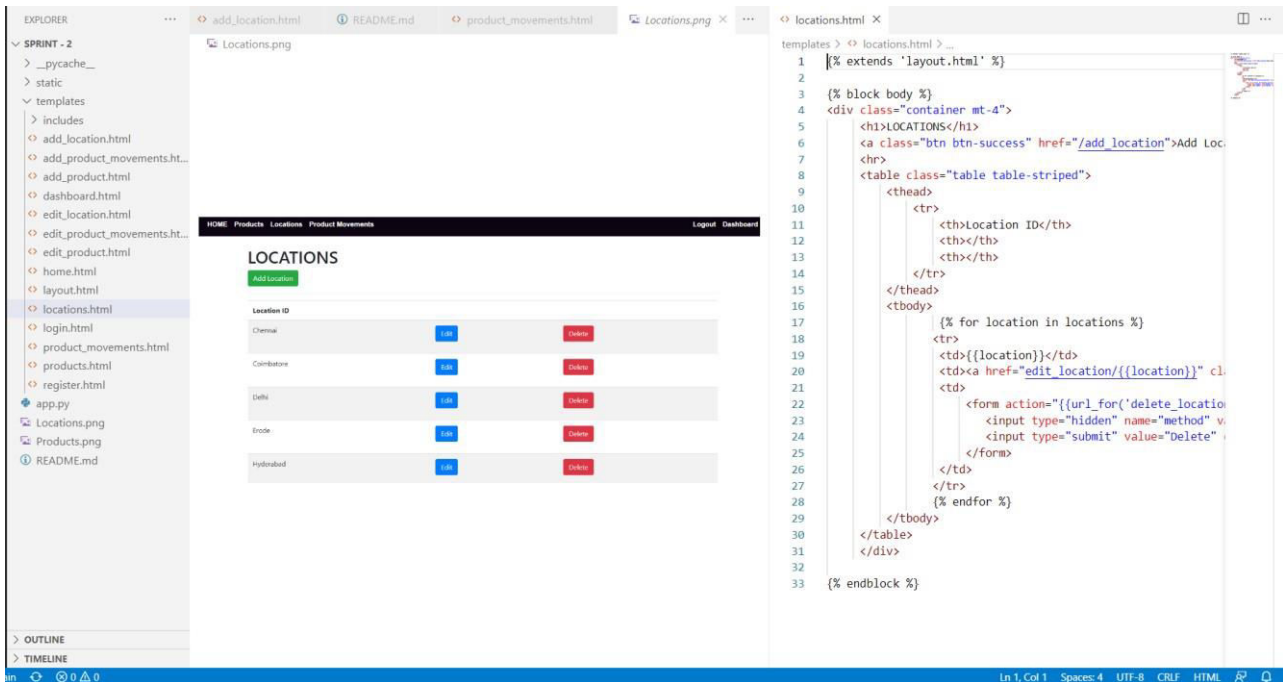
Frontend of the Home Page



Frontend of the Registration Page

7.2 Feature 2:

- Code the backend part to link the frontend pages created in Sprint 1.
- Create the main pages and functionalities of the application - Products Page, Locations Page, Product Movements Page and Dashboard.



Location Page



Add Location

Location ID

Add

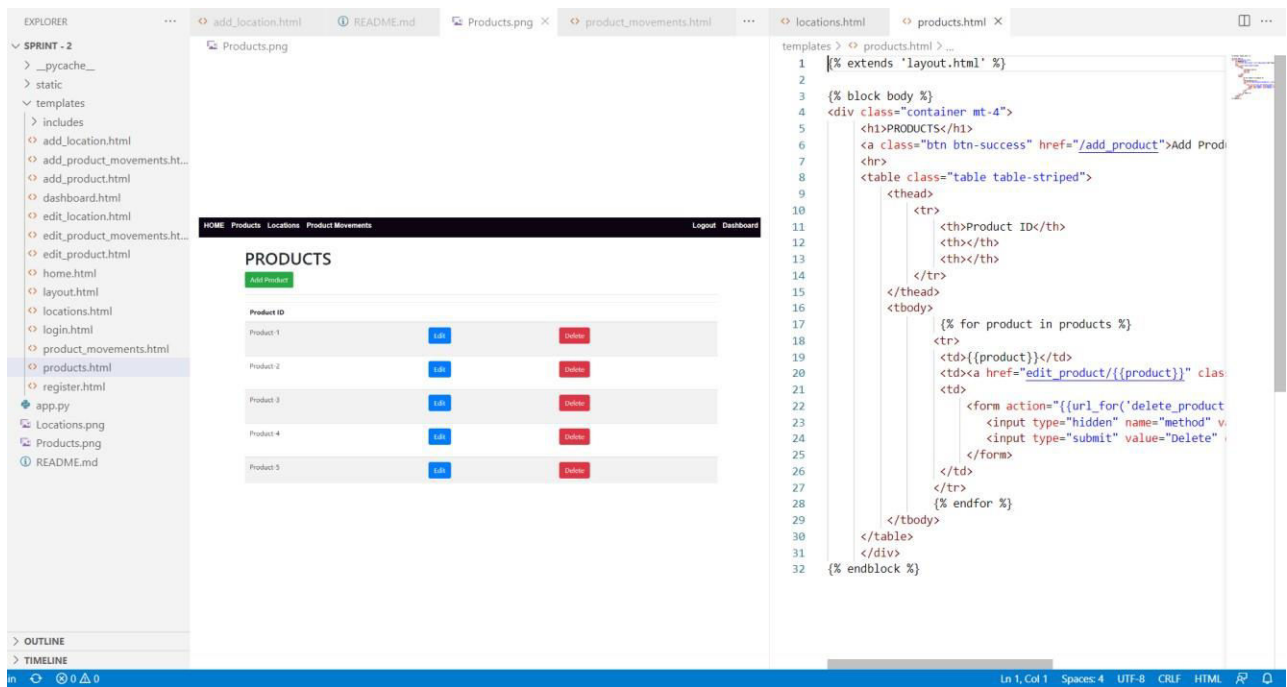


Edit Product

Product ID

Update

To add and edit product



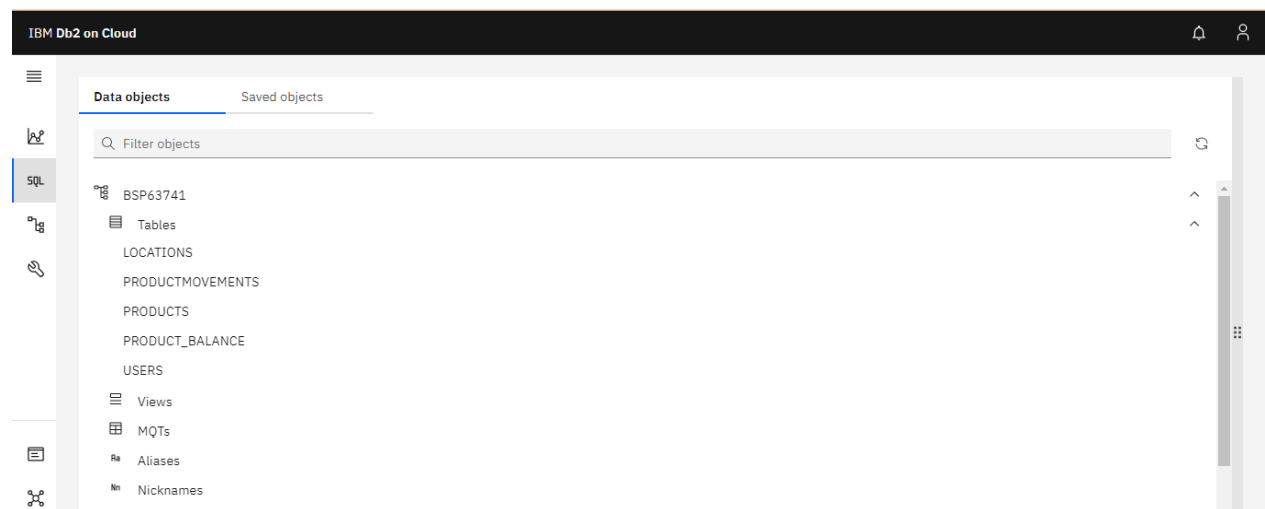
Products Page

| HOME | Products | Locations | Product Movements | Logout | Dashboard |
|-----------------------|----------------------------|---------------|-------------------|------------|-----------|
| PRODUCT MOVEMENTS | | | | | |
| Add Product Movements | | | | | |
| Movement ID | Time | From Location | To Location | Product ID | Quantity |
| 54 | 2022-11-20 03:10:48 | Chennai | Erode | Product-2 | 150 |
| 56 | 2022-08-10 08:06:48 | Trichy | Erode | Product-4 | 50 |
| 57 | 2022-01-01 10:12:48 | Coimbatore | Chennai | Product-5 | 150 |
| 58 | 2022-08-10 08:06:48 | Tirupur | Chennai | Product-1 | 70 |
| 59 | 2022-01-01 10:12:48 | Erode | Delhi | Product-6 | 120 |
| 72 | 2022-11-12 17:33:08.315496 | Trichy | Tirupur | Product-2 | 30 |

Product Movements Page

7.3 Database Scheme:

- Create Database and necessary Tables
- Insert Values into the Database
- Create Login Page and New Registration Page.
- Create Frontend Pages of the application



Database

Table details

PRODUCTMOVEMENTS

12 rows 32.0 KB

Find

| Name | Data type | Nullable | Length | Scale |
|---------------|-----------|----------|--------|-------|
| MOVEMENT_ID | INTEGER | N | | 0 |
| TIME | TIMESTAMP | N | 10 | 6 |
| FROM_LOCATION | VARCHAR | Y | 255 | 0 |
| TO_LOCATION | VARCHAR | Y | 255 | 0 |
| PRODUCT_ID | VARCHAR | Y | 255 | 0 |
| QTY | INTEGER | Y | | 0 |

Product movements table

Table details

PRODUCTS


6 rows 32.0 KB

Find

| Name | Data type | Nullable | Length | Scale |
|------------|-----------|----------|--------|-------|
| PRODUCT_ID | VARCHAR | N | 255 | 0 |

Products table

Table details

 **PRODUCT_BALANCE**

13 rows

 Find

| Name | Data type | Nullable | Length | Scale |
|-------------|-----------|----------|--------|-------|
| ID | INTEGER | N | | 0 |
| PRODUCT_ID | VARCHAR | Y | 255 | 0 |
| LOCATION_ID | VARCHAR | Y | 255 | 0 |
| QTY | INTEGER | Y | | 0 |

Product Balance table

| ID | PRODUCT_ID | LOCATION_ID | QTY |
|----|------------|-------------|-----|
| 15 | Product-1 | Delhi | 20 |
| 16 | Product-3 | Chennai | 120 |
| 17 | Product-3 | Coimbatore | 30 |
| 18 | Product-4 | Erode | 100 |
| 19 | Product-6 | Trichy | 500 |
| 20 | Product-5 | Kochi | 80 |
| 21 | Product-2 | Tirupur | 80 |
| 22 | Product-2 | Hyderabad | 30 |
| 23 | Product-1 | Hyderabad | 100 |
| 24 | Product-2 | Trichy | 100 |

Product Balance table values

Table details

 **USERS**

4 rows

32.0 KB

 Find



| Name | Data type | Nullable | Length | Scale |
|---------------|-----------|----------|--------|-------|
| ID | INTEGER | N | | 0 |
| NAME | VARCHAR | Y | 100 | 0 |
| EMAIL | VARCHAR | Y | 100 | 0 |
| USERNAME | VARCHAR | Y | 30 | 0 |
| PASSWORD | VARCHAR | Y | 100 | 0 |
| REGISTER_DATE | TIMESTAMP | N | 10 | 6 |

Users Table

8. TESTING:

TEST CASES:

| Features to be tested | Test Description |
|--|--|
| Login to the system | This tests the login interface of the system. |
| Adding a Recipe to database | This test is conducted to verify if a recipe is successfully added to the database. This will check if the recipe is added to its header table and also check if the recipe details are added to the recipe details table. |
| Adding an Ingredient to database | This tests checks if new ingredient is added correctly to the database with the specified details. |
| Adding a Vendor to the database | This test checks if the newly added vendor is correctly added to the database with the specified details. |
| Checking the threshold levels | This test is conducted to verify if the ingredients that are below the threshold levels are listed by the function when called by the user. The verification is done by referring to the database. |
| Updating the sales for the day | This test is conducted to test the sales update in the database. The test checks if the database is updated with the correct ingredient values based on the sales data input to the system. |
| Updating the order reception to database | This test is conducted to test the correct updating of the database after receiving the order from the vendor. |
| Create Orders | This test is conducted to check the order creation capability of the system. The list of ingredients that is generated for order must comply with the set conditions of threshold levels |

USER ACCEPTANCE TESTING:

Test case : Testing the Add Products Interface and its functioning

Case 1: Testing the Quantity input field.

Case 2: Testing the Username and Password field.

Case 3: Testing the Product list and Warehouse list.

Case 4: Testing the available Quantity list.

Case 5: Testing the all the above cases together and checking if the entries are updated to the tables in database.

Test Case : Check Threshold Interface

Case 1: Check the Quantities if under the threshold values are shown in the Products below threshold list.

Case 2: Check if the Add Product button asks the user to enter values for all the Product ID listed under the Products.

Case 3: Check if pressing the Add Product button saves in the Database with the product details in it.

Test Case : Testing the Update after add product interface

Case 1: Test the Product list box.

Case 2: Test the quantity text field.

Case 3: Test the product sold list box & quantity sold list box.

Case 4: Test if the details are updated to the database when requested.

9. RESULTS

9.1 Performance Metrics:

Inventory Performance is a measure of how effectively and efficiently inventory is used and replenished. The goal of inventory performance metrics is to compare actual on-hand dollars versus forecasted cost of goods sold. Many Lean practitioners claim that inventory performance is the single best indicator of the overall operational performance of a facility.

Inventory performance looks at and is measured using either Inventory Days On-Hand (DOH) or Inventory Turns.

- **Inventory Days On-Hand:** The number of days it would take to consume current on-hand inventory. Always measure multiple inventory item numbers in terms of currency (i.e. COGS).
- **Inventory Turns:** The number of times inventory is replaced in a year.

10. ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

- 1. It helps to maintain the right amount of stocks:** contrary to the belief that is held by some people, inventory management does not seek to reduce the amount of inventory that you have in stock, however, it seeks to maintain an equilibrium point where your inventory is working at a maximum efficiency and you do not have to have many stocks or too few stocks at hand at any particular point in time. The goal is to find that zone where you are never losing money in your inventory in either direction. With the aid of an efficient inventory management strategy, it is easy to improve the accuracy of inventory order.
- 2. It leads to a more organized warehouse:** with the aid of a good inventory management system, you can easily organize your warehouse. If your warehouse is not organized, you will find it very difficult to manage your inventory. A lot of businesses choose to optimize their warehouse by putting the items that have the highest sales together in a place that is easy to access in the warehouse. This ultimately helps to speed up order fulfilment and keeps clients happy.

DISADVANTAGES:

1. **Bureaucracy:** even though inventory management allows employees at every level of the company to read and manipulate company stock and product inventory, the infrastructure required to build such a system adds a layer of bureaucracy to the whole process and the business in general. In instances where inventory control is in-house, this includes the number of new hires that are not present to regulate the warehouse and facilitate transactions. In instances where the inventory management is in the hands of a third party, the cost is a subscription price and a dependence on another separate company to manage its infrastructure. No matter the choice you go for, it translates to a higher overhead cost and more layers of management between the owner and the customer. From the view point of the customer, a problem that requires senior management to handle will take a longer period of time before it will be trashed out.
2. **Impersonal touch:** another disadvantage of inventory management is a lack of personal touch. Large supply chain management systems make products more accessible across the globe and most provide customer service support in case of difficulty, but the increase in infrastructure can often mean a decrease in the personal touch that helps a company to stand out above the rest. For instance, the sales manager of a small manufacturing company that sells plumbing supplies to local plumbers can throw in an extra box of washers or elbows at no charge to the customer without raising any alarms. This is done for the sake of customer relations and often makes the customer feel like he is special. While free materials can also be provided under inventory management, processing time and paper work make obtaining the material feel more like a chore for the customer or even an entitlement.

11. CONCLUSION:

The project “Inventory Management System for Calculation and Ordering of Available and Processed Resources” mainly as the name suggests deals with the calculation of the available and processed resources for an accurate inventory control and process management for a domain specific client who are related to the subject of food chains/outlets. This enables the inventory to be applied at every level in the hierarchy of the products and its complex combinations of recipes. A system that accurately calculates the atomic ingredients used for making a recipe then automatically performs the back end operation pertaining to a database of many relational tables onto which the changes are being made with each and every operation performed on the front end and which also shows up if at the time of retrieval.

The most important part of Inventory controlling is its ability to check for threshold levels and alert the manager to replenish the stock before it reaches a danger zone. So as when an ingredient level goes below the threshold level then it routes an alert to the manager. Then if needed accordingly an automated order form is produced so as to each specific vendor along with the quantities needed for replenishment. As a part of the standard maintaining a drill of risk management is done in order to sustain during the days of special occasion or holidays when the demand reaches to rather more different scale as compared to other days. These occasions call on for special inclusions into the menu which reflects on the recipes and in turn reflects the ingredients being used up eventually. Thus was provided the liberty of adding special recipe to the menu for some special occasion and is regarded as a key feature.

12.FUTURE SCOPE:

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

13. APPENDIX :

Source Code:

```
from flask import Flask, render_template, flash,
redirect, url_for, session, request, logging
from wtforms import Form, StringField,
TextAreaField, PasswordField, validators,
SelectField, IntegerField
import ibm_db
from passlib.hash import sha256_crypt
from functools import wraps

from ma import *

#creating an app instance
app = Flask(__name__)

app.secret_key='a'

conn=ibm_db.connect("DATABASE=bludb;HOST
NAME=9938aec0-8105-433e-8bf9-
0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.app
domain.cloud;PORT=32459;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=
wlc83769;PWD=A2zmu4rlGJerlkWz;",",")

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

#Products
@app.route('/products')
def products():
    sql = "SELECT * FROM products"
    stmt = ibm_db.prepare(conn, sql)
    result=ibm_db.execute(stmt)

    products=[]
    row = ibm_db.fetch_assoc(stmt)
    while(row):
        products.append(row)
```

```

        row = ibm_db.fetch_assoc(stmt)
    products=tuple(products)
    #print(products)

    if result>0:
        returnrender_template('products.html', products
= products)
    else:
        msg='No products found'
        returnrender_template('products.html',
msg=msg)

#Locations
@app.route('/locations')
deflocations():

    sql = "SELECT * FROM locations"
    stmt = ibm_db.prepare(conn, sql)
    result=ibm_db.execute(stmt)

    locations=[]
    row = ibm_db.fetch_assoc(stmt)
    while(row):
        locations.append(row)
        row = ibm_db.fetch_assoc(stmt)
    locations=tuple(locations)
    #print(locations)

    if result>0:
        returnrender_template('locations.html',
locations = locations)
    else:
        msg='No locations found'
        returnrender_template('locations.html',
msg=msg)

#Product Movements
@app.route('/product_movements')
defproduct_movements():

    sql = "SELECT * FROM productmovements"
    stmt = ibm_db.prepare(conn, sql)
    result=ibm_db.execute(stmt)

```

```

movements=[]

row = ibm_db.fetch_assoc(stmt)
while(row):
    movements.append(row)
    row = ibm_db.fetch_assoc(stmt)
movements=tuple(movements)
#print(movements)

if result>0:

returnrender_template('product_movements.html',
movements = movements)
else:
    msg='No product movements found'

returnrender_template('product_movements.html',
msg=msg)

#Register Form Class
classRegisterForm(Form):
    name = StringField('Name',
[validators.Length(min=1, max=50)])
    username = StringField('Username',
[validators.Length(min=1, max=25)])
    email = StringField('Email',
[validators.length(min=6, max=50)])
    password = PasswordField('Password', [
        validators.DataRequired(),
        validators.EqualTo('confirm',
message='Passwords do not match')
    ])
    confirm = PasswordField('Confirm Password')

#user register
@app.route('/register', methods=['GET','POST'])
defregister():
    form = RegisterForm(request.form)
    # sql1="Select * from users"
    # stmt1 = ibm_db.prepare(conn, sql1)
    # result=ibm_db.execute(stmt1)
    # d=ibm_db.fetch_assoc(stmt1)

```

```

# print(d)

if request.method == 'POST' and form.validate():

    name = form.name.data
    email = form.email.data
    username = form.username.data
    password =
sha256_crypt.hash(form.password.data)
    # password = form.password.data
    print(password)

    sql1="INSERT INTO users(name, email,
username, password) VALUES(?,?,?,?)"
    stmt1 = ibm_db.prepare(conn, sql1)
    ibm_db.bind_param(stmt1,1,name)
    ibm_db.bind_param(stmt1,2,email)
    ibm_db.bind_param(stmt1,3,username)
    ibm_db.bind_param(stmt1,4,password)
    ibm_db.execute(stmt1)
    #for flash messages taking parameter and the
category of message to be flashed
    flash("You are now registered and can log in",
"success")

    #when registration is successful redirect to
home
    return redirect(url_for('login'))
    return render_template('register.html', form =
form)

#User login
@app.route('/login', methods = ['GET', 'POST'])
def login():
    if request.method == 'POST':
        #Get form fields
        username = request.form['username']
        password_candidate = request.form['password']

        sql1="Select * from users where username = ?"
        stmt1 = ibm_db.prepare(conn, sql1)
        ibm_db.bind_param(stmt1,1,username)
        result=ibm_db.execute(stmt1)

```

```

d=ibm_db.fetch_assoc(stmt1)
if result >0:
    #Get the stored hash
    data = d
    password = data['PASSWORD']

    #compare passwords
    if sha256_crypt.verify(password_candidate,
password):
        #Passed
        session['logged_in'] = True
        session['username'] = username

        flash("you are now logged in","success")
        return redirect(url_for('dashboard'))
    else:
        error = 'Invalid Login'
        returnrender_template('login.html',
error=error)
        #Close connection
        cur.close()
    else:
        error = 'Username not found'
        returnrender_template('login.html',
error=error)
        returnrender_template('login.html')

#check if user logged in
defis_logged_in(f):
    @wraps(f)
    defwrap(*args, **kwargs):
        if'logged_in'in session:
            returnf(*args, **kwargs)
        else:
            flash('Unauthorized, Please login','danger')
            return redirect(url_for('login'))
    return wrap

#Logout
@app.route('/logout')
@is_logged_in
deflogout():
    session.clear()

```

```

flash("You are now logged out", "success")
return redirect(url_for('login'))

#Dashboard
@app.route('/dashboard')
@is_logged_in
defdashboard():
    sql2="SELECT product_id, location_id, qty
FROM product_balance"
    sql3="SELECT location_id FROM locations"
    stmt2 = ibm_db.prepare(conn, sql2)
    stmt3 = ibm_db.prepare(conn, sql3)

    result=ibm_db.execute(stmt2)
    ibm_db.execute(stmt3)

    products=[]
    row = ibm_db.fetch_assoc(stmt2)
    while(row):
        products.append(row)
        row = ibm_db.fetch_assoc(stmt2)
    products=tuple(products)

    locations=[]
    row2 = ibm_db.fetch_assoc(stmt3)
    while(row2):
        locations.append(row2)
        row2 = ibm_db.fetch_assoc(stmt3)
    locations=tuple(locations)

    locs = []
    foriin locations:
        locs.append(list(i.values())[0])

    if result>0:
        returnrender_template('dashboard.html',
products = products, locations = locs)
    else:
        msg='No products found'
        returnrender_template('dashboard.html',
msg=msg)

#Product Form Class

```



```

class ProductForm(Form):
    product_id = StringField('Product ID',
    [validators.Length(min=1, max=200)])
    product_cost = StringField('Product Cost',
    [validators.Length(min=1, max=200)])
    product_num = StringField('Product Num',
    [validators.Length(min=1, max=200)])

#Add Product
@app.route('/add_product', methods=['GET',
'POST'])
@is_logged_in
def add_product():
    form = ProductForm(request.form)
    if request.method == 'POST' and form.validate():
        product_id = form.product_id.data
        product_cost = form.product_cost.data
        product_num = form.product_num.data

        sql1 = "INSERT INTO products(product_id,
product_cost, product_num) VALUES(?,?,?)"
        stmt1 = ibm_db.prepare(conn, sql1)
        ibm_db.bind_param(stmt1, 1, product_id)
        ibm_db.bind_param(stmt1, 2, product_cost)
        ibm_db.bind_param(stmt1, 3, product_num)

        ibm_db.execute(stmt1)

        flash("Product Added", "success")

        return redirect(url_for('products'))

    return render_template('add_product.html',
form=form)

#Edit Product
@app.route('/edit_product/<string:id>',
methods=['GET', 'POST'])
@is_logged_in
def edit_product(id):
    sql1 = "Select * from products where product_id=
?"
    stmt1 = ibm_db.prepare(conn, sql1)

```

```

ibm_db.bind_param(stmt1,1,id)
result=ibm_db.execute(stmt1)
product=ibm_db.fetch_assoc(stmt1)

print(product)
#Get form
form = ProductForm(request.form)

#populate product form fields
form.product_id.data = product['PRODUCT_ID']
form.product_cost.data =
str(product['PRODUCT_COST'])
form.product_num.data =
str(product['PRODUCT_NUM'])

if request.method == 'POST' and form.validate():
    product_id = request.form['product_id']
    product_cost = request.form['product_cost']
    product_num = request.form['product_num']

    sql2="UPDATE products SET
product_id=?,product_cost=?,product_num=?
WHERE product_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,product_id)
    ibm_db.bind_param(stmt2,2,product_cost)
    ibm_db.bind_param(stmt2,3,product_num)
    ibm_db.bind_param(stmt2,4,id)
    ibm_db.execute(stmt2)

    flash("Product Updated", "success")

    return redirect(url_for('products'))

    return render_template('edit_product.html',
form=form)

#Delete Product
@app.route('/delete_product/<string:id>',
methods=['POST'])
@is_logged_in
def delete_product(id):

```

```

        sql2="DELETE FROM products WHERE
product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,id)
        ibm_db.execute(stmt2)

        flash("Product Deleted", "success")

        return redirect(url_for('products'))

#Location Form Class
class LocationForm(Form):
    location_id = StringField('Location ID',
[validators.Length(min=1, max=200)])

#Add Location
@app.route('/add_location', methods=['GET',
'POST'])
@is_logged_in
def add_location():
    form = LocationForm(request.form)
    if request.method == 'POST' and form.validate():
        location_id = form.location_id.data

        sql2="INSERT into locations VALUES(?)"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,location_id)
        ibm_db.execute(stmt2)

        flash("Location Added", "success")

        return redirect(url_for('locations'))

    return render_template('add_location.html',
form=form)

#Edit Location
@app.route('/edit_location/<string:id>',
methods=['GET', 'POST'])
@is_logged_in
def edit_location(id):

```

```

    sql2="SELECT * FROM locations where
location_id= ?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,id)
    result=ibm_db.execute(stmt2)
    location=ibm_db.fetch_assoc(stmt2)
    #Get form
    form = LocationForm(request.form)
    print(location)

    #populate article form fields
    form.location_id.data =
location['LOCATION_ID']

    ifrequest.method == 'POST'andform.validate():
        location_id = request.form['location_id']

        sql2="UPDATE locations SET location_id=?
WHERE location_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,location_id)
        ibm_db.bind_param(stmt2,2,id)
        ibm_db.execute(stmt2)

        flash("Location Updated", "success")

        return redirect(url_for('locations'))

    returnrender_template('edit_location.html',
form=form)

#Delete Location
@app.route('/delete_location/<string:id>',
methods=['POST'])
@is_logged_in
defdelete_location(id):
    sql2="DELETE FROM locations WHERE
location_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,id)
    ibm_db.execute(stmt2)

    flash("Location Deleted", "success")

```

```

    return redirect(url_for('locations'))

#Product Movement Form Class
class ProductMovementForm(Form):
    from_location = SelectField('From Location',
    choices=[])
    to_location = SelectField('To Location',
    choices=[])
    product_id = SelectField('Product ID', choices=[])
    qty = IntegerField('Quantity')

class CustomError(Exception):
    pass

#Add Product Movement
@app.route('/add_product_movements',
methods=['GET', 'POST'])
@is_logged_in
def add_product_movements():
    form = ProductMovementForm(request.form)

    sql2="SELECT product_id FROM products"
    sql3="SELECT location_id FROM locations"
    stmt2 = ibm_db.prepare(conn, sql2)
    stmt3 = ibm_db.prepare(conn, sql3)

    result=ibm_db.execute(stmt2)
    ibm_db.execute(stmt3)

    products=[]
    row = ibm_db.fetch_assoc(stmt2)
    while(row):
        products.append(row)
        row = ibm_db.fetch_assoc(stmt2)
    products=tuple(products)

    locations=[]
    row2 = ibm_db.fetch_assoc(stmt3)
    while(row2):
        locations.append(row2)
        row2 = ibm_db.fetch_assoc(stmt3)
    locations=tuple(locations)

```

```

prods = []
for p in products:
    prods.append(list(p.values())[0])

locs = []
for i in locations:
    locs.append(list(i.values())[0])

form.from_location.choices = [(l,l) for l in locs]
form.from_location.choices.append(("Main
Inventory", "Main Inventory"))
form.to_location.choices = [(l,l) for l in locs]
form.to_location.choices.append(("Main
Inventory", "Main Inventory"))
form.product_id.choices = [(p,p) for p in prods]

if request.method == 'POST' and form.validate():
    from_location = form.from_location.data
    to_location = form.to_location.data
    product_id = form.product_id.data
    qty = form.qty.data

    if from_location == to_location:
        raise CustomError("Please Give different
From and To Locations!!")

    elif from_location == "Main Inventory":
        sql2 = "SELECT * from product_balance
where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2, 1, to_location)
        ibm_db.bind_param(stmt2, 2, product_id)
        result = ibm_db.execute(stmt2)
        result = ibm_db.fetch_assoc(stmt2)
        print("-----")
        print(result)
        print("-----")
        app.logger.info(result)
        if result != False:
            if len(result) > 0:
                Quantity = result["QTY"]
                q = Quantity + qty

```

```
        sql2="UPDATE product_balance set  
qty=? where location_id=? and product_id=?"  
        stmt2 = ibm_db.prepare(conn, sql2)  
        ibm_db.bind_param(stmt2,1,q)
```

```
ibm_db.bind_param(stmt2,2,to_location)
```

```
ibm_db.bind_param(stmt2,3,product_id)  
ibm_db.execute(stmt2)
```

```
        sql2="INSERT into  
productmovements(from_location, to_location,  
product_id, qty) VALUES(?, ?, ?, ?)"  
        stmt2 = ibm_db.prepare(conn, sql2)
```

```
ibm_db.bind_param(stmt2,1,from_location)
```

```
ibm_db.bind_param(stmt2,2,to_location)
```

```
ibm_db.bind_param(stmt2,3,product_id)  
ibm_db.bind_param(stmt2,4,qty)  
ibm_db.execute(stmt2)
```

else:

```
        sql2="INSERT into  
product_balance(product_id, location_id, qty)  
values(?, ?, ?)"  
        stmt2 = ibm_db.prepare(conn, sql2)  
        ibm_db.bind_param(stmt2,1,product_id)  
        ibm_db.bind_param(stmt2,2,to_location)  
        ibm_db.bind_param(stmt2,3,qty)  
        ibm_db.execute(stmt2)
```

```
        sql2="INSERT into  
productmovements(from_location, to_location,  
product_id, qty) VALUES(?, ?, ?, ?)"  
        stmt2 = ibm_db.prepare(conn, sql2)
```

```
ibm_db.bind_param(stmt2,1,from_location)  
ibm_db.bind_param(stmt2,2,to_location)  
ibm_db.bind_param(stmt2,3,product_id)  
ibm_db.bind_param(stmt2,4,qty)
```

```

        ibm_db.execute(stmt2)

        sql = "select product_num from products
where product_id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,product_id)
        current_num=ibm_db.execute(stmt)
        current_num = ibm_db.fetch_assoc(stmt)

        sql2="Update products set product_num=?
where product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)

        ibm_db.bind_param(stmt2,1,current_num['PRODU
CT_NUM']-qty)
        ibm_db.bind_param(stmt2,2,product_id)
        ibm_db.execute(stmt2)

        alert_num=current_num['PRODUCT_NUM']-qty

        if(alert_num<=0):
            alert("Please update the quantity of the
product { }, Atleast { } number of pieces must be
added to finish the pending Product
Movements!".format(product_id,-alert_num))

        elif to_location=="Main Inventory":
            sql2="SELECT * from product_balance
where location_id=? and product_id=?"
            stmt2 = ibm_db.prepare(conn, sql2)
            ibm_db.bind_param(stmt2,1,from_location)
            ibm_db.bind_param(stmt2,2,product_id)
            result=ibm_db.execute(stmt2)
            result=ibm_db.fetch_assoc(stmt2)

            app.logger.info(result)
            if result!=False:
                if len(result)>0:
                    Quantity = result["QTY"]
                    q = Quantity - qty

```



```
        sql2="UPDATE product_balance set
qty=? where location_id=? and product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
        ibm_db.bind_param(stmt2,1,q)
```

```
ibm_db.bind_param(stmt2,2,to_location)
```

```
ibm_db.bind_param(stmt2,3,product_id)
ibm_db.execute(stmt2)
```

```
        sql2="INSERT into
productmovements(from_location, to_location,
product_id, qty) VALUES(?, ?, ?, ?)"
        stmt2 = ibm_db.prepare(conn, sql2)
```

```
ibm_db.bind_param(stmt2,1,from_location)
```

```
ibm_db.bind_param(stmt2,2,to_location)
```

```
ibm_db.bind_param(stmt2,3,product_id)
ibm_db.bind_param(stmt2,4,qty)
ibm_db.execute(stmt2)
```

```
flash("Product Movement Added",
"success")
```

```
        sql = "select product_num from
products where product_id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,product_id)
        current_num=ibm_db.execute(stmt)
        current_num =
ibm_db.fetch_assoc(stmt)
```

```
        sql2="Update products set
product_num=? where product_id=?"
        stmt2 = ibm_db.prepare(conn, sql2)
```

```
ibm_db.bind_param(stmt2,1,current_num['PRODU
CT_NUM']+qty)
```

```
ibm_db.bind_param(stmt2,2,product_id)
ibm_db.execute(stmt2)
```

```

        alert_num=q
        if(alert_num<=0):
            alert("Please Add { } number of { } to
{ } warehouse!".format(-
q,product_id,from_location))
        else:
            raiseCustomError("There is no product
named { } in { }.".format(product_id,from_location))

```

else: #will be executed if both from_location
and to_locationare specified

```

        f=0
        sql = "SELECT * from product_balance
where location_id=? and product_id=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,from_location)
        ibm_db.bind_param(stmt,2,product_id)
        result=ibm_db.execute(stmt)
        result = ibm_db.fetch_assoc(stmt)

```

```

        ifresult!=False:
            if(len(result))>0:
                Quantity = result["QTY"]
                q = Quantity - qty

```

```

                sql2="UPDATE product_balance set
qty=? where location_id=? and product_id=?"
                stmt2 = ibm_db.prepare(conn, sql2)
                ibm_db.bind_param(stmt2,1,q)

```

```

        ibm_db.bind_param(stmt2,2,from_location)

```

```

        ibm_db.bind_param(stmt2,3,product_id)
        ibm_db.execute(stmt2)
        f=1

```

```

        alert_num=q
        if(alert_num<=0):
            alert("Please Add { } number of { } to
{ } warehouse!".format(-
q,product_id,from_location))

```

```

else:
    raiseCustomError("There is no product
named { } in { }.".format(product_id,from_location))

if(f==1):
    sql = "SELECT * from product_balance
where location_id=? and product_id=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,to_location)
    ibm_db.bind_param(stmt,2,product_id)
    result=ibm_db.execute(stmt)
    result = ibm_db.fetch_assoc(stmt)

    ifresult!=False:
        if(len(result))>0:
            Quantity = result["QTY"]
            q = Quantity + qty

            sql2="UPDATE product_balance set
qty=? where location_id=? and product_id=?"
            stmt2 = ibm_db.prepare(conn, sql2)
            ibm_db.bind_param(stmt2,1,q)

ibm_db.bind_param(stmt2,2,to_location)

ibm_db.bind_param(stmt2,3,product_id)
ibm_db.execute(stmt2)

else:

    sql2="INSERT into
product_balance(product_id, location_id, qty)
values(?, ?, ?)"
    stmt2 = ibm_db.prepare(conn, sql2)

ibm_db.bind_param(stmt2,1,product_id)

ibm_db.bind_param(stmt2,2,to_location)
ibm_db.bind_param(stmt2,3,qty)
ibm_db.execute(stmt2)

```

```

        sql2="INSERT into
productmovements(from_location, to_location,
product_id, qty) VALUES(?, ?, ?, ?)"
        stmt2 = ibm_db.prepare(conn, sql2)

ibm_db.bind_param(stmt2,1,from_location)
        ibm_db.bind_param(stmt2,2,to_location)
        ibm_db.bind_param(stmt2,3,product_id)
        ibm_db.bind_param(stmt2,4,qty)
        ibm_db.execute(stmt2)

        flash("Product Movement Added",
"success")

        render_template('products.html',form=form)

        return redirect(url_for('product_movements'))

returnrender_template('add_product_movements.ht
ml', form=form)

#Delete Product Movements
@app.route('/delete_product_movements/<string:id
>', methods=['POST'])
@is_logged_in
defdelete_product_movements(id):

    sql2="DELETE FROM productmovements
WHERE movement_id=?"
    stmt2 = ibm_db.prepare(conn, sql2)
    ibm_db.bind_param(stmt2,1,id)
    ibm_db.execute(stmt2)

    flash("Product Movement Deleted", "success")

    return redirect(url_for('product_movements'))

if __name__ == '__main__':
    app.secret_key = "secret123"
    #when the debug mode is on, we do not need to
    restart the server again and again
    app.run(debug=True)

```

GitHub and Project demo link:

GitHub link : <https://github.com/IBM-EPBL/IBMProject-39093-1660394652>

Deployed application link: <http://169.51.203.125:32155/>