**HX8001- PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTERPRENEURSHIP**

**INVENTORY MANAGEMENT SYSTEM FOR RETAILERS**

**TEAM ID: PNT2022TMID36552**

**TEAM MEMBERS**

**B.ARAVIND**

**M.SAKTHI PONNAPPAN**

**V.JAGADESH**

**M.SARAN**

**In partial fulfilment for the award of the degree**

**Of**

**BACHELOR OF ENGINEERING**

**In**

**COMPUTER SCIENCE AND ENGINEERING**

**ST PETER'S COLLEGE OF ENGINEERING AND TECHNOLOGY**

**ANNA UNIVERSITY: CHENNAI 600 025**

**NOV-DEC 2022**

**TABLE OF CONTENTS**

**1. INTRODUCTION**

1.1 Project Overview

1.2 Purpose

**2. LITERATURE SURVEY**

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

**3. IDEATION & PROPOSED SOLUTION**

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

**4. REQUIREMENT ANALYSIS**

4.1 Functional requirement

4.2 Non-Functional requirements

**5. PROJECT DESIGN**

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

**6. PROJECT PLANNING & SCHEDULING**

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

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

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

**8. TESTING**

8.1 Test Cases

8.2 User Acceptance Testing

**9. RESULTS**

9.1 Performance Metrics

**10. ADVANTAGES & DISADVANTAGES**

**11. CONCLUSION**

**12. FUTURE SCOPE**

**13. APPENDIX**

13.1 Source Code

13.2 GitHub & Project Demo Link

**CHAPTER 1**

**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 in order to run its business in manual system.

**1.1 PROJECT OVERVIEW**

Inventory management system is an application which is helpful for business operate. It is a cloud based web application that is specifically implemented to make the lives of warehouse workers much easier. It is an inventory management system for all the retailers out there in the market where they can manage, add and delete and track their goods that are being imported and exported through all locations. By managing inventory, retailers meet customer demand without running out of stock or carrying excess supply. This results in lower costs and gives them a better understanding on sales patterns. Inventory management is a challenging problem area in supply chain management. Companies need to have inventories in warehousesin 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

**1.2 PURPOSE**

The purpose is to help retailers track and manage stocks related to their own products. The system will ask the retailers to create their accounts by providing essential details. Once retailers login successfully into 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 their inventory whenever they wish and we have used Send Grid email service which sends an alert to retailers through email If there is no stock found in their accounts. And they can order new stock at that time.

**CHAPTER 2**

**2. LITERATURE SURVEY**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S,NO** | **TITLE** | **AUTHOR** | **YEAR** | **ABSTRACT** |
| 1 | Realignment of the physical distribution process in omni-channel fulfillment | Rafay Ishfaq, C. Clifford Defee and Brian J. Gibson | 2015 | The purpose of this paper is to identify the realignment of the physical distribution process for store-based retailers in their efforts to integrate the online channel into their business model. |
| 2 | Demand uncertainty and inventory turnover performance | GülşahHançerlioğulları, AlperŞen, EsraAğcaAktunç | 2016 | Different fashion brands have different inventory service targets. To achieve the desirable inventory service level without the need of holding a lot of inventory, fashion companies commonly adopt the quick response practice. |
| 3 | Operations management in multi-channel retailing: an exploratory study | Alexander Hübner, Andreas Holzapfel, Heinrich Kuhn | 2015 | Multi-channel (MC) shopping is revolutionizing retail operations. For traditional retailers, the growing importance of online sales means creating new supply chain models. This requires a solid understanding of the operations processes. |
| 4 | Inventory Service Target in Quick Response Fashion Retail Supply Chains | Tsan-Ming Cho | 2016 | Different fashion brands have different inventory service targets. To achieve the desirable inventory service level without the need of holding a lot of inventory. |
| 5 | Managing Inventory Systems of Slow-Moving Items | G.J. Hahn, A. Leucht | 2015 | Slow-moving demand patterns frequently occur with spare parts as well as items in decentralized retail supply chains with large assortments. These patterns are commonly called lumpy since they exhibit comparably high demand variation and a high fraction of zero-demand events. |

**2.1 EXISTING PROBLEM**

Products are considered as the businessresources 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 came up with 2Inventory Management System existing in the market, 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 otherinventory management system we decided to include some of common and key features that should be included in every inventory management system.

**2.2 REFERENCE**

[https://www.camcode.com/asset-tags/what-is-an-inventory-management-](https://www.camcode.com/asset-tags/what-is-an-inventory-management-system/) [system/](https://www.camcode.com/asset-tags/what-is-an-inventory-management-system/)

* Jimmy Wales, onlineencyclopediaWiKipedia ,

[http://www.wikipedia.org](http://www.wikipedia.org/)

* James Gosling. Java (Programming Language) ,

[http://www.java.com](http://www.java.com/)

* Names Allaire, Netbeans-Fully-featured Java IDE,

[http://www.netbeans.org](http://www.netbeans.org/)

* how-to feature and columns by Java expert;news; Java applets;sample code ; tips ,

[http://www.javaworld.com](http://www.javaworld.com/)

**2.3 PROBLEM STATEMENT DEFINITION**

**Domain:** Cloud Application Development

**Title:** Inventory Management For Retailer

**1.WHO DOES THE PROBLEM AFFECT?**

✓ Lack of Centralised Inventory Hub

✓ Stocktaking becomes very challenging when you have inventories in multiple

locations.

**2.WHAT IS THE ISSUE?**

✓ Poor or mismanagement of resources

✓ Lack of cash flow ✓ Lack of tracking

✓ Time delay 3.WHAT IS THE IMPACT OF THE ISSUE?

✓ Managing warehouse spaces

✓ Inventory loss and overstocking

✓ Delay in delivering of orders

**4.WHAT WOULD HAPPEN IF WE DIDN’T SOLVE THE PROBLEM?**

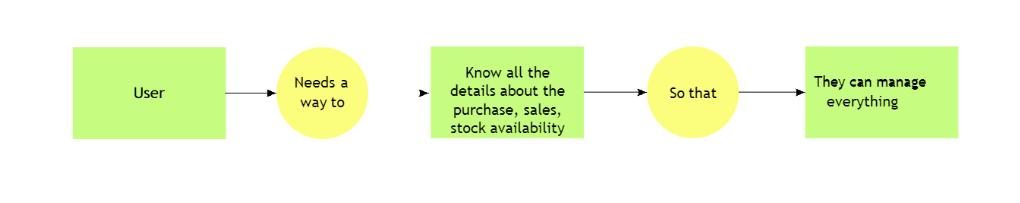
Discrete stock data from various locations makes shipping complex , resulting in delays. It’s one of the biggest and continual challenges faced by most businesses today.

**5.WHAT WOULD HAPPEN WHEN IT IS FIXED?**

You can significantly reduce expenses and save a great deal of time by simply creating centralised inventory hub for your inventory-related data, including stock-taking.

**6.WHY IS IT IMPORTANT THAT WE FIX THE PROBLEM?**

This gives you comprehensive visibility and control of inventory and data in one single location making stock management simple.It also becomes much easier to track the inventory that enters and leaves your business premises.



|  |  |
| --- | --- |
| 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 movementof specific items. When a company wants a dynamic and systematic system to record and keep their inventory data. |

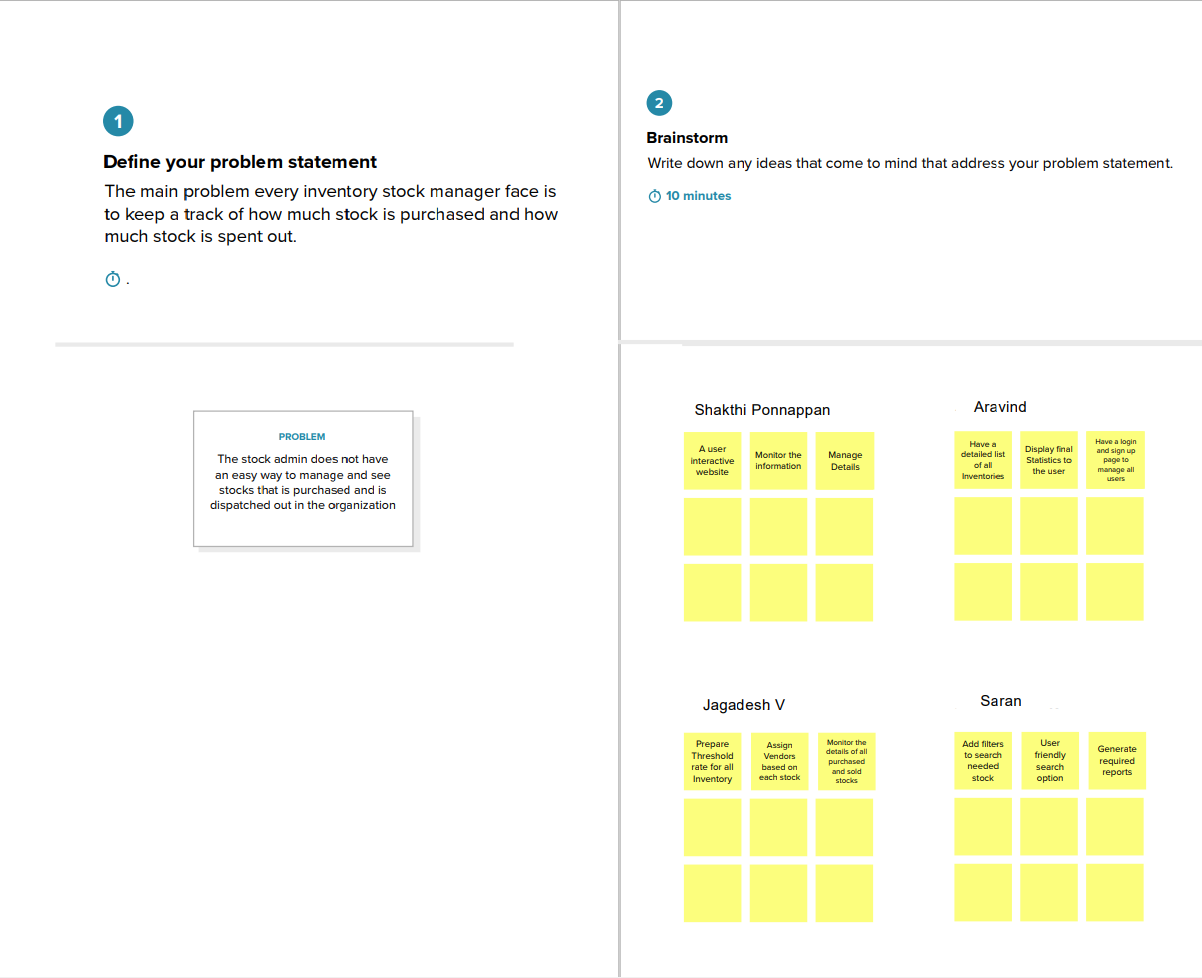
**CHAPTER 3**

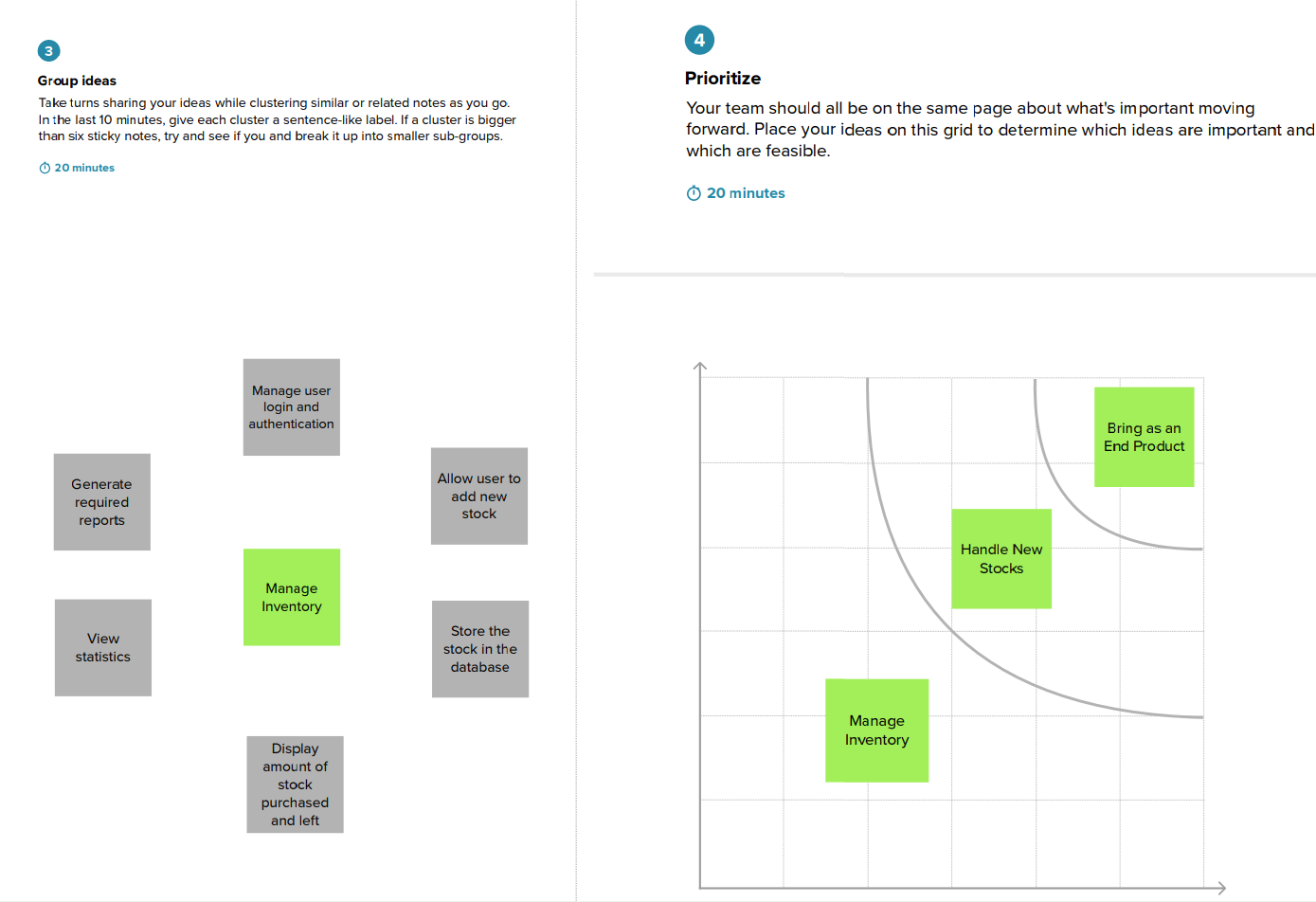
**3. IDEATION & PROPOSED SOLUTION**

**3.1 EMPATHY MAP CANVAS**

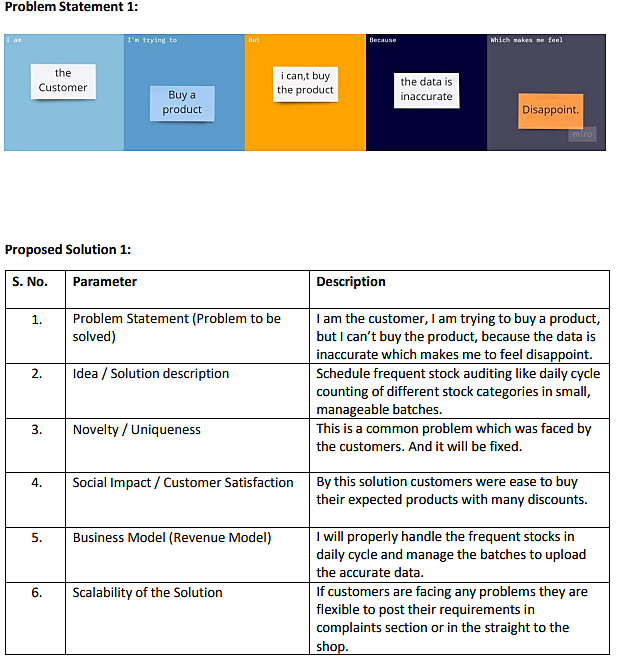


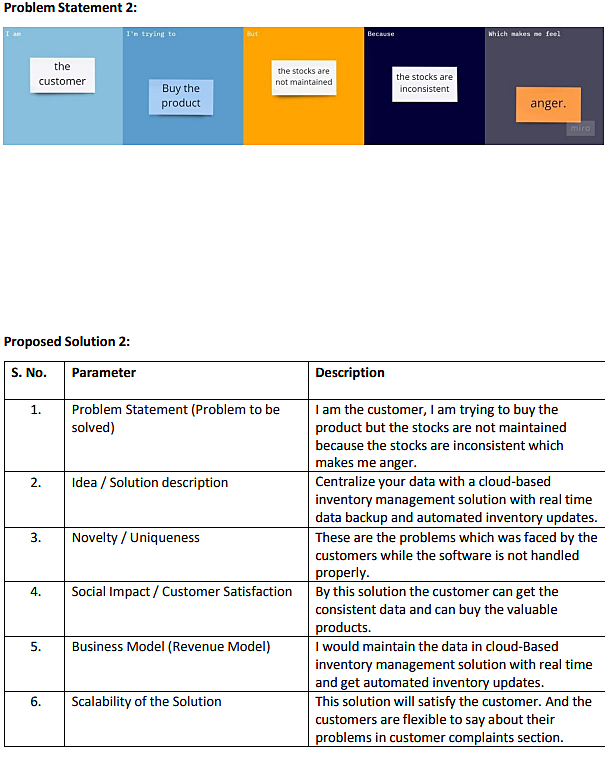
**3.2 IDEATION & BRAINSTORMING**

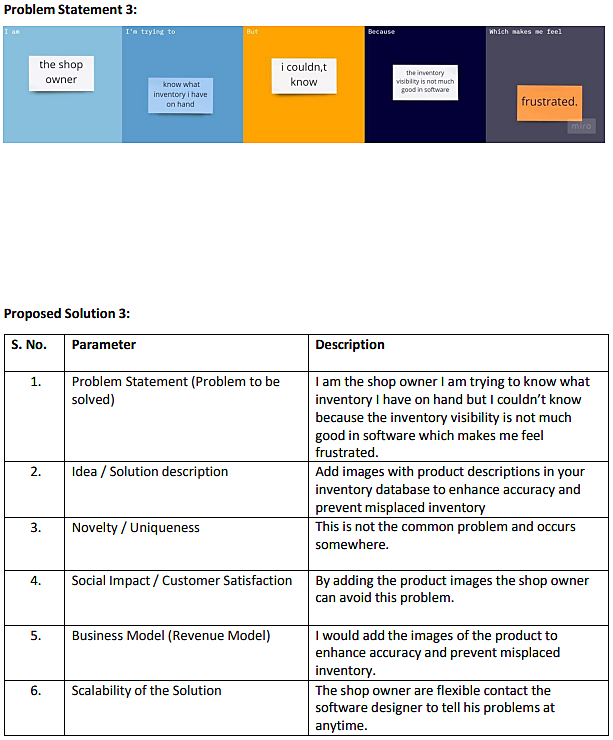


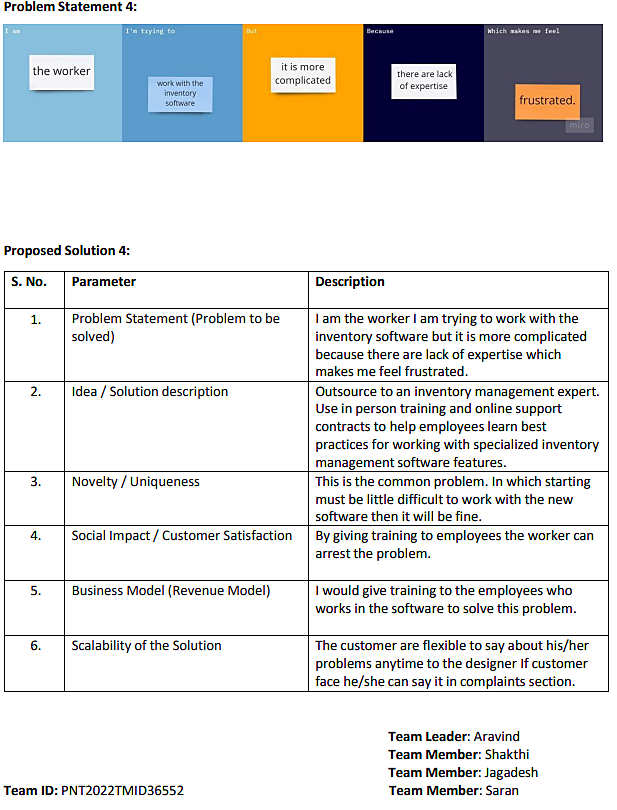


**3.3 PROPOSED SOLUTION**









**3.4 PROBLEM SOLUTOION FIT**



**CHAPTER 4**

**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 Gmail  Registration through LinkedIN |
| FR-2 | User Confirmation | Confirmation via Email  Confirmation via OTP |
| FR-3 | Business regulations | Many needs may fit under this category |
| FR-4 | Product management | Easily track product information  Quickly produce reports for single or multiple sold products |
| FR-5 | Audit Monitoring | The technique of tracking crucial data is known as audit tracking |
| FR-6 | Historical Data | Specify the amount of storage you need to handle this expansion |

**4.2 NON- FUNCTIONAL REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | Backups for database are available |
| NFR-2 | **Reliability** | The security requirements deal with the primary security. only authorized users can access the system with user name and password of administrator |
| NFR-3 | **Reliability** | The software will not be able to connect to the database in the event of the server being down due to a hardware or software failure |
| NFR-4 | **Performance** | Easy tracking of records and updating can be done . |
| NFR-5 | **Availability** | The software will be available only to administrator of the organization and the product as well as customer details will be recorded by him. He can add customers, Update and delete them as well as add new products and manage them |
| NFR-6 | **Scalability** | The ability of a system to handle a growing amount of work |

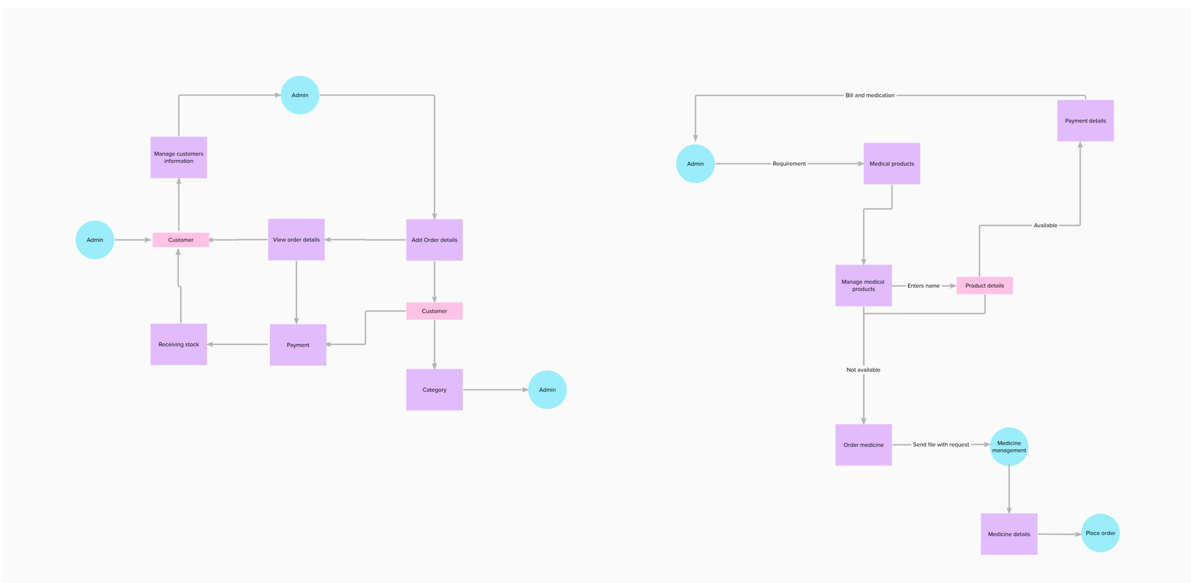
**CHAPTER 5**

**5. PROJECT DESIGN**

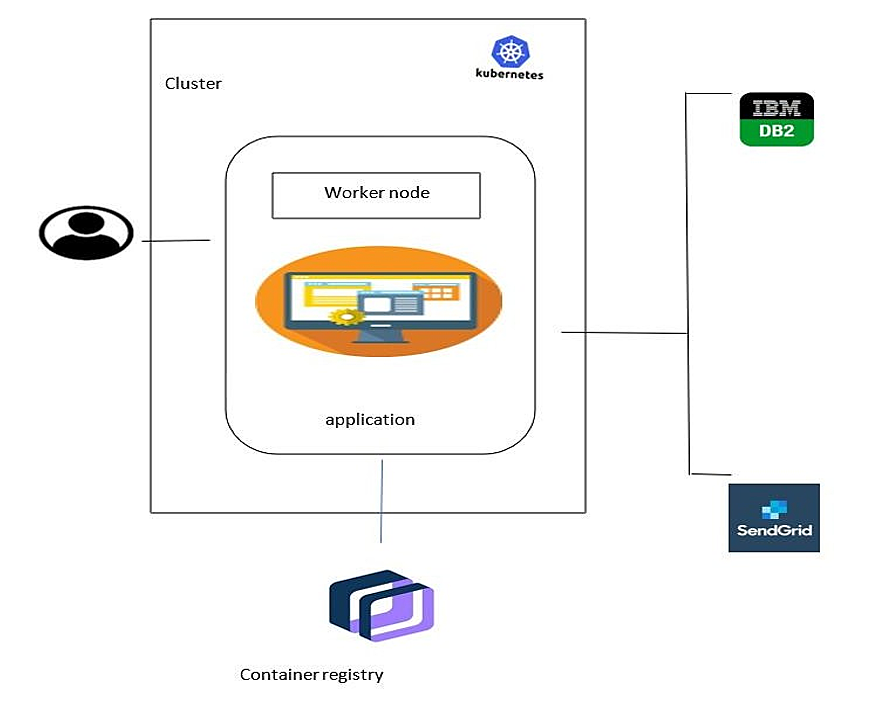
**5.1 DATA FLOW DIAGRAMS**

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

**Example:** [**(Simplified)**](https://developer.ibm.com/patterns/visualize-unstructured-text/)



**5.2 SOLUTION & TECHNICAL ARCHITECTURE**



# SOFTWARES USED IN TECHNICALARCHITECTURE AND ITS PURPOSE:

**Kubernetes cluster:**

A Kubernetes cluster is a set of nodes that run containerized applications. Containerizing applications packages an app with its dependences and some necessary services. They are more lightweight and flexible than virtual machines.

**Container Registry:**

Container Registry is a single place for your team to manage Docker images, perform vulnerability analysis, and decide who can access what with fine- grained access control. Existing CI/CD integrations let you set up fully automated Docker pipelines to get fast feedback.

**IBM DB2:**

Db2, or Database 2, is a set of relational database products built and offeredby IBM. DB2 is designed to store, analyze and retrieve the data efficiently. In this project, it stores the customers data.

**Sendgrid:**

SendGrid is a cloud-based SMTP provider that allows you to send email without having to maintain email servers. SendGrid manages all of the technical details, from scaling the infrastructure to ISP outreach and reputation monitoring to whitelist services and real time analytics. It is a cloud-based email marketing tool that assistsmarketers and developers with campaign management and audience engagement. In this project, it sends the email alert if there is no stock.

**5.3 USER STORIES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 confirmation email once I have registered for the application | I can receive confirmation email & click confirm | High | Sprint-1 |
| USN-3 | As a user, I can register for the application through Facebook | I can register & access the dashboard with Facebook Login | Low | Sprint-2 |
| USN-4 | As a user, I can register for the application through Gmail | I can register & application  Through Gmail | Medium | Sprint-1 |
| Login | USN-5 | As a user, I can log into the application by entering email & password | I can access my account | High | Sprint-1 |
| Dashboard | USN-6 | As a user,i can log into my account for the mobile | I can access my account /Dashboard | High | Sprint-1 |
| Customer (Web user) | Registration | USN-7 | 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-8 | As a user, I will receive confirmation email once I have registered for the application | I can receive confirmation email & click confirm | High | Sprint-1 |
| USN-9 | As a user, I can register for the application through Facebook | I can register & access the dashboard with Facebook Login | Low | Sprint-2 |
| USN-10 | As a user ,I can upload a Profile photo and add my name to my account | I can upload my Profile photo/Name in my account | Medium | Sprint-1 |
| Customer Care Executive | Customer Support | USN-11 | As a user,I can support for customers to handle queries and complaints from their customers | I can support for customers to clear complaints | High | Sprint-1 |
| Administrator | Responsibility | USN-12 | As a system administrator I want to be able to add new users when required so that | I Can add new users | High | Sprint -1 |

**CHAPTER 6**

**6. PROJECT PLANNING & SCHEDULING**

**6.1 SPRINT PLANNING & ESTIMATION**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 5 | High | Aravind ShakthiJagadesh Saran |
| Sprint-2 |  | USN-2 | As a user, I will receiveconfirmation email onceI have registered for the application | 3 | Medium | Aravind Shakthi Jagadesh  Saran |
| Sprint-4 |  | USN-3 | As a user, I can register for the application through Facebook | 8 | Low | Aravind ShakthiJagadesh  Saran |
| Sprint-3 |  | USN-4 | As a user, I can register for the application through Gmail | 8 | High | Aravind ShakthiJagadesh  Saran |
| Sprint-1 | Login | USN-5 | As a user, I can log into theapplication by entering email & password | 5 | High | Aravind Shakthi  Jagadesh Saran |
| Sprint-2 |  | USN-4 | As a user, I can logininto the application through Google one Tap Sign in | 3 | Medium | Aravind ShakthiJagadesh Saran |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Dashboard | USN-5 | As a user, I must be able to see my details on  the dashboard. | 3 | High | Aravind Shakthi  Jagadesh Saran |
| Sprint-2 |  | USN-6 | As a user, I should be able to change password whenever I prefer. | 2 | Medium | Aravind ShakthiJagadesh  Saran |
| Sprint-1 | Inventory | USN-7 | As a retailer, I should be able to alter productdetails in the app | 2 | Medium | Aravind ShakthiJagadesh  Saran |
| Sprint-2 |  | USN-8 | As a retailer, I should be able to add or remove quantity of products in the app. | 3 | Medium | Aravind Shakthi  Jagadesh Saran |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| Sprint-3 |  | USN-9 | As a retailer, I should get alert on stock shortage or unavailability. | 5 | Medium | Aravind Shakthi Jagadesh Saran |
| Sprint-1 | Order | USN-7 | As a user, I should be able to order items on the app | 2 | High | Aravind Shakthi Jagadesh  Saran |
| Sprint-2 |  | USN-8 | As a user, I should be able to verifyand pay in a secure payment gateway | 3 | High | Aravind Shakthi  Jagadesh Saran |
| Sprint-3 |  | USN-9 | As a user, I should be able to get the producton time. | 5 | Low | Aravind Shakthi Jagadesh  Saran |
| Sprint-1 | Maintenance | USN-1 | As a administrator, I should be able to edit details of the users of the app. | 8 | High | Aravind ShakthiJagadesh  Saran |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-2 |  | USN-2 | Termination user accounts temporarily or permanently if needed. | 5 | Low | Aravind Shakthi  Jagadesh Saran |
| Sprint-1 | Feedback | USN-1 | As a customer careteam member, I should be able to getfeedback from the users. | 2 | High | Aravind ShakthiJagadesh  Saran |
| Sprint-2 |  | USN-2 | As a customer care team member, I should beavailable 24/7 to increase customer base | 8 | Medium | Aravind Shakthi  Jagadesh Saran |

**6.2 SPRINT DELIVERY SCHEDULE**

**Project Tracker, Velocity &Burndown Chart: (4 Marks)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 | 18 | 6 Days | 31 Oct 2022 | 05 Nov 2022 | 18 | 05 Nov 2022 |
| Sprint-3 | 21 | 6 Days | 07 Nov 2022 | 12 Nov 2022 | 21 | 12 Nov 2022 |
| Sprint-4 | 21 | 6 Days | 14 Nov 2022 | 19 Nov 2022 | 21 | 19 Nov 2022 |

# Velocity:

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

# 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](https://www.visual-paradigm.com/scrum/what-is-agile-software-development/) methodologies suchas [Scrum](https://www.visual-paradigm.com/scrum/scrum-in-3-minutes/). However, burn down charts can be applied to any project containing measurable progress over time.

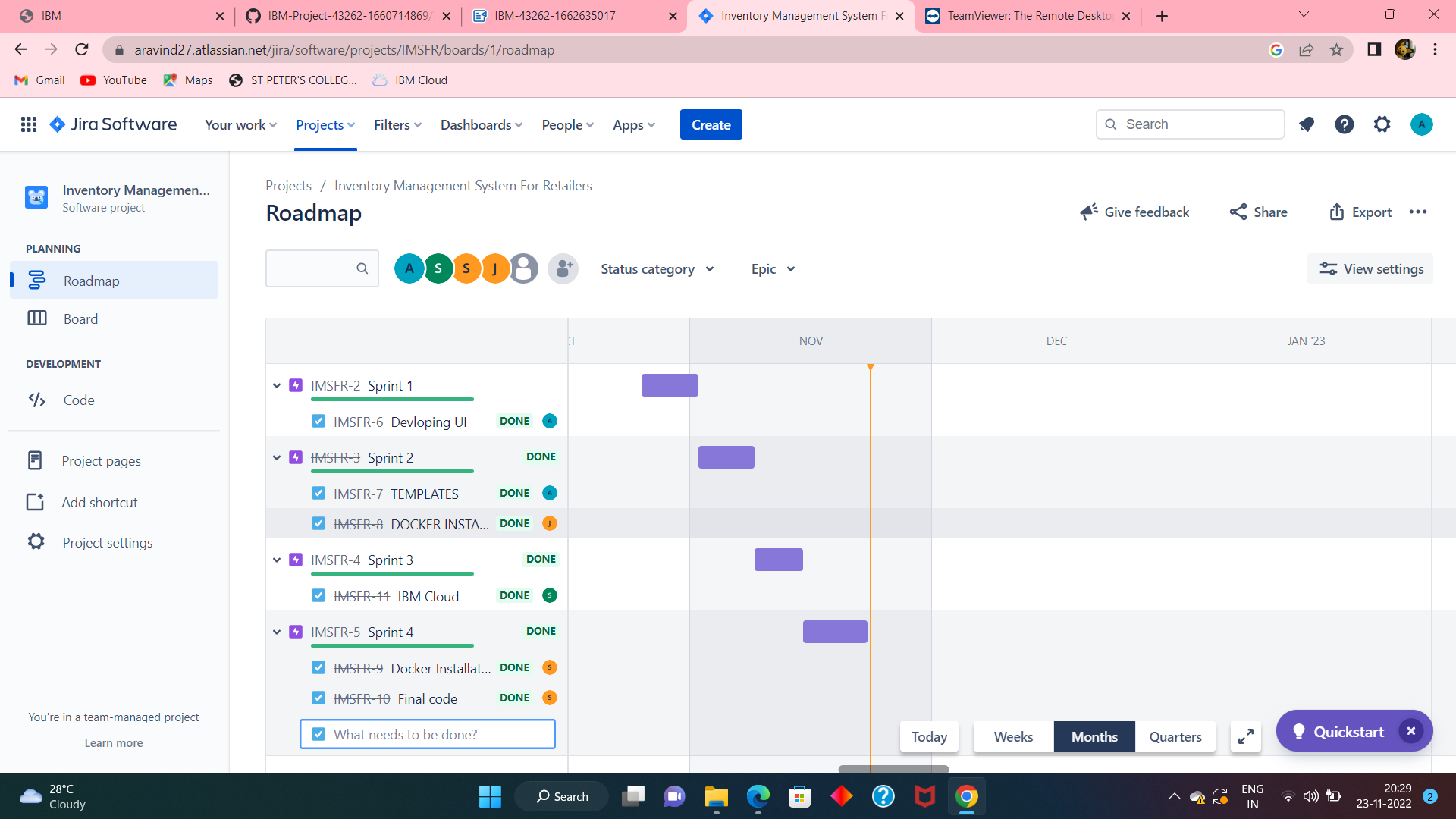
# Reference:

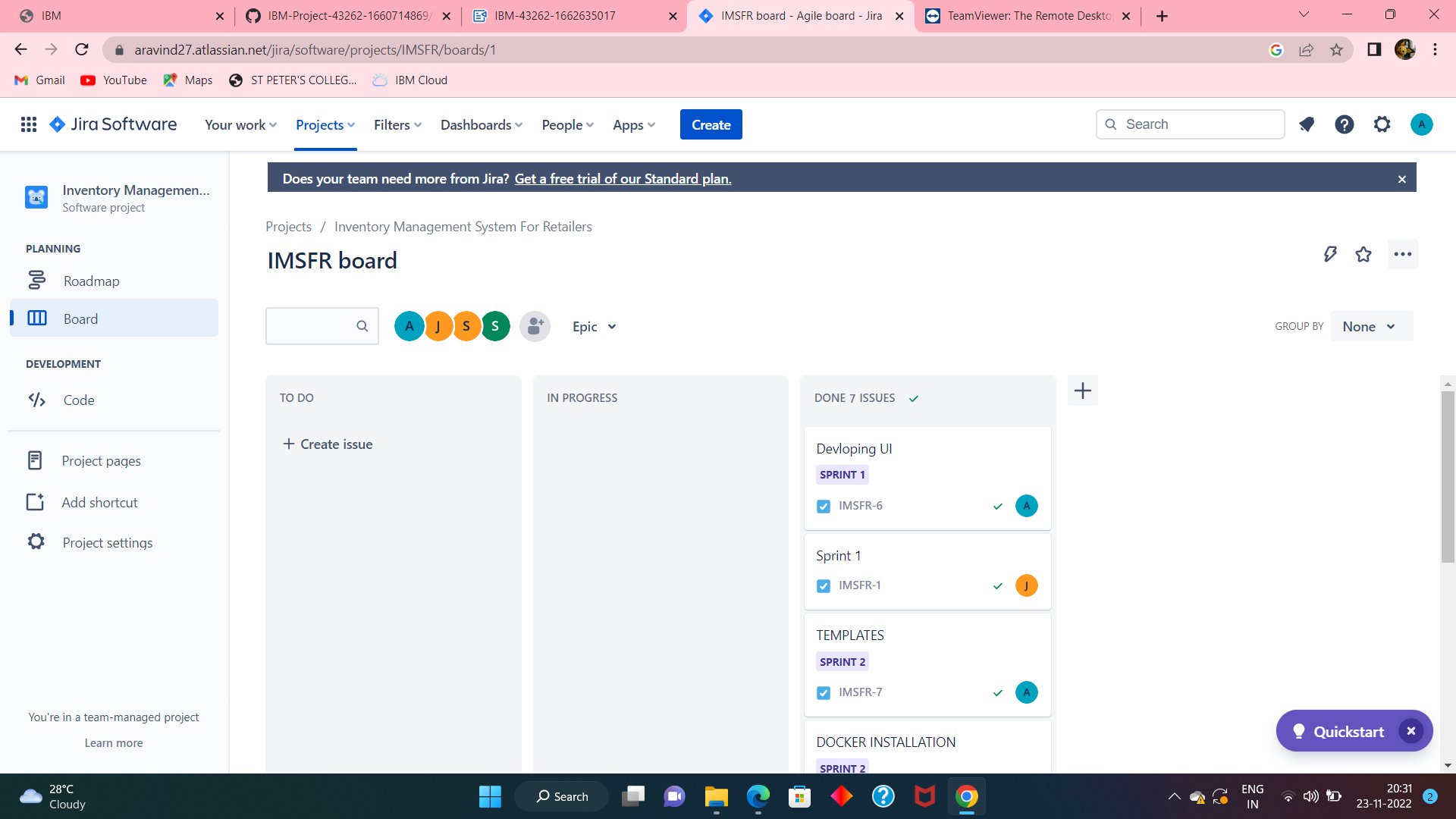
<https://www.atlassian.com/agile/project-management> <https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software>

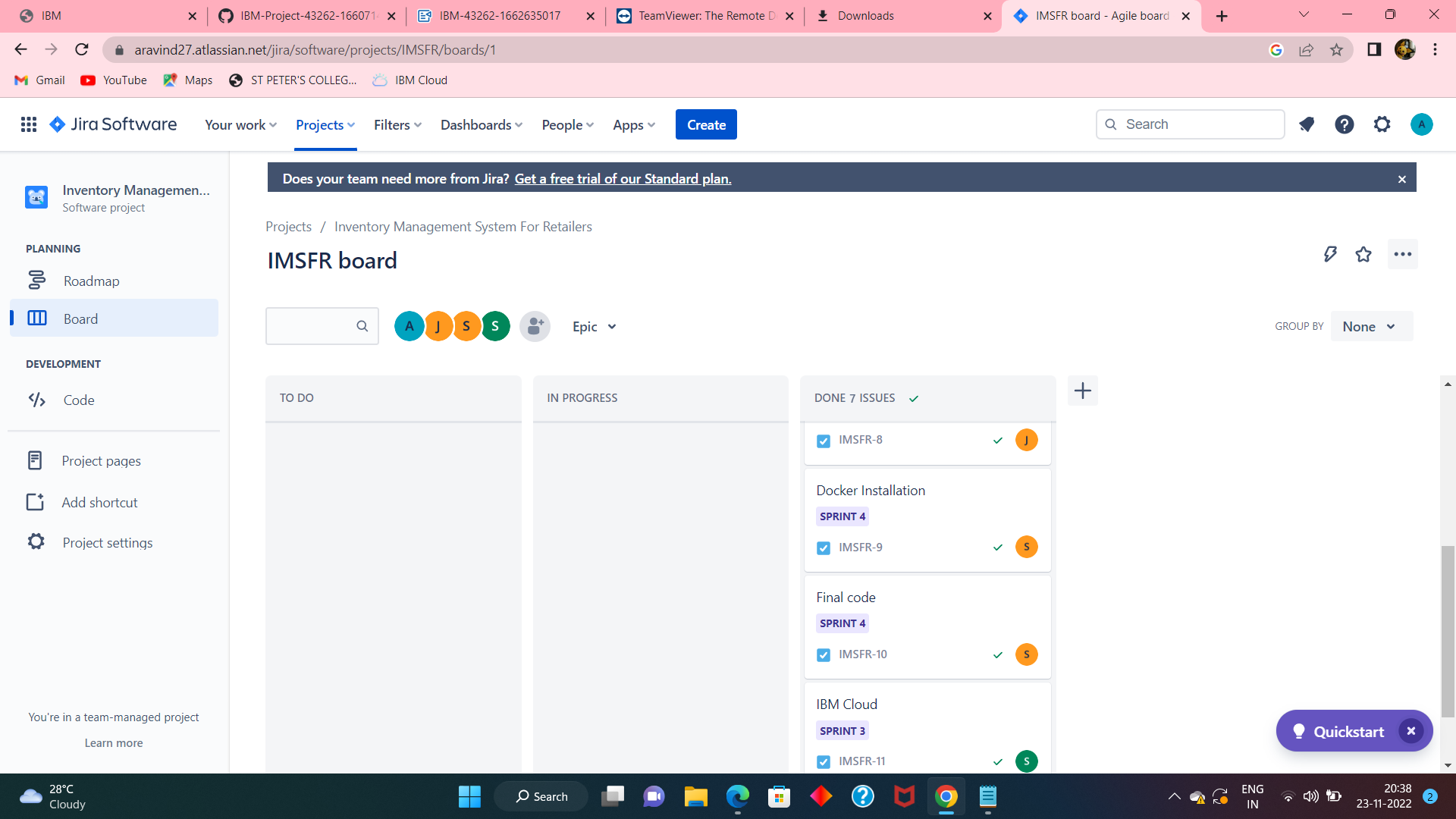
**6.3 REPORTS FROM JIRA**

IT organization have the challenge of ensuring system uptime,supporting users, and managing inventory of both hardware and software. ITteams gain significant efficiencies when one tool can support multiple businessoperations. According to Gartner, mastering the discipline of effective asset

management is a huge cost savings for companies







**CHAPTER 7**

**7. CODING & SOLUTIONING**

**7.1 FEATURE 1**

We have added the data visualization methods for expenditure. The pie chart have been used to represent the monthly expenses. The pie chart is a pictorial representation of data that makes it possible to visualize the relationships between the parts and the whole of a variable. For example, it is possible to understand the industry count or percentage of a variable level from the division by areas or sectors. The recommended use for pie charts is two-dimensional, as three-dimensional use can be confusing. The dimensions form sectors of the measurement values; they can have one or two sizes and up to two measures. The first dimension is used to define the angle of each sector that makes up the chart and the second dimension optionally determines the radius of each sector. Additionally, these plots are useful for comparing data over a fixed period since they do not show changes over time.

Therefore, their use should be considered if:

● You are looking to categorize and compare a set of data.

● You only have positive values.

● You have less than seven categories since a larger number can make it difficult to perceive each segment

**Register.html**

<h1>Register</h1>

{% from "includes/\_formhelpers.html" import render\_field %}

<form method="POST" action="">

<div class="form-group">

{{render\_field(form.name, class\_="form-control")}}

</div>

<div class="form-group">

{{render\_field(form.email, class\_="form-control")}}

</div>

<div class="form-group">

{{render\_field(form.username, class\_="form-control")}}

</div>

<div class="form-group">

{{render\_field(form.password, class\_="form-control")}}

</div>

<div class="form-group">

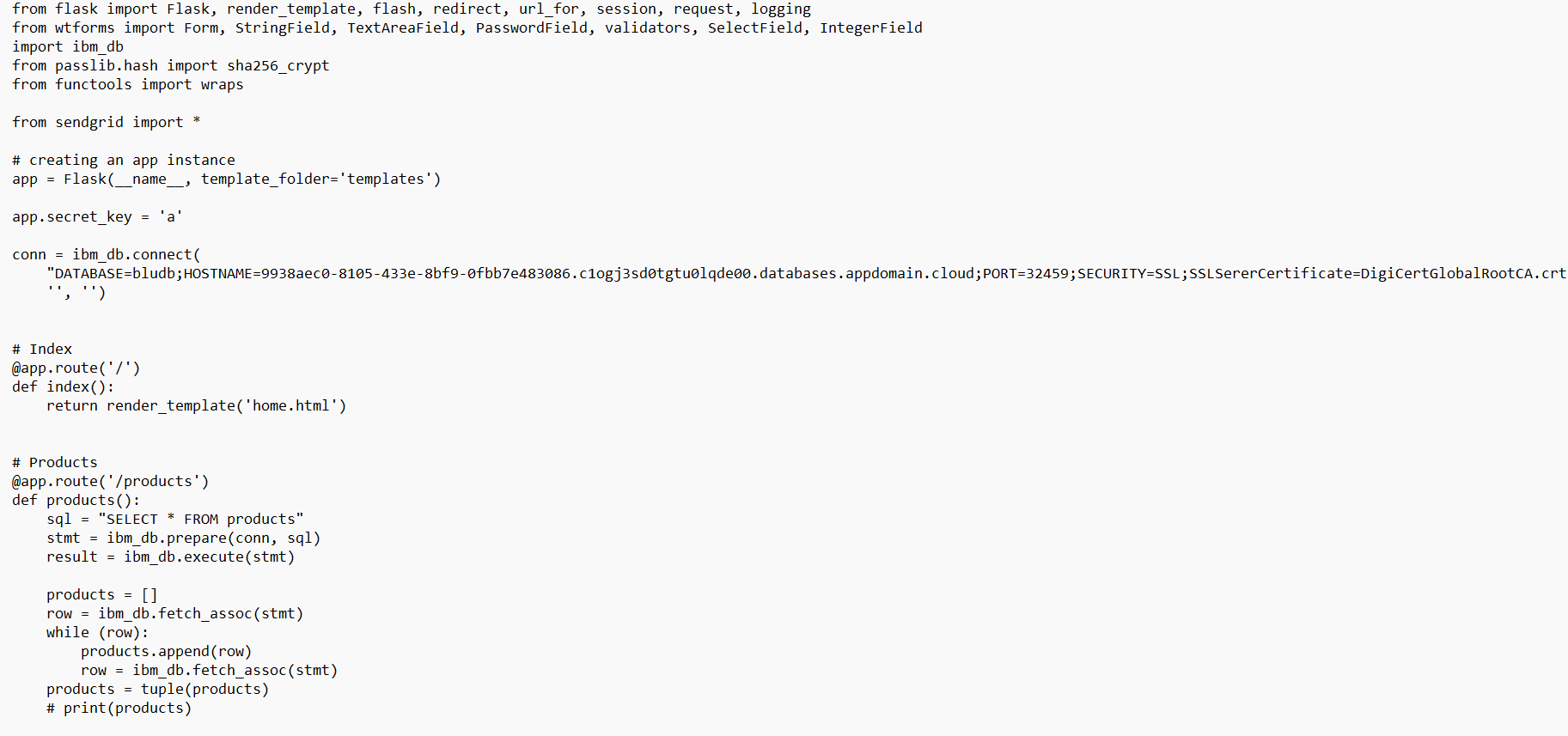
{{render\_field(form.confirm, class\_="form-control")}}

</div>

<p><button type="submit" class="btn btn-primary" value="Submit">Submit</button></p>

</form>

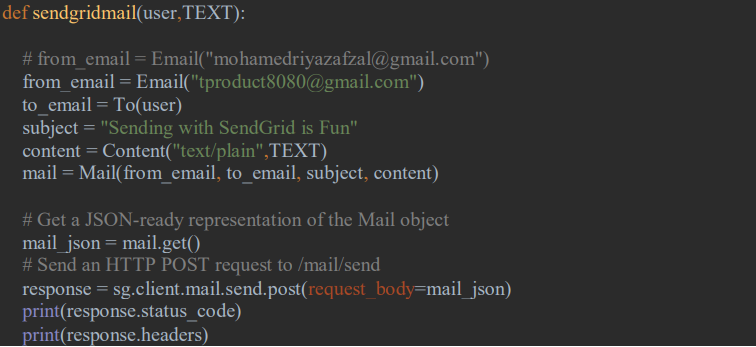
**App.py**



**7.2. FEATURE 2**

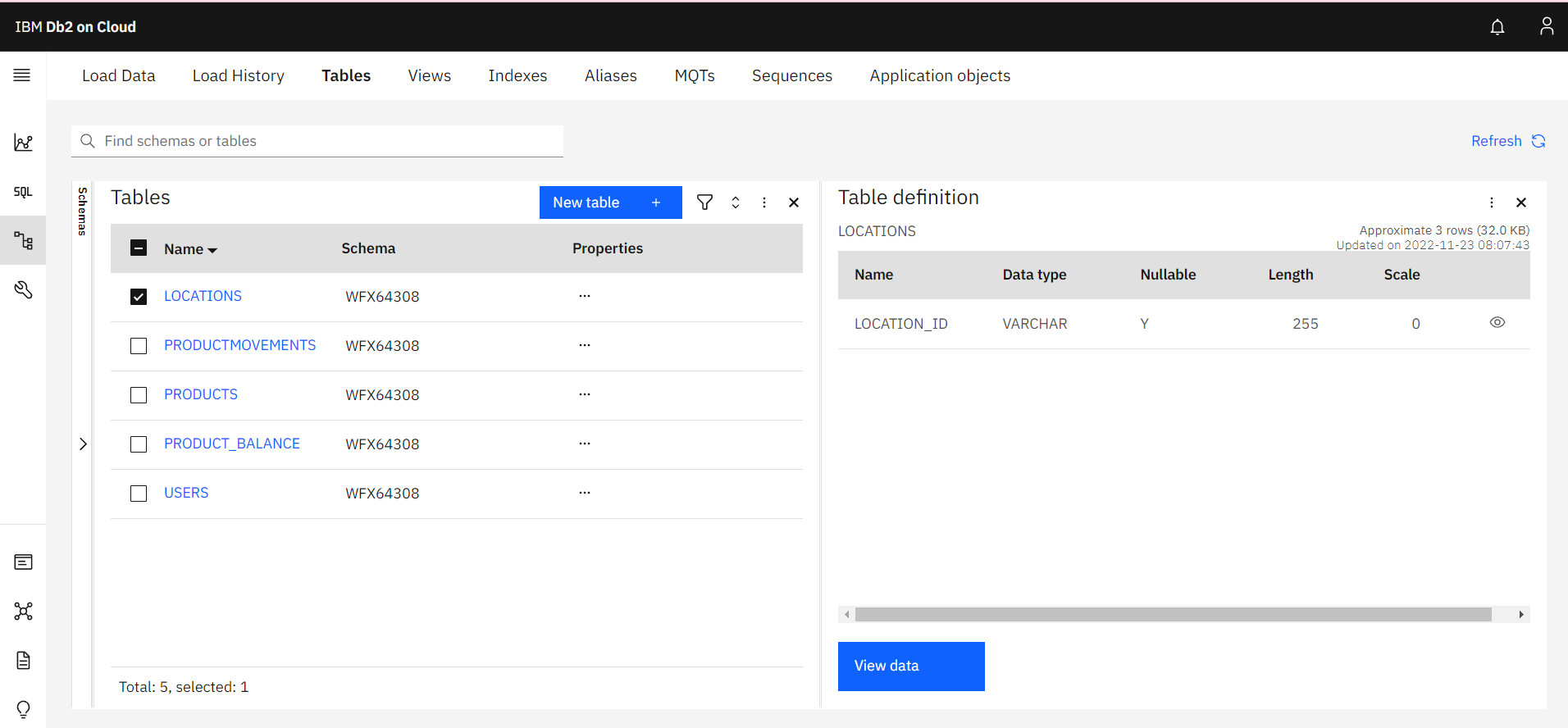
Email notifications will be sent to the users once they cross the expenditure limit through send grid mail system. Most notifications are transactional, meaning a recipient’s action or account activity triggers them. But some notifications are marketing related, encouraging the recipient to take a specific action. Ecommerce product notifications inform recipients about new products or discounts. Plus, unlike general marketing emails, these are highly personalized and focus on a singleproduct.

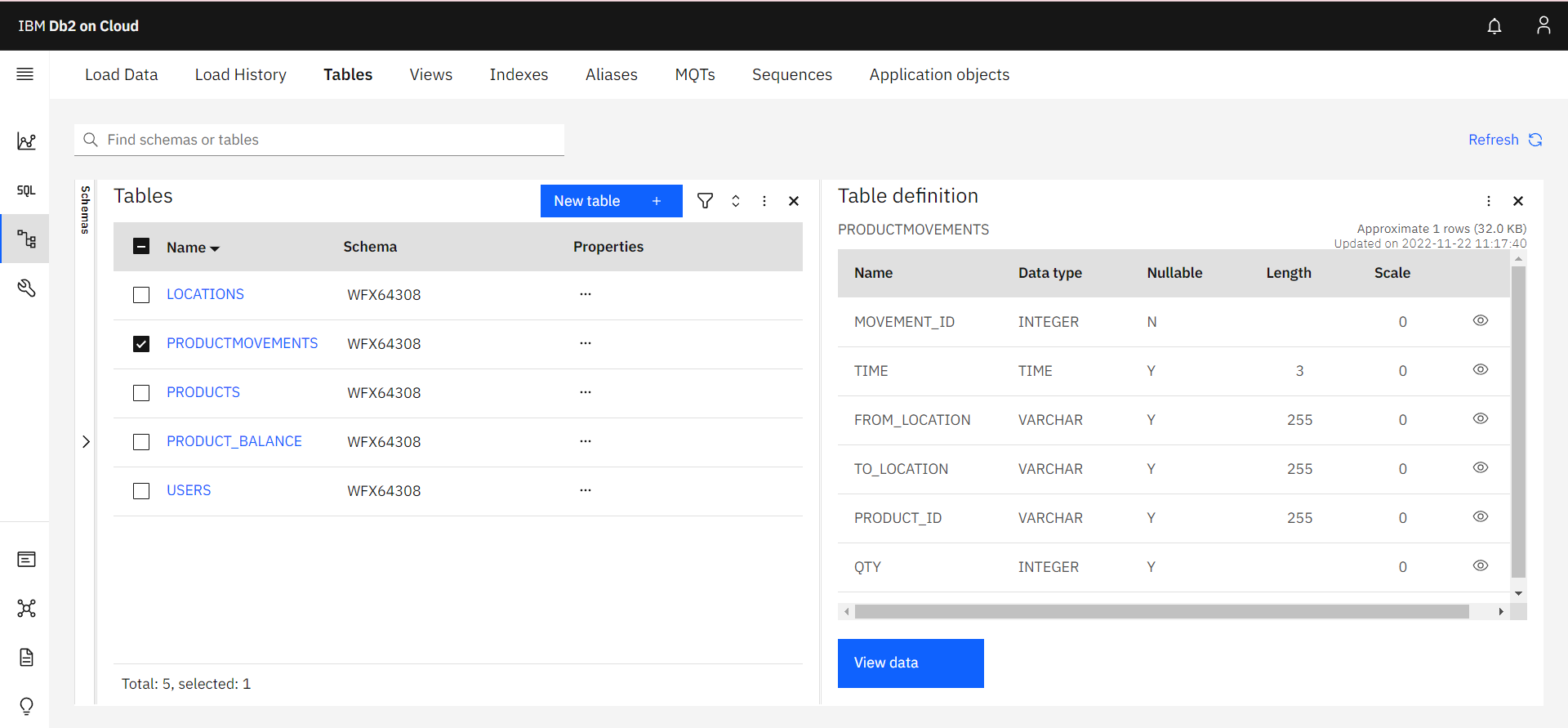
**Sendgrid.py**

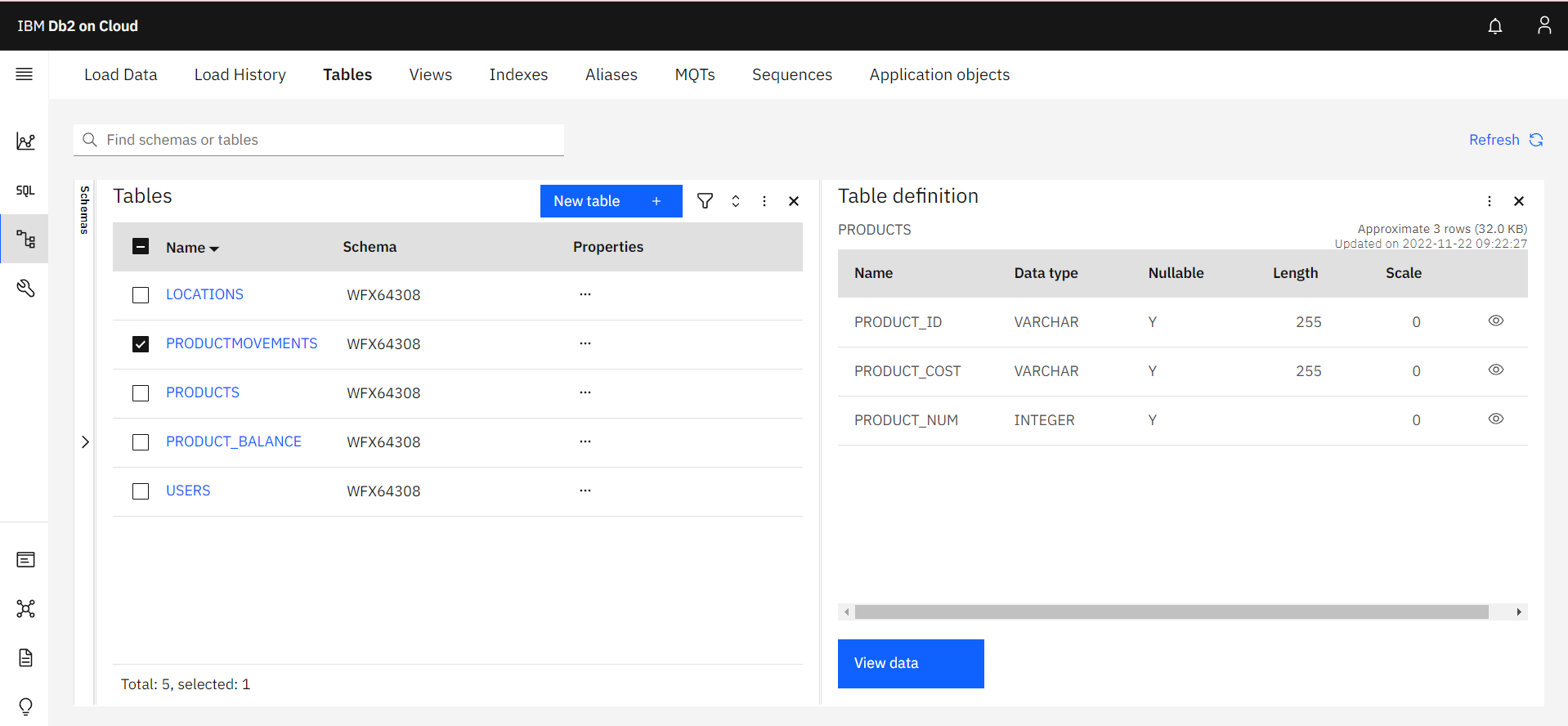


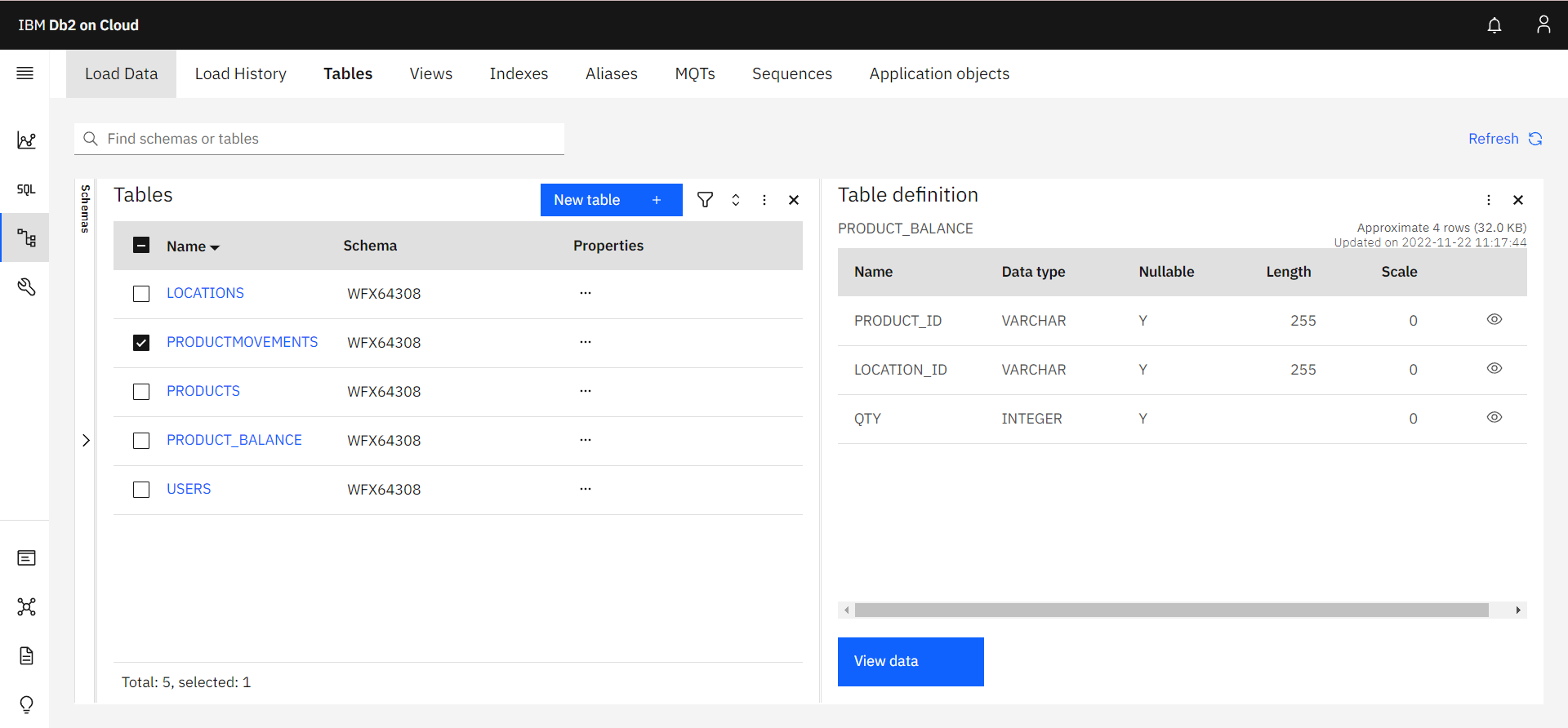
**7.3. FEATURE 3 (DATA SCHEMA)**

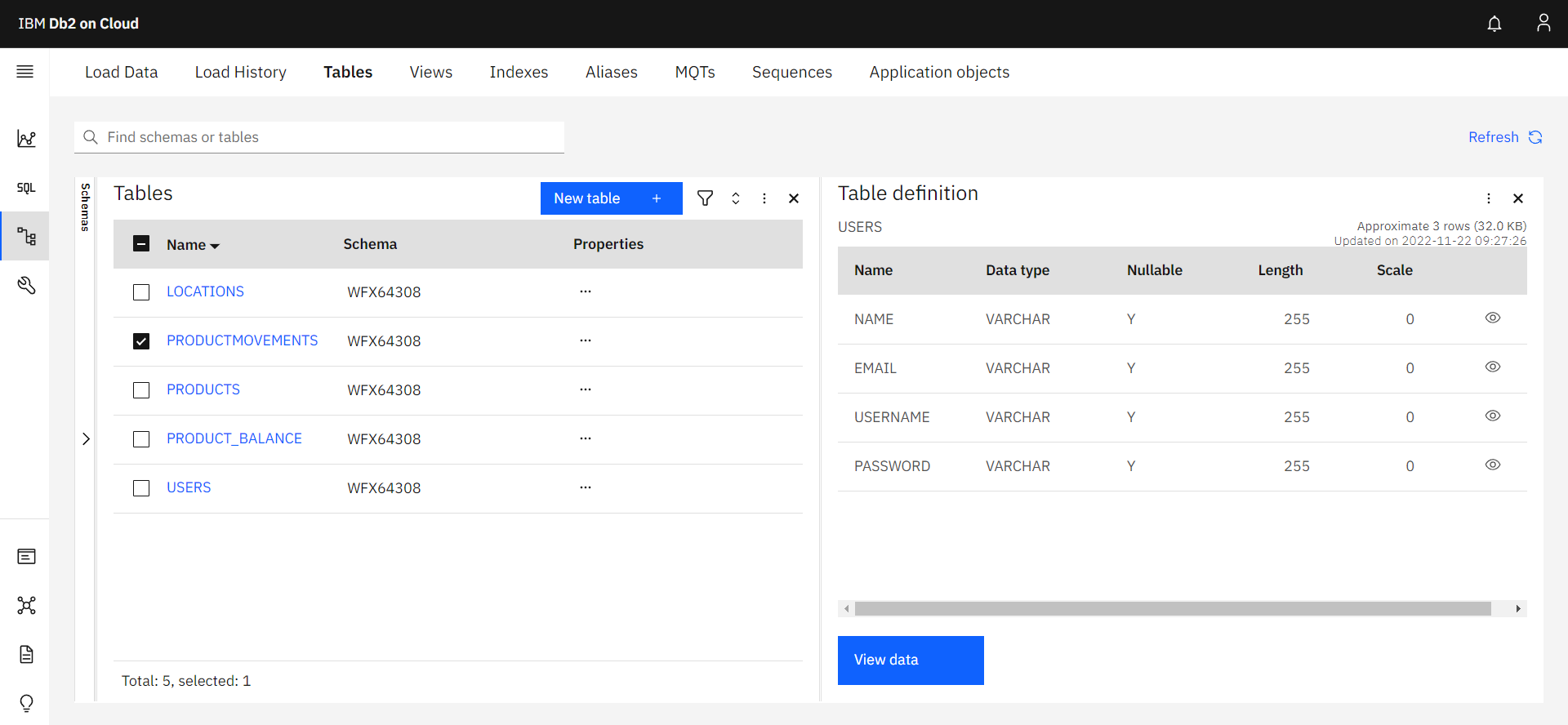
* Location
* Product
* Product\_movement
* Product\_balance
* users











**CHAPTER 8**

**8. TESTING**

**8.1 TEST CASES**

**NFT-RISK ASSESSMENT**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Project Name** | **Scope/feature** | **Functional changes** | **Impact of Downtime** | **Load/Volume changes** | **Risk Score** | **Justification** |
| Inventory Management system for retailers | New | Low | Unable to sign up or register | >5 to 10% | ORANGE | User may have given incorrect details |
| Inventory Management  system for retailers | Existing | No Changes | Couldn’t change | <5% | GREEN | Given wrong details about user |
| Inventory Management system for retailers | New | moderate | Unable to modify user | <10% | ORANGE | Duplication of user arised |
| Inventory Management  system for retailers | New | Low | Unable to predict product | <5% | GREEN | Wrong product detailsgiven |

**NFT DETAILED TEST PLAN**

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Overview** | **NFT test approach** | **Assumptions/dependencies** | **Approvals/Signoff** |
| Inventory Management system for retailers | Incorrect username | Not registered | Login unsuccessful |
| Inventory Management system for retailers | Repeating same data | User updation | Login successful |
| Inventory Management system for retailers | Incorrect user details | Login validation | Login unsuccessful |
| Inventory Management system for  retailers | No product updation | Database Updation | Update available products |

**END OF THE REPORT :**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project Overview** | **NFT test approach** | **Test outcome** | **Recommendations** | **Detected/closed/open** | **Approvals/Signoff** |
| Inventory management | New user | Registration successful | Validate logindetails | closed | Success |
| Dashboard Page | Check history | Product historynot  available | Validate user details | closed | Fail |
| Product stock | Checking updation | Updation incomplete | None | closed | Success |
| Admin page | Manage products | Product stock incomplete | Update product details | closed | Success |

**8.2 USER ACCEPTANCE TESTING**

# 1.Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Inventory Management system for Retailers project at the time of the release to User Acceptance Testing (UAT).

### 2. Defect Analysis

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Resolution** | **Severity 1** | **Severity 2** | **Severity 3** | **Severity 4** | **Subtotal** |
| By Design | 6 | 7 | 3 | 2 | 18 |
| Duplicate | 2 | 5 | 2 | 4 | 13 |
| External | 1 | 2 | 5 | 2 | 10 |
| Fixed | 10 | 1 | 5 | 10 | 26 |
| Not Reproduced | 1 | 0 | 1 | 0 | 2 |
| Skipped | 2 | 1 | 1 | 1 | 5 |
| Won't Fix | 4 | 1 | 1 | 0 | 6 |
| Totals | 26 | 17 | 18 | 19 | 80 |

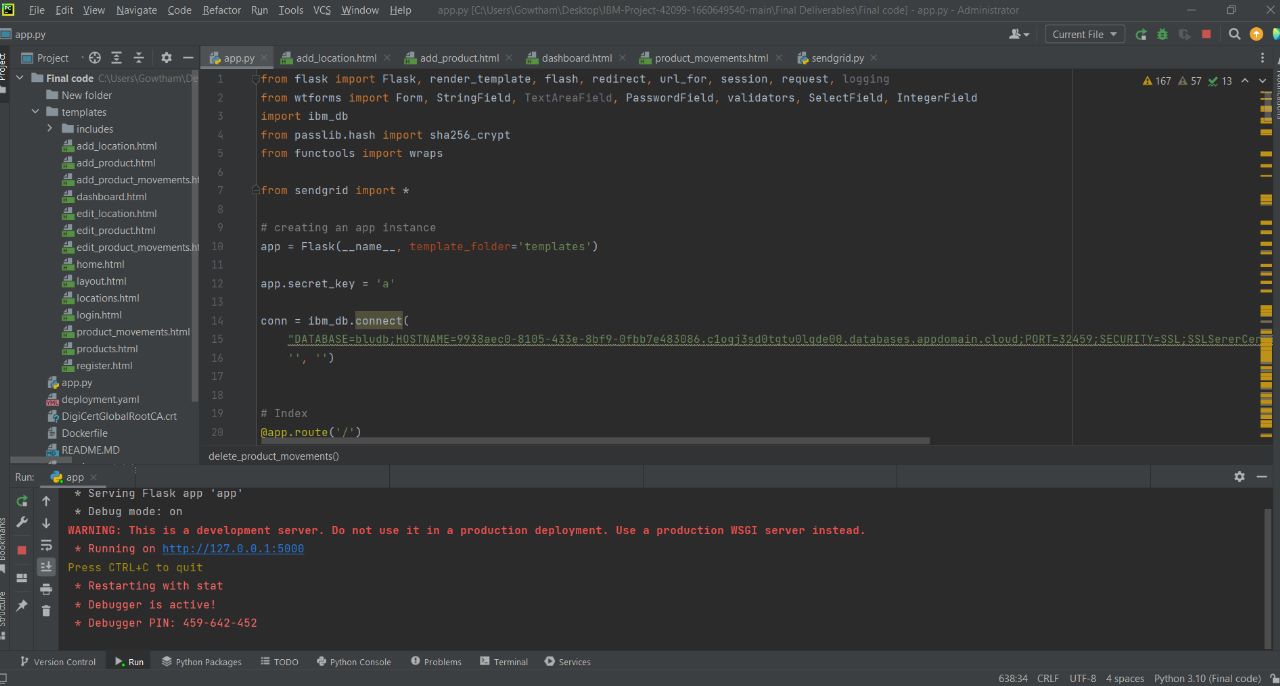
#### 1. Test Case Analysis

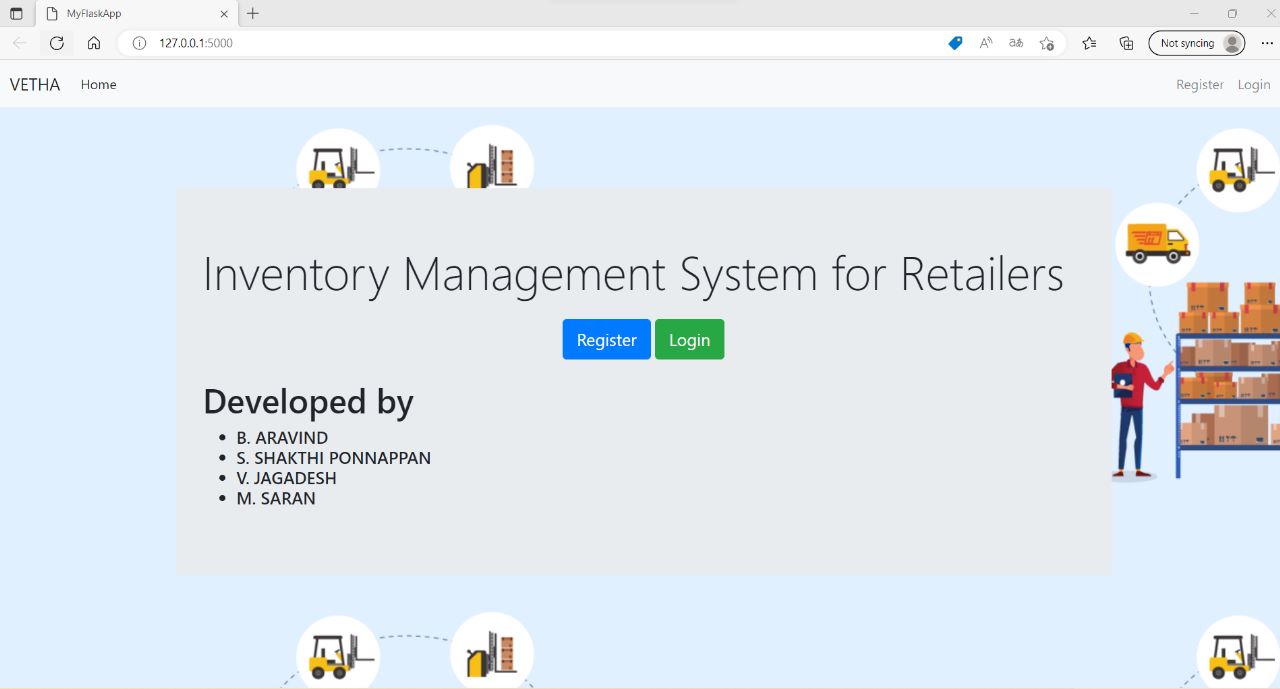
This report shows the number of test casesthat have passed,failed, and untested

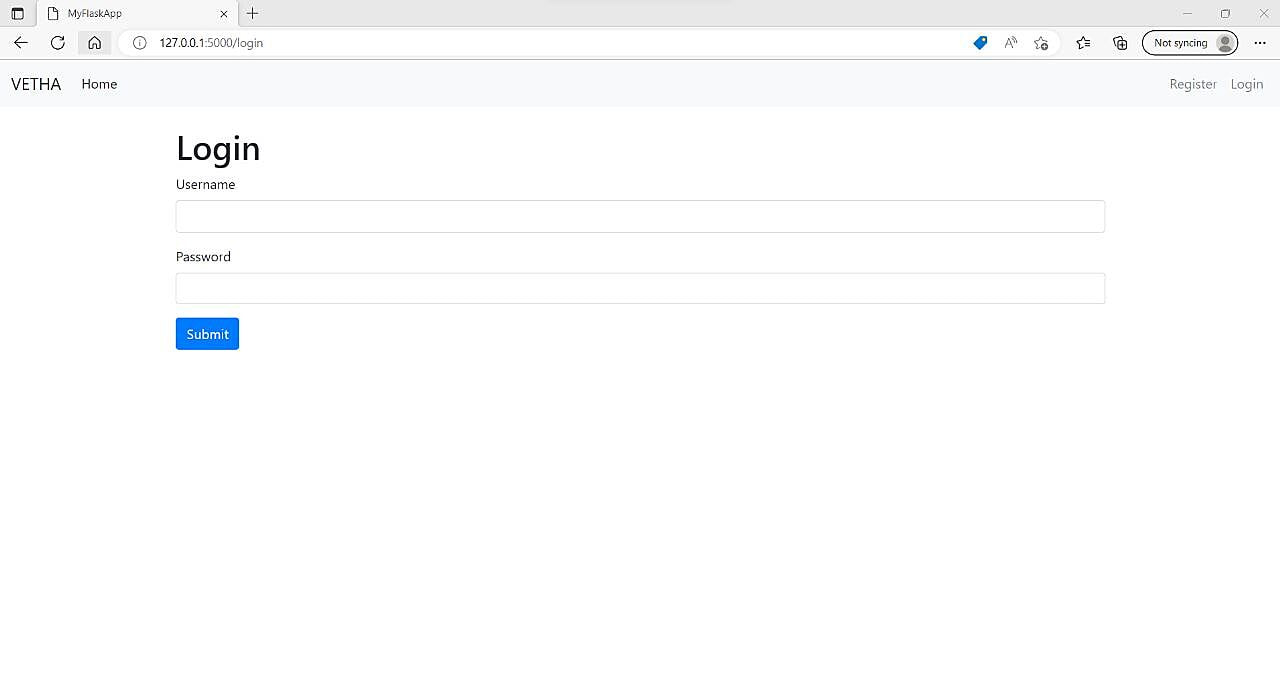
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section** | **Total Cases** | **Not Tested** | **Fail** | **Pass** |
| Print Engine | 7 | 0 | 0 | 7 |
| Client Application | 70 | 0 | 0 | 70 |
| Security | 4 | 0 | 0 | 4 |
| Outsource Shipping | 9 | 0 | 0 | 9 |
| Exception Reporting | 6 | 0 | 0 | 6 |
| Final Report Output | 4 | 0 | 0 | 4 |
| Version Control | 4 | 0 | 0 | 4 |

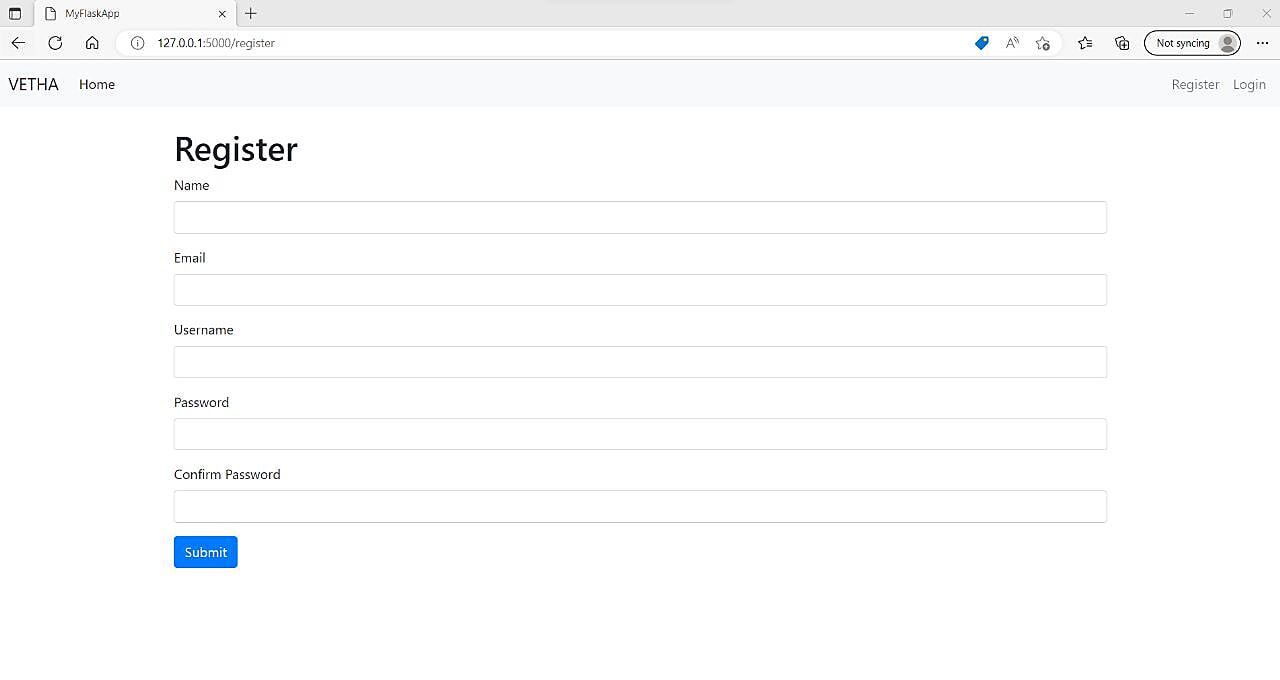
**CHAPTER 9**

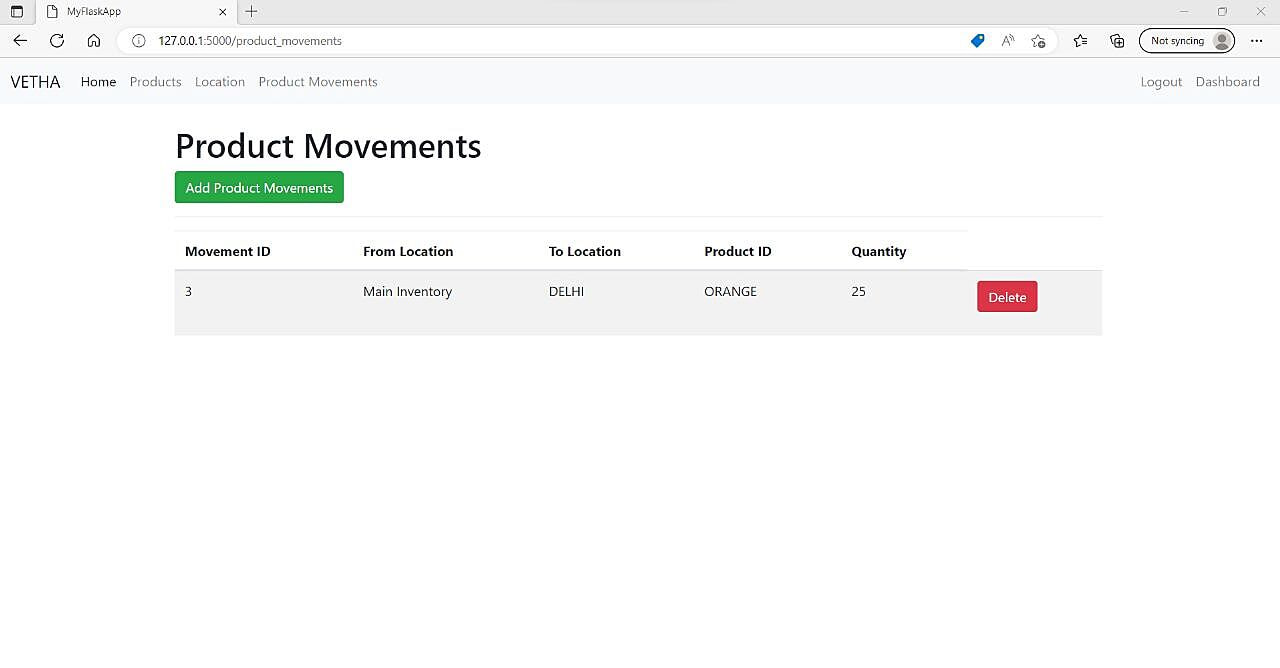
**9. RESULTS**

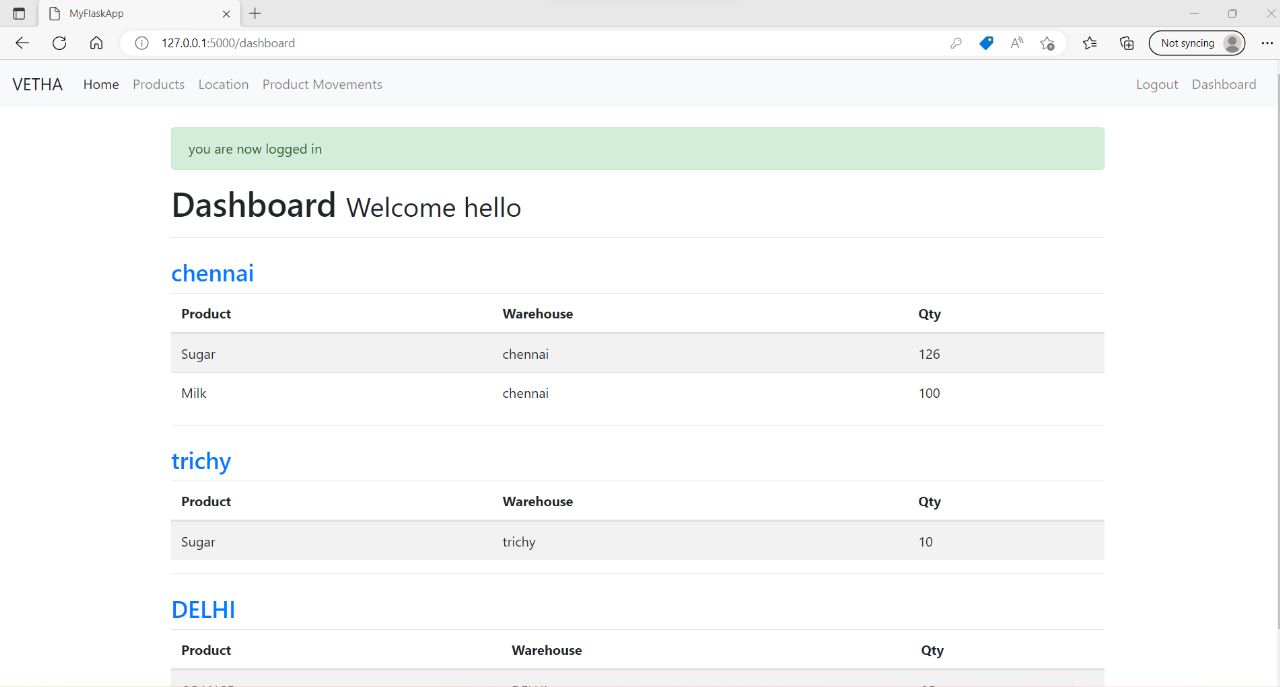


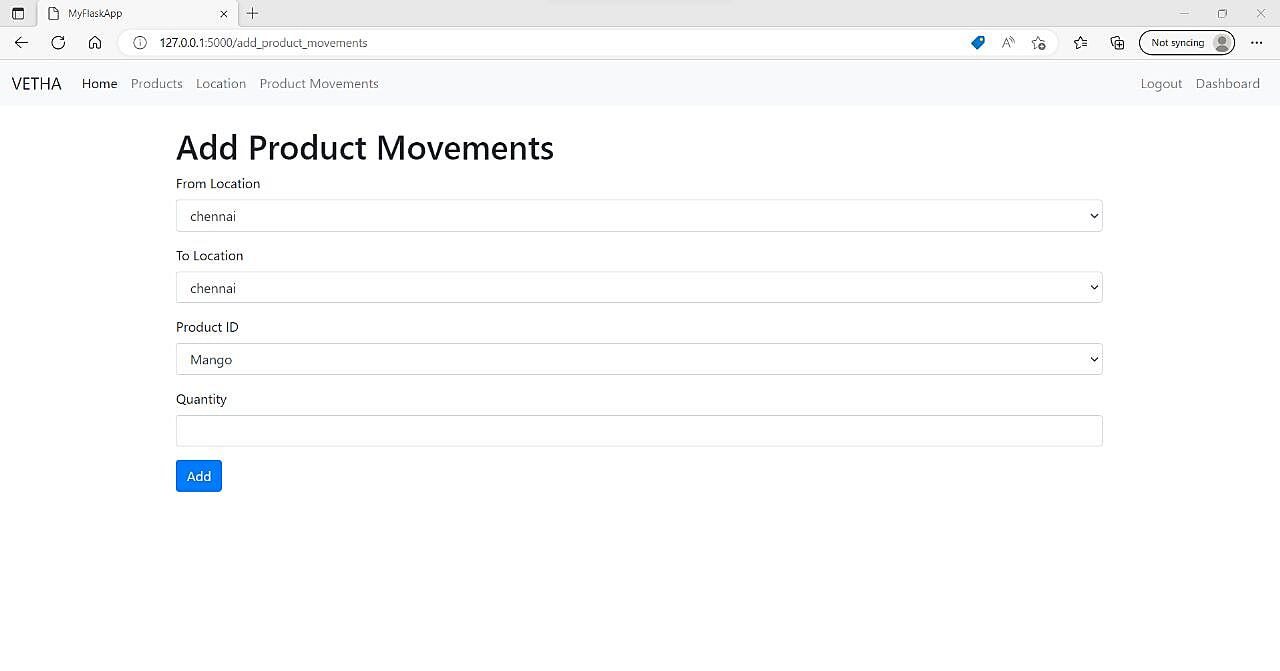


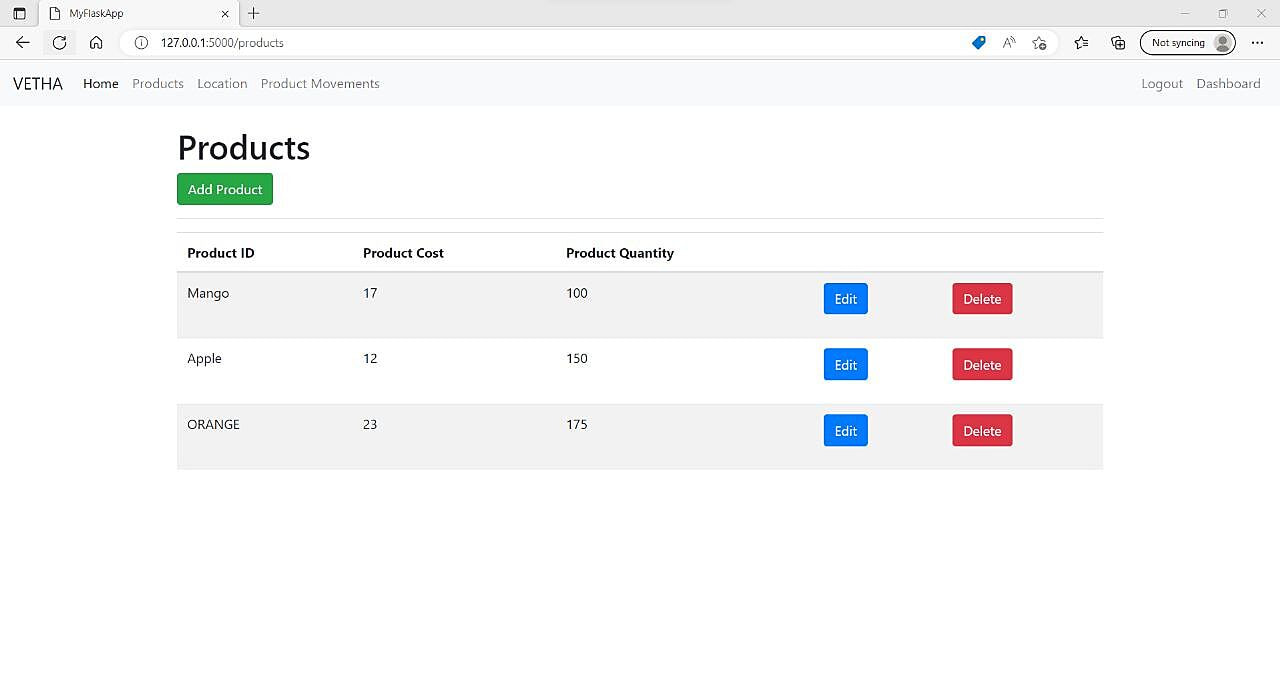












**9.1. PERFORMANCE METRICS**

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 OnHand (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

**CHAPTER 10**

**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.

**3. It saves time and money:**

An effective inventory management system can translate to time and money saved on the part of the business. By keeping track of the product that you already have at hand, you can save yourself the hassles of having to do an inventory recount in order to ensure your records are accurate. It also allows you to save cash that would have otherwise been spent on slow moving products.

**4. Improves efficiency and productivity:**

Inventory management devices like bar code scanners and inventory management software can help to greatly increase the efficiency and productivity of a business. They do this by eliminating the manual way of doing things thus allowing employees to do other more important 33 things for the business.

**5. A well-structured inventory management system leads to improved customer retention:**

For customers to keep patronizing you, you will need to always have the goods they want, at the amount they want, and at the time they want it. Inventory management helps you to meet up this demand by allowing you to have the right products all the times so that you and your customers are never stranded.

**6. Avoid lawsuits and regulatory fines:**

Like mentioned previously, inventory management allows you to keep your warehouse or facility in order. If it is not kept in order, it can result in lawsuits, injury and fines associated with not following regulatory guidelines and rules. In addition, proper inventory management (including keeping records of your staff activities) helps document your actions in the event of an undesirable situation.

**7. Schedule maintenance:**

Once you get hold of a new appliance, you can begin to schedule routine and preventative maintenance, issue work order to your staff and track that the maintenance was actually carried out. This will help to elongate the life span of that particular asset

**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 boxof washers or elbows at no charge to the customer without raising any alarms. This is done for the sake ofcustomer 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 materialfeelmore like a chore for the customer or even an entitlement.

**3. Production problem:**

Even though inventory management can reveal to you the amount of stock you have at hand and the amount that you have sold off, it can also hide production problems that could leadto customer service disasters. Since 36 the management places almost all of its focus on inventory management to the detriment of quality control, broken or incorrect items that would normally be discarded are shipped along with wholesome

**4. Increased space is need to hold the inventory:**

In order to hold inventory, you will need to have spaceso unless the goods you deal in are really small in size, then you will need a warehouse to store it. In addition, you will also need to buy shelves and racks to store your goods, forklifts to move around the stock and of course staff

**CHAPTER 11**

**11 .CONCLUSION**

The project “Inventory Management System for Retailers” mainly as the name suggests deal with the calculation of the available and processed resources for an accurate inventory control and process management for a domain specific client- 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 levelsand 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 alertto 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.

**CHAPTER 12**

**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

**CHAPTER 13**

**13. APPENDIX**

**13.1 SOURCE CODE**

**APP.PY**

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 sendgrid import \*

# creating an app instance

app = Flask(\_\_name\_\_, template\_folder='templates')

app.secret\_key = 'a'

conn = ibm\_db.connect(

"DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;SECURITY=SSL;SSLSererCertificate=DigiCertGlobalRootCA.crt;UID=wfx64308;PWD=0XBT2pfnUN05lby4",

'', '')

# 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:

return render\_template('products.html', products=products)

else:

msg = 'No products found'

return render\_template('products.html', msg=msg)

# Locations

@app.route('/locations')

def locations():

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:

return render\_template('locations.html', locations=locations)

else:

msg = 'No locations found'

return render\_template('locations.html', msg=msg)

# Product Movements

@app.route('/product\_movements')

def product\_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:

return render\_template('product\_movements.html', movements=movements)

else:

msg = 'No product movements found'

return render\_template('product\_movements.html', msg=msg)

# Register Form Class

class RegisterForm(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'])

def register():

form = RegisterForm(request.form)

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

name = form.name.data

email = form.email.data

username = form.username.data

password = sha256\_crypt.encrypt(str(form.password.data))

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'

return render\_template('login.html', error=error)

# Close connection

cur.close()

else:

error = 'Username not found'

return render\_template('login.html', error=error)

return render\_template('login.html')

# check if user logged in

def is\_logged\_in(f):

@wraps(f)

def wrap(\*args, \*\*kwargs):

if 'logged\_in' in session:

return f(\*args, \*\*kwargs)

else:

flash('Unauthorized, Please login', 'danger')

return redirect(url\_for('login'))

return wrap

# Logout

@app.route('/logout')

@is\_logged\_in

def logout():

session.clear()

flash("You are now logged out", "success")

return redirect(url\_for('login'))

# Dashboard

@app.route('/dashboard')

@is\_logged\_in

def dashboard():

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 = []

for i in locations:

locs.append(list(i.values())[0])

if result > 0:

return render\_template('dashboard.html', products=products, locations=locs)

else:

msg = 'No products found'

return render\_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']

if request.method == 'POST' and form.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'))

return render\_template('edit\_location.html', form=form)

# Delete Location

@app.route('/delete\_location/<string:id>', methods=['POST'])

@is\_logged\_in

def delete\_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['PRODUCT\_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['PRODUCT\_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:

raise CustomError("There is no product named {} in {}.".format(product\_id, from\_location))

else: # will be executed if both from\_location and to\_location are 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)

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, 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:

raise CustomError("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)

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)

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'))

return render\_template('add\_product\_movements.html', form=form)

# Delete Product Movements

@app.route('/delete\_product\_movements/<string:id>', methods=['POST'])

@is\_logged\_in

def delete\_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)

**13.2 GIT HUB & PROJECT DEMO LINK :**

**GIT HUB**

[**https://github.com/IBM-EPBL/IBM-Project-43262-1660714869**](https://github.com/IBM-EPBL/IBM-Project-43262-1660714869)

**DEMO LINK**

[**http://bitly.ws/x3tz**](http://bitly.ws/x3tz)