DEPARTMENTAL STORE MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

MOHAMMED SUHAIB V	**220701169**
MADHAN RAJ P	**220701148**
MOHAMED NAWFAL SALAM M	**220701168**

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI-602105

2024

BONAFIDE CERTIFICATE

Certified that this project report "**DEPARTMENTAL STORE MANAGEMENT SYSTEM**" is the bonafide work of "**MOHAMMED SUHAIB V (220701169), MADHAN RAJ P (220701148), MOHAMED NAWFAL SALAM M (220701168)**" who carried out the project under my supervision.

Submitted for the Practical Examination held on ________________________

SIGNATURE
Dr.R.SABITHA
Professor and II Year Academic Head
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam, Chennai - 602 105

SIGNATURE
Ms.D.KALPANA
Assistant Professor (SG),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam, Chennai - 602 105

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

The Departmental Store Management System is a comprehensive application designed to streamline and automate the various functions of a departmental store. Developed using Python's Tkinter for the graphical user interface and SQLite for database management, this system offers an intuitive and efficient way to manage crucial aspects of store operations. The application encompasses multiple functionalities, including the management of stores, employees, customers, sales, suppliers, products, and categories. Each of these components is represented by a dedicated tab within the user interface, allowing users to add, view, and delete records seamlessly.

This project exemplifies the practical integration of database management and user interface design to solve real-world business challenges. By automating routine tasks and providing a user-friendly interface, the Departmental Store Management System significantly reduces the administrative burden on store personnel, thereby allowing them to focus on more strategic activities. This system not only facilitates better data management but also supports informed decision-making through easy access to critical information.

TABLE OF CONTENTS

CHAPTER 1

1.1 INTRODUCTION

The Departmental Store Management System provides an intuitive interface where users can easily add, view, and delete records related to different entities such as stores, employees, customers, sales, suppliers, products, and categories. This centralized approach not only simplifies the data management process but also enhances the accuracy and accessibility of information. The application is designed to cater to the needs of both small and large departmental stores, offering scalability and flexibility to adapt to varying business requirements.

1.2 OBJECTIVES

A key objective is to improve operational efficiency by automating routine tasks, allowing store personnel to focus on strategic activities rather than mundane data entry. The system is designed to facilitate easy access to and management of critical store data through an intuitive interface, ensuring that information is readily available for decision-making. Maintaining data integrity and security is also a priority, with SQLite providing a reliable and secure platform for data management.

1.3 MODULES

- Store Management Module.
- Employee Management Module.
- Customer Management Module.
- Sales Management Module.
- Supplier Management Module.
- Product Management Module.
- Category Management Module.
- Database Management Module.

CHAPTER-2

2.1 SOFTWARE DESCRIPTION

Visual studio Code:

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

2.2 LANGUAGES

1. Python:

- It is used for scripting the application's logic, managing database operations, and integrating different modules.

2. Tkinter:

- Tkinter is the standard Python interface to the Tk GUI toolkit. It is used for developing the graphical user interface (GUI) of the application.

3. SQLite:

- SQLite is used as the database management system for the project. It is an embedded SQL database engine that provides a lightweight and efficient way to store and manage data.

CHAPTER-3

REQUIREMENT AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION:

The Departmental Store Management System must enable users to efficiently manage various store operations. Users should be able to add, view, and delete store details such as Store ID, Store Name, and Location. Employee management functionalities should allow adding, viewing, and deleting employee records, including Employee ID, First Name, Last Name, Role, and Salary. Customer management features should support adding, viewing, and deleting customer records with details like Customer ID, First Name, Last Name, Email, and Phone.

Sales management should facilitate recording, viewing, and deleting sales transactions, including Sale ID, Date, and Total Price. Supplier management should enable adding, viewing, and deleting supplier information such as Supplier ID, Supplier Name, and Contact Info. Product management should allow adding, viewing, and deleting product details including Product ID, Product Name, Description, Price, and Stock Quantity. Additionally, category management should support adding, viewing, and deleting categories with Category ID and Category Name.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS:

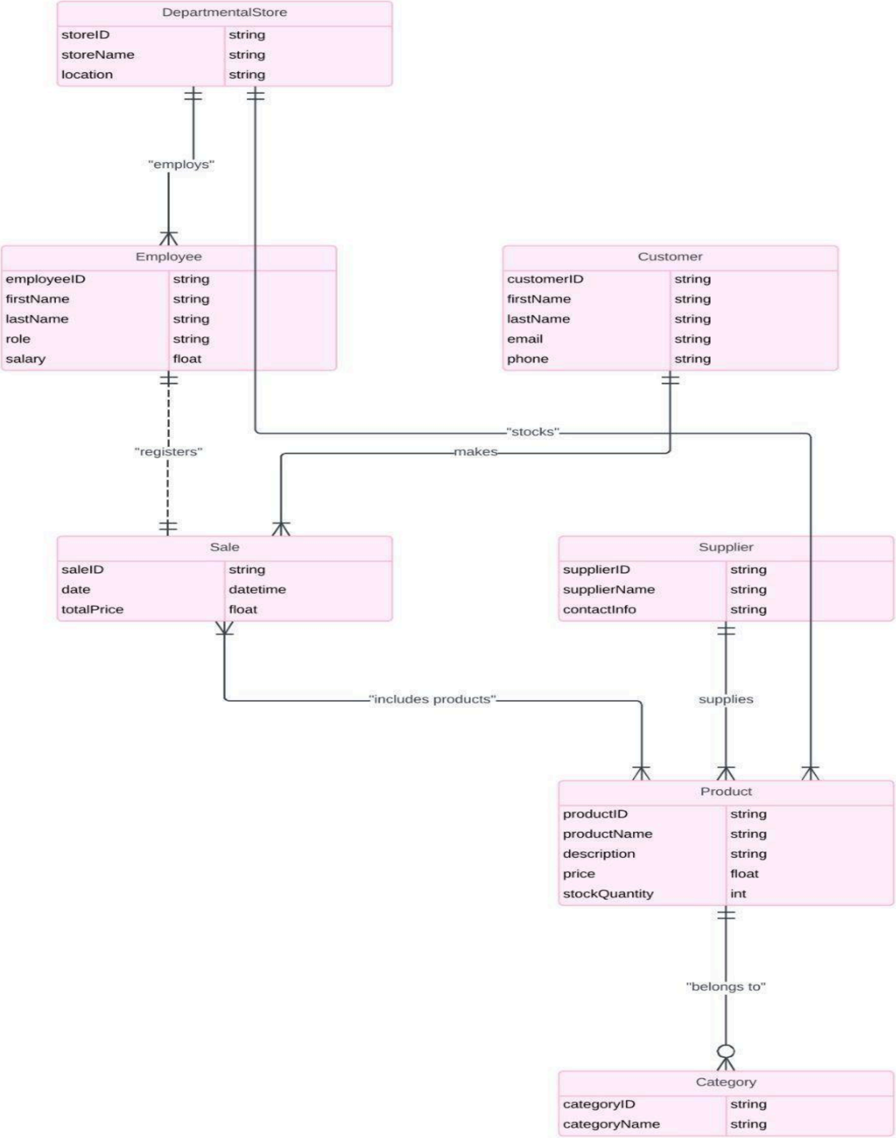
Hardware Requirements:-

- Processor: 1 GHz or faster processor
- RAM: 2 GB or more
- Storage: At least 500 MB of available disk space
- Display: Minimum resolution of 1024x768
- Input Devices: Keyboard and mouse

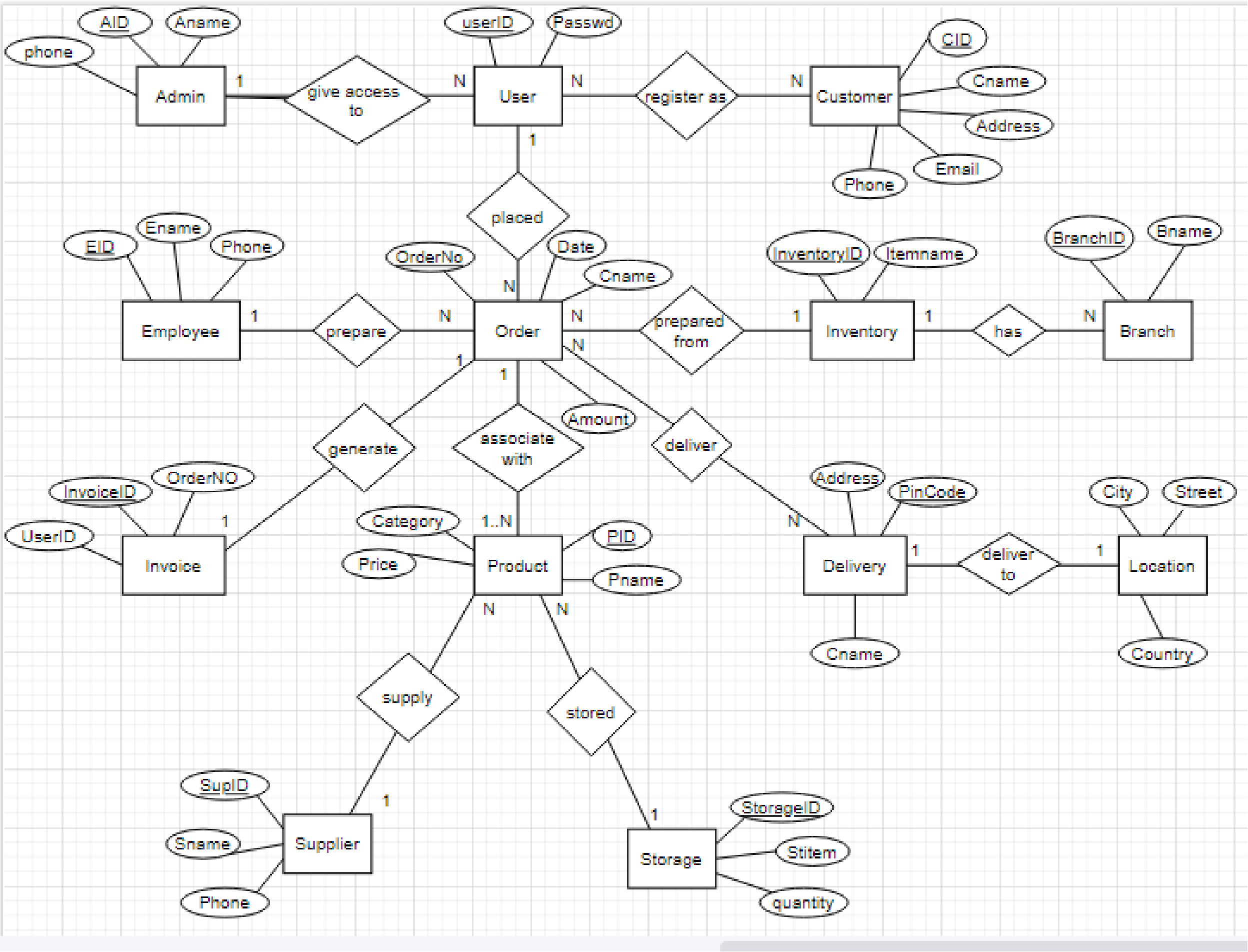
Software Requirements:-

- Operating System: Windows 7 or later, macOS, or Linux
- Python: Version 3.6 or higher
- SQLite: Version 3 or higher
- Python Libraries:
  - ‘tkinter’ for GUI development (included with Python)
  - ‘sqlite3’ for database management (included with Python)

3.3 ARCHITECTURE DIAGRAM:



3.4 ER DIAGRAM:



CHAPTER-4

PROGRAM CODE

```
import sqlite3
from tkinter import *
from tkinter import messagebox, ttk
from datetime import datetime

conn = sqlite3.connect('departmental_store.db')
c = conn.cursor()
c.execute('''CREATE TABLE IF NOT EXISTS DepartmentalStore (
            StoreID TEXT PRIMARY KEY,
            StoreName TEXT NOT NULL,
            Location TEXT NOT NULL
         )''')

c.execute('''CREATE TABLE IF NOT EXISTS Employee (
            EmployeeID TEXT PRIMARY KEY,
            FirstName TEXT NOT NULL,
            LastName TEXT NOT NULL,
            Role TEXT NOT NULL,
            Salary REAL NOT NULL
         )''')

c.execute('''CREATE TABLE IF NOT EXISTS Customer (
            CustomerID TEXT PRIMARY KEY,
            FirstName TEXT NOT NULL,
            LastName TEXT NOT NULL,
            Email TEXT NOT NULL,
            Phone TEXT NOT NULL
         )''')

c.execute('''CREATE TABLE IF NOT EXISTS Sales (
            SaleID TEXT PRIMARY KEY,
            Date TEXT NOT NULL,
            TotalPrice REAL NOT NULL
         )''')

c.execute('''CREATE TABLE IF NOT EXISTS Supplier (
            SupplierID TEXT PRIMARY KEY,
            SupplierName TEXT NOT NULL,
            ContactInfo TEXT NOT NULL
         )''')

c.execute('''CREATE TABLE IF NOT EXISTS Product (
            ProductID TEXT PRIMARY KEY,
            ProductName TEXT NOT NULL,
            Description TEXT NOT NULL,
            Price REAL NOT NULL,
            StockQuantity INTEGER NOT NULL
         )''')
```

```
c.execute('''CREATE TABLE IF NOT EXISTS Category (
                CategoryID TEXT PRIMARY KEY,
                CategoryName TEXT NOT NULL
            )''')

conn.commit()

def add_store():
    store_id = entry_store_id.get()
    store_name = entry_store_name.get()
    location = entry_store_location.get()
    c.execute('INSERT INTO DepartmentalStore (StoreID, StoreName, Location)
VALUES (?, ?, ?)',
              (store_id, store_name, location))
    conn.commit()
    messagebox.showinfo("Success", "Store added successfully.")
    entry_store_id.delete(0, END)
    entry_store_name.delete(0, END)
    entry_store_location.delete(0, END)

def view_stores():
    for row in tree_stores.get_children():
        tree_stores.delete(row)
    c.execute('SELECT * FROM DepartmentalStore')
    stores = c.fetchall()
    for store in stores:
        tree_stores.insert("", "end", values=store)
```

```
def add_employee():
    emp_id = entry_emp_id.get()
    first_name = entry_emp_first_name.get()
    last_name = entry_emp_last_name.get()
    role = entry_emp_role.get()
    salary = float(entry_emp_salary.get())
    c.execute('INSERT INTO Employee (EmployeeID, FirstName, LastName, Role,
Salary) VALUES (?, ?, ?, ?, ?)',
              (emp_id, first_name, last_name, role, salary))
    conn.commit()
    messagebox.showinfo("Success", "Employee added successfully.")
    entry_emp_id.delete(0, END)
    entry_emp_first_name.delete(0, END)
    entry_emp_last_name.delete(0, END)
    entry_emp_role.delete(0, END)
    entry_emp_salary.delete(0, END)

def view_employees():
    for row in tree_employees.get_children():
        tree_employees.delete(row)
    c.execute('SELECT * FROM Employee')
    employees = c.fetchall()
    for emp in employees:
        tree_employees.insert("", "end", values=emp)

def add_customer():
    cust_id = entry_cust_id.get()
    first_name = entry_cust_first_name.get()
    last_name = entry_cust_last_name.get()
    email = entry_cust_email.get()
    phone = entry_cust_phone.get()
    c.execute('INSERT INTO Customer (CustomerID, FirstName, LastName, Email, Phone) VALUES (?, ?, ?, ?, ?)',
              (cust_id, first_name, last_name, email, phone))
    conn.commit()
    messagebox.showinfo("Success", "Customer added successfully.")
    entry_cust_id.delete(0, END)
    entry_cust_first_name.delete(0, END)
    entry_cust_last_name.delete(0, END)
    entry_cust_email.delete(0, END)
    entry_cust_phone.delete(0, END)

def view_customers():
    for row in tree_customers.get_children():
        tree_customers.delete(row)
    c.execute('SELECT * FROM Customer')
    customers = c.fetchall()
    for cust in customers:
        tree_customers.insert("", "end", values=cust)

def add_sale():
    sale_id = entry_sale_id.get()
    date = entry_sale_date.get()
    total_price = float(entry_sale_total_price.get())
    c.execute('INSERT INTO Sales (SaleID, Date, TotalPrice) VALUES (?, ?, ?)',
              (sale_id, date, total_price))
    conn.commit()
    messagebox.showinfo("Success", "Sale recorded successfully.")
    entry_sale_id.delete(0, END)
    entry_sale_date.delete(0, END)
    entry_sale_total_price.delete(0, END)

def view_sales():
    for row in tree_sales.get_children():
        tree_sales.delete(row)
    c.execute('SELECT * FROM Sales')
    sales = c.fetchall()
    for sale in sales:
        tree_sales.insert("", "end", values=sale)

def add_supplier():
    supplier_id = entry_supplier_id.get()
    supplier_name = entry_supplier_name.get()
    contact_info = entry_supplier_contact_info.get()
    c.execute('INSERT INTO Supplier (SupplierID, SupplierName, ContactInfo)
VALUES (?, ?, ?)',
              (supplier_id, supplier_name, contact_info))
    conn.commit()
    messagebox.showinfo("Success", "Supplier added successfully.")
    entry_supplier_id.delete(0, END)
    entry_supplier_name.delete(0, END)
    entry_supplier_contact_info.delete(0, END)

def view_suppliers():
    for row in tree_suppliers.get_children():
        tree_suppliers.delete(row)
    c.execute('SELECT * FROM Supplier')
    suppliers = c.fetchall()
    for supplier in suppliers:
        tree_suppliers.insert("", "end", values=supplier)

def add_product():
    product_id = entry_product_id.get()
    product_name = entry_product_name.get()
    description = entry_product_description.get()
    price = float(entry_product_price.get())
    stock_quantity = int(entry_product_stock_quantity.get())
    c.execute('INSERT INTO Product (ProductID, ProductName, Description, Price,
StockQuantity) VALUES (?, ?, ?, ?, ?)',
              (product_id, product_name, description, price, stock_quantity))
    conn.commit()
    messagebox.showinfo("Success", "Product added successfully.")
    entry_product_id.delete(0, END)
    entry_product_name.delete(0, END)
    entry_product_description.delete(0, END)
    entry_product_price.delete(0, END)
    entry_product_stock_quantity.delete(0, END)
```

```
def view_products():
    for row in tree_products.get_children():
        tree_products.delete(row)
    c.execute('SELECT * FROM Product')
    products = c.fetchall()
    for product in products:
        tree_products.insert("", "end", values=product)

def add_category():
    category_id = entry_category_id.get()
    category_name = entry_category_name.get()
    c.execute('INSERT INTO Category (CategoryID, CategoryName) VALUES (?,
?)',
              (category_id, category_name))
    conn.commit()
    messagebox.showinfo("Success", "Category added successfully.")
    entry_category_id.delete(0, END)
    entry_category_name.delete(0, END)

def view_categories():
    for row in tree_categories.get_children():
        tree_categories.delete(row)
    c.execute('SELECT * FROM Category')
    categories = c.fetchall()
    for category in categories:
        tree_categories.insert("", "end", values=category)
```

```
root = Tk()
root.title("Departmental Store Management System")

tabControl = ttk.Notebook(root)
tab_stores = ttk.Frame(tabControl)
tab_employees = ttk.Frame(tabControl)
tab_customers = ttk.Frame(tabControl)
tab_sales = ttk.Frame(tabControl)
tab_suppliers = ttk.Frame(tabControl)
tab_products = ttk.Frame(tabControl)
tab_categories = ttk.Frame(tabControl)

tabControl.add(tab_stores, text='Stores')
tabControl.add(tab_employees, text='Employees')
tabControl.add(tab_customers, text='Customers')
tabControl.add(tab_sales, text='Sales')
tabControl.add(tab_suppliers, text='Suppliers')
tabControl.add(tab_products, text='Products')
tabControl.add(tab_categories, text='Categories')

tabControl.pack(expand=1, fill="both")

frame_store_form = Frame(tab_stores)
frame_store_form.pack(side=TOP, fill=X, pady=10, padx=10)
```

```
Label(frame_store_form, text="Store ID").grid(row=0, column=0, padx=5, pady=5)
entry_store_id = Entry(frame_store_form)
entry_store_id.grid(row=0, column=1, padx=5, pady=5)

Label(frame_store_form, text="Store Name").grid(row=1, column=0, padx=5, pady=5)
entry_store_name = Entry(frame_store_form)
entry_store_name.grid(row=1, column=1, padx=5, pady=5)

Label(frame_store_form, text="Location").grid(row=2, column=0, padx=5, pady=5)
entry_store_location = Entry(frame_store_form)
entry_store_location.grid(row=2, column=1, padx=5, pady=5)

Button(frame_store_form, text="Add Store", command=add_store).grid(row=3, column=1, pady=10)

frame_store_tree = Frame(tab_stores)
frame_store_tree.pack(side=TOP, fill=BOTH, expand=True)

tree_stores = ttk.Treeview(frame_store_tree, columns=("StoreID", "StoreName", "Location"), show='headings')
tree_stores.heading("StoreID", text="Store ID")
tree_stores.heading("StoreName", text="Store Name")
tree_stores.heading("Location", text="Location")
tree_stores.pack(fill=BOTH, expand=True)
```

```
Button(frame_store_tree, text="View Stores",
command=view_stores).pack(side=LEFT, padx=5, pady=5)

frame_emp_form = Frame(tab_employees)
frame_emp_form.pack(side=TOP, fill=X, pady=10, padx=10)

Label(frame_emp_form, text="Employee ID").grid(row=0, column=0, padx=5,
pady=5)
entry_emp_id = Entry(frame_emp_form)
entry_emp_id.grid(row=0, column=1, padx=5, pady=5)

Label(frame_emp_form, text="First Name").grid(row=1, column=0, padx=5,
pady=5)
entry_emp_first_name = Entry(frame_emp_form)
entry_emp_first_name.grid(row=1, column=1, padx=5, pady=5)

Label(frame_emp_form, text="Last Name").grid(row=2, column=0, padx=5,
pady=5)
entry_emp_last_name = Entry(frame_emp_form)
entry_emp_last_name.grid(row=2, column=1, padx=5, pady=5)

Label(frame_emp_form, text="Role").grid(row=3, column=0, padx=5, pady=5)
entry_emp_role = Entry(frame_emp_form)
entry_emp_role.grid(row=3, column=1, padx=5, pady=5)

Label(frame_emp_form, text="Salary").grid(row=4, column=0, padx=5, pady=5)
```

```
entry_emp_salary = Entry(frame_emp_form)
entry_emp_salary.grid(row=4, column=1, padx=5, pady=5)

Button(frame_emp_form, text="Add Employee",
command=add_employee).grid(row=5, column=1, pady=10)

frame_emp_tree = Frame(tab_employees)
frame_emp_tree.pack(side=TOP, fill=BOTH, expand=True)

tree_employees = ttk.Treeview(frame_emp_tree, columns=("EmployeeID",
"FirstName", "LastName", "Role", "Salary"), show='headings')
tree_employees.heading("EmployeeID", text="Employee ID")
tree_employees.heading("FirstName", text="First Name")
tree_employees.heading("LastName", text="Last Name")
tree_employees.heading("Role", text="Role")
tree_employees.heading("Salary", text="Salary")
tree_employees.pack(fill=BOTH, expand=True)

Button(frame_emp_tree, text="View Employees",
command=view_employees).pack(side=LEFT, padx=5, pady=5)

frame_cust_form = Frame(tab_customers)
frame_cust_form.pack(side=TOP, fill=X, pady=10, padx=10)

Label(frame_cust_form, text="Customer ID").grid(row=0, column=0, padx=5,
pady=5)
entry_cust_id = Entry(frame_cust_form)
```

```
entry_cust_id.grid(row=0, column=1, padx=5, pady=5)

Label(frame_cust_form, text="First Name").grid(row=1, column=0, padx=5,
pady=5)
entry_cust_first_name = Entry(frame_cust_form)
entry_cust_first_name.grid(row=1, column=1, padx=5, pady=5)

Label(frame_cust_form, text="Last Name").grid(row=2, column=0, padx=5,
pady=5)
entry_cust_last_name = Entry(frame_cust_form)
entry_cust_last_name.grid(row=2, column=1, padx=5, pady=5)

Label(frame_cust_form, text="Email").grid(row=3, column=0, padx=5, pady=5)
entry_cust_email = Entry(frame_cust_form)
entry_cust_email.grid(row=3, column=1, padx=5, pady=5)

Label(frame_cust_form, text="Phone").grid(row=4, column=0, padx=5, pady=5)
entry_cust_phone = Entry(frame_cust_form)
entry_cust_phone.grid(row=4, column=1, padx=5, pady=5)

Button(frame_cust_form, text="Add Customer",
command=add_customer).grid(row=5, column=1, pady=10)

frame_cust_tree = Frame(tab_customers)
frame_cust_tree.pack(side=TOP, fill=BOTH, expand=True)
```

```
tree_customers = ttk.Treeview(frame_cust_tree, columns=("CustomerID",
"FirstName", "LastName", "Email", "Phone"), show='headings')
tree_customers.heading("CustomerID", text="Customer ID")
tree_customers.heading("FirstName", text="First Name")
tree_customers.heading("LastName", text="Last Name")
tree_customers.heading("Email", text="Email")
tree_customers.heading("Phone", text="Phone")
tree_customers.pack(fill=BOTH, expand=True)

Button(frame_cust_tree, text="View Customers",
command=view_customers).pack(side=LEFT, padx=5, pady=5)

frame_sale_form = Frame(tab_sales)
frame_sale_form.pack(side=TOP, fill=X, pady=10, padx=10)

Label(frame_sale_form, text="Sale ID").grid(row=0, column=0, padx=5, pady=5)
entry_sale_id = Entry(frame_sale_form)
entry_sale_id.grid(row=0, column=1, padx=5, pady=5)

Label(frame_sale_form, text="Date").grid(row=1, column=0, padx=5, pady=5)
entry_sale_date = Entry(frame_sale_form)
entry_sale_date.grid(row=1, column=1, padx=5, pady=5)

Label(frame_sale_form, text="Total Price").grid(row=2, column=0, padx=5,
pady=5)
entry_sale_total_price = Entry(frame_sale_form)
entry_sale_total_price.grid(row=2, column=1, padx=5, pady=5)
```

```
Button(frame_sale_form, text="Add Sale", command=add_sale).grid(row=3,
column=1, pady=10)

frame_sale_tree = Frame(tab_sales)
frame_sale_tree.pack(side=TOP, fill=BOTH, expand=True)

tree_sales = ttk.Treeview(frame_sale_tree, columns=("SaleID", "Date",
"TotalPrice"), show='headings')
tree_sales.heading("SaleID", text="Sale ID")
tree_sales.heading("Date", text="Date")
tree_sales.heading("TotalPrice", text="Total Price")
tree_sales.pack(fill=BOTH, expand=True)

Button(frame_sale_tree, text="View Sales",
command=view_sales).pack(side=LEFT, padx=5, pady=5)

frame_supplier_form = Frame(tab_suppliers)
frame_supplier_form.pack(side=TOP, fill=X, pady=10, padx=10)

Label(frame_supplier_form, text="Supplier ID").grid(row=0, column=0, padx=5,
pady=5)
entry_supplier_id = Entry(frame_supplier_form)
entry_supplier_id.grid(row=0, column=1, padx=5, pady=5)

Label(frame_supplier_form, text="Supplier Name").grid(row=1, column=0,
padx=5, pady=5)
```

```
entry_supplier_name = Entry(frame_supplier_form)
entry_supplier_name.grid(row=1, column=1, padx=5, pady=5)

Label(frame_supplier_form, text="Contact Info").grid(row=2, column=0, padx=5,
pady=5)
entry_supplier_contact_info = Entry(frame_supplier_form)
entry_supplier_contact_info.grid(row=2, column=1, padx=5, pady=5)

Button(frame_supplier_form, text="Add Supplier",
command=add_supplier).grid(row=3, column=1, pady=10)

frame_supplier_tree = Frame(tab_suppliers)
frame_supplier_tree.pack(side=TOP, fill=BOTH, expand=True)

tree_suppliers = ttk.Treeview(frame_supplier_tree, columns=("SupplierID",
"SupplierName", "ContactInfo"), show='headings')
tree_suppliers.heading("SupplierID", text="Supplier ID")
tree_suppliers.heading("SupplierName", text="Supplier Name")
tree_suppliers.heading("ContactInfo", text="Contact Info")
tree_suppliers.pack(fill=BOTH, expand=True)

Button(frame_supplier_tree, text="View Suppliers",
command=view_suppliers).pack(side=LEFT, padx=5, pady=5)

frame_product_form = Frame(tab_products)
frame_product_form.pack(side=TOP, fill=X, pady=10, padx=10)
```

```
Label(frame_product_form, text="Product ID").grid(row=0, column=0, padx=5,
pady=5)
entry_product_id = Entry(frame_product_form)
entry_product_id.grid(row=0, column=1, padx=5, pady=5)

Label(frame_product_form, text="Product Name").grid(row=1, column=0,
padx=5, pady=5)
entry_product_name = Entry(frame_product_form)
entry_product_name.grid(row=1, column=1, padx=5, pady=5)

Label(frame_product_form, text="Description").grid(row=2, column=0, padx=5,
pady=5)
entry_product_description = Entry(frame_product_form)
entry_product_description.grid(row=2, column=1, padx=5, pady=5)

Label(frame_product_form, text="Price").grid(row=3, column=0, padx=5, pady=5)
entry_product_price = Entry(frame_product_form)
entry_product_price.grid(row=3, column=1, padx=5, pady=5)

Label(frame_product_form, text="Stock Quantity").grid(row=4, column=0,
padx=5, pady=5)
entry_product_stock_quantity = Entry(frame_product_form)
entry_product_stock_quantity.grid(row=4, column=1, padx=5, pady=5)

Button(frame_product_form, text="Add Product",
command=add_product).grid(row=5, column=1, pady=10)
```

```
frame_product_tree = Frame(tab_products)
frame_product_tree.pack(side=TOP, fill=BOTH, expand=True)

tree_products = ttk.Treeview(frame_product_tree, columns=("ProductID",
"ProductName", "Description", "Price", "StockQuantity"), show='headings')
tree_products.heading("ProductID", text="Product ID")
tree_products.heading("ProductName", text="Product Name")
tree_products.heading("Description", text="Description")
tree_products.heading("Price", text="Price")
tree_products.heading("StockQuantity", text="Stock Quantity")
tree_products.pack(fill=BOTH, expand=True)

Button(frame_product_tree, text="View Products",
command=view_products).pack(side=LEFT, padx=5, pady=5)

frame_category_form = Frame(tab_categories)
frame_category_form.pack(side=TOP, fill=X, pady=10, padx=10)

Label(frame_category_form, text="Category ID").grid(row=0, column=0, padx=5,
pady=5)
entry_category_id = Entry(frame_category_form)
entry_category_id.grid(row=0, column=1, padx=5, pady=5)

Label(frame_category_form, text="Category Name").grid(row=1, column=0,
padx=5, pady=5)
entry_category_name = Entry(frame_category_form)
entry_category_name.grid(row=1, column=1, padx=5, pady=5)
```

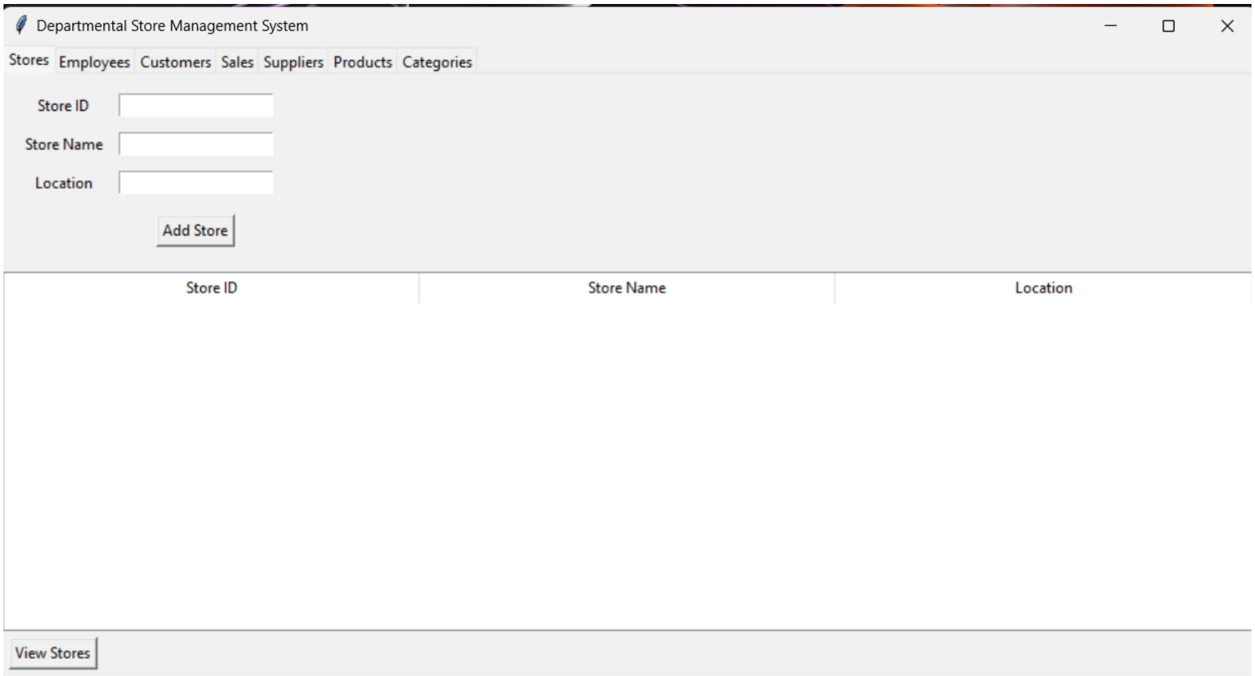
```
Button(frame_category_form, text="Add Category",
command=add_category).grid(row=2, column=1, pady=10)

frame_category_tree = Frame(tab_categories)
frame_category_tree.pack(side=TOP, fill=BOTH, expand=True)

tree_categories = ttk.Treeview(frame_category_tree, columns=("CategoryID",
"CategoryName"), show='headings')
tree_categories.heading("CategoryID", text="Category ID")
tree_categories.heading("CategoryName", text="Category Name")
tree_categories.pack(fill=BOTH, expand=True)

Button(frame_category_tree, text="View Categories",
command=view_categories).pack(side=LEFT, padx=5, pady=5)

root.mainloop()

conn.close()
```

CHAPTER-5

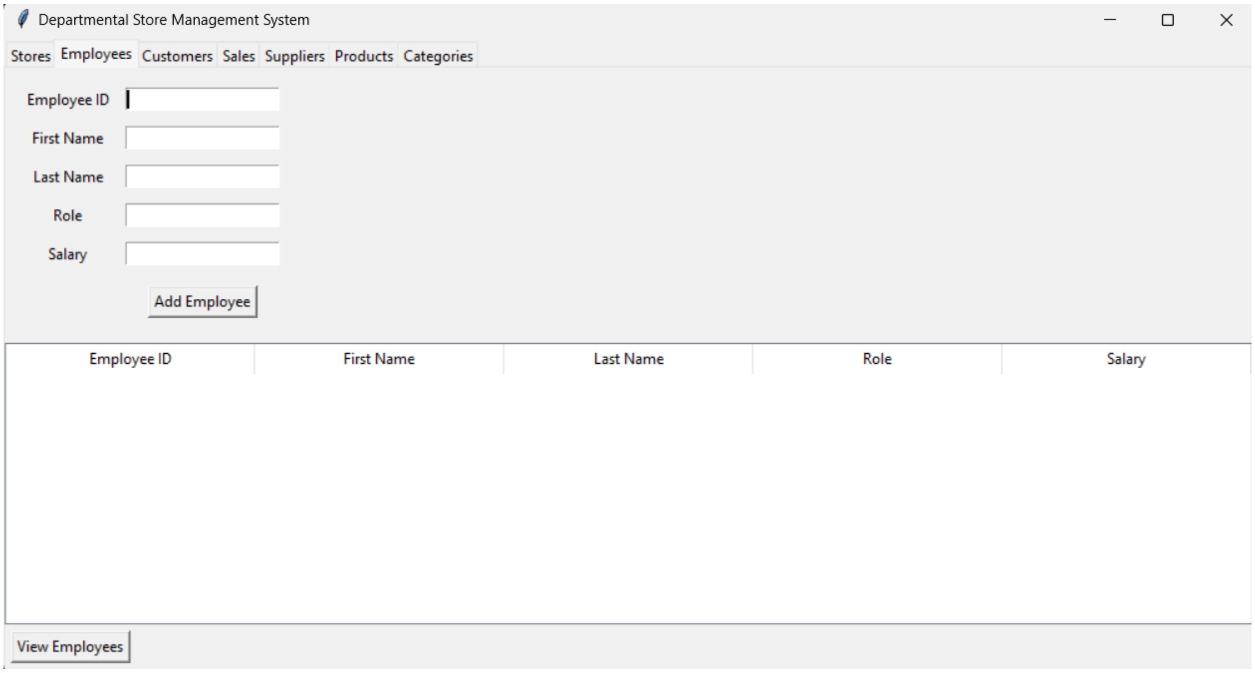
RESULTS AND DISCUSSION

5.1 USER DOCUMENTATION:

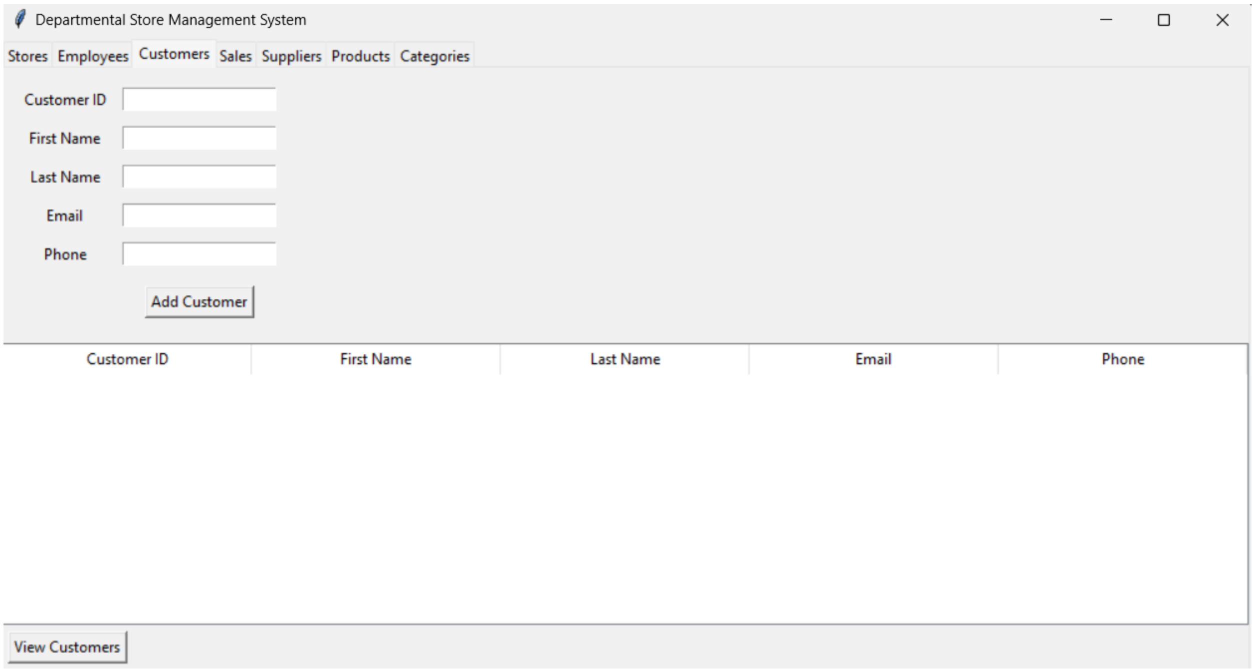
STORE MANAGEMENT MODULE:



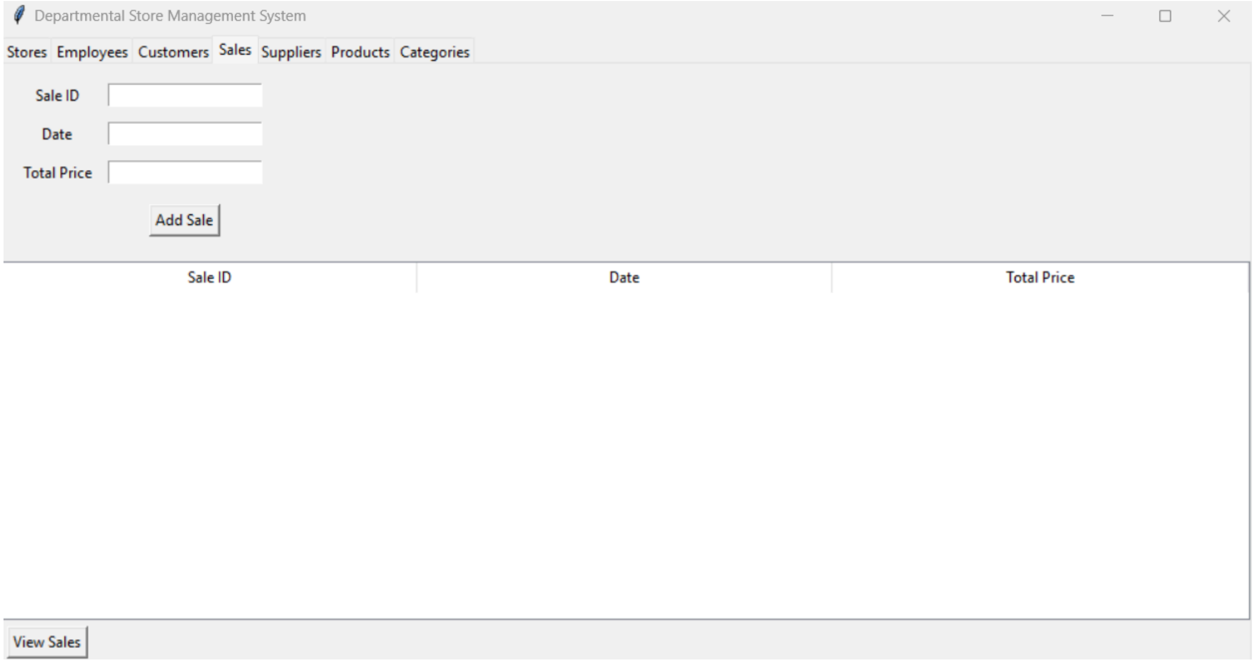
EMPLOYEES MANAGEMENT MODULE:



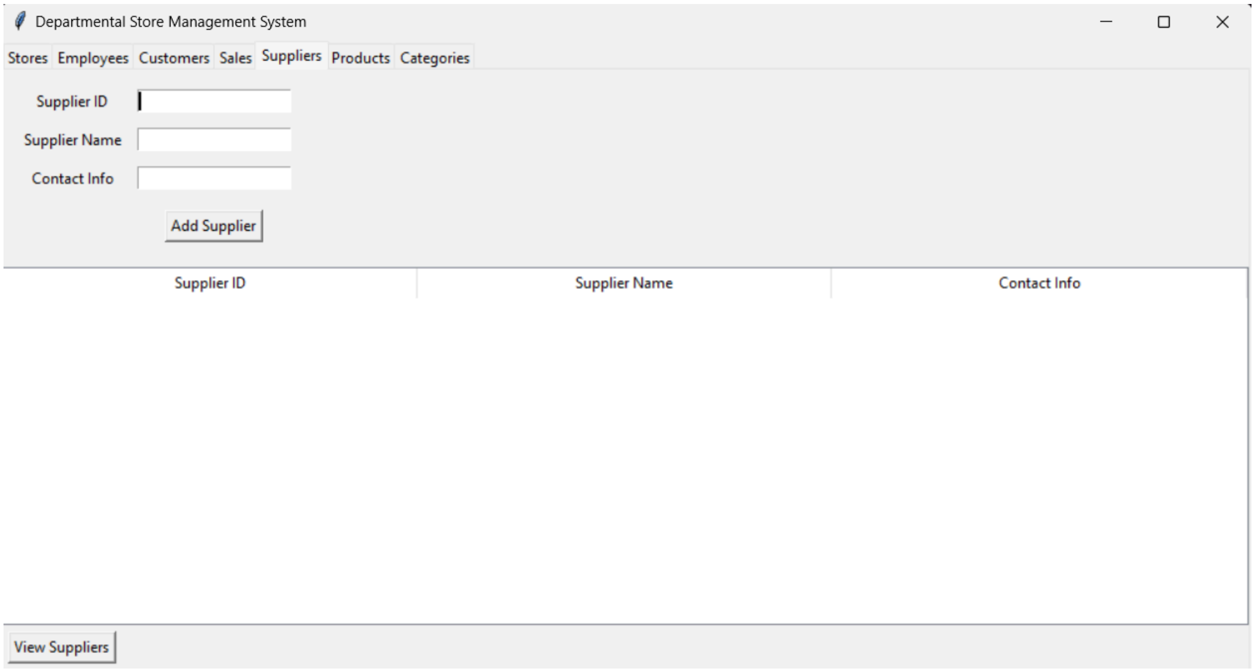
CUSTOMERS MANAGEMENT MODULE:



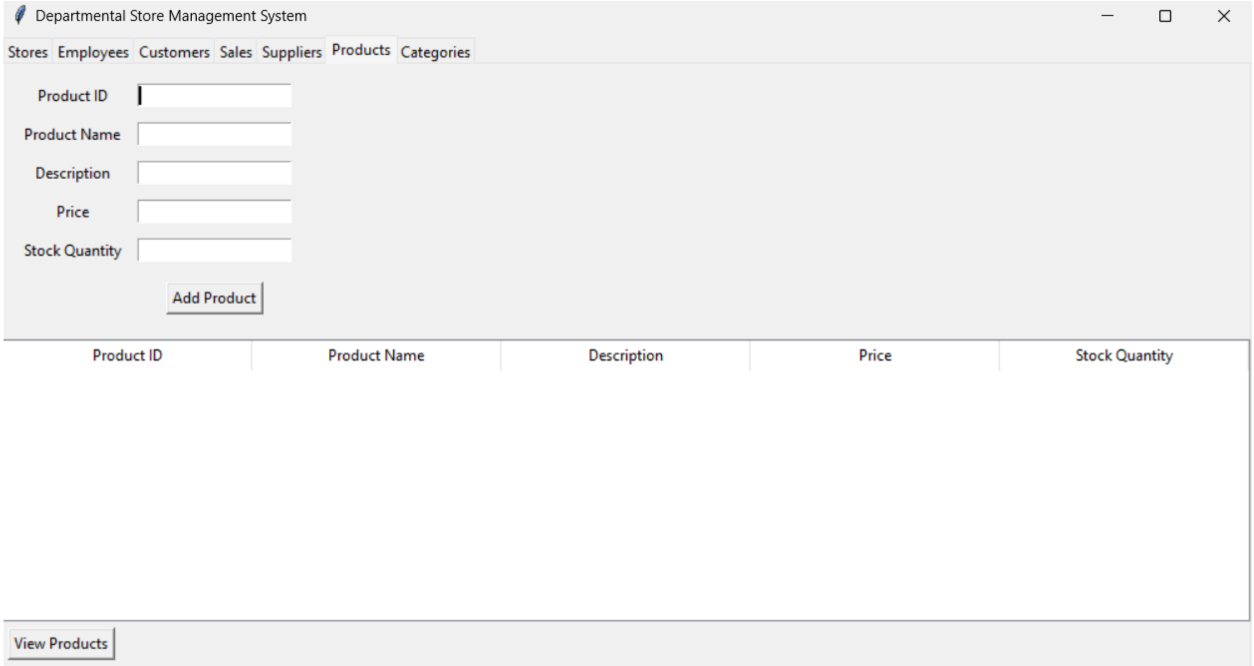
SALES MANAGEMENT MODULE:



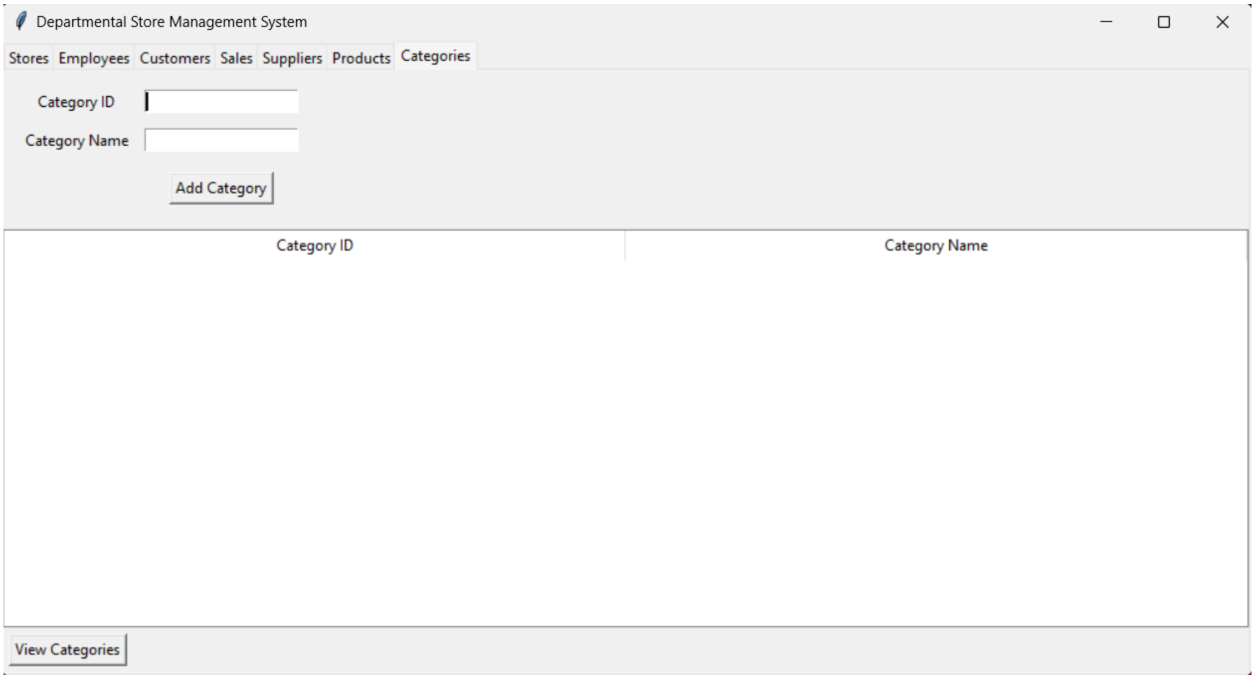
SUPPLIERS MANAGEMENT MODULE:



PRODUCTS MANAGEMENT MODULE:



CATEGORIES MANAGEMENT MODULE:



CHAPTER-6

6.1 CONCLUSION:

After completing the Departmental Store Management System project, we are confident that it effectively addresses the challenges present in traditional store management systems. This computerized system is designed to reduce human errors and significantly enhance operational efficiency. The primary goal of this project was to minimize human effort by automating routine tasks and streamlining the management processes.For instance, users can simply type a search string to quickly find specific records, and editing records is simplified through easy-to-use update functions.

Overall, the Departmental Store Management System achieves its primary objective of providing accurate and efficient management of store operations. By automating data management and simplifying navigation and editing, the system ensures that store personnel can focus on more strategic tasks. This project demonstrates the potential of integrating modern software solutions to enhance the effectiveness of retail management, ultimately leading to better decision-making and improved business outcomes.

CHAPTER-7

7.1 REFERENCES:

1. Python Documentation: Python Software Foundation. (n.d.). Python Documentation. Retrieved from https://docs.python.org/3/

2. Tkinter Documentation: TkDocs. (n.d.). Tkinter Tutorial. Retrieved from https://tkdocs.com/tutorial/

3. SQLite Documentation: SQLite. (n.d.). SQLite Documentation. Retrieved from https://www.sqlite.org/docs.html

4. Automate the Boring Stuff with Python: Sweigart, A. (2015). Automate the Boring Stuff with Python: Practical Programming for Total Beginners. No Starch Press.

5. Python GUI Programming with Tkinter: Grayson, J. E. (2000). Python and Tkinter Programming. Manning Publications.

6. SQLite Database Programming: Owens, M. (2006). The Definitive Guide to SQLite. Apress.

7. Database System Concepts: Silberschatz, A., Korth, H. F., & Sudarshan, S. (2010). Database System Concepts (6th ed.). McGraw-Hill.

8. GeeksforGeeks: Various authors. (n.d.). GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/

9. W3Schools: W3Schools. (n.d.). SQL Tutorial. Retrieved from https://www.w3schools.com/sql/