**Inventory Mnagement  
System**

**List of Experiments**

|  |  |  |
| --- | --- | --- |
| 1.Problem Analysis | 1.1 Overview of the Project | |
| 1.2 Identification of Project Scope | |
| 1.3 Objectives | |
| 1.4 Infrastructure | |
| 2.Software Requirement Analysis and Planning | 2.1 Description of individual phase/module | 2.1.1  User characteristics |
| 2.1.2 General constraints |
| 2.1.3 Assumption: Dependency |
| 2.1.4  Functional requirements:-Input, Output, Description |
| 3.Data Modeling | Architecture, Usecase | |
| Activity | |
| DFD | |
| Class | |
| 4.Development | Database Structure | |
| Coding | |
| 5.Software Testing | Test Plan | |
| 6.Project Presentation with Demo |  | |
| 7.Report Presentation and Submission |  | |

**Problem Analysis:**

The manual inventory management system is time-consuming, error-prone, and inefficient. Difficulty in tracking inventory levels, leading to stockouts or excess inventory. Lack of real-time data makes decision-making challenging. Inaccurate records may result in financial discrepancies.

**1.1Overview of the Project:**

Implementing a computerized Inventory Management System (IMS). Utilizing technology to streamline processes, reduce errors, and enhance efficiency. Integration with other business systems for seamless data flow.

Why Computerized?

- Faster and more accurate data entry.

- Real-time updates on inventory levels and stock movements.

- Improved decision-making through data analytics.

- Enhanced security and data integrity.

**Scope, Objective, Purpose:**

**Scope:** Comprehensive management of inventory from procurement to sales.

**Objective:** To automate and optimize inventory processes for increased efficiency.

**Purpose:** Improve overall business operations by ensuring accurate, timely, and accessible inventory data.

**1.2 Identification of Project Scope:**

Inclusion of features such as product tracking, order processing, and reporting. Integration with other business modules like sales, procurement, and finance. Compatibility with existing hardware and software infrastructure.

**Tasks Involved:**

**Requirement Gathering:** Identify user needs, system functionalities, and integration requirements.

**System Design:** Create a blueprint for the IMS, outlining database structure, user interfaces, and system architecture.

**Development:** Code and implement the IMS based on the design specifications.

**Testing:** Conduct thorough testing to ensure system functionality, data accuracy, and security.

**Training:** Train users on the new system to ensure smooth adoption.

**Implementation:** Roll out the system in stages or, depending on the organization's strategy.

**Maintenance and Support:** Provide ongoing support, updates, and maintenance to address issues and keep the system up-to-date.

**// upto task -1**

**1.3 Objectives:F**

**1.Efficient Inventory Tracking:** The primary objective of an inventory management system is to accurately track the movement of goods throughout the supply chain. This includes monitoring stock levels, tracking sales, and managing replenishment.

**2.Optimized Inventory Levels:** The system should help in maintaining optimal inventory levels, ensuring that there is neither excess stock nor shortages. This involves forecasting demand, setting reorder points, and managing stock replenishment.

**3.Cost Reduction:** By optimizing inventory levels and streamlining processes, the system should help in reducing costs associated with excess inventory holding, stockouts, and obsolescence.

**4.Improved Customer Service:** An inventory management system should contribute to improved customer service by ensuring product availability, reducing order fulfillment times, and minimizing backorders.

**5.Data Analysis and Insights:** The system should provide actionable insights through data analysis, helping in identifying trends, forecasting demand, and making informed decisions.

**1.4 Infrastructure:**

**1.Software Platform:** The infrastructure typically begins with the selection of an appropriate software platform for inventory management. This could range from standalone inventory management software to integrated enterprise resource planning (ERP) systems.

**2.Database:** A robust database is essential for storing all inventory-related data, including product information, stock levels, sales history, supplier details, and transaction records.

**3.User Interface:** The system should have an intuitive user interface that allows users to easily input and access inventory data, generate reports, and perform various tasks such as placing orders and updating stock levels.

**4.Integration Capabilities:** It's important for the system to integrate seamlessly with other business systems such as accounting software, order management systems, and customer relationship management (CRM) software to ensure smooth data flow and process automation.

**5.Hardware:** Depending on the scale of operations, the infrastructure may require hardware components such as barcode scanners, RFID readers, mobile devices, and computers to facilitate inventory tracking and management tasks.

**6.Network Infrastructure:** A reliable network infrastructure is necessary to ensure connectivity between different system components, especially in multi-location or distributed warehouse environments.

**7.Security Measures:** Robust security measures, including access controls, data encryption, and regular backups, should be implemented to protect inventory data from unauthorized access, loss, or theft.

**8.Scalability:** The infrastructure should be designed to scale according to the growing needs of the business, accommodating increases in the volume of inventory, users, and transactions over time.

**2. SOFTWARE REQUIREMENT ANALYSIS AND PLANNING**

**2.1 Description of Individual Phase/Module:**

**2.1.1 User Characteristics:**

The user characteristics module outlines the various types of users who will interact with the inventory management system and their respective roles and responsibilities. This includes:

**1.Administrators:** Responsible for system configuration, user management, and overall system maintenance.

**2.Inventory Managers:** Tasked with overseeing inventory levels, stock movements, and replenishment strategies.

**3.Purchasing Managers:** In charge of procurement activities, supplier management, and purchase order processing.

**4.Sales Representatives:** Engaged in order entry, sales monitoring, and customer interaction.

**5.Warehouse Personnel:** Involved in receiving, picking, packing, and shipping of goods.

**6.Accounting Staff:** Responsible for financial transactions, invoicing, and reconciliation.

Each user group may have specific access rights and permissions tailored to their functional requirements within the system.

**2.1.2 General Constraints of Inventory Management System:**

The general constraints module identifies the limitations or boundaries that need to be considered during the design and development of the inventory management system. These constraints may include:

**1.Budgetary Constraints:** Limitations on available funds for software development, hardware procurement, and ongoing maintenance.

**2.Time Constraints:** Deadlines for system implementation, testing, and deployment.

**3.Resource Constraints:** Limitations on human resources, expertise, and availability for system development and support.

**4.Technological Constraints:** Compatibility with existing hardware, software platforms, and infrastructure.

**5.Regulatory Constraints:** Compliance with industry regulations, standards, and data privacy laws.

**6.Scalability Constraints:** The system should be scalable to accommodate future growth and increased transaction volumes.

**7.Performance Constraints:** The system should meet performance requirements in terms of response times, throughput, and reliability.

**8.Security Constraints:** Implementation of robust security measures to protect sensitive inventory data from unauthorized access, modification, or theft.

**9.Usability Constraints:** The system should be user-friendly, intuitive, and easy to navigate for users with varying levels of technical expertise.

Addressing these constraints ensures that the inventory management system meets the functional, operational, and regulatory requirements of the business while staying within budget and time constraints.

**2.1.3 Assumptions: Dependency**

The assumption module outlines the dependencies and assumptions made during the planning and development of the inventory management system. These dependencies could include:

**1.Integration Dependencies:** Assumptions about the availability and compatibility of third-party systems or APIs for integration with the inventory management system, such as accounting software, e-commerce platforms, or supplier databases.

**2.Data Dependencies:** Assumptions regarding the availability, accuracy, and consistency of data sources used by the inventory management system, including product catalogs, inventory levels, sales data, and supplier information.

**3.Infrastructure Dependencies:** Assumptions about the availability and reliability of underlying infrastructure components, including hardware, networks, and databases required for system operation.

**4.Resource Dependencies:** Assumptions about the availability of human resources, expertise, and support required for system development, implementation, and maintenance.

**5.Regulatory Dependencies:** Assumptions about compliance with industry regulations, standards, and data privacy laws, including GDPR, HIPAA, or PCI-DSS.

**6.Vendor Dependencies:** Assumptions about the reliability, support, and service levels provided by software vendors, hardware suppliers, or service providers involved in the implementation and operation of the inventory management system.

**2.1.4 Functional Requirements:**

**Input:**

**1.Product Information:** Ability to input and update product details such as name, description, SKU (Stock Keeping Unit), unit cost, and supplier information.

**2. Purchase Orders:** Capability to input purchase orders, including supplier details, quantities, prices, delivery dates, and payment terms.

**3.Sales Orders:** Ability to input sales orders, including customer details, product quantities, prices, shipping information, and payment terms.

**4.Inventory Transactions:** Capability to input inventory transactions such as receipts, shipments, adjustments, transfers, and returns.

**Output:**

**1.Inventory Reports**: Generation of inventory reports including stock levels, stock movements, aging analysis, and stock valuation.

**2.Order Reports:** Generation of order reports including purchase order status, sales order status, backorder reports, and order fulfillment metrics.

**3.Financial Reports:** Generation of financial reports including revenue, cost of goods sold (COGS), profit margins, and inventory turnover.

**4.Alerts and Notifications:** Provision of alerts and notifications for low stock levels, stockouts, pending orders, and critical inventory transactions.

**Description for Inventory Management System:**

The inventory management system is designed to facilitate the efficient tracking, management, and optimization of inventory levels across the supply chain. It provides functionalities for:

**1.Inventory Tracking:** Real-time tracking of inventory levels, stock movements, and transactions across multiple warehouses or locations.

**2.Order Management:** Processing of purchase orders, sales orders, and transfer orders, including order entry, order fulfillment, and order status tracking.

**3.Inventory Optimization:** Analysis of demand patterns, forecasting, and optimization of inventory levels to minimize stockouts, reduce excess inventory holding costs, and improve cash flow.

**4.Supplier Management:** Management of supplier information, purchase requisitions, supplier performance evaluation, and supplier relationship management.

**5.Reporting and Analysis:** Generation of various reports and analytics to monitor key inventory metrics, track performance, identify trends, and make data-driven decisions.

By fulfilling these functional requirements, the inventory management system enhances operational efficiency, improves inventory accuracy, and ultimately contributes to the profitability and competitiveness of the business.

**Data Modeling for Inventory Management System:**

**Architecture:**

The architecture of the inventory management system involves the arrangement of components and their interactions to facilitate the storage, processing, and retrieval of inventory-related data. A typical architecture for such a system may include the following layers:

**1.Presentation Layer:** This layer provides the user interface for interacting with the inventory management system. It includes web or desktop interfaces through which users can input data, view reports, and perform various inventory management tasks.

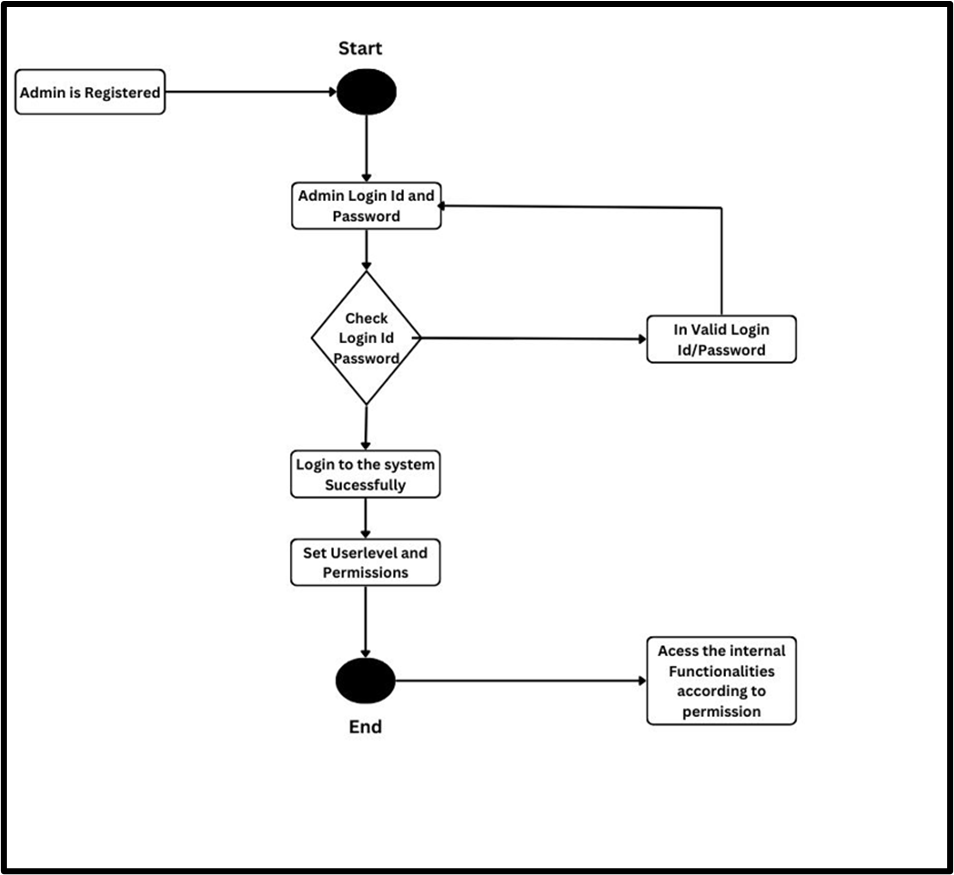
**2.Application Layer:** The application layer contains the business logic and processing functionalities of the system. It handles tasks such as inventory tracking, order management, reporting, and integration with external systems. This layer often comprises modules or services responsible for specific functions within the system.

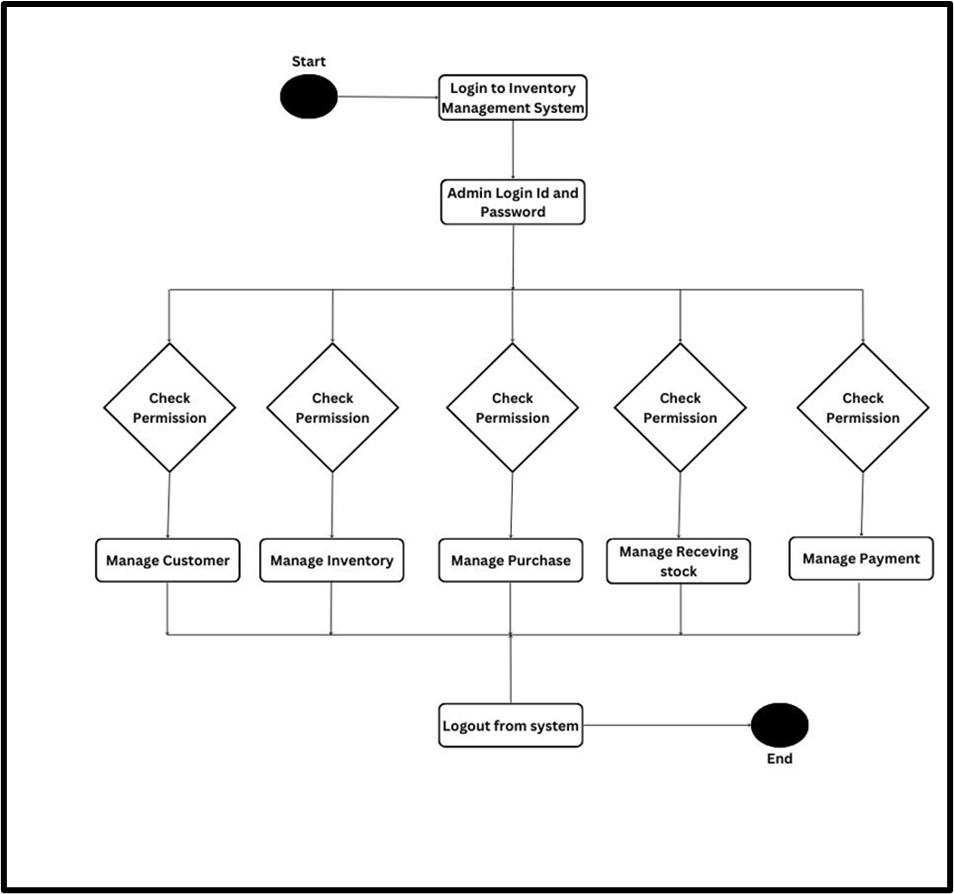
**3.Data Access Layer:** The data access layer is responsible for interacting with the underlying database or data storage system. It includes components for querying, inserting, updating, and deleting data from the database. This layer ensures efficient and secure access to inventory data while abstracting the complexities of the underlying data storage mechanism.

**4.Database Layer:** This layer consists of the database or data storage system where inventory-related data is stored. It includes tables, indexes, and other database objects optimized for storing and retrieving inventory data efficiently. Common database technologies used in inventory management systems include relational databases (e.g., MySQL, PostgreSQL) or NoSQL databases (e.g., MongoDB, Cassandra) depending on the specific requirements of the system.

***Use case Diagrams:***

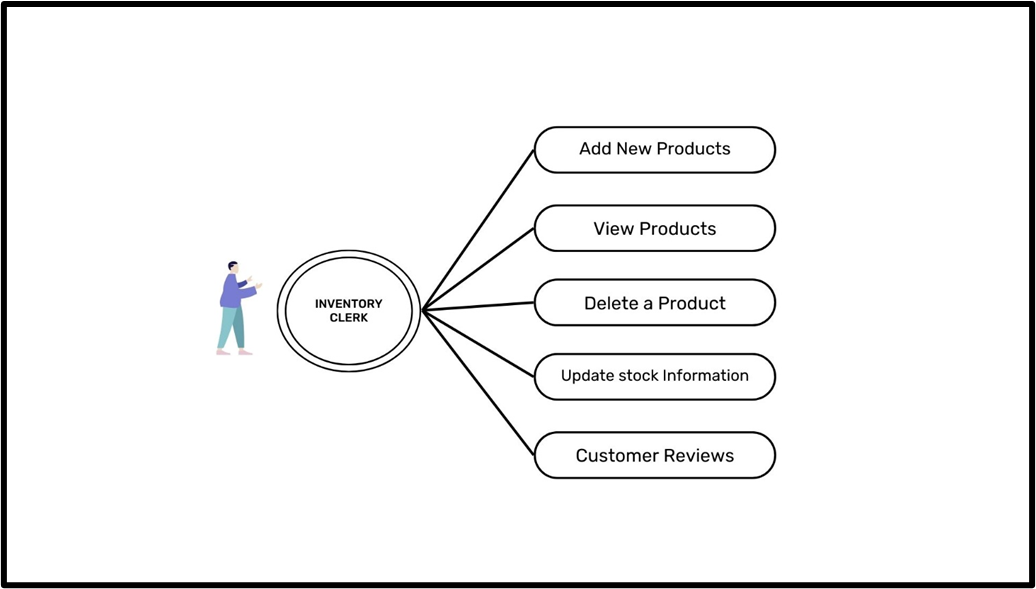
A Use Case Diagram is an important tool in system design because it visually represents **how users interact with a system**. It serves as a blueprint for understanding a system's functional needs from the perspective of its users, directing the development process.

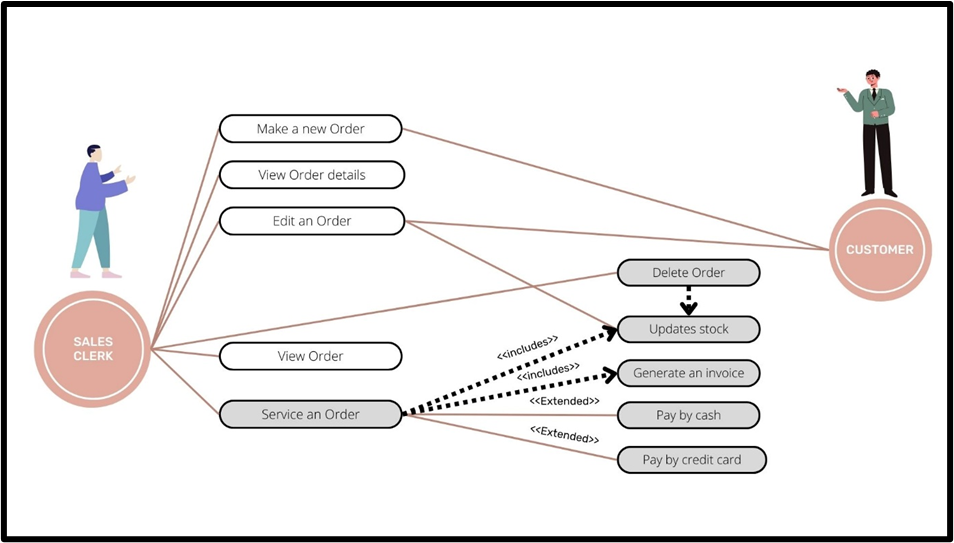




***Activity Diagrams:***

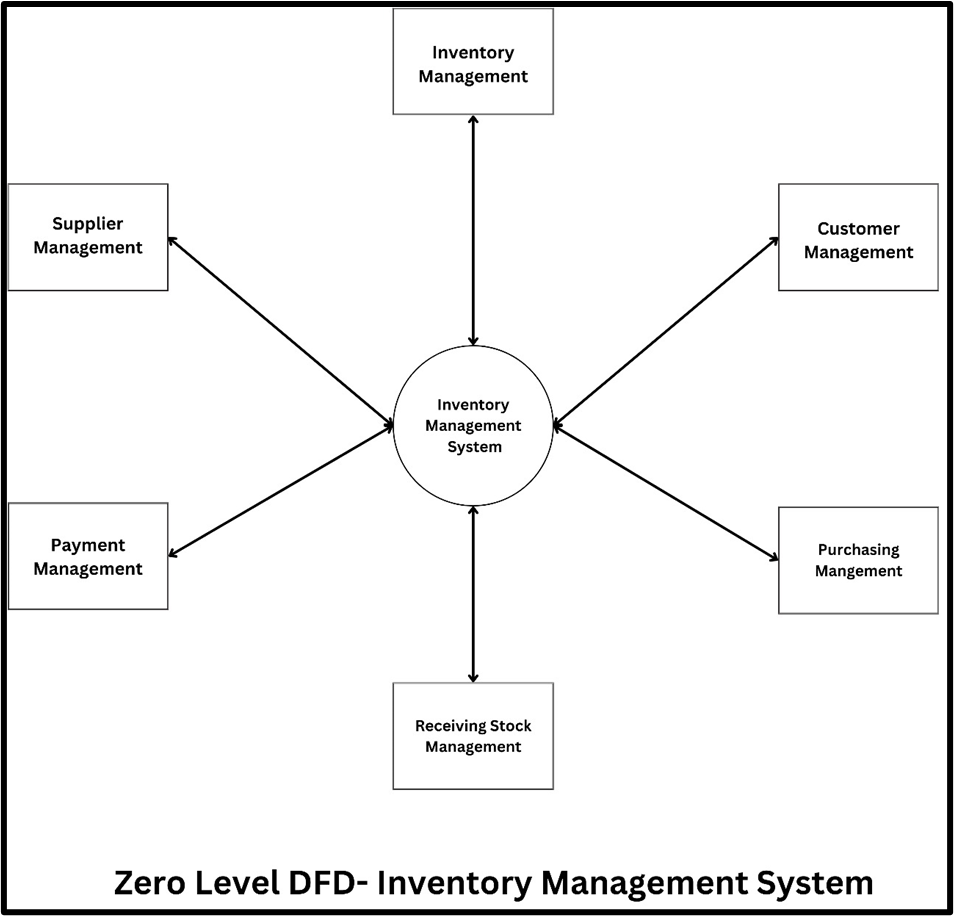
Activity diagrams showcase the process from start to end, highlighting the several decision paths that occur as events pass through the activity. These are used to illustrate the **flow of control in a system** and refer to the **steps involved in the execution of a use case**. We can depict both sequential processing and concurrent processing of activities using an activity diagram

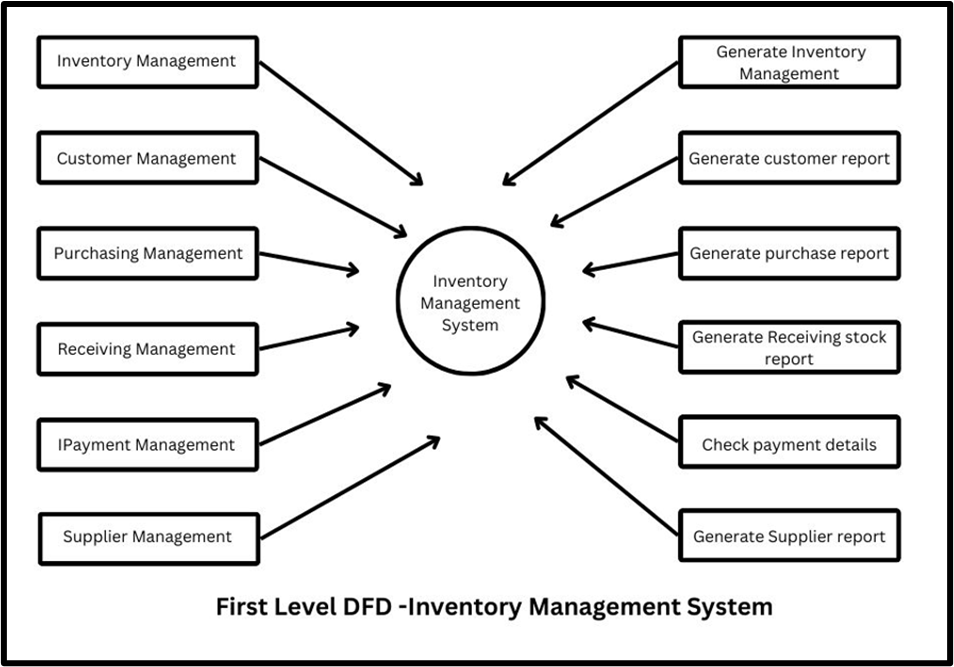




***Data Flow Diagrams:***

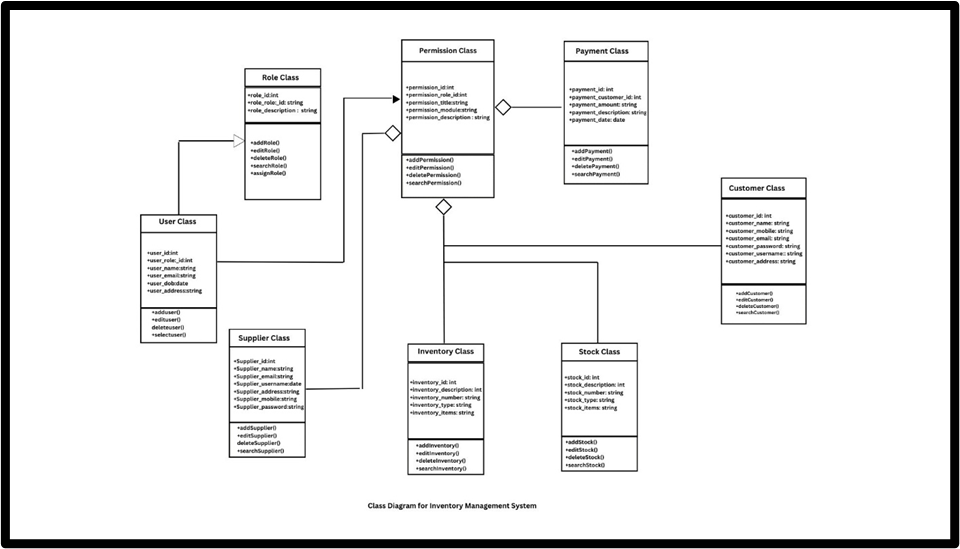
DFD represents **the flow of data inside a system** or process. It also provides information on each entity's inputs and outputs, as well as the overall process. These are very useful in understanding a system and can be **effectively used during problem analysis**.

******

******

***Class Diagrams:***

Class diagrams are a sort of UML (Unified Modelling Language) diagram that is used in software engineering to graphically **express the structure as well as the relationships** of classes inside a system.

******

**4. DEVELOPMENT:**

We developed a responsive website that uses Python for backend and frontend part.

**4.1 Database Structure:**

**SQLite** is a lightweight, self-contained, serverless, and open-source SQL database engine. Its database structure follows the principles of relational databases.

The structure of SQlite contains tables, views and Indexes. Primary key, foreign keys and constraints allow us in data retrieval and querying.

**Reason for choosing SQlite:** SQLite is known for its simplicity and minimalism, making it easy to use for small to medium-sized databases or embedded systems.

The database contains details regarding user details, their emergency contacts and their details. Whenever the emergency button is pressed, the get location method is used to get the location data of the user and is sent to emergency contacts via mail and text message.

**CODING:**

**Dashboard:**

**from tkinter import\***

**from PIL import Image, ImageTk *# pip install pillow***

**import tkinter**

**from employee import employeeClass**

**from product import productClass**

**from sales import salesClass**

**class IMS:**

**def \_\_init\_\_(self, root):**

***self*.root = root**

***self*.root.state('zoomed')**

***self*.root.title("Inventory management system")**

***# screen width and height***

**width = *self*.root.winfo\_screenwidth()**

**height = *self*.root.winfo\_screenheight()**

***# title designing***

**title = Label(**

***self*.root,**

**text="WELCOME USER",**

**font=("Times New Roman", 30, "bold"),**

**bg="black",**

**fg="white",**

**anchor="sw",**

**padx=25**

**).place(x=0, y=0, relwidth=1, height=70)**

***# Left menu***

**LeftMenu = Frame(**

***self*.root,**

**bd=2, relief=RIDGE,**

**bg="grey"**

**)**

**LeftMenu.place(x=0, y=70, width=200)**

**lbl\_menu = Label(**

**LeftMenu,**

**text="MENU",**

**font=("times new roman", 20, "bold"),**

***# fg="white",***

**bg="black",**

**fg="white"**

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

**btn\_employee = Button(LeftMenu,**

**text="Empolyee",**

**font=("times new roman", 20),**

**command=*self*.employee,**

**bd=3,**

**cursor="hand2",**

**bg="black",**

**fg="white"**

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

**btn\_product = Button(LeftMenu,**

**text="Products",**

**font=("times new roman", 20),**

**bg="black",**

**fg="white",**

**command=*self*.product,**

**bd=3,**

**cursor="hand2"**

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

**btn\_sales = Button(LeftMenu,**

**text="Sales",**

**font=("times new roman",**

**20),**

**bg="black",**

**fg="white",**

**command=*self*.sales,**

**bd=3,**

**cursor="hand2"**

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

**btn\_exit = Button(LeftMenu,**

**text="Exit",**

**font=("times new roman", 20),**

**bg="black",**

**fg="white",**

**command=quit,**

**bd=3,**

**cursor="hand2"**

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

***# Content image designing***

***self*.img1 = Image.open("images/Dashboard.png")**

***self*.img1 = *self*.img1.resize((width-150, height-50))**

***self*.img1 = ImageTk.PhotoImage(*self*.img1)**

***self*.lbl\_img = Label(*self*.root, image=*self*.img1)**

***self*.lbl\_img.place(x=200, y=70)**

***# defining function for employee***

**def employee(self):**

***self*.new\_win = Toplevel(*self*.root)**

***self*.new\_obj = employeeClass(*self*.new\_win)**

***# defining function for product***

**def product(self):**

***self*.new\_win = Toplevel(*self*.root)**

***self*.new\_obj = productClass(*self*.new\_win)**

***# defining function for sales***

**def sales(self):**

***self*.new\_win = Toplevel(*self*.root)**

***self*.new\_obj = salesClass(*self*.new\_win)**

***# # exit function***

**def quit():**

**root.mainloop()**

***# main modification***

**if \_\_name\_\_ == "\_\_main\_\_":**

**root = Tk()**

**obj = IMS(root)**

**root.mainloop()**

**Creating database:**

**import sqlite3**

**def create\_db():**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

***# ========employee table ==============***

**cur.execute("CREATE TABLE IF NOT EXISTS employee(eid INTEGER PRIMARY KEY AUTOINCREMENT,name text,email text,utype text,salary text)")**

**con.commit()**

***# =========product table =============***

**cur.execute("CREATE TABLE IF NOT EXISTS product(pid INTEGER PRIMARY KEY AUTOINCREMENT,name text,price text,qty text,status text)")**

**con.commit()**

**create\_db()**

**Employee**

**from tkinter import\***

**from tkinter import ttk,messagebox**

**import sqlite3**

**class employeeClass:**

**def \_\_init\_\_(self,root):**

***self*.root=root**

***self*.root.geometry("1100x500+220+130")**

***self*.root.title("Inventory Management System")**

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

***self*.root.focus\_force()**

***#=============================***

***# All Variables==========***

***self*.var\_searchby=StringVar()**

***self*.var\_searchtxt=StringVar()**

***self*.var\_emp\_id=StringVar()**

***self*.var\_name=StringVar()**

***self*.var\_email=StringVar()**

***self*.var\_utype=StringVar()**

***self*.var\_salary=StringVar()**

***#========searchFrame======***

**SearchFrame=LabelFrame(*self*.root,text="Search Employee",font=("goudy old style",12,"bold"),bd=2,relief=RIDGE,bg="white")**

**SearchFrame.place(x=250,y=20,width=600,height=70)**

***#=======options======***

**cmb\_search=ttk.Combobox(SearchFrame,textvariable=*self*.var\_searchby,values=("Select","Email","Name"),state='readonly',justify=CENTER,font=("time new roman",15))**

**cmb\_search.place(x=10,y=10,width=180)**

**cmb\_search.current(0)**

**txt\_search=Entry(SearchFrame,textvariable=*self*.var\_searchtxt,font=("goudy old style",15),bg="lightyellow").place(x=200,y=10)**

**btn\_search=Button(SearchFrame,text="Search",command=*self*.search,font=("goudy old style",15),bg="#4caf50",fg="white").place(x=410,y=10,width=150,height=30)**

***#====title========***

**title=Label(*self*.root,text="Emplyoee Details",font=("goudy old style",15),bg="#0f4d7d",fg="white").place(x=50,y=100,width=1000)**

***#======content=====***

**lbl\_empid=Label(*self*.root,text="Emp ID",font=("goudy old style",15),bg="white").place(x=50,y=150)**

**lbl\_name=Label(*self*.root,text="Name",font=("goudy old style",15),bg="white").place(x=350,y=150)**

**lbl\_email=Label(*self*.root,text="Email Id",font=("goudy old style",15),bg="white").place(x=750,y=150)**

**txt\_empid=Entry(*self*.root,textvariable=*self*.var\_emp\_id,font=("goudy old style",15),bg="white").place(x=150,y=150,width=180)**

**txt\_name=Entry(*self*.root,textvariable=*self*.var\_name,font=("goudy old style",15),bg="white").place(x=500,y=150,width=180)**

**txt\_email=Entry(*self*.root,textvariable=*self*.var\_email,font=("goudy old style",15),bg="white").place(x=850,y=150,width=180)**

***#====row 2============***

**lbl\_utype=Label(*self*.root,text="User Type",font=("goudy old style",15),bg="lightyellow").place(x=350,y=200)**

**txt\_utype=Entry(*self*.root,textvariable=*self*.var\_utype,font=("goudy old style",15),bg="lightyellow").place(x=500,y=200,width=180)**

**cmb\_utype=ttk.Combobox(*self*.root,textvariable=*self*.var\_utype,values=("Admin","Employee"),state='readonly',justify=CENTER,font=("time new roman",15))**

**cmb\_utype.place(x=500,y=200,width=180)**

**cmb\_utype.current(0)**

**lbl\_salary=Label(*self*.root,text="Salary",font=("goudy old style",15),bg="white").place(x=350,y=250)**

**txt\_salary=Entry(*self*.root,textvariable=*self*.var\_salary,font=("goudy old style",15),bg="lightyellow").place(x=500,y=250)**

***#=======buttons======***

**btn\_add=Button(*self*.root,text="Save",command=*self*.add,font=("goudy old style",15),bg="#2196f3",fg="white",cursor="hand2").place(x=400,y=300,width=110,height=25)**

**btn\_update=Button(*self*.root,text="Update",command=*self*.update,font=("goudy old style",15),bg="#4caf50",fg="white",cursor="hand2").place(x=520,y=300,width=110,height=25)**

**btn\_delete=Button(*self*.root,text="Delete",command=*self*.delete,font=("goudy old style",15),bg="#f44336",fg="white",cursor="hand2").place(x=640,y=300,width=110,height=25)**

**btn\_clear=Button(*self*.root,text="Clear",command=*self*.clear,font=("goudy old style",15),bg="#607d8b",fg="white",cursor="hand2").place(x=760,y=300,width=110,height=25)**

***#=====Employee Details====***

**emp\_frame=Frame(*self*.root,bd=3,relief=RIDGE)**

**emp\_frame.place(x=0,y=350,relwidth=1,height=150)**

**scrolly=Scrollbar(emp\_frame,orient=VERTICAL)**

**scrollx=Scrollbar(emp\_frame,orient=HORIZONTAL)**

***self*.EmployeeTable=ttk.Treeview(emp\_frame,columns=("eid","name","email","utype","salary"),yscrollcommand=scrolly.set,xscrollcommand=scrollx.set)**

**scrollx.pack(side=BOTTOM,fill=X)**

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

**scrollx.config(command=*self*.EmployeeTable.xview)**

**scrolly.config(command=*self*.EmployeeTable.yview)**

***self*.EmployeeTable.heading("eid",text="EMP ID")**

***self*.EmployeeTable.heading("name",text="Name")**

***self*.EmployeeTable.heading("email",text="Email")**

***self*.EmployeeTable.heading("utype",text="User type")**

***self*.EmployeeTable.heading("salary",text="Salary")**

***self*.EmployeeTable["show"]="headings"**

***self*.EmployeeTable.pack(fill=BOTH,expand=1)**

***self*.EmployeeTable.bind("<ButtonRelease-1>",*self*.get\_data)**

***self*.show()**

***#========================================================================================***

**def add(self):**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

**try:**

**if *self*.var\_emp\_id.get()=="":**

**messagebox.showerror("Error","Employee ID Must be Required",parent=*self*.root)**

**else:**

**cur.execute("Select \* from employee where eid=?",(*self*.var\_emp\_id.get(),))**

**row=cur.fetchone()**

**if row!=None:**

**messagebox.showerror("Error","This Employee ID already assigned, try different",parent=*self*.root)**

**else:**

**cur.execute("Insert into employee(eid,name,email,utype,salary)values(?,?,?,?,?)",(**

***self*.var\_emp\_id.get(),**

***self*.var\_name.get(),**

***self*.var\_email.get(),**

***self*.var\_utype.get(),**

***self*.var\_salary.get(),**

**))**

**con.commit()**

**messagebox.showinfo("Success","Employee Added Successfully",parent=*self*.root)**

***self*.show()**

**except Exception as ex:**

**messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=*self*.root)**

**def show(self):**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

**try:**

**cur.execute("select \* from employee")**

**rows=cur.fetchall()**

***self*.EmployeeTable.delete(\**self*.EmployeeTable.get\_children())**

**for row in rows:**

***self*.EmployeeTable.insert('',END,values=row)**

**except Exception as ex:**

**messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=*self*.root)**

**def get\_data(self,ev):**

**f=*self*.EmployeeTable.focus()**

**content=(*self*.EmployeeTable.item(f))**

**row=content['values']**

***# print(row)***

***self*.var\_emp\_id.set(row[0])**

***self*.var\_name.set(row[1])**

***self*.var\_email.set(row[2])**

***self*.var\_utype.set(row[3])**

***self*.var\_salary.set(row[4])**

**def update(self):**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

**try:**

**if *self*.var\_emp\_id.get()=="":**

**messagebox.showerror("Error","Employee ID Must be Required",parent=*self*.root)**

**else:**

**cur.execute("Select \* from employee where eid=?",(*self*.var\_emp\_id.get(),))**

**row=cur.fetchone()**

**if row==None:**

**messagebox.showerror("Error","Invalid Employee ID",parent=*self*.root)**

**else:**

**cur.execute("Update employee set name=?,email=?,utype=?,salary=? where eid=?",(**

***self*.var\_name.get(),**

***self*.var\_email.get(),**

***self*.var\_utype.get(),**

***self*.var\_salary.get(),**

***self*.var\_emp\_id.get(),**

**))**

**con.commit()**

**messagebox.showinfo("Success","Employee Updated Successfully",parent=*self*.root)**

***self*.show()**

**except Exception as ex:**

**messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=*self*.root)**

**def delete(self):**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

**try:**

**if *self*.var\_emp\_id.get()=="":**

**messagebox.showerror("Error","Employee ID Must be Required",parent=*self*.root)**

**else:**

**cur.execute("Select \* from employee where eid=?",(*self*.var\_emp\_id.get(),))**

**row=cur.fetchone()**

**if row==None:**

**messagebox.showerror("Error","Invalid Employee ID",parent=*self*.root)**

**else:**

**op=messagebox.askyesno("Confirm","Do You Really Want To Delete?",parent=*self*.root)**

**if op==True:**

**cur.execute("delete from employee where eid=?",(*self*.var\_emp\_id.get(),))**

**con.commit()**

**messagebox.showinfo("Delete","Employee Deleted successfully",parent=*self*.root)**

***self*.clear()**

**except Exception as ex:**

**messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=*self*.root)**

**def clear(self):**

***self*.var\_emp\_id.set("")**

***self*.var\_name.set("")**

***self*.var\_email.set("")**

***self*.var\_utype.set("Admin")**

***self*.var\_salary.set("")**

***self*.var\_searchtxt.set("")**

***self*.var\_searchby.set("Select")**

***self*.show()**

**def search(self):**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

**try:**

**if *self*.var\_searchby.get()=="Select":**

**messagebox.showerror("Error","Select Search By Option",parent=*self*.root)**

**elif *self*.var\_searchtxt.get()=="":**

**messagebox.showerror("Error","Select input should be required",parent=*self*.root)**

**else:**

**cur.execute("select \* from employee where "+*self*.var\_searchby.get()+" LIKE '%"+*self*.var\_searchtxt.get()+"%'")**

**rows=cur.fetchall()**

**if len(rows)!=0:**

***self*.EmployeeTable.delete(\**self*.EmployeeTable.get\_children())**

**for row in rows:**

***self*.EmployeeTable.insert('',END,values=row)**

**else:**

**messagebox.showerror("Error","No Record Found!!!",parent=*self*.root)**

**except Exception as ex:**

**messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=*self*.root)**

**if \_\_name\_\_=="\_\_main\_\_":**

**root=Tk()**

**obj=employeeClass(root)**

**root.mainloop()**

**Product**

**from tkinter import\***

**from PIL import Image, ImageTk**

**from tkinter import ttk,messagebox**

**import sqlite3**

**class productClass:**

**def \_\_init\_\_(self,root):**

***self*.root=root**

***self*.root.geometry("1100x500+220+130")**

***self*.root.title("Inventory Management System")**

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

***self*.root.focus\_force()**

***#===========All variables==================***

***self*.var\_searchby=StringVar()**

***self*.var\_searchtxt=StringVar()**

***self*.var\_pid=StringVar()**

***self*.var\_name=StringVar()**

***self*.var\_price=StringVar()**

***self*.var\_qty=StringVar()**

***self*.var\_status=StringVar()**

**product\_Frame=Frame(*self*.root,bd=3,relief=RIDGE,bg="white")**

**product\_Frame.place(x=10,y=10,width=450,height=480)**

***#====title========***

**title=Label(product\_Frame,text="Manage Product Details",font=("goudy old style",18),bg="#0f4d7d",fg="white").pack(side=TOP,fill=X)**

**lbl\_product=Label(product\_Frame,text="Name",font=("goudy old style",18),bg="white").place(x=30,y=60)**

**lbl\_price=Label(product\_Frame,text="Price",font=("goudy old style",18),bg="white").place(x=30,y=110)**

**lbl\_qty=Label(product\_Frame,text="Quantity",font=("goudy old style",18),bg="white").place(x=30,y=160)**

**lbl\_status=Label(product\_Frame,text="Status",font=("goudy old style",18),bg="white").place(x=30,y=210)**

***# =====column 2 =============***

**txt\_name = Entry(product\_Frame,textvariable=*self*.var\_name,font=("goudy old style",15),bg="light yellow").place(x=150,y=60,width=200)**

**txt\_price = Entry(product\_Frame,textvariable=*self*.var\_price,font=("goudy old style",15),bg="light yellow").place(x=150,y=110,width=200)**

**txt\_qty = Entry(product\_Frame,textvariable=*self*.var\_qty,font=("goudy old style",15),bg="light yellow").place(x=150,y=160,width=200)**

**cmb\_status = ttk.Combobox(product\_Frame,textvariable=*self*.var\_status,values=("Active", "Inactive"),state='readonly',justify=CENTER,font=("goudy old style",15))**

**cmb\_status.place(x=150,y=210,width=200)**

**cmb\_status.current(0)**

***#=======buttons======***

**btn\_add=Button(product\_Frame,text="Save",command=*self*.add,font=("goudy old style",15),bg="#2196f3",fg="white",cursor="hand2").place(x=10,y=400,width=100,height=40)**

**btn\_update=Button(product\_Frame,text="Update",command=*self*.update,font=("goudy old style",15),bg="#4caf50",fg="white",cursor="hand2").place(x=120,y=400,width=100,height=40)**

**btn\_delete=Button(product\_Frame,text="Delete",command=*self*.delete,font=("goudy old style",15),bg="#f44336",fg="white",cursor="hand2").place(x=230,y=400,width=100,height=40)**

**btn\_clear=Button(product\_Frame,text="Clear",command=*self*.clear,font=("goudy old style",15),bg="#607d8b",fg="white",cursor="hand2").place(x=340,y=400,width=100,height=40)**

***#========searchFrame======***

**SearchFrame=LabelFrame(*self*.root,text="Search Product",font=("goudy old style",12,"bold"),bd=2,relief=RIDGE,bg="white")**

**SearchFrame.place(x=480,y=10,width=600,height=80)**

***#=======options======***

**cmb\_search=ttk.Combobox(SearchFrame,textvariable=*self*.var\_searchby,values=("Select","Product Name"),state='readonly',justify=CENTER,font=("time new roman",15))**

**cmb\_search.place(x=10,y=10,width=180)**

**cmb\_search.current(0)**

**txt\_search=Entry(SearchFrame,textvariable=*self*.var\_searchtxt,font=("goudy old style",15),bg="lightyellow").place(x=200,y=10)**

**btn\_search=Button(SearchFrame,text="Search",command=*self*.search,font=("goudy old style",15),bg="#4caf50",fg="white").place(x=410,y=10,width=150,height=30)**

***# ============= Product details ====================================================================================================***

**p\_frame=Frame(*self*.root,bd=3,relief=RIDGE)**

**p\_frame.place(x=480,y=100,width=600,height=390)**

**scrolly=Scrollbar(p\_frame,orient=VERTICAL)**

**scrollx=Scrollbar(p\_frame,orient=HORIZONTAL)**

***self*.product\_table=ttk.Treeview(p\_frame,columns=("pid","name","price","qty","status"),yscrollcommand=scrolly.set,xscrollcommand=scrollx.set)**

**scrollx.pack(side=BOTTOM,fill=X)**

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

**scrollx.config(command=*self*.product\_table.xview)**

**scrolly.config(command=*self*.product\_table.yview)**

***self*.product\_table.heading("pid",text="Product ID")**

***self*.product\_table.heading("name",text="Product Name")**

***self*.product\_table.heading("price",text="Price")**

***self*.product\_table.heading("qty",text="Quantity")**

***self*.product\_table.heading("status",text="Status")**

***self*.product\_table["show"]="headings"**

***self*.product\_table.pack(fill=BOTH,expand=1)**

***self*.product\_table.bind("<ButtonRelease-1>",*self*.get\_data)**

***self*.product\_table.column("pid",width=100)**

***self*.product\_table.column("name",width = 90)**

***self*.product\_table.column("price",width = 90)**

***self*.product\_table.column("qty",width=90)**

***self*.product\_table.column("status",width = 90)**

***self*.show()**

***#========================================================================================***

**def add(self):**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

**try:**

**if *self*.var\_name.get()=="" or *self*.var\_qty.get()=="" or *self*.var\_price.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 name is already added, try different",parent=*self*.root)**

**else:**

**cur.execute("Insert into product(name,price,qty,status)values(?,?,?,?)",(**

***self*.var\_name.get(),**

***self*.var\_price.get(),**

***self*.var\_qty.get(),**

***self*.var\_status.get(),**

**))**

**con.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)}",parent=*self*.root)**

**def show(self):**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

**try:**

**cur.execute("select \* from product")**

**rows=cur.fetchall()**

***self*.product\_table.delete(\**self*.product\_table.get\_children())**

**for row in rows:**

***self*.product\_table.insert('',END,values=row)**

**except Exception as ex:**

**messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=*self*.root)**

**def get\_data(self,ev):**

**f=*self*.product\_table.focus()**

**content=(*self*.product\_table.item(f))**

**row=content['values']**

***# print(row)***

***self*.var\_pid.set(row[0])**

***self*.var\_name.set(row[1])**

***self*.var\_price.set(row[2])**

***self*.var\_qty.set(row[3])**

***self*.var\_status.set(row[4])**

**def update(self):**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

**try:**

**if *self*.var\_pid.get()=="":**

**messagebox.showerror("Error","Please select product from list",parent=*self*.root)**

**else:**

**cur.execute("Select \* from product where pid=?",(*self*.var\_pid.get(),))**

**row=cur.fetchone()**

**if row==None:**

**messagebox.showerror("Error","Invalid product ID",parent=*self*.root)**

**else:**

**cur.execute("Update product set name=?,price=?,qty=?,status=? where pid=?",(**

***self*.var\_name.get(),**

***self*.var\_price.get(),**

***self*.var\_qty.get(),**

***self*.var\_status.get(),**

***self*.var\_pid.get()**

**))**

**con.commit()**

**messagebox.showinfo("Success","Product Updated Successfully",parent=*self*.root)**

***self*.show()**

**except Exception as ex:**

**messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=*self*.root)**

**def delete(self):**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

**try:**

**if *self*.var\_pid.get()=="":**

**messagebox.showerror("Error","Select product from list",parent=*self*.root)**

**else:**

**cur.execute("Select \* from product where pid=?",(*self*.var\_pid.get(),))**

**row=cur.fetchone()**

**if row==None:**

**messagebox.showerror("Error","Invalid Product",parent=*self*.root)**

**else:**

**op=messagebox.askyesno("Confirm","Do You Really Want To Delete?",parent=*self*.root)**

**if op==True:**

**cur.execute("delete from product where pid=?",(*self*.var\_pid.get(),))**

**con.commit()**

**messagebox.showinfo("Delete","Product Deleted successfully",parent=*self*.root)**

***self*.clear()**

**except Exception as ex:**

**messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=*self*.root)**

**def clear(self):**

***self*.var\_name.set("")**

***self*.var\_price.set("")**

***self*.var\_qty.set("")**

***self*.var\_pid.set("")**

***self*.var\_status.set("Active")**

***self*.var\_searchtxt.set("")**

***self*.var\_searchby.set("Select")**

***self*.show()**

**def search(self):**

**con=sqlite3.connect(database=r'inventorymanagementsystem.db')**

**cur=con.cursor()**

**try:**

**if *self*.var\_searchby.get()=="Select":**

**messagebox.showerror("Error","Select Search By Option",parent=*self*.root)**

**elif *self*.var\_searchtxt.get()=="":**

**messagebox.showerror("Error","Product Name Input is required",parent=*self*.root)**

**else:**

**cur.execute("select \* from product where "+*self*.var\_searchby.get()+" LIKE '%"+*self*.var\_searchtxt.get()+"%'")**

**rows=cur.fetchall()**

**if len(rows)!=0:**

***self*.product\_table.delete(\**self*.product\_table.get\_children())**

**for row in rows:**

***self*.product\_table.insert('',END,values=row)**

**else:**

**messagebox.showerror("Error","No Record Found!!!",parent=*self*.root)**

**except Exception as ex:**

**messagebox.showerror("Error",f"Error due to : {str(ex)}",parent=*self*.root)**

**if \_\_name\_\_=="\_\_main\_\_":**

**root=Tk()**

**obj=productClass(root)**

**root.mainloop()**

**Sales**

**from sqlite3.dbapi2 import Cursor**

**from tkinter import\***

**from PIL import Image, ImageTk**

**from tkinter import ttk,messagebox**

**import sqlite3**

**import os**

**class salesClass:**

**def \_\_init\_\_(self,root):**

***self*.root=root**

***self*.root.geometry("1100x500+220+130")**

***self*.root.title("Inventory Management System")**

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

***self*.root.focus\_force()**

***#============= Variables ==================***

***self*.bill\_list = []**

***self*.var\_invoice=StringVar()**

***#====title========***

**title=Label(*self*.root,text="View Customer Bill",font=("goudy old style",30),bg="#0f4d7d",fg="white").place(x=0,y=0,width=1100)**

**lbl\_invoice = Label(*self*.root, text="Invoice", 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 ="light yellow").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="light grey",fg="black",cursor = 'hand2').place(x=360,y=100,width=120,height=28)**

**btn\_clear = Button (*self*.root, text= "Clear",command=*self*.clear,font=("Times New roman",15,"bold") ,bg="Light grey",fg="black",cursor = 'hand2').place(x=490,y=100,width=120,height=28)**

***#========== Bill Lists ==========***

**sales\_Frame = Frame(*self*.root,bd=3,relief=RIDGE)**

**sales\_Frame.place(x=50,y=140,width=200,height=300)**

**scrolly = Scrollbar(sales\_Frame,orient=VERTICAL)**

***self*.sales\_List = Listbox(sales\_Frame,font=("goudy old style",15),bg = "white" ,yscrollcommand= scrolly.set)**

***self*.sales\_List.pack(fill=BOTH,expand=1)**

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

**scrolly.config(command=*self*.sales\_List.yview)**

***self*.sales\_List.bind("<ButtonRelease-1>", *self*.get\_data)**

***#========== Bill Area ==========***

**bill\_Frame = Frame(*self*.root,bd=3,relief=RIDGE)**

**bill\_Frame.place(x=280,y=140,width=410,height=300)**

**lbl\_title2=Label(bill\_Frame,text="Customer Bill Area",font=("goudy old style",20),bg= "black",fg="white").pack(side=TOP,fill=X)**

**scrolly2 = Scrollbar(bill\_Frame,orient=VERTICAL)**

***self*.bill\_area = Listbox(bill\_Frame,font=("goudy old style",15),bg = "light grey" ,yscrollcommand= scrolly2.set)**

***self*.bill\_area.pack(fill=BOTH,expand=1)**

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

**scrolly2.config(command=*self*.bill\_area.yview)**

***# ================ Image ================***

***self*.bill\_photo = Image.open("sales.png")**

***self*.bill\_photo=*self*.bill\_photo.resize((350,315), Image.ANTIALIAS)**

***self*.bill\_photo=ImageTk.PhotoImage(*self*.bill\_photo)**

**lbl\_image = Label(*self*.root,image = *self*.bill\_photo)**

**lbl\_image.place(x=700,y=110)**

***self*.show()**

***# ===============================================================***

**def show(self):**

**del *self*.bill\_list[:]**

***self*.sales\_List.delete(0,END)**

**for i in os.listdir('bill'):**

**if i.split('.')[-1]=='txt':**

***self*.sales\_List.insert(END,i)**

***self*.bill\_list.append(i.split('.')[0])**

**def get\_data(self,ev):**

**index\_=*self*.sales\_List.curselection()**

**file\_name = *self*.sales\_List.get(index\_)**

***# print(file\_name)***

***self*.bill\_area.delete(0,END)**

**fp= open(f'bill/{file\_name}','r')**

**for i in fp:**

***self*.bill\_area.insert(END,i)**

**fp.close()**

**def search(self):**

**if *self*.var\_invoice.get()=="":**

**messagebox.showerror("Error","Invoice no. should be required", parent = *self*.root)**

**else:**

**if *self*.var\_invoice.get() in *self*.bill\_list:**

***# print("yes")***

**fp=open(f'bill/{*self*.var\_invoice.get()}.txt','r')**

***self*.bill\_area.delete(0,END)**

**for i in fp:**

***self*.bill\_area.insert(END,i)**

**fp.close()**

**else:**

**messagebox.showerror("Error","Invalid Invoice No.", parent = *self*.root)**

**def clear(self):**

***self*.show()**

***self*.bill\_area.delete(0,END)**

**if \_\_name\_\_=="\_\_main\_\_":**

**root=Tk()**

**obj=salesClass(root)**

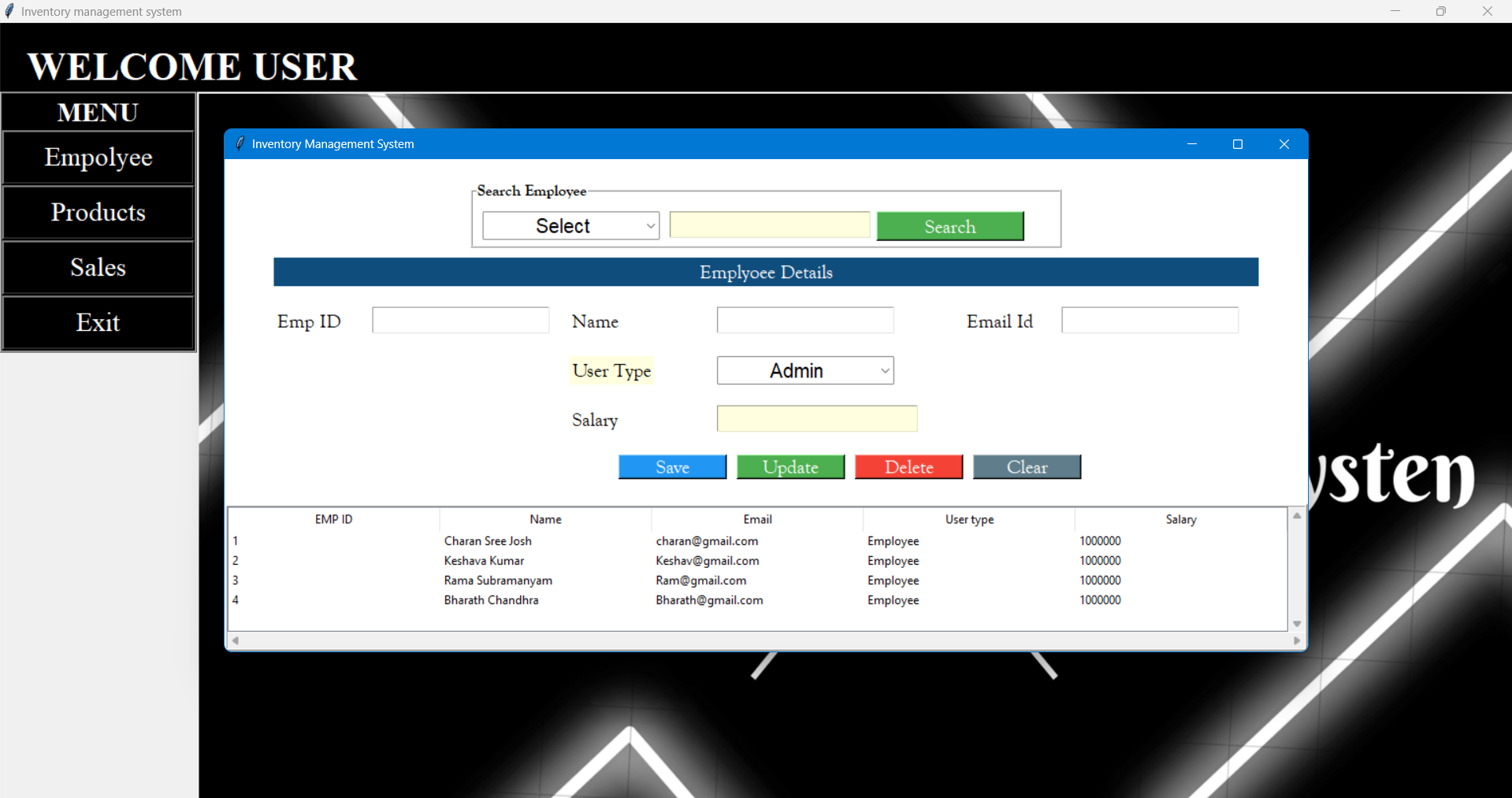
**root.mainloop()**

**5. SOFTWARE TESTING:**

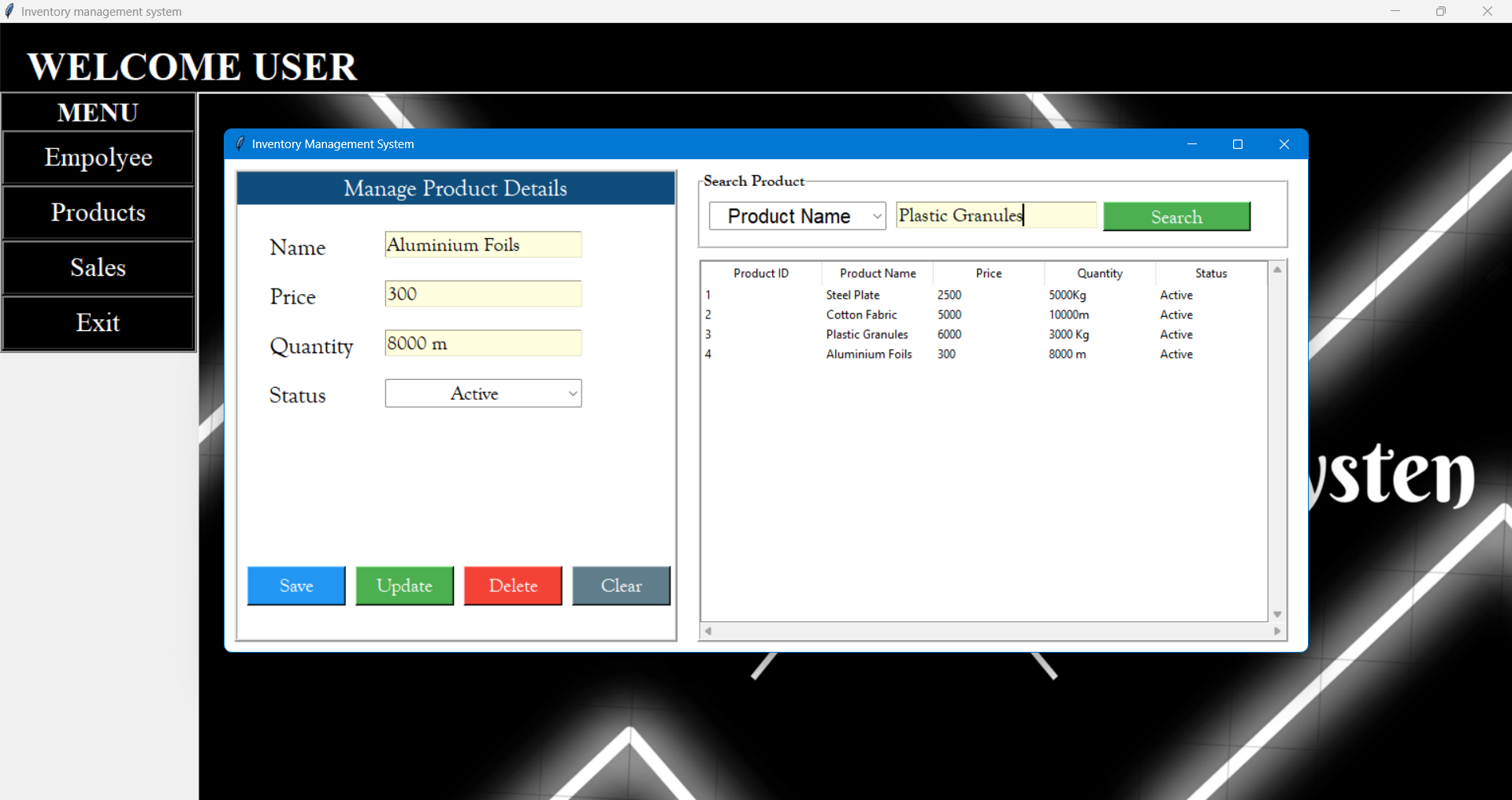
1. **Dashboard:**

****

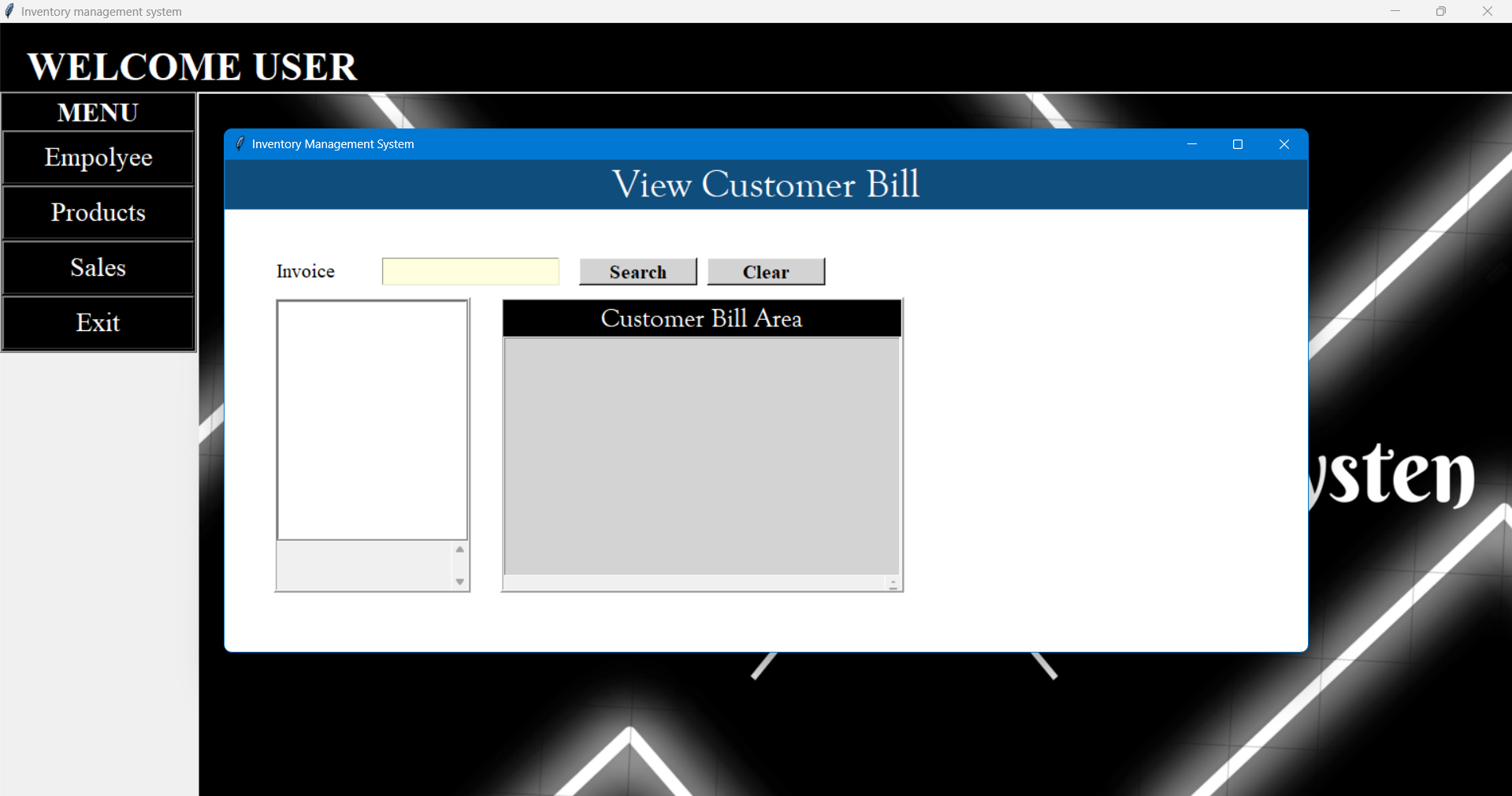
1. **Testing of Adding Employee details:**

****

1. **Testing of Adding Product details:**

****

1. **Testing of Customer billing area:**

****