Table of Content

CHAPTER 1

INTRODUCTION

1.1. PROJECT OVERVIEW

The objective of the project is to deliver an efficient inventory management system whose main functionality apart from calculating the inventory include predicting the requirement for the next demand and if there is a “Special Occasion” then accordingly the manager selects the particular occasion and extra requirements are added to the next issuing order to the vendors which needs to be approved by the manager.

The product also aims to keep track of the shelf life of resources. If any resource nears the end of its shelf life, it would acknowledge to the manager (admin) the details of the quantity that is near its expiration date.

The success criteria depend on:

- The accuracy in keeping the inventory levels.
- The accuracy in predicting the requirements of the next demand.
- The accuracy in relating recipes to their respective constituents.
- Ease of use when it comes to updating inventory levels and placing orders to vendors.

1.2. PURPOSE

The Inventory Management System is a real-time inventory database capable of connecting multiple stores. This can be used to track the inventory of a single store or to manage the delivery of stock between several branches of a larger franchise. However, the system merely records sales and restocking data and provides warning of low stock at any location through email at a specified interval.

The goal is to reduce the stress of tracking rather than to holder all store maintenance. Further features may consist of the ability to create reports of sales, but again the explanation is left to the management. In addition, since theft does occasionally occur, the system provides solutions for confirming the store inventory and for correcting stock quantities.

Production units use an inventory management system to reduce their transport costs. The system is used to track products and parts as they are transported from a seller to a storeroom, between storerooms, and finally to a retail location or directly to a customer.

The inventory management system is used for various purposes, including:

- Maintaining and recording the information between too much and too little inventory in the company.
- Keep track of inventories as it is transported between different locations.
- Recording product information in a warehouse or other location.
- Having a record of Picking, packing, and selling products from a warehouse.
- Reduction of product obsolescence and decay.
- Avoiding out-of-stock situations.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing problem

Inventory Management is a crucial aspect of managing a company successfully. Inventory is a vital part of current assets mainly in manufacturing concerns. Huge funds are committed to inventories as to ensure smooth flow of production to meet consumer demand. Maintaining Inventory also involves holding or carrying costs along with opportunity cost. An efficient inventory management ensures continuous production by maintaining inventory at a satisfactory level. It also minimizes capital investment and cost of inventory by avoiding stock-pile of product. Efficient and Effective Inventory Management goes a long way in successful running and survival of business firm. Keywords: Inventory Management, Survival, Working Capital, Liquidity and Profitability. Introduction: The present paper focuses on the review of existing literature in the field of Inventory Management which helps in capturing both conceptual and research based studies. A Number of studies have been conducted to find the determinants of investment in inventories and the process is still going on. The present study is the summary of critical points of a particular topic consisting of essential findings as well as theoretical and methodological contributions. My paper shall discuss conceptual studies of both Indian and other nationals.

2.2 Reference

Abramovitz and Modigliani (1957) They highlighted the relationship between capacity utilization and inventory investment. Existing stock of inventories was expected to adjust to the desired levels. Thus the variable, existing stock of inventories, was essential to be negatively related with the desired stock. The result was that there is positive relation among the ratio of inventory to sales and inventory investment. High ratio of stocks to sales in the past suggests requirement of high levels of inventories in the past and promising high investment in inventories in the current period also. Krishna Murthy (1964) Study was aggregative and dealt with inventories in the private sector of Indian economy as a whole for the period 1948-61. This study used

sales to represent demand for the product and suggested the importance of accelerator. Short term rate of interest had also been found to be significant. R.S. Chadda (1964) Study had been made on inventory management practices of Indian companies. The analysis suggested application of modern scientific inventory control techniques like operations research. These modern scientific techniques furnish opportunities for the companies, Companies can minimize their investment in inventory but there is continuous flow of production. He argued that industrially advanced countries, like, USA, were engaged in developing highly sophisticated mathematical models and techniques for modernizing and redefining the existing tools of inventory investment. National Council of Applied Economic Research (NCAER) (1966) Conducted a study in 1966 regarding working capital management of three industries namely cement, fertilizer and sugar. This study mainly devoted to ratio analysis of composition, utilization and financing of working capital for the period of 1959 to 1963. The study reveals that inventory constituted a major portion of working capital i.e. 74.06 per cent in the sugar industry followed by cement industry (63.1%) and fertilizer industry (59.58%). It was observed that inventory had not managed properly. So far as the utilization of working capital was concerned, cement and fertilizer industry had better implementation of working capital. The sugar industry had huge accumulation of stocks so there was inefficient utilization of working capital heavily. Krishnamurty and Sastry (1970) It is the most comprehensive study on manufacturers' inventories. They used the CMI data and the consolidated balance sheet data of public limited companies published by the RBI, in order to analyse each of the major components, like the raw materials, goods-in-process and finished goods, for 21 industries over the period ranging from 1946-62. The study was a time series one although there were some inter-industry cross-section analyses that were carried out in the analysis. The Accelerator represented by change in sales, bank finance and short-term interest rate was found to be an important determinant. The utilisation of productive capacity and price anticipations was also found to be relevant in the study. George (1972) It was the study on cross section analysis of balance sheet data of 52 public limited companies for the period of 1967- 70. Accelerator, internal and external finance variables were considered in the formulation of equations for raw materials including goods-in-process inventories. However, equations for finished goods inventories conceive only output variable. Deliberation was given on accelerator and external finance variables. Mishra (1975) It is the study of six major public sector enterprises. He concluded that (i) inventory constitutes the most important component of working capital of public

enterprises (ii) efficiency of working capital funds employed in receivables is terribly low in the selected enterprises and (iii) In all units both the current assets and the quick ratios are greater than their standards. Enterprises need proper control on receivables. Lambrix and Singhvi (1979) Adopted working capital cycle approach in working capital management, also suggested that investment in working capital can be optimized and cash flows can be improved by reducing the time frame of physical flow starting from the receipt of raw material to the shipment of finished goods, i.e. inventory management, and by improving the terms and conditions on which firm sells goods as well as receipt of cash Lal (1981) He studied Modi Steels Limited as a case study, his study focused on inventory management. He originated a model which involve price variable in inventory management; earlier price variable in inventory was not considered in that company. The analysis recommended solid policies, which would look after internal and external factors, ultimately it would help in bringing in efficient working capital management. Farzaneh (1997) Presented a mathematical model, to assist the companies in their decision to switch from EOQ to JIT purchasing policy. He defines JIT as "to produce and deliver finished goods just in time to be sold, sub-assemblies just in time to be assembled in goods and purchased material just in time to be transformed into fabricated parts". He highlights that the EOQ model focuses on minimizing the inventory costs rather than minimizing the inventory. Under the ideal condition where all the conditions meet, it is economically better off to choose the JIT over the EOQ because it results in purchase price, ordering cost. Rich Lavely (1998) Asserts that inventory means "Piles of Money" on the shelf and the profit for the firm. However, he notices that 30% of the inventory of most retail shops is dead. Therefore, he argues that the inventory control is facilitate the shop operations by reducing rack time and thus increases profit. He also elaborates the two types of inventory calculations that determine the inventory level required for profitability. The two calculations are "cost to order" and "cost to keep". Finally, he proposes seven steps to inventory control. Dave Piasecki (2001) He focused on inventory model for calculating the optimal order quantity that used the Economic Order Quantity method. He points out that many companies are not using EOQ model because of poor results resulted from inaccurate data input. He says that EOQ is an accounting formula that determines the point at which the combination of order costs and inventory costs are the least. He high lights that EOQ method would not conflict with the JIT approach. He further elaborates the EOQ formula that includes the parameters such as annual usage in unit, order cost and carrying cost. Finally, he proposes several steps to follow in implementing the EOQ model. The

limitation of this literature is that it does not elaborate further relationship between EOQ and JIT. It does not associate the inventory turns with the EOQ formula and fails to mention the profit gain with the quantity is calculated. Gaur, Fisher and Raman (2005) In their study examined firm-level inventory behaviour among retailing companies. They took a sample of 311 public-listed retail firms for the years 1987–2000 to examine the relationship of inventory turnover with gross margin, capital intensity and sales surprise. They observed that inventory turnover for retailing firms was positively related to capital intensity and sales surprise while inversely associated with gross margins. They also suggested models that yield an alternative metric of inventory productivity, adjusted inventory turnover that can be used in study of performance analysis and managerial decision-making. S. Singh (2006) Analysed the inventory control practices of single fertilizer company named IFFCO. He statistically examined the inventory system with consumption, sales and other variables along with growth of these variables and inventory patterns. He concluded that an increase in components of inventory lead to an increase in the proportion of inventory in current assets. A special focus was made on stores and spares in order to calculate excess purchases resulting in loss of profit. Pradeep singh (2008) In his study made an attempt to examine the inventory and working capital management of Indian Farmers Fertilizer Cooperative Limited (IFFCO) and National Fertilizer Limited (NFL). He concluded that the overall position of the working capital of IFFCO and NFL is satisfactory. But there is a need for improvement in inventory in case of IFFCO. However inventory was not properly utilized and maintained bay IFFCO during study period. The management of NFL must try to properly utilize the inventory and try to maintain the inventory as per the requirements. So that liquidity will not interrupt. Capkun, Hameri and Weiss (2009) Statistically analysed the relationship between inventory performance and financial performance in manufacturing companies using the financial information of a large sample of USbased manufacturing firms over a 26-year period, that is, 1980 to 2005. They inferred that a significant relationship existed between inventory performance along with the performance of its components and profitability. Raw material inventory performance was highly correlated to gross profit and operating profit. Work in progress inventory was highly correlated to gross profit measures while finished goods inventory performance was more correlated with operating profit measures. Gaur and Bhattacharya (2011) Attempted to study the linkage between the performance of the components of inventory such as raw material, work in progress and finished goods and financial performance of Indian manufacturing firms. The study revealed that finished goods

inventory as inversely associated with business performance while raw material inventory and work in progress did not have much effect on same. They emphasised that instead of focusing on total inventory, an attempt should be made to concentrate on individual components of inventory so as to adequately manage the same. They concluded that managers not paying heed to inventory performance may become weak in combating competitors. Eneje et al (2012) Investigated the effects of raw materials inventory management on the profitability of brewery firms in Nigeria using a cross sectional data from 1989 to 2008 which was gath ered for the analysis from the annual reports of the sampled brewery firms. Measures of profitability were examined and related to proxies for raw materials inventory management by brewers. The Ordinary Least Squares (OLS) stated in the form of a multiple regression model was applied in the analysis. The study revealed that the local variable raw materials inventory management designed to capture the effect of efficient management of raw material inventory by a company on its profitability is significantly strong and positive and influences the profitability of the brewery firms in Nigeria. They concluded that efficient management of raw material inventory is a major factor to be contained with by Nigerian brewers in enhancing or boosting their profitability. Nyabwanga and Ojera (2012) They Highlighted the association between inventory management practices and business performance of smallscale enterprises (SSEs), in Kisii Municipality, Kisii County, Kenya. They used a cross-sectional survey study based on a small sample size of 79 SSEs. The study inferred that inventory comprised the maximum portion of working capital, and improper management of working capital was one of the major reasons of SSE failures. The empirical results disclosed that a positive significant relationship existed between business performance and inventory management practices with inventory budgeting having the maximum influence on business performance ensued by shelf-space management. The study suggested that by following effective inventory management practices business performance can be enhanced. Sahari, Tinggi and Kadri (2012) Empirically analysed the relationship between inventory management and firm performance along with capital intensity. For the purpose they took a sample of 82 construction firms in Malaysia for the period 2006–2010. Using the regression and correlation analysis methods, they deduced that inventory management is positively correlated with firm performance. In addition, the results indicate that there is a positive link between inventory management and capital intensity Soni (2012) Made an in depth study of practices followed in regard to inventory management in the engineering goods industry in Punjab. The analysis used a sample of 11 companies for a period five years,

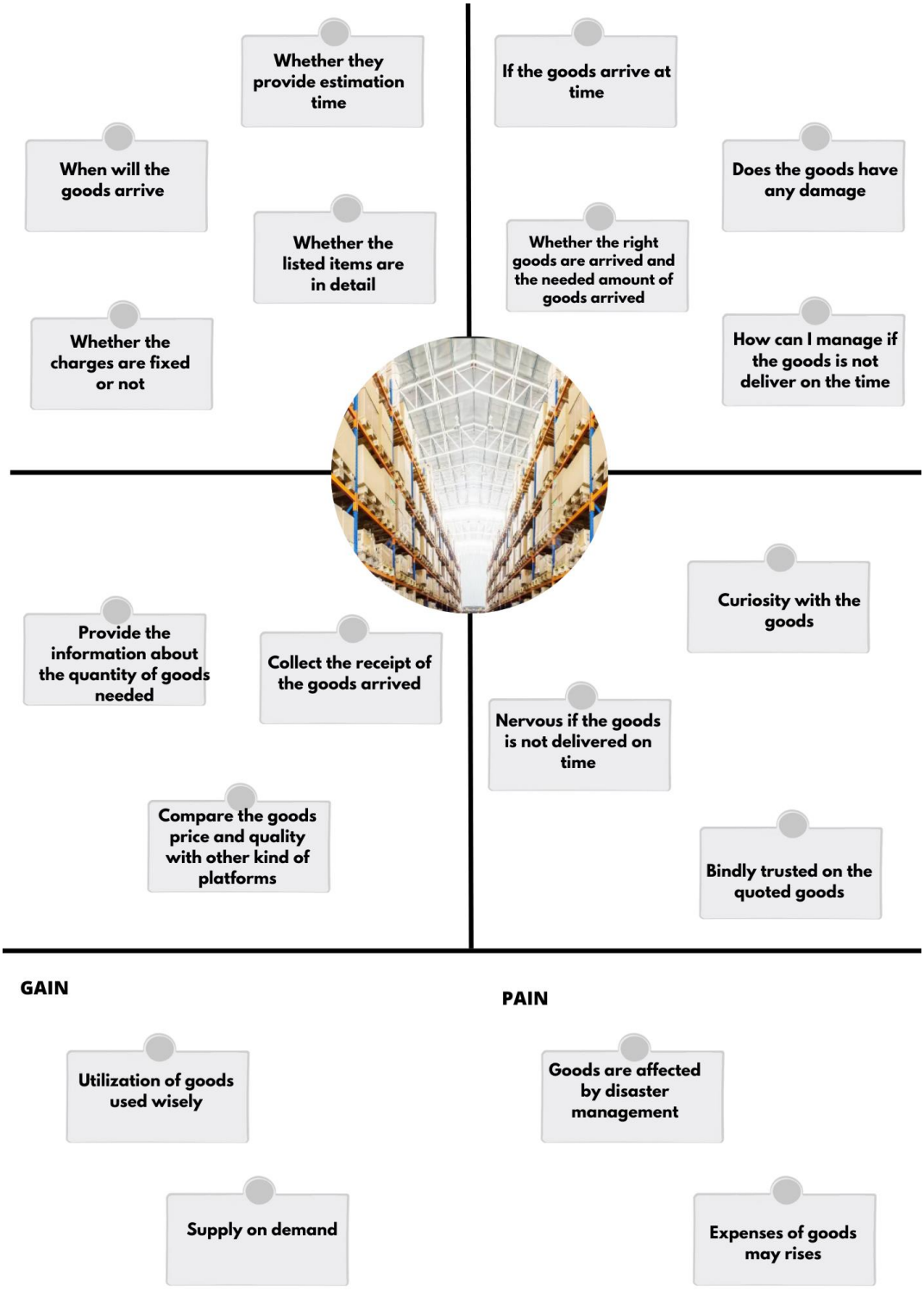
that is, 2004–2009 and was done using panel data set. The adequate and timely flow of inventory determines the success of an industry. She concluded that size of inventory enhanced marginally over the period as compared to a hike in current assets and net working capital. Inventories constituted half of the working capital which was due to over stocking of inventory as a result of low inventory turnover especially for finished goods and raw materials. Rise in sales and favourable market conditions lead to a rise in inventory levels. It was also inferred that sales increased more as compared to inventory Lwiki et al (2013) A survey conducted on all the eight (8) sugar manufacturing firms in Kenya established that there is generally positive correlation between each of inventory management practices. Specific performance indicators were proved to depend on the level of inventory management practices. They established that Return on Equity had a strong correlation with lean inventory system and strategic supplier partnerships. As such, they concluded that the performance of sugar firms could therefore be stated as being a function of their inventory management practices. Panigrahi (2013) Undertook an in-depth study of inventory management practices followed by Indian cement companies and its affect on working capital efficiency. The study also investigated the relationship between profitability and inventory conversion days. The study, using a sample of the top five cement companies of India over a period of 10 years from 2001 to 2010, concluded that a considerable inverse linear relationship existed between inventory conversion period and profitability. Madishetti and Kibona (2013) Found that a well designed and executed inventory management contributes positively to a small or medium-sized enterprises (SMEs) profitability. They studied the association between inventory conversion period and profitability and the impact of inventory management on SMEs profitability. They took a sample of 26 Tanzanian SMEs, and used the data from financial statements for the period 2006–2011. Regression analysis was adopted to determine the impact of inventory conversion period over gross operating profit. The results cleared out that significant negative linear relationship occurred between inventory conversion period and profitability. Srinivas Rao Kasisomayajula(2014) An analytical study was conducted on" Inventory Management in Commercial Vehicle Industry In India". A sample of five companies' was selected for study. The study concluded that all the units in the commercial vehicle industry have significant relationship between Inventory and Sales. Proper management of inventory is important to maintain and improve the health of an organization. Efficient management of inventories will improve the profitability of the organization. Edwin Sitienei and Florence Memba(2015) Conducted a study on Effect of Inventory

Management on profitability of Cement Manufacturing Companies in Kenya. The study concluded that Gross profit margin is negatively correlated with the inventory conversion period, Increase in sales, which denotes the firm size enriches the firm's inventory levels, which pushes profits upwards due to optimal inventory levels. It is also noted that firms inventory systems must maintain an appropriate inventory levels to enhance profitability and reduce the inventory costs associated with holding excessive stock in warehouses. REFERENCES [1] Abramovitz & Modigliani, Franco (1957), "Business Reasons for Holding Inventories and Their Macro Economic Implications", Problems of Capital Formation, Studies in Income and Wealth, Vol. 19, NBER, pp. 495-511. [2] Anichebe, N. A. & Agu, O. A. (2013). Effect of Inventory Management on Organizational Effectiveness. Information and Knowledge Management, vol.3 ,iss.8, pp. 92 – 100. [3] Chadda, R.S (1964), "Inventory Management in India", Allied Publishers, Bombay, 1964. [4] Capkun, Vedran, Hameri, Ari-Pekka & Weiss, Lawrence A. (2009). On the relationship between inventory and financial performance in manufacturing. International Journal of Operations & Production Management, vol.29, iss.8, pp.789–806. [5] Edwin Sitienei, Florence Memba(2015-16) " The Effect of Inventory Management on Profitability of Cement Manufacturing Companies in Kenya: A Case Study of Listed Cement Manufacturing Companies in Kenya" International Journal of Management and Commerce Innovations Vol. 3, Iss. 2, pp. 111- 119. [6] Eneje, B. C., Nweze, A .U. & Udeh, A. (2012). Effect of Efficient Inventory Management on Profitability: Evidence from Selected Brewery Firms in Nigeria. International Journal of Current Research, vol.4, iss.11, pp.350-354. [7] Gaur, Jighyasu & Bhattacharya, Sourabh. (2011). The relationship of financial and inventory performance of manufacturing firms in Indian context. California Journal of Operations Management, vol. 9, iss.2, pp.70–77. [8] Gaur, V., Fisher, M. & Raman, A. (2005)." An econometric analysis of inventory turnover performance in retail services". Management Science, vol.5,iss.2, pp.181–194. [9] George, P. V (1972), "Inventory Behaviour and Efficacy of Credit Control", Anvesak, No.2, Vol.II, 1972, pp. 168-175. [10] Krishnamurty K., "Private Investment Behaviour in India: A Macro Time Series Study", Arthaniti, January 1964. [11] Krishnankutty, Raveesh. (2011). Panel data analysis on retail inventory productivity. The Economic Research Guardian 1(1),pp.16–23. [12] Krishnamurthy S. & Sastry D.U.,inventories in Indian Manufacturing, Institute of Economic Growth…, Books Ltd., Mumbai, 1970; and Investment and Financing in Corporate Sector in India, Tata McHill publishing Company, New Delhi, 1975. [13] Lal, A.B (1981), "Inventory Models and Problems of Price Fluctuation", Shree Publishing

House, New Delhi, 1981. [14] Lambrix, R.J and Singhvi, S.S (1979), “Managing the Working Capital Cycle”, Financial Executive, June 1979, pp. 32-41. [15] Lieberman, M.B. & Demeester, L. (1999). Inventory reduction and productivity growth: Linkages in the Japanese automotive industry. Management Science,vol.45, iss.4, pp.466–476. [16] Madishetti, Srinivas & Kibona, Deogratias. (2013). Impact of inventory management on the profitability of SMEs in Tanzania. International Journal of Research in Commerce & Management, vol.4,iss.2, pp.1–6. [17] Mishra (1975), “Problems of Working Capital with special reference to selected Public Sector Undertakings in India, Somiya Publications Private Limited, 1975. [18] NCAER, Structure of Working Capital, New Delhi, 1966. [19] Nyabwanga, Robert Nyamao & Ojera, Patrick. (2012). Inventory management practices and business performance for small scale enterprises in Kenya. KCA Journal of Business Management, vol.4,iss.1, pp.11–28. [20] Panigrahi, Ashok K. (2013). Relationship between inventory management and profitability: An empirical analysis of Indian cement companies. Asia Pacific Journal of Marketing & Management Review, vol.2,iss.7, pp.107–120. [21] Pradeep Singh(2008),” Inventory and Working Capital Management- An Empirical Analysis”, The ICFAI Journal of Accounting and Research, Vol.VII, NO.2, pp.53-73 [22] Sahari, Salawati, Tinggi, Michael & Kadri, Norlina. (2012). Inventory management in Malaysian construction firms: Impact on performance. SIU Journal of Management, vol.2,iss.1, pp.59–72. [23] Sanjiv Mittal, R.K. Mittal ,Gagandeep Singh, Sunil Gupta (2014).”Inventory Management in Fertiliser Industry of India: An Empirical Analysis” Asia-Pacific Journal of Management Research and Innovation ,vol.10,iss.4,pp. 291–303. [24] Singh, Sukhdev. (2006). Inventory control practices in IFFCO.The Management Accountant, vol.41,iss.7, pp.577–582. [25] Soni, Anita. (2012). Inventory management of engineering goods industry in Punjab: An empirical analysis. International Journal of Multidisciplinary Research, vol.2,iss.2, pp.247–261. [26] Srinivasa Rao Kasisomayajula(2014) “An Analytical Study on Inventory Management in Commercial Vehicle Industry in India”, International Journal of Engineering Research, Vol.3, Iss.6, pp.378-383. [27] Viplesh Shardeo(2015), “Impact of Inventory Management on the Financial Performance of the firm” IOSR Journal of Business and Management (IOSR JBM). Vol. 17, Iss. 4, pp. 01-12

CHAPTER 3
3. IDEATION & PROPOSED SOLUTION

3.1Empathy Map Canvas



3.2 Ideation & Brainstorming

3.3 Proposed Solution

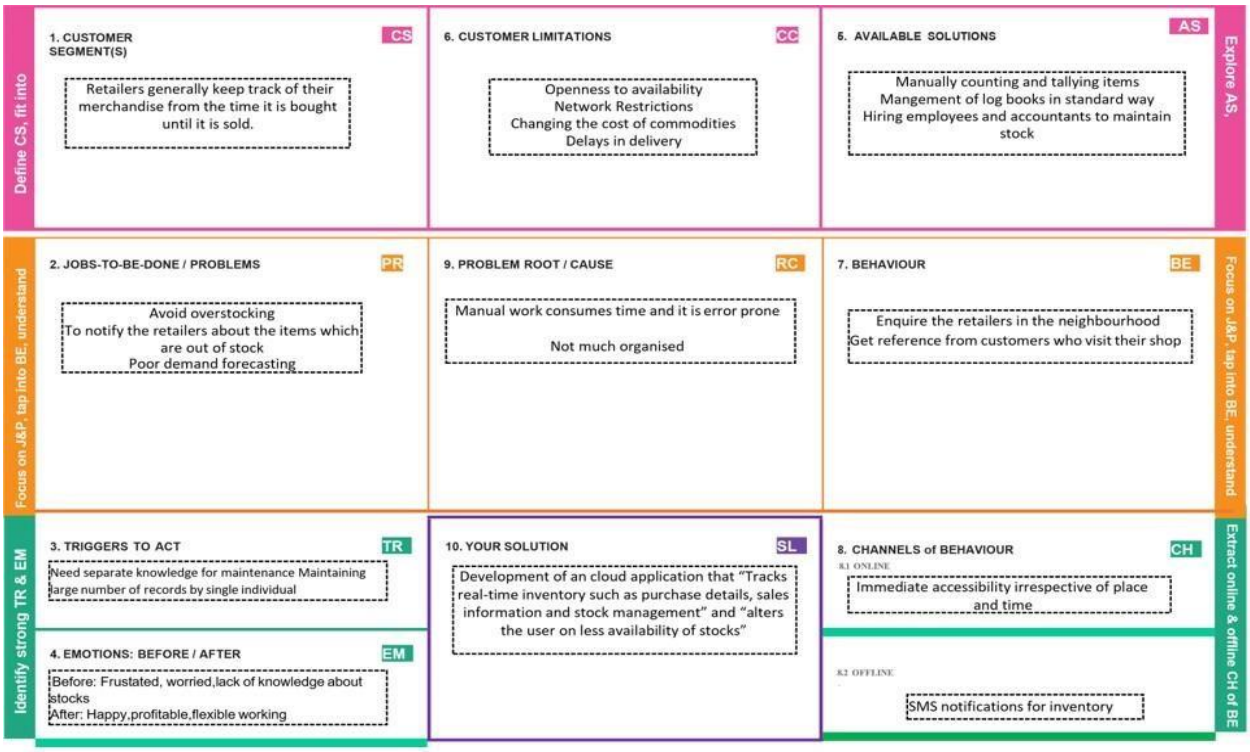
S.No.	Parameter	Description
☐	Problem Statement (Problem to be solved)	• The retailers generally facing issues in recording the stocks and its threshold limit available. • The retailers doesn't know which product is getting expired and when it is being expired. • The retailers couldn't track the availability of all the stocks up-to date. • The customers are not satisfied with the retailers store since it doesn't have enough supplements and the deliveries were not made on time.
☐	Idea / Solution description	• This proposed system will have a daily update system whenever a product is sold or it is renewed more. • The system will have an alert triggered to indicate both the expired product and soon going to expire products. • The product availability is tracked daily and an alert system in again kept on to indicate those products which falls below the threshold limit.

		• All the customers can register their accounts after which they will be given a login credentials which they can use whenever they feel like buying the stocks. • The application allows the customers to know all the present time available stocks and also when the new stock will be available on the store for them to buy. • Tracking the order have become easy with this application for both the retailers and the customers.

	Novelty / Uniqueness	• Certain machine learning algorithms are used to predict the seasonal high selling products which can be made available during that time. • Prediction of the best selling brand of all certain products based on their popularity, price and customer trust and satisfaction will be implemented. • 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. • Notification will be sent to the customers who buys any certain products regularly when the new arrivals are stocked up. • Exclusive discounts and offers are given for regular customers to keep them engaged with the store regularly.
□	Social Impact / Customer Satisfaction	• The customers will be highly satisfied since the wasting of time while searching for an unavailable product is reduced.

		• The work load of the retailers will be minimized if the system is automated every day and during every purchase. • The customer satisfaction will be improved for getting appropriate response from the retailers and that too immediately.
□	Business Model (Revenue Model)	• Hereby we can provide a robust and most reliable inventory management system by using: 1. ML algorithms for all the prediction purposes using all the past dataset since datasets are undoubtedly available in huge amounts. 2. Can deploy the most appropriate business advertising models. 3. To establish a loss preventing strategy. 4. And to ensure the all time, any where availability of products system. 5. Usage of freebies business strategy for dragging the customer's attention.
□	Scalability of the Solution	• This system can even work more efficiently with large volume of data. • Implementation of anyone and anywhere using system can be helpful for even a commoner to buy the products. • Daily and Each time purchase updation of the stock for preventing inventory shrinkage. • Direct chat system with the retailers and the customers for providing best customer service.

3.4 Proposed Solution Fit



CHAPTER 4
REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through own application Form Registration through Gmail Registration through LinkedIN Registration through Google Docs.
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login through User name and password. Login through mail I'D and password. Login through OTP through mail I'd and password. Login through Phone number.
FR-4	Records of the products	Product name Product category Product I'd Stock Count Vendor details
FR-5	Login details	Login Details along with time through E-mail. Login Details along with time through phone number.
FR-6	Updation of inventory Details.	Update through E-mail Update through User account.
FR-7	Unavailability Alert	Alert Message through mail or phone number.
FR-8	Monitoring of stock	Audit monitoring through incoming and outgoing stock.
FR-9	Database	Usage of standard database for storing the data.

4.2 Non-Functional requirements

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

FR No.	Non-Functional Requirement	Description

NFR-1	**Usability**	• Once retailers successfully log in to the application they can update their inventory details, also users will be able to add new stock by submitting essential details related to the stock. They can view details of the current inventory. The System will automatically send an email alert to the retailers if there is no stock found in their accounts. So that they can order new stock. • It can use by wide variety of client as it is very simple to learn and not complex to proceed • Easy to use, User-friendly and Responsive.
NFR-2	**Security**	• Applications have been developed to help retailers track and manage stocks related to their own products. The System will ask retailers to create their accounts by providing essential details. Retailers can access their accounts by logging into the application. With Registered Mail id only retailers can log into the application. So it provide authentication. • We are using login for the user and the information will be hashed so that it will be very secure to use.
NFR-3	**Reliability**	• It will be reliable that it can update with very time period so that the accuracy will be good.
NFR-4	**Performance**	• User can track the record of goods available using the application. Inventory tracking helps to improve inventory management and ensures that having optimal stock available to fulfill orders.Reduces manpower , cost and saves time. Emails will be sent automatically While stocks are not available.Makes the business process more efficient.Improves organizations performance. • It will be perform fast and secure even at the lower bandwidth

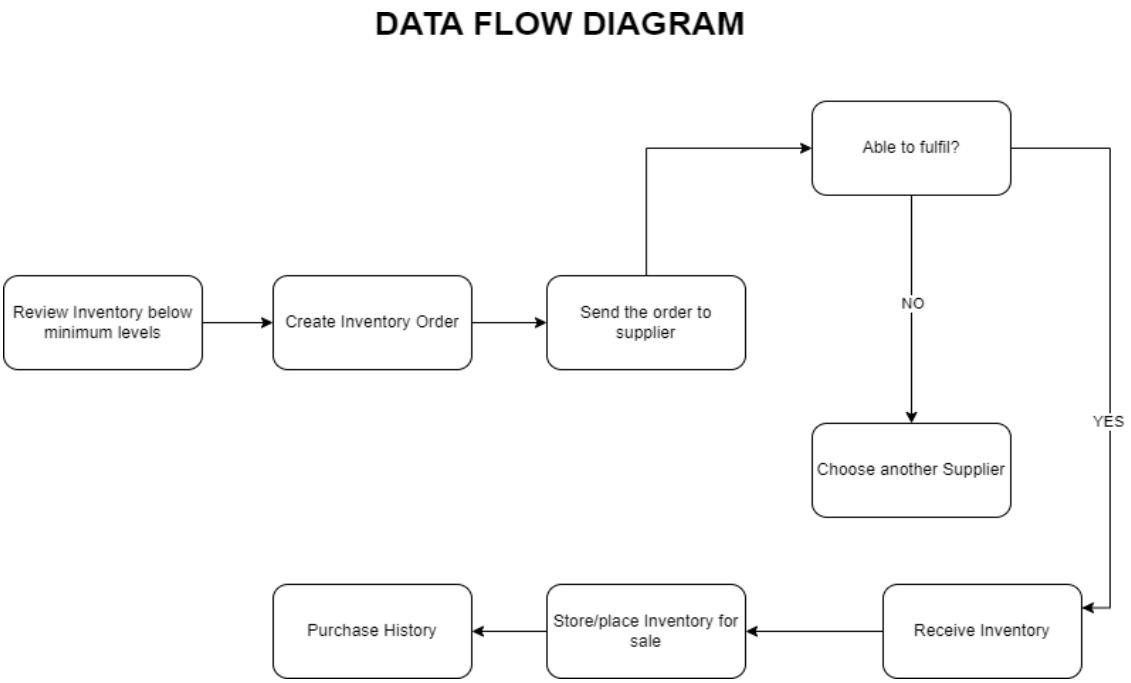
NFR-5	**Availability**	• The availability of product is just one way in which an inventory management system creates customer satisfaction. Inventory management systems are designed to monitor product availability, determine
		purchasing schedules for better customer interaction. • Prediction will be available for every user but only for premium user news,database and price alert will be alert
NFR-6	**Scalability**	• Scalability is an aspect or rather a functional quality of a system, software or solution.This proposed system for inventory management system can accommodate expansion without restricting the existing workflow and ensure an increase in the output or efficiency of the process • It is scalable that we are going to use data in kilobytes so that the quite amount of storage is satisfied

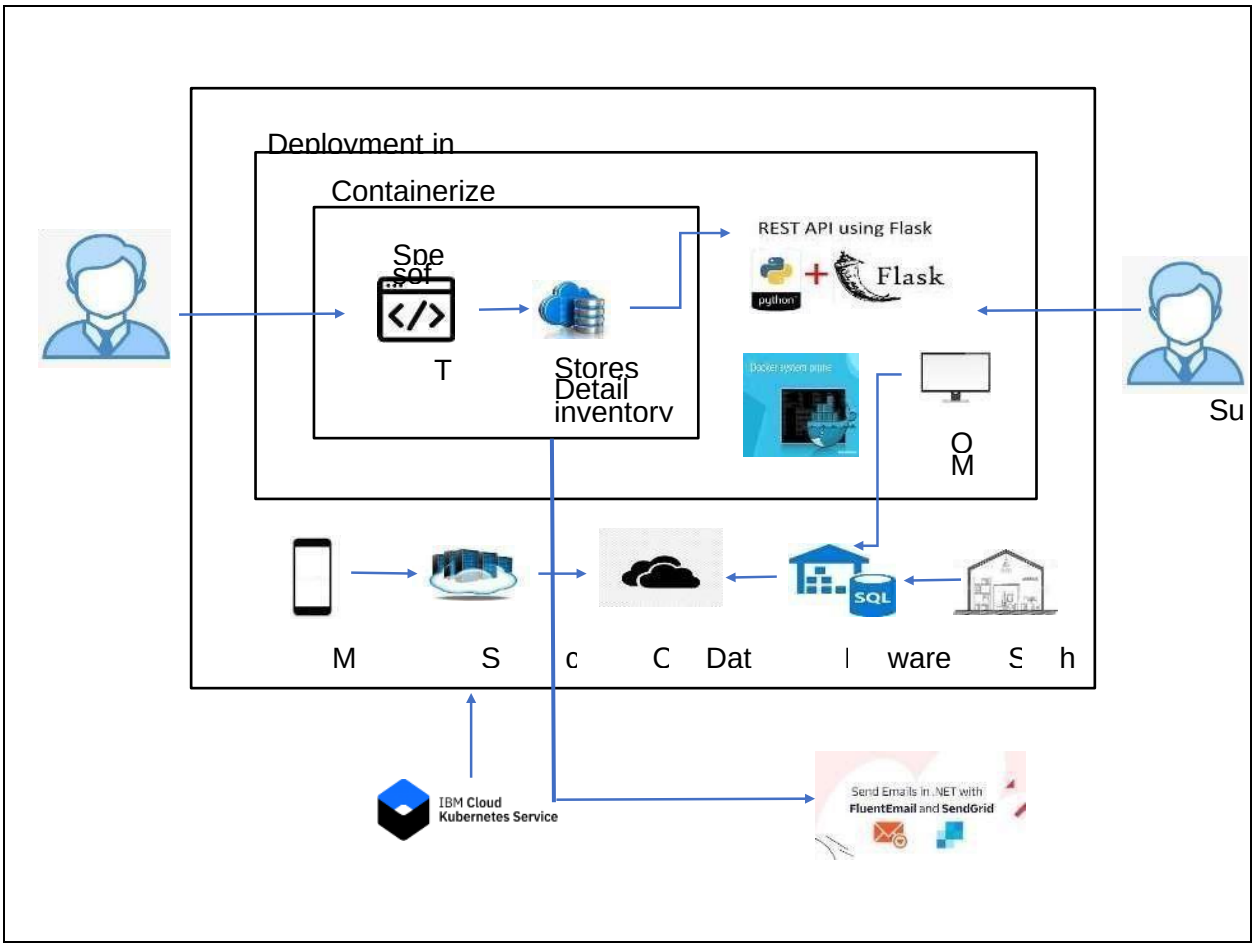
CHAPTER 5
PROJECT DESIGN

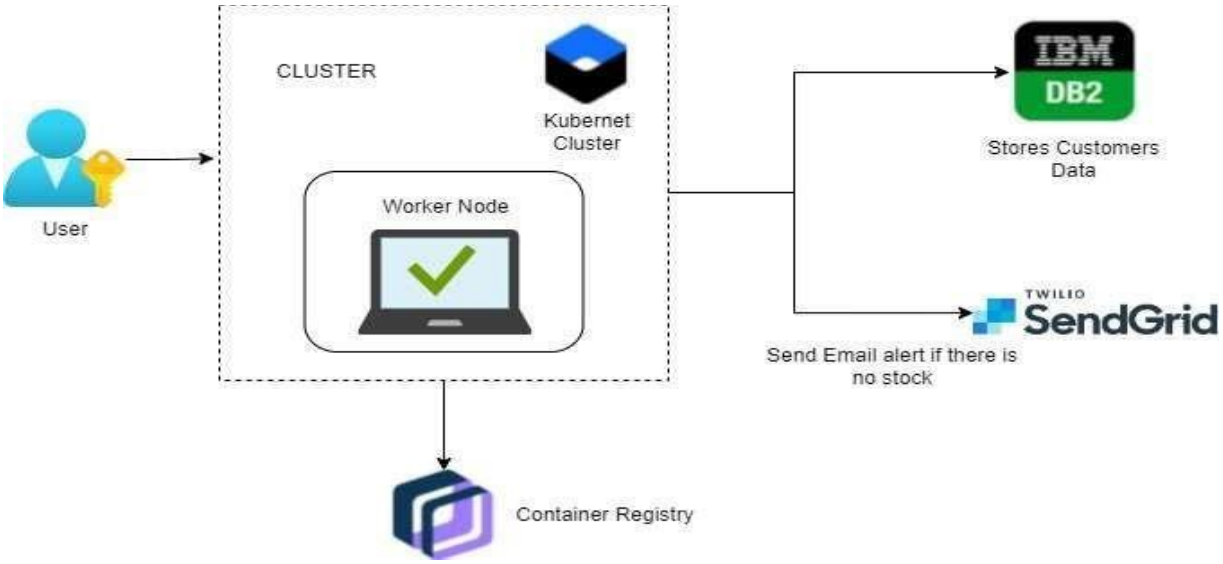
5.1 Data Flow Diagrams

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.

5.2 Solution & Technical Architecture




Deployment in
Containerize
Spe
sof
T
Stores
Detail
inventorv
REST API using Flask
python
Flask
O
M
Su
M
S
c
C
Dat
l
ware
S
h
SQL
IBM Cloud
Kubernetes Service
Send Emails in .NET with
FluentEmail and SendGrid


CLUSTER
Kubernet
Cluster
Worker Node
User
IBM
DB2
Stores Customers
Data
TWILIO
SendGrid
Send Email alert if there is
no stock
Container Registry

Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Through web application, the information processed willbe sent to the user via mail.	HTML, CSS, jQuery, JavaScript, python, etc.
2.	Application Logic-1	User registration through form and confirmation will be sent to the user via email.	Flask, SendGrid
3.	Application Logic-2	Dashboard is used by which the system will Maintain tracking of sales of product and inventory levels.	Flask
4.	Application Logic-3	User will get notified about the stock status.	Flask
5.	Database	The data can be stored in database and user can retrieve or manipulate the data whenever required.	IBM DB2.
6.	Cloud Database	Information of the stocks will be stored and hosted on the cloud.	IBM DB2.
7.	File Storage	Requirements to store files	IBM Block Storage or Other Storage Service or Local File system
8.	External API-1	SendGrid used in application will send the email alert if there is less number or no stock to the user	SendGrid
9.	External API-2	IBM container Registry enables you to store and distribute Docker images in a managed private registry	IBM container registry
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / CloudLocal Server Configuration:localhost:5001(Flask) Cloud Server Configuration : Kubernetes	Local, Cloud Foundry, Kubernetes, etc.

Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	SendGrid will send email alert, if there is less number of stock to user, Kubernetes for manipulating Kubernetes API objects, IBM DB2 is used for storing and retrieving the data efficiently.	Flask, SendGrid, IBMDB2, Kubernetes
2.	Security Implementations	We use login for the user and the information will be hashed so that it will be very secure to use.	IBM container registry.
3.	Scalable Architecture	It is scalable that we are going to use data in kb so that the quite amount of storage is satisfied.	Flask
4.	Availability	Prediction will be available for every user but only for premium user news, database and price alertwill be alert.	Flask.
5.	Performance	It will perform fast and secure	Flask, IBM container registry, IBM DB2.

		even at the lower bandwidth.	

5.3User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer r)	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-3
		USN-4	As a user, I can register for the application through Gmail	I can register for the application through Gmail	Medium	Sprint-2
	Login	USN-5	As a user, I can log into the application by entering email & password	I can log in by entering Gmail & password	High	Sprint-1
	Dashboard	USN-6	As a user, I can track data of sales of products and	I can track data of sales of products and inventory levels.	High	Sprint-1

			inventory levels			
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-3
		USN-10	As a user, I can register for the application through Gmail	I can register for the application through Gmail	Medium	Sprint-2
	Login	USN-11	As a user, I can log into the application by entering email & password	I can log in by entering Gmail & password	High	Sprint-1
	Dashboard	USN-12	As a user, I can track data of sales of products and inventory levels	I can track data of sales of products and inventory levels.	High	Sprint-1
Customer Care Executive	Support	USN-13	As a Executive, I Provide answers for the queries asked by users.	I provide the answers for the queries asked by the users.	High	Sprint-1

CHAPTER 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 using my email & password and confirming my login credentials.	3	High	G.Bavesh Jayasuriya S.Agathiyan M.Dheepthi K.Tamilkumaran
Sprint-1		USN-2	As a user, I can login through my E-mail.	3	Medium	G.Bavesh Jayasuriya S.Agathiyan M.Dheepthi K.Tamilkumaran
Sprint-1	Confirmation	USN-3	As a user, I can receive my confirmation email once I have registered for the application.	2	High	G.Bavesh Jayasuriya S.Agathiyan M.Dheepthi K.Tamilkumaran
Sprint-1	Login	USN-4	As a user, I can log in to the authorized account by entering the registered email and password.	3	Medium	G.Bavesh Jayasuriya S.Agathiyan M.Dheepthi K.Tamilkumaran
Sprint-2	Dashboard	USN-5	As a user, I can view the products that are available currently.	4	High	G.Bavesh Jayasuriya S.Agathiyan M.Dheepthi K.Tamilkumaran

Sprint-2	Stocks update	USN-6	As a user, I can add products which are not available in the inventory and restock the products.	3	Medium	G.Bavesh Jayasuriya S.Agathiyan M.Dheepthi K.Tamilkumaran
Sprint-3	Sales prediction	USN-7	As a user, I can get access to sales prediction tool which can help me to predict better restock management of product.	6	Medium	G.Bavesh Jayasuriya S.Agathiyan M.Dheepthi K.Tamilkumaran
Sprint-4	Request for customer care	USN-8	As a user, I am able to request customer care to get in touch with the administrators and enquire the doubts and problems.	4	Medium	G.Bavesh Jayasuriya S.Agathiyan M.Dheepthi K.Tamilkumaran
Sprint-4	Giving feedback	USN-9	As a user, I am able to send feedback forms reporting any ideas for improving or resolving any issues I am facing to get it resolved.	3	Medium	G.Bavesh Jayasuriya S.Agathiyan M.Dheepthi K.Tamilkumaran

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	11	6 Days	24 Oct 2022	29 Oct 2022	11	29 Oct 2022
Sprint-2	7	6 Days	31 Oct 2022	05 Nov 2022	7	05 Nov 2022
Sprint-3	6	6 Days	07 Nov 2022	12 Nov 2022	6	12 Nov 2022
Sprint-4	7	6 Days	14 Nov 2022	19 Nov 2022	7	19 Nov 2022

CHAPTER 7
CODING & SOLUTIONING

7.1 Front-End:

HYPER TEXT MARKUP LANGUAGE (HTML)

HTML is an application of the Standard Generalized Markup Language (SGML), which was approved as an international standard in the year 1986. SGML provides a way to encode hyper documents so they can be interchanged SGML is also a meta language for formally describing document markup system. In fact HTML uses SGML to define a language that describes a WWW hyper document's structure and inter connectivity. Followingthe rigors of SGML, TBL boreHTML to the world in 1990. Since then, many of us have it to be easy to use but sometimes quite limiting. These limiting factors are being addressed but the World Wide Web Consortium (aka W3c) at MIT. But HTML had to start somewhere, and its success argues that it didn't start out too badly.

CASCADING STYLE SHEET

If your HTML or CSS code is not rendering in the browser as intended, make sure you have written the code exactly as written in the tutorial. Though we encourage you to manually write out the code for the purpose of learning, copy and pasting can be helpful at times to ensure that your code matches the examples.

HTML and CSS errors can be caused by a number of things. Check your markup and CSS rules for extra or missing spaces, missing or misspelled tags, and missing or incorrect punctuation or characters.Curly quotes are designed for human-readable text and will cause an error in your code as they are not recognized as quotation marks by browsers. By typing quotation marks directly into your code editor, you can make sure you are using the right kind.

BOOTSTRAP

The full Bootstrap framework is composed of one CSS stylesheet and one to three JavaScript files, depending on which version of Bootstrap you're using.

To add Bootstrap to an existing project, you can add the required link tag to your `head` element and the script tags right before the closing `</body>` tag. After adding Bootstrap, you should see some of the styles on your web page change.

7.2 Back End

PYTHON

Popularity is a reasonably good reference point when it comes to choosing the best technology for app development and, in particular, for designing the app's back-end. And it rarely comes out of thin air – there is the positive experience of thousands or even millions of people who remained satisfied upon using the language. At the same time, the positivity of the experience is neither groundless nor accidental – it is the result of people's making good use of what Python has to offer.

While on the subject of Python-based technologies, we cannot but mention frameworks. Django and Flask are the most widely known Python web development frameworks that are both, though at different levels, add efficiency to the development process. The latter is a micro-framework that accelerates the development of simple web applications. The former is a full-stack web framework that comes with a bunch of pre-built functionality contributing to the framework's well-deserved popularity. Django improves code clarity and maintainability, boosts performance, and increases the ease of development. The idea behind Django is to help software engineers create code that can be passed from one person to another without facing the risk of misinterpretation, as there will be basically only one suitable way of problem-solving. Python's use in web development, data processing, scripting, and desktop app development is already reason enough to say that Python is in the safe zone thus far.

FLASK

A Web Application Framework or a simply a Web Framework represents a collection of libraries and modules that enable web application developers to write applications without worrying about low-level details such as protocol, thread management, and so on.

Flask is a web application framework written in Python. It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Poocco. Flask is based on the Werkzeg WSGI toolkit and the Jinja2 template engine.Both are Pocco projects.

Unlike the Django framework, Flask is very Pythonic. It's easy to get started with Flask, because it doesn't have a huge learning curve.

On top of that it's very explicit, which increases readability. To create the "Hello World" app, you only need a few lines of code.

This is a boilerplate code example.

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():

 return 'Hello World!'
if __name__ == '__main__':
  app.run()
```

```
    $ python server.py
* Serving Flask app "hello"
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

If you want to develop on your local computer, you can do so easily.Save this program as server.py and run it with python server.py.

JAVASCRIPT

JavaScript is a client-side programming language that allows web developers to write customized client-side scripts to make web pages more dynamic and interactive. At the same time, developers can create server-side code in JavaScript using cross-platform runtime engines like Node. js.

KUBERNETES

with deploying and managing your application easier. Providing automated container orchestration, Kubernetes improves your reliability and reduces the time and resources attributed to daily operations.

Kubernetes has built-in commands to handle a lot of the heavy lifting that goes into application management, allowing you to automate day-to-day operations. You can make sure applications are always running the way you intended them to run.

When you install Kubernetes, it handles the compute, networking, and storage on behalf of your workloads. This allows developers to focus on applications and not worry about the underlying environment.

Kubernetes can automatically adjust the size of a cluster required to run a service. This enables you to automatically scale your applications, up and down, based on the demand and run them efficiently.

DOCKER

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allows you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

Docker provides tooling and a platform to manage the lifecycle of your containers:

- Develop your application and its supporting components using containers.
- The container becomes the unit for distributing and testing your application.
- When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

7.3 Database Schema

IBM CLOUD STORAGE

IBM Cloud is a suite of cloud computing services from IBM that offers both platform as a service (PaaS) and infrastructure as a service (IaaS).

With IBM Cloud IaaS, organizations can deploy and access virtualized IT resources -- such as compute power, storage and networking -- over the internet. For compute, organizations can choose between bare-metal or virtual servers.

With IBM Cloud PaaS -- which is based on the open source cloud platform Cloud Foundry -- developers can use IBM services to create, manage, run and deploy various types of applications for the public cloud, as well as for local or on-premises environments. IBM Cloud supports various programming languages, such as Java, Node.js, PHP and Python and extends to support other languages.

IBM CLOUD FEATURES

There are a number of IBM cloud services that are a part of the IBM cloud. These services are grouped into 16 categories:

- AI/machine learning: A collection of Watson-based AI resources and tools for building your own AI models.
- Automation: Automation resources enable business workflows to be automated using IBM Cloud Pak. Turbonomic is also available as an automation resource and can be used for application resource management and cost optimization.
- Containers: IBM offers its own cloud Kubernetes service, as well as access to the container registry, Red Hat OpenShift and Istio (a server mesh for microservices).
- IBM Cloud Paks: IBM Cloud Paks are applications that are certified for use on Red Hat Open Shift. Cloud Paks exist for business automation, data, integration, network automation, security and Watson.
- Quantum: Provides the ability to run workloads on quantum systems through IBM Quantum composer, the IBM Quantum Lab and the Qiskit SDK.
- Compute: Offers various compute resources, including bare-metal servers, VMs and serverless computing on which enterprises can host their workloads.
- Networking: Provides cloud networking services, such as a load balancer, a content delivery network, VPN tunnels and firewalls.
- Storage: IBM's cloud storage offerings include object, block and file storage for cloud data.
- Logging and monitoring: Provides tools to log, manage and monitor cloud deployments, including Cloud Activity Tracker, Cloud Log Analysis and Cloud Monitoring.

IBM DB2

IBM Db2 is a family of data management products, including the Db2 relational database. The products feature AI-powered capabilities to help you modernize the management of both structured and unstructured data across on-premises and multicloud environments. By helping to make your data simple and accessible, the Db2 family positions your business to pursue the value of AI.

Most of the Db2 family is available on the IBM Cloud Pak® for Data platform, either as an add-on or an included data source service, making virtually all of your data available across hybrid or multicloud environments to fuel your AI applications. Easily converge your transactional data stores and rapidly derive insights through universal, intelligent querying of data across disparate sources. Cut costs with the multimodel capability that eliminates the need for data replication and migration. Enhance agility by running Db2 on any cloud vendor.

CHAPTER 8
TESTING

8.1 Test Cases

Sr.no	Test cases	Action	Steps	Input Data	Expected Result	Actual Result	Status
1	TC-1	User name	Enter user name	Input *xxx@yyy*	It should accepted user name	Valid user name	Pass
2	TC-2	Password	Enter password	Input*……*	Valid password	Valid password	Pass
3	TC-3	Item Name	Select item name	-	Item name should be automatically reflected	Item name is automatically reflected	Pass
4	TC-4	Available item stock	Click on textbox	-	It should reflect automatically item stock	Item stock reflecting automatically	Pass
5	TC-5	Quantity	Enter item quantity	Input *50*	Item quantity should be accepted	Item quantity is accepting	Pass
6	TC-6	Mail transfer	Send a mail for retailers	-	Transfer mail for retailers	Mail transferred	Pass
7	TC-7	Db2	Click datas	-	View datas	View datas	Pass
8	TC-8	Item list	View list items	-	Receive items list	Receive items result	Pass
9	TC-9	Add item	Click on add item	-	It should be add item reflecting in database	Add item in a reflecting a database	Pass

8.2. SYSTEM TESTING

It is the process of exercising software with the intent of finding and ultimately correcting errors. This fundamental philosophy does not change for web applications, because web-based system and applications reside on network and inter-operate with many different operating systems, browsers, hardware platforms and communication protocols. Thus searching for errors is significant challenge for web applications.

Testing issues:

- Client GUI should be considered.
- Target environment and platform considerations
- Distributed database considerations

- Distributed processing consideration

Testing and Methodologies

System testing is the state of implementation, which is aimed at ensuring that the system works accurately and efficiently as expect before live operation, commences. It certifies that the whole set of programs hang together System testing requires a test plan, that consists of several key activities and steps for run program, string, system and user acceptance testing. The implementation of newly design package is important in adopting a successful new system.

Testing is important stage in software development. System test is implementation should be a confirmation that all is correct and an opportunityto show the users that the system works as they expected It accounts the largest percentage of technical effort in software development process.

Testing phase is the development phase that validates the code against the functional specifications. Testing is a vital to the achievement of the system goals. The objective of testing is to discover errors. To fulfill this objective a series of test step such as the unit test, integration test, validation and system test where planned and executed.

Unit testing

Here each program is tested individually so any error apply unit is debugged. The sample data are given for the unit testing. The unit test results are recorded for further references. During unit testing the functions of the program unit validation and the limitations are tested.

Unit testing is testing changes made in a existing or new program this test is carried out during the programming and each module is found to be working satisfactorily. For example, in the registration form after entering all the fields we click the submit button. When submit button is clicked, all the data in form are validated. Only after validation entries will be added to the database.

Unit testing comprises the set of tests performed by an individual prior to integration of the unit into large system. The situation is illustrated in as follows

Coding-> Debugging ->Unit testing -> Integration testing

The four categories of test that a programmer will typically perform on a program unit

a. Functional test
b. Performance test
c. Stress Test
d. Structure test

Functional test involve exercising the code with nominal input values for which the expected results are known as well as boundary values and special values.

Performance testing determines the amount of execution time spent in various parts of unit program through put and response time and device utilization by the program.

A variation of stress testing called sensitivity testing in same situations a very small range of data contained in a bound of valid data may cause extreme and even erroneous processing or profound performance degradation.

Structured testing is concerned with a exercising the internal logic of a program and traversing paths. Functional testing, stress testing performance testing are referred as “black box” testing and structure testing is referred as “white box” testing

Testing results

All the tests should be traceable to customer requirements the focus of testing will shift progressively from programs exhaustive testing is not possible To be more effective testing should be which has probability of finding errors

The following are the attributes of good test

- A good test has a probability of finding a errors
- A good test should be “best of breeds”
- A good test to neither simple nor too complex

CHAPTER 9
RESULTS

9.1 SYSTEM IMPLEMENTATION

System implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system and giving a user confidence in that the new system will work efficiently and effectively in the implementation stage. The stage consists of

- Testing a developed program with sample data
- Detection and correction of error
- Creating whether the system meets a user requirements
- Making necessary changes as desired by users.
- Training user personal

The implementation phase is less creative than system design. A system designmay be dropped at any time prior to implementation, although it becomes more difficult when it goes to the design phase. The final report of the implementation phase includes procedural flowcharts, record layouts, and a workable plan for implementing the candidate system design into a operational design. PYTHON,FLASK,DOCKER and IBM CLOUD STORAGE(IBM DB2) has offer very efficient yet a simple implementation technique for development of the project.

CHAPTER 10
ADVANTAGES & DISADVANTAGES

10.1. ADVANTAGES

It helps to maintain the right amount of stocks.

It leads to a more organized warehouse.

It saves time and money.

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

Improves efficiency and productivity.

Increased information transparency.

10.2. DISADVANTAGES

- Bureaucracy
- Impersonal touch
- Increased space is need to hold the inventory
- Production problem
- High implementation costs
- Complexity

CHAPTER 11
CONCLUSION

The “INVENTORY MANAGEMENT SYSTEM FOR RETAILERS” has been developed to satisfy all proposed requirements. The process is maintained more simple and easy. The system is highly scalable and user friendly. Almost all the system objectives have been met. The system has been tested under all criteria. The system minimizes the problem arising in the existing manual system and it eliminates the human errors to zero level. The design of the database is flexible ensuringthat the system can be implemented. It is implemented and gone through all validation. All phases of developmentwere conceived using methodologies. User with little training can get the required report. The software executes successfully by fulfilling the objectives of the project. Further extensions to this system can be made required with minor modifications.

CHAPTER 12
FUTURE SCOPE

'**Companies can reap a 25% increase in productivity, a 20% gain in space usage, and a 30% improvement in stock use efficiency if they use integrated order processing for their inventory system**. Advanced mobile applications allow companies to manage their inventory and supply chains effectively.

As future work, the project can be enhanced with these algorithm with different dataset with different algorithms.

CHAPTER 13
APPENDIX

Source Code

App.py

```
from flask import Flask, render_template, url_for, request, redirect, session, make_response
import sqlite3 as sql
from functools import wraps
import re
import ibm_db
#import os
#from sendgrid import SendGridAPIClient
#from sendgrid.helpers.mail import Mail
from datetime import datetime, timedelta
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=21fecfd8-47b7-4937-840d-
d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31864;SECURITY=SSL;SSLServerCerti
ficate=DigiCertGlobalRootCA.crt;UID=hlh30208;PWD=7W3q7muS8eHRjGdm", '', '')

app = Flask(__name__)
app.secret_key = 'king'


def rewrite(url):
    view_func, view_args = app.create_url_adapter(request).match(url)
    return app.view_functions[view_func](**view_args)


def login_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        if "id" not in session:
            return redirect(url_for('login'))
        return f(*args, **kwargs)
    return decorated_function


@app.route('/')
def root():
    return render_template('login.html')


@app.route('/user/<id>')
@login_required
def user_info(id):
```

```
    with ibm_db.connect('users.db') as con:
        con.row_factory = sql.Row
        cur = con.cursor()
        cur.execute(f'SELECT * FROM users WHERE email="{id}"')
        user = cur.fetchall()
    return render_template("user_info.html", user=user[0])


@app.route('/login', methods=['GET', 'POST'])
def login():
    global userid
    msg = ''

    if request.method == 'POST':
        un = request.form['username']
        pd = request.form['password_1']
        print(un, pd)
        sql = "SELECT * FROM users WHERE email =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, un)
        ibm_db.bind_param(stmt, 2, pd)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['loggedin'] = True
            session['id'] = account['EMAIL']
            userid = account['EMAIL']
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'

            return rewrite('/dashboard')
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg=msg)


@app.route('/signup', methods=['POST', 'GET'])
def signup():
    mg = ''
    if request.method == "POST":
        username = request.form['username']
        email = request.form['email']
        pw = request.form['password']
        sql = 'SELECT * FROM users WHERE email =?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        acnt = ibm_db.fetch_assoc(stmt)
```

```
        print(acnt)

        if acnt:
            mg = 'Account already exits!!'

        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            mg = 'Please enter the avalid email address'
        elif not re.match(r'[A-Za-z0-9]+', username):
            ms = 'name must contain only character and number'
        else:
            insert_sql = 'INSERT INTO users (USERNAME,FIRSTNAME,LASTNAME,EMAIL,PASSWORD)
VALUES (?,?,?,?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, username)
            ibm_db.bind_param(pstmt, 2, "firstname")
            ibm_db.bind_param(pstmt, 3, "lastname")
            # ibm_db.bind_param(pstmt,4,"123456789")
            ibm_db.bind_param(pstmt, 4, email)
            ibm_db.bind_param(pstmt, 5, pw)
            print(pstmt)
            ibm_db.execute(pstmt)
           # mg = 'You have successfully registered click login!'
            #message = Mail(
             #   from_email=os.environ.get('MAIL_DEFAULT_SENDER'),
              #  to_emails=email,
               # subject='New SignUp',
                #html_content='<p>Hello, Your Registration was successfull. <br><br> Thank you for choosing us.</p>')

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

            #response = sg.send(message)
            #print(response.status_code, response.body)
            return render_template("login.html", meg=mg)

    elif request.method == 'POST':
        msg = "fill out the form first!"
    return render_template("signup.html", meg=mg)


@app.route('/dashboard', methods=['POST', 'GET'])
@login_required
def dashBoard():
    sql = "SELECT * FROM stocks"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_assoc(stmt)
    stocks = []
    headings = [*dictionary]
    while dictionary != False:
```

```
        stocks.append(dictionary)
        # print(f"The ID is : ", dictionary["NAME"])
        # print(f"The name is : ", dictionary["QUANTITY"])
        dictionary = ibm_db.fetch_assoc(stmt)

    return render_template("dashboard.html", headings=headings, data=stocks)


@app.route('/addstocks', methods=['POST'])
@login_required
def addStocks():
    if request.method == "POST":
        print(request.form['item'])
        try:
            item = request.form['item']
            quantity = request.form['quantity']
            price = request.form['price']
            total = int(price) * int(quantity)
            insert_sql = 'INSERT INTO stocks (NAME,QUANTITY,PRICE_PER_QUANTITY,TOTAL_PRICE)
VALUES (?,?,?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.bind_param(pstmt, 2, quantity)
            ibm_db.bind_param(pstmt, 3, price)
            ibm_db.bind_param(pstmt, 4, total)
            ibm_db.execute(pstmt)

        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('dashBoard'))


@app.route('/updatestocks', methods=['POST'])
@login_required
def UpdateStocks():
    if request.method == "POST":
        try:
            item = request.form['item']
            print("hello")
            field = request.form['input-field']
            value = request.form['input-value']
            print(item, field, value)
            insert_sql = 'UPDATE stocks SET ' + field + "= ?" + " WHERE NAME=?"
            print(insert_sql)
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
```

```
            ibm_db.bind_param(pstmt, 2, item)
            ibm_db.execute(pstmt)
            if field == 'PRICE_PER_QUANTITY' or field == 'QUANTITY':
                insert_sql = 'SELECT * FROM stocks WHERE NAME= ?'
                pstmt = ibm_db.prepare(conn, insert_sql)
                ibm_db.bind_param(pstmt, 1, item)
                ibm_db.execute(pstmt)
                dictonary = ibm_db.fetch_assoc(pstmt)
                print(dictonary)
                total = dictonary['QUANTITY'] * dictonary['PRICE_PER_QUANTITY']
                insert_sql = 'UPDATE stocks SET TOTAL_PRICE=? WHERE NAME=?'
                pstmt = ibm_db.prepare(conn, insert_sql)
                ibm_db.bind_param(pstmt, 1, total)
                ibm_db.bind_param(pstmt, 2, item)
                ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('dashBoard'))


@app.route('/deletestocks', methods=['POST'])
@login_required
def deleteStocks():
    if request.method == "POST":
        print(request.form['item'])
        try:
            item = request.form['item']
            insert_sql = 'DELETE FROM stocks WHERE NAME=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('dashBoard'))


@app.route('/update-user', methods=['POST', 'GET'])
@login_required
def updateUser():
    if request.method == "POST":
        try:
            email = session['id']
            field = request.form['input-field']
```

```
            value = request.form['input-value']
            insert_sql = 'UPDATE users SET ' + field + '= ? WHERE EMAIL=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, email)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            # print(msg)
            return redirect(url_for('profile'))


@app.route('/update-password', methods=['POST', 'GET'])
@login_required
def updatePassword():
    if request.method == "POST":
        try:
            email = session['id']
            password = request.form['prev-password']
            curPassword = request.form['cur-password']
            confirmPassword = request.form['confirm-password']
            insert_sql = 'SELECT * FROM  users WHERE EMAIL=? AND PASSWORD=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, email)
            ibm_db.bind_param(pstmt, 2, password)
            ibm_db.execute(pstmt)
            dictionary = ibm_db.fetch_assoc(pstmt)
            print(dictionary)
            if curPassword == confirmPassword:
                insert_sql = 'UPDATE users SET PASSWORD=? WHERE EMAIL=?'
                pstmt = ibm_db.prepare(conn, insert_sql)
                ibm_db.bind_param(pstmt, 1, confirmPassword)
                ibm_db.bind_param(pstmt, 2, email)
                ibm_db.execute(pstmt)
        except Exception as e:
            msg = e
        finally:
            # print(msg)
            return render_template('result.html')


@app.route('/orders', methods=['POST', 'GET'])
@login_required
def orders():
    query = "SELECT * FROM orders"
    stmt = ibm_db.exec_immediate(conn, query)
    dictionary = ibm_db.fetch_assoc(stmt)
```

```
    orders = []
    headings = [*dictionary]
    while dictionary != False:
        orders.append(dictionary)
        dictionary = ibm_db.fetch_assoc(stmt)
    return render_template("orders.html", headings=headings, data=orders)


@app.route('/createOrder', methods=['POST'])
@login_required
def createOrder():
    if request.method == "POST":
        try:
            stock_id = request.form['stock_id']
            query = 'SELECT PRICE_PER_QUANTITY FROM stocks WHERE ID= ?'
            stmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(stmt, 1, stock_id)
            ibm_db.execute(stmt)
            dictionary = ibm_db.fetch_assoc(stmt)
            if dictionary:
                quantity = request.form['quantity']
                date = str(datetime.now().year) + "-" + str(
                    datetime.now().month) + "-" + str(datetime.now().day)
                delivery = datetime.now() + timedelta(days=7)
                delivery_date = str(delivery.year) + "-" + str(
                    delivery.month) + "-" + str(delivery.day)
                price = float(quantity) * \
                    float(dictionary['PRICE_PER_QUANTITY'])
                query = 'INSERT INTO orders (STOCKS_ID,QUANTITY,DATE,DELIVERY_DATE,PRICE) VALUES
(?,?,?,?,?)'
                pstmt = ibm_db.prepare(conn, query)
                ibm_db.bind_param(pstmt, 1, stock_id)
                ibm_db.bind_param(pstmt, 2, quantity)
                ibm_db.bind_param(pstmt, 3, date)
                ibm_db.bind_param(pstmt, 4, delivery_date)
                ibm_db.bind_param(pstmt, 5, price)
                ibm_db.execute(pstmt)
        except Exception as e:
            print(e)

        finally:
            return redirect(url_for('orders'))


@app.route('/updateOrder', methods=['POST'])
@login_required
def updateOrder():
    if request.method == "POST":
        try:
```

```
            item = request.form['item']
            field = request.form['input-field']
            value = request.form['input-value']
            query = 'UPDATE orders SET ' + field + "= ?" + " WHERE ID=?"
            pstmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            print(e)

        finally:
            return redirect(url_for('orders'))


@app.route('/cancelOrder', methods=['POST'])
@login_required
def cancelOrder():
    if request.method == "POST":
        try:
            order_id = request.form['order_id']
            query = 'DELETE FROM orders WHERE ID=?'
            pstmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(pstmt, 1, order_id)
            ibm_db.execute(pstmt)
        except Exception as e:
            print(e)

        finally:
            return redirect(url_for('orders'))


@app.route('/suppliers', methods=['POST', 'GET'])
@login_required
def suppliers():
    sql = "SELECT * FROM suppliers"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_assoc(stmt)
    suppliers = []
    orders_assigned = []
    headings = [*dictionary]
    while dictionary != False:
        suppliers.append(dictionary)
        orders_assigned.append(dictionary['ORDER_ID'])
        dictionary = ibm_db.fetch_assoc(stmt)

# get order ids from orders table and identify unassigned order ids
    sql = "SELECT ID FROM orders"
    stmt = ibm_db.exec_immediate(conn, sql)
```

```
    dictionary = ibm_db.fetch_assoc(stmt)
    order_ids = []
    while dictionary != False:
        order_ids.append(dictionary['ID'])
        dictionary = ibm_db.fetch_assoc(stmt)

    unassigned_order_ids = set(order_ids) - set(orders_assigned)
    return render_template("suppliers.html", headings=headings, data=suppliers, order_ids=unassigned_order_ids)


@app.route('/updatesupplier', methods=['POST'])
@login_required
def UpdateSupplier():
    if request.method == "POST":
        try:
            item = request.form['name']
            field = request.form['input-field']
            value = request.form['input-value']
            print(item, field, value)
            insert_sql = 'UPDATE suppliers SET ' + field + "= ?" + " WHERE NAME=?"
            print(insert_sql)
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, value)
            ibm_db.bind_param(pstmt, 2, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            return redirect(url_for('suppliers'))


@app.route('/addsupplier', methods=['POST'])
@login_required
def addSupplier():
    if request.method == "POST":
        try:
            name = request.form['name']
            order_id = request.form.get('order-id-select')
            print(order_id)
            print("Hello world")
            location = request.form['location']
            insert_sql = 'INSERT INTO suppliers (NAME,ORDER_ID,LOCATION) VALUES (?,?,?)'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, name)
            ibm_db.bind_param(pstmt, 2, order_id)
            ibm_db.bind_param(pstmt, 3, location)
            ibm_db.execute(pstmt)
```

```
        except Exception as e:
            msg = e

        finally:
            return redirect(url_for('suppliers'))


@app.route('/deletesupplier', methods=['POST'])
@login_required
def deleteSupplier():
    if request.method == "POST":
        try:
            item = request.form['name']
            insert_sql = 'DELETE FROM suppliers WHERE NAME=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, item)
            ibm_db.execute(pstmt)
        except Exception as e:
            msg = e

        finally:
            return redirect(url_for('suppliers'))


@app.route('/profile', methods=['POST', 'GET'])
@login_required
def profile():
    if request.method == "GET":
        try:
            email = session['id']
            insert_sql = 'SELECT * FROM users WHERE EMAIL=?'
            pstmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(pstmt, 1, email)
            ibm_db.execute(pstmt)
            dictionary = ibm_db.fetch_assoc(pstmt)
            print(dictionary)
        except Exception as e:
            msg = e
        finally:
            # print(msg)
            return render_template("profile.html", data=dictionary)


@app.route('/logout', methods=['GET'])
@login_required
def logout():
    print(request)
    resp = make_response(render_template("login.html"))
    session.clear()
```

```
    return resp


if __name__ == '__main__':
    app.run(debug=True)

# ALTER TABLE stocks ALTER COLUMN ID SET GENERATED BY DEFAULT AS IDENTITY
```

Login.html:

```
{% extends 'base.html '%}
{% block head %}
<title>Login page</title>
{% endblock%}

{%block body%}

<main class="container ">
    <div class="mx-auto mt-5 border bg-light login-card " style="width:500px;">
        <h2 class='mx-4 mt-2'>LOGIN</h2>
        <form action="{{url_for('login') }}" method="post">
            <div class="mx-4 mt-2 text-danger">{{ msg }}</div>
            <div class="my-2 mx-4">
                <label for="username">username</label>
                <input type="text" class="form-control" placeholder="adc@gmail.com" name="username" required />
            </div>
            <div class="my-2 mx-4">
                <label for="password_1">password</label>
                <input type="password" class="form-control" name="password_1" required />
            </div>
            <input type="submit" value="submit" class="btn btn-primary my-4 mt-2 mx-4" />

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

</main>
{% endblock%}
```

Dashboard.html:

```
{% extends 'base2.html '%} {% block head %}
<title>Dashboard</title>
{% endblock%} {%block body%}
<h2>Dashboard</h2>
<p>
  Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
```

```
</p>
{% include 'table.html' %}
<div class="forms-wrapper">
  <form action="{{url_for('UpdateStocks') }}" method="post">
    <h3>Update Stock</h3>
    <div class="field">
      <label class="custom-label" for="item"> Enter Item</label>
      <input class="text-inputs" type="text" name="item" placeholder="milk" />
    </div>
    <div class="field">
      <label for="input-field">Choose a field :</label>
      <select name="input-field" id="field">
        <option value="NAME">NAME</option>
        <option value="PRICE_PER_QUANTITY">PRICE_PER_QUANTITY</option>
        <option value="QUANTITY">QUANTITY</option>
      </select>
    </div>
    <div class="field">
      <label class="custom-label" for="input-value"> Enter Value</label>
      <input
        class="text-inputs"
        type="text"
        name="input-value"
        placeholder=" "
      />
    </div>
    <button class="submit-button">Update</button>
  </form>

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

```
  </form>
  <form action="{{url_for('deleteStocks') }}" method="post">
    <h3>Remove stocks</h3>
    <div class="field">
      <label class="custom-label" for="item"> Enter the item</label>
      <input class="text-inputs" name="item" type="text" placeholder="juice" />
    </div>
    <button class="submit-button red-button">Remove</button>
  </form>
</div>

{% endblock%}
```

Suppliers.html:

```
{% extends 'base2.html '%} {% block head %}
<title>Suppliers</title>
{% endblock%} {%block body%}
<h2>Suppliers</h2>
<p>
  Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
</p>
{% include 'table.html' %}
<div class="forms-wrapper">
  <form action="{{url_for('UpdateSupplier') }}" method="post">
    <h3>Update Supplier</h3>
    <div class="field">
      <label class="custom-label" for="name"> Enter Name</label>
      <input class="text-inputs" type="text" name="name" placeholder="Supplier name" />
    </div>
    <div class="field">
      <label for="input-field">Choose a field :</label>
      <select name="input-field" id="field">
        <option value="NAME">NAME</option>
        <option value="LOCATION">LOCATION</option>
      </select>
    </div>
    <div class="field">
      <label class="custom-label" for="input-value"> Enter Value</label>
      <input
        class="text-inputs"
        type="text"
        name="input-value"
        placeholder=" "
      />
    </div>
    <button class="submit-button">Update</button>
  </form>
```

```
 <form action="{{url_for('addSupplier') }}" method="post">
  <h3>Add New Supplier</h3>
  <div class="field">
   <label class="custom-label" for="name"> Enter the Supplier</label>
   <input class="text-inputs" name="name" type="text" placeholder="Supplier name" />
  </div>
  <div class="field">
   <label class="custom-label" for="quantity"> Enter Order ID : </label>
   <select name="order-id-select" id="field">
    {% for order_id in order_ids %}
     <option value="{{ order_id }}">{{order_id}}</option>
    {% endfor %}
   </select>
  </div>
  <div class="field">
   <label class="custom-label" for="location"> Enter Location</label>
   <input class="text-inputs" type="text" name="location" placeholder="Location" />
  </div>
  <button class="submit-button">Add Stock</button>
 </form>
 <form action="{{url_for('deleteSupplier') }}" method="post">
  <h3>Delete Supplier</h3>
  <div class="field">
   <label class="custom-label" for="name"> Enter the name</label>
   <input class="text-inputs" name="name" type="text" placeholder="Supplier Name" />
  </div>
  <button class="submit-button red-button">Delete</button>
 </form>
</div>

{% endblock%}
```

GitHub & Project Demo Link
Github Link:
https://github.com/IBM-EPBL/IBM-Project-44871-1660727146

Demo Video Link:
https://drive.google.com/file/d/1tWsQBov2Xtlhh3M_PAIzcro8eS_Fu9Ii/view?usp=share_link