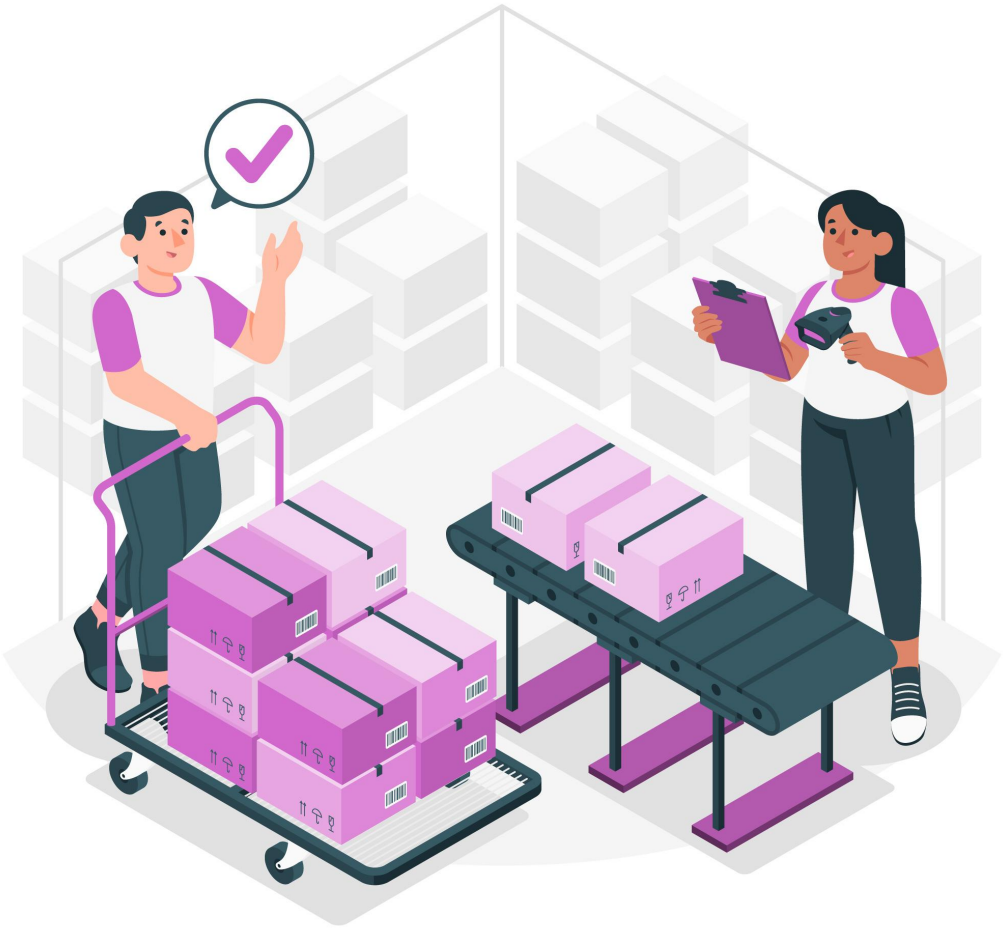
Inventory Management System (Python GUI)



Inventory Management System (Python GUI)

***MAJOR** Project Report submitted in partial fulfillment of the requirements for the award of the degree of*

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

With Specialization in Data Analytics

BY

Subhankar Mandal — UID: TNU2019021100001

Asish Kumar Singh — UID: TNU2019021100003

Kuntal Chaudhury — UID:TNU2019021100002

Nikita Jha

UID: TNU2019021200006

Under the supervision of

Dr. Sandipan Chakravorty

Department of Computer Science & Engineering

The Neotia University, West Bengal, India

AmbujaNeotia

THE NEOTIA UNIVERSITY

ज्ञानम् आत्म प्रदीपाय

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

THE NEOTIA UNIVERSITY

D.H. ROAD, SOUTH 24-PARGANA, WEST BENGAL, INDIA, PIN-743368

MAY 2023

Certificate

We hereby recommend that the Project entitled ***"Inventory Management System(Python GUI)"*** worked under our guidance may please be accepted in the partial fulfillment of the requirement for the degree of "Bachelor in Technology" in the Computer Science and Engineering with specialization in Data Analytics of 'The Neotia University'. The project report in our opinion is worthy for its acceptance. During the work '**Subhankar Mandal, Asish Kumar Singh, Kuntal Chaudhury, Nikita Jha**' was found to be sincere, regular and hard working and has successfully completed the thesis work assigned to her.

……..
Inventory Management
System(Python GUI)
Sandipan Chakrovarty
The Neotia University

…….

(HOD, CSE)
The Neotia University

Declaration of Originality and Compliance of Academic Ethics

I/We hereby declare that this report contains literature survey and original research work done by the undersigned candidate/s, as part of my/our **"Bachelor in Technology Studies**.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I/we also declare that, as required by these rules and conduct, I/We have cited and referenced all materials that are not original to this work.

Inventory Management System (Python GUI)

Student 1

Name: Subhankar Mandal

UID: TNU2019021100001

Signature with Date:

Student 2

Name: Ashish Kumar Singh

UID: TNU2019021100003

Signature with Date:

Student 3

Name: Kuntal Chaudhury

UID: TNU2019021100002

Signature with Date:

Student 4

Name: Nikita Jha

UID: TNU2019021200006

Signature with Date:

ACKNOWLEDGEMENT

We gratefully acknowledge for the assistance, cooperation, guidance and clarifications provided by THE NEOTIA UNIVERSITY during the development of the Inventory Management System website. Our extreme gratitude to Mr.Sandipan Chakravorty who guided us throughout the project. Without his willing disposition, spirit of accommodation, frankness, timely clarification and above all faith in us, this project could not have been completed in due time.
His readiness to discuss all important matters at work deserves special attention. We would also like to thank the whole of the faculty of the college for the project.

Student 1
Name: Subhankar Mandal
UID: TNU2019021100001
CSE-Data Analytics

Student 2
Name: Asish Kumar Singh
UID: TNU2019021100003
CSE-Data Analytics

Student 3
Name: Kuntal Chaudhury
UID: TNU2019021100002
CSE-Data Analytics

Student 4
Name: Nikita Jha
UID: TNU2019021200006
CSE-Data Analytics

Place: The Neotia University

Date:

Contents

Introduction

1.1 Introduction to IMS

1.2 Purpose of the IMS

1.3 Objective of the Project

1.4 Scope of the Application

1.5 Significance of the Project

1.6 Problem Statement

1.7 Features of the project

1.8 0verview Of The Project

2 System Analysis

2.1 The Proposed System

2.2 The Theoretical Background

2.3 Description of System

2.4 Limitations of Existing Project

2.5 Advantages of Proposed System

2.6 Feasibility Study

2.7 Background Knowledge

2.8 Software and hardware requirements

2.9 Justification of technology

3. **System design**

3.1 Process flow diagram

3.2 Use case diagram

3.3 ER diagram

3.4 DFD Diagram ·

4. Other requirements

9. ER DIAGRAM
10. Problem Description & Scope
11. Hardware & Requirement
12. Analysis & Report of the Present Manual System
13. Coding
14. Testing & Implication

Introduction

In today's generation, we are engaged in highly computerized technology aiming to make an individual lifestyle more comfortable and easier, especially in the world of business. The manual system is now considered the first process after the birth of the computerized system. Online transactions are now very common to widen the target market of the company. It becomes more efficient to the customer considering it can save time and is considered hassle-free. The most commonly used system by several companies is the sales system and inventory system creating a web-based system. Advanced system on sale provides more reliable recording of sales of the company with comparison to its actual cost. In addition to the data, information needed by the company to decide matters in relation to inventory can be easily generated. Moreover, the inventory control which ensures stocking the in-demand and correct items in the correct quantities.

This sales and inventory system can help the company to avoid overstocking. When the company overstock, the fund will not back easy because the cycle of the stock is a stop for unneeded items requiring time, space, and funds which could have been used on more critical assets.

The computer is a useful technology in our society that makes our work easy to wrap-up. Nowadays, computer technology continues to evolve and grow fast. Then every business in our society started shifting from using the manual system to an automated one that makes our work easier and faster.

Point of Sales and Inventory System is a computer is an easy way of checking and listing the sales of the company, it is faster and more reliable rather than doing manually. The system can minimize human errors in editing and be easily accessed anytime by the company. Sales and inventory systems make the company more effective, more productive and are convenient for the company and its customers. The system is made to help the establishment show more relevant items to the customers.

1.2 Purpose

The purpose of this document is to present a detailed description of the Online Inventory Management System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

In this document we will try introducing our stakeholders along with their respective viewpoints, describe the existing problem, combining those various viewpoints, balancing those to reach an ultimate “theoretical” solution of the identified problems, generating graphical reviews through unified modeling language (UML) to formulate the problems and the proposed solution, the project scope and the project schedule. Next we present the solution including system analysis, the deviation between final and initial design, the function of our Inventory management system and testing plan. Finally we evaluate our work on different aspects, present areas of improvement and conclusion.

1.3 Objective Of The Project:

General Objective

The main objective of this research is to improve the manual system of TH Garments in checking and adding the stock from the supplier. The new system can fix the problem in managing records for easier monitoring and tracking and help the workers to have a faster way of recording details of each transaction.

Specific Objectives

- To develop a module that generates reports monthly sales and inventory
- To develop security in terms of keeping the records of the inventory
- To develop a system that can monitor the stocks in and stock out and the inventory in a fast and efficient manner
- To accurately record, compute and produce a report of sales

1.4 Scope

The outcome of the project would be automated inventory management service The software will have all common features and functionalities along with some other special facilities. To provide a user efficient working environment.

- User friendly interface for the target stakeholders.
- Proper monitoring facility for the authority.
- Easy maintenance and administration
- Ensure a high level of security.

This system will help in tracking records so that past records can be verified through them and one can make decisions based on the past records. This system will complete the work in a very less time resulting in less time consumption and high level of efficiency.

1.5 Significance of the study

This study is beneficial and helpful for the following:

Customer - They will be given more quality service which will be more convenient for them. The new system will lessen the time spent by customers in the buying and paying process.

Manager - If the manager wants to check the status of their sale, he just needs to login into the system than to check it manually.

Owner - The owner will also benefit from the system because he or she can manage the system which provides everything for the user, he can also benefit from the accurate records the system could produce.

Staff - The staff will have an easy way of recording the orders. They will not have problems in product orders since it's now computerized. They will not be having a hard time searching and updating the stocks.

Future Researcher - The future researchers can use this project as a sample study guide if ever they conduct some study to develop a system related to this kind of system. This project can serve as their reference; they can use the data in this project for further studies.

1.5 Problem Statement

Using manual inventory tracking procedures across different software and spreadsheets is time-consuming, redundant and vulnerable to errors. Even small businesses can benefit from a centralized inventory tracking system that includes accounting features.

Nowadays, in an era that has advanced technology and a place in the world. Everything can be linked only at your fingertips in the times of rapidly developing with the sophisticated technology of today. Therefore, an inventory system is also not lagging behind in introducing a method of keeping an inventory data systematically and safely. The system plays a very important role in improving the competitiveness of a business. Usually, organizations today face too many challenges to achieve the cost, speed and reliability. Efficient inventory systems really help in order to make sure the store's performance and data record is always in good condition and secured from abusers.

The problem faced by the company is they do not have any systematic system to record and keep their inventory data. It is difficult for the admin to record the inventory data quickly and safely because they only keep it in the logbook and not properly organized.

The company now uses a manual system The company problem is they use a chaos system and it is difficult for the admin to estimate their profit. With the new system developed, the company can manage their inventory data easily,quickly.and more securely.Time Consuming:To record the inventory data will cost a time. Admin of the company only has one person so he needs to record every stock detail clearly or else it may lead to lack Information about the data.The data is very important.

The company does not have any secure system for their inventory system. The data can easily be lost because they only use a logbook to record their inventory data. With the system, it will help more on the security of the data. Inventory loss is hard to detect because the admin needs to review one by one page in the logbook, but with the system developed, it may help the admin to detect the inventory in and out from the system.

1.5 Features

1.Centralized Tracking:

Consider upgrading to tracking software that provides automated features for re-ordering and procurement. Inventory management platforms provide centralized, cloud-based databases for accurate, automatic inventory updates and real-time data backup.

2.Transparent Performance:

Measure and report warehouse performance metrics like inventory turnover, customer satisfaction and order processing speed to overcome warehouse inefficiencies. Share this data with employees and suppliers.

3.Stock Auditing:

Frequent stock auditing processes, like daily cycle counting, reduce human error and provide more accurate, up-to-date inventory data for managing cash flow. Organize audits by category and cycle count smaller inventory samples on a predictable schedule for more accurate financial data.

4.Demand Forecasting:

Some inventory management platforms include demand forecasting tools. This feature integrates with accounting and sales data to help you predict demand and schedule orders based on shifting customer preferences, material availability or seasonal trends.

5.Add Imagery:

Add images with product descriptions in your inventory database to improve purchasing and receiving processes, enhance accuracy and prevent misplaced inventory.

6.Go Paperless:

Give employees the right inventory tools for the job. They need software to replace manual inventory documentation, and paperless transactions for invoices and purchase orders.

7.Preventive Control:

Implement stock control systems to manage problem inventory, such as perishable stock, fragile equipment or obsolete materials. Perform regular preventive maintenance on machinery and equipment stock in storage if required by the manufacturer. Catalog data on problem stock location, cost and quantity to monitor shelf life and prevent waste.

8.Measure Service Levels:

Monitor and track supplier data, such as shipment errors, damaged or defective products and missed delivery appointments. Measure your supplier’s performance to find and fix supply chain disruptions, reduce complexity and streamline logistics.

9.Optimize Space:

Use inventory management systems with warehouse management features to optimize storage space and inventory flow. Categorize inventory storage down to shelf, bin and compartment, and automate order picking, packing and shipping workflows.

10.Automate Reorders:

Backordered inventory delays production and creates poor customer experiences. Use inventory management software to set automatic reorder points based on preset stock levels and current availability to avoid overselling.

11.Safety Stock:

Maintain safety stock to offset supply chain disruptions and help manage increased lead times due to shifting international competition for raw materials. Proper inventory planning helps operations adapt to dynamic global supply chains.

12.Classify Inventory:

Create inventory classifications to manage changing trends, such as packaging initiatives to reduce plastic waste. Categorize stock by packaging type, dimensions and product. Use this information to control shipping costs and storage location better.

13.Multi-Location Warehousing:

Use multi-location warehouse management features to track and control expanding inventories. Take advantage of receiving and put-away schedules with automated inventory tracking alerts and scheduling features that keep tabs on warehouse location and in-transit inventory.

System Analysis

2.1 The Proposed System

Background of the Study manual method in making their inventory is a very old method. The researchers conducted an interview with THE Garments Center and found out that the establishment is still using the manual system, the problem is in keeping records of sales and inventories; the establishment has encountered several problems regarding the monitoring and the stocks checking. In the case of TH Garments center, they are currently using the manual sales and inventory system wherein they list the items and compute sales manually. The proposed system of computerized sales and inventory with the TH Garments center, the tracking of sales, controlling of inventory, recording of products, calculating of numerical data and searching of the item will be translated into a computerized process considering speed, accuracy, and maintainability of the system.

Related System:

Sales and Inventory System

According to Shah Janat (2010), the individual in a supply chain, even wholesaler, retailer or manufacturer or vendor, prefers to reduce inventories and then maintain customer services so as not to lose customers because of the non-availability of goods. Huge inventories are a drain on resources from the supplier, as it increases the profits cost of operations. So, it is no surprise that all firms want to reduce inventory in the supply chain. The inventory may be divided into various categories. In general, there are the six selected categories of inventories: First, the return of the stock, the inventory resulting from production or purchase in storages is called return of stocks, since the lots are produced or purchased in cyclical lots.

Second is the safety stocks, as the name selected, are maintained as security against uncertainties of demand and supply. The third is the decoupling of stocks, which gives the efficiency needed by each decision-making unit to manage its operations independently and to optimize its capacity to anticipate inventory which includes stocks accumulated in advance of the expected peak in sales or that which manage some Important event that does not occur on a regular basis.

This has two categories, the time where stocks are the most sold and the speculation stocks. The fifth is the pipeline inventory which consists of materials actually being updated on or being changed from one place to another. Lastly, the dead stock which refers to the part of the non-selected inventory and is not the same to any further use in supply chain operations or markets.

Sales Monitoring System

According to Celkon Mobiles, (2010), Celkon is one of the leading manufacturing companies in India. We have available mobile phone solutions and that is easy to use because it is wireless technologies in India. Celkon caters to the increasing smart needs because there are mobile users in the world. Our work is providing innovative mobile technology to every customer. At Celkon, we believe that every user needs and knowledge more experience than ever. We are dedicated to manufacturing customized user affordable phones. Our company added features to make sure the personality of the phone matches the taste of the user. The long-range capacity of products available at Celkon ensures that there is a phone for every pocket. Every Celkon product undergoes stringent quality tests at every stage of production.

According to Navarro (2012), Sales and Inventory System, a computer has his is general purpose using devicewear can be programmed or be-programmed to carry out the finite set arithmetic or local operation, the computer has a big role in our nation today because of our technology. Wherever you go, computers still exist, especially in business. It makes the procedure easy and efficient by programming the manual system into a computerized system. The purpose of technology in our daily life today has a big reason.

Technology is the marking, modification, usage, and knowledge of a new machine or a tool to craft a system method of organization in order to solve a problem, achieve a goal or perform its own function. It also refers to the stocks of such tools, machinery, modification arrangement, and procedure. Technologies significantly affect human as well other animal species ability to control and adapt to their natural environments.

2.2 The Theoretical Background

Inventory is the goods or materials a business intends to sell to customers for profit. Inventory management, a critical element of the supply chain, is the tracking of inventory from manufacturers to warehouses and from these facilities to a point of sale. The goal of inventory management is to have the right products in the right place at the right time. This requires inventory visibility — knowing when to order, how much to order and where to store stock. The basic steps of inventory management include:

1. **Purchasing inventory:** Ready-to-sell goods are purchased and delivered to the warehouse or directly to the point of sale.

2. **Storing inventory:** Inventory is stored until needed. Goods or materials are transferred across your fulfillment network until ready for shipment.

3. **Profiting from inventory:** The amount of product for sale is controlled. Finished goods are pulled to fulfill orders. Products are shipped to customers.

2.3 Description Of The System:

Periodic inventory management

The periodic inventory system is a method of inventory valuation for financial reporting purposes in which a physical count of the inventory is performed at specific intervals. This accounting method takes inventory at the beginning of a period, adds new inventory purchases during the period and deducts ending inventory to derive the cost of goods sold (COGS).

Barcode inventory management

Businesses use barcode inventory management systems to assign a number to each product they sell. They can associate several data points to the number, including the supplier, product dimensions, weight, and even variable data, such as how many are in stock.

RFID inventory management

RFID or radio frequency identification is a system that wirelessly transmits the identity of a product in the form of a unique serial number to track items and provide detailed product information. The warehouse management system based on RFID can improve efficiency, increase inventory visibility and ensure the rapid self-recording of receiving and delivery.

2.4 Limitation Of The Existing Project:

1. **Inconsistent Tracking:**

   Using manual inventory tracking procedures across different software and spreadsheets is time-consuming, redundant and vulnerable to errors. Even small

businesses can benefit from a centralized inventory tracking system that includes accounting features.

2. Warehouse Efficiency:

Inventory management controls at the warehouse is labor-intensive and involves several steps, including receiving and putaway, picking, packing and shipping. The challenge is to perform all these tasks in the most efficient way possible.

3. Inaccurate Data:

You need to know, at any given moment, exactly what inventory you have. Gone are the days when inventory could be counted once a year with an all-hands-on-deck approach.

4. Changing Demand:

Customer demand is constantly shifting. Keeping too much could result in obsolete inventory you're unable to sell, while keeping too little could leave you unable to fulfill customer orders. Order strategies for core items, as well as technology to create and execute an inventory plan, can help compensate for changing demand.

5. Limited Visibility:

When your inventory is hard to identify or locate in the warehouse, it leads to incomplete, inaccurate or delayed shipments. Receiving and finding the right stock is vital to efficient warehouse operations and positive customer experiences.

2.5 Advantages Of The Project:

1. **Centralized Tracking:**

Consider upgrading to tracking software that provides automated features for re-ordering and procurement. Inventory management platforms provide centralized, cloud-based databases for accurate, automatic inventory updates and real-time data backup.

2. **Transparent Performance:**

   Measure and report warehouse performance metrics like inventory turnover, customer satisfaction and order processing speed to overcome warehouse inefficiencies. Share this data with employees and suppliers.

3. **Stock Auditing:**

   Frequent stock auditing processes, like daily cycle counting, reduce human error and provide more accurate, up-to-date inventory data for managing cash flow. Organize audits by category and cycle count smaller inventory samples on a predictable schedule for more accurate financial data.

4. **Demand Forecasting:**

   Some inventory management platforms include demand forecasting tools. This feature integrates with accounting and sales data to help you predict demand and schedule orders based on shifting customer preferences, material availability or seasonal trends.

5. **Add Imagery:**

   Add images with product descriptions in your inventory database to improve purchasing and receiving processes, enhance accuracy and prevent misplaced inventory.

6. **Go Paperless:**

   Give employees the right inventory tools for the job. They need software to replace manual inventory documentation, and paperless transactions for invoices and purchase orders.

7. **Preventive Control:**

   Implement stock control systems to manage problem inventory, such as perishable stock, fragile equipment or obsolete materials. Perform regular preventive

maintenance on machinery and equipment stock in storage if required by the manufacturer. Catalog data on problem stock location, cost and quantity to monitor shelf life and prevent waste.

2.6 Feasibility Study:

Inventory Management System must be designed to meet the dictates of the marketplace and support the company's strategic plan. Along with the inventory system Garment industries need some more special attributes. The Software's for Garment are specially made for managing the various steps in order processing of garments manufacturing process. This software is modular in design and is web enabled for remote access as well as intranet usage without the need to install in every machine. Inventory Management System has some features which are badly needed for a complete Inventory system of Garment industries. So the features are

- Web based interface
- User based Login - password based authentication for data protection.
- Dynamic Modular structure – Each and every section related to a production has separate menus.
- Manage master details of buyer, supplier and vendor.
- Job Work tracking.
- Raw material tracking.
- Shipment tracking.
- Product category and product stock information with value.
- Reports include purchase, issues, stocks & categories.
- Customized package
- Notification System in case of any process delay.
- Check sheet automation.

These features are the primary requirement of our client for this inventory management system. This feature list will change in terms of client demand.

2.7 Background Knowledge:

- Inventory management is the entire process of managing inventories from raw materials to finished products.
- Inventory management tries to efficiently streamline inventories to avoid both gluts and shortages.
- Four major inventory management methods include just-in-time management (JIT), materials requirement planning (MRP), economic order quantity (EOQ) , and days sales of inventory (DSI).

2.8 Software And Hardware Requirements:

Hardware Interfaces

In the current version of the software, it will have no special hardware interface with other external systems. It will run in a general-purpose computer system with general-purpose hardware and software.

Software Interfaces

The Current version of this system will be built on the following software:

Server:

- Internet Information system.
- SQLite3.

Client:

- Python.
- OpenSSL.

2.9 Justification of the Technology:

Inventory management can be a huge pain point for companies. It's not uncommon for business owners to lose track of what they have on hand and order more than they need. Technology can help with that problem by keeping track of your inventory to help you make better decisions about purchasing additional products for your business.
Below are some ways that technology has helped enhance the inventory management process for businesses:

Inventory Management Is Faster Because of Technology

Before the rise of innovations in technology, inventory management processes were slow. It was more challenging to keep track of inventory and communicate with suppliers because the technology wasn't as advanced as it is now. Presently, business owners have tools like inventory management software and tracking software to keep a better eye on their business.

Inventory tracking software can help business owners manage their inventory faster than ever before by making requests, checking on orders, and getting status updates from manufacturers all at once. This process is more convenient compared to having to communicate with manufacturers via phone or email.

Because technology has made inventory management more efficient, businesses can manage their inventory faster, which ultimately helps companies improve productivity and reduce costs at the same time. Inventory tracking technology facilitates communication between suppliers and customers by giving access through a computer or mobile device, allowing business leaders to respond to issues in real-time.

Inventory Management Is More Accurate Because of Technology

Nowadays, inventory management systems are less prone to errors because of technology. Inventory technology helps businesses track where their inventory is and allows them to keep an eye on the number of items left in stock, which makes it easier for companies to monitor how many products they have remaining and order more before running out completely.

Fewer errors mean more accurate inventory management processes for businesses. This translates to improved productivity and cost-effectiveness for businesses, as they will avoid losing products or running out of stock altogether.

Technology Empowers Business Leaders with Data-driven Insight

Technology also allows companies to review historical data to help them better understand how their inventory management system works with the current systems in place. This will enable businesses to set up an improved plan moving forward. As a result, technology helps enterprises manage their inventory more effectively.

Technology Helps Entrepreneurs Better Deal with Inventory Problems

Inventory management technology allows businesses to respond more effectively when issues arise with suppliers or customers. For example, technology can help companies to expedite deliveries when a customer needs an item quickly. It also allows them to track inventory that customers or suppliers have returned.

Software Development Methodology

The establishment and use of sound engineering principles in order to obtain economically developed software that is reliable and works efficiently on real machines is called *software engineering.*

Software engineering is the discipline whose aim is:

1. Production of quality software
2. software that is delivered on time
3. cost within the budget
4. satisfies all requirements.

Software process is the way in which we produce the software. Apart from hiring smart, knowledgeable engineers and buying the latest development tools, an effective software development process is also needed, so that engineers can systematically use the best technical and managerial practices to successfully complete their projects.

A **software life cycle** is the series of identifiable stages that a software product undergoes during its lifetime .A software lifecycle model is a descriptive and diagrammatic representation of the software life cycle .A life cycle model represents all the activities required to make a software product transit through its lifecycle phases .It also captures the order in which these activities are to be taken .

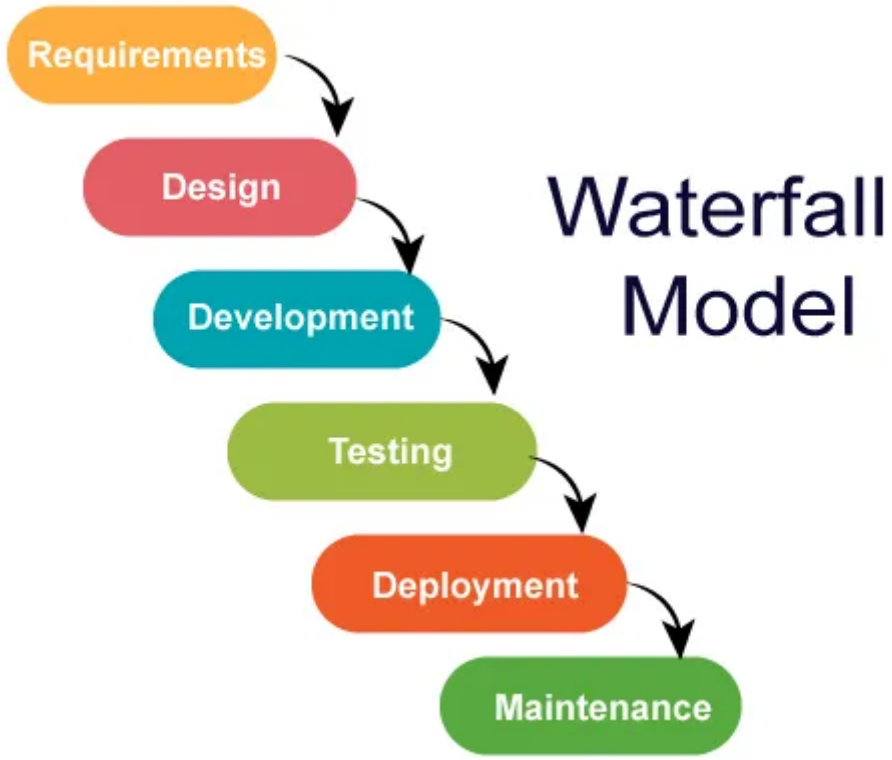
Life Cycle Models

There are various life cycle models to improve the software processes.

- Waterfall Model
- Prototype Model
- Spiral Model

In this Project we have used a **Waterfall Model**.

Waterfall Model



Waterfall Model

This model contains 6 phases:

- **Feasibility study**

  The feasibility study activity involves the analysis of the problem and collection of the relevant information relating to the product. The main aim of the feasibility study is to determine whether it would be financially and technically feasible to develop the product.

- **Requirement analysis and specification**

  The goal of this phase is to understand the exact requirements of the customer and to document them properly.(SRS)

- **Design**

  The goal of this phase is to transform the requirement specification into a structure that is suitable for implementation in some programming language.

- **Implementation and unit testing**

  During this phase the design is implemented. Initially small modules are tested in isolation from the rest of the software product.

- **Integration and system testing**

  In this all the modules are integrated and then tested altogether.

- **Operation and maintenance**

The phases always occur in this order and do not overlap.

WHY WE USE WATERFALL MODEL

The waterfall model remains a relevant choice today because of its straightforward and streamlined approach to development. It may not be suitable for every task or industry, but it also sets the stage to ensure completed work transfers to each step instead of relying on impartial results.

When we have small teams to manage with consistently predictable projects to complete, then this methodology provides a significant number of benefits to consider. If our teams are larger or the work is unpredictable, then another approach might produce better results for us

System Design

The process used to collect information and data for the purpose of making a business settlement. The methodology can include disclosure of the research, interviews, surveys and the other research study techniques, and could include the same present and historical information.

- **Planning**

The group conducted an interview to gather data information. We have noticed that by updating their products they have to write manuals in the record books and check the remaining items. For our proposed system, this will help to make it faster and accurate to control the inventory of their products by updating and adding using this computerized sales and inventory system.

- **Analysis**

Doing the computerized sales and inventory system takes several risks, first the submission of the letter to the company, and also preparing for the problem that the system might encounter especially with regards to the security, unauthorized access and or problem to the operator's computer system, this includes viruses that might pop in the system.

- **Design**

The software design is the development that matches the environment of the Group for them to have an easy system. The proposed system will be a major change for the establishment. All the transactions will be computerized. With the use of the new system, it will be much easier to modify information because the database of the system holds all the messages and a user-friendly interface is provided to customize the data.

- **Implementation**

After completing all the documents about the system, conducted test, and training on how to use the system and finally complete ready to be delivered in the to be used for their business work.

- **Software Development**

The development of the system and the programming style or way that the developers do their work.

- **Testing**

The group conducted some tests to make sure that the system is free of bugsn and errors, this includes performance and other testing procedures. It turned out good because the expected result came out to be actual results of the program. The testing is done every time. After adding some code in the system. There are times that the expected result happens.

- **Maintenance**

The maintenance will be conducted once the unit is delivered and deployed to the client to make sure that our system has better performance. If the company suggests updating some feature or adding some function for the system to make it more efficient and reliable to them, they just need to contact the maintenance.

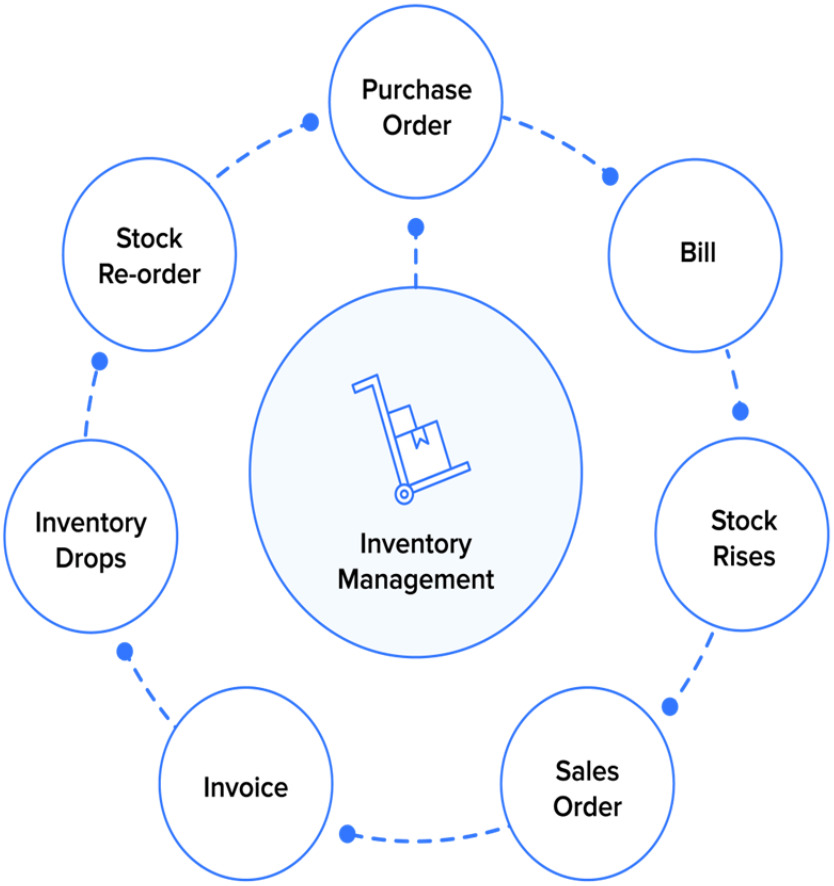
Process Flow Diagram

The Inventory Management Flow Chart outlines the steps involved in managing inventory. The first step is to receive inventory and verify that it matches the purchase order. By receiving inventory properly, businesses can ensure that they have accurate information about their inventory levels and avoid errors or discrepancies.

The next step in the **inventory management process** is to order inventory. This involves determining the amount of inventory needed and placing an order with the supplier. By ordering inventory properly, businesses can ensure that they have adequate inventory levels to meet customer demand.

The next step in the inventory management process is to update inventory records. This involves recording the receipt of new inventory and adjusting inventory levels in the system. By updating inventory records properly, businesses can ensure that they have accurate information about their inventory levels and avoid errors or **discrepancies**.

During the inventory management process, it is also important for businesses to monitor inventory levels and make adjustments as necessary. This can involve conducting regular inventory counts, analyzing **inventory data**, and adjusting inventory levels based on changes in demand or supply. By monitoring **inventory levels**, businesses can ensure that they have adequate inventory levels to meet customer demand while avoiding excess inventory or stockouts.

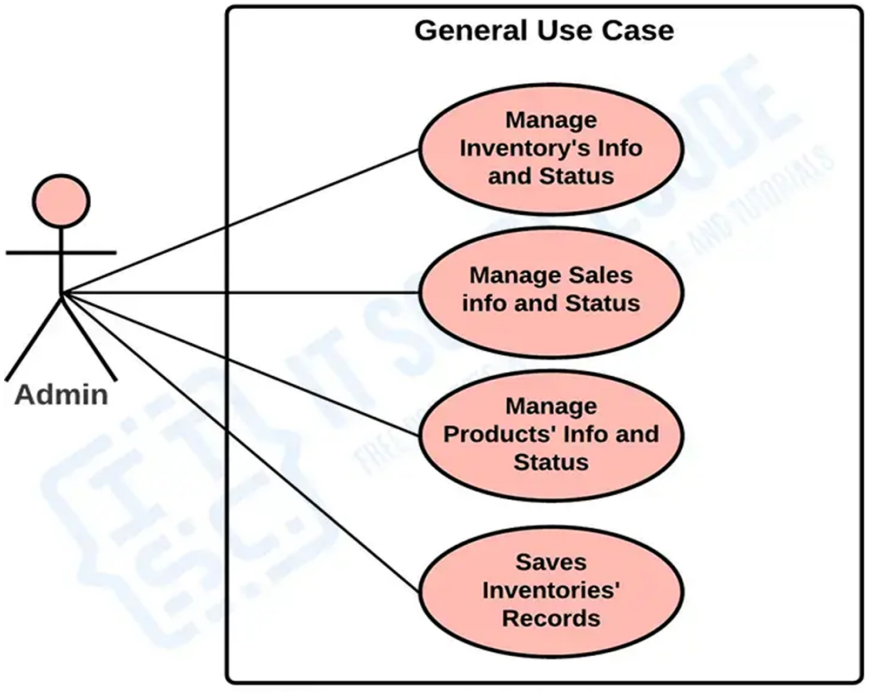

Purchase Order
Bill
Stock Rises
Sales Order
Invoice
Inventory Drops
Stock Re-order
Inventory Management

Use Case Diagram

The **use case diagram for the inventory management system** shows the sample behavior of the software. It includes the project functions using use cases, actors, and their connections.

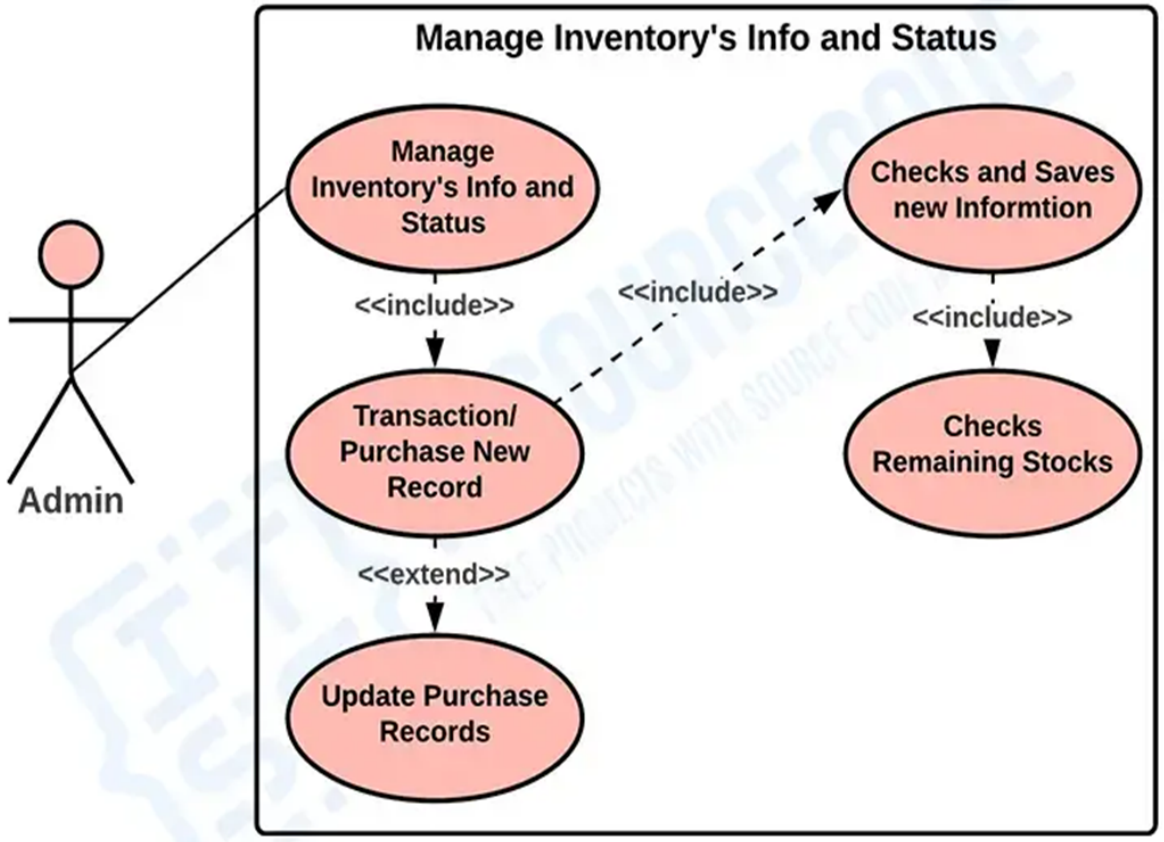
- **GENERAL USE CASE**

The general use case is the most common application of a use case diagram. The image below depicts the main components of the UML use case diagram for the Inventory Management System.



- **MONITOR AND MANAGE INVENTORY INFORMATION AND STATUS**

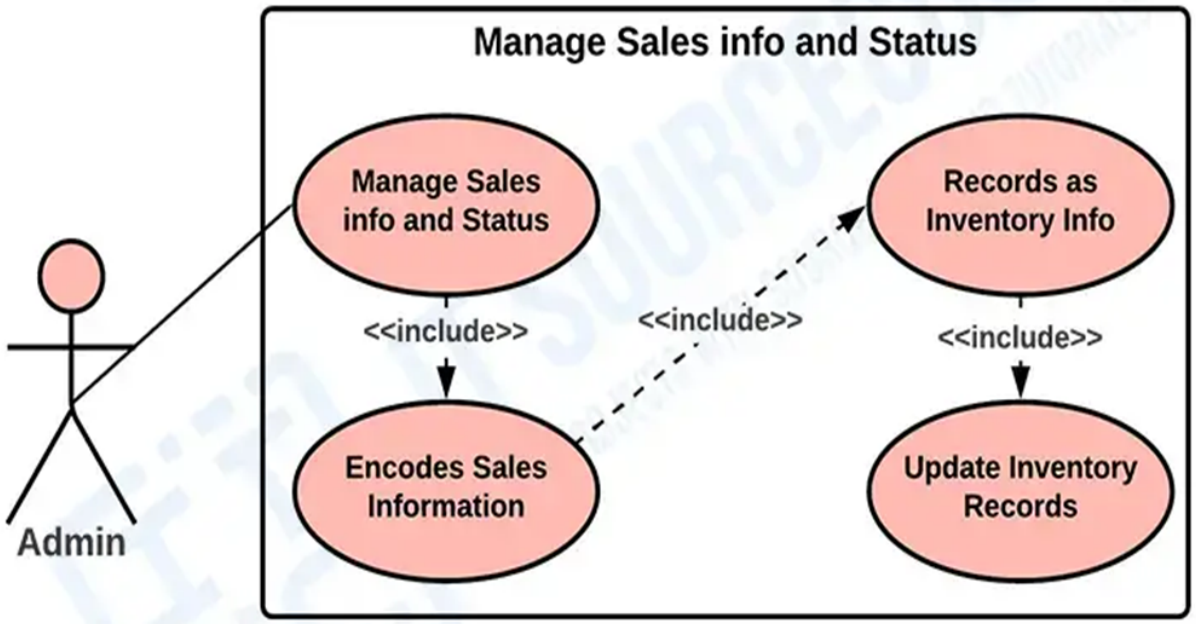
This function lets the admin monitor the purchases or sales information and the company’s inventories.



-

- **MANAGE SALES INFO AND STATUS OF USE CASE**

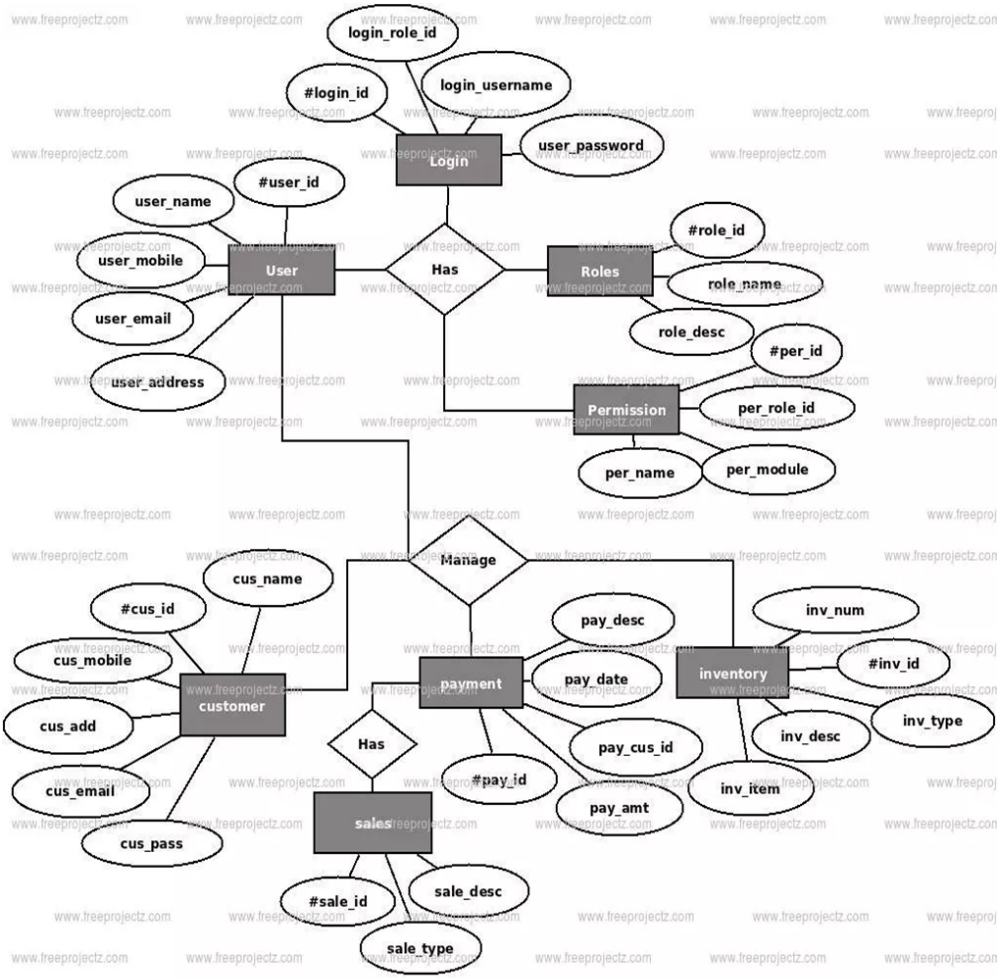
This is the process where the admin manages every transaction done by the customer/client. The data records are the reference for future use.



ER Diagram

ER diagram for inventory management system depicts the key concepts and relationships required for inventory resource management. It is not a complete data model showing every necessary relational database table, nor is it intended to be a prescriptive design for resource management system implementations.

Entity keys are used to identify instances of entities in a unique way. Candidate keys are attributes with unique values, and one of them is designated as the primary key. The attribute domains should be pre-defined. The entities in an ER diagram for inventory management system are the Fundamental entity, Subordinate entity, Associative entity, Generalisation entity, and Aggregation entity.



ER Diagram For Sales And Inventory Management System

DFD Diagram

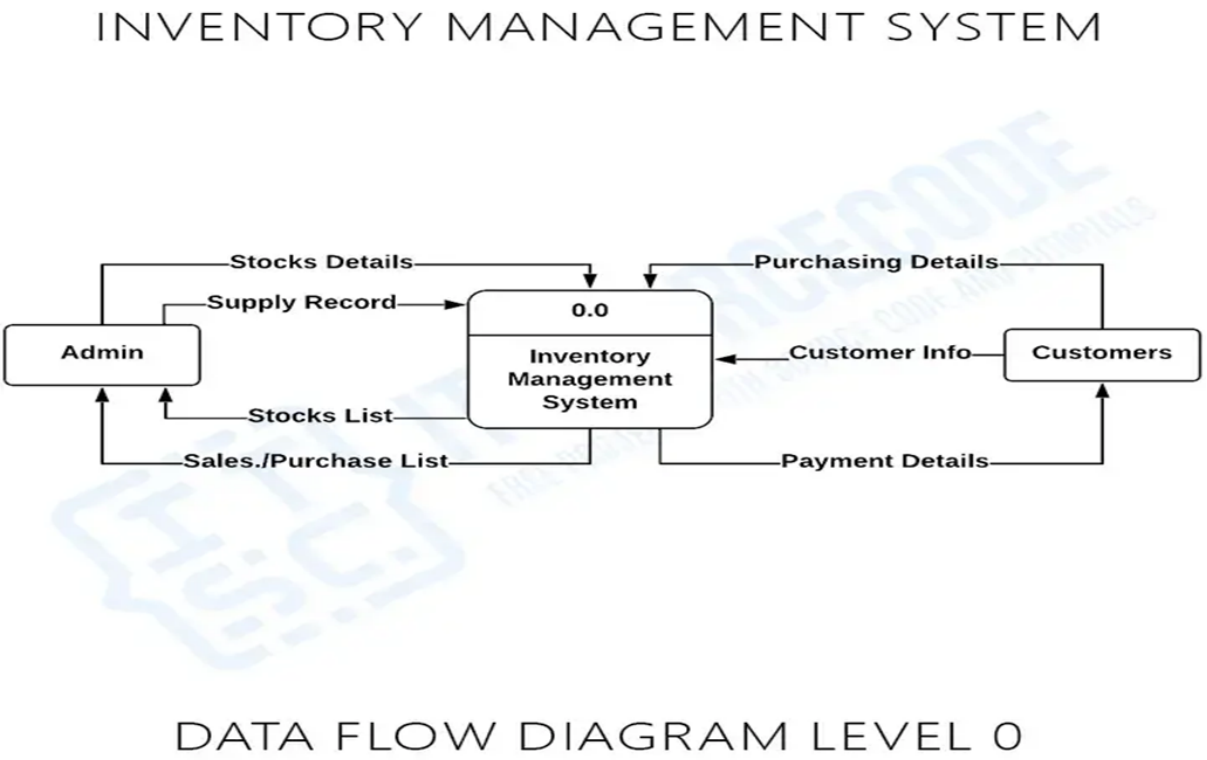
The **Data Flow Diagram (DFD)** depicts the data flow and transformations that occur when data enters and exits a system. This **DFD** represents and describes the **inventory management system as including input, processing,** and **output .**This **DFD** represents and describes the **inventory management system as a** whole, including **input, processing,** and **output.**

The **inventory management system DFD** consists of **DFD levels 0, 1,** and **2.** It also uses **entities, processes,** and **data** to define the whole system.

- **0 Level DFD for Inventory Management System**

The parameters of the inventory management system are specified in a context diagram (**level 0 data-flow diagram**). It illustrates how information moves between the system and its external entities. Through **DFD level 0,** the entire concept of the system is demonstrated in a single process.

For example:



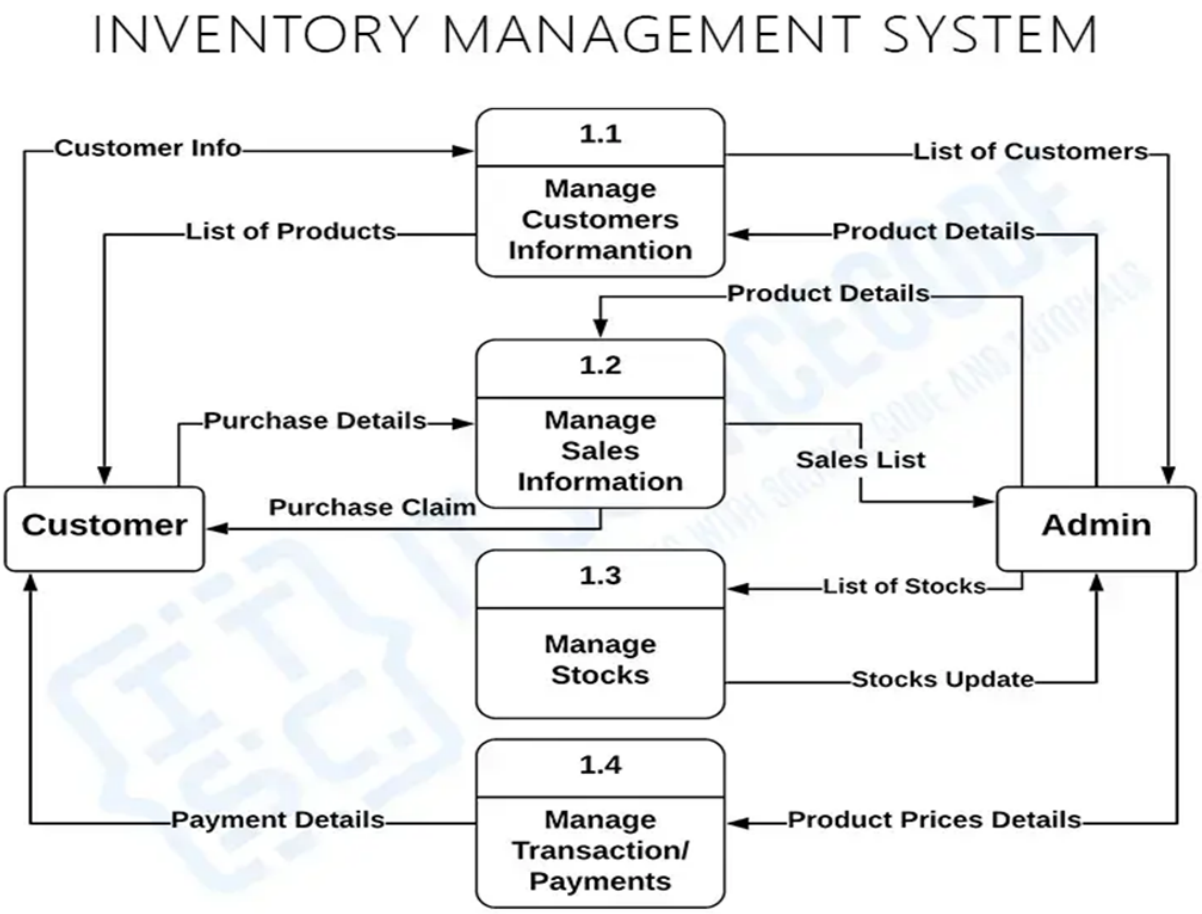
The external entities that cause the system to perform a certain function are as follows:

- **Customers**
- **System Admin**

As a result, the following illustrations begin at the system's DFD level 0.

- **Level 1 DFD for Inventory Management System**

  The **first level DFD of the inventory management system** describes each of the system's primary sub-processes. This level represents the context diagram's "**extended viewpoint**."



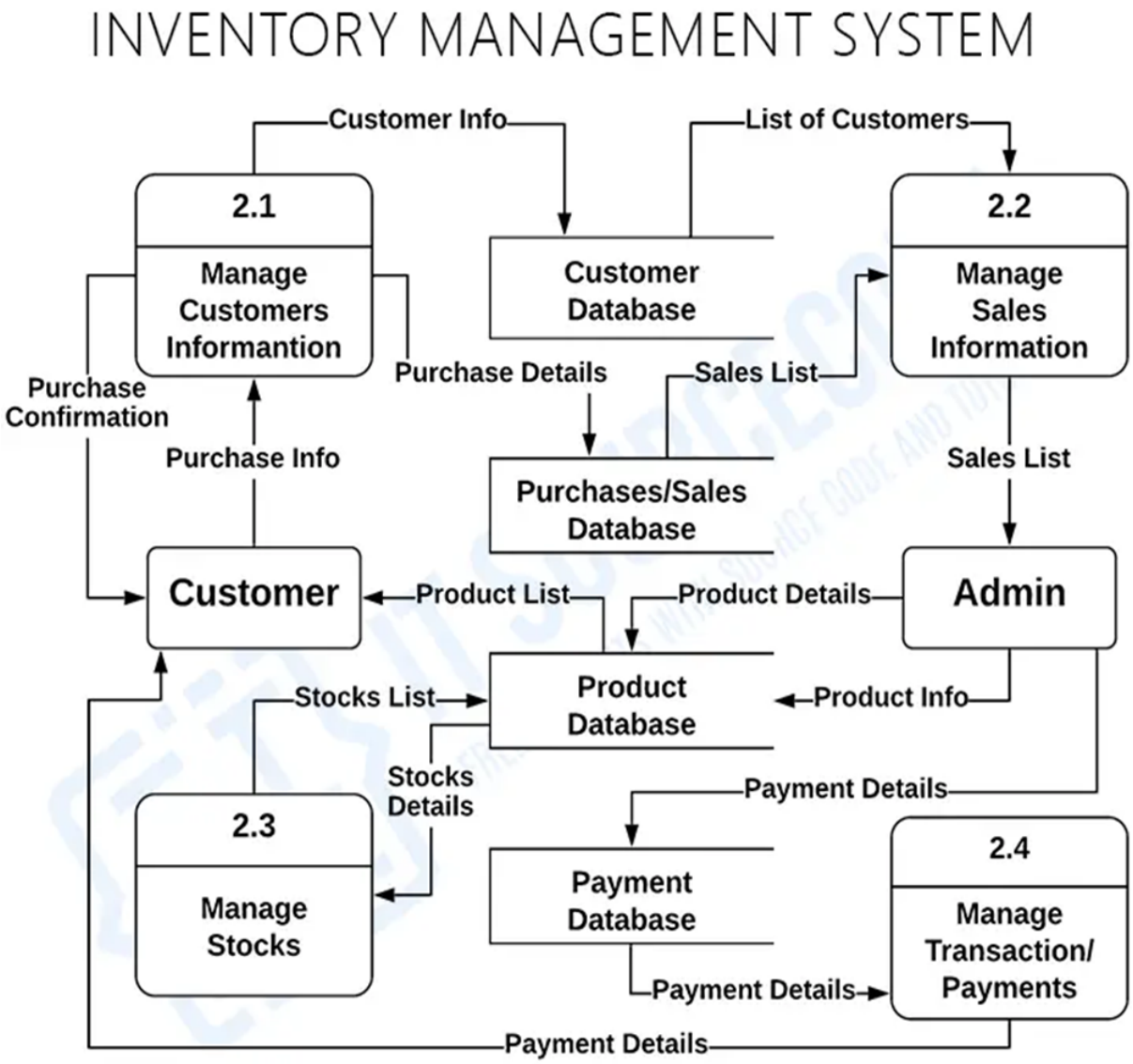
DATA FLOW DIAGRAM LEVEL 1

- **DFD Level 2 for Inventory Management System**

Among the preceding levels, **DFD level 2** has the highest concept abstraction. The reason for this is that this level describes the processes (if any) that fall under the level 1 sub-process.

At this point, let's focus on a crucial aspect of the data flow diagram.

For example, Databases



Testing

Software testing is the process of executing a program with the intention of finding errors in the code. It is a process of evolution of a system or its parts by manual or automatic means to verify that it is satisfying specified requirements or not.

Generally, no system is perfect due to communication problems between user and developer, time constraints, or conceptual mistakes by developer.

The purpose of system testing is to check and find out these errors or faults as early as possible so losses due to it can be saved.

Testing is the fundamental process of software success.

Testing is not a distinct phase in system development life cycle but should be applicable throughout all phases i.e. design development and maintenance phase.

Testing is used to show incorrectness and considered to be successful when an error is detected.

Strategy For Software Testing

Different levels of testing are used in the test process; each level of testing aims to test different aspects of the system.

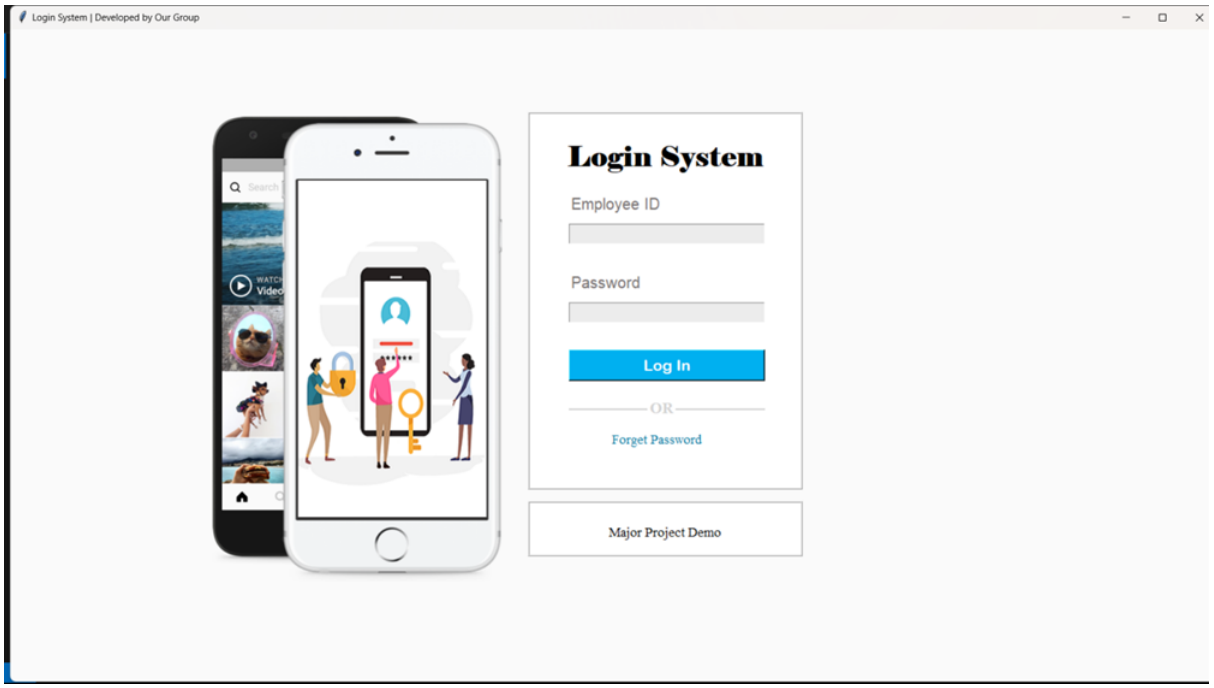
- The first level is ***unit testing***. In this testing, individual components are tested to ensure that they operate correctly. It focuses on verification efforts.

- The second level is ***integration testing***. It is a systematic technique for constructing the program structure. In this testing, many tested modules are combined into the subsystem which are then tested. The good thing here is to see if the modules can be integrated properly.

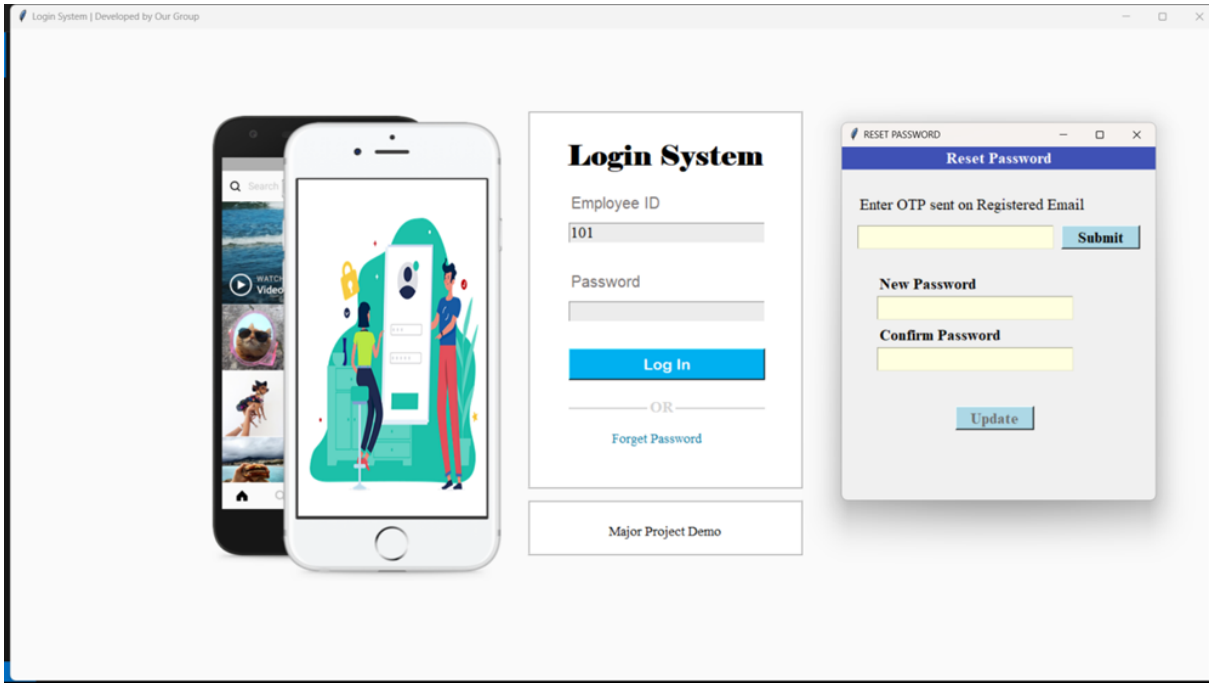
- Third level is ***integration testing***. System testing is actually a series of different tests whose primary purpose is to fully exercise computer based systems. These tests fall outside scope of the software process and are not conducted solely by software engineers.

Screenshots

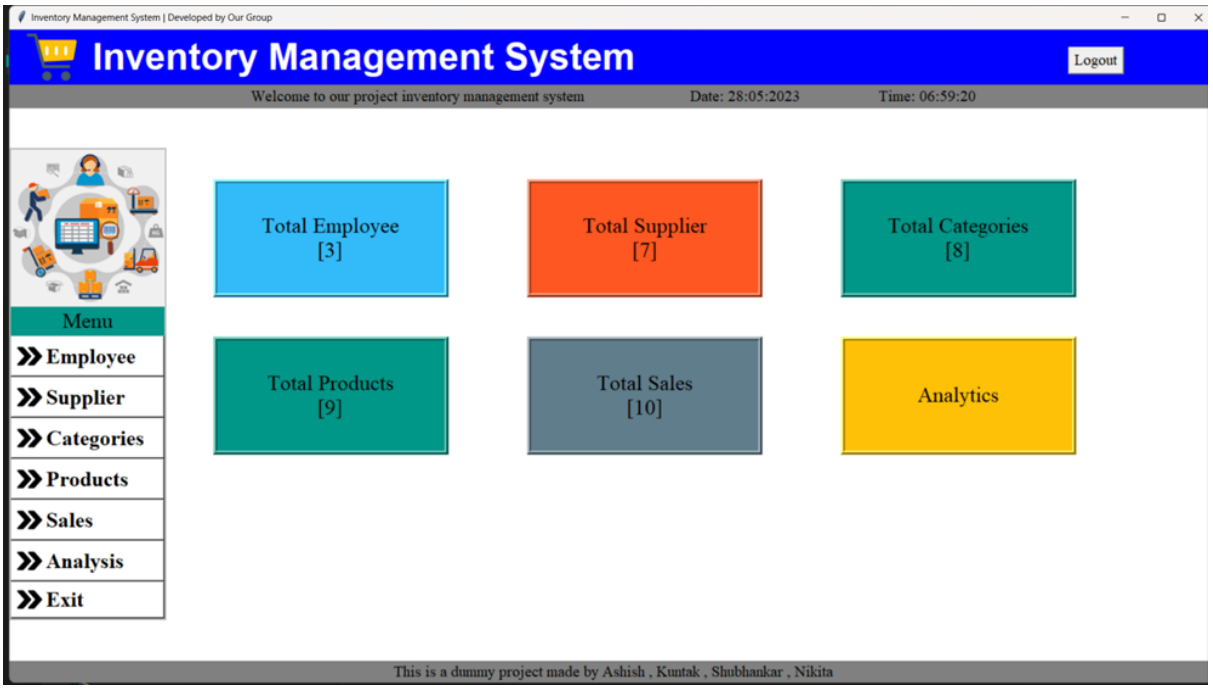
LogIn Page



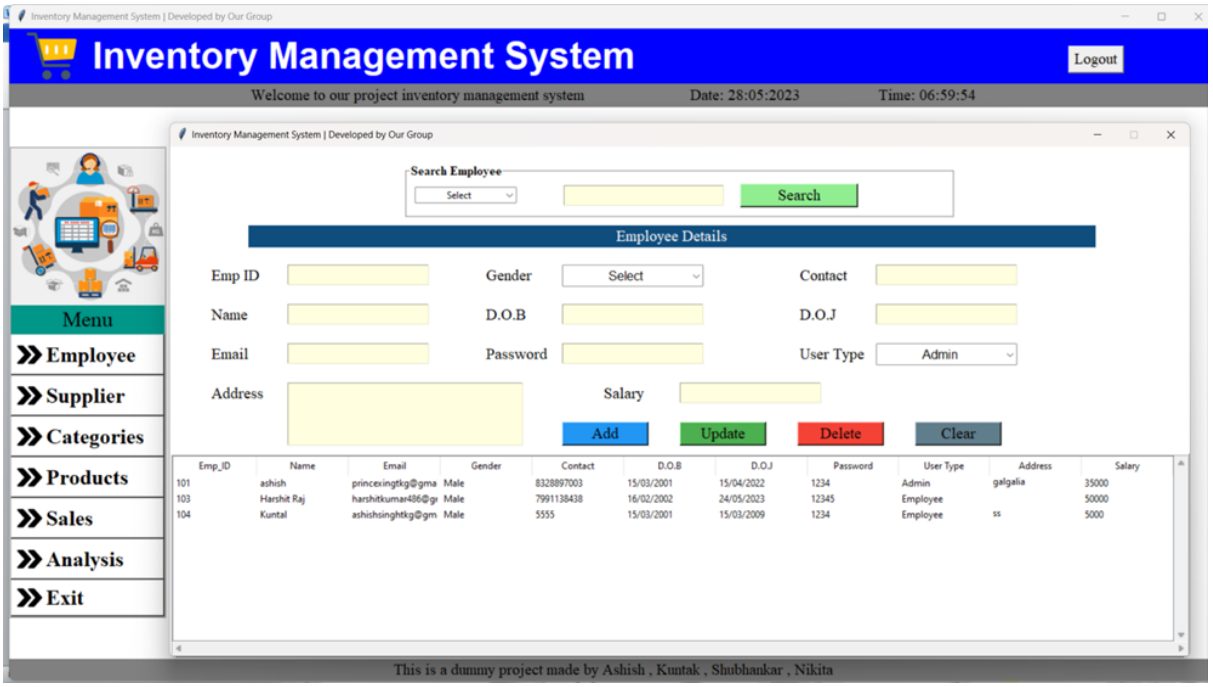
Password Forget Page



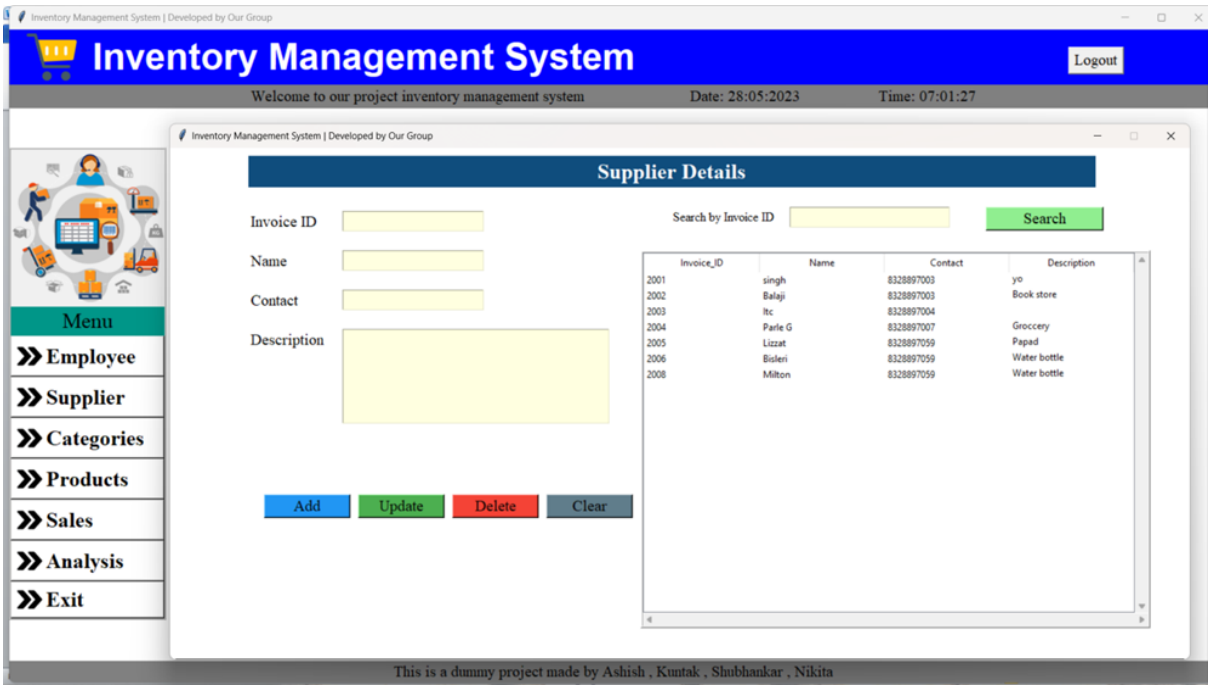
LogIn As Admin



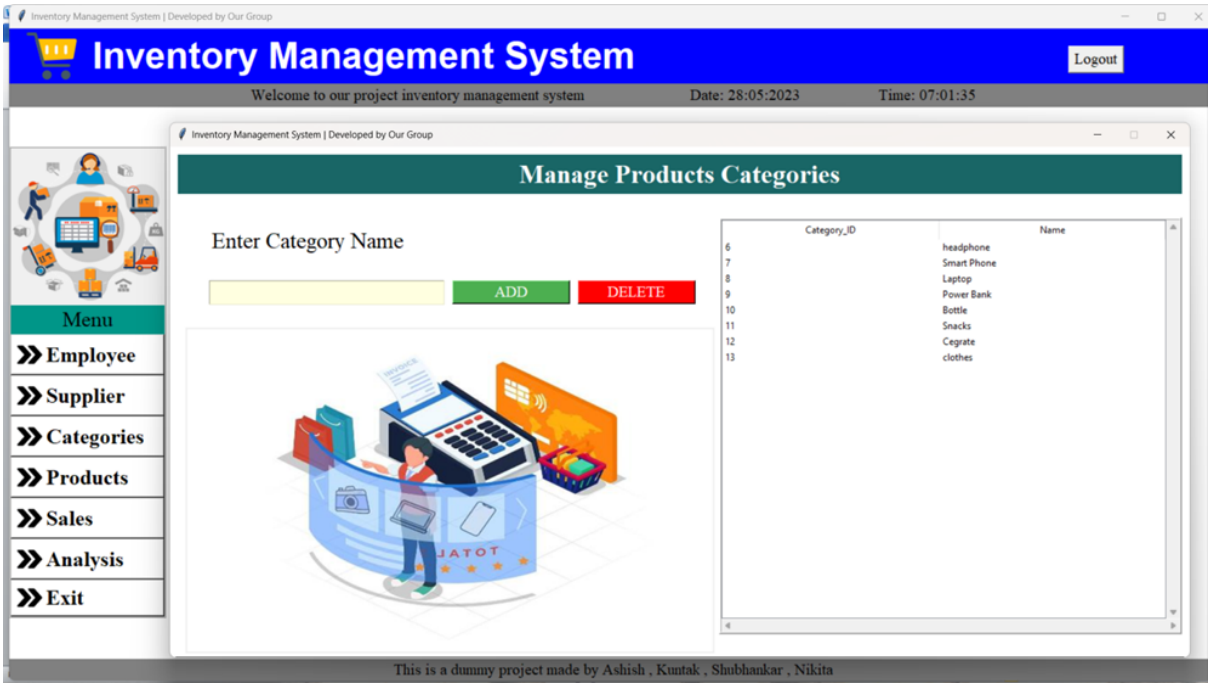
Employee



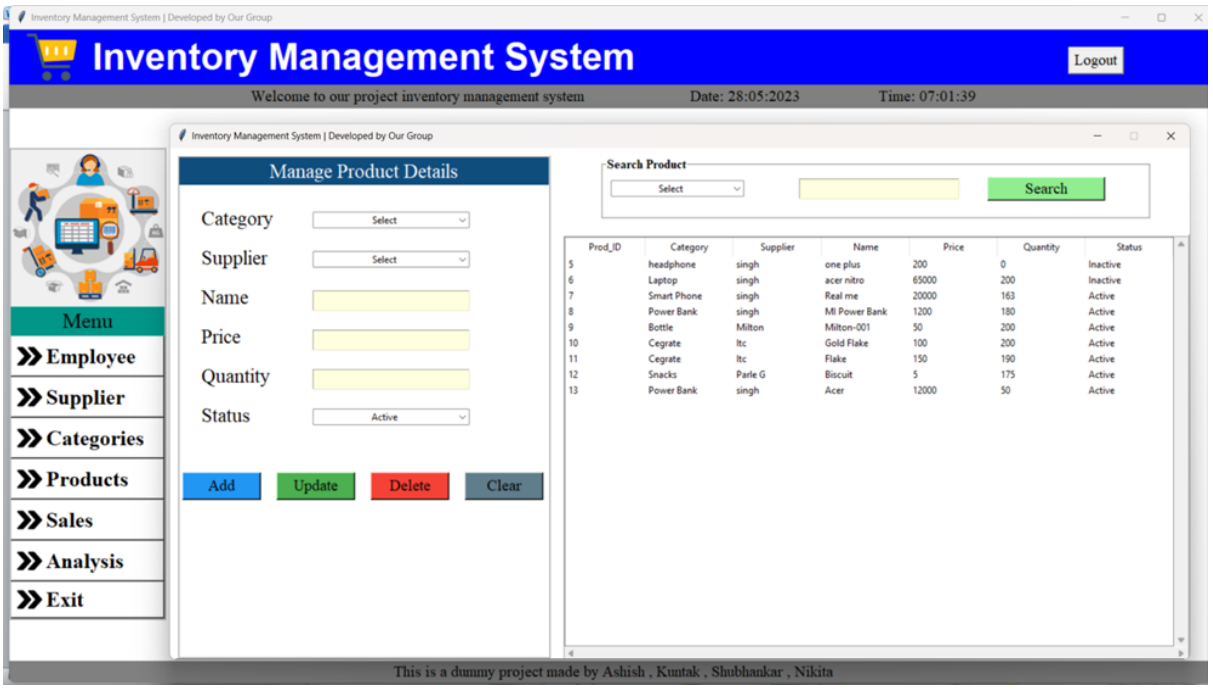
Supplier



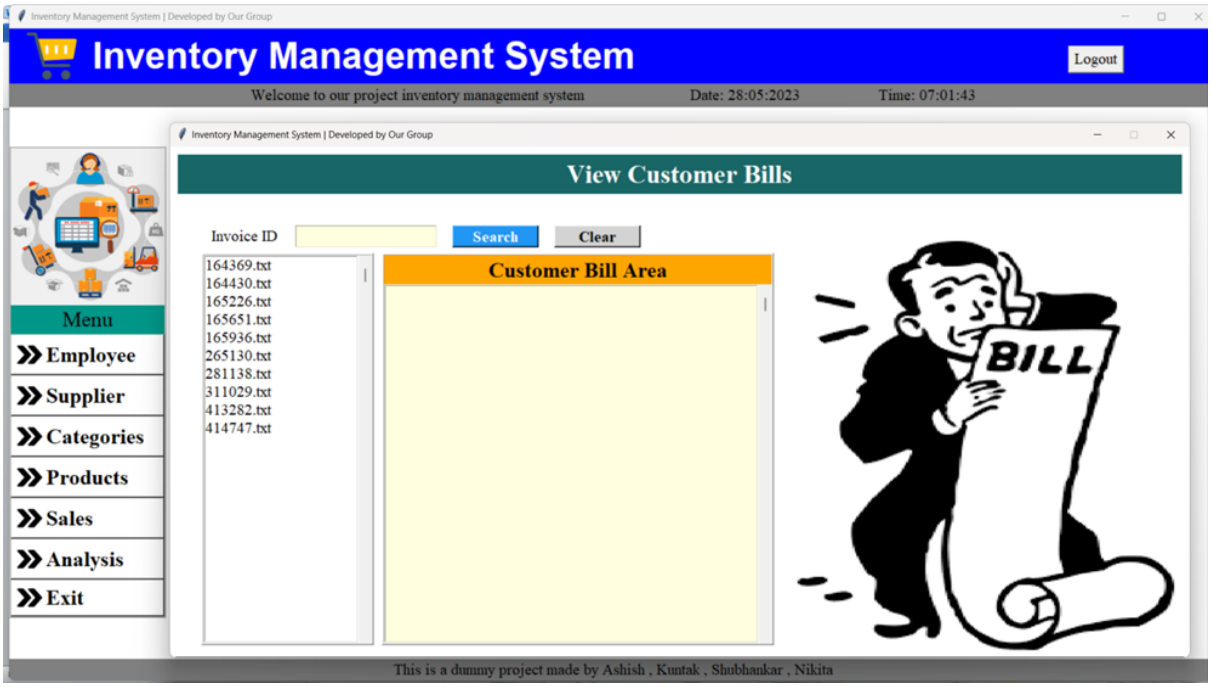
Category

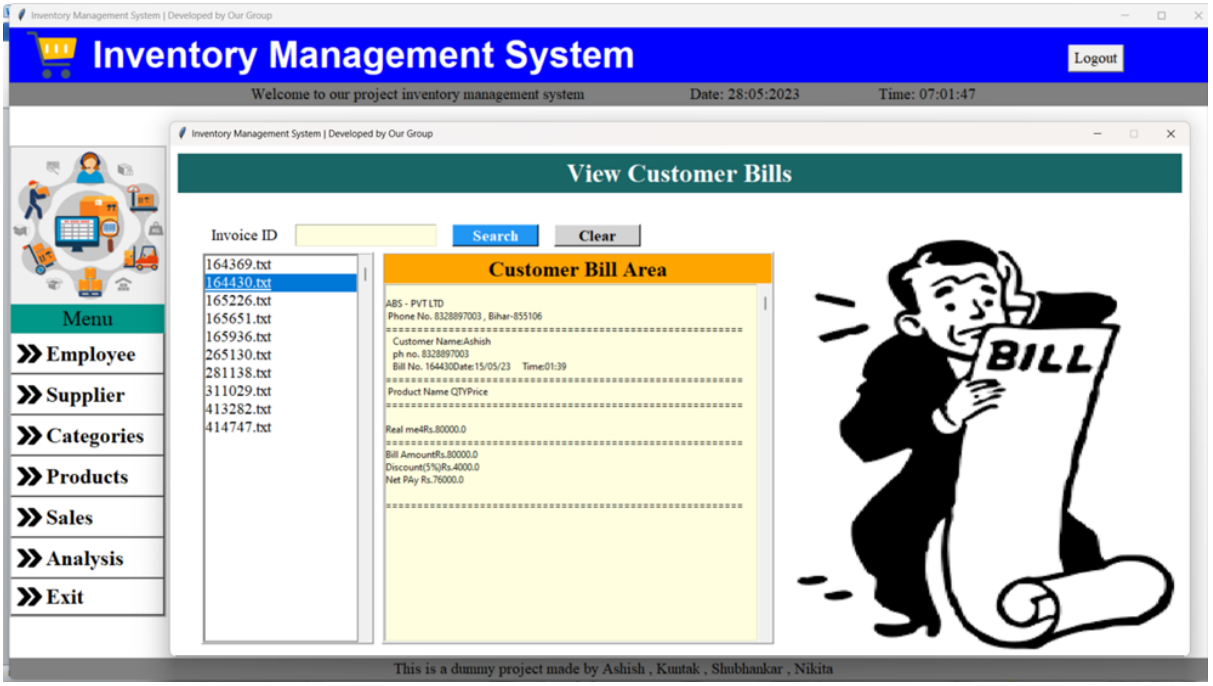


Products

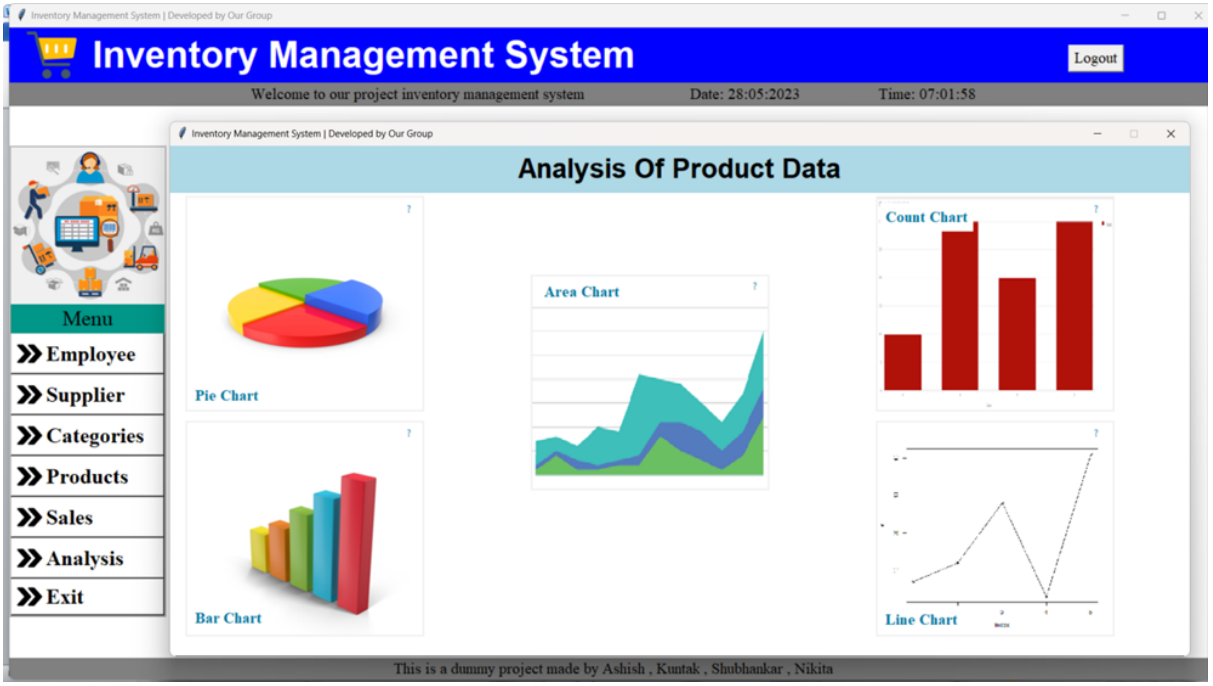


Sales

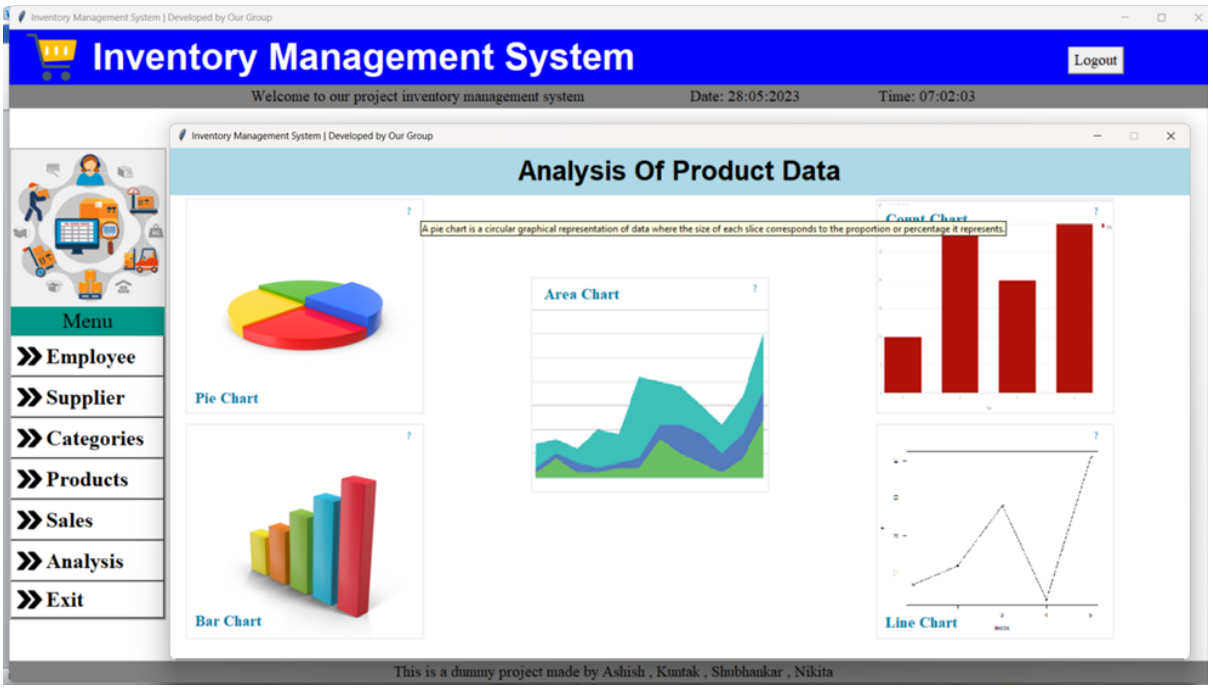


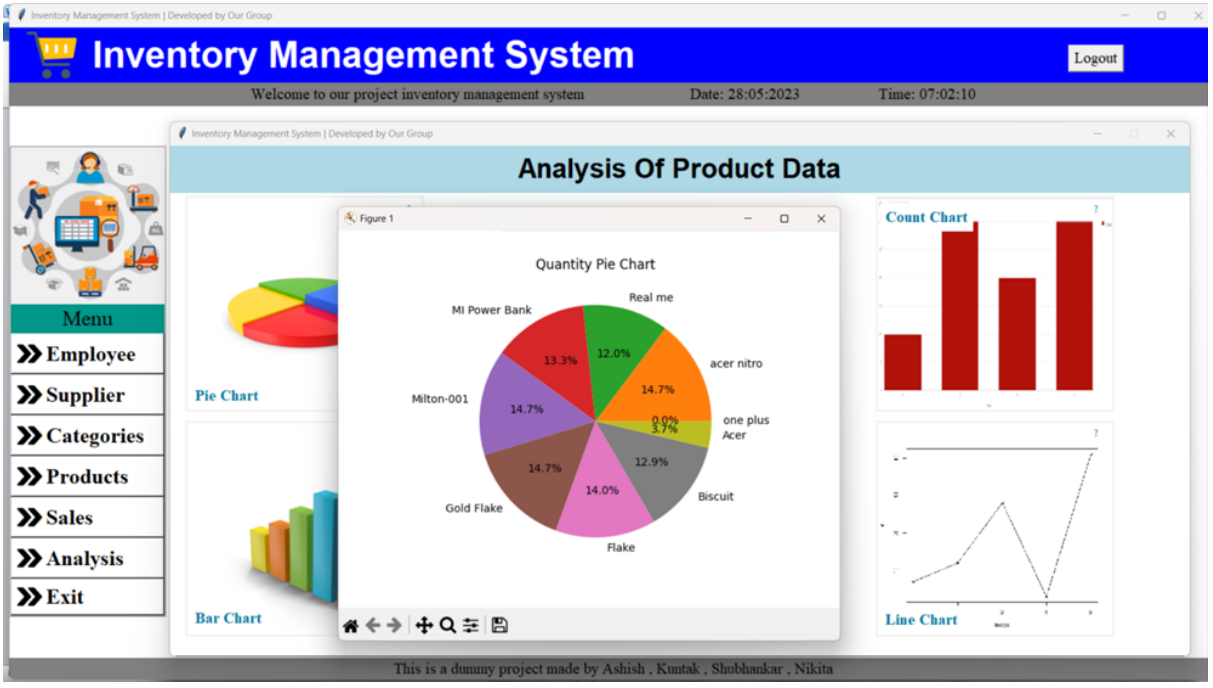

Inventory Management System | Developed by Our Group
Inventory Management System
Logout
Welcome to our project inventory management system
Date: 28:05:2023
Time: 07:01:47
Inventory Management System | Developed by Our Group
View Customer Bills
Invoice ID
Search
Clear
164369.txt
164430.txt
165226.txt
165651.txt
165936.txt
265130.txt
281138.txt
311029.txt
413282.txt
414747.txt
Customer Bill Area
ABS - PVT LTD
Phone No. 8328897003 , Bihar-855106
Customer Name:Ashish
ph no. 8328897003
Bill No. 164430Date:15/05/23 Time:01:39
Product Name QTYPrice
Real me4Rs.80000.0
Bill AmountRs.80000.0
Discount(5%)Rs.4000.0
Net PAy Rs.76000.0
BILL
Menu
Employee
Supplier
Categories
Products
Sales
Analysis
Exit
This is a dummy project made by Ashish , Kuntak , Shubhankar , Nikita

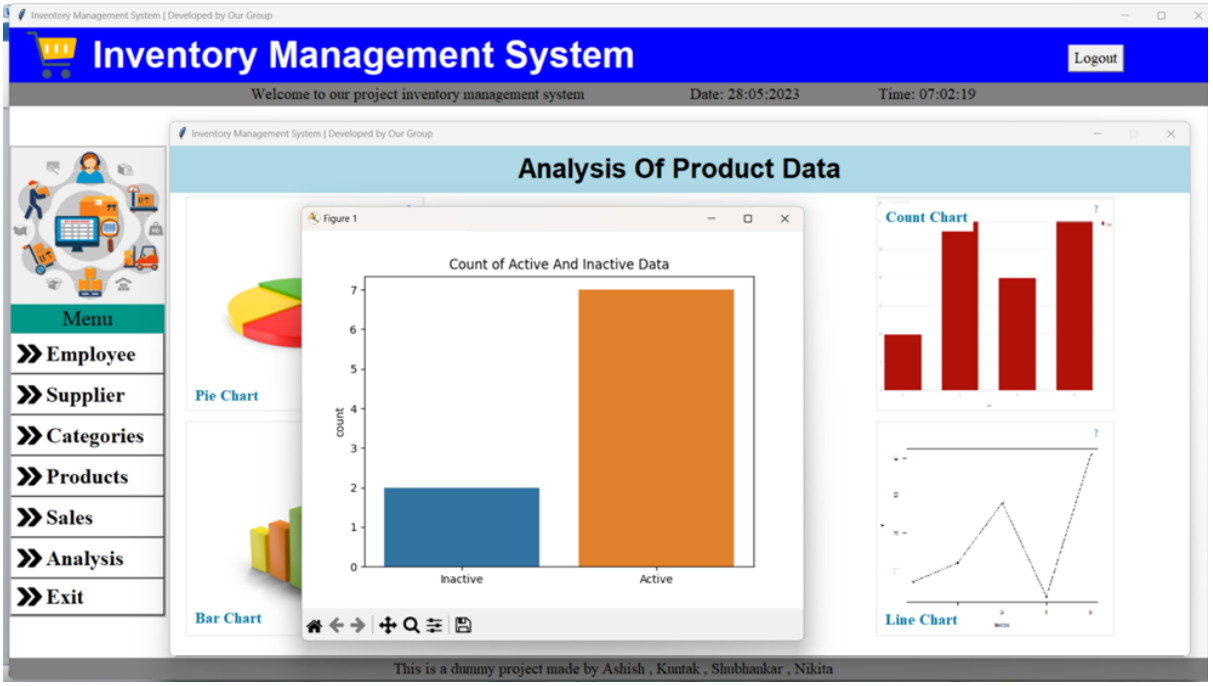
Analysis


Inventory Management System | Developed by Our Group
Inventory Management System
Logout
Welcome to our project inventory management system
Date: 28:05:2023
Time: 07:01:58
Inventory Management System | Developed by Our Group
Analysis Of Product Data
Pie Chart
Area Chart
Count Chart
Bar Chart
Line Chart
Menu
Employee
Supplier
Categories
Products
Sales
Analysis
Exit
This is a dummy project made by Ashish , Kuntak , Shubhankar , Nikita

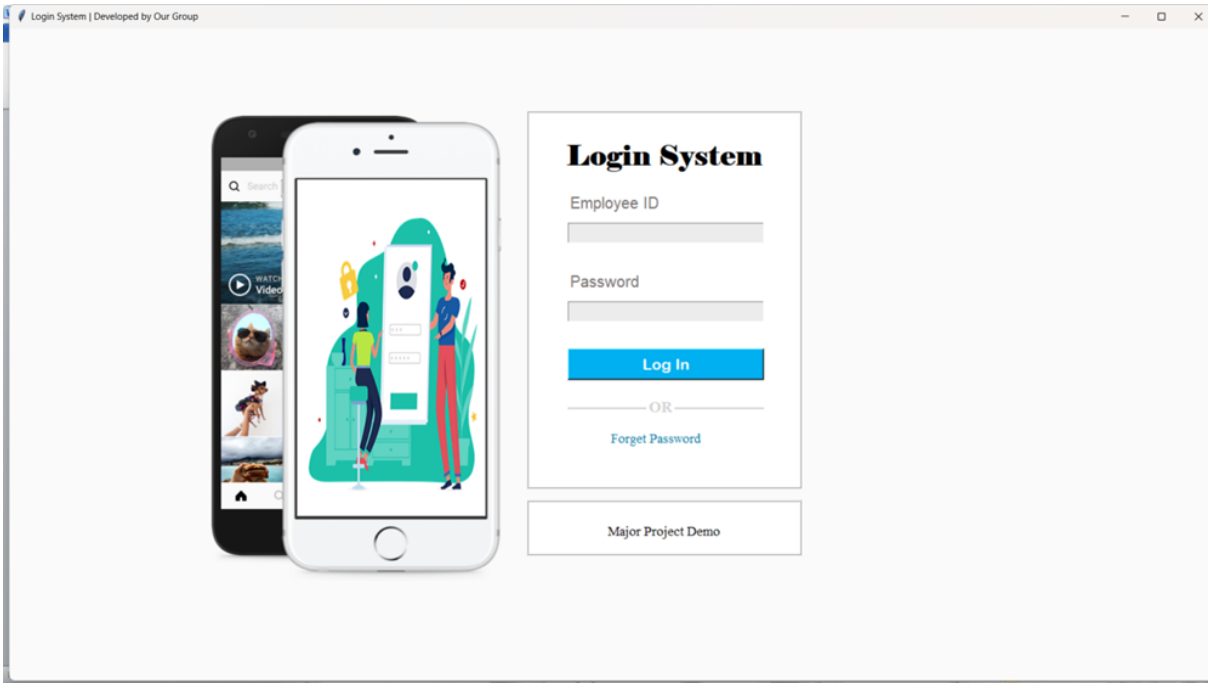
In Analysis We Have Added



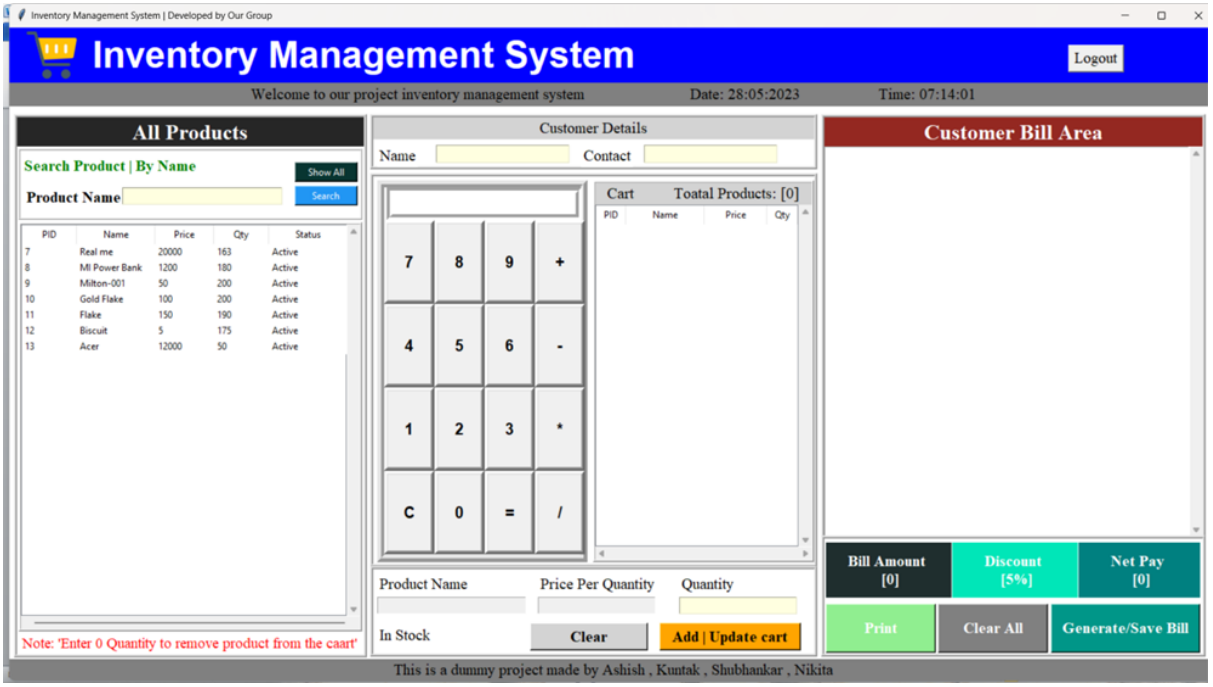


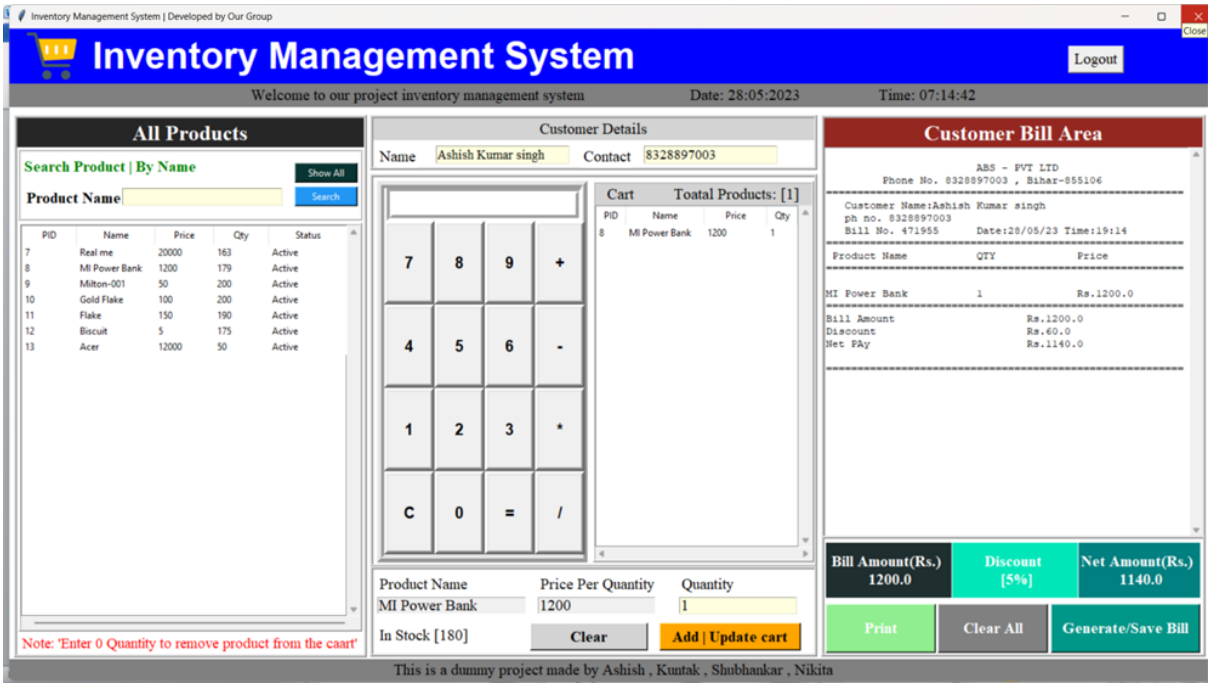


LogOut And Exit



LogIn As Employee





Source Code

LogIn.py

```
from tkinter import*

from PIL import ImageTk

import time

import sqlite3

from tkinter import messagebox

import email_

import smtplib #pip install smtplib

import os

class Login_System:

        def _init_(self,root):

        self.root=root

    self.root.geometry("1900x830+0+0")

    #self.root.resizable(False, False)

    self.root.config(bg="#fafafa")

    self.root.focus_force()

    self.otp="

    self.root.title("Login System | Developed by Our Group")

        #----------images

        self.photo_image=ImageTk.PhotoImage(file="E:\Self Study\Major
Project\images\phone.png")

    self.lbl_photo_img=Label(self.root,image=self.photo_image,bd=0).place(x=200,y=90)

        #---------variables
```

```
        self.employee_id=StringVar()

        self.password=StringVar()

#----------login frame

        login_frame=Frame(self.root,bd=2,relief=RIDGE,bg="white")

            login_frame.place(x=660,y=105,width=350,height=480)

        labl_login_system=Label(login_frame,text="Login
System",font=("Elephant",26,"bold"),bg="white").place(x=0,y=30,relwidth=1)

        labl_user_name=Label(login_frame,text="Employee
ID",font=("Andalus",15),bg="white",fg="#767171").place(x=50,y=100)

        txt_user_name=Entry(login_frame,textvariable=self.employee_id,font=("times new
roman",15),bg="#ECECEC").place(x=50,y=140,width=250)

labl_user_password=Label(login_frame,text="Password",font=("Andalus",15),bg="white",fg
="#767171").place(x=50,y=200)

    txt_user_password=Entry(login_frame,textvariable=self.password,show="*",font=("times
new roman",15),bg="#ECECEC").place(x=50,y=240,width=250)

        btn_login=Button(login_frame,command=self.login_,text="Log In",font=("Arial
Rounded MT
Bold",15),fg="white",bg="#00B0F0",cursor="hand2",activebackground="#00B0F0",activefo
reground="white").place(x=50,y=300,width=250,height=40)

        hr=Label(login_frame,bg="lightgray").place(x=50,y=375,width=250,height=2)

        or_=Label(login_frame,text="OR",font=("times new
roman",15,"bold"),bg="white",fg="lightgray").place(x=150,y=360)



        btn_forget=Button(login_frame,command=self.forget_window,text="Forget
Password",font=("times new



roman",13),bg="white",fg="#00759E",bd=0,activebackground="white",activeforeground="#
00759E",cursor="hand2").place(x=100,y=400)

            #----frame 2

                frame2=Frame(self.root,bd=2,relief=RIDGE,bg="white")
```

```
        frame2.place(x=660,y=600,width=350,height=70)
    txt_label=Label(frame2,text="Major Project Demo",font=("times new
roman",13),bg="white",justify=CENTER).place(x=0,y=25,relwidth=1)
         #------Animation
    self.im1=ImageTk.PhotoImage(file="E:\Self Study\Major Project\images\im1.png")
    self.im2=ImageTk.PhotoImage(file="E:\Self Study\Major Project\images\im2.png")
    self.im3=ImageTk.PhotoImage(file="E:\Self Study\Major Project\images\im3.png")
    self.lbl_change_img=Label(self.root,bg="white")
    self.lbl_change_img.place(x=367,y=192,width=240,height=428)
        self.animate()
        def animate(self):
        self.im=self.im1
        self.im1=self.im2
        self.im2=self.im3
    self.im3=self.im
    self.lbl_change_img.config(image=self.im)
        self.lbl_change_img.after(2000,self.animate)
        def login_(self):
    con=sqlite3.connect(database='ims.db')
         cur=con.cursor()
        try:
        if self.employee_id.get()=="" or self.password.get()=="":
            messagebox.showerror("Error","All fields are required",parent=self.root)
        cur.execute("select user_type from employee where emp_id=? and password=?",(
            self.employee_id.get(),
            self.password.get()
```

```
        ))
        user=cur.fetchone()
        if user==None:
            messagebox.showerror("Error","Invalid Employee ID | Password",parent=self.root)
        else:
                if user[0]=="Admin":
                self.root.destroy()
                os.system("py Dashboard.py")
                else:
                self.root.destroy()
                os.system("py billing.py")
        except Exception as ex:
        messagebox.showerror("Error",f"Error due to :{str(ex)}")
        def forget_window(self):
    con=sqlite3.connect(database='ims.db')
        cur=con.cursor()
        try:
        if self.employee_id.get()=="":
                messagebox.showerror("Error","Employee ID required",parent=self.root)
        else:



            cur.execute("select email from employee where
emp_id=?",(self.employee_id.get(),))
                email=cur.fetchone()
                if email==None:
```

```
            messagebox.showerror("Error","Invalid Employee ID , Try
again",parent=self.root)

                else:

            self.var_otp=StringVar()

                self.var_new_pass=StringVar()

            self.var_cnf_pass=StringVar()


                #call send email function

            chk=self.send_email(email[0])

                if chk !='s':

                messagebox.showerror("Error","connection error,Try again",parent=self.root)

                else:

                self.forget_win=Toplevel(self.root)

                self.forget_win.title('RESET PASSWORD')

                self.forget_win.geometry('400x450+1060+150')

                  self.forget_win.focus_force()

                title=Label(self.forget_win,text="Reset Password",font=("times new
roman",15,"bold"),bg="#3f51b5",fg="white").pack(side=TOP,fill=X)

                reset_lbl=Label(self.forget_win,text="Enter OTP sent on Registered
Email",font=("times new roman",15)).place(x=20,y=60)

                txt_reset=Entry(self.forget_win,textvariable=self.var_otp,font=("times new
roman",15,"bold"),bg="lightyellow").place(x=20,y=100,width=250,height=30)



self.btn_reset=Button(self.forget_win,text="Submit",command=self.validate_otp,font=("time
s new roman",15,"bold"),bg="lightblue")

                self.btn_reset.place(x=280,y=100,width=100,height=30)

                new_pass=Label(self.forget_win,text="New Password",font=("times new
roman",15,"bold")).place(x=45,y=160)
```

```
txt_new_pass=Entry(self.forget_win,textvariable=self.var_new_pass,font=("times new roman",15),bg="lightyellow").place(x=45,y=190,width=250,height=30)

                cnf_new_pass=Label(self.forget_win,text="Confirm Password",font=("times new roman",15,"bold")).place(x=45,y=225)
txt_cnf_new_pass=Entry(self.forget_win,textvariable=self.var_cnf_pass,font=("times new roman",15),bg="lightyellow").place(x=45,y=255,width=250,height=30)



self.btn_update=Button(self.forget_win,command=self.update_pss,text="Update",font=("times new roman",15,"bold"),bg="lightblue",state=DISABLED)

                self.btn_update.place(x=145,y=330,width=100,height=30)

        except Exception as ex:

        messagebox.showerror("Error",f"Error due to :{str(ex)}")

        def update_pss(self):

        if self.var_new_pass.get()=="" or self.var_cnf_pass.get()=="":

        messagebox.showerror("Error","Password is required",parent=self.forget_win)

        elif self.var_new_pass.get() != self.var_cnf_pass.get():

        messagebox.showerror("Error","Password must be same",parent=self.forget_win)

        else:

        con=sqlite3.connect(database='ims.db')



cur=con.cursor()

        try:

            cur.execute("update employee set password=? where emp_id=?",(self.var_new_pass.get(),self.employee_id.get()))

                con.commit()

            messagebox.showinfo("Success",'Password updated successfully',parent=self.forget_win)
```

```
            self.forget_win.destroy()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to :{str(ex)}")
        def validate_otp(self):
        if int(self.otp)==int(self.var_otp.get()):
        self.btn_update.config(state=NORMAL)
        self.btn_reset.config(state=DISABLED)
        else:
        messagebox.showerror("Error","Invalid OTP,try again",parent=self.forget_win)
        def send_email(self,to_):
    s=smtplib.SMTP_SSL('smtp.gmail.com',465)
        #s.starttls() 487
    get_email=email_.email_
    get_pass=email_.pass_
    s.login(get_email,get_pass)
    self.otp=int(time.strftime("%H%S%M"))+int(time.strftime("%S")
        subj='IMS-Reset Password OTP'
        msg=f'Dear Sir / Madam ,\n\nYour OTP to reset password is {str(self.otp)}.\n\nWith
Regards,\nIMS Team'
        msg="Subject:{}\n\n{}".format(subj,msg)
    s.sendmail(get_email,to_,msg)
    chk=s.ehlo()
        if chk[0]==250:
        return "s"
        else:
        return 'f'
```

```
if _name=="__main_":
        root=Tk()
        obj=Login_System(root)


    root.mainloop()
```

Dashboard.py

```
from tkinter import*
from Employee import EmployeeClass
from Supplier import SupplierClass
from Category import CategoryClass
from Products import ProductClass
from Sales       import SalesClass
from analysis import AnalysisClass
import time
import sqlite3
from tkinter import messagebox
import os
class IMS:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1900x830+0+0")
        self.root.config(bg="white")
        self.root.focus_force()
        self.root.title("Inventory Management System | Developed by Our Group")


        #----TITLE
        self.icon_title=PhotoImage(file="E:\Self Study\Major Project\images\logo1.png")
        title=Label(self.root,text="Inventory Management
System",image=self.icon_title,compound=LEFT,font=("Famtasy",35,"bold"),fg="white",bg=
"blue",anchor="w",padx=20).place(x=0,y=0,height=70,relwidth=1)
        #-----Logout button
```

```
        btn_logout=Button(self.root,command=self.logout,text="Logout",font=("times new
roman",15),cursor="hand2").place(x=1350,y=22,height=35)



        #----Making Date As well as time

        self.lbl_clock=Label(self.root,text="Welcome to our project inventory management
system\t\tDate:DD-MM-YYYY\t\tTime: HH:MM:SS",font=("times new
roman",15),bg="grey",fg="black")

        self.lbl_clock.place(x=0,y=70,height=30,relwidth=1)

        #---Left menu button

        self.menu_logo=PhotoImage(file="E:\Self Study\Major Project\images\menu_im.png")

        left_menu=Frame(self.root,bd=2,relief=RIDGE,bg="white")

        left_menu.place(x=0,y=150,height=600,width=200)

        #logo

        lbl_menu_logo=Label(left_menu,image=self.menu_logo)

        lbl_menu_logo.pack(side=TOP,fill=X)

        #side icon

        self.side_icon=PhotoImage(file="E:\Self Study\Major Project\images\side.png")

        #menu button

        lbl_menu=Label(left_menu,text="Menu",font=("times new
roman",20),bg="#009688").pack(side=TOP,fill=X)


btn_employee=Button(left_menu,text="Employee",image=self.side_icon,padx=5,compound=
LEFT,command=self.Employee,anchor="w",font=("times new
roman",20,"bold"),bg="white",bd=2,cursor="hand2").pack(side=TOP,fill=X)


btn_supplier=Button(left_menu,text="Supplier",image=self.side_icon,padx=5,compound=LE
FT,command=self.Supplier,anchor="w",font=("times new
roman",20,"bold"),bg="white",bd=2,cursor="hand2").pack(side=TOP,fill=X)


btn_categories=Button(left_menu,text="Categories",image=self.side_icon,padx=5,compound
```

```
=LEFT,command=self.Category,anchor="w",font=("times new
roman",20,"bold"),bg="white",bd=2,cursor="hand2").pack(side=TOP,fill=X)


btn_products=Button(left_menu,text="Products",image=self.side_icon,padx=5,compound=L
EFT,command=self.Product,anchor="w",font=("times new
roman",20,"bold"),bg="white",bd=2,cursor="hand2").pack(side=TOP,fill=X)


btn_sales=Button(left_menu,text="Sales",image=self.side_icon,padx=5,compound=LEFT,co
mmand=self.Sales,anchor="w",font=("times new
roman",20,"bold"),bg="white",bd=2,cursor="hand2").pack(side=TOP,fill=X)


btn_analysis=Button(left_menu,text="Analysis",image=self.side_icon,padx=5,compound=LE
FT,command=self.analysis,anchor="w",font=("times new
roman",20,"bold"),bg="white",bd=2,cursor="hand2").pack(side=TOP,fill=X)


btn_exit=Button(left_menu,text="Exit",image=self.side_icon,padx=5,compound=LEFT,com
mand=self.logout,anchor="w",font=("times new
roman",20,"bold"),bg="white",bd=2,cursor="hand2").pack(side=TOP,fill=X)

#--contents

    self.lbl_employee=Label(self.root,text="Total Employee\n [0]",font=("times new
roman",20),bd=5,relief=RIDGE,bg="#33bbf9")

    self.lbl_employee.place(x=260,y=190,height=150,width=300)

    self.lbl_supplier=Label(self.root,text="Total Supplier\n [0]",font=("times new
roman",20),bd=5,relief=RIDGE,bg="#ff5722")

    self.lbl_supplier.place(x=660,y=190,height=150,width=300)

    self.lbl_categories=Label(self.root,text="Total Categories\n [0]",font=("times new
roman",20),bd=5,relief=RIDGE,bg="#009688")

    self.lbl_categories.place(x=1060,y=190,height=150,width=300)

    self.lbl_products=Label(self.root,text="Total Products\n [0]",font=("times new
roman",20),bd=5,relief=RIDGE,bg="#009688")

    self.lbl_products.place(x=260,y=390,height=150,width=300)

    self.lbl_sales=Label(self.root,text="Total Sales\n [0]",font=("times new
roman",20),bd=5,relief=RIDGE,bg="#607d8b")
```

```
        self.lbl_sales.place(x=660,y=390,height=150,width=300)

        self.lbl_analysis=Label(self.root,text="Analytics",font=("times new
roman",20),bd=5,relief=RIDGE,bg="#ffc107")

        self.lbl_analysis.place(x=1060,y=390,height=150,width=300)




#----Footer

        lbl_footer=Label(root,text="This is a dummy project made by Ashish , Kuntak ,
Shubhankar , Nikita",font=("times new
roman",15),bg="grey",fg="black").pack(side=BOTTOM,fill=X)

        self.update_content()

#=========================================

    def Employee(self):

        self.new_win=Toplevel(self.root)

        self.new_obj=EmployeeClass(self.new_win)

    def Supplier(self):

        self.new_win=Toplevel(self.root)

        self.new_obj=SupplierClass(self.new_win)

    def Category(self):

        self.new_win=Toplevel(self.root)

        self.new_obj=CategoryClass(self.new_win)



    def Product(self):

        self.new_win=Toplevel(self.root)

        self.new_obj=ProductClass(self.new_win)

    def Sales(self):
```

```
        self.new_win=Toplevel(self.root)
        self.new_obj=SalesClass(self.new_win)
    def analysis(self):
        self.new_win=Toplevel(self.root)
        self.new_obj=AnalysisClass(self.new_win)

    def update_content(self):
        con=sqlite3.connect(database='ims.db')
        cur=con.cursor()
        try:
            cur.execute("select * from product")
            product=cur.fetchall()
            self.lbl_products.config(text=f"Total Products\n[{str(len(product))}]")
            cur.execute("select * from employee")
            employee=cur.fetchall()
            self.lbl_employee.config(text=f"Total Employee\n[{str(len(employee))}]")
            cur.execute("select * from supplier")
            supplier=cur.fetchall()
            self.lbl_supplier.config(text=f"Total Supplier\n[{str(len(supplier))}]")
            cur.execute("select * from categories")
            categories=cur.fetchall()
            self.lbl_categories.config(text=f"Total Categories\n[{str(len(categories))}]")



self.lbl_sales.config(text=f"Total Sales\n[{str(len(os.listdir('bill')))}]")
```

```
            Time=time.strftime("%I:%M:%S")
            date=time.strftime("%d:%m:%Y")



 self.lbl_clock.config(text=f"Welcome to our project inventory management system\t\tDate: {str(date)}\t\tTime: {str(Time)}")
            self.lbl_clock.after(100,self.update_content)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to :{str(ex)}")
    def logout(self):
        self.root.destroy()
        os.system("python login.py")
if __name__=="__main__":

    root=Tk()
    obj=IMS(root)
    root.mainloop()
```

Employee.py

```
from tkinter import*
from tkinter import ttk,messagebox
import sqlite3
class EmployeeClass:
    def __init__(self,root):
        self.root=root
        self.root.geometry("1300x650+205+150")
        self.root.config(bg="white")
        self.root.title("Inventory Management System | Developed by Our Group")
        self.root.resizable(False, False)
        self.root.focus_force()
        #-------All Variables
        self.var_searchby=StringVar()
        self.var_search_txt=StringVar()
        self.var_emp_id=StringVar()
        self.var_emp_name=StringVar()
        self.var_emp_email=StringVar()
        self.var_emp_gender=StringVar()
        self.var_emp_contact=StringVar()
        self.var_emp_dob=StringVar()
        self.var_emp_doj=StringVar()
        self.var_emp_password=StringVar()
        self.var_emp_user_type=StringVar()
        self.var_emp_address=StringVar()
```

```
        self.var_emp_salary=StringVar()



#====SEARCH BAR

        search_frame=LabelFrame(self.root,text="Search Employee",bg="White",font=("times new roman",12,"bold"))

        search_frame.place(x=300,y=20,width=700,height=70)

        #=====options


self.search=ttk.Combobox(search_frame,values=("Select","Email","Name","Contact"),textvariable=self.var_searchby,state="readonly",justify="center")

        self.search.place(x=10,y=10,width=130)

        self.search.current(0)

        #text field

        self.txt_search=Entry(search_frame,font=("times new roman",15),bg="lightyellow",textvariable=self.var_search_txt).place(x=200,y=8)

        #button in the search field


self.search_bttn=Button(search_frame,text="Search",command=self.search_data,font=("times new roman",15),bg="lightgreen",fg="black",cursor="hand2").place(x=425,y=6,width=150,height=30)

        #---title

        title=Label(self.root,text="Employee Details",fg="white",bg="#0f4d7d",font=("times new roman",15)).place(x=100,y=100,width=1080)

        #----content

        #------row 1

        lbl_emp_id=Label(self.root,text="Emp ID",font=("times new roman",15),bg="white").place(x=50,y=150)
```

```
        lbl_gender=Label(self.root,text="Gender",font=("times new
roman",15),bg="white").place(x=400,y=150)

        lbl_contact=Label(self.root,text="Contact",font=("times new
roman",15),bg="white").place(x=800,y=150)

        txt_emp_id=Entry(self.root,textvariable=self.var_emp_id,font=("times new
roman",15),bg="lightyellow").place(x=150,y=150,width=180)

        #lbl_gender=Entry(self.root,textvariable=self.var_emp_gender,font=("times new
roman",15),bg="white").place(x=450,y=150,width=180)


self.gender=ttk.Combobox(self.root,values=("Select","Male","Female","Other"),font=(15),te
xtvariable=self.var_emp_gender,state="readonly",justify="center")

        self.gender.place(x=500,y=150,width=180,height=28)

        self.gender.current(0)

        txt_contact=Entry(self.root,textvariable=self.var_emp_contact,font=("times new
roman",15),bg="lightyellow").place(x=900,y=150,width=180)

        #----row 2

        lbl_emp_name=Label(self.root,text="Name",font=("times new
roman",15),bg="white").place(x=50,y=200)

        lbl_dob=Label(self.root,text="D.O.B",font=("times new
roman",15),bg="white").place(x=400,y=200)

        lbl_doj=Label(self.root,text="D.O.J",font=("times new
roman",15),bg="white").place(x=800,y=200)




 txt_emp_name=Entry(self.root,textvariable=self.var_emp_name,font=("times new
roman",15),bg="lightyellow").place(x=150,y=200,width=180)

        txt_dob=Entry(self.root,textvariable=self.var_emp_dob,font=("times new
roman",15),bg="lightyellow").place(x=500,y=200,width=180)

        txt_doj=Entry(self.root,textvariable=self.var_emp_doj,font=("times new
roman",15),bg="lightyellow").place(x=900,y=200,width=180)
```

```
#----row 3

lbl_emp_email=Label(self.root,text="Email",font=("times new
roman",15),bg="white").place(x=50,y=250)

lbl_password=Label(self.root,text="Password",font=("times new
roman",15),bg="white").place(x=400,y=250)

lbl_usertype=Label(self.root,text="User Type",font=("times new
roman",15),bg="white").place(x=800,y=250)

txt_emp_email=Entry(self.root,textvariable=self.var_emp_email,font=("times new
roman",15),bg="lightyellow").place(x=150,y=250,width=180)

txt_password=Entry(self.root,textvariable=self.var_emp_password,font=("times new
roman",15),bg="lightyellow").place(x=500,y=250,width=180)

self.usertype=ttk.Combobox(self.root,values=("Admin","Employee"),font=(15),textvariable=
self.var_emp_user_type,state="readonly",justify="center")

self.usertype.place(x=900,y=250,width=180,height=28)

self.usertype.current(0)

#-----row 4

lbl_emp_address=Label(self.root,text="Address",font=("times new
roman",15),bg="white").place(x=50,y=300)

lbl_salary=Label(self.root,text="Salary",font=("times new
roman",15),bg="white").place(x=550,y=300)

self.txt_emp_address=Text(self.root,font=("times new roman",15),bg="lightyellow")

self.txt_emp_address.place(x=150,y=300,width=300,height=80)

txt_password=Entry(self.root,textvariable=self.var_emp_salary,font=("times new
roman",15),bg="lightyellow").place(x=650,y=300,width=180)
```

```
        #-----row 5 buttons

        self.add_bttn=Button(self.root,text="Add",command=self.add_data,font=("times new
roman",15),bg="#2196f3",fg="black",cursor="hand2").place(x=500,y=350,width=110,height
=30)


self.update_bttn=Button(self.root,text="Update",command=self.update_data,font=("times
new
roman",15),bg="#4caf50",fg="black",cursor="hand2").place(x=650,y=350,width=110,height
=30)

        self.delete_bttn=Button(self.root,text="Delete",command=self.delete_data,font=("times
new
roman",15),bg="#f44336",fg="black",cursor="hand2").place(x=800,y=350,width=110,height
=30)

        self.clear_bttn=Button(self.root,text="Clear",command=self.clear_data,font=("times
new
roman",15),bg="#607d8b",fg="black",cursor="hand2").place(x=950,y=350,width=110,height
=30)

        #---making triview

        #Employee details

        emp_frame=Frame(self.root,bd=3,relief=RIDGE)

        emp_frame.place(x=0,y=390,relwidth=1,height=259)

        scroly=Scrollbar(emp_frame,orient=VERTICAL)

        scrolx=Scrollbar(emp_frame,orient=HORIZONTAL)


self.EmployeeTable=ttk.Treeview(emp_frame,columns=("emp_id","name","email","gender",
"contact","dob","doj","password","user_type","address","salary"),yscrollcommand=scroly.set
,xscrollcommand=scrolx.set)



        scrolx.pack(side=BOTTOM,fill=X)

        scroly.pack(side=RIGHT,fill=Y)

        scrolx.config(command=self.EmployeeTable.xview)

        scroly.config(command=self.EmployeeTable.yview)
```

```
self.EmployeeTable.heading("emp_id",text="Emp_ID")
self.EmployeeTable.heading("name",text="Name")
self.EmployeeTable.heading("email",text="Email")
self.EmployeeTable.heading("gender",text="Gender")
self.EmployeeTable.heading("contact",text="Contact")

self.EmployeeTable.heading("dob",text="D.O.B")
self.EmployeeTable.heading("doj",text="D.O.J")
self.EmployeeTable.heading("password",text="Password")
self.EmployeeTable.heading("user_type",text="User Type")
self.EmployeeTable.heading("address",text="Address")
self.EmployeeTable.heading("salary",text="Salary")
self.EmployeeTable["show"]="headings"
self.EmployeeTable.column("emp_id",width=90)
self.EmployeeTable.column("name",width=100)
self.EmployeeTable.column("email",width=100)
self.EmployeeTable.column("gender",width=100)
self.EmployeeTable.column("contact",width=100)
self.EmployeeTable.column("dob",width=100)
self.EmployeeTable.column("doj",width=100)
self.EmployeeTable.column("password",width=100)
self.EmployeeTable.column("user_type",width=100)
self.EmployeeTable.column("address",width=100)
self.EmployeeTable.column("salary",width=100)
self.EmployeeTable.pack(fill=BOTH,expand=1)
```

```
        self.show()
        self.EmployeeTable.bind("<ButtonRelease-1>",self.get_data)
#-------------------------------------------------------------------------------------------------------------
-
    def add_data(self):
        conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()




try:
            if self.var_emp_id.get()=="" or self.var_emp_name.get()=="":
                messagebox.showerror("Error","All fields are required",parent=self.root)
            else:
                        cur.execute("select * from employee where
emp_id=?",(self.var_emp_id.get(),))
                        row=cur.fetchone()




 if row != None:
                    messagebox.showerror("Error","This Employee ID is already assigned")
                        else:
                        cur.execute("INSERT INTO
employee(emp_id,name,email,gender,contact,dob,doj,password,user_type,address,salary)
VALUES(?,?,?,?,?,?,?,?,?,?,?)",(
```

```
                    self.var_emp_id.get(),
                    self.var_emp_name.get(),
                    self.var_emp_email.get(),
                    self.var_emp_gender.get(),
                    self.var_emp_contact.get(),
                    self.var_emp_dob.get(),
                    self.var_emp_doj.get(),
                    self.var_emp_password.get(),
                    self.var_emp_user_type.get(),
                    self.txt_emp_address.get('1.0',END),
                    self.var_emp_salary.get()
                ))
                conn.commit()
            messagebox.showinfo("Success","Employee data added successfully",parent=self.root)
                self.show()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to :{str(ex)}")
    def show(self):
        conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
```

```
            cur.execute("SELECT * FROM employee")
            rows=cur.fetchall()
            self.EmployeeTable.delete(*self.EmployeeTable.get_children())
            for rows in rows:
                    self.EmployeeTable.insert('',END,values=rows)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to :{str(ex)}")
    def get_data(self,ev):
        f=self.EmployeeTable.focus()
        content=(self.EmployeeTable.item(f))
        row=content['values']

        self.var_emp_id.set(row[0]),
        self.var_emp_name.set(row[1]),
        self.var_emp_email.set(row[2]),
        self.var_emp_gender.set(row[3]),
        self.var_emp_contact.set(row[4]),
        self.var_emp_dob.set(row[5]),
        self.var_emp_doj.set(row[6]),
        self.var_emp_password.set(row[7]),
        self.var_emp_user_type.set(row[8]),
        self.txt_emp_address.delete('1.0',END),
        self.txt_emp_address.insert(END,row[9]),
        self.var_emp_salary.set(row[10])

#-------------UPDATING DATA
```

```
    def update_data(self):
        conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
            if self.var_emp_id.get()=="" or self.var_emp_name.get()=="":
                messagebox.showerror("Error","All fields are required",parent=self.root)
            else:
                    cur.execute("select * from employee where
emp_id=?",(self.var_emp_id.get(),))
                    row=cur.fetchone()
                    if row == None:
                    messagebox.showerror("Error","Invalid Employee ID")
                    else:
                    cur.execute("UPDATE employee set
name=?,email=?,gender=?,contact=?,dob=?,doj=?,password=?,user_type=?,address=?,salary
=? where emp_id=?",(


                            self.var_emp_name.get(),
                            self.var_emp_email.get(),
                            self.var_emp_gender.get(),
                            self.var_emp_contact.get(),
                            self.var_emp_dob.get(),
                            self.var_emp_doj.get(),
                            self.var_emp_password.get(),
                            self.var_emp_user_type.get(),
                              self.txt_emp_address.get('1.0',END),
                            self.var_emp_salary.get(),
```

```
                    self.var_emp_id.get()

                ))

                conn.commit()

            messagebox.showinfo("Success","Employee data updated
successfully",parent=self.root)

                self.show()

        except Exception as ex:

            messagebox.showerror("Error",f"Error due to :{str(ex)}")

#-------------Deleting the data

    def delete_data(self):




conn=sqlite3.connect(database=r'ims.db')

        cur=conn.cursor()

        try:

            if self.var_emp_id.get()=="" or self.var_emp_name.get()=="":

                messagebox.showerror("Error","All fields are required",parent=self.root)

            else:

                    cur.execute("select * from employee where
emp_id=?",(self.var_emp_id.get(),))

                    row=cur.fetchone()

                    if row == None:

                messagebox.showerror("Error","Invalid Employee ID")

                    else:

                op=messagebox.askyesno("Confirm","Do you Want to delete?",parent=self.root)
```

```
if op==True:
                    cur.execute("delete from employee where emp_id=?",(self.var_emp_id.get(),))
                    conn.commit()
                    messagebox.showinfo("Success","Employee ID Deleted
Successfully",parent=self.root)
                    self.clear_data()
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to :{str(ex)}")
#-------------clear
    def clear_data(self):
        self.var_emp_id.set(""),
        self.var_emp_name.set(""),
        self.var_emp_email.set(""),
        self.var_emp_gender.set("Select"),
        self.var_emp_contact.set(""),
        self.var_emp_dob.set(""),
        self.var_emp_doj.set(""),
        self.var_emp_password.set(""),
        self.var_emp_user_type.set("Admin"),
        self.txt_emp_address.delete('1.0',END),
        self.var_emp_salary.set("")
        self.var_search_txt.set("")
        self.var_searchby.set("Select")
        self.show()
```

```
#-------searching
    def search_data(self):
        conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
            if self.var_searchby.get()=="Select":
                messagebox.showerror("Error","Select search by option",parent=self.root)
            elif self.var_search_txt.get()=="":
                messagebox.showerror("Error","Search input required",parent=self.root)
else:
                    cur.execute("SELECT * FROM employee where " +self.var_searchby.get()+"
LIKE
'%"+self.var_search_txt.get()+"%'")
                    rows=cur.fetchall()
                    if len(rows)!=0:
                self.EmployeeTable.delete(*self.EmployeeTable.get_children())
                    for rows in rows:
                    self.EmployeeTable.insert(',END,values=rows)
                    else:
                messagebox.showerror("Error","No Record found",parent=self.root)
        except Exception as ex:
            messagebox.showerror("Error",f"Error due to :{str(ex)}")
if __name__=="__main__":


    root=Tk()
    obj=EmployeeClass(root)
```

```
root.mainloop()
```

Supplier.py

```
from tkinter import*

from tkinter import ttk,messagebox

import sqlite3

class SupplierClass:

        def __init__(self,root):

        self.root=root

    self.root.geometry("1300x650+205+150")

    self.root.config(bg="white")

    self.root.title("Inventory Management System | Developed by Our Group")

        self.root.resizable(False, False)

    self.root.focus_force()

        #-------All Variables

    self.var_searchby=StringVar()

    self.var_search_txt=StringVar()

    self.var_supp_invoice=StringVar()

    self.var_sup_name=StringVar()

            self.var_sup_contact=StringVar()

        #====SEARCH BAR

        #====options

    lbl_search=Label(self.root,text="Search by Invoice ID",bg="white",font=("times
new roman",12))

        lbl_search.place(x=640,y=75,width=130)

        #text field

    self.txt_search=Entry(self.root,font=("times new
roman",15),bg="lightyellow",textvariable=self.var_search_txt).place(x=790,y=75)
```

```
        #button in the search field


self.search_bttn=Button(self.root,text="Search",command=self.search_data,font=("ti
mes new
roman",15),bg="lightgreen",fg="black",cursor="hand2").place(x=1040,y=75,width=15
0,height=30)

        #---title

      title=Label(self.root,text="Supplier Details",fg="white",bg="#0f4d7d",font=("times
new roman",20,"bold")).place(x=100,y=10,height=40,width=1080)

        #----content

        #------row 1

      lbl_sup_invoice=Label(self.root,text="Invoice ID",font=("times new
roman",15),bg="white").place(x=100,y=80)

      txt_sup_invoice=Entry(self.root,textvariable=self.var_supp_invoice,font=("times
new roman",15),bg="lightyellow").place(x=220,y=80,width=180)

        #----row 2

      lbl_sup_name=Label(self.root,text="Name",font=("times new
roman",15),bg="white").place(x=100,y=130)



      txt_sup_name=Entry(self.root,textvariable=self.var_sup_name,font=("times new
roman",15),bg="lightyellow").place(x=220,y=130,width=180)



        #----row 3

      lbl_sup_contact=Label(self.root,text="Contact",font=("times new
roman",15),bg="white").place(x=100,y=180)

        txt_sup_contact=Entry(self.root,textvariable=self.var_sup_contact,font=("times
new roman",15),bg="lightyellow").place(x=220,y=180,width=180)

        #-----row 4

      lbl_sup_desc=Label(self.root,text="Description",font=("times new
roman",15),bg="white").place(x=100,y=230)

      self.txt_desc=Text(self.root,font=("times new roman",15),bg="lightyellow")
```

```
        self.txt_desc.place(x=220,y=230,width=340,height=120)



          #-----row 5 buttons

        self.add_bttn=Button(self.root,text="Add",command=self.add_data,font=("times
new
roman",15),bg="#2196f3",fg="black",cursor="hand2").place(x=120,y=440,width=110,
height=30)


self.update_bttn=Button(self.root,text="Update",command=self.update_data,font=("ti
mes new
roman",15),bg="#4caf50",fg="black",cursor="hand2").place(x=240,y=440,width=110,
height=30)


self.delete_bttn=Button(self.root,text="Delete",command=self.delete_data,font=("tim
es new
roman",15),bg="#f44336",fg="black",cursor="hand2").place(x=360,y=440,width=110,
height=30)


self.clear_bttn=Button(self.root,text="Clear",command=self.clear_data,font=("times
new
roman",15),bg="#607d8b",fg="black",cursor="hand2").place(x=480,y=440,width=110,
height=30)

          #---making triview

          #Supplier details

        sup_frame=Frame(self.root,bd=3,relief=RIDGE)

        sup_frame.place(x=600,y=130,height=480,width=650)

        scroly=Scrollbar(sup_frame,orient=VERTICAL)

        scrolx=Scrollbar(sup_frame,orient=HORIZONTAL)


self.SupplierTable=ttk.Treeview(sup_frame,columns=("invoice_id","name","contact","
desc"),yscrollcommand=scroly.set,xscrollcommand=scrolx.set)



        scrolx.pack(side=BOTTOM,fill=X)
```

```
        scroly.pack(side=RIGHT,fill=Y)
        scrolx.config(command=self.SupplierTable.xview)
        scroly.config(command=self.SupplierTable.yview)
        self.SupplierTable.heading("invoice_id",text="Invoice_ID")
        self.SupplierTable.heading("name",text="Name")
        self.SupplierTable.heading("contact",text="Contact")
        self.SupplierTable.heading("desc",text="Description")

        self.SupplierTable["show"]="headings"
        self.SupplierTable.column("invoice_id",width=90)
        self.SupplierTable.column("name",width=100)
        self.SupplierTable.column("contact",width=100)
          self.SupplierTable.column("desc",width=100)

        self.SupplierTable.pack(fill=BOTH,expand=1)

        self.SupplierTable.bind("<ButtonRelease-1>",self.get_data)
          self.show()
#-------------------------------------------------------------------------------------------------------
-
          def add_data(self):
        conn=sqlite3.connect(database=r'ims.db')
          cur=conn.cursor()
          try:
          if self.var_supp_invoice.get()=="" or self.var_sup_name.get()="":
```

```
                messagebox.showerror("Error","All fields are
required",parent=self.root)
        else:
            cur.execute("select * from supplier where
invoice_id=?",(self.var_supp_invoice.get(),))
            row=cur.fetchone()
                if row != None:
                messagebox.showerror("Error","This Invoice ID is already assigned")
                 else:
                cur.execute("INSERT INTO supplier(invoice_id,name,contact,desc)
VALUES(?,?,?,?)",(
                            self.var_supp_invoice.get(),
                            self.var_sup_name.get(),
                            self.var_sup_contact.get(),
                            self.txt_desc.get('1.0',END),
                 ))
               conn.commit()
                messagebox.showinfo("Success","Supplier data added
successfully",parent=self.root)
                self.show()
        except Exception as ex:
         messagebox.showerror("Error",f"Error due to :{str(ex)}")
        def show(self):
      conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
         cur.execute("SELECT * FROM supplier")
```

```
            rows=cur.fetchall()
            self.SupplierTable.delete(*self.SupplierTable.get_children())
          for rows in rows:
              self.SupplierTable.insert('',END,values=rows)
          except Exception as ex:
            messagebox.showerror("Error",f"Error due to :{str(ex)}")
          def get_data(self,ev):
          f=self.SupplierTable.focus()
        content=(self.SupplierTable.item(f))
        row=content['values']
        self.var_supp_invoice.set(row[0]),
        self.var_sup_name.set(row[1]),
        self.var_sup_contact.set(row[2]),
        self.txt_desc.delete('1.0',END),
          self.txt_desc.insert(END,row[3]),
#-------------UPDATING DATA
          def update_data(self):
        conn=sqlite3.connect(database=r'ims.db')
          cur=conn.cursor()
          try:
          if self.var_supp_invoice.get()=="" or self.var_sup_name.get()=="":
              messagebox.showerror("Error","All fields are required",parent=self.root)
          else:
              cur.execute("select * from supplier where
invoice_id=?",(self.var_supp_invoice.get(),))
              row=cur.fetchone()
```

```
                if row == None:
                messagebox.showerror("Error","Invalid Invoice ID")
                else:
                cur.execute("UPDATE supplier set name=?,contact=?,desc=? where
invoice_id=?",(
                        self.var_sup_name.get(),
                        self.var_sup_contact.get(),
                        self.txt_desc.get('1.0',END),
                        self.var_supp_invoice.get()
                ))
                conn.commit()
                messagebox.showinfo("Success","Supplier data updated
successfully",parent=self.root)
                self.show()
        except Exception as ex:
        messagebox.showerror("Error",f"Error due to :{str(ex)}")
#------------Deleting the data
        def delete_data(self):
       conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
        if self.var_supp_invoice.get()=="" or self.var_sup_name.get()=="":
            messagebox.showerror("Error","All fields are required",parent=self.root)
        else:
            cur.execute("select * from supplier where
invoice_id=?",(self.var_supp_invoice.get(),))
```

```
            row=cur.fetchone()
                if row == None:
                messagebox.showerror("Error","Invalid Invoice ID")
                else:
                op=messagebox.askyesno("Confirm","Do you Want to
delete?",parent=self.root)
                if op==True:
                  cur.execute("delete from supplier where
invoice_id=?",(self.var_supp_invoice.get(),))
                  conn.commit()
                  messagebox.showinfo("Success","Supplier ID Deleted
Successfully",parent=self.root)
                  self.clear_data()
        except Exception as ex:
         messagebox.showerror("Error",f"Error due to :{str(ex)}")
#------------clear
        def clear_data(self):
      self.var_supp_invoice.set(""),
        self.var_sup_name.set(""),
      self.var_sup_contact.set(""),
      self.txt_desc.delete('1.0',END),
      self.var_search_txt.set("")
      self.var_searchby.set("Select")
        self.show()

#-------searching
        def search_data(self):
```

```
        conn=sqlite3.connect(database=r'ims.db')
            cur=conn.cursor()
            try:
            #if self.var_searchby.get()=="Select":
                #messagebox.showerror("Error","Select search by
option",parent=self.root)
            if self.var_search_txt.get()=="":
                messagebox.showerror("Error","Search input required",parent=self.root)
            else:
                cur.execute("SELECT * FROM supplier where invoice_id=?
",(self.var_search_txt.get(),))
                rows=cur.fetchone()
                    if rows != None:
                   self.SupplierTable.delete(*self.SupplierTable.get_children())
                    for rows in rows:
                      self.SupplierTable.insert('',END,values=rows)
                    else:
                        messagebox.showerror("Error","No Record found",parent=self.root)
            except Exception as ex:
             messagebox.showerror("Error",f"Error due to :{str(ex)}")
if __name__=="__main__":
            root=Tk()
            obj=SupplierClass(root)
    root.mainloop()
```

Categories.py

```
from tkinter import*

from tkinter import ttk,messagebox

from PIL import Image,ImageTk

import sqlite3

class CategoryClass:

        def __init__(self,root):

        self.root=root

      self.root.geometry("1300x650+205+150")

      self.root.config(bg="white")

      self.root.resizable(False, False)

      self.root.title("Inventory Management System | Developed by Our Group")

      self.root.focus_force()

        #--------variables

      self.var_cat_id=StringVar()

      self.var_cat_name=StringVar()

        #========Title



 lbl_title=Label(self.root,text="Manage Products Categories",font=("times new
roman",25,"bold"),fg="white",bg="#196666",justify=CENTER).place(x=10,y=10,width
=1280,height=50)

        #----label

      lbl_cat_name=Label(self.root,text="Enter Category Name",font=("times new
roman",22),bg="white").place(x=50,y=100)
```

```
        lbl_name_entry=Entry(self.root,textvariable=self.var_cat_name,font=("times
new roman",18),bg="lightyellow").place(x=50,y=170,width=300)

#--------bttn

        btn_add=Button(self.root,text="ADD",command=self.add_data,font=("times new
roman",15),fg="white",bg="#4caf50",cursor="hand2").place(x=360,y=170,width=150,
height=30)


btn_delete=Button(self.root,text="DELETE",command=self.delete_data,font=("times
new
roman",15),fg="white",bg="#ff0000",cursor="hand2").place(x=520,y=170,width=150,h
eight=30)

#Adding tree view

        cat_frame=Frame(self.root,bd=3,relief=RIDGE)

        cat_frame.place(x=700,y=90,height=530,width=590)

        scroly=Scrollbar(cat_frame,orient=VERTICAL)

        scrolx=Scrollbar(cat_frame,orient=HORIZONTAL)


self.CategoryTable=ttk.Treeview(cat_frame,columns=("cat_id","name"),yscrollcomma
nd=scroly.set,xscrollcommand=scrolx.set)



        scrolx.pack(side=BOTTOM,fill=X)

        scroly.pack(side=RIGHT,fill=Y)

        scrolx.config(command=self.CategoryTable.xview)

        scroly.config(command=self.CategoryTable.yview)

        self.CategoryTable.heading("cat_id",text="Category_ID")

        self.CategoryTable.heading("name",text="Name")



        self.CategoryTable["show"]="headings"

        self.CategoryTable.column("cat_id",width=90)
```

```
        self.CategoryTable.column("name",width=100)


        self.CategoryTable.pack(fill=BOTH,expand=1)


        self.CategoryTable.bind("<ButtonRelease-1>",self.get_data)
          self.show()


          #-----images
        self.im1=Image.open("E:\Self Study\Major Project\images\cat2.jpg")
        self.im1=self.im1.resize((670,410),Image.ANTIALIAS)
        self.im1=ImageTk.PhotoImage(self.im1)
        self.im1_lbl=Label(self.root,image=self.im1)
        self.im1_lbl.place(x=20,y=230)
          #-------functions
          def add_data(self):
        conn=sqlite3.connect(database=r'ims.db')
          cur=conn.cursor()
          try:
          if  self.var_cat_name.get()=="":
              messagebox.showerror("Error","All fields are required",parent=self.root)
          else:
                  cur.execute("select * from categories where
name=?",(self.var_cat_name.get(),))
              row=cur.fetchone()
                  if row != None:
                 messagebox.showerror("Error","This Category is already assigned")
```

```
            else:
                cur.execute("INSERT INTO categories(name) VALUES(?)",(
                    self.var_cat_name.get(),
                 ))
                conn.commit()
                messagebox.showinfo("Success","Categories added
successfully",parent=self.root)
                self.show()
        except Exception as ex:
         messagebox.showerror("Error",f"Error due to :{str(ex)}")
        def show(self):
      conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
         cur.execute("SELECT * FROM categories")
         rows=cur.fetchall()
         self.CategoryTable.delete(*self.CategoryTable.get_children())
        for rows in rows:
            self.CategoryTable.insert('',END,values=rows)
        except Exception as ex:
         messagebox.showerror("Error",f"Error due to :{str(ex)}")
        def get_data(self,ev):
      f=self.CategoryTable.focus()
      content=(self.CategoryTable.item(f))
        row=content['values']
      self.var_cat_id.set(row[0]),
```

```
        self.var_cat_name.set(row[1]),

          def delete_data(self):

        conn=sqlite3.connect(database=r'ims.db')

          cur=conn.cursor()

          try:

          if self.var_cat_id.get()=="":

                    messagebox.showerror("Error","Please Select or Enter Category
name",parent=self.root)

          else:

               cur.execute("select * from categories where
cat_id=?",(self.var_cat_id.get(),))

               row=cur.fetchone()

                    if row == None:

                   messagebox.showerror("Error","Invalid Category ID")

                    else:

                  op=messagebox.askyesno("Confirm","Do you Want to
delete?",parent=self.root)

                    if op==True:

                    cur.execute("delete from categories where
cat_id=?",(self.var_cat_id.get(),))

                      conn.commit()

                      messagebox.showinfo("Success","Category Deleted
Successfully",parent=self.root)

                      self.show()

                      self.var_cat_id.set("")

                      self.var_cat_name.set("")

          except Exception as ex:

           messagebox.showerror("Error",f"Error due to :{str(ex)}")
```

```
if __name__=="__main__":

        root=Tk()
        obj=CategoryClass(root)
        root.mainloop()
```

Products.py

```
from tkinter import *
from tkinter import ttk,messagebox
from PIL import Image,ImageTk
import sqlite3
class ProductClass:
        def __init__(self,root):
        self.root=root
        self.root.geometry("1300x650+205+150")
    self.root.config(bg="white")
    self.root.resizable(False, False)
    self.root.title("Inventory Management System | Developed by Our Group")
    self.root.focus_force()
#-------variables
        self.var_prod_id=StringVar()
    self.var_cat=StringVar()
    self.var_supplier=StringVar()
    self.var_name=StringVar()
    self.var_price=StringVar()
    self.var_quantity=StringVar()
    self.var_status=StringVar()
        self.var_searchby=StringVar()
    self.var_search_txt=StringVar()
        self.cat_list=[]
        self.supp_list=[]
```

```
        self.fetch_cat_sup()

#-----------Frame

        product_frame=Frame(self.root,bd=1.5,relief=RIDGE,bg="white")

        product_frame.place(x=10,y=10,width=475,height=638)

#---title

        title=Label(product_frame,text="Manage Product
Details",fg="white",bg="#0f4d7d",font=("times new roman",19)).pack(side=TOP,fill=X)

#----------column 1

        lbl_category=Label(product_frame,text="Category",bg="white",font=("times new
roman",19)).place(x=25,y=60)

        lbl_supplier=Label(product_frame,text="Supplier",bg="white",font=("times new
roman",19)).place(x=25,y=110)

        lbl_name=Label(product_frame,text="Name",bg="white",font=("times new
roman",19)).place(x=25,y=160)

        lbl_price=Label(product_frame,text="Price",bg="white",font=("times new
roman",19)).place(x=25,y=210)

        lbl_quantity=Label(product_frame,text="Quantity",bg="white",font=("times new
roman",19)).place(x=25,y=260)

          lbl_status=Label(product_frame,text="Status",bg="white",font=("times new
roman",19)).place(x=25,y=310)



#----column 2


cmb_cat=ttk.Combobox(product_frame,values=self.cat_list,textvariable=self.var_cat,
state="readonly",justify="center")

          cmb_cat.place(x=170,y=69,width=200)

          cmb_cat.current(0)


cmb_supplier=ttk.Combobox(product_frame,values=self.supp_list,textvariable=self.v
ar_supplier,state="readonly",justify="center")
```

```
        cmb_supplier.place(x=170,y=119,width=200)

           cmb_supplier.current(0)

        txt_name=Entry(product_frame,font=("times new
roman",15),bg="lightyellow",textvariable=self.var_name).place(x=170,y=169,width=2
00)

        txt_price=Entry(product_frame,font=("times new
roman",15),bg="lightyellow",textvariable=self.var_price).place(x=170,y=219,width=20
0)

        txt_quantity=Entry(product_frame,font=("times new
roman",15),bg="lightyellow",textvariable=self.var_quantity).place(x=170,y=269,width
=200)



cmb_status=ttk.Combobox(product_frame,values=("Active","Inactive"),textvariable=s
elf.var_status,state="readonly",justify="center")

        cmb_status.place(x=170,y=319,width=200)

        cmb_status.current(0)


self.add_bttn=Button(product_frame,text="Add",command=self.add_data,font=("time
s new
roman",15),bg="#2196f3",fg="black",cursor="hand2").place(x=4,y=400,width=100,hei
ght=35)


self.update_bttn=Button(product_frame,text="Update",command=self.update_data,fo
nt=("times new
roman",15),bg="#4caf50",fg="black",cursor="hand2").place(x=124,y=400,width=100,
height=35)


self.delete_bttn=Button(product_frame,text="Delete",command=self.delete_data,font
=("times new
roman",15),bg="#f44336",fg="black",cursor="hand2").place(x=244,y=400,width=100,
height=35)


self.clear_bttn=Button(product_frame,text="Clear",command=self.clear_data,font=("ti
mes new
```

```
roman",15),bg="#607d8b",fg="black",cursor="hand2").place(x=364,y=400,width=100
,height=35)

        #--------searchbar

      search_frame=LabelFrame(self.root,text="Search
Product",bg="White",font=("times new roman",12,"bold"))

      search_frame.place(x=550,y=10,width=700,height=80)

        #====options


self.search=ttk.Combobox(search_frame,values=("Select","Category","Supplier","Na
me"),textvariable=self.var_searchby,state="readonly",justify="center")

      self.search.place(x=10,y=10,width=170)

      self.search.current(0)

        #text field

      self.txt_search=Entry(search_frame,font=("times new
roman",15),bg="lightyellow",textvariable=self.var_search_txt).place(x=250,y=8)

        #button in the search field


self.search_bttn=Button(search_frame,text="Search",command=self.search_data,fon
t=("times new
roman",15),bg="lightgreen",fg="black",cursor="hand2").place(x=490,y=6,width=150,h
eight=30)

        #---making triview

        #Product details

      prod_frame=Frame(self.root,bd=3,relief=RIDGE)

      prod_frame.place(x=500,y=110,width=800,height=543)

      scroly=Scrollbar(prod_frame,orient=VERTICAL)

      scrolx=Scrollbar(prod_frame,orient=HORIZONTAL)


self.ProductTable=ttk.Treeview(prod_frame,columns=("prod_id","Category","Supplier
","name","price","qty","status"),yscrollcommand=scroly.set,xscrollcommand=scrolx.se
t)
```

```
scrolx.pack(side=BOTTOM,fill=X)
scroly.pack(side=RIGHT,fill=Y)
scrolx.config(command=self.ProductTable.xview)
scroly.config(command=self.ProductTable.yview)
self.ProductTable.heading("prod_id",text="Prod_ID")
self.ProductTable.heading("Category",text="Category")
  self.ProductTable.heading("Supplier",text="Supplier")
self.ProductTable.heading("name",text="Name")
self.ProductTable.heading("price",text="Price")
self.ProductTable.heading("qty",text="Quantity")
self.ProductTable.heading("status",text="Status")
self.ProductTable["show"]="headings"
self.ProductTable.column("prod_id",width=90)
self.ProductTable.column("Category",width=100)
self.ProductTable.column("Supplier",width=100)
  self.ProductTable.column("name",width=100)
self.ProductTable.column("price",width=100)
self.ProductTable.column("qty",width=100)
self.ProductTable.column("status",width=100)

self.ProductTable.pack(fill=BOTH,expand=1)



  self.show()
self.ProductTable.bind("<ButtonRelease-1>",self.get_data)
```

```
#-------------------------------------------------------------------------------------------------------
-
        def fetch_cat_sup(self):
        conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
         cur.execute("select name from categories")
         cat=cur.fetchall()
         self.cat_list.append("Empty")
        if len(cat)>0:
                del self.cat_list[:]
            self.cat_list.append("Select")
                for i in cat:
                self.cat_list.append(i[0])
         cur.execute("select name from supplier")
         supp=cur.fetchall()
         self.supp_list.append("Empty")
        if len(supp)>0:
                del self.supp_list[:]
            self.supp_list.append("Select")
                for i in supp:
                self.supp_list.append(i[0])
        except Exception as ex:
         messagebox.showerror("Error",f"Error due to :{str(ex)}")
```

```
    def add_data(self):
        conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
        if self.var_cat.get()=="Select" or self.var_cat.get()=="Empty" or self.var_supplier.get()=="Select" or self.var_supplier.get()=="Empty" or self.var_name.get()=="":
            messagebox.showerror("Error","All fields are required",parent=self.root)
        else:
            cur.execute("select * from product where name=?",(self.var_name.get(),))
            row=cur.fetchone()
                if row != None:
                messagebox.showerror("Error","This Product is already assigned try different product")
                 else:
                cur.execute("INSERT INTO product(Category,Supplier,name,price,qty,status) VALUES(?,?,?,?,?,?)",(
                            self.var_cat.get(),
                            self.var_supplier.get(),
                            self.var_name.get(),
                            self.var_price.get(),
                            self.var_quantity.get(),
                            self.var_status.get(),
                 ))
                conn.commit()
                  messagebox.showinfo("Success","Product added successfully",parent=self.root)
                self.show()
```

```
        except Exception as ex:
         messagebox.showerror("Error",f"Error due to :{str(ex)}")
        def show(self):
        conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
         cur.execute("SELECT * FROM product")
         rows=cur.fetchall()
         self.ProductTable.delete(*self.ProductTable.get_children())
        for rows in rows:
            self.ProductTable.insert('',END,values=rows)
        except Exception as ex:
         messagebox.showerror("Error",f"Error due to :{str(ex)}")
        def get_data(self,ev):
      f=self.ProductTable.focus()
      content=(self.ProductTable.item(f))
      row=content['values']
      self.var_prod_id.set(row[0]),
      self.var_cat.set(row[1]),
      self.var_supplier.set(row[2]),
      self.var_name.set(row[3]),
      self.var_price.set(row[4]),
      self.var_quantity.set(row[5]),
      self.var_status.set(row[6]),

#------------UPDATING DATA
```

```
    def update_data(self):
        conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
        if self.var_prod_id.get()=="":
                messagebox.showerror("Error","Please select product from
list",parent=self.root)
        else:
            cur.execute("select * from product where
prod_id=?",(self.var_prod_id.get(),))
            row=cur.fetchone()
                if row == None:
                messagebox.showerror("Error","Invalid Product")
                else:
                cur.execute("UPDATE product set
Category=?,Supplier=?,name=?,price=?,qty=?,status=? where prod_id=?",(
                    self.var_cat.get(),
                    self.var_supplier.get(),
                    self.var_name.get(),
                    self.var_price.get(),
                    self.var_quantity.get(),
                    self.var_status.get(),
                      self.var_prod_id.get()
                ))
                conn.commit()
                messagebox.showinfo("Success","Product data updated
successfully",parent=self.root)
```

```
                self.show()
            except Exception as ex:
             messagebox.showerror("Error",f"Error due to :{str(ex)}")
#------------Deleting the data
            def delete_data(self):
          conn=sqlite3.connect(database=r'ims.db')
            cur=conn.cursor()
            try:
            if self.var_prod_id.get()=="":
                messagebox.showerror("Error","Select product from list",parent=self.root)
            else:
                cur.execute("select * from product where
prod_id=?",(self.var_prod_id.get(),))
                row=cur.fetchone()
                    if row == None:
                   messagebox.showerror("Error","Invalid Product",parent=self.root)
                    else:
                   op=messagebox.askyesno("Confirm","Do you Want to
delete?",parent=self.root)
                    if op==True:
                          cur.execute("delete from product where
prod_id=?",(self.var_prod_id.get(),))
                      conn.commit()
                      messagebox.showinfo("Success","Product Deleted
Successfully",parent=self.root)
                       self.clear_data()
            except Exception as ex:
```

```
            messagebox.showerror("Error",f"Error due to :{str(ex)}")
#-------------clear
            def clear_data(self):
        self.var_prod_id.set("")
        self.var_cat.set("Select"),
            self.var_supplier.set("Select"),
        self.var_name.set(""),
        self.var_price.set(""),
        self.var_quantity.set(""),
        self.var_status.set("Active"),



        self.var_search_txt.set("")
        self.var_searchby.set("Select")


            self.show()
#-------searching
            def search_data(self):
        conn=sqlite3.connect(database=r'ims.db')
            cur=conn.cursor()
            try:
            if self.var_searchby.get()=="Select":
                messagebox.showerror("Error","Select search by option",parent=self.root)
            elif self.var_search_txt.get()=="":
                messagebox.showerror("Error","Search input required",parent=self.root)
            else:
```

```
            cur.execute("SELECT * FROM product where " +self.var_searchby.get()+"
LIKE '%"+self.var_search_txt.get()+"%'")
            rows=cur.fetchall()
                if len(rows)!=0:
                self.ProductTable.delete(*self.ProductTable.get_children())
                for rows in rows:
                  self.ProductTable.insert('',END,values=rows)
                else:
                messagebox.showerror("Error","No Record found",parent=self.root)
        except Exception as ex:
         messagebox.showerror("Error",f"Error due to :{str(ex)}")


if __name__=="__main__":


        root=Tk()
        obj=ProductClass(root)
        root.mainloop()
```

Sales.py

```
from tkinter import *

from tkinter import ttk,messagebox

from PIL import Image,ImageTk

import sqlite3

import os

class SalesClass:

        def __init__(self,root):

        self.root=root

    self.root.geometry("1300x650+205+150")

    self.root.config(bg="white")

    self.root.resizable(False, False)

    self.root.title("Inventory Management System | Developed by Our Group")

        self.root.focus_force()

        #------Variables

        self.billing_list=[]

    self.var_invoice=StringVar()

        #--------title

    lbl_title=Label(self.root,text="View Customer Bills",font=("times new
roman",25,"bold"),fg="white",bg="#196666",justify=CENTER).place(x=10,y=10,width
=1280,height=50)

    lbl_invoice=Label(self.root,text="Invoice ID",font=("times new
roman",15),bg="white").place(x=50,y=100)

    txt_invoice=Entry(self.root,textvariable=self.var_invoice,font=("times new
roman",15),bg="lightyellow").place(x=160,y=100,width=180,height=28)

    btn_search=Button(self.root,text="Search",command=self.search,font=("times
new
```

```
roman",15,"bold"),bg="#2196f3",fg="white",cursor="hand2").place(x=360,y=100,widt
h=110,height=28)

        btn_clear=Button(self.root,text="Clear",command=self.clear,font=("times new
roman",15,"bold"),bg="lightgray",cursor="hand2").place(x=490,y=100,width=110,heig
ht=28)

          #---------sales list

        sales_frame=Frame(self.root,bd=3,relief=RIDGE)

        sales_frame.place(x=40,y=135,height=499,width=220)

        scrolly=Scrollbar(sales_frame,orient=VERTICAL)

        self.sales_list=Listbox(sales_frame,font=("times new
roman",15),bg="white",yscrollcommand=scrolly)

          scrolly.pack(side=RIGHT,fill=Y)

        scrolly.config(command=self.sales_list.yview)

        self.sales_list.pack(fill=BOTH,expand=1)

        self.sales_list.bind("<ButtonRelease-1>",self.get_data)

          #-----billing area

        bill_frame=Frame(self.root,bd=3,relief=RIDGE)

        bill_frame.place(x=270,y=135,height=499,width=500)




        lbl_title2=Label(bill_frame,text="Customer Bill Area",font=("times new
roman",20,"bold"),bg="orange",justify=CENTER).pack(side=TOP,fill=X)

          scrolly2=Scrollbar(bill_frame,orient=VERTICAL)

        self.bill_list=Listbox(bill_frame,bg="lightyellow",yscrollcommand=scrolly2)

        scrolly2.pack(side=RIGHT,fill=Y)

        scrolly2.config(command=self.bill_list.yview)

          self.bill_list.pack(fill=BOTH,expand=1)

          #---images
```

```
self.bill_img=Image.open("E:\Self Study\Major Project\images\ill.png")
self.bill_img=self.bill_img.resize((480,520),Image.ANTIALIAS)
  self.bill_img=ImageTk.PhotoImage(self.bill_img)
lbl_image=Label(self.root,image=self.bill_img,bd=0)
  lbl_image.place(x=800,y=120)
  self.show()
  #-----------------
  def show(self):
  del self.billing_list[:]
  self.sales_list.delete(0,END)
  for i in os.listdir('E:\Self Study\Major Project\Bill'):
  if i.split('.')[-1]=='txt':
      self.sales_list.insert(END,i)
          self.billing_list.append(i.split('.')[0])
  def get_data(self,ev):
  index_=self.sales_list.curselection()
  file_name=self.sales_list.get(index_)
  self.bill_list.delete('0',END)
fp=open(f'Bill/{file_name}','r')
  for i in fp:
   self.bill_list.insert(END,i)
  fp.close()
  def search(self):
  if self.var_invoice.get()=="":
   messagebox.showerror("Error","Invoice ID requires",parent=self.root)
  else:
```

```
        if self.var_invoice.get() in self.billing_list:
            fp=open(f'Bill/{self.var_invoice.get()}.txt','r')
            self.bill_list.delete(0,END)
                for i in fp:
                self.bill_list.insert(END,i)
                fp.close()
        else:
            messagebox.showerror("Error","Invalid invoice number",parent=self.root)
        def clear(self):
        self.show()
        self.bill_list.delete('0',END)
if __name__=="__main__":
            root=Tk()
        obj=SalesClass(root)
        root.mainloop()
```

Analysis.py

```
from tkinter import*

from tkinter import ttk,messagebox

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import sqlite3

from PIL import Image,ImageTk

from idlelib.tooltip import Hovertip

class AnalysisClass:

        def __init__(self,root):

        self.root=root

    self.root.geometry("1300x650+205+150")

    self.root.config(bg="white")

    self.root.resizable(False, False)

    self.root.title("Inventory Management System | Developed by Our Group")

    self.root.focus_force()

        #title

    #self.icon_title=PhotoImage(file="E:\Self Study\Major Project\images\cat.jpg")

    title=Label(self.root,text="Analysis Of Product
Data",justify=CENTER,font=("Famtasy",26,"bold"),fg="black",bg="lightblue",padx=20
).place(x=0,y=0,height=60,relwidth=1)

#setting images

        self.im1=Image.open("E:\Self Study\Major Project\images\pie.jpg")

    self.im1=self.im1.resize((300,270))

    self.im1=ImageTk.PhotoImage(self.im1)
```

```
        self.im1_lbl=Label(self.root,image=self.im1)
        self.im1_lbl.place(x=20,y=64)
          self.im2=Image.open("E:\Self Study\Major Project\images\ibar.jpg")
        self.im2=self.im2.resize((300,270))
        self.im2=ImageTk.PhotoImage(self.im2)
        self.im2_lbl=Label(self.root,image=self.im2)
        self.im2_lbl.place(x=20,y=350)
          self.im3=Image.open("E:\Self Study\Major Project\images\iarea_chart.png")
        self.im3=self.im3.resize((300,270))
        self.im3=ImageTk.PhotoImage(self.im3)
        self.im3_lbl=Label(self.root,image=self.im3)
          self.im3_lbl.place(x=460,y=164)
        self.im5=Image.open("E:\Self Study\Major Project\images\icount.png")
        self.im5=self.im5.resize((300,270))
        self.im5=ImageTk.PhotoImage(self.im5)
        self.im5_lbl=Label(self.root,image=self.im5)
        self.im5_lbl.place(x=900,y=64)
        self.im6=Image.open("E:\Self Study\Major Project\images\line.jpg")
        self.im6=self.im6.resize((300,270))
        self.im6=ImageTk.PhotoImage(self.im6)
        self.im6_lbl=Label(self.root,image=self.im6)
        self.im6_lbl.place(x=900,y=350)
        #bar_button=Button(text="Bar-Graph")
#---------Lebeling images
        self.im1_label=Button(self.root,text="Pie
Chart",command=self.plot_pie,font=("times new
```

```
roman",15,"bold"),bg="white",fg="#00759E",bd=0,activebackground="white",activefor
eground="#00759E",cursor="hand2")

        self.im1_label.place(x=25,y=300)

            myBtn =
Button(self.root,text='?',bg="white",fg="#00759E",bd=0,activebackground="white",act
iveforeground="#00759E")

        myBtn.place(x=299,y=70)

            myTip = Hovertip(myBtn,'A pie chart is a circular graphical representation of
data where the size of each slice corresponds to the proportion or percentage it
represents.')

        self.im2_label=Button(self.root,text="Bar
Chart",command=self.plot_bar,font=("times new
roman",15,"bold"),bg="white",fg="#00759E",bd=0,activebackground="white",activefor
eground="#00759E",cursor="hand2")

        self.im2_label.place(x=25,y=583)

            myBtn2 =
Button(self.root,text='?',bg="white",fg="#00759E",bd=0,activebackground="white",act
iveforeground="#00759E")

        myBtn2.place(x=299,y=355)

            myTip2 = Hovertip(myBtn2,'A bar chart is a graphical representation of data
using rectangular bars, where the length of each bar corresponds to the value it
represents.')

            self.im3_label=Button(self.root,text="Area
Chart",command=self.plot_area,font=("times new
roman",15,"bold"),bg="white",fg="#00759E",bd=0,activebackground="white",activefor
eground="#00759E",cursor="hand2")

        self.im3_label.place(x=470,y=168)

            myBtn3 =
Button(self.root,text='?',bg="white",fg="#00759E",bd=0,activebackground="white",act
iveforeground="#00759E")

        myBtn3.place(x=740,y=168)

            myTip3 = Hovertip(myBtn3,'An area chart is a graphical representation of data
that uses a series of filled areas to display the cumulative magnitude or value of
multiple data points over time or categories.')
```

```
        self.im4_label=Button(self.root,text="Count
Chart",command=self.plot_scatter,font=("times new
roman",15,"bold"),bg="white",fg="#00759E",bd=0,activebackground="white",activefor
eground="#00759E",cursor="hand2")

        self.im4_label.place(x=905,y=73)

          myBtn4 =
Button(self.root,text='?',bg="white",fg="#00759E",bd=0,activebackground="white",act
iveforeground="#00759E")

          myBtn4.place(x=1175,y=70)

          myTip4 = Hovertip(myBtn4,'A count chart is a graphical representation of data
that displays the frequency or count of different categories or data points using bars
or columns.')

        self.im5_label=Button(self.root,text="Line
Chart",command=self.plot_line,font=("times new
roman",15,"bold"),bg="white",fg="#00759E",bd=0,activebackground="white",activefor
eground="#00759E",cursor="hand2")

        self.im5_label.place(x=905,y=585)

          myBtn5 =
Button(self.root,text='?',bg="white",fg="#00759E",bd=0,activebackground="white",act
iveforeground="#00759E")

        myBtn5.place(x=1175,y=355)

          myTip5 = Hovertip(myBtn5,'A line chart is a graphical representation of data
that uses connected data points with straight lines to show the trend or change in
values over time or any other continuous variable.')

          def plot_bar(self):

        con=sqlite3.connect(database='ims.db')

          cur=con.cursor()

        cur.execute("select Category,Supplier,name,price,qty,status from product")

          result=cur.fetchall()

          Category=[]

          Supplier=[]

          name=[]
```

```
  price=[]
  qty=[]
  status=[]
  try:
  for row in result:
      Category.append(row[0])
      Supplier.append(row[1])
          name.append(row[2])
      price.append(int(row[3]))
      qty.append(row[4])
      status.append(row[5])
  #print(Category) print(Supplier)print(name)print(price)print(qty)print(status)
  sns.barplot(x=name,y=price)
   #plt.bar(name,price)
  #plt.ylim(0, 5)
   plt.xlabel("Name of product")
   plt.ylabel("Price of product")

  plt.show()
  plt.close()
  except Exception as ex:
   messagebox.showerror("Error",f"Error due to :{str(ex)}")
  def plot_pie(self):
con=sqlite3.connect(database='ims.db')
  cur=con.cursor()
cur.execute("select Category,Supplier,name,price,qty,status from product")
```

```
result=cur.fetchall()
Category=[]
Supplier=[]
name=[]
price=[]
qty=[]
status=[]
try:
for row in result:
    Category.append(row[0])
    Supplier.append(row[1])
    name.append(row[2])
    price.append(row[3])
    qty.append(row[4])
    status.append(row[5])
#print(Category) print(Supplier)print(name)print(price)print(qty)print(status)
plt.pie(qty,labels=name, autopct='%.1f%%')
 #plt.legend(title='Product Names',loc="upper left")

 plt.title("Quantity Pie Chart")

plt.show()
plt.close()
except Exception as ex:
 messagebox.showerror("Error",f"Error due to :{str(ex)}")
```

```
  def plot_area(self):
con=sqlite3.connect(database='ims.db')
  cur=con.cursor()
cur.execute("select Category,Supplier,name,price,qty,status from product")
result=cur.fetchall()
  Category=[]
  Supplier=[]
  name=[]
  price=[]
  qty=[]
  status=[]
  try:
  for row in result:
      Category.append(row[0])
      Supplier.append(row[1])
      name.append(row[2])
      price.append(row[3])
      qty.append(int(row[4]))
      status.append(row[5])
  #print(Category) print(Supplier)print(name)print(price)print(qty)print(status)
  plt.fill_between(name,qty)
  #plt.ylim(0,5)
   plt.title("Name Vs Price")
   plt.xlabel("Name of product")
   plt.ylabel("Quantity of product")
```

```
  plt.show()
  plt.close()
  except Exception as ex:
   messagebox.showerror("Error",f"Error due to :{str(ex)}")
  def plot_scatter(self):
con=sqlite3.connect(database='ims.db')
  cur=con.cursor()
cur.execute("select Category,Supplier,name,price,qty,status from product")
result=cur.fetchall()
  Category=[]
  Supplier=[]
  name=[]
  price=[]
  qty=[]
  status=[]
  try:
  for row in result:
      Category.append(row[0])
          Supplier.append(row[1])
      name.append(row[2])
      price.append(row[3])
      qty.append(row[4])
      status.append(row[5])
  #print(Category) print(Supplier)print(name)print(price)print(qty)print(status)
   sns.countplot(x=status)
   plt.title("Count of Active And Inactive Data")
```

```
    plt.show()
    plt.close()



    except Exception as ex:
     messagebox.showerror("Error",f"Error due to :{str(ex)}")
    def plot_line(self):
con=sqlite3.connect(database='ims.db')
    cur=con.cursor()
cur.execute("select Category,Supplier,name,price,qty,status from product")
result=cur.fetchall()
    Category=[]
    Supplier=[]
    name=[]
    price=[]
    qty=[]
    status=[]
    try:
    for row in result:
        Category.append(row[0])
        Supplier.append(row[1])
        name.append(row[2])
              price.append(row[3])
```

```
            qty.append(int(row[4]))
            status.append(row[5])
        #print(Category) print(Supplier)print(name)print(price)print(qty)print(status)
         plt.plot(Supplier,qty)
        #plt.ylim(0, 5)
         plt.xlabel("Name of product")
         plt.ylabel("Price of product")

        plt.show()
        plt.close()
        except Exception as ex:
         messagebox.showerror("Error",f"Error due to :{str(ex)}")

if __name__=="__main__":

        root=Tk()
        obj=AnalysisClass(root)
        root.mainloop()
```

Billing.py

```
from tkinter import*
from PIL import Image,ImageTk
from tkinter import ttk,messagebox
import sqlite3
import time
import os
import tempfile
class BillClass:
        def __init__(self,root):
        self.root=root
      self.root.geometry("1900x830+0+0")
      self.root.config(bg="white")
      #self.root.resizable(False, False)
      self.root.focus_force()
        self.root.title("Inventory Management System | Developed by Our Group")
        self.cart_list=[]
        self.chk_print=0

        #----TITLE
      self.icon_title=PhotoImage(file="E:\Self Study\Major Project\images\logo1.png")
        title=Label(self.root,text="Inventory Management
System",image=self.icon_title,compound=LEFT,font=("Famtasy",35,"bold"),fg="white
",bg="blue",anchor="w",padx=20).place(x=0,y=0,height=70,relwidth=1)
        #----Footer
```

```
        lbl_footer=Label(root,text="This is a dummy project made by Ashish , Kuntak , Shubhankar , Nikita",font=("times new roman",15),bg="grey",fg="black").pack(side=BOTTOM,fill=X)

          #-----Logout button

        btn_logout=Button(self.root,command=self.logout,text="Logout",font=("times new roman",15),cursor="hand2").place(x=1350,y=22,height=35)



          #----Making Date As well as time

        self.lbl_clock=Label(self.root,text="Welcome to our project inventory management system\t\tDate:DD-MM-YYYY\t\tTime: HH:MM:SS",font=("times new roman",15),bg="grey",fg="black")

        self.lbl_clock.place(x=0,y=70,height=30,relwidth=1)

#--------Column1 no.1 Frame----------------

          #product frame

          #----variables under product frame

        self.var_search=StringVar()

          ProductFrame1=Frame(self.root,bd=4,relief=RIDGE,bg="white")

        ProductFrame1.place(x=6,y=110,height=690,width=450)

        pTitle=Label(ProductFrame1,text="All Products",font=("times new roman",20,"bold"),bg="#262626",fg="white").pack(side=TOP,fill=X)

          ProductFrame2=Frame(ProductFrame1,bd=2,relief=RIDGE,bg="white")

        ProductFrame2.place(x=2,y=40,height=90,width=438)

          #product search frame

        lbl_search=Label(ProductFrame2,text="Search Product | By Name",font=("times new roman",15,"bold"),bg="white",fg="green").place(x=2,y=5)

        lbl_search=Label(ProductFrame2,text="Product Name",font=("times new roman",15,"bold"),bg="White").place(x=5,y=45)

        txt_search=Entry(ProductFrame2,textvariable=self.var_search,font=("times new roman",15),bg="lightyellow").place(x=130,y=47,height=22)
```

```
btn_search=Button(ProductFrame2,command=self.search_data,text="Search",bg="#2196f3",fg="white",cursor="hand2").place(x=350,y=45,width=80,height=25)

        btn_showAll=Button(ProductFrame2,command=self.show,text="Show All",bg="#083531",fg="white",cursor="hand2").place(x=350,y=15,width=80,height=25)

          #---making triview

        product_frame3=Frame(ProductFrame1,bd=3,relief=RIDGE)

        product_frame3.place(x=2,y=133,height=520,width=438)

           scroly=Scrollbar(product_frame3,orient=VERTICAL)

        scrolx=Scrollbar(product_frame3,orient=HORIZONTAL)


self.productTable=ttk.Treeview(product_frame3,columns=("prod_id","name","price","qty","status"),yscrollcommand=scroly.set,xscrollcommand=scrolx.set)



        scrolx.pack(side=BOTTOM,fill=X)

        scroly.pack(side=RIGHT,fill=Y)

        scrolx.config(command=self.productTable.xview)

        scroly.config(command=self.productTable.yview)

        self.productTable.heading("prod_id",text="PID")

        self.productTable.heading("name",text="Name")

        self.productTable.heading("price",text="Price")

        self.productTable.heading("qty",text="Qty")

        self.productTable.heading("status",text="Status")


          self.productTable["show"]="headings"

        self.productTable.column("prod_id",width=70)

        self.productTable.column("name",width=100)
```

```
        self.productTable.column("price",width=75)

        self.productTable.column("qty",width=70)

          self.productTable.column("status",width=100)



        self.productTable.pack(fill=BOTH,expand=1)

        lbl_note=Label(ProductFrame1,text="Note: 'Enter 0 Quantity to remove product
from the caart'",font=("times new
roman",14),fg="red",bg="white").pack(side=BOTTOM,fill=X)



        self.productTable.bind("<ButtonRelease-1>",self.get_data)

          #self.show()

#--------------Column 2 ------------------

          #--------Customer Frame

          #-----variables

        self.var_cname=StringVar()

          self.var_contact=StringVar()

        CustomerFrame=Frame(self.root,bd=4,relief=RIDGE,bg="white")

        CustomerFrame.place(x=460,y=110,height=70,width=570)

        cTitle=Label(CustomerFrame,text="Customer Details",font=("times new
roman",15),bg="lightgray").pack(side=TOP,fill=X)

        lbl_name=Label(CustomerFrame,text="Name",font=("times new
roman",15),bg="white").place(x=5,y=35)

        txt_name=Entry(CustomerFrame,textvariable=self.var_cname,font=("times new
roman",13),bg="lightyellow").place(x=80,y=35,width=170)

        lbl_contact=Label(CustomerFrame,text="Contact",font=("times new
roman",15),bg="white").place(x=265,y=35)

        txt_contact=Entry(CustomerFrame,textvariable=self.var_contact,font=("times
new roman",13),bg="lightyellow").place(x=345,y=35,width=170)

          #----cal cart frame
```

```
        Cal_Cart_frame=Frame(self.root,bd=4,relief=RIDGE,bg="white")

        Cal_Cart_frame.place(x=460,y=185,height=500,width=570)

          #---calculator frame

#making variables under calculator frame

            self.var_cal_input=StringVar()

        Cal_frame=Frame(Cal_Cart_frame,bd=9,relief=RIDGE,bg="white")

        Cal_frame.place(x=5,y=5,height=485,width=268)


text_cal_input=Entry(Cal_frame,textvariable=self.var_cal_input,font=("arial",15,"bold"
),width=21,border=8,relief=GROOVE,justify=RIGHT).grid(row=0,columnspan=4)


btn_7=Button(Cal_frame,text="7",font=("arial",15,"bold"),command=lambda:self.get_i
nput(7),bd=5,width=4,pady=30,cursor="hand2").grid(row=1,column=0)


btn_8=Button(Cal_frame,text="8",font=("arial",15,"bold"),command=lambda:self.get_i
nput(8),bd=5,width=4,pady=30,cursor="hand2").grid(row=1,column=1)


btn_9=Button(Cal_frame,text="9",font=("arial",15,"bold"),command=lambda:self.get_i
nput(9),bd=5,width=4,pady=30,cursor="hand2").grid(row=1,column=2)


btn_sum=Button(Cal_frame,text="+",font=("arial",15,"bold"),command=lambda:self.g
et_input('+'),bd=5,width=4,pady=30,cursor="hand2").grid(row=1,column=3)


btn_4=Button(Cal_frame,text="4",font=("arial",15,"bold"),command=lambda:self.get_i
nput(4),bd=5,width=4,pady=30,cursor="hand2").grid(row=2,column=0)


btn_5=Button(Cal_frame,text="5",font=("arial",15,"bold"),command=lambda:self.get_i
nput(5),bd=5,width=4,pady=30,cursor="hand2").grid(row=2,column=1)


btn_6=Button(Cal_frame,text="6",font=("arial",15,"bold"),command=lambda:self.get_i
nput(6),bd=5,width=4,pady=30,cursor="hand2").grid(row=2,column=2)
```

```
btn_sub=Button(Cal_frame,text="-",font=("arial",15,"bold"),command=lambda:self.get
_input("-"),bd=5,width=4,pady=30,cursor="hand2").grid(row=2,column=3)


btn_1=Button(Cal_frame,text="1",font=("arial",15,"bold"),command=lambda:self.get_i
nput(1),bd=5,width=4,pady=30,cursor="hand2").grid(row=3,column=0)


btn_2=Button(Cal_frame,text="2",font=("arial",15,"bold"),command=lambda:self.get_i
nput(2),bd=5,width=4,pady=30,cursor="hand2").grid(row=3,column=1)


btn_3=Button(Cal_frame,text="3",font=("arial",15,"bold"),command=lambda:self.get_i
nput(3),bd=5,width=4,pady=30,cursor="hand2").grid(row=3,column=2)


btn_mul=Button(Cal_frame,text="*",font=("arial",15,"bold"),command=lambda:self.ge
t_input('*'),bd=5,width=4,pady=30,cursor="hand2").grid(row=3,column=3)


btn_c=Button(Cal_frame,text="C",font=("arial",15,"bold"),command=self.clear_cal,bd
=5,width=4,pady=30,cursor="hand2").grid(row=4,column=0)


btn_0=Button(Cal_frame,text="0",font=("arial",15,"bold"),command=lambda:self.get_i
nput(0),bd=5,width=4,pady=30,cursor="hand2").grid(row=4,column=1)


btn_eq=Button(Cal_frame,text="=",font=("arial",15,"bold"),command=self.perform_ca
l,bd=5,width=4,pady=30,cursor="hand2").grid(row=4,column=2)


btn_div=Button(Cal_frame,text="/",font=("arial",15,"bold"),command=lambda:self.get
_input('/'),bd=5,width=4,pady=30,cursor="hand2").grid(row=4,column=3)

          #---making triview

          #-------cart frame

        Cart_frame=Frame(Cal_Cart_frame,bd=3,relief=RIDGE)

        Cart_frame.place(x=280,y=5,height=485,width=282)

        self.cartTitle=Label(Cart_frame,text="Cart \t Toatal Products: [0]",font=("times
new roman",15),bg="lightgray")

        self.cartTitle.pack(side=TOP,fill=X)
```

```
        scroly=Scrollbar(Cart_frame,orient=VERTICAL)
        scrolx=Scrollbar(Cart_frame,orient=HORIZONTAL)

self.CartTable=ttk.Treeview(Cart_frame,columns=("pid","name","price","qty"),yscrollc
ommand=scroly.set,xscrollcommand=scrolx.set)

        scrolx.pack(side=BOTTOM,fill=X)
        scroly.pack(side=RIGHT,fill=Y)
        scrolx.config(command=self.CartTable.xview)
        scroly.config(command=self.CartTable.yview)
        self.CartTable.heading("pid",text="PID")
        self.CartTable.heading("name",text="Name")
        self.CartTable.heading("price",text="Price")
        self.CartTable.heading("qty",text="Qty")

          self.CartTable["show"]="headings"
        self.CartTable.column("pid",width=38)
        self.CartTable.column("name",width=100)
        self.CartTable.column("price",width=80)
        self.CartTable.column("qty",width=38)

        self.CartTable.pack(fill=BOTH,expand=1)

        self.CartTable.bind("<ButtonRelease-1>",self.get_data_cart)

          #----Add cart buttons
```

```
#making variable

        self.var_prod_id=StringVar()

        self.var_pname=StringVar()

        self.var_price=StringVar()

        self.var_qty=StringVar()

        self.var_stock=StringVar()

        Cart_button_frame=Frame(self.root,bd=4,relief=RIDGE,bg="white")

        Cart_button_frame.place(x=460,y=684,height=115,width=570)

          #widgets addng

          lbl_p_name=Label(Cart_button_frame,text="Product Name",font=("times new
roman",15),bg="white").place(x=5,y=5)

        txt_p_name=Entry(Cart_button_frame,textvariable=self.var_pname,font=("times
new
roman",15),bg="lightyellow",state='readonly').place(x=5,y=35,width=190,height=22)

        lbl_p_price=Label(Cart_button_frame,text="Price Per Quantity",font=("times
new roman",15),bg="white").place(x=210,y=5)

        txt_p_price=Entry(Cart_button_frame,textvariable=self.var_price,font=("times
new
roman",15),bg="lightyellow",state='readonly').place(x=210,y=35,width=150,height=22
)

        lbl_p_qty=Label(Cart_button_frame,text="Quantity",font=("times new
roman",15),bg="white").place(x=390,y=5)

        txt_p_qty=Entry(Cart_button_frame,textvariable=self.var_qty,font=("times new
roman",15),bg="lightyellow").place(x=390,y=35,width=150,height=22)

        self.lbl_instock=Label(Cart_button_frame,text="In Stock",font=("times new
roman",15),bg="white")

        self.lbl_instock.place(x=5,y=70)

          #Adding buttons


btn_clear_cart=Button(Cart_button_frame,command=self.clear_cart,text="Clear",font
=("times new
```

```
roman",15,"bold"),bg="lightgray",cursor="hand2").place(x=200,y=68,width=150,heigh
t=35)


btn_add_update_cart=Button(Cart_button_frame,command=self.add_update_cart,te
xt="Add | Update cart",font=("times new
roman",15,"bold"),bg="orange",cursor="hand2").place(x=365,y=68,width=180,height
=35)

#---------COLUMN 3

        #------BILLING AREA

      bill_frame=Frame(self.root,relief=RIDGE,bd=4,bg="white")

        bill_frame.place(x=1035,y=110,width=490,height=540)

      BTitle=Label(bill_frame,text="Customer Bill Area",font=("times new
roman",20,"bold"),bg="#942821",fg="white").pack(side=TOP,fill=X)

      scroly=Scrollbar(bill_frame,orient=VERTICAL)

        scroly.pack(side=RIGHT,fill=Y)

      self.txt_bill_area=Text(bill_frame,yscrollcommand=scroly.set)

      self.txt_bill_area.pack(fill=BOTH,expand=1)

      scroly.config(command=self.txt_bill_area.yview)

#-----------billing buttons

      bill_menu_frame=Frame(self.root,relief=RIDGE,bd=4,bg="white")

      bill_menu_frame.place(x=1035,y=645,width=490,height=154)

      self.lbl_amount=Label(bill_menu_frame,text="Bill Amount \n [0]",font=("times
new roman",15,"bold"),fg="white",bg="#1f2e2e")

        self.lbl_amount.place(x=2,y=5,width=160,height=70)

      self.lbl_discount=Label(bill_menu_frame,text="Discount \n [5%]",font=("times
new roman",15,"bold"),fg="white",bg="#00e6b8")

      self.lbl_discount.place(x=163,y=5,width=160,height=70)

        self.lbl_net_pay=Label(bill_menu_frame,text="Net Pay \n [0]",font=("times
new roman",15,"bold"),fg="white",bg="#008080")
```

```
        self.lbl_net_pay.place(x=324,y=5,width=155,height=70)


btn_print=Button(bill_menu_frame,command=self.print_bill,text="Print",font=("times
new roman",15,"bold"),fg="white",bg="lightgreen",cursor="hand2")

        btn_print.place(x=2,y=83,width=140,height=62)

        btn_clear_all=Button(bill_menu_frame,command=self.clear_all,text="Clear
All",font=("times new roman",15,"bold"),fg="white",bg="gray",cursor="hand2")

        btn_clear_all.place(x=145,y=83,width=140,height=62)


btn_generate_bill=Button(bill_menu_frame,command=self.generate_bill,text="Gener
ate/Save Bill",font=("times new
roman",15,"bold"),fg="white",bg="#009688",cursor="hand2")

        btn_generate_bill.place(x=289,y=83,width=190,height=62)

#Showing product data in column 1

          self.show()

          self.add_date_time()

          #self.bill_top()

          def get_input(self,num):

        xnum=self.var_cal_input.get()+str(num)

        self.var_cal_input.set(xnum)

          def clear_cal(self):

        self.var_cal_input.set('')


          def perform_cal(self):

        result=self.var_cal_input.get()

        self.var_cal_input.set(eval(result))

          def show(self):

          conn=sqlite3.connect(database=r'ims.db')
```

```
        cur=conn.cursor()
        try:
         cur.execute("SELECT prod_id,name,price,qty,status FROM product where
status='Active'")
         rows=cur.fetchall()
         self.productTable.delete(*self.productTable.get_children())
        for rows in rows:
            self.productTable.insert('',END,values=rows)
        except Exception as ex:
         messagebox.showerror("Error",f"Error due to :{str(ex)}")
#-------searching
        def search_data(self):
       conn=sqlite3.connect(database=r'ims.db')
        cur=conn.cursor()
        try:
        if self.var_search.get()=="":
            messagebox.showerror("Error","Select search by Name",parent=self.root)
        else:
            cur.execute("SELECT prod_id,name,price,qty,status FROM product where
name LIKE '%"+self.var_search.get()+"%' AND status='Active'")
            rows=cur.fetchall()
                if len(rows)!=0:
               self.productTable.delete(*self.productTable.get_children())
                for rows in rows:
                  self.productTable.insert('',END,values=rows)
                else:
```

```
            messagebox.showerror("Error","No Record found",parent=self.root)
    except Exception as ex:
     messagebox.showerror("Error",f"Error due to :{str(ex)}")
    def get_data(self,ev):
f=self.productTable.focus()
content=(self.productTable.item(f))
row=content['values']
self.var_prod_id.set(row[0])
self.var_pname.set(row[1])
self.var_price.set(row[2])
self.lbl_instock.config(text=f"In Stock [{str(row[3])}]")
self.var_stock.set(row[3])
self.var_qty.set('1')
    def get_data_cart(self,ev):
    f=self.CartTable.focus()
content=(self.CartTable.item(f))
row=content['values']
self.var_prod_id.set(row[0])
self.var_pname.set(row[1])
self.var_price.set(row[2])
self.var_qty.set(row[3])
    self.lbl_instock.config(text=f"In Stock [{str(row[4])}]")
self.var_stock.set(row[4])
    def add_update_cart(self):
    if self.var_prod_id.get()=="":
     messagebox.showerror('Error',"Please select product from the list")
```

```
        elif self.var_qty.get()=="":
         messagebox.showerror('Error',"Please Enter Product
Quantity",parent=self.root)
        elif int(self.var_qty.get())>int(self.var_stock.get()):
         messagebox.showerror("Error","Invalid Quantity",parent=self.root)
        else:
         price_cal=float(int(self.var_qty.get())*float(self.var_price.get()))

cart_data=[self.var_prod_id.get(),self.var_pname.get(),self.var_price.get(),self.var_qt
y.get(),self.var_stock.get()]

        #update cart
            present=0
        index_=0
        for row in self.cart_list:
                if self.var_prod_id.get()==row[0]:
               present='yes'
                break
                index_+=1
        if present=='yes':
            op=messagebox.askyesno('Confirm',"Product is already present Do you
want to update | remove  from the cart list",parent=self.root)
                if op==True:
                if self.var_qty.get()=="0":
                  self.cart_list.pop(index_)
                else:
                  #self.cart_list[index_][2]=price_cal
```

```
                self.cart_list[index_][3]=self.var_qty.get()
    else:
        self.cart_list.append(cart_data)
    self.show_cart()
     self.bill_update()
    def bill_update(self):
    self.bill_amt=0
    self.net_amt=0
    self.discount=0
    for row in self.cart_list:
     self.bill_amt=self.bill_amt+(float(row[2])*int(row[3]))
    #discount
self.discount=(self.bill_amt*5)/100
self.net_amt=self.bill_amt-((self.bill_amt*5)/100)   #discount
self.lbl_amount.config(text=f"Bill Amount(Rs.) \n {str(self.bill_amt)}")
self.lbl_net_pay.config(text=f"Net Amount(Rs.) \n {str(self.net_amt)}")
self.cartTitle.config(text=f"Cart \t Toatal Products: [{str(len(self.cart_list))}]")
    def show_cart(self):
    try:
     self.CartTable.delete(*self.CartTable.get_children())
    for rows in self.cart_list:
        self.CartTable.insert('',END,values=rows)
    except Exception as ex:
     messagebox.showerror("Error",f"Error due to :{str(ex)}")
    def generate_bill(self):
    if self.var_cname.get()=="" or self.var_contact.get()=="":
```

```
            messagebox.showerror("Error",f"Customer Details
Required",parent=self.root)
            elif len(self.cart_list)==0:
             messagebox.showerror("Error","Please Add Product for billing")
            else:
            #=====bill top
              self.bill_top()
            #=====bill middle
            self.middle()
            #=====bill bottom
             self.bill_bottom()
            #pass
             fp=open(f'bill/{str(self.invoice)}.txt','w')
             fp.write(self.txt_bill_area.get('1.0',END))
            fp.close()
             messagebox.showinfo('Saved',"Bill has been generated/Saved in
backend",parent=self.root)
            self.chk_print=1
            def bill_top(self):
        self.invoice=int(time.strftime("%H%M%S"))+int(time.strftime("%d%m%y"))
        bill_top_temp=f"""
\t\t\tABS - PVT LTD
\t Phone No. 8328897003 , Bihar-855106
{str("="*57)}
   Customer Name:{self.var_cname.get()}
   ph no. {self.var_contact.get()}
```

```
   Bill No.
{str(self.invoice)}\t\t\tDate:{str(time.strftime("%d/%m/%y"))}\tTime:{str(time.strftime("
%H:%M"))}

{str("="*57)}

 Product Name\t\t\tQTY\t\tPrice

{str("="*57)}

          """

        self.txt_bill_area.delete('1.0',END)

        self.txt_bill_area.insert('1.0',bill_top_temp)

           def bill_bottom(self):

           bill_bottom_temp=f"""

{str("="*57)}

Bill Amount\t\t\t\tRs.{self.bill_amt}

Discount\t\t\t\tRs.{self.discount}

Net PAy\t\t\t\tRs.{self.net_amt}\n

{str("="*57)}\n

          """

        self.txt_bill_area.insert(END,bill_bottom_temp)

           def middle(self):

           con=sqlite3.connect(database=r'ims.db')

           cur=con.cursor()

           try:

           for row in self.cart_list:


                prod_id=row[0]

                     name=row[1]
```

```
        qty=int(row[4])-int(row[3])
            if int(row[3]) ==int(row[4]):
           status='Inactive'
            if int(row[3]) !=int(row[4]):
           status='Active'
        price=float(row[2])*int(row[3])
        price=str(price)
            self.txt_bill_area.insert(END,"\n"+name+"\t\t\t"+row[3]+"\t\tRs."+price)
            #=====Update quantity in product table
        cur.execute("update product set qty=?,status=? where prod_id=?",(
            qty,
            status,
                prod_id
            ))
            con.commit()
   con.close()
   self.show()
   except Exception as ex:
    messagebox.showerror("Error",f"Error due to :{str(ex)}")
   def clear_cart(self):
    self.var_prod_id.set("")
self.var_pname.set("")
self.var_price.set("")
self.lbl_instock.config(text=f"In Stock")
self.var_stock.set("")
   self.var_qty.set('')
```

```
    def clear_all(self):
    del self.cart_list[:]
    self.var_cname.set("")
  self.var_contact.set("")
  self.txt_bill_area.delete('1.0',END)
    self.clear_cart()
    self.show()
    self.show_cart()
  self.var_search.set('')
  self.cartTitle.config(text=f"Cart \t Toatal Products: [0]")
  self.lbl_net_pay.config(text=f"Net Amount(Rs.) \n[0]")
  self.lbl_amount.config(text=f"Bill Amount(Rs.) \n[0]")
        def add_date_time(self):
  Time=time.strftime("%I:%M:%S")
  date=time.strftime("%d:%m:%Y")
  self.lbl_clock.config(text=f"Welcome to our project inventory management
system\t\tDate: {str(date)}\t\tTime: {str(Time)}")
    self.lbl_clock.after(100,self.add_date_time)
       def print_bill(self):
    if self.chk_print==1:
         messagebox.showinfo("Print","Please wait while printing",parent=self.root)
     new_file=tempfile.mktemp('.txt')
     open(new_file,'w').write(self.txt_bill_area.get('1.0',END))
     os.startfile(new_file,'print')
    else:
     messagebox.showerror("print","Please Generate bill to get the receipt")
```

```
        def logout(self):
        self.root.destroy()
        os.system("python login.py")

if __name__=="__main__":

        root=Tk()
        obj=BillClass(root)
        root.mainloop()
```

Email.py

email_='ashishsinghtkg@gmail.com'

pass_='**********'

Createdb.py

```
import sqlite3

def create_db():

    conn=sqlite3.connect(database=r'ims.db')

        cur=conn.cursor()

        cur.execute("CREATE TABLE IF NOT EXISTS employee(emp_id integer
PRIMARY KEY AUTOINCREMENT,name text,email text,gender text,contact text,dob
text,doj text,password text,user_type ext,address text,salary text)")

        conn.commit()



        cur.execute("CREATE TABLE IF NOT EXISTS supplier(invoice_id integer
PRIMARY KEY AUTOINCREMENT,name text,contact text,desc text)")

        conn.commit()



        cur.execute("CREATE TABLE IF NOT EXISTS categories(cat_id integer
PRIMARY KEY AUTOINCREMENT,name text)")

        conn.commit()


#prod_id","Category","Supplier","name","price","qty","status

        cur.execute("CREATE TABLE IF NOT EXISTS product(prod_id integer
PRIMARY KEY AUTOINCREMENT,Category text,Supplier text,name text,price
text,qty text,status text)")

        conn.commit()




create_db()
```

Conclusion

In conclusion, the development and implementation of the Inventory Management System (Python GUI) have proven to be a valuable solution for efficient and streamlined inventory management processes. This project report has provided a comprehensive overview of the system's design, functionality, and its potential impact on organizational operations.

The Inventory Management System offers several key advantages over traditional manual methods, including real-time tracking of inventory, automated data entry, accurate forecasting, and streamlined reporting. By leveraging the power of Python and its graphical user interface, the system provides a user-friendly experience, allowing employees to easily manage inventory levels, monitor stock movements, and generate insightful reports.

Throughout the project, careful attention was given to the system's architecture, usability, and scalability. The use of object-oriented programming principles in Python enabled modular and extensible code, facilitating future enhancements and maintenance. The integration of a graphical user interface ensured a visually appealing and intuitive interface for users, improving their overall experience and efficiency.

The successful implementation of the Inventory Management System has the potential to significantly impact various aspects of an organization's operations. By ensuring accurate inventory records and efficient stock management, the system helps minimize stockouts, reduce carrying costs, and optimize order fulfillment. Moreover, it enables data-driven decision-making, empowers forecasting and demand planning, and facilitates strategic inventory control.

However, it is important to acknowledge that any technological solution is subject to limitations and challenges. The Inventory Management System may require regular updates and maintenance to address emerging needs and potential issues. Additionally, adequate training and support should be provided to users to maximize the system's benefits and mitigate any resistance to change.

In conclusion, the Inventory Management System (Python GUI) has demonstrated its potential to revolutionize inventory management practices, improving operational efficiency and decision-making capabilities. It is a valuable tool for organizations seeking to streamline their inventory processes, reduce costs, and enhance customer satisfaction.

Scope For Future Development

The Inventory Management System (Python GUI) presents several opportunities for future development and enhancement. Here are some potential areas to explore:

1. **Integration with other systems:** The system can be integrated with other business systems such as sales, purchasing, and accounting software to establish a seamless flow of data across different departments. This integration would enable more accurate and real-time inventory updates, as well as improve overall business processes.

2. **Advanced reporting and analytics:** Enhancing the reporting capabilities of the system can provide deeper insights into inventory trends, stock turnover, and demand forecasting. By implementing advanced analytics techniques and visualizations, users can make data-driven decisions and optimize inventory strategies.

3. **Barcode and RFID integration:** Incorporating barcode or RFID technology can streamline the process of updating inventory information, reducing manual data entry errors and improving efficiency. This integration would involve scanning items upon receipt or sale, automatically updating the inventory system.

4. **Supplier management**: Expanding the system to include supplier management features would enable tracking supplier information, lead times, and pricing. This enhancement would facilitate better supplier relationship management and support decision-making regarding supplier selection and negotiation.

5. **Mobile application:** Developing a mobile application version of the Inventory Management System would provide flexibility and convenience for users to access and update inventory data on the go. It could include features such as barcode scanning, stock alerts, and remote access to reports.

6. **Demand forecasting and optimization:** Implementing advanced algorithms and machine learning techniques can enhance the system's ability to forecast demand accurately. This functionality would assist in optimizing inventory levels, minimizing stockouts, and improving overall inventory control.

7. **Multi-location support:** Adapting the system to handle multiple warehouse or store locations would be beneficial for businesses with distributed inventory. This expansion would involve managing stock transfers, tracking inventory across locations, and optimizing stock allocation based on demand patterns.

8. **User customization and permissions:** Allowing users to customize the system's interface and defining different access levels and permissions for users can enhance usability and security. This customization would cater to specific user preferences and enable better control over system functionality.

These future development areas present exciting opportunities to further enhance the Inventory Management System (Python GUI) and make it even more robust, efficient, and user-friendly. By leveraging emerging technologies and addressing specific business needs, the system can continue to drive improvements in inventory management processes and contribute to organizational growth.

References

1. **SQLite Official Website:** https://www.sqlite.org/
   The official website of SQLite provides comprehensive documentation, tutorials, and resources for understanding SQLite's features, SQL syntax, and usage.

2. **Python SQLite3 Documentation:** https://docs.python.org/3/library/sqlite3.html
   - The official Python documentation offers detailed information on using the SQLite3 module in Python. It covers topics such as connecting to a database, executing SQL statements, querying data, and managing transactions.

3. **"Python and SQLite:** Data persistence and GUI projects" by Fletcher Heisler
   - This article provides a practical guide to using SQLite with Python, including setting up a database, creating tables, inserting data, and performing queries. It also explores how to integrate SQLite with GUI projects.

4. **"Python SQLite Tutorial" by Tutorialspoint:**
   https://www.tutorialspoint.com/sqlite/sqlite_python.htm
   Tutorialspoint offers a step-by-step tutorial on working with SQLite in Python. It covers essential topics such as creating a database, executing queries, and managing transactions using the SQLite3 module.

5. **"Using SQLite with Python" by Real Python:**
   https://realpython.com/python-sqlite-sqlalchemy/
   Real Python provides an in-depth tutorial on working with SQLite in Python. It covers not only the basics of SQLite but also explores advanced topics such as using SQLAlchemy, an Object-Relational Mapping (ORM) library, with SQLite.

6. **SQLite3 in Python - An Overview:**
   https://stackabuse.com/sqlite3-tutorial-with-python/

7. - This tutorial offers an overview of SQLite3 in Python, explaining how to create a database, execute SQL commands, and retrieve data. It also covers error handling and provides code examples.

8. **"SQLite with Python - Full Course" by freeCodeCamp.org:**
   This comprehensive video tutorial on YouTube provides a step-by-step guide to using SQLite with Python. It covers topics such as creating a database, executing SQL queries, managing data, and integrating SQLite with Python projects. [Link: https://www.youtube.com/watch?v=byHcYRpMgI4]

9. **"Python SQLite3 Tutorial" by Corey Schafer:**
   This video tutorial on YouTube walks you through the basics of SQLite3 in Python. It covers connecting to a database, executing SQL commands, fetching data, and working with transactions.
   [Link: https://www.youtube.com/watch?v=pd-0G0MigUA]

10. **"Python SQLite Tutorial" by ProgrammingKnowledge:**
    This video tutorial on YouTube provides an introduction to SQLite and demonstrates its usage in Python. It covers topics such as creating tables, inserting data, querying data, and updating records.
    [Link: https://www.youtube.com/watch?v=byHcYRpMgI4]

11. **"Webcode - Python & Flask Tutorials" by Webcode:**
    - The Webcode YouTube channel provides a variety of tutorials on Python and Flask, which is a popular web framework in Python. The tutorials cover topics such as web development, database integration, and creating web applications using Flask.
    [Link: https://www.youtube.com/@webcode4805]

    The Webcode channel can be a valuable resource if you are looking to integrate your Inventory Management System with a web-based interface or explore web development aspects related to Python and Flask.

12. **"Python SQLite3 Example Programs"** by SQLite Tutorial:
    This online tutorial provides a collection of Python SQLite3 example programs. It includes code snippets and explanations for various SQLite operations, such as creating tables, inserting data, querying data, and performing joins.
    [Link: https://www.sqlitetutorial.net/sqlite-python/]

13. **"SQLite Python Tutorial" by SQLite Tutorial:**
    This online tutorial offers a comprehensive overview of SQLite in Python. It covers topics such as creating a database, executing SQL statements, querying data, and handling exceptions.
    [Link: https://www.sqlitetutorial.net/sqlite-python/]

14. **"Python SQLite Database" by Guru99:** This tutorial provides a step-by-step guide to using SQLite3 in Python. It covers connecting to a database, creating tables, inserting data, retrieving data, and updating records.
    [Link: https://www.guru99.com/python-sqlite3.html]


Thank
you