

IBM NALAIYA THIRAN

PROJECT REPORT

Inventory Management System for Retailers

Team ID	PNT2022TMID04814
Project Name	Inventory Management System for Retailers
Team Members	Jayaprakash P Jeevaneshwaran M Jeffrey H Jero Jenisha J

Table Of Contents

SI No	Title	Page No
1	INTRODUCTION 1.1 Project Overview 1.2 Purpose	4 4
2	LITERATURE SURVEY 2.1 Existing problem 2.2 References 2.3 Problem Statement Definition	5 5 6
3	IDEATION & PROPOSED SOLUTION 3.1 Empathy Map Canvas 3.2 Ideation & Brainstorming 3.3 Proposed Solution 3.4 Problem Solution fit	7 7 9 11
4	REQUIREMENT ANALYSIS 4.1 Functional requirement 4.2 Non-Functional requirements	12 13
5	PROJECT DESIGN 5.1 Data Flow Diagrams 5.2 Solution & Technical Architecture	14 16

6	PROJECT PLANNING & SCHEDULING 6.1 Sprint Planning & Estimation 6.2 Sprint Delivery Schedule	17 18
7	CODING & SOLUTIONING 7.1 Feature 1 7.2 Feature 2 7.3 Database Schema (if Applicable)	19 20 20
8	TESTING 8.1 Test Cases 8.2 User Acceptance Testing	22 23
9	RESULTS 9.1 Performance Metrics	25
10	ADVANTAGES & DISADVANTAGES	28
11	CONCLUSION	29
12	FUTURE SCOPE	29
13	APPENDIX 13.1 Source Code 13.2 Git hub & Demo link	30 41

1. INTRODUCTION

1.1 Project Overview

Inventory management is a challenging problem in supply chain management. A tool or system to aid the inventory management would be a beneficial tool in this area. The term inventory refers to a company's stockpile of material and the components that make up the output. Inventory management refers to managing the quantity, quality, location and transportation of various products utilised in manufacturing by various industrial organisations or in sales by various retailers.

Accurately maintaining the quantity (in numbers) of the finished goods in the inventory makes it possible to quickly assess the quantity of products needed for the upcoming sales. It also improves the communication between the entities of the supply chain like retailers, manufacturers, customers, etc.

1.2 Purpose

To ensure there is enough goods or materials to meet demand without creating overstock, or excess inventory.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

- Inability to locate the inventory stocks
- Choosing a manual inventory process
- Outdated products
- Inability to manage inventory waste and defects
- Overstocking problems

2.2 REFERENCE

Several experiments have been carried out over the years by different groups of researchers. Here are some of the following groups:

[1] Abisoye, O. A., Boboye, F., & Abisoye, B. O. (2013). Design of a computerized inventory management system for supermarkets.

[2] Li, S. G., & Kuo, X. (2008). The inventory management system for automobile spare parts in a central warehouse. *Expert Systems with Applications*, 34(2), 1144-1153.

[3] Saleem, A. (2020). Automated inventory management systems and its impact on supply chain risk management in manufacturing firms of Pakistan. *Int J Supply Chain Manag*, 9, 220-231.

[4] Bose, R., Mondal, H., Sarkar, I., & Roy, S. (2022). Design of smart inventory management system for construction sector based on IoT and cloud computing. *e- Prime-Advances in Electrical Engineering, Electronics and Energy*, 2, 100051.

[5] Madishetti, S., & Kibona, D. (2013). IMPACT OF INVENTORY MANAGEMENT ON THE PROFITABILITY OF SMES IN TANZANIA CLEAR International Journal of Research in Commerce & Management, 4.

2.3 Problem Statement Definition

Inventory management is a challenging problem in supply chain management. The problem faced by the company is that they do not have any system to keep track of inventory data. It is difficult for the retailer to record the inventory data.

Every inventory stock manager's main problem is keeping track of how much stock is purchased and how much stock is spent. A tool or system to aid the inventory

management would be a beneficial tool in this area. Inventory management refers to managing the quantity, quality, location and transportation of various products

utilized in manufacturing by various industrial organizations or in sales by various retailers. Usually, Inventory Management systems are limited and fixed to a selected range of items and cannot be modified and extended based on the customer's needs. The Inventory Management System focuses on making it expandable and usable easily by the end user and with constant customer support to alter the use. Unlike other software that provides similar functionalities, Inventory Management System focuses on making it easier by adding details of various other entities that is a part of organization.

3. IDEATION AND PROPOSED SYSTEM

3.1 Empathy Map Canvas



3.2. Ideation and Brainstorming

Template

Brainstorm & idea prioritization

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

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

➔

Before you collaborate

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

⌚ 10 minutes

A Team gathering

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

B Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.

C Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) ➔

1

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

To create an inventory management system that will allow retailers to satisfy demand from customers without running out of inventory or holding too much on hand.

2

Brainstorm

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

10 minutes

Tip
Don't select a sticky note and then start writing a sticky note to start writing.



3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes



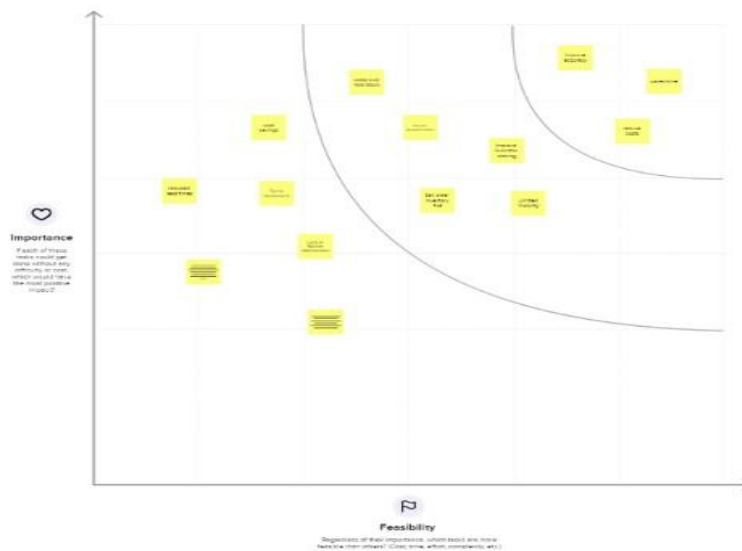
Tip
Add a sentence-like label to every sticky note to make it easier to find, share, organize, and integrate relevant ideas as you build your mural.

4

Prioritize

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

20 minutes



5

After you collaborate

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

Quick add-ons

- Share the mural:** Share a viewable link to the mural with collaborators to keep track of the map about the outcomes of the process.
- Export the mural:** Export a copy of the mural as a PNG or PDF to attach to emails, include in slides, or save in your drive.

Keep moving forward

- Strategy blueprint:** Define the components of a new idea or strategy.
[Open the template](#)
- Customer experience journey map:** Understand customer needs, motivations, and obstacles for an experience.
[Open the template](#)
- Strengths, weaknesses, opportunities & threats:** Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
[Open the template](#)

[Share template feedback](#)

3.3 Proposed Solution

Problem Statement:

- The retailer's one of the challenging problems is that they face lot of issues in keeping track of the inventory.
- The retailers must know the expiry date of the products so as to avoid wastage of products and loss of money.
- The retailers should also keep track of the fast-moving goods and also the dead stocks to avoid going out of stocks for fast moving goods and also surplus incoming of dead goods.
- The customers lose satisfaction and reliability over the retailers as there is a high chance the product is not delivered on time or it may be a very old one (in case of dead goods).

Idea /Solution description:

- This proposed system/app will keep track of the details of every incoming and outgoing goods.
- The system will notify or alert the retailer over the expiry date of the products.
- The availability of stocks of each product is kept in track and the retailer is notified when it goes below the threshold limit.
- All the customers will have their own account on the app which they can use to buy products from the retailers.
- Each customer can see the details of retailers available in their zone, check for product availability and order their product.
- Both the retailers and the customers can track the order easily with this application.

Novelty /Uniqueness:

- Notifications will be sent to the retailers if any product that the customers have been looking for is not available so that the product can be stocked up soon.
- Also retailers will notified about the dead stocks in the inventory so that they could stop stocking them since customers are not preferring them.
- Notification will be sent to the customers who buys certain products regularly when the new arrivals are stocked up.
- Notifications are sent to the customer for the products in their wishlist to intimate them that the product is available or about any discounts on those products
- Exclusive discounts and offers are given for regular customers to keep them engaged with the store regularly.

Social Impact /Customer Satisfaction:

- One important reason, the customers are highly satisfied with this app is that they won't waste time on the product which is unavailable. They can check availability from the app itself.
- Since the app is automated and it is constantly updating after every purchase the work of keeping track of products is
- almost NIL for retailers.
- The customer satisfaction is improved reasonably due to the timely service offered to them.
- The money wasted on expired and dead goods is greatly reduced which helps the retailers a lot.

Business Model (Revenue Model):

- Hereby we can provide a robust and most reliable inventory management system by using:
- Can deploy the most appropriate business advertising models.
- Can implement loss preventing strategies with this model.

3.4 Problem Statement Fit

<p>Define CS, fit into CC</p> <p>1. CUSTOMER SEGMENT(S) CS</p> <ul style="list-style-type: none"> ❖ Customers are retailers, shop owners, business people who are struggling to keep track of their inventory. ❖ Due to this issue, they face many issues like: <ul style="list-style-type: none"> ✓ Loss due to dead products in the inventory, unavailability of fast moving products, etc. ✓ Unnecessary headache due to improper maintenance of inventory. 	<p>6. CUSTOMER CONSTRAINTS CC</p> <ul style="list-style-type: none"> ❖ Since most of the softwares like these will be a subscription model, the customer must be paying as they use them. This may be against their budget. ❖ To use this software the customer must be trained or he must hire a person to do that for him. ❖ To deploy this software, the customer must have a powerful device which is compatible with the software. 	<p>5. AVAILABLE SOLUTIONS AS</p> <ul style="list-style-type: none"> ❖ Solution: The traditional solution for the inventory management problem is to track the incoming and outgoing goods with a pen and paper. ❖ Pros: <ul style="list-style-type: none"> ✓ Easy to use ✓ Less cost ❖ Cons: <ul style="list-style-type: none"> ✓ Error rate is high ✓ Manual tracking is a tedious work <p>Explore AS, differentiate</p>
<p>Focus on J&P, tap into BE, understand RC</p> <p>2. JOBS-TO-BE-DONE / PROBLEMS J&P</p> <ul style="list-style-type: none"> ❖ The objective of the software is to make the inventory tracking easier by automating the inventory. Example, the initial stocks information is fed to the software and from there it tracks the details of incoming and outgoing products. ❖ This can generate automatic alerts/notifications to help the user in their work. Example, Alert for dead stocks in inventory, Alert for the goods which is to be refilled, Notifications for the user defined conditions like if sales go higher than certain limits etc... ❖ Graphical representation of sales is also possible. 	<p>9. PROBLEM ROOT CAUSE RC</p> <ul style="list-style-type: none"> ❖ The primary reason for this problem to exist is the periodic change in demand of the customers. ❖ This indirectly affects the inventory as change in customers needs is proportional to the sale of a particular products. ❖ This keeping track of inventory effectively helps in managing the dead and fast moving products. 	<p>7. BEHAVIOUR BE</p> <ul style="list-style-type: none"> ❖ The customer must find an effective inventory tracking software. ❖ He must implement it in his business to streamline his work and make more profit. ❖ He must volunteer himself to learn to use the software or be ready to hire a person who can do it for him. <p>Focus on J&P, tap into BE, understand RC</p>
<p>Identify strong TR & EM</p> <p>3. TRIGGERS TR</p> <ul style="list-style-type: none"> ❖ Understanding the fact that using a software to automate inventory system helps him to make more money and also make his work easier. Also seeing other retailers making more money using this software. <p>4. EMOTIONS: BEFORE / AFTER EM</p> <p>Before: They feel lost due to losses which occur due to improper management of inventory (Manual pen and paper tracking).</p> <p>After: They feel like success after making increased profits, reducing the mistakes that happen in manual process.</p>	<p>10. YOUR SOLUTION SL</p> <ul style="list-style-type: none"> ✓ Design a flask based Inventory management system application. ✓ Enable email based alerts for dead and fast moving products using sendgrid framework. ✓ Provide an option for graphical view of sales 	<p>8. CHANNELS of BEHAVIOUR CH</p> <p>8.1 ONLINE</p> <p>Online Inventory trackers which come for free may steal personal information of users and it may also contain a lot of ads.</p> <p>8.2 OFFLINE</p> <p>Manual logs can be maintained. Employees can be hired to maintain the inventory system logs when the business grows.</p> <p>Extract online & offline CH of BE</p>

4. REQUIREMENT ANALYSIS

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	Account Creation	Creation through Google Creation through Email Creation through Linked IN Creation through Git hub
FR-2	User Confirmation	Confirmation via Email
FR-3	Successful Login	Notification through Email
FR-4	Update inventory details	Notification through Email
FR-5	Add new stock	Notification through Email
FR-6	Unavailability of stock	Alert notification through Email

4.2 Non-functional Requirements:

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

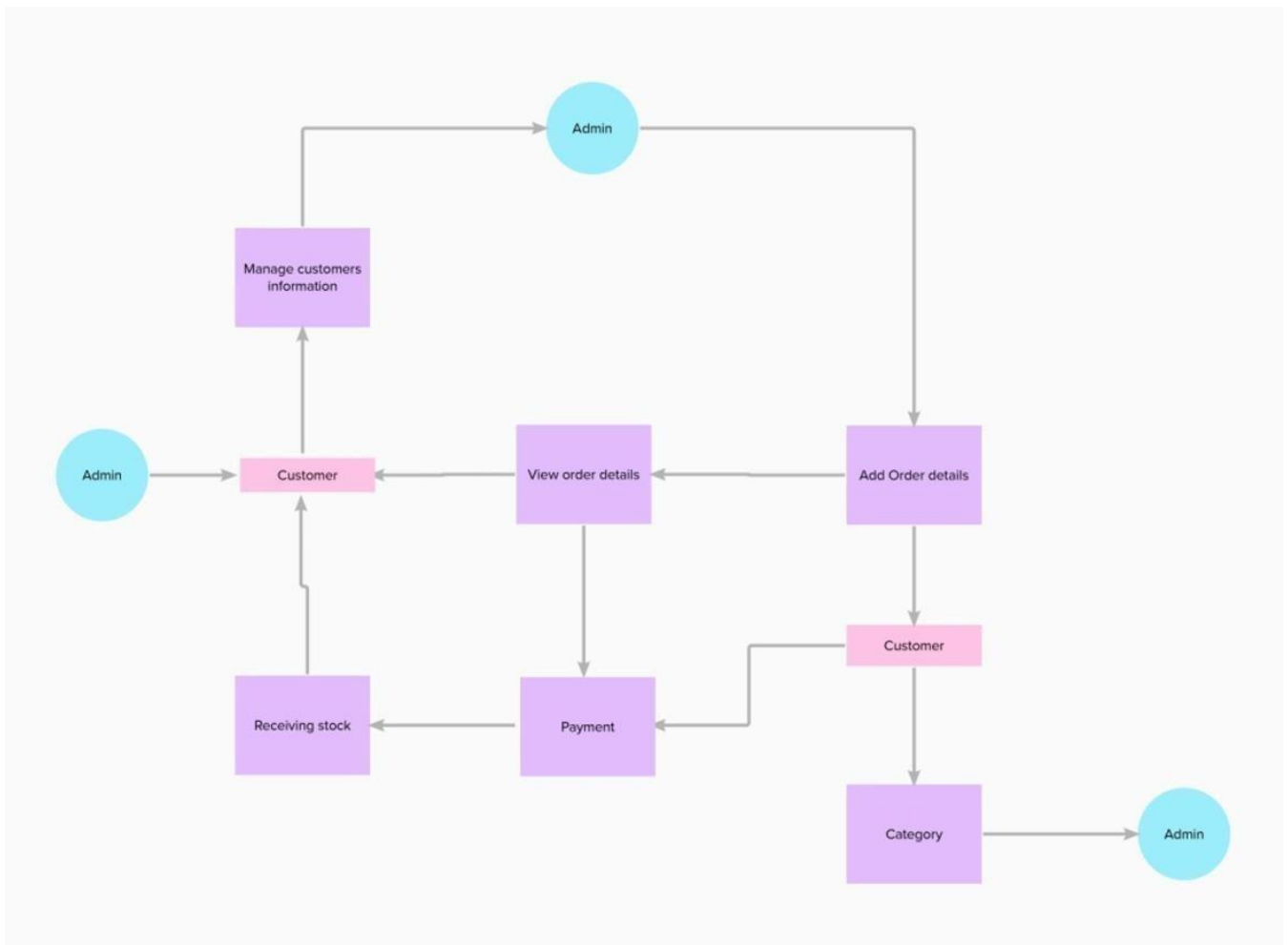
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The application must have a good looking user friendly interface.
NFR-2	Security	The application must be secured with proper user name and passwords.
NFR-3	Reliability	The application should work properly, even when faults occur.
NFR-4	Performance	The application must perform well in different scenarios.
NFR-5	Availability	The application must available 24 hours a day with no bandwidth issues.
NFR-6	Scalability	The application should able to increase or decrease in performance and cost in response to changes in application and system processing demands.

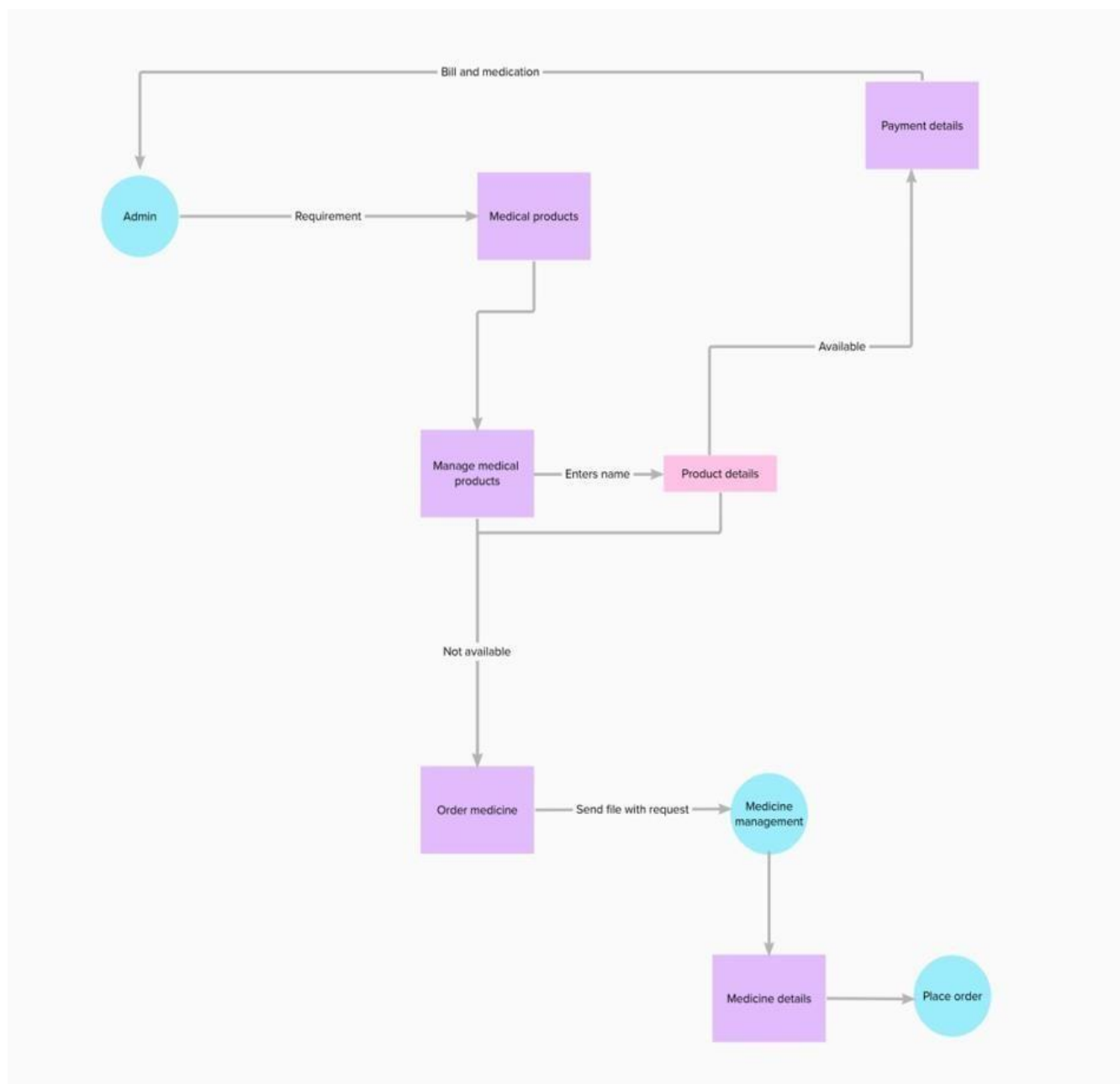
5. PROJECT DESIGN

5.1 Data Flow Diagrams:

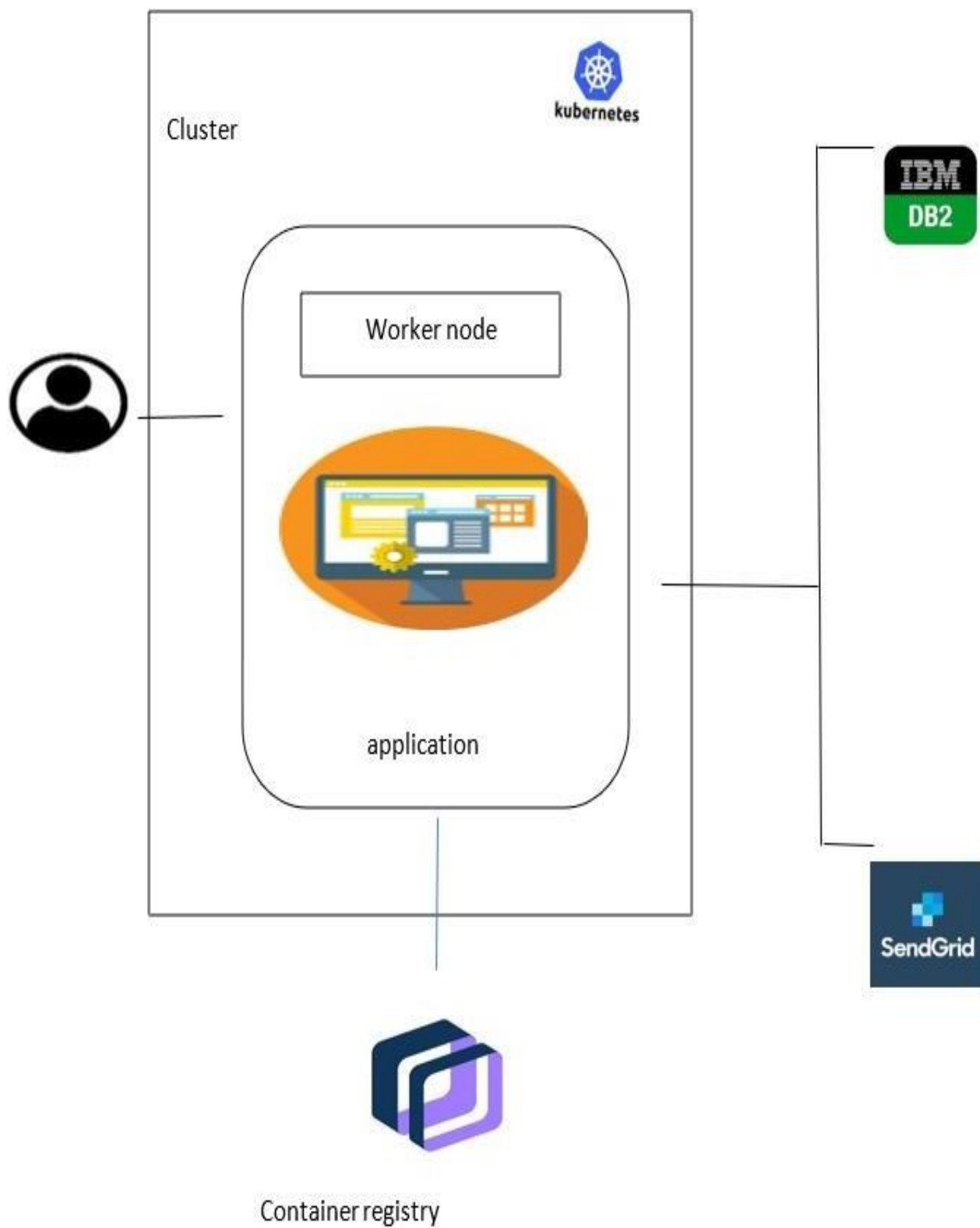
Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Example: Simplified





5.2 Solution and Technical Architecture :



6. PROJECT PLANNING AND 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.	2	High	4
Sprint- 1		USN-2	As a user, I can register for the application through E-mail	1	Medium	4
Sprint- 1	Confirmation	USN-3	As a user, I will receive confirmation email once I have registered for the application	2	Medium	4
Sprint- 1	Login	USN-4	As a user, I can log into the application by entering email & password	2	High	4
Sprint- 2	Dashboard	USN-5	As a user, I can view the products which are available	4	High	4
Sprint- 2	Add items to cart	USN-6	As a user, I can add the products I wish to buy to the carts.	5	Medium	4
Sprint- 3	Stock Update	USN-7	As a user, I can add products which are not available in the dashboard to the stock list.	5	Medium	4

Sprint- 4	Request to Customer Care	USN-8	As a user, I can contact the Customer Care Executive and request any services I want from the customer care.	5	Low	4
Sprint- 4	Contact Administrator	USN-9	I can be able to report any difficulties I experience as a report.	5	Medium	4

6.2 Sprint delivery schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	7	6 Days	27 Oct 2022	29 Oct 2022	7	29 Oct 2022
Sprint-2	9	6 Days	01 Nov 2022	05 Nov 2022	9	05 Nov 2022
Sprint-3	5	6 Days	07 Nov 2022	12 Nov 2022	5	12 Nov 2022
Sprint-4	10	6 Days	14 Nov 2022	19 Nov 2022	10	19 Nov 2022

7. CODING & SOLUTIONING

7.1 Feature 1:

Python

- Python is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics, used for general-purpose programming. It's everywhere, and people use numerous Python-powered devices on a daily basis, whether they realize it or not.
- Python was created by Guido van Rossum, and first released on February 20, 1991.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages.
- Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL)
- It is easy to learn – the time needed to learn Python is shorter than for many other languages; this means that it's possible to start the actual programming fast
- It is easy to use for writing new software – it's often possible to write code faster when using Python.
- It is easy to obtain, install and deploy – Python is free, open and multiplatform; not all languages can boast that.
- Programming skills prepare you for careers in almost any industry and are required if you want to continue to more advanced and higher-paying software development and engineering roles.
- Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

7.2 Feature 2:

Flask

- **Flask** is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.
- It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.
- Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.
- Applications that use the Flask framework include Pinterest and LinkedIn.

7.3 Database Scheme

IBM Db2

- DB2 is a database product from IBM.
- It is a Relational Database Management System (RDBMS). DB2 is designed to store, analyze and retrieve the data efficiently.
- DB2 product is extended with the support of Object-Oriented features and non-relational structures with XML.
- Provide a massively parallel processing (MPP) architecture Exploits Hive, HBase and Apache Spark concurrently for best-in-class analytic capabilities.

- Provides low latency support for ad-hoc and complex queries, high performance, and federation capabilities Understands dialects from other vendors and various products from Oracle, IBM® Db2® and IBM Netezza® Enables advanced row and column security

Kubernetes

- **Kubernetes** is also known as '**k8s**'.
- **Kubernetes** is an extensible, portable, and open-source platform designed by **Google** in **2014**.
- It is mainly used to automate the deployment, scaling, and operations of the container-based applications across the cluster of nodes.
- Kubernetes helps to manage containerised applications in various types of physical, virtual, and cloud environments.
- Google Kubernetes is a highly flexible container tool to consistently deliver complex applications running on clusters of hundreds to thousands of individual servers
- Kubernetes is the Linux kernel which is used for distributed systems.
- It helps you to be abstract the underlying hardware of the nodes(servers) and offers a consistent interface for applications that consume the shared pool of resources.

8. TESTING

8.1 Test case

- It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectation and does not fail in an unacceptable manner.
- There are various types of test. Each test type addresses a specific testing requirement

				Date	03-Nov-22								
				Team ID	PNT2022TMD04814								
				Project Name	Inventory Management System for Retailer								
Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_001	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button.		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Signup popup displayed or	Inventorymanagement.localhost	Login/Signup popup should display	Working as expected	Pass				
LoginPage_TC_002	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Signup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link e.Last password? Recovery password link	Username: mjeewan5791@gmail.com password: jeevan123	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link e.Last password? Recovery	Working as expected	Fail	Steps are not clear to follow		BUG-1234	Admin
LoginPage_TC_003	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text	Username: mjeewan5791@gmail.com password: jeevan123	User should navigate to user account homepage	Working as expected	pass				
LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter Invalid username/email in Email text box 4.Enter valid password in password text box	Username: mjeewan5791@gmail.com password: jeevan123	Application should show 'Incorrect email or password' validation message.	Working as expected	Fail	need to verify		bug-1235	customer
LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text	Username: mjeewan5791@gmail.com password: jeevan123	Application should show 'Incorrect email or password' validation message.	Working as expected	pass				
LoginPage_TC_005	Functional	Login page	Verify user is able to log into application with Invalid credentials		1.Enter URL and click go 2.Click on My Account dropdown button 3.Enter Invalid username/email in Email text box 4.Enter Invalid password in password text box	Username: mjeewan5791@gmail.com password: jeevan123	Application should show 'Incorrect email or password' validation message.						

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 they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Sub total
By Design	8	7	1	2	18
Duplicate	2	0	2	0	4
External	2	3	1	2	8
Fixed	12	1	5	17	35
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	16	13	23	76

3. Test Case Analysis

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

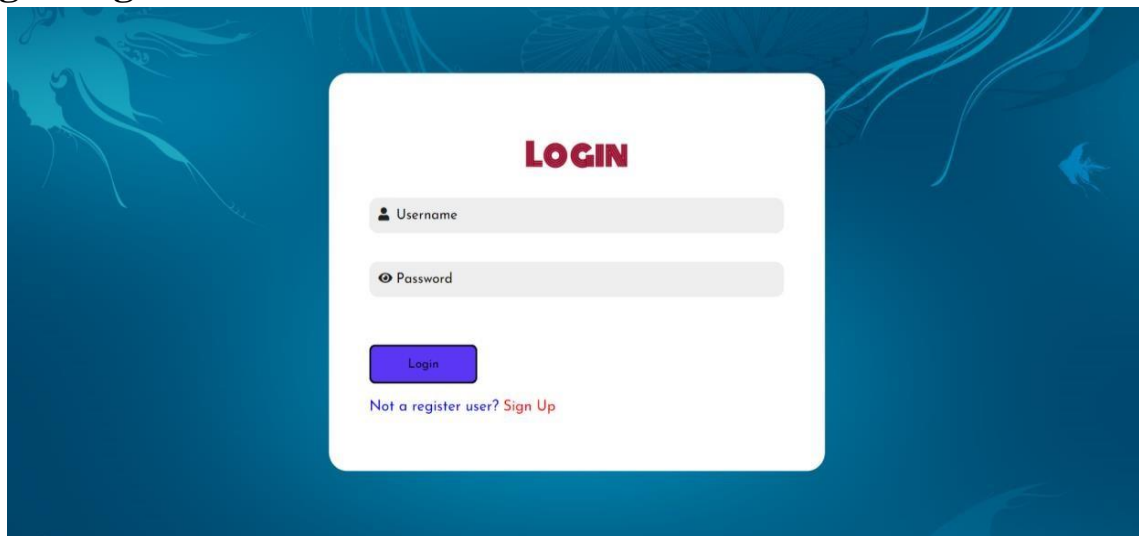
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	8	0	0	8
Client Application	50	0	0	50
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	10	0	0	10
Final Report Output	6	0	0	6
Version Control	3	0	0	3

9. RESULTS

9.1 Performance Metrics

- Project metrics are used to track the progress and performance of a project.
- Monitoring parts of a project like productivity, scheduling, and scope make it easier for team leaders to see what's on track.
- As a project evolves, managers need access to changing
- deadlines or budgets to meet their client's expectations

Login Page:

A screenshot of a login page with a blue background featuring abstract white and light blue patterns. In the center is a white rounded rectangle containing the word "LOGIN" in bold red capital letters. Below it are two input fields: "Username" with a person icon and "Password" with an eye icon. A blue "Login" button is positioned below the password field. At the bottom of the white box, it says "Not a register user? Sign Up" in red text.

LOGIN

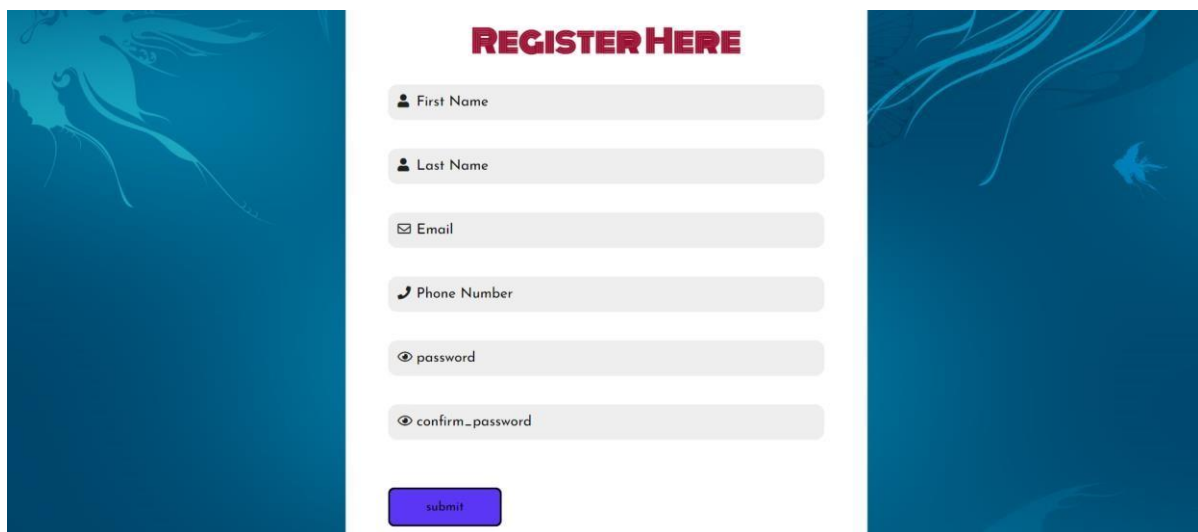
Username

Password

Login

Not a register user? [Sign Up](#)

Register Page:

A screenshot of a registration page with a blue background featuring abstract white and light blue patterns. In the center is a white rounded rectangle containing the words "REGISTER HERE" in bold red capital letters. Below it are six input fields: "First Name", "Last Name", "Email" with an envelope icon, "Phone Number" with a phone icon, "password" with an eye icon, and "confirm_password" with an eye icon. A blue "submit" button is at the bottom of the white box.

REGISTER HERE

First Name

Last Name

Email

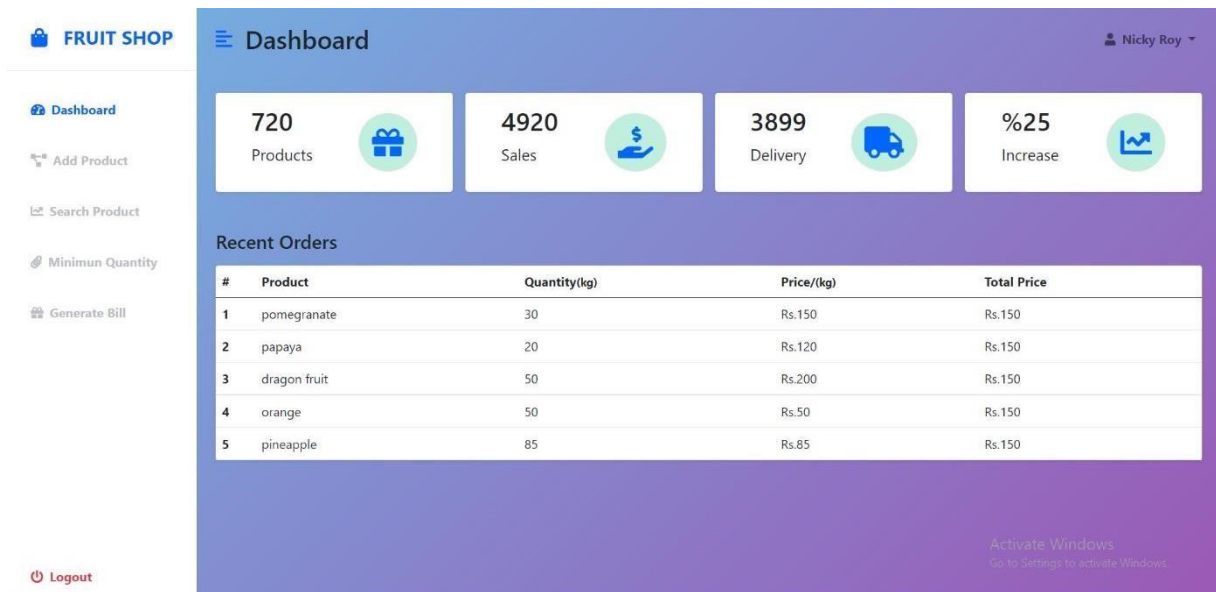
Phone Number

password

confirm_password

submit

Dashboard Page:



Add Product Page:

The 'Add Product' form is centered on the page. It contains four input fields for 'Product Name', 'Stock Quantity', 'Product Price / (kg)', and 'Stock Alert min. Qty'. A purple 'Add Product' button is at the bottom. A Windows activation notice is in the bottom right corner.

Add Product

Product Name

Stock Quantity

Product Price / (kg)

Stock Alert min. Qty

Add Product

IBM DB2 :

IBM Db2 on Cloud

Load DataLoad History**Tables**ViewsIndexesAliasesMQTSequencesApplication objects

Find schemas or tables

Refresh

Tables

New table +

Filter

Sort

Columns

Close

<input checked="" type="checkbox"/>	Name	Schema	Properties
<input checked="" type="checkbox"/>	PROJECT_DATASET	TMH29490	...

Total: 1, selected: 1

Table definition

No statistics available.

Name	Data type	Nullable	Length	Scale
SCHOOL	VARCHAR	Y	2	0
SEX	VARCHAR	Y	1	0
AGE	SMALLINT	Y		0

View data

IBM Db2 on Cloud

Load DataLoad History**Tables**ViewsIndexesAliasesMQTSequencesApplication objects

Load Data +

	STATUS	SOURCE	FILENAME	TARGET	REQUESTED BY	ROWS LOADED	ROWS REJECTED
<input checked="" type="checkbox"/>	Success	My computer	Project_dataset.cs	TMH29490.PRC	tmh29490	649	0

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- **Better Inventory Accuracy:** With solid inventory management, you know what's in stock and order only the amount of inventory you need to meet demand.
- **Reduced Risk of Overselling:** Inventory management helps track what's in stock and what's on backorder, so you don't oversell products.
- **Cost Savings:** Stock costs money until it sells. Carrying costs include storage handling and transportation fees, insurance and employee salaries. Inventory is also at risk of theft, loss from natural disasters or obsolescence.
- **Avoiding Stockouts and Excess Stock:** Better planning and management helps a business minimize the number of days, if any, that an item is out of stock and avoid carrying too much inventory. Learn more about solving for stockouts in our "Essential Guide to Inventory Control."
- **Greater Insights:** With inventory tracking and stock control, you can also easily spot sales trends or track recalled products or expiry dates.

DISADVANTAGES:

- **Expensive for Small Businesses:** The cost of inventory management software can seem daunting to a small business, but the investment often pays for itself in increased profits and improved customer loyalty. Additionally, cloud-based systems have made software that was once the domain of large enterprises available to smaller businesses.
- **Complex to Learn:** Business software is sometimes tricky to learn. However, managers can help by investing in online training to quickly bring users up to speed.
- **Risk of System Crashes:** Software does crash. However, you can remove the risk of data and productivity loss by using cloud-based platforms.
- **Malicious Hacks:** Malicious hacks are a risk to all businesses. The Internet of Things (IoT) adds even more complexity. Cloud-based software typically has greater security than a single company would offer on its own because of the risk a breach would have on the vendor.

11. CONCLUSION

- The efficient way of finding products for the people is implemented using the inventory website that is hosted on IBM Cloud platform.
- To ensure the smooth functioning of the web site operation. I have hosted the website in IBM Db2 & Kubernetes Cluster to make sure the operations are running successfully Cloud lambda function is used and to deploy the application IBM Db2 service is used.

12. FUTURE SCOPE

- Upgrading the UI that is more user-friendly which will help many users to access the website and also ensures that many stocks can be added into the list.
- Using elastic load balancer, it helps to handle multiple requests at the same time which will maintain the uptime of the website with negligible downtime

13. APPENDIXES

13.1.SAMPLE SOURCE CODE:

main.py

```
From flask import Flask,render_template,request,redirect, url_for,flash
app = Flask(__name__)
import ibm_db
from flask_login import
login_user,current_user,logout_user,login_required,LoginManager,UserMixin
import datetime
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
dsn_hostname = "b0aebb68-94fa-46ec-
a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.d
atabases.appdomain.cloud"
dsn_uid = "tmh29490"
dsn_pwd = "kuO6Gg9kukf26CAb"
dsn_driver = "{IBM DB2 ODBC DRIVER}"
dsn_database = "bludb"
dsn_port = "32459"
dsn_protocol = "TCPIP"
dsn_security = "SSL"
```

```

dsn = (
    "DRIVER={0};"
    "DATABASE={1};"
    "HOSTNAME={2};"
    "PORT={3};"
    "PROTOCOL={4};"
    "UID={5};"
    "PWD={6};"

    "SECURITY={7};").format(dsn_driver,
    dsn_database, dsn_hostname, dsn_port,
    dsn_protocol, dsn_uid,
    dsn_pwd,dsn_security)

print(dsn)

try:
    conn = ibm_db.connect(dsn, "", "")

    print ("Connected to database: ", dsn_database, "as user: ", dsn_uid, "on host: ", dsn_hostname)

except:
    print ("Unable to connect: ", ibm_db.conn_errormsg() )

SECRET_KEY = 'Jeevaneshwaran'

@app.route("/",methods=['GET', 'POST'])
def home():

```

```

if request.method == 'POST':
    name=request.form.get('username')
    password=request.form.get('password')
    print(name,password)
    sql = "SELECT * FROM users
WHERE user_name =? AND
password=?"

    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,name)
    ibm_db.bind_param(stmt,2,password)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    print (account)
    if account:
        return redirect(url_for('dashboard'))
    else:
        msg='invalid user name and password'
        return render_template('loginpage.html',msg=msg)
else:
    return render_template('loginpage.html')

@app.route("/register",methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        name=request.form.get('name')

```



```

user_name=request.form.get('lastname')

    email=request.form.get('gmail')

    phone=request.form.get('ph_no')

    password=request.form.get('password')

confirm= request.form.get('confirm_password')

    print(confirm)

    if password==confirm:

        sql ="SELECT id FROM users ORDER BY ID DESC limit 1"

stm=ibm_db.exec_immediate(conn,sql)

    while ibm_db.fetch_row(stm) != False:

        count=ibm_db.result(stm,0)

        print(count)

        insert=("insert into users values
({int(count)+1 }, '{name}', '{user_name}',
'{email}', '{password}', '{phone}')"

        table=ibm_db.exec_immediate(conn,insert)

        return redirect(url_for('home'))

    else:

        msg='invalid user name and password'

        return render_template('register.html',msg=msg)

    else:

        return render_template('register.html')

```

```

@app.route("/searchproduct",methods=['
GET', 'POST'])
def searchproduct():
    if request.method == 'POST':
        Product=request.form.get('search')
        sql = "SELECT * FROM products WHERE product =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,Product)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print (account)
        return render_template('searchproduct.html',product=account)
    else:
        return render_template('searchproduct.html')

@app.route("/viewbill",methods=['GET', 'POST'])
def viewbill():
    sql ="SELECT * FROM billing"
    stmt = ibm_db.exec_immediate(conn, sql)
    bill=[]
    amount=0
    while ibm_db.fetch_row(stmt) != False:
        dic=dict()
        dic['invoice']=ibm_db.result(stmt, 1)
        dic['product']=ibm_db.result(stmt, 3)

```

```

        dic['price']=ibm_db.result(stmt, 4)

        price=ibm_db.result(stmt, 4)

        dic['quantity']=ibm_db.result(stmt,5)

        quantity=ibm_db.result(stmt,5)

        dic['total']=int(price)*int(quantity)

        total=int(price)*int(quantity)

        amount +=total

        bill.append(dic)

    print(bill)

    print(amount)

    return render_template('viewbill.html',datas=bill)

@app.route("/minimum",methods=['GET', 'POST'])
def minimum():

    sql ="SELECT  * FROM products"

    # stmt = ibm_db.prepare(conn, sql)

    # ibm_db.bind_param(stmt,1,name)

    # ibm_db.bind_param(stmt,2,password)

    stmt = ibm_db.exec_immediate(conn, sql)

    datas=[]

    while ibm_db.fetch_row(stmt) != False:

        dic=dict()

        dic['product']=ibm_db.result(stmt, 0)

        dic['stock']=ibm_db.result(stmt, 1)

        dic['price']=ibm_db.result(stmt, 2)

```

```

        dic['alert']=ibm_db.result(stmt, 3)

        datas.append(dic)

        print(datas)

        # ibm_db.execute(stmt)

        # account = ibm_db.fetchall(stmt)

        # print (account)

    return render_template('minimum.html',datas=datas)

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

    sql ="SELECT * FROM products"

    stmt = ibm_db.exec_immediate(conn, sql)

    datas=[]

    low=0

    count=0

    while ibm_db.fetch_row(stmt) != False:

        dic=dict()

        dic['product']=ibm_db.result(stmt, 0)

        dic['stock']=ibm_db.result(stmt, 1)

        stock=ibm_db.result(stmt, 1)

        dic['price']=ibm_db.result(stmt, 2)

        dic['alert']=ibm_db.result(stmt, 3)

        alert=ibm_db.result(stmt, 3)

        if int(stock)< int(alert):

            low += 1

```

```

    datas.append(dic)

    count+=1

print(datas)

sql ="SELECT  * FROM billing"

stmt = ibm_db.exec_immediate(conn, sql)


bill=[]

amount=0

bill_count=0

while ibm_db.fetch_row(stmt) != False:

    dic=dict()

    dic['invoice']=ibm_db.result(stmt, 1)

    dic['product']=ibm_db.result(stmt, 3)

    dic['price']=ibm_db.result(stmt, 4)

    price=ibm_db.result(stmt, 4)

    dic['quantity']=ibm_db.result(stmt,5)

    quantity=ibm_db.result(stmt,5)

    dic['total']=int(price)*int(quantity)

    total=int(price)*int(quantity)

    amount +=total

    bill_count+=1

    bill.append(dic)

print(bill)

print(amount)

```

```

    return
    render_template('dashboard.html',datas=datas,low=low,amount=amount,count=
count,bill_count=bill_count)

@app.route("/billing",methods=['GET', 'POST'])
def billing():

    date=datetime.datetime.today().date()

    if request.method == 'POST':

        invoice=request.form.get('invoice')

        date=request.form.get('date')

        product=request.form.get('product')

        quantity=request.form.get('quantity')

        price=request.form.get('price')

        insert=("insert into billing values (
{invoice[-1]}, '{invoice}', '{date}',
'{product}', {int(quantity)}, {int(price)})")

        table=ibm_db.exec_immediate(conn,insert)

        sql = "SELECT * FROM products WHERE product =?"

        stmt = ibm_db.prepare(conn, sql)

        ibm_db.bind_param(stmt,1,product)

        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        try:

            count=account['STOCK']

            alert=account['Alert']

            update_count=int(count)-int(quantity)

```

```

if int(update_count) < int(alert):

    print('email code')

    #write sendgrid email code here

    message = Mail(

        from_email='jeffreylawrence2121@gmail.com',

        to_emails='mjeevan5791@gmail.com',

        subject='Sending with Twilio SendGrid is Fun',

        html_content='<strong>and easy to do anywhere, even with
        Python</strong>')

    try:

        sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))

        response = sg.send(message)

        print(response.status_code)

        print(response.body)

        print(response.headers)

    except Exception as e:

        print(e)

```

```

update=("UPDATE products SET stock = {update_count} WHERE
product = '{product}'")

```

```

table=ibm_db.exec_immediate(conn,update)

```

```

return redirect(url_for('dashboard'))

```

```

except:

```

```

    return redirect(url_for('dashboard'))

```

```

else:

```

```

    return render_template('billing.html',date=date,invoice=invoice_no())

```

```

@app.route("/addproduct",methods=['GET', 'POST'])
def addproduct():
    if request.method == 'POST':
        Product=request.form.get('Product')
        Stock=request.form.get('Stock')
        Price=request.form.get('Price')
        Alert=request.form.get('Alert')
        print(Product,Stock,Price,Alert)
        insert=("insert into products values (
'{Product}', {int(Stock)}, {int(Price)},
{int(Alert)})")
        table=ibm_db.exec_immediate(conn,insert)
        return redirect(url_for('dashboard'))
    else:
        return render_template('addproduct.html')
def invoice_no():
    sql ="SELECT id FROM billing ORDER BY ID DESC limit 1"
    stm=ibm_db.exec_immediate(conn,sql)
    while ibm_db.fetch_row(stm) != False:
        count=ibm_db.result(stm,0)
    if count:
        return f'bill00{int(count)+1}'
    else:
        return 'bill001'
app.run(debug=True)

```


13.2 GITHUB Link:

<https://github.com/IBM-EPBL/IBM-Project-29473-1660125855>