# Warehouse Management System

CMPT 308N

Section 200

**The Big Boyz Team**


Marist College

School Of Computer Science and Mathematics

Submitted to:
Dr. Reza Sadeghi

Fall 2022

# Table of Content



Table Of Figures

# Progress Report: The Big Boyz Warehouse Management System

**Team Name**

The Big Boys

**Team Members:**

Ricky Junior Isheja Rickyjunior.isheja1@marist.edu

My name is Ricky Junior Isheja, I am an international student from Rwanda, and I am a sophomore majoring in computer science, my general passions are coding, basketball, and soccer. I chose my teammates on the basis that we all believe in hard work, and we are all willing to invest in the necessary efforts to finish this project together and succeed in the course.

Saeed Abdilahi Saeed.abdilahi1@marist.edu

My name is Saeed Abdilahi, and I am from Somaliland. I am Junior, studying computer science. The way I selected my current teammates was by asking them if we could be in a group team.

Descartes Tuyishime descartes.tuyishime1@marist.edu (Team Leader)

My name is Descartes Tuyishime, and I am an international student from Rwanda. I am a senior, majoring in Computer Science and Data Analytics. I am interested in the use of machine learning to improve agricultural and medical field. I selected my team based on flexibility, motivation, and responsibility. I admire people who are flexible and motivated to do what needs to be done. I trust my team members to be responsible enough to take this course and project seriously.

## Group Project Selected: Project 4 (Warehouse Management System)

The Big boys warehouse management system is a system that will help people to know, manage and maintain materials kept in the warehouse. The System will be managing details like the number of products, their store time, the price and the weight of the products, The system will also help the user to search a specific product. The system will make easier the accounting processes of the user's business which might in the overall increase the profit. The objective of the big boys is to make a warehouse

management system by the end of this course that can help an admin user to enter the food types, books, carts and any products, the store time in the warehouse, the pick out time, see the prices and search a product. And we shall also create a user's page for the user to buy and check out products.

## Review the related work

- **Amazon Warehouse**

Amazon Warehouse works by label and packaging item and order that came through electronic orders that people who connect with Amazon made. The processes that the packages are labeled and boxed are physically by using human hands [1].

- **Target Warehouse**

Target Warehouses uses similar processes of packaging and distribution and then delivering to the right destination, the same as Amazon. However, the only difference is the brand of the item or product and its price sector [2].

- **Apple Warehouse**

Apple Warehouse is the same as the other two warehouses above, but the difference is that apple mainly focuses on electronics whereas the two provide multiple products. All in all, Apple, Amazon, and Target use the same warehouse management system. All of these are held by humans [3].

**Advantages of a Warehouse Management System**

- Reduced Operating & Processing Expenses
- Reduced Mispacks
- Improved Customer Relations

**Costs that come with having a Warehouse Management Systems**

- Requires Expert Knowledge
- Requires Tight Security
- High Initial Investment

## Merits Of the Project

1. Our Warehouse Management System (WMS) will lead to a drastic change in operating expenses since it reduces the manpower and other resources that are needed to manage and control the products that are in the warehouse.

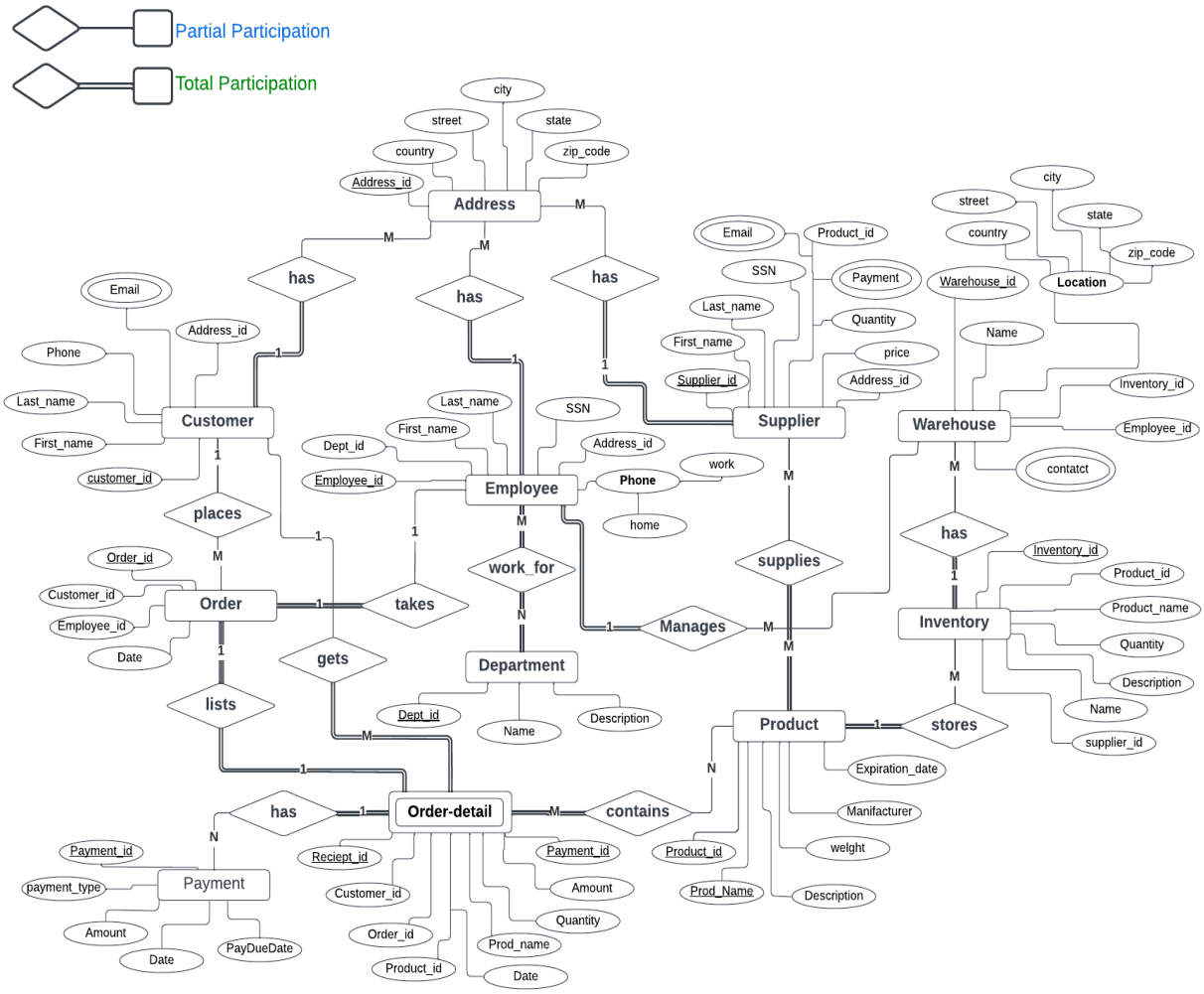
2. The Labor will be better allocated with our system since most of the work is done by the system so the labor will have time to do their work in a more effective way while also being efficient.
3. WMS will increase the overall customer satisfaction, since the Service provided will be faster and easier for both the seller and the buyer, which will also increase the customers loyalty to the business thus increased profitability on the businesses side.
4. WMS will facilitate and help to manage the inventory that comes in and gets out (2), which overall will facilitate the accounting process at the end of the fiscal year.

## Entity Relationship Diagram (ER Diagram)

Our Warehouse Management System (WMS) consists of 11 entities which are: Warehouse, inventory, supplier, product, employee, department, customer, address, order, order detail, and payment. Below are the business rules that determined the entities in our WMS (Check out figure 1 below.).

A warehouse has a unique identification, a name, location, an inventory, a manager, and contacts for communications. A warehouse has one or more inventories, and an inventory belongs to one and only one warehouse. An inventory has a unique identification, name, product identification, product name, quantity, and product description, supplier identification. Inventory contains stores products, and products have unique identification, product name, description, weight, manufacturer, and expiration date. A warehouse is managed by an employee, and an employee has employee unique identification, department identification, first name, last name, Social Security number, Address identification, and phone number. Each employee belongs to a certain department, and a department has department unique identification, name, and Description. Each product is supplied by a supplier, and a supplier has unique identification, first name, last name, social security number, email, product identification, payment, quantity, and Address identification. Our WMS has customers, and a customer is represented by customer unique identification, first name, last name, email, and address identification. Customer, Employee, and Supplier all have address, and the Address has address unique identification, country, street, city, state, zip code. A Customer can place an order, and an order is taken by the employee. An order has a unique identification, customer identification, employee identification, and date. Each order has an order detail, and an order detail contains products. Each order detail list receipt unique identification, customer identification, order identification, product identification, date, product name, quantity, amount, and payment identification. Each detail has payment, and each payment consists of unique payment identification, payment type, amount, date, and pay due date. To

## Figure 1: The Big Boyz Warehouse Management System ER Diagram




*Designed using Lucid.app.*

## Enhanced Entity Relationship Diagram (EER diagram)

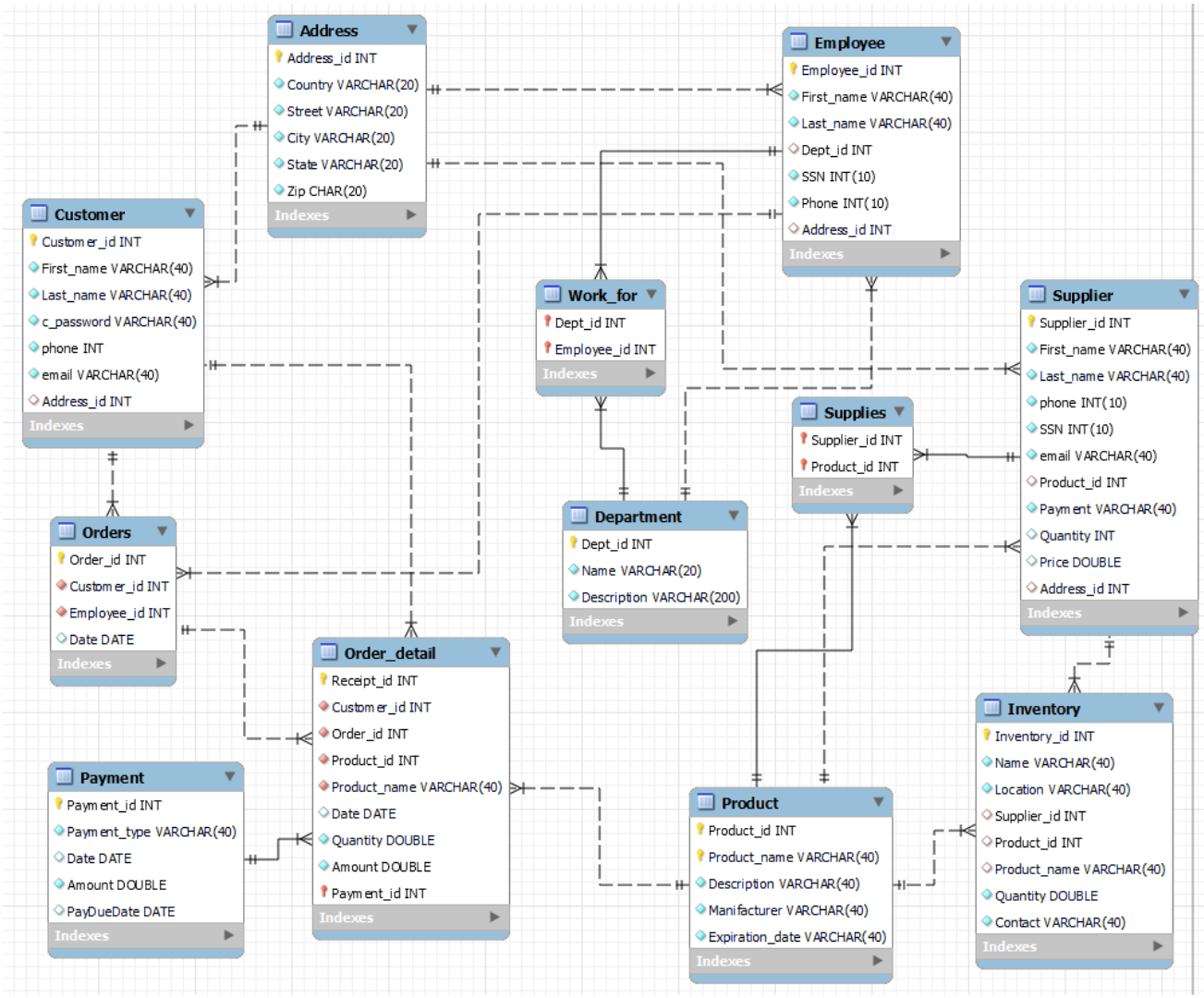
As described above, our warehouse management system is composed of 11 entities which are: Warehouse, inventory, supplier, product, employee, department, customer, address, order, order detail, and payment (Check figure 2).

A warehouse has many inventories, and an inventory belongs to one warehouse. A warehouse entity has warehouse identification (warehouse_id) as the primary key and has foreign keys inventory identification (inventory_id), and employee identification (employee_id) from inventory and employee respectively. The inventory stores may products, and a product is in an inventory. The suppliers supply products to one to more inventory. The inventory entity has primary key in inventory identification (inventory_id), and it

has foreign keys in product identification (product_id), product name (product_name), and supplier identification(supplier_id). Supplier entity has supplier identification as a primary key, and it has foreign keys in product identification (product_id) and address identification (address_id). The product entity has product identification (product_id) and product name (product_name) as a composite primary key. The employee entity has employee identification (employee_id) as a primary key. Department entity has department identification (dept_id) as a primary key. Customer entity has customer identification (customer_id) as a primary key. Address entity has address identification (address_id) as a primary key. Order entity has order identification (order_id) as a primary key. Order detail has receipt identification (receipt_id) and payment identification (payment_id) as composite primary keys. Every order detail has a payment, and every payment corresponds to one and only one order detail. The payment entity has payment identification (payment_id) as a primary key.

## Figure 2: The Big Boyz Warehouse Management System EER Diagram

CMPT308N-200_Project Progress Report Phase #6
_The Big Boyz Team




## Database Development

We first created a database called warehouse and in the same line we checked, if there is no other database with the same name, we then created a table called address with 5 attributes namely address id (primary key), Country, street, city, state, and zip. This table is used to record the addresses of three tables the customer, the supplier, and the employee and it will have address_id as foreign key in all of the three tables. We created a table called customer which will be recording customer details. It has 6 attributes including Customer_id (Primary key), first_name, Last_name, phone, email and address_id (foreign key) from address table. This table will have 3 main connections with the order details table, the order table and the adrerss table. we then created the table of Department, this table will be recording the various departments we have in our Warehouse management company. It has 3 main attributes namely dept_id(Primary key), name and description. This table is directly related to the employee table. we then created a table called employee and its in charge of keeping data of our employees.  It has 7 attributes, employee_id (primary_key), first_name, Last_name, dept_id (foreign key), ssn, phone, address_id (foreign_key). This table is related directly to 4 tables, the warehouse table, the address table, the department table and the order table. Here we then created the products table, this table records a specific product that is being taken into the inventory or out of the inventory. It has 5 attributes, the product_id

(Primary key), product_name, description, Manifacturer and expiration date. This table is related to three other tables, the order_detail. After that create a table of order, it handles the order information and saves it. This table has 4 attributes, Order_id (Primary_Key) , customer_id (Foreign_Key), Employee_Id (Foreign Key) and the date. This table is directly related to the customer table, the order detail table and the employee table. We then create the payment table that records the payment details related to the customer and the order detail. It has five attributes, Payment_id(Primary Key), payment_type, date, amount and paydueDate. This table is related to order detail table. The order deatil table is a table that records information of the order that has been made. It has nine attributes, Receipt_id ( Primary Key), customer_id (Foreign_key), Order_id( Foreign Key), Product_id (Foreign_key), Product_name( Foreign_key), date, quantity,amount, payment_id( Primary Key). This table is directly related to Customer table, Payment table, order table, product table. we then create a table called inventory, it will hold details of the products that are in the inventory. This table has eight attributes, Inventory_id( Primary_key), name,location, supplier_id (Foreign_key), product_id(Foreign Key), Product_name( Foreign_Key), quantity, contact. This table is related to the supplier table, the warehouse supplier and the products table. After that We create a table called Warehouse, this table is of recording what enters and gets out of our warehouse. It has six attributes, warehouse_id ( Primary_key), name, location, Inventory ( Foreign Key), Employee(Foreign Key), Contact. This table is connected directly to the employee table and to the inventory. Lastly we create a table called roles that will be recording a user id whether the user is a supplier, Admin, or buyer.

## Code Implementation

```
drop database Warehouse;
create database if not exists Warehouse;
show databases;
use Warehouse;

CREATE TABLE Address (
  Address_id INT AUTO_INCREMENT,
  Country varchar(20) NOT NULL,
  Street varchar(20) NOT NULL ,
  City varchar(20) NOT NULL ,
  State varchar(20) NOT NULL ,
  Zip char(20) NOT NULL,
  PRIMARY KEY (Address_id)
);

CREATE TABLE Customer (
  Customer_id INT AUTO_INCREMENT,
  First_name varchar(40) NOT NULL,
  Last_name varchar(40) NOT NULL ,
  phone int(10) NOT NULL ,
  email varchar(40) NOT NULL ,
  Address_id INT,
  PRIMARY KEY (Customer_id),
  FOREIGN KEY (Address_id) REFERENCES Address(Address_id)
);

CREATE TABLE Department (
  Dept_id INT AUTO_INCREMENT,
```

```
  Name varchar(40) NOT NULL,
  Description varchar(40) NOT NULL ,
  PRIMARY KEY (Dept_id)
);

CREATE TABLE Employee (
  Employee_id INT AUTO_INCREMENT,
  First_name varchar(40) NOT NULL,
  Last_name varchar(40) NOT NULL,
  Dept_id INT,
  SSN int(10) NOT NULL ,
  Phone int(10) NOT NULL ,
  Address_id INT,
  PRIMARY KEY (Employee_id),
  FOREIGN KEY (Address_id) REFERENCES Address(Address_id),
  FOREIGN KEY (Dept_id) REFERENCES Department(Dept_id)
);

CREATE TABLE Product (
  Product_id INT AUTO_INCREMENT,
  Product_name varchar(40) NOT NULL,
  Description varchar(40) NOT NULL ,
  Manifacturer varchar(40) NOT NULL ,
  Expiration_date varchar(40) NOT NULL ,
  PRIMARY KEY (Product_id, Product_name )
);

select * from Product;
CREATE TABLE Supplier (
  Supplier_id INT AUTO_INCREMENT,
  First_name varchar(40) NOT NULL,
  Last_name varchar(40) NOT NULL ,
  phone int(10) NOT NULL ,
  SSN int(10) NOT NULL ,
  email varchar(40) NOT NULL ,
  Product_id INT,
  Payment varchar(40) NOT NULL,
  Quantity INT,
  Price DOUBLE,
  Address_id INT,
  PRIMARY KEY (Supplier_id),
  FOREIGN KEY (Address_id) REFERENCES Address(Address_id),
  FOREIGN KEY (Product_id) REFERENCES Product(Product_id)
);

CREATE TABLE _Order (
  Order_id INT AUTO_INCREMENT,
  Customer_id INT NOT NULL,
  Employee_id INT NOT NULL ,
  Date date DEFAULT NULL,
  PRIMARY KEY (Order_id),
```

```
  FOREIGN KEY (Customer_id) REFERENCES Customer(Customer_id),
  FOREIGN KEY (Employee_id) REFERENCES Employee(Employee_id)
);

CREATE TABLE Payment (
  Payment_id INT AUTO_INCREMENT,
  Payment_type varchar(40) NOT NULL,
  Date date DEFAULT NULL,
  Amount double NOT NULL,
  PayDueDate date DEFAULT NULL,
  PRIMARY KEY (Payment_id)
);

CREATE TABLE Order_detail (
  Receipt_id INT AUTO_INCREMENT,
  Customer_id INT NOT NULL,
  Order_id INT NOT NULL,
  Product_id INT NOT NULL,
  Product_name varchar(40) NOT NULL,
  Date date DEFAULT NULL,
  Quantity double NOT NULL,
  Amount double NOT NULL,
  Payment_id INT NOT NULL,
  FOREIGN KEY (Product_id, Product_name) REFERENCES
Product(Product_id, Product_name),
  FOREIGN KEY (Customer_id) REFERENCES Customer(Address_id),
  FOREIGN KEY (Order_id) REFERENCES _Order(Order_id),
  FOREIGN KEY (Payment_id) REFERENCES Payment(Payment_id),
  PRIMARY KEY (Receipt_id, Payment_id)
);

CREATE TABLE Inventory (
  Inventory_id INT AUTO_INCREMENT,
  Name varchar(40) NOT NULL,
  Location varchar(40) NOT NULL ,
  Supplier_id INT NOT NULL ,
  Product_id INT NOT NULL ,
  Product_name varchar(40) NOT NULL ,
  Quantity double NOT NULL,
  Contact varchar(40) NOT NULL ,
  PRIMARY KEY (Inventory_id),
  FOREIGN KEY (Supplier_id) REFERENCES Supplier(Supplier_id),
  FOREIGN KEY (Product_id, Product_name) REFERENCES
Product(Product_id, Product_name)
);

CREATE TABLE Warehouse (
  Warehouse_id INT AUTO_INCREMENT,
  Name varchar(40) NOT NULL,
  Location varchar(40) NOT NULL ,
  Inventory_id INT NOT NULL ,
```

```
  Employee_id INT NOT NULL ,
  Contact varchar(40) NOT NULL ,
  PRIMARY KEY (Warehouse_id),
  FOREIGN KEY (Employee_id) REFERENCES Employee(Employee_id),
  FOREIGN KEY (Inventory_id) REFERENCES Inventory(Inventory_id)
);

create table roles( Role_name varchar(80), Role_specification
varchar(80));
```

## Importing Data

We started by inserting data into the Adress table in the rows of country, city, state and zip code. We inserted 10 values in each row. To avoid the foreign key problem, we used the set check method which will deactivate the foreign key check when we enter the data and reactivate it after we enter it. We then entered 10 values in each row of the customer table. We then inserted three departments in the departments table with their descriptions and we displayed the output. We also inserted 8 employees in the employees' table and we also added information for 6 entities namely firstname, lastname, department id as foreign key, social security number, phone number and address id . We never faced a foreign key problem because we inserted data correctly in the right order. We then insert information in the products table and its 5 entities namely product name, description, manufacturer and expiration date. We inserted data of the suppliers' details with its entities, We entered 7 values in each entity successfully. We entered values for the order details table which holds data for the details of every product. we just entered the way the customers paid for the products that they bought and we entered 5 values for each entity lastly we entered products in the inventory table, each product has its name, location, quantity, and contact and we added 8 products to the inventory.

**Inserting data error.**


We set the foreign_key_checks to zero before importing data into the tables. The we set foreign_key_check to 1 to maintain the integrity of the tables.

```
insert into Address(Country,street,city,state,zip) values
('Rwanda','45lake','kigali','Kimironko',0000),

('USA','45lake','NY','Poughkeepsie',2435),

('USA','75clark','NY','fishkill',7890),

('Belgium','45AVE','La ville','manic',335),

('USA','45lake','NY','Poughkeepsie',2435),

('USA','45lake','NY','Poughkeepsie',2435),

('USA','45lake','NY','Poughkeepsie',2435),

('USA','45lake','NY','Poughkeepsie',2435),
```

```
('USA','45lake','NY','Poughkeepsie',2435),

('USA','5lake','LA','newtown',1565);

select * from Address;

set foreign_key_checks=0;

insert into customer(First_name,Last_name,phone,email,Address_id)

values

('Jacob','jane',646785893,'bhac@gmail.com',1),

('Job','Mark',646720893,'bcdc@gmail.com',2),

('Bill','Toamso',657867786,'hbah@gmail.com',3),

('Nico','jane',646785893,'bhac@gmail.com',6),

('Ricky','Isheja',646785893,'bhac@gmail.com',8),

('Descartes','Abdilahi',646785893,'bhac@gmail.com',9),

('Saeed','Musoni',646785893,'bhac@gmail.com',10);

set foreign_key_checks=1;

select * from customer;

insert into department(name,description)values('IT','In charge of
IT'),

('HR','In charge of Human Resources'),('Operations','In charge of
all business related Operations');

select * from department;

insert into
employee(First_name,Last_name,Dept_id,SSN,phone,Address_id)
values('John','doe',1,000000222,222111333,9),

('John','doe',1,000000333,111222333,3),

('Jo','doe',2,000000444,444222333,4),

('Jahn','doe',3,000000555,000111222,5),

('ohn','doe',3,000000678,212323434,6),

('Ricky','doe',3,000000777,121212121,7),

('Junior','doe',1,000000888,111000222,8),

('Descartes','doe',2,000000999,909090909,9),
```

```
('Saeed','doe',2,000000181,756876890,2);

select * from employee;

insert into
Product(Product_name,description,manifacturer,Expiration_date)values
('riham','biscuit','axzam',12/3/2021),

('MT dew','Drink','dew',1/3/2021),

('T-shirt','Clothes','Gucci',12/4/2021),

('Hoddie','Clothes','LV',12/3/2021),

('laptop','Tech','HP',13/4/2022),

('Macbook pro','tech','Apple',4/3/2022),

('Books','paperterie ','axzam',12/3/2023);

insert into
Supplier(First_name,Last_name,phone,SSN,email,payment,Quantity,Price
,Address_id,Product_id)

values('John','Doe',832456789,000000111,'jaha@gxc.com','Credit
card',5,5000,1,1),

('John','Doe',832456908,000000111,'jaha@gxc.com','Debit
Card',5,2000,9,5),

('John','Doe',832263876,000000111,'jaha@gxc.com','Credit
card',5,3000,3,1),

('John','Doe',832798725,000000111,'jaha@gxc.com','credit
card',5,4000,2,2),

('John','Doe',832798723,000000111,'jaha@gxc.com','credit
card',5,5000,1,3),

('John','Doe',832798725,000000111,'jaha@gxc.com','credit
card',5,7000,4,5),

('John','Doe',832798722,000000111,'jaha@gxc.com','credit
card',5,1000,5,7);

select * from Supplier;

insert into Orders(Customer_id,Employee_id)values(1,2),(3,4),(1,6);

select * from Orders;

insert into
Payment(Payment_type,Date,Amount,PayDueDate)value('Check',12/3/2021,
2000,24/3/2021),
```

```
('Debit Card',12/6/2021,2000,24/12/2021),

('Check',12/5/2021,6000,24/11/2021),

('Credit Card',12/7/2021,5000,24/10/2021),

('Check',12/8/2021,200,24/9/2021);

select * from payment;

insert into
Inventory(Name,Location,Quantity,Contact)value('biscuits','3rd',2000
,387873),

('Drinks','4th',2000,384573),

('Clothes','1st',2000,383873),

('Tech','5th',2000,387323),

('biscuits','3rd',2000,387873),

('biscuits','3rd',2000,387873),

('biscuits','3rd',2000,387873),

('biscuits','3rd',2000,387873);
```

## Data Manipulation

| Customer_id | First_name | Last_name | phone | email | Address_id |
|---|---|---|---|---|---|
| 1 | Jacob | jane | 646785893 | bhac@gmail.com | 1 |
| 2 | Jacob | jane | 646785893 | bhac@gmail.com | 1 |
| 3 | Job | Mark | 646720893 | bcdc@gmail.com | 2 |
| 4 | Jacob | jane | 646785893 | bhac@gmail.com | 1 |
| 5 | Job | Mark | 646720893 | bcdc@gmail.com | 2 |
| 6 | Bill | Toamso | 657867786 | hbah@gmail.com | 3 |
| 7 | Nico | jane | 646785893 | bhac@gmail.com | 6 |
| 8 | Ricky | Isheja | 646785893 | bhac@gmail.com | 8 |

Alter table Customer add column parents varchar(30);

Alter table customer modify column parents int;

| # | Field | Schema | Table | Type |
|---|---|---|---|---|
| 1 | Customer_id | warehouse | customer | INT |
| 2 | First_name | warehouse | customer | VARCHAR |
| 3 | Last_name | warehouse | customer | VARCHAR |
| 4 | phone | warehouse | customer | INT |
| 5 | email | warehouse | customer | VARCHAR |
| 6 | Address_id | warehouse | customer | INT |
| 7 | parents | warehouse | customer | INT |

Alter table customer drop column parents;

```
insert into employee(First_name,Last_name,Dept_id,SSN,
('John','doe',1,000000333,111222333,3),
('John','doe',2,000000444,444222333,4),
('John','doe',3,000000555,000111222,5),
('John','doe',3,000000678,212323434,6),
('John','doe',3,000000777,121212121,7),
('John','doe',1,000000888,111000222,8),
('John','doe',2,000000999,909090909,9),
('John','doe',2,000000181,756876890,2);
select * from employee;
```

Update employee set First_name="Ricky" where Employee_id=1;

Update employee set First_name="Junior" where Employee_id=2;

Update employee set First_name="Isheja" where Employee_id=3;

Update employee set First_name="Big" where Employee_id=4;

Update employee set First_name="Boys" where Employee_id=5;

update employee set SSN=394039245 where employee_id=2;

SELECT SSN FROM employee WHERE First_name REGEXP 'or$';

update employee set SSN=78907654 WHERE First_name REGEXP 'or$';

| Employee_id | First_name | Last_name | Dept_id | SSN | Phone | Address_id |
|---|---|---|---|---|---|---|
| 1 | Ricky | doe | 1 | 222 | 222111333 | 9 |
| 2 | Junior | doe | 1 | 394039245 | 111222333 | 3 |
| 3 | Isheja | doe | 2 | 444 | 444222333 | 4 |
| 4 | Big | doe | 3 | 555 | 111222 | 5 |
| 5 | Boys | doe | 3 | 678 | 212323434 | 6 |
| 6 | John | doe | 3 | 777 | 121212121 | 7 |
| 7 | John | doe | 1 | 888 | 111000222 | 8 |
| 8 | John | doe | 2 | 999 | 909090909 | 9 |
| 9 | John | doe | 2 | 181 | 756876890 | 2 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

**Optimization Code**

**Table of Address**

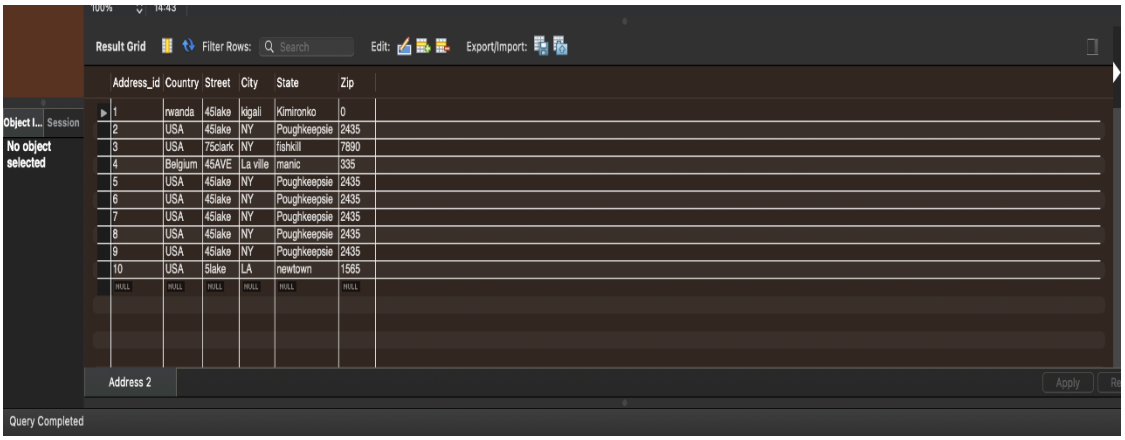
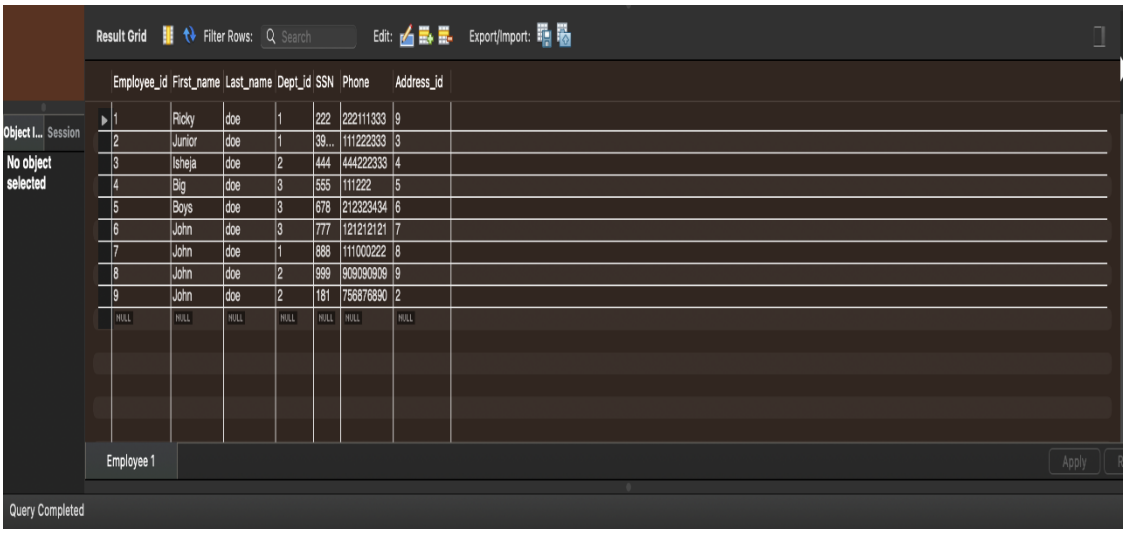



**Table of Employee**



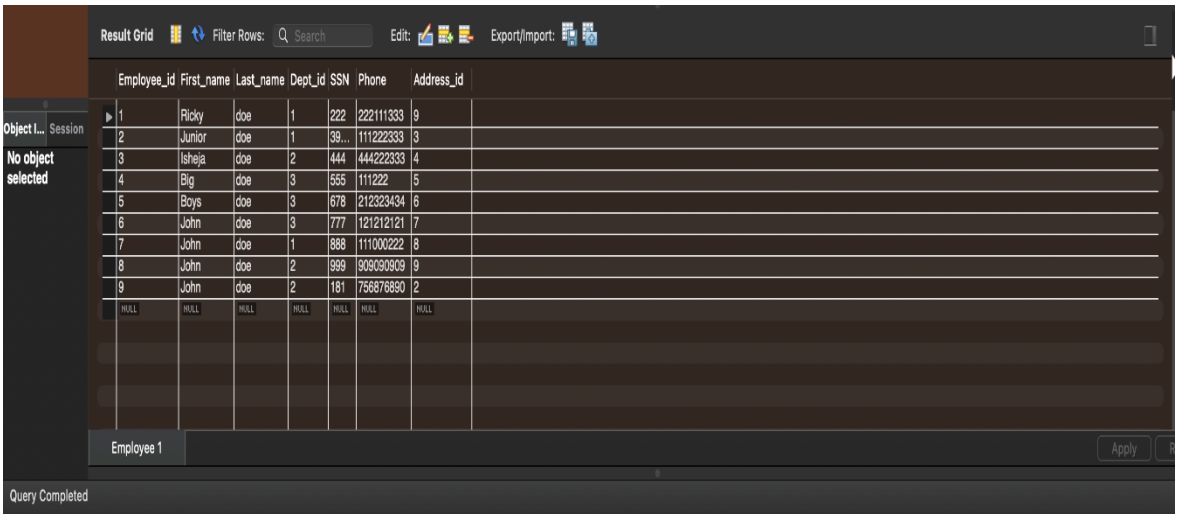

Using join to optimize the above table

**SELECT Employee.EmployeeID, FullName, Department, SSN, Country, State, City**

**FROM Employee**

**INNER JOIN Address ON Employee.EmployeeID = Address.EmployeeId;**

Combined table




# Graphical User Experience

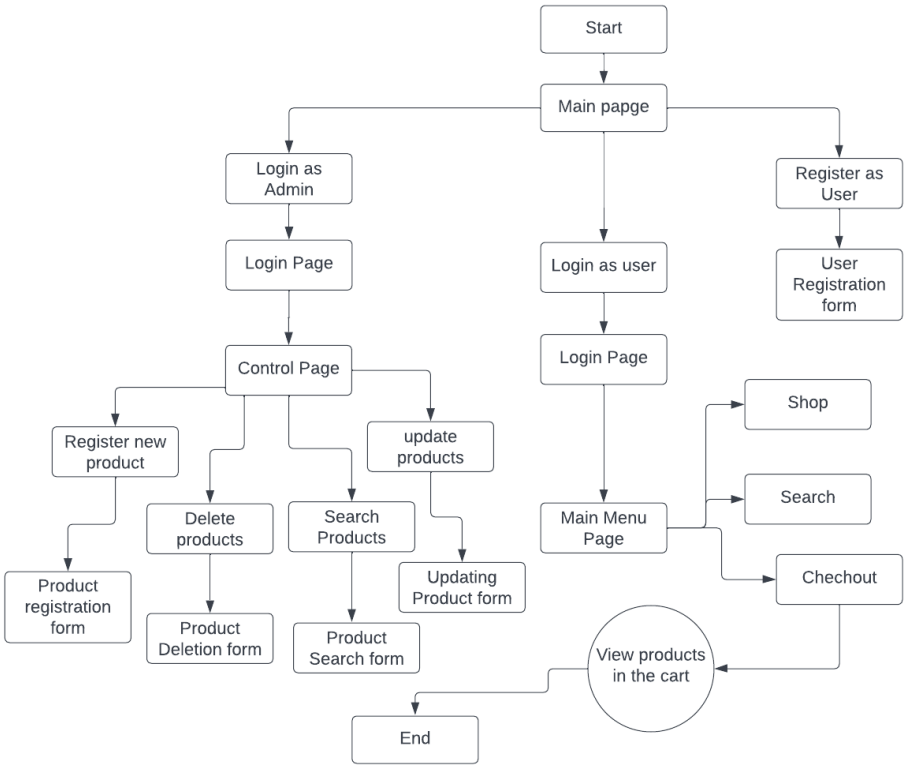



Figure4

## Graphical user experience design

Our Warehouse management system will start by displaying the main page with options of logging in as an admin, user or registering/ signing up. When they login as admin, they shall have the option to manipulate the products table in the database from the front end but since must be done by employees only we gave the privilege to only one user called bobjohn@gmail.com with a password of 123. Any other user who tries to log in will be getting an error box. The admin then after logging in has the option to register products, Update, delete and search products from the products table. We also have the option to log in as a user where the user shall have the option to first register if they are not signed up or log in if they are. After registering and logging in, the user can then look up products to continue shopping.

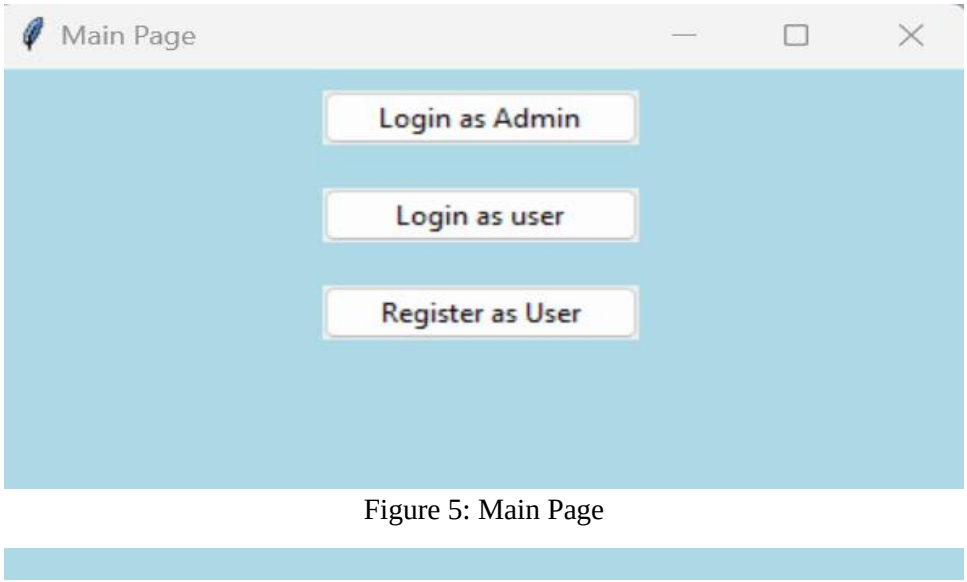



Figure 5: Main Page

Figure 6: Admin Login

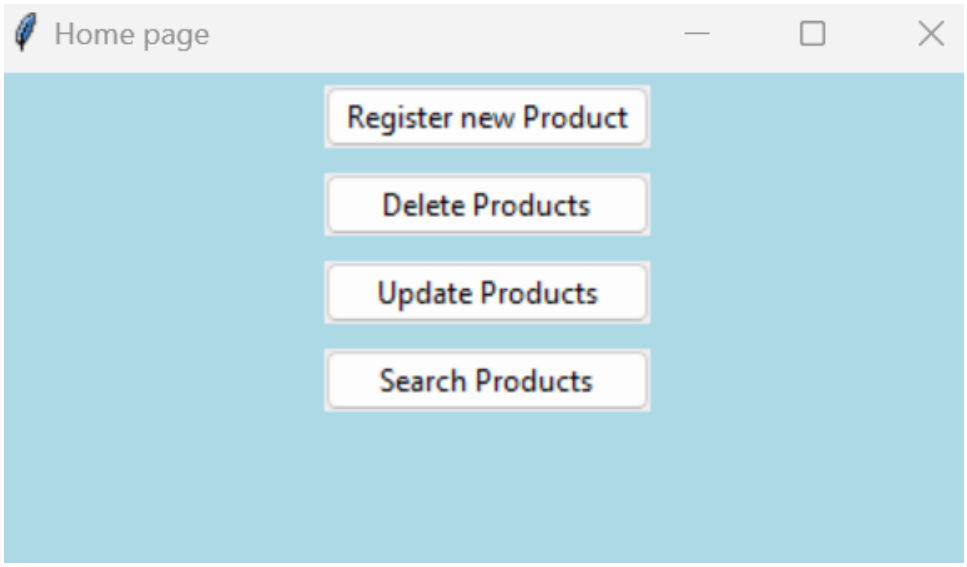


Figure 7:Home Page Administrator

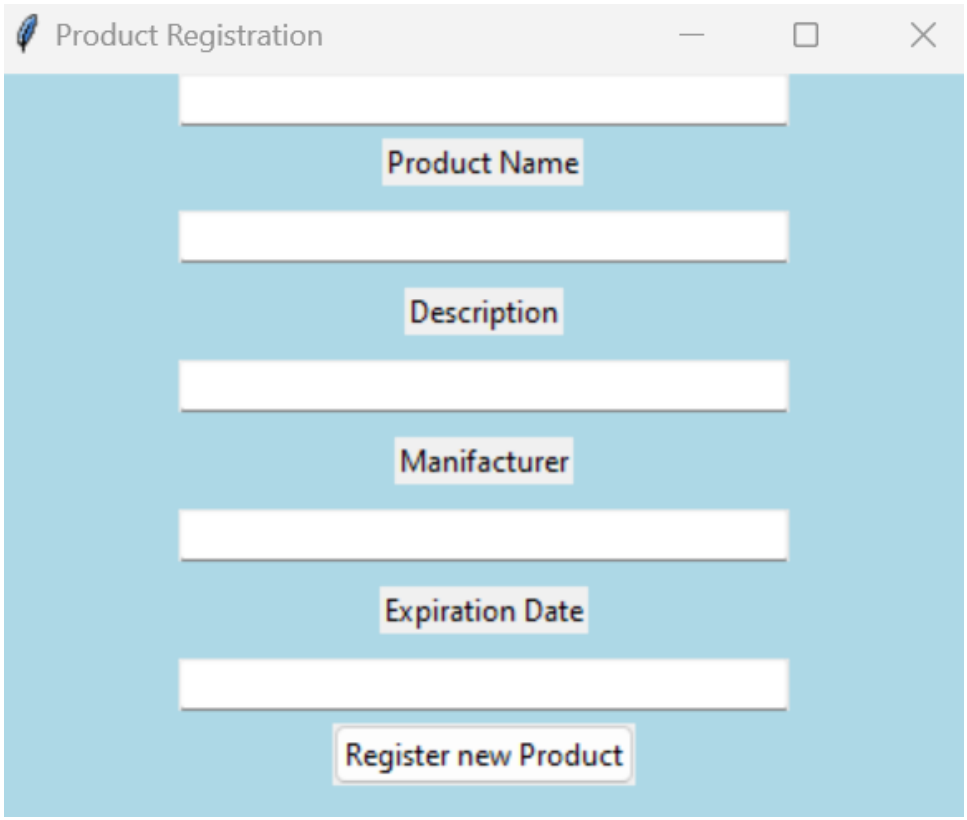



Figure8: Product Registration

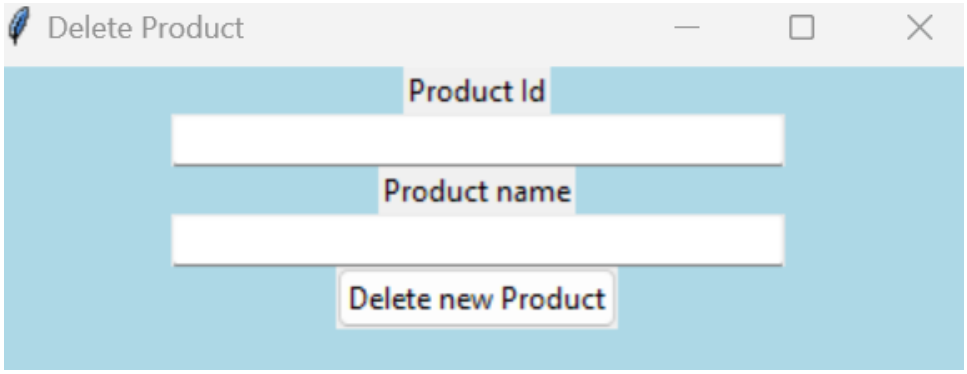



Figure9:Delete Products

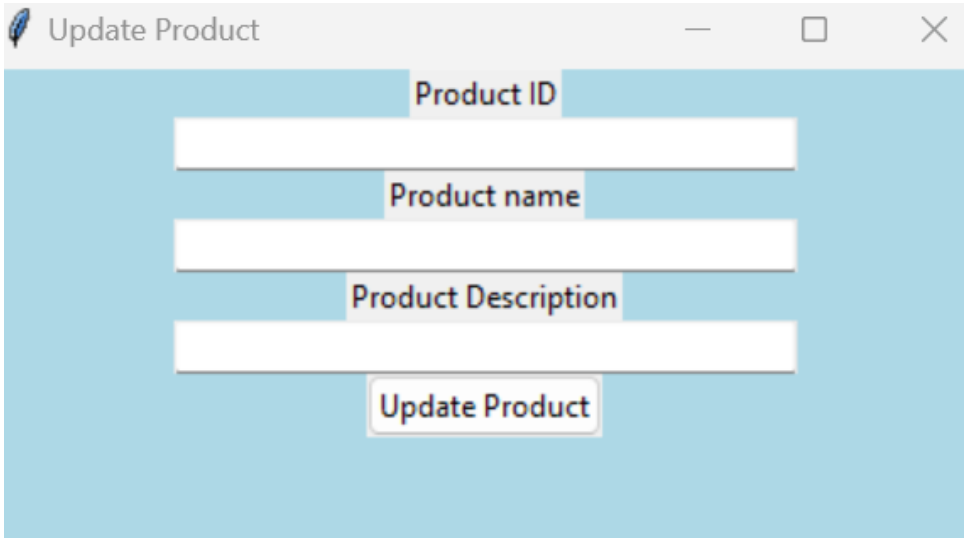



Figure10:Update Products

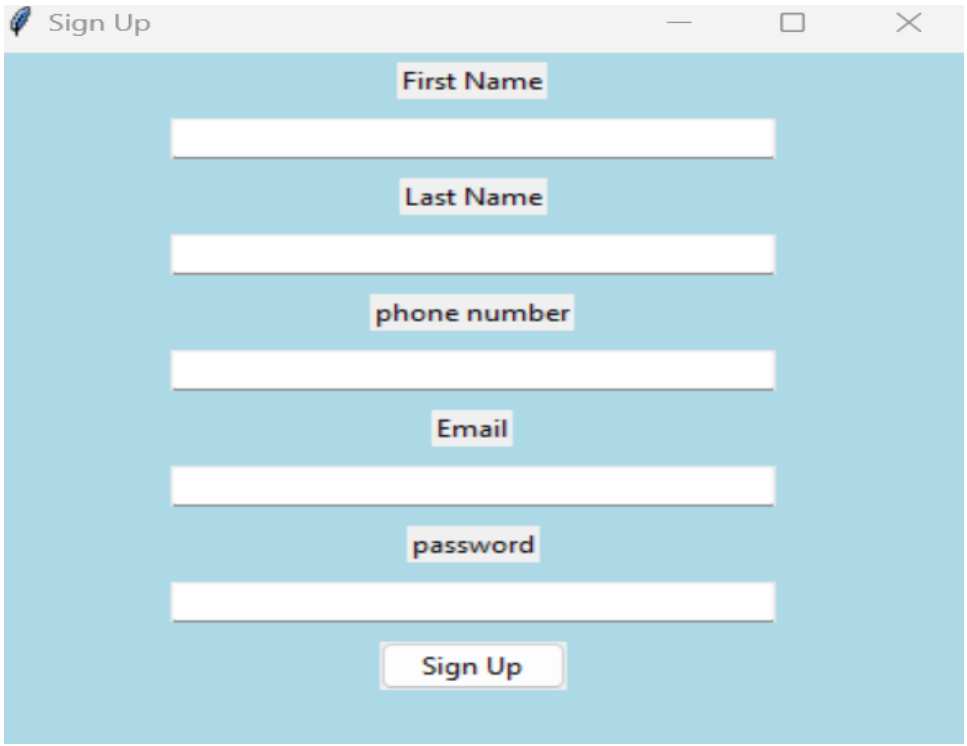



Figure11:Signup Page


Figure 12:User Login

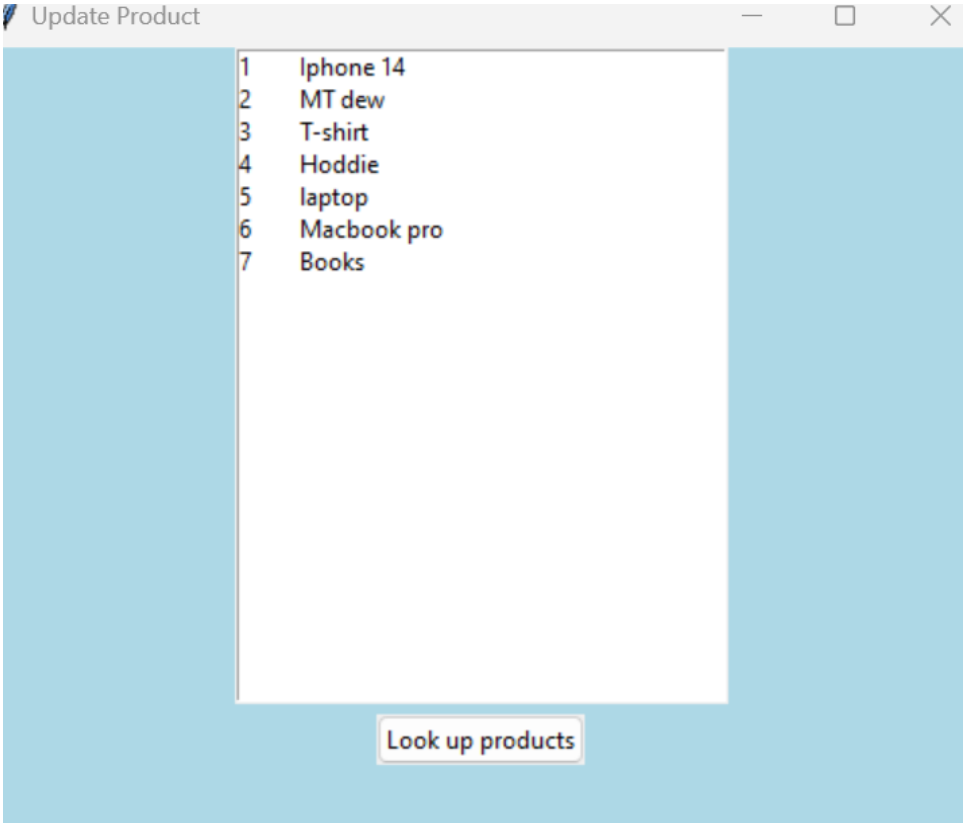



Figure13: User Main Page

## User Interface and Connection Code

```
import tkinter
from tkinter import *
from tkinter.ttk import *
```

```
import mysql.connector
import tkinter.messagebox as messagebox
#from PIL import ImageTK

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="rootroot",
    port="3306",
    database="Warehouse"
)
cursor = mydb.cursor()


main_window= Tk()
main_window.geometry("400x200")
main_window.title("Main Page")
main_window.configure(background='lightblue')
#bg=ImageTK.photo


def Login_page():
    global new_window
    global Uname_Entry
    global Password_Entry
    new_window = Toplevel(main_window)
    new_window.geometry("400x400")
    new_window.title("Login")
    new_window.configure(background='lightblue')
    Username_Label = Label(new_window, text="User_name:")
    Uname_Entry = Entry(new_window, width=40)
    Password_Label = Label(new_window, text="Password:")
    Password_Entry= Entry(new_window, width=40)
    Login_btn = Button(new_window, text="Login", command=loginUS_con)
    Username_Label.pack(padx=5,pady=5)
    Uname_Entry.pack(padx=5,pady=5)
    Password_Label.pack(padx=5,pady=5)
    Password_Entry.pack(padx=5,pady=5)
    Login_btn.pack(padx=5,pady=5)

def loginUS_con():
    usern=Uname_Entry.get()
    passw=Password_Entry.get()
    if usern=="" or passw=="":
        messagebox.showinfo("Error","All fields are Required")
    else:
        try:
            cursor.execute("select * from customer where email=%s and
C_password=%s",(usern,passw))
            op=cursor.fetchone()
            print(op[1])
            if op==None:
                messagebox.showinfo("Error","Invalid Username or Password")
```

```
            else:
                customerview()
        except EXCEPTION as es:
            messagebox.showinfo("Error"f"Error due to:{str(es)}")



def Login_admin():
    global new_window
    global name_Entry
    global Passwor_Entry
    new_window = Toplevel(main_window)
    new_window.geometry("400x400")
    new_window.title("Login")
    new_window.configure(background='lightblue')
    Username_Label = Label(new_window, text="User_name:")
    name_Entry = Entry(new_window, width=40)
    Password_Label = Label(new_window, text="Password:")
    Passwor_Entry= Entry(new_window, width=40)
    Login_btn = Button(new_window, text="Login", command=loginAD_con)
    Username_Label.pack(padx=5,pady=5)
    name_Entry.pack(padx=5,pady=5)
    Password_Label.pack(padx=5,pady=5)
    Passwor_Entry.pack(padx=5,pady=5)
    Login_btn.pack(padx=5,pady=5)

def loginAD_con():
    usern = name_Entry.get()
    passw = Passwor_Entry.get()
    if usern == "" or passw == "":
        messagebox.showinfo("Error", "All fields are Required")
    else:
        try:
            cursor.execute("select * from customer where email=%s and C_password=%s",
(usern, passw))
            op = cursor.fetchone()
            print(op[1])
            if op == None:
                messagebox.showinfo("Error", "Invalid Username or Password")

            else:
                if usern=="bobjohn@gmail.com" and passw=="123":
                    main()
                else:
                    messagebox.showinfo("Error","Admin not registered")
        except EXCEPTION as es:
            messagebox.showinfo("Error"f"Error due to:{str(es)}")


def Sign_up():
    global new_window2
    global Fname_Entry
```

```
    global Lname_textbox
    global Phone_textbox
    global Email_textbox
    global Pass_textbox
    global Fid_Entry
    new_window2= Toplevel(main_window)
    new_window2.geometry("400x400")
    new_window2.title("Sign Up")
    new_window2.configure(background="lightblue")
    Fid_Entry = Entry(new_window2, width=40)
    Fid_Entry.pack()
    Fname_label=Label(new_window2,text="First Name")
    Fname_Entry=Entry(new_window2,width=40)
    Lname_label= Label(new_window2, text="Last Name")
    Lname_textbox = Entry(new_window2, width=40)
    Phone_label = Label(new_window2, text="phone number")
    Phone_textbox = Entry(new_window2, width=40)
    Email_label = Label(new_window2, text="Email")
    Email_textbox = Entry(new_window2, width=40)
    Pass_label= Label(new_window2, text="password")
    Pass_textbox = Entry(new_window2, width=40)
    sign_btn = Button(new_window2, text="Sign Up", command=insert)
    Fname_label.pack(padx=5,pady=5)
    Fname_Entry.pack(padx=5,pady=5)
    Lname_label.pack(padx=5,pady=5)
    Lname_textbox.pack(padx=5,pady=5)
    Phone_label.pack(padx=5,pady=5)
    Phone_textbox.pack(padx=5,pady=5)
    Email_label.pack(padx=5,pady=5)
    Email_textbox.pack(padx=5,pady=5)
    Pass_label.pack(padx=5,pady=5)
    Pass_textbox.pack(padx=5,pady=5)
    sign_btn.pack(padx=5,pady=5)

def insert():
    ID=Fid_Entry.get()
    First_name=Fname_Entry.get()
    Last_name=Lname_textbox.get()
    Phone=Phone_textbox.get()
    Email=Email_textbox.get()
    C_password=Pass_textbox.get()

    if (First_name=="" or Last_name=="" or C_password==""):
        messagebox.showinfo("Insert satus","Missing Fields")
    else:
        cursor.execute("insert into Customer
values('"+ID+"','"+First_name+"','"+Last_name+"','"+Phone+"','"+Email+"','"+C_password+"'
)")
        cursor.execute("commit");
        messagebox.showinfo("Insert status","Inserted Succesfully")
        cursor.close();

def main():
```

```
    new_window3 = Toplevel(main_window)
    new_window3.geometry("400x400")
    new_window3.title("Home page")
    new_window3.configure(background="lightblue")

    register_product=Button(new_window3,text="Register new Product", command=register_p,
width=20)
    delete_product=Button(new_window3,text="Delete Products",command=delete_p, width=20)
    Update_product=Button(new_window3,text="Update Products",command=update_p,width=20)
    search_products=Button(new_window3,text="Search Products",command=search_p,width=20)
    register_product.pack(padx=5,pady=5)
    delete_product.pack(padx=5,pady=5)
    Update_product.pack(padx=5,pady=5)
    search_products.pack(padx=5,pady=5)

def register_p():
    global P_ID
    global pname_box
    global pdesc_box
    global Pmanifactuer_box
    global pExpiration_box
    new_window4= Toplevel(main_window)
    new_window4.geometry("400x400")
    new_window4.title("Product Registration")
    new_window4.configure(background="lightblue")
    P_ID = Entry(new_window4, width=40)
    P_ID.pack()
    p_name=Label(new_window4,text="Product Name")
    pname_box=Entry(new_window4,width=40)
    p_desc=Label(new_window4,text="Description")
    pdesc_box=Entry(new_window4,width=40)
    p_Manifacturer = Label(new_window4, text="Manifacturer")
    Pmanifactuer_box = Entry(new_window4, width=40)
    p_Expiration = Label(new_window4, text="Expiration Date")
    pExpiration_box = Entry(new_window4, width=40)
    p_name.pack(padx=5,pady=5)
    pname_box.pack(padx=5,pady=5)
    p_desc.pack(padx=5,pady=5)
    pdesc_box.pack(padx=5,pady=5)
    p_Manifacturer.pack(padx=5, pady=5)
    Pmanifactuer_box.pack(padx=5, pady=5)
    p_Expiration.pack(padx=5, pady=5)
    pExpiration_box.pack(padx=5, pady=5)
    register_product = Button(new_window4, text="Register new Product",
command=Register_con)
    register_product.pack()
def Register_con():
    ID=P_ID.get()
    Product_name=pname_box.get()
    description=pdesc_box.get()
    manifacturer=Pmanifactuer_box.get()
    Expiration_date=pExpiration_box.get()
```

```
    if (Product_name=="" or description==""):
        messagebox.showinfo("Insert satus","Missing Fields")
    else:
        cursor.execute("insert into Product
values('"+ID+"','"+Product_name+"','"+description+"','"+manifacturer+"','"+Expiration_dat
e+"')")
        cursor.execute("commit");
        messagebox.showinfo("Insert status","Inserted Succesfully")
        cursor.close();

def delete_p():
    global p_id
    new_window5= Toplevel(main_window)
    new_window5.geometry("400x400")
    new_window5.title("Delete Product")
    new_window5.configure(background="lightblue")
    idlabel=Label(new_window5,text="Product Id")
    p_id=Entry(new_window5,width=40)
    name_label=Label(new_window5,text="Product name")
    pname=Entry(new_window5,width=40)
    idlabel.pack()
    p_id.pack()
    name_label.pack()
    pname.pack()
    delete_product = Button(new_window5, text="Delete new Product", command=delete_con)
    delete_product.pack()

def delete_con():
    Product_id=p_id.get()
    if (Product_id == ""):
        messagebox.showinfo("Delete Status", "IDmust be completed")
    else:
        cursor.execute("Delete from product where Product_id='" + Product_id + "'")

        cursor.execute("commit");

        messagebox.showinfo("Delete Status", "Deletion Succesfull")


def update_p():
    global id_box
    global pname
    global pdesc
    new_window6 = Toplevel(main_window)
    new_window6.geometry("400x400")
    new_window6.title("Update Product")
    new_window6.configure(background="lightblue")
    pid=Label(new_window6,text="Product ID")
    id_box=Entry(new_window6,width=40)
    namelabel= Label(new_window6, text="Product name")
    pname= Entry(new_window6, width=40)
    productdesc = Label(new_window6, text="Product Description")
    pdesc = Entry(new_window6, width=40)
```

```
    pid.pack()
    id_box.pack()
    namelabel.pack()
    pname.pack()
    productdesc.pack()
    pdesc.pack()
    Update_product = Button(new_window6, text="Update Product", command=update_con)
    Update_product.pack()

def update_con():
    Product_id=id_box.get()
    Product_name = pname.get()
    description = pdesc.get()

    if (Product_name == "" or description == ""):
        messagebox.showinfo("Insert satus", "Missing Fields")
    else:
        cursor.execute("Update Product set description='" + description + "' where
Product_id='" +Product_id+ "')")
        cursor.execute("commit");
        messagebox.showinfo("Insert status", "Inserted Succesfully")
        cursor.close(); # cant figure out what the problem is

def search_p():
    global listi
    new_window7 = Toplevel(main_window)
    new_window7.geometry("400x400")
    new_window7.title("Update Product")
    new_window7.configure(background="lightblue")

    listi=Listbox(new_window7,width=30,height=20)
    listi.pack()
    Search=Button(new_window7,text='search',command=search_con)
    Search.pack(padx=5,pady=5)
def search_con():
    cursor.execute("Select * from Product")
    Products = cursor.fetchall();

    for product in Products:
        insertdata = str(product[0]) + '         ' + product[1]
        listi.insert(listi.size() + 1, insertdata)
        cursor.close()


def customerview():
    global list
    new_window7 = Toplevel(main_window)
    new_window7.geometry("500x500")
    new_window7.title("Update Product")
    new_window7.configure(background="lightblue")
    list=Listbox(new_window7,height=20, width=40)
    list.pack()
```

```
    search=Button(new_window7,text="Look up products", command=show)
    search.pack(padx=5,pady=5)

def show():
    cursor.execute("Select * from Product")
    Products = cursor.fetchall();

    for product in Products:
        insertdata=str(product[0])+'        '+product[1]
        list.insert(list.size()+1,insertdata)
        cursor.close()

import tkinter
from tkinter import *
from tkinter.ttk import *
import mysql.connector
import tkinter.messagebox as messagebox
#from PIL import ImageTK

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="rootroot",
    port="3306",
    database="Warehouse"
)
cursor = mydb.cursor()


main_window= Tk()
main_window.geometry("400x200")
main_window.title("Main Page")
main_window.configure(background='lightblue')
#bg=ImageTK.photo


def Login_page():
    global new_window
    global Uname_Entry
    global Password_Entry
    new_window = Toplevel(main_window)
    new_window.geometry("400x400")
    new_window.title("Login")
    new_window.configure(background='lightblue')
    Username_Label = Label(new_window, text="User_name:")
    Uname_Entry = Entry(new_window, width=40)
    Password_Label = Label(new_window, text="Password:")
    Password_Entry= Entry(new_window, width=40)
    Login_btn = Button(new_window, text="Login", command=loginUS_con)
    Username_Label.pack(padx=5,pady=5)
    Uname_Entry.pack(padx=5,pady=5)
    Password_Label.pack(padx=5,pady=5)
```

```
    Password_Entry.pack(padx=5,pady=5)
    Login_btn.pack(padx=5,pady=5)

def loginUS_con():
    usern=Uname_Entry.get()
    passw=Password_Entry.get()
    if usern=="" or passw=="":
        messagebox.showinfo("Error","All fields are Required")
    else:
        try:
            cursor.execute("select * from customer where email=%s and
C_password=%s",(usern,passw))
            op=cursor.fetchone()
            print(op[1])
            if op==None:
                messagebox.showinfo("Error","Invalid Username or Password")
            else:
                customerview()
        except EXCEPTION as es:
            messagebox.showinfo("Error"f"Error due to:{str(es)}")



def Login_admin():
    global new_window
    global name_Entry
    global Passwor_Entry
    new_window = Toplevel(main_window)
    new_window.geometry("400x400")
    new_window.title("Login")
    new_window.configure(background='lightblue')
    Username_Label = Label(new_window, text="User_name:")
    name_Entry = Entry(new_window, width=40)
    Password_Label = Label(new_window, text="Password:")
    Passwor_Entry= Entry(new_window, width=40)
    Login_btn = Button(new_window, text="Login", command=loginAD_con)
    Username_Label.pack(padx=5,pady=5)
    name_Entry.pack(padx=5,pady=5)
    Password_Label.pack(padx=5,pady=5)
    Passwor_Entry.pack(padx=5,pady=5)
    Login_btn.pack(padx=5,pady=5)

def loginAD_con():
    usern = name_Entry.get()
    passw = Passwor_Entry.get()
    if usern == "" or passw == "":
        messagebox.showinfo("Error", "All fields are Required")
    else:
        try:
            cursor.execute("select * from customer where email=%s and C_password=%s",
(usern, passw))
            op = cursor.fetchone()
            print(op[1])
```

```
            if op == None:
                messagebox.showinfo("Error", "Invalid Username or Password")

            else:
                if usern=="bobjohn@gmail.com" and passw=="123":
                    main()
                else:
                    messagebox.showinfo("Error","Admin not registered")
        except EXCEPTION as es:
            messagebox.showinfo("Error"f"Error due to:{str(es)}")


def Sign_up():
    global new_window2
    global Fname_Entry
    global Lname_textbox
    global Phone_textbox
    global Email_textbox
    global Pass_textbox
    global Fid_Entry
    new_window2= Toplevel(main_window)
    new_window2.geometry("400x400")
    new_window2.title("Sign Up")
    new_window2.configure(background="lightblue")
    Fid_Entry = Entry(new_window2, width=40)
    Fid_Entry.pack()
    Fname_label=Label(new_window2,text="First Name")
    Fname_Entry=Entry(new_window2,width=40)
    Lname_label= Label(new_window2, text="Last Name")
    Lname_textbox = Entry(new_window2, width=40)
    Phone_label = Label(new_window2, text="phone number")
    Phone_textbox = Entry(new_window2, width=40)
    Email_label = Label(new_window2, text="Email")
    Email_textbox = Entry(new_window2, width=40)
    Pass_label= Label(new_window2, text="password")
    Pass_textbox = Entry(new_window2, width=40)
    sign_btn = Button(new_window2, text="Sign Up", command=insert)
    Fname_label.pack(padx=5,pady=5)
    Fname_Entry.pack(padx=5,pady=5)
    Lname_label.pack(padx=5,pady=5)
    Lname_textbox.pack(padx=5,pady=5)
    Phone_label.pack(padx=5,pady=5)
    Phone_textbox.pack(padx=5,pady=5)
    Email_label.pack(padx=5,pady=5)
    Email_textbox.pack(padx=5,pady=5)
    Pass_label.pack(padx=5,pady=5)
    Pass_textbox.pack(padx=5,pady=5)
    sign_btn.pack(padx=5,pady=5)

def insert():
    ID=Fid_Entry.get()
    First_name=Fname_Entry.get()
    Last_name=Lname_textbox.get()
```

```
    Phone=Phone_textbox.get()
    Email=Email_textbox.get()
    C_password=Pass_textbox.get()

    if (First_name=="" or Last_name=="" or C_password==""):
        messagebox.showinfo("Insert satus","Missing Fields")
    else:
        cursor.execute("insert into Customer
values('"+ID+"','"+First_name+"','"+Last_name+"','"+Phone+"','"+Email+"','"+C_password+"'
)")
        cursor.execute("commit");
        messagebox.showinfo("Insert status","Inserted Succesfully")
        cursor.close();

def main():
    new_window3 = Toplevel(main_window)
    new_window3.geometry("400x400")
    new_window3.title("Home page")
    new_window3.configure(background="lightblue")

    register_product=Button(new_window3,text="Register new Product", command=register_p,
width=20)
    delete_product=Button(new_window3,text="Delete Products",command=delete_p, width=20)
    Update_product=Button(new_window3,text="Update Products",command=update_p,width=20)
    search_products=Button(new_window3,text="Search Products",command=search_p,width=20)
    register_product.pack(padx=5,pady=5)
    delete_product.pack(padx=5,pady=5)
    Update_product.pack(padx=5,pady=5)
    search_products.pack(padx=5,pady=5)

def register_p():
    global P_ID
    global pname_box
    global pdesc_box
    global Pmanifactuer_box
    global pExpiration_box
    new_window4= Toplevel(main_window)
    new_window4.geometry("400x400")
    new_window4.title("Product Registration")
    new_window4.configure(background="lightblue")
    P_ID = Entry(new_window4, width=40)
    P_ID.pack()
    p_name=Label(new_window4,text="Product Name")
    pname_box=Entry(new_window4,width=40)
    p_desc=Label(new_window4,text="Description")
    pdesc_box=Entry(new_window4,width=40)
    p_Manifacturer = Label(new_window4, text="Manifacturer")
    Pmanifactuer_box = Entry(new_window4, width=40)
    p_Expiration = Label(new_window4, text="Expiration Date")
    pExpiration_box = Entry(new_window4, width=40)
    p_name.pack(padx=5,pady=5)
    pname_box.pack(padx=5,pady=5)
    p_desc.pack(padx=5,pady=5)
```

```
    pdesc_box.pack(padx=5,pady=5)
    p_Manifacturer.pack(padx=5, pady=5)
    Pmanifactuer_box.pack(padx=5, pady=5)
    p_Expiration.pack(padx=5, pady=5)
    pExpiration_box.pack(padx=5, pady=5)
    register_product = Button(new_window4, text="Register new Product",
command=Register_con)
    register_product.pack()
def Register_con():
    ID=P_ID.get()
    Product_name=pname_box.get()
    description=pdesc_box.get()
    manifacturer=Pmanifactuer_box.get()
    Expiration_date=pExpiration_box.get()

    if (Product_name=="" or description==""):
        messagebox.showinfo("Insert satus","Missing Fields")
    else:
        cursor.execute("insert into Product
values('"+ID+"','"+Product_name+"','"+description+"','"+manifacturer+"','"+Expiration_dat
e+"')")
        cursor.execute("commit");
        messagebox.showinfo("Insert status","Inserted Succesfully")
        cursor.close();

def delete_p():
    global p_id
    new_window5= Toplevel(main_window)
    new_window5.geometry("400x400")
    new_window5.title("Delete Product")
    new_window5.configure(background="lightblue")
    idlabel=Label(new_window5,text="Product Id")
    p_id=Entry(new_window5,width=40)
    name_label=Label(new_window5,text="Product name")
    pname=Entry(new_window5,width=40)
    idlabel.pack()
    p_id.pack()
    name_label.pack()
    pname.pack()
    delete_product = Button(new_window5, text="Delete new Product", command=delete_con)
    delete_product.pack()

def delete_con():
    Product_id=p_id.get()
    if (Product_id == ""):
        messagebox.showinfo("Delete Status", "IDmust be completed")
    else:
        cursor.execute("Delete from product where Product_id='" + Product_id + "'")

        cursor.execute("commit");

        messagebox.showinfo("Delete Status", "Deletion Succesfull")
```

```
def update_p():
    global id_box
    global pname
    global pdesc
    new_window6 = Toplevel(main_window)
    new_window6.geometry("400x400")
    new_window6.title("Update Product")
    new_window6.configure(background="lightblue")
    pid=Label(new_window6,text="Product ID")
    id_box=Entry(new_window6,width=40)
    namelabel= Label(new_window6, text="Product name")
    pname= Entry(new_window6, width=40)
    productdesc = Label(new_window6, text="Product Description")
    pdesc = Entry(new_window6, width=40)
    pid.pack()
    id_box.pack()
    namelabel.pack()
    pname.pack()
    productdesc.pack()
    pdesc.pack()
    Update_product = Button(new_window6, text="Update Product", command=update_con)
    Update_product.pack()

def update_con():
    Product_id=id_box.get()
    Product_name = pname.get()
    description = pdesc.get()

    if (Product_name == "" or description == ""):
        messagebox.showinfo("Insert satus", "Missing Fields")
    else:
        cursor.execute("Update Product set description='" + description + "' where
Product_id='" +Product_id+ "')")
        cursor.execute("commit");
        messagebox.showinfo("Insert status", "Inserted Succesfully")
        cursor.close(); # cant figure out what the problem is

def search_p():
    global listi
    new_window7 = Toplevel(main_window)
    new_window7.geometry("400x400")
    new_window7.title("Update Product")
    new_window7.configure(background="lightblue")

    listi=Listbox(new_window7,width=30,height=20)
    listi.pack()
    Search=Button(new_window7,text='search',command=search_con)
    Search.pack(padx=5,pady=5)
def search_con():
    cursor.execute("Select * from Product")
    Products = cursor.fetchall();
```

```
    for product in Products:
        insertdata = str(product[0]) + '        ' + product[1]
        listi.insert(listi.size() + 1, insertdata)
        cursor.close()


def customerview():
    global list
    new_window7 = Toplevel(main_window)
    new_window7.geometry("500x500")
    new_window7.title("Update Product")
    new_window7.configure(background="lightblue")
    list=Listbox(new_window7,height=20, width=40)
    list.pack()
    search=Button(new_window7,text="Look up products", command=show)
    search.pack(padx=5,pady=5)

def show():
    cursor.execute("Select * from Product")
    Products = cursor.fetchall();

    for product in Products:
        insertdata=str(product[0])+'        '+product[1]
        list.insert(list.size()+1,insertdata)
        cursor.close()

login=Button(main_window, text="Login as Admin", command=Login_admin, width=20)
login_user=Button(main_window, text="Login as user", command=Login_page, width=20)
sign_up=Button(main_window, text="Register as User", command=Sign_up, width=20)
login.pack(padx=30,pady=10)
login_user.pack(padx=30,pady=10)
sign_up.pack(padx=30,pady=10)
mainloop()
```

## Conclusive Remarks

We are thankful as a group for this whole project that we have been working on for three months and what it taught us to do. We learnt in a depper way how to not only develop the database by following the normalization rules but also, we learnt how to do the documentation that comes with it and the front end. We believe that the combination of all of this learnt will help us in our future Carrers as developers, Systems administrators and entrepreneurs. With much more resources like time and money we would have built an even more helpful system, but we believe that as a group we still have a lot to learn and hopefully we will be building even much bigger and more complex systems in the future.

# Reference

1.https://wdgcorp.com/request-more-information/?gclid=EAIaIQobChMI7q_xi5DM-gIVEr3ICh0r2gvBEAAYBCAAEgI8OfD_BwE

2.https://www.netsuite.com/portal/resource/articles/erp/warehouse-management.shtml

3.https://www.jrcded.com/advantages-and-disadvantages-of-warehousing/

4.GitHub Resources[1]

https://github.com/The-Big-Boys-Inc/Warehouse-Management-System.git

5.ER Diagram

https://lucid.app/lucidchart/0cd5589d-adbc-4fc7-8cde-2a64a2257a23/edit?viewport_loc=45%2C179%2C1897%2C1057%2C0_0&invitationId=inv_89f1f654-3612-447c-b252-7a72f98b60e0#

[1]https://www.quora.com/What-are-the-pros-and-cons-of-Apple

CMPT308N-200_Project Progress Report Phase #6
_The Big Boyz Team