Inventory Management System

Submitted in partial fulfillment of the requirements

of the degree

BACHELOR OF ENGINEERING IN

INFORMATION TECHNOLOGY

By

Aditya Konda - 21

Anmol Gupta - 12

Sohan Bhandare - 02

Arpit Mishra - 25

Supervisor

Prof. Aarti Abhyankar

Department of Information Technology

K C College of Engineering and Management Studies and Research

Mith Bunder Road, Near Hume Pipe, Kopri, Thane (East)

University of Mumbai

(AY 2022-23)

CERTIFICATE

This is to certify that the Mini Project entitled **“ Inventory Management System ”** is a bonafide work of **Aditya Konda - 21, Anmol Gupta - 12, Sohan Bhandare - 02, ArpitKumar Mishra - 25** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Information Technology” .**

(Prof. Aarti Abhyankar)
Supervisor

(Prof._____)
Head of Department

(Prof._____)
Principal

Mini Project Approval

This Mini Project entitled " Inventory Management System**"** by **Aditya Konda - 21, Anmol Gupta - 12, Sohan Bhandare - 02, ArpitKumar Mishra - 25** is approved for the degree of **Bachelor of Engineering** in **Information Technology.**

Examiners

1……
(Internal Examiner Name & Sign)

2……
(External Examiner name & Sign)

Date:

Place:

Contents

1.Introduction

1.1 Introduction:

An inventory management system is a software application designed to help businesses manage and track their inventory levels, sales, and product information. An effective inventory management system can improve the accuracy of inventory data, reduce costs, and streamline business operations.

Python is a popular programming language that is widely used for web development, data analysis, and automation. XAMPP is a free and open-source cross-platform web server solution that includes Apache, MySQL, PHP, and Perl.

The purpose of this project is to develop an inventory management system using Python and XAMPP that can handle inventory data, including product details, stock levels, and sales information. The system will also allow for the creation of reports and analysis of the inventory data.

1.2 Motivation:

1. Efficiency: An inventory management system can help to streamline and automate inventory management processes, making them more efficient and reducing the risk of human error.

2. Cost Reduction: An inventory management system can help to reduce the costs associated with inventory management, such as excess inventory, stockouts, and overstocking.

3. Improved Accuracy: An inventory management system can help to improve the accuracy of inventory data, which is essential for making informed business decisions.

4. Scalability: An inventory management system can be designed to scale as the organization grows, allowing it to manage larger amounts of inventory and sales data.

5. Better Customer Service: An inventory management system can help to improve customer service by ensuring that the organization has the right products in stock, and can deliver them quickly and accurately.

1.3 Problem Statement & Objectives:

Many small businesses struggle with keeping track of their product inventory and sales data, which can lead to lost sales, excess inventory, and inefficient operations. There is a need for a simple, user-friendly inventory management system that can help businesses track their inventory levels, sales data, and generate reports to inform decision-making. The proposed inventory management system will address this problem by providing a centralized database for storing and managing product data, a user interface for adding and updating inventory information, and a reporting module for generating sales reports and forecasting inventory needs. This system will improve the accuracy and efficiency of inventory management for small businesses and ultimately contribute to their overall success.

Objective:-

1. To create a centralized database for storing and managing product inventory data, including product information, quantities, and prices.
2. To develop a user interface that allows users to easily add, update, and view product inventory data.
3. To implement a reporting module that generates sales reports and forecasts inventory needs based on historical data.
4. To improve the accuracy and efficiency of inventory management by reducing manual data entry and minimizing errors.
5. To reduce excess inventory and improve cash flow by providing real-time visibility into inventory levels and sales data.
6. To enhance decision-making by providing actionable insights through reports and analytics.
7. To increase customer satisfaction by ensuring that products are in stock and available for purchase.
8. To enable scalability and growth by providing a flexible, customizable system that can adapt to changing business needs.

These objectives provide a clear direction for the project and set specific goals for measuring success. By achieving these objectives, the inventory management system will improve the accuracy and efficiency of inventory management, reduce excess inventory, enhance decision-making, and ultimately contribute to the success of the business.

2. Literature Survey

2.1 Survey of Existing System:

An inventory management system is a software application that helps businesses keep track of their inventory levels, orders, sales, and deliveries. It enables organizations to optimize their inventory levels, reduce wastage, increase efficiency, and improve customer service.
There are various features that an inventory management system may include, such as:

1. Barcode scanning: This feature allows users to scan barcodes of products to quickly retrieve information about the product, including its inventory level.
2. Order management: This feature enables users to manage orders, including creating, editing, and fulfilling orders.
3. Real-time inventory tracking: This feature provides real-time updates on inventory levels, allowing users to see when stock is running low or when there is excess inventory.
4. Automated reordering: This feature automates the process of reordering stock when inventory levels fall below a certain threshold.
5. Reporting: This feature provides insights into inventory levels, sales, and other metrics to help businesses make data-driven decisions.

When selecting an inventory management system, businesses should consider factors such as their specific needs, budget, scalability, ease of use, and support.

2.2 Limitation Existing system or Research Gap :-

There are several limitations of existing inventory management systems, and potential research gaps that could be addressed to improve the efficacy of such systems. Some of these include:

1. Limited integration with other systems: Many existing inventory management systems may not be fully integrated with other key systems used by a business, such as accounting or purchasing systems, which can lead to data silos and inefficiencies.
2. Manual data entry: Some inventory management systems still rely heavily on manual data entry, which can be time-consuming, error-prone, and limit the ability to automate processes.
3. Lack of predictive capabilities: While many inventory management systems provide real-time data on inventory levels, they may not have robust predictive capabilities to help businesses anticipate demand or forecast future inventory needs.
4. Inflexibility: Some inventory management systems may be inflexible in terms of customizability, limiting their ability to adapt to unique business needs and processes.
5. Cost: Many inventory management systems can be costly, particularly for small businesses, which can limit their accessibility.

To address these limitations and research gaps, future developments in inventory management systems could focus on enhancing integration with other key systems, increasing automation and predictive capabilities, providing greater customizability, and improving accessibility for small businesses through more affordable pricing models. Additionally, further research could explore the impact of these systems on supply chain performance and competitiveness.

2.3 Mini Project Plan & Contribution :

Project Plan for an Inventory Management System:

1. Identify and list all the items that need to be tracked and managed.
2. Choose an inventory management system that meets the requirements of the business.
3. Configure the system to track all items, including their current stock levels, reorder points, and supplier information.
4. Set up automated alerts and notifications to alert staff when stock levels fall below a certain threshold.
5. Develop a process for receiving and recording inventory shipments, as well as tracking outgoing orders.
6. Train staff on how to use the inventory management system and ensure that they understand the importance of accurate record-keeping.
7. Perform regular inventory audits to ensure that the system is working correctly and that stock levels are accurate.
8. Use the data generated by the inventory management system to inform purchasing decisions and optimize inventory levels.
9. Continuously review and improve the inventory management system to ensure that it is meeting the needs of the business.

Contribution of an Inventory Management System:

1. Better inventory control: An inventory management system can help businesses keep track of their stock levels, reorder points, and lead times. This information can be used to optimize inventory levels and reduce the risk of stockouts or overstocking.
2. Increased efficiency: By automating the inventory management process, businesses can reduce the time and effort required to manage their inventory. This can free up staff to focus on other tasks

and improve overall efficiency.
3. Improved accuracy: Manual inventory management processes are prone to errors and inaccuracies. An inventory management system can help reduce these errors and ensure that stock levels are accurate.
4. Cost savings: By optimizing inventory levels and reducing the risk of stockouts or overstocking, businesses can save money on inventory carrying costs and lost sales.
5. Better decision-making: The data generated by an inventory management system can be used to inform purchasing decisions and optimize inventory levels. This can help businesses make better-informed decisions and improve their bottom line.

3. Proposed System :

3.1. Introduction :

An Inventory Management System is a tool that can help businesses keep track of their inventory levels, orders, and sales. It is an essential component of any successful supply chain management system. Here are some features that could be included in an effective Inventory Management System:

1. Inventory tracking: The system should be able to track the number of items in stock, their location, and their movement in and out of the inventory. This tracking should be done in real-time, so the inventory levels are always up-to-date.
2. Order management: The system should be able to manage orders, including receiving orders, processing orders, and shipping orders. It should also be able to handle returns and refunds.
3. Automatic reordering: The system should have a feature that automatically generates orders to restock inventory when levels fall below a certain threshold. This feature helps ensure that the business never runs out of stock of a particular item.
4. Reporting: The system should be able to generate reports that show inventory levels, sales trends, and other key metrics. This reporting helps businesses make informed decisions about inventory management.
5. Integration with other systems: The system should be able to integrate with other systems, such as a point of sale system or a shipping system. This integration helps streamline the entire supply chain process.
6. Barcode scanning: The system should be able to use barcode scanning to quickly and accurately input data into the system. This feature can save time and reduce errors in the inventory management process.

7. User permissions: The system should allow for different levels of access and permissions based on the user. This feature helps ensure that only authorized personnel can access and make changes to the inventory data.

3.2 **Architecture/ Framework** :

There are several popular architecture and framework options for developing an Inventory Management System. Here are a few examples:

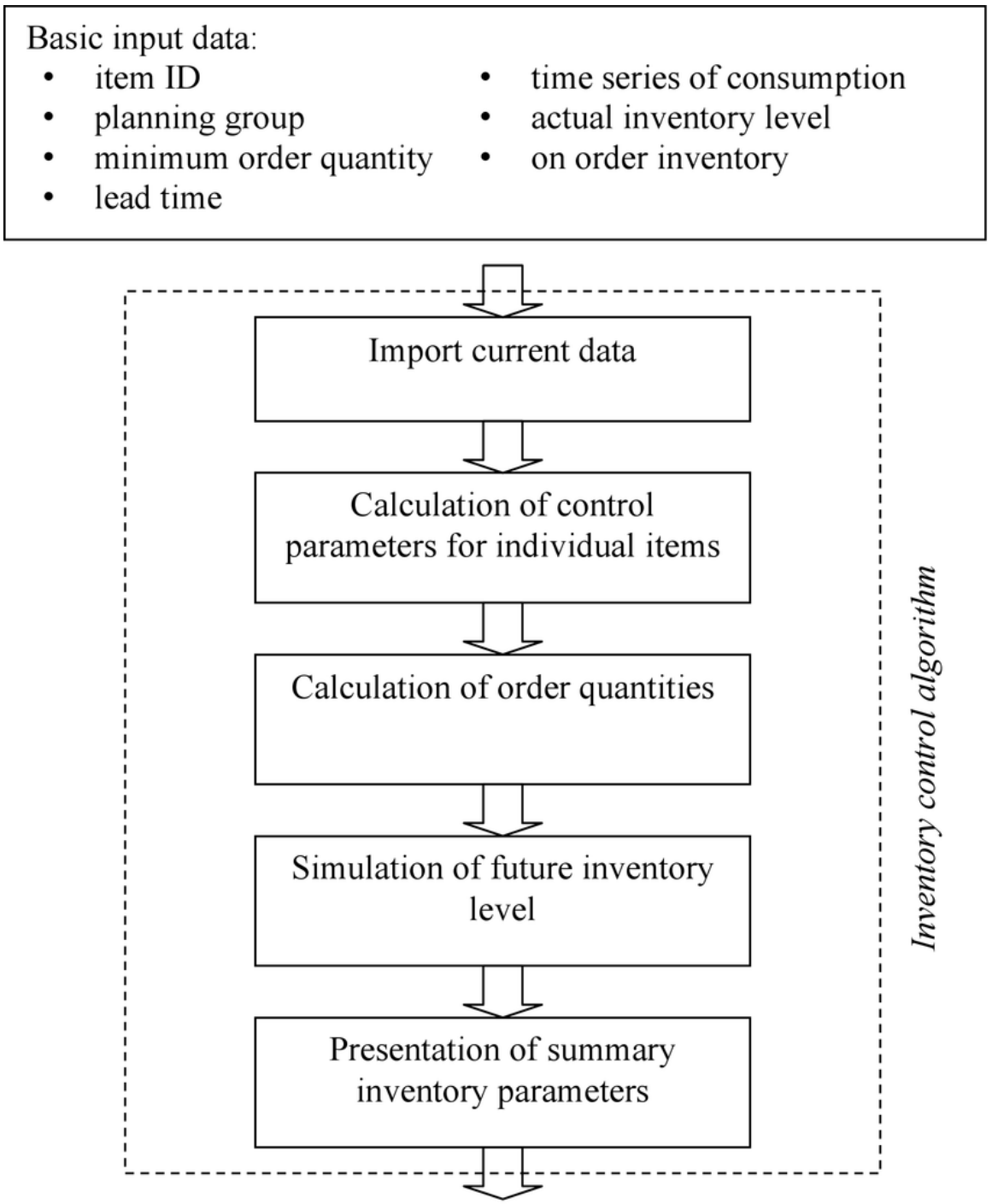
1. Database: The database is the backbone of the system and stores all inventory-related data, such as product information, stock levels, and transaction history. The database can be built using a relational database management system (RDBMS) such as MySQL or PostgreSQL.
2. User Interface: The user interface is the front-end of the system and provides a way for users to interact with the database. The user interface can be built using a web application framework such as Django or Flask or a desktop application framework such as Qt or Tkinter.
3. Business Logic: The business logic of the system defines the rules and workflows that govern inventory management processes, such as ordering, receiving, stocking, and selling products. The business logic can be implemented using a programming language such as Python or Java.
4. Reporting and Analytics: The reporting and analytics component of the system provides insights into inventory performance and helps identify trends and areas for improvement. Reporting and analytics can be implemented using tools such as Tableau, Power BI, or Matplotlib.
5. Integration with other systems: The inventory management system may need to integrate with other systems, such as point-of-sale systems, accounting systems, or supplier portals. Integration can be achieved through application programming interfaces (APIs) or other methods.
6. Security: Security is a critical component of any system, and an

inventory management system must be designed with security in mind. Security features may include authentication, access control, data encryption, and vulnerability management.

By designing an inventory management system with these components and principles in mind, organizations can create a system that is reliable, efficient, and scalable, and that meets the specific needs of the organization.

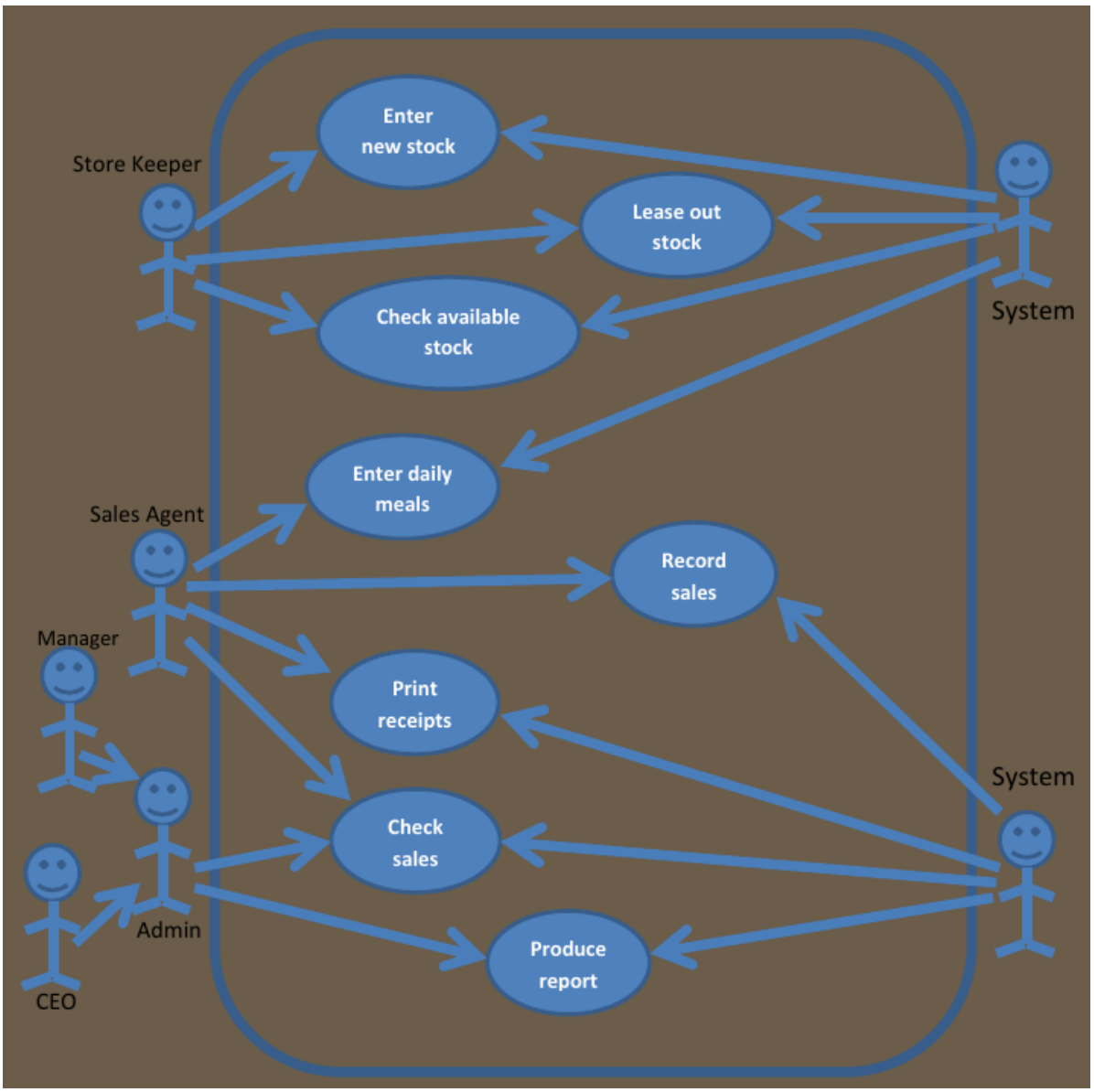
3.3 Algorithm and Process Design :

Algorithm Design:

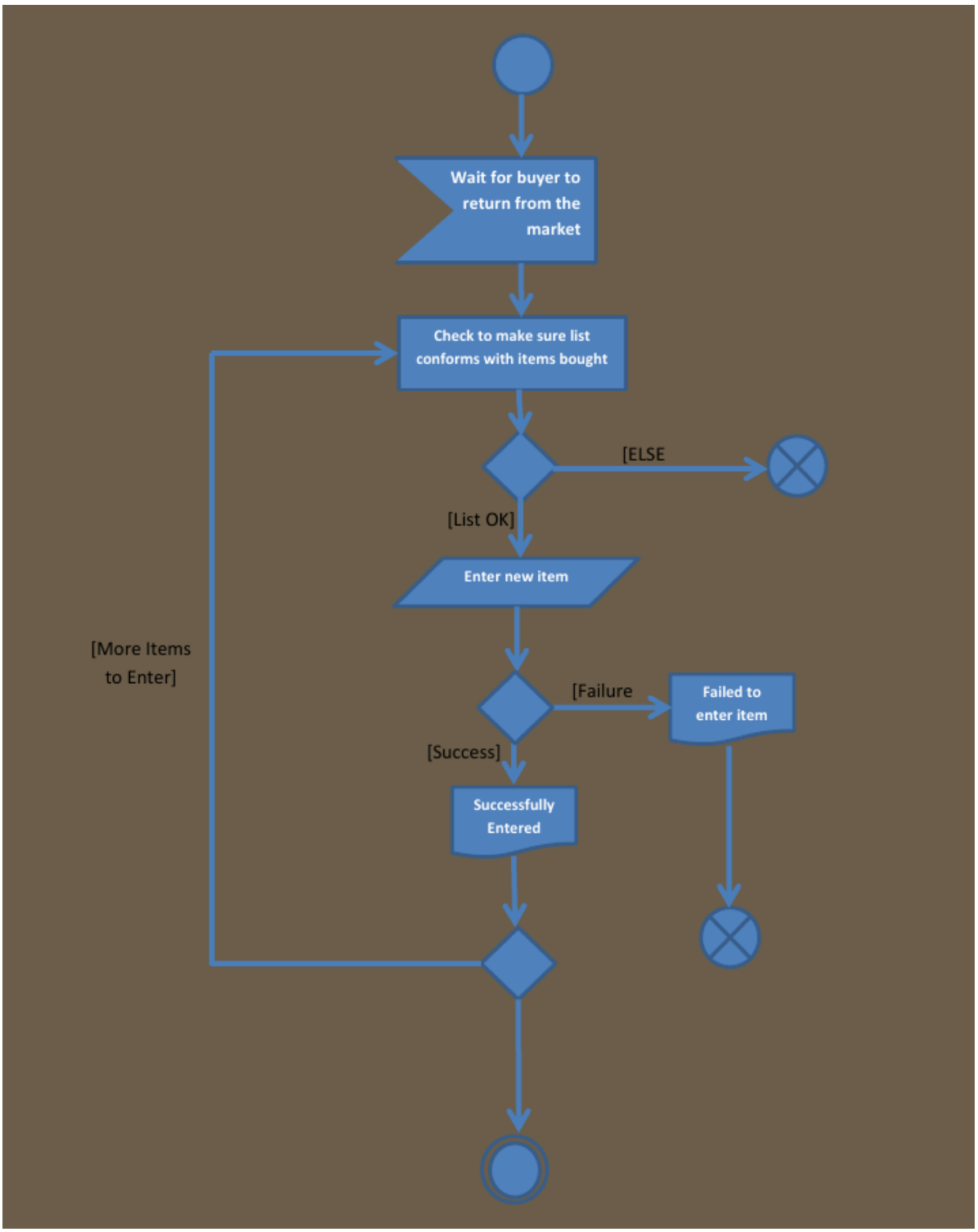


Process Design:-

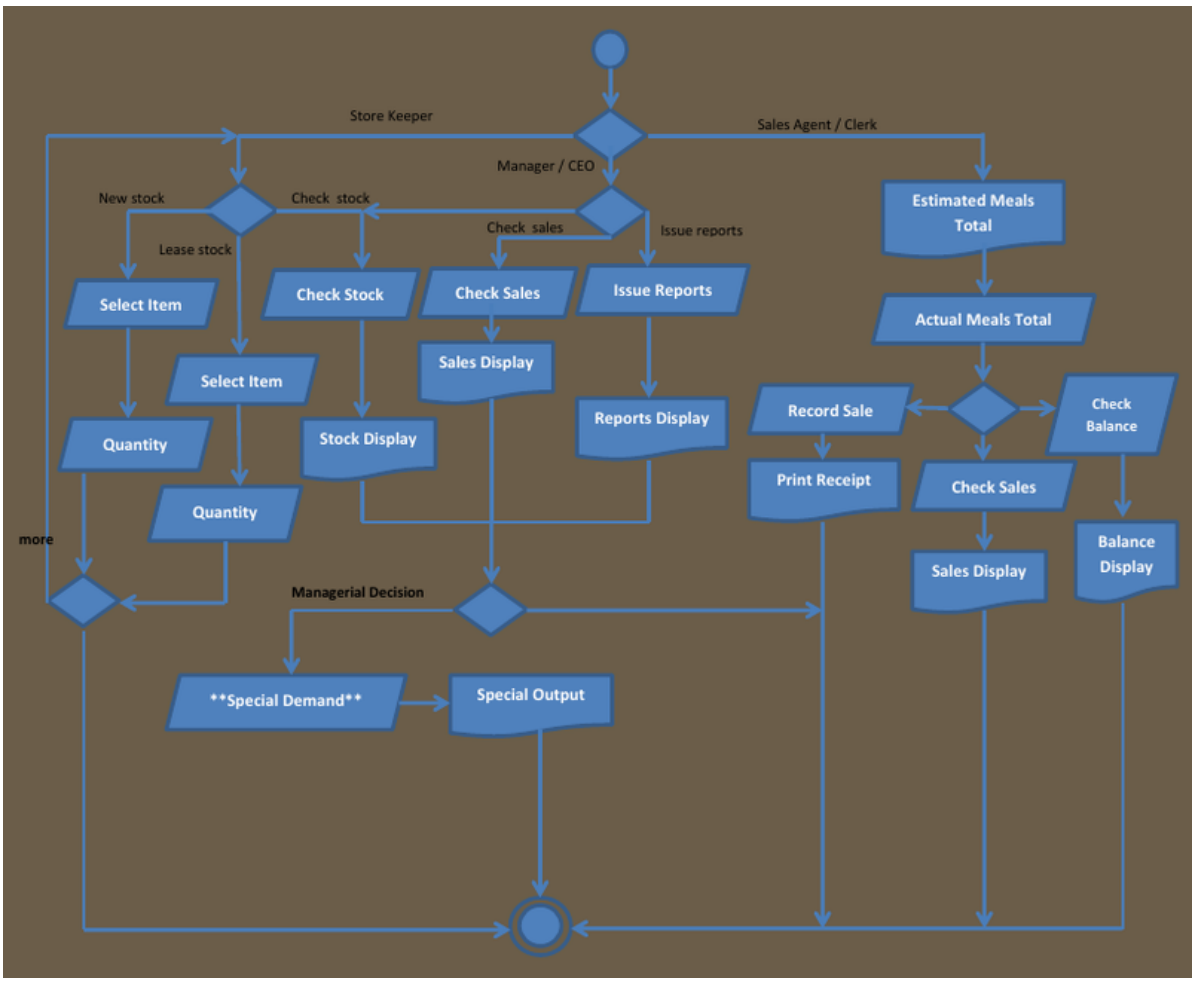
USE CASE DIAGRAM



Process of entering new stock



COMPLETE ACTIVITY DIAGRAM



3.4 Details of Hardware & Software:

Hardware Requirements: The hardware requirements for an Inventory Management System will depend on the scale of the system and the number of users accessing it. Here are the recommended hardware requirements:

1. Server:
- Processor: Intel Xeon or AMD Opteron
- RAM: 16 GB or more
- Storage: 500 GB or more
- Network Interface: 1 Gbps Ethernet

2. Client:
- Processor: Intel Core i5 or AMD Ryzen 5
- RAM: 8 GB or more
- Storage: 256 GB or more
- Display: 1080p or higher resolution
- Network Interface: 1 Gbps Ethernet or Wi-Fi

Software Requirements:-The software requirements for an Inventory Management System will depend on the architecture and framework selected for the project. Here are the recommended software requirements:

1. Server:
- Operating System: Linux (e.g., Ubuntu, CentOS) or Windows Server
- Database Management System: MySQL, XAMPP, PostgreSQL, or Oracle
- Server Framework: Node.js, Spring Framework, or Django

2. Client:
- Operating System: Windows, MacOS, or Linux
- Web Browser: Google Chrome, Mozilla Firefox, or Safari
- User Interface Framework: React, Angular, or Vue.js

3. Additional software requirements:
- Version Control System: Git
- Integrated Development Environment (IDE): Visual Studio Code, IntelliJ IDEA, PyCharm or Eclipse

3.4 Experiment and Results :

Experiment:

Source Code:-

File No.1- main.py

```
#import all the modules
from tkinter import *
from tkinter import messagebox
from PIL import ImageTk, Image
import mysql.connector
from mysql.connector import Error
import tkinter.messagebox
import datetime
import math

date=datetime.datetime.now().date()
#temporary list like sessions
products_list=[]
product_price=[]
product_quantity=[]
product_id=[]
r = []

class Application():
    def __init__(self,master,*args,**kwargs):

        self.master=master



        self.left=Frame(master,width=700,height=768,bg='turquoise')
        self.left.pack(side=LEFT)

        self.right = Frame(master, width=666, height=768,
bg='turquoise')
        self.right.pack(side=RIGHT)


        #components
        self.heading=Label(self.left,text="WooCommerce Inventory
```

```
Management",font=('arial 20 bold'),fg='black',bg='turquoise')
        self.heading.place(x=0,y=0)

        self.date_l=Label(self.right,text="Today's Date:
"+str(date),font=('arial 10 bold'),bg='turquoise',fg='black')
        self.date_l.place(x=0,y=0)

        #table
invoice=============================================================
        self.tproduct=Label(self.right,text="Products",font=('arial 12
bold'),bg='turquoise',fg='black')
        self.tproduct.place(x=0,y=60)

        self.tquantity = Label(self.right, text="Quantity",
font=('arial 12 bold'), bg='turquoise', fg='black')
        self.tquantity.place(x=300, y=60)

        self.tamount = Label(self.right, text="Amount", font=('arial
12 bold'), bg='turquoise', fg='black')
        self.tamount.place(x=500, y=60)

        #enter stuff
        self.enterid=Label(self.left,text="Enter Product's
ID",font=('arial 12 bold'),fg='black',bg='turquoise')
        self.enterid.place(x=0,y=80)



        self.enteride=Entry(self.left,width=25,font=('arial 12
bold'),bg='lightblue')
        self.enteride.place(x=220,y=80)
        self.enteride.focus()

        #button

self.search_btn=Button(self.left,text="Search",width=22,height=2,bg='o
range',command=self.ajax)
        self.search_btn.place(x=380,y=120)
        #fill it later by the fuction ajax

        self.productname=Label(self.left,text="",font=('arial 27
bold'),bg='black',fg='steelblue')
        self.productname.place(x=0,y=250)

        self.pprice = Label(self.left, text="", font=('arial 27
```

```
bold'), bg='black', fg='steelblue')
        self.pprice.place(x=0, y=290)

        #total label
        self.total_l=Label(self.right,text="",font=('arial 40
bold'),bg='lightblue',fg='black')
        self.total_l.place(x=0,y=600)
    def ajax(self,*args,**kwargs):
        self.conn = mysql.connector.connect(host='localhost',
                                            database='inventory_system',
                                            user='root',
                                            password='')
        self.get_id=self.enteride.get()
        #get the product info with that id and fill i the labels above
        self.mycursor = self.conn.cursor()
        self.mycursor.execute("SELECT * FROM inventory WHERE id=
%s",[self.get_id])
        self.pc = self.mycursor.fetchall()
        if self.pc:
          for self.r in self.pc:
            self.get_id=self.r[0]
            self.get_name=self.r[1]
            self.get_price=self.r[3]
            self.get_stock=self.r[2]
          self.productname.configure(text="Product's Name: "
+str(self.get_name),fg='black',bg='turquoise')
          self.pprice.configure(text="Price:Rs.
"+str(self.get_price),fg='black',bg='turquoise')


        #craete the quantity and the discount label
          self.quantityl=Label(self.left,text="Enter
Quantity",font=('arial 12 bold'),fg='black',bg='turquoise')
          self.quantityl.place(x=0,y=370)

          self.quantity_e=Entry(self.left,width=25,font=('arial 12
bold'),bg='lightblue')
          self.quantity_e.place(x=190,y=370)
          self.quantity_e.focus()

        #discount
          self.discount_l = Label(self.left, text="Enter Discount ",
font=('arial 12 bold'),fg='black',bg='turquoise')
          self.discount_l.place(x=0, y=410)
```

```
          self.discount_e = Entry(self.left, width=25, font=('arial 12
bold'), bg='lightblue')
          self.discount_e.place(x=190, y=410)
          self.discount_e.insert(END,0)


        #add to cart button
          self.add_to_cart_btn = Button(self.left, text="Add to Cart",
width=22, height=2, bg='orange',command=self.add_to_cart)
          self.add_to_cart_btn.place(x=350, y=450)

        #genrate bill and change
          self.change_l=Label(self.left,text="Given
Amount",font=('arial 12 bold'),fg='black',bg='turquoise')
          self.change_l.place(x=0,y=550)

          self.change_e=Entry(self.left,width=25,font=('arial 12
bold'),bg='lightblue')
          self.change_e.place(x=190,y=550)

          self.change_btn= Button(self.left, text="Calculate Change",
width=22, height=2, bg='orange',command=self.change_func)
          self.change_btn.place(x=350, y=590)

        #geneerate bill button
          self.bill_btn = Button(self.left, text="Generate Bill",
width=100, height=2, bg='red',fg='black',command=self.generate_bill)
          self.bill_btn.place(x=0, y=640)
        else:
            messagebox.showinfo("Succesfully Done Everything
Smoothly", "Thank you for your recent purchase. We are honored to gain
you as a customer and hope to serve you for a long time. We just want
to drop a quick note to express our genuine gratitude. Your purchase
allows us at WooCommerce Inventory Management to continue to do what
we love and provide you with quality products.")

    def add_to_cart(self,*args,**kwargs):
        self.quantity_value=int(self.quantity_e.get())

        if  self .quantity_value >int(self.get_stock):
            tkinter.messagebox.showinfo("Error","We Don't have that
much products in stock.")
```

```
        else:
            #calculate the price first
            self.final_price=(float(self.quantity_value) *
float(self.get_price))-(float(self.discount_e.get()))
            products_list.append(self.get_name)
            product_price.append(self.final_price)
            product_quantity.append(self.quantity_value)
            product_id.append(self.get_id)

            self.x_index=0
            self.y_index=100
            self.counter=0
            for self.p in products_list:

self.tempname=Label(self.right,text=str(products_list[self.counter]),f
ont=('arial 12 bold'),bg='turquoise',fg='black')
                self.tempname.place(x=0,y=self.y_index)
                self.tempqt = Label(self.right,
text=str(product_quantity[self.counter]), font=('arial 12 bold'),
bg='turquoise', fg='black')
                self.tempqt.place(x=300, y=self.y_index)
                self.tempprice = Label(self.right,
text=str(product_price[self.counter]), font=('arial 12 bold'),
bg='turquoise', fg='black')
                self.tempprice.place(x=500, y=self.y_index)

                self.y_index+=40
                self.counter+=1


                #total confugure
                self.total_l.configure(text="Total : Rs.
"+str(sum(product_price)),bg='turquoise',fg='black')
                #delete
                self.quantity_e.place_forget()
                self.discount_l.place_forget()
                self.discount_e.place_forget()
                self.productname.configure(text="")
                self.pprice.configure(text="")
                self.add_to_cart_btn.destroy()
                #autofocus to the enter id
                self.enteride.focus()
                self.quantityl.focus()
                self.enteride.delete(0,END)
```

```
    def change_func(self,*args,**kwargs):
        self.amount_given=float(self.change_e.get())
        self.our_total=float(sum(product_price))

        self.to_give=self.amount_given-self.our_total

        #label change
        self.c_amount=Label(self.left,text="Change: Rs.
"+str(self.to_give),font=('arial 12 bold'),fg='red',bg='turquoise')
        self.c_amount.place(x=0 ,y=600)

    def generate_bill(self,*args,**kwargs):
        self.mycursor.execute("SELECT * FROM inventory WHERE
id=%s",[self.get_id])
        self.pc = self.mycursor.fetchall()
        for r in self.pc:
            self.old_stock=r[2]
        for i in products_list:
            for r in self.pc:
                self.old_stock = r[2]
            self.new_stock=int(self.old_stock) -
int(self.quantity_value)
            #updating the stock
            self.mycursor.execute("UPDATE inventory SET stock=%s WHERE
id=%s",[self.new_stock,self.get_id])
            self.conn.commit()

            #inster into transcation
            self.mycursor.execute("INSERT INTO transaction
(product_name,quantity,amount,date)
VALUES(%s,%s,%s,%s)",[self.get_name,self.quantity_value,self.get_price
,date])
            self.conn.commit()
            print("Decreased, Your Item is booked and sent soon within
Working Days","Thank you for your recent purchase. We are honored to
gain you as a customer and hope to serve you for a long time. We just
want to drop a quick note to express our genuine gratitude. Your
purchase allows us at WooCommerce Inventory Management to continue to
do what we love and provide you with quality products.")


        tkinter.messagebox.showinfo("Succesfully Done Everything
Smoothly", "Thank you for your recent purchase. We are honored to gain
```

```
you as a customer and hope to serve you for a long time. We just want
to drop a quick note to express our genuine gratitude. Your purchase
allows us at WooCommerce Inventory Management to continue to do what
we love and provide you with quality products.")
        tkinter.messagebox.showinfo("Thank You","You Will Get your
order soon")


root=Tk()
Application(root)
root.geometry("1366x768+0+0")
root.title("WooCommerce Inventory Management by Aditya Konda")
root.mainloop()
```

File No.2- Add_To-Database

```
#import all the modules
from tkinter import *
import mysql.connector
from mysql.connector import Error
import tkinter.messagebox

conn=mysql.connector.connect(host='localhost',
                                        database='inventory_system',
                                        user='root',
                                        password='')
mycursor = conn.cursor()
mycursor.execute("SELECT Max(id) from inventory")
result = mycursor.fetchall()
for r in result:
    id=r[0]

class Database:
    def __init__(self,master,*args,**kwargs):
        self.master=master
        self.heading=Label(master,text="Add in the
database",font=('arial 40 bold'),fg='steelblue')
        self.heading.place(x=400,y=0)


        #lables  for the window
        self.name_l=Label(master,text="Enter Product
Name",font=('arial 18 bold'))
```

```
        self.name_l.place(x=0,y=70)

        self.stock_l=Label(master,text="Enter Stocks",font=('arial 18
bold'))
        self.stock_l.place(x=0,y=120)

        self.cp_l = Label(master, text="Enter Cost Price ",
font=('arial 18 bold'))
        self.cp_l.place(x=0, y=170)


       #enteries for window

        self.name_e=Entry(master,width=25,font=('arial 18 bold'))
        self.name_e.place(x=380,y=70)

        self.stock_e = Entry(master, width=25, font=('arial 18
bold'))
        self.stock_e.place(x=380, y=120)

        self.cp_e = Entry(master, width=25, font=('arial 18 bold'))
        self.cp_e.place(x=380, y=170)

        #button to add to the database
        self.btn_add=Button(master,text='Add to
Database',width=25,height=2,bg='steelblue',fg='white',command=self.get
_items)
        self.btn_add.place(x=520,y=220)

        self.btn_clear=Button(master,text="Clear All
Fields",width=18,height=2,bg='red',fg='white',command=self.clear_all)
        self.btn_clear.place(x=350,y=220)

         #text box for the log
        self.tbBox=Text(master,width=60,height=18)
        self.tbBox.place(x=750,y=70)
        self.tbBox.insert(END,"ID has reached up to:"+str(id))

        self.master.bind('<Return>', self.get_items)
        self.master.bind('<Up>', self.clear_all)

    def get_items(self, *args, **kwargs):
    # get from entries
       self.name = self.name_e.get()
```

```
        self.stock = self.stock_e.get()
        self.cp = self.cp_e.get()

    # dynamic entries
        if self.name == '' or self.stock == '' or self.cp == '':
         tkinter.messagebox.showinfo("Error", "Please Fill all the
entries.")
        else:
         mycursor.execute("INSERT INTO inventory (name, stock, price)
VALUES(%s,%s,%s)",[self.name,self.stock,self.cp])
         conn.commit()
         # textbox insert
         self.tbBox.insert(END, "\n\nInseted " + str(self.name) + "
into the database with the quantity of " + str(self.stock))
         tkinter.messagebox.showinfo("Success", "Successfully added to
the database")


    def clear_all(self, *args, **kwargs):
        num = id + 1
        self.name_e.delete(0, END)
        self.stock_e.delete(0, END)
        self.cp_e.delete(0, END)



root=Tk()
b=Database(root)
root.geometry("1366x768+0+0")
root.title("Add in the database")
root.mainloop()
```

File No.3- Update Products

```
#import all the modules
from tkinter import *
import mysql.connector
from mysql.connector import Error
import tkinter.messagebox

conn=mysql.connector.connect(host='localhost',
                                        database='inventory_system',
```

```
                                        user='root',
                                        password='')
mycursor = conn.cursor()
mycursor.execute("SELECT Max(id) from inventory")
result = mycursor.fetchall()
for r in result:
    id=r[0]

class Database:
    def __init__(self,master,*args,**kwargs):
        self.master=master
        self.heading=Label(master,text="Update to the
databse",font=('arial 40 bold'),fg='steelblue')
        self.heading.place(x=400,y=0)

        #label and entry for id
        self.id_le=Label(master,text="Enter ID",font=('arial 18
bold'))
        self.id_le.place(x=0,y=70)

        self.id_leb=Entry(master,font=('arial 18 bold'),width=10)
        self.id_leb.place(x=380,y=70)


self.btn_search=Button(master,text="search",width=15,height=2,bg='oran
ge',command=self.search)
        self.btn_search.place(x=550,y=70)

        #lables  for the window
        self.name_l=Label(master,text="Enter Product
Name",font=('arial 18 bold'))
        self.name_l.place(x=0,y=120)

        self.stock_l=Label(master,text="Enter Stocks",font=('arial 18
bold'))
        self.stock_l.place(x=0,y=170)

        self.cp_l = Label(master, text="Enter Cost Price ",
font=('arial 18 bold'))
        self.cp_l.place(x=0, y=220)


       #enteries for window
```

```
        self.name_e=Entry(master,width=25,font=('arial 18 bold'))
        self.name_e.place(x=380,y=120)

        self.stock_e = Entry(master, width=25, font=('arial 18
bold'))
        self.stock_e.place(x=380, y=170)

        self.cp_e = Entry(master, width=25, font=('arial 18 bold'))
        self.cp_e.place(x=380, y=220)


        #button to add to the database
        self.btn_add=Button(master,text='Update
Database',width=25,height=2,bg='steelblue',fg='white',command=self.upd
ate)
        self.btn_add.place(x=520,y=300)



         #text box for the log
        self.tbBox=Text(master,width=60,height=18)
        self.tbBox.place(x=750,y=70)
        self.tbBox.insert(END,"ID has reached up to:"+str(id))

    def search(self, *args, **kwargs):
        mycursor.execute("SELECT * FROM inventory WHERE
id=%s",[self.id_leb.get()])
        result = mycursor.fetchall()
        for r in result:
            self.n1 = r[1]  # name
            self.n2 = r[2]  # stock
            self.n3 = r[3]  # cp
        conn.commit()

         #inster into the enteries to update
        self.name_e.delete(0,END)
        self.name_e.insert(0, str(self.n1))

        self.stock_e.delete(0, END)
        self.stock_e.insert(0, str(self.n2))

        self.cp_e.delete(0, END)
        self.cp_e.insert(0, str(self.n3))
```

```
    def update(self,*args,**kwargs):
          self.u1=self.name_e.get()
          self.u2 = self.stock_e.get()
          self.u3 = self.cp_e.get()


          mycursor.execute("UPDATE  inventory SET
name=%s,stock=%s,price=%s WHERE
id=%s",[self.u1,self.u2,self.u3,self.id_leb.get()])
          conn.commit()
          tkinter.messagebox.showinfo("Success","Update Database
successfully")

root=Tk()
b=Database(root)
root.geometry("1366x768+0+0")
root.title("Update to the database")
root.mainloop()
```

File No.4- Sql Query For Inventory Management System.

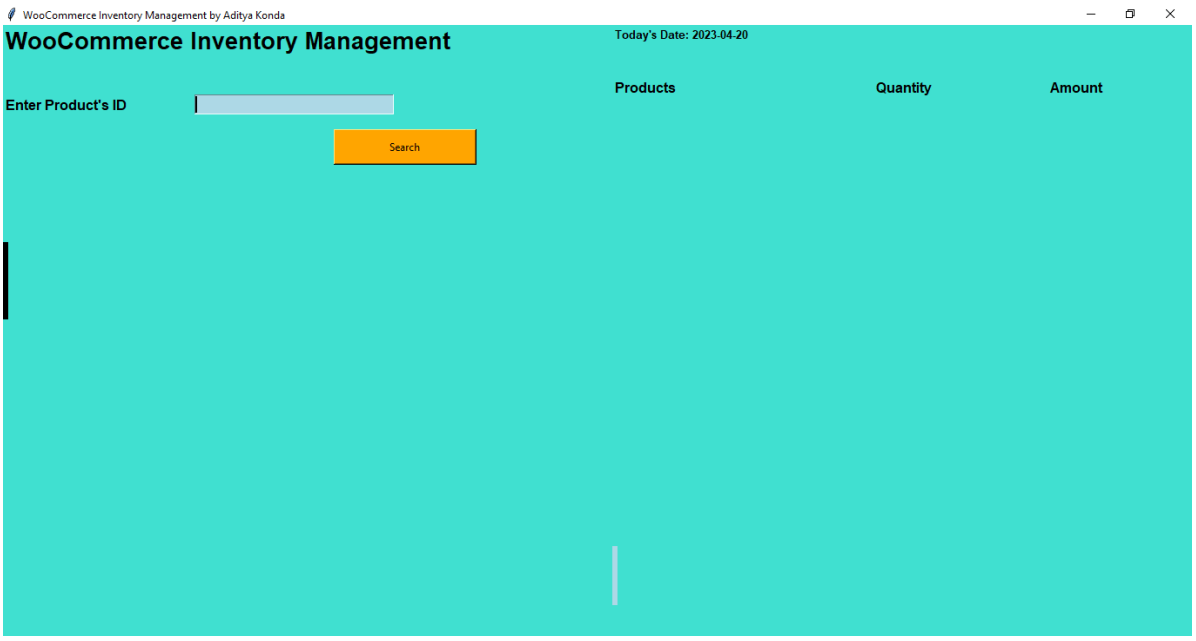
```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `inventory_system`
--

-- --------------------------------------------------------

--
-- Table structure for table `inventory`
```

```sql
--

CREATE TABLE `inventory` (
  `id` int(11) NOT NULL,
  `name` varchar(300) NOT NULL,
  `stock` int(11) NOT NULL,
  `price` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `inventory`
--

INSERT INTO `inventory` (`id`, `name`, `stock`, `price`) VALUES
(1, 'tinapay', 992, 100),
(2, 'pukloe pandesal', 500, 2),
(3, 'wowowow sardines', 1000, 20),
(4, 'lanson', 100, 5),
(5, 'okoy okoy', 100, 5);

-- --------------------------------------------------------

--
-- Table structure for table `transaction`
--

CREATE TABLE `transaction` (
  `id` int(11) NOT NULL,
  `product_name` varchar(300) NOT NULL,
  `quantity` int(11) NOT NULL,
  `amount` varchar(30) NOT NULL,
  `date` varchar(300) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `transaction`
--

INSERT INTO `transaction` (`id`, `product_name`, `quantity`, `amount`,
`date`) VALUES
(1, '0', 0, '0', '2020-06-27'),
(2, 'tinapay', 1, '100', '2020-06-27'),
(3, 'tinapay', 2, '100', '2020-06-27'),
(4, 'tinapay', 2, '100', '2020-06-27');
```

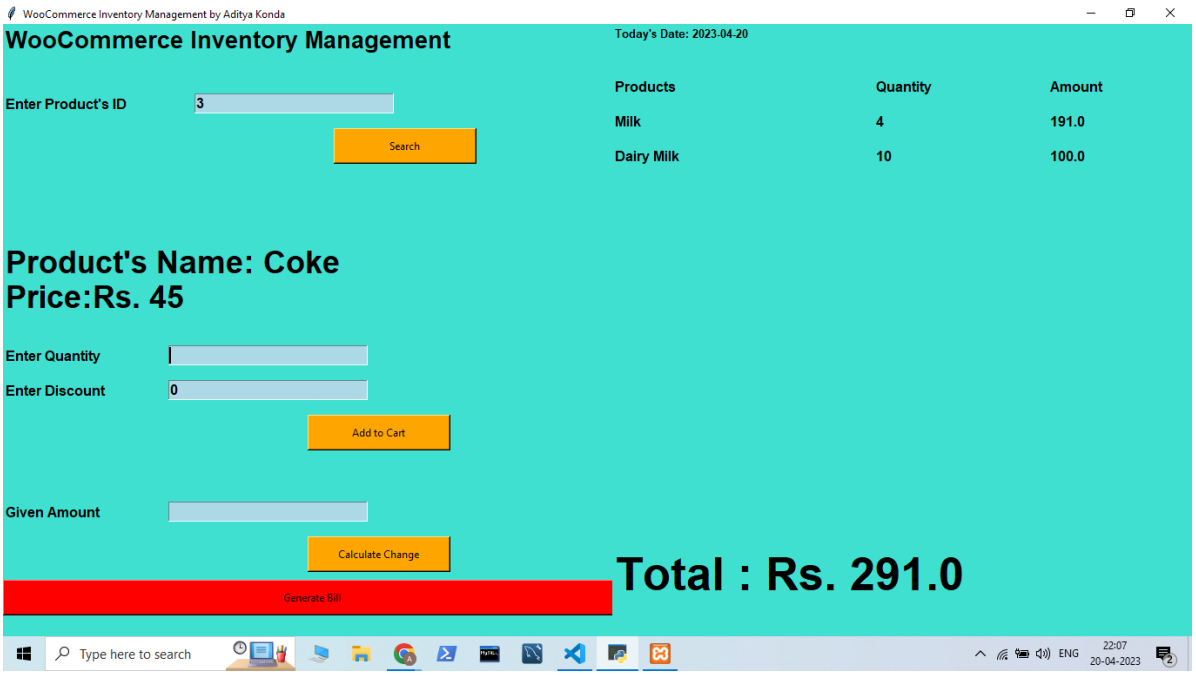
```sql
--
-- Indexes for dumped tables
--

--
-- Indexes for table `inventory`
--
ALTER TABLE `inventory`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `transaction`
--
ALTER TABLE `transaction`
  ADD PRIMARY KEY (`id`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `inventory`
--
ALTER TABLE `inventory`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;

--
-- AUTO_INCREMENT for table `transaction`
--
ALTER TABLE `transaction`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

Results / Output:-

1) Main.py all Outputs

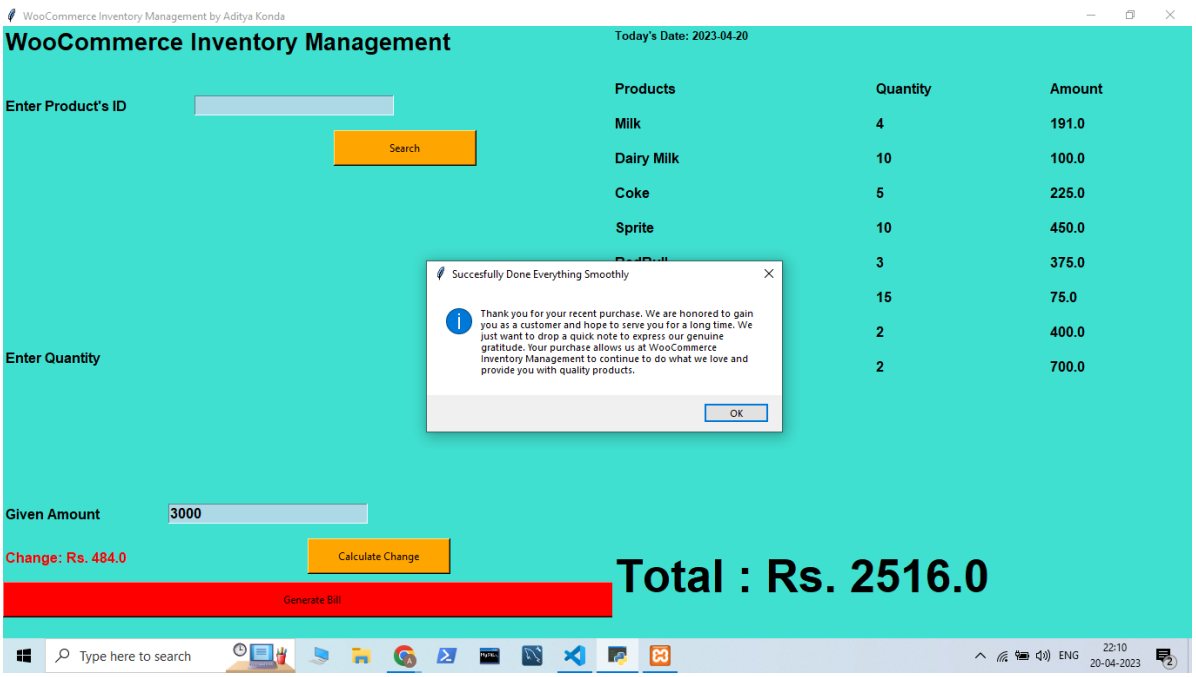


Added Products to Cart:-

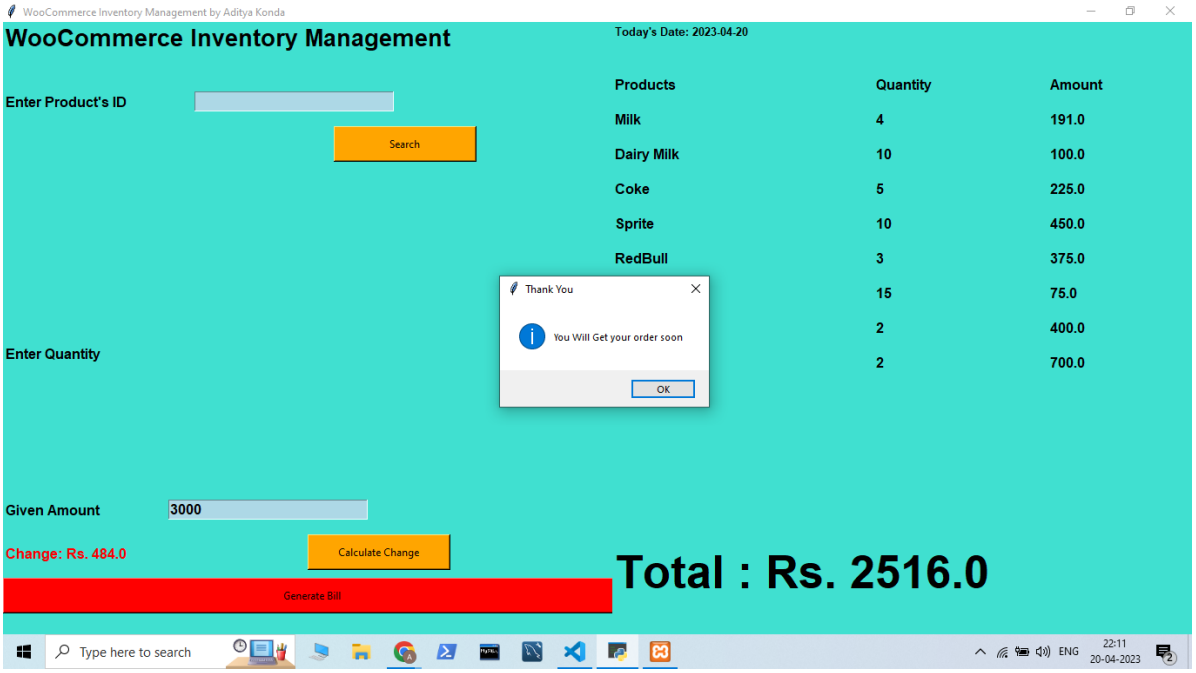


After Done with selecting all products:-

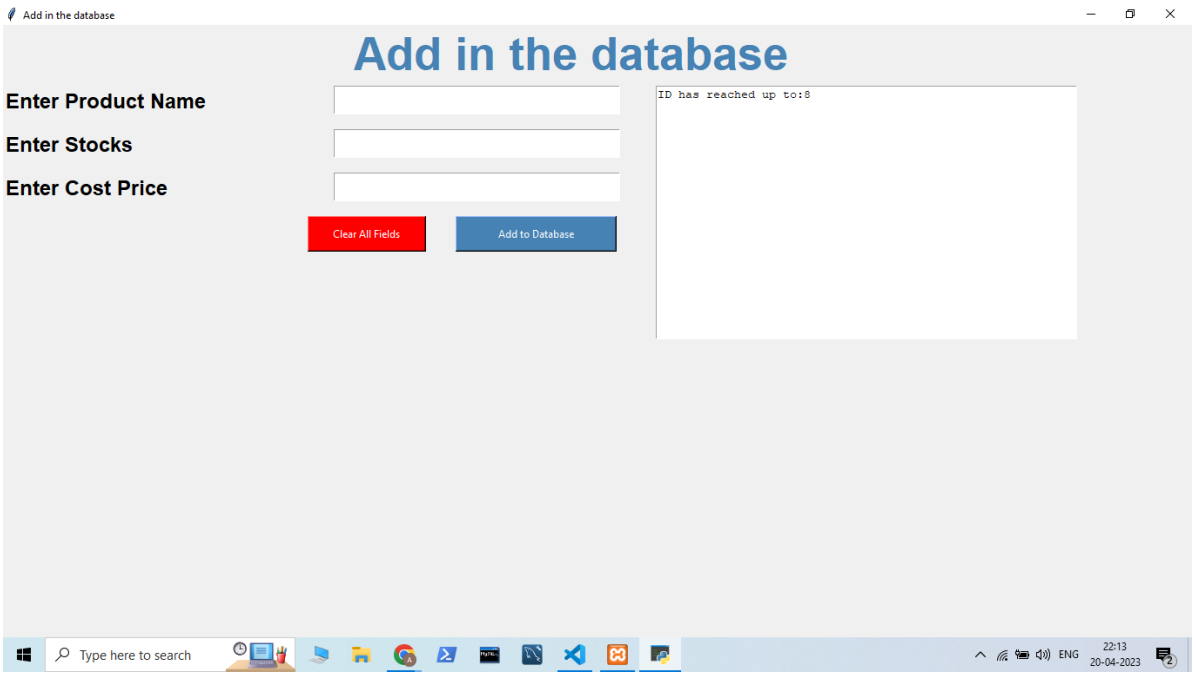
After Generating Bill:-



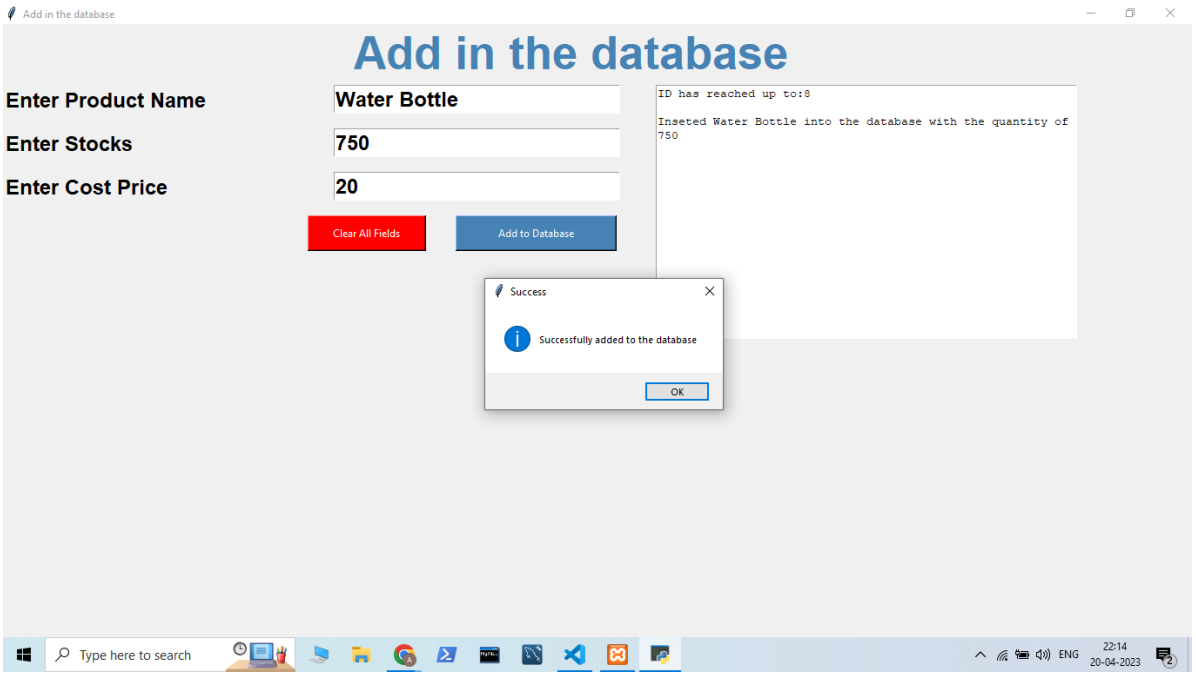
Thank You Message:-



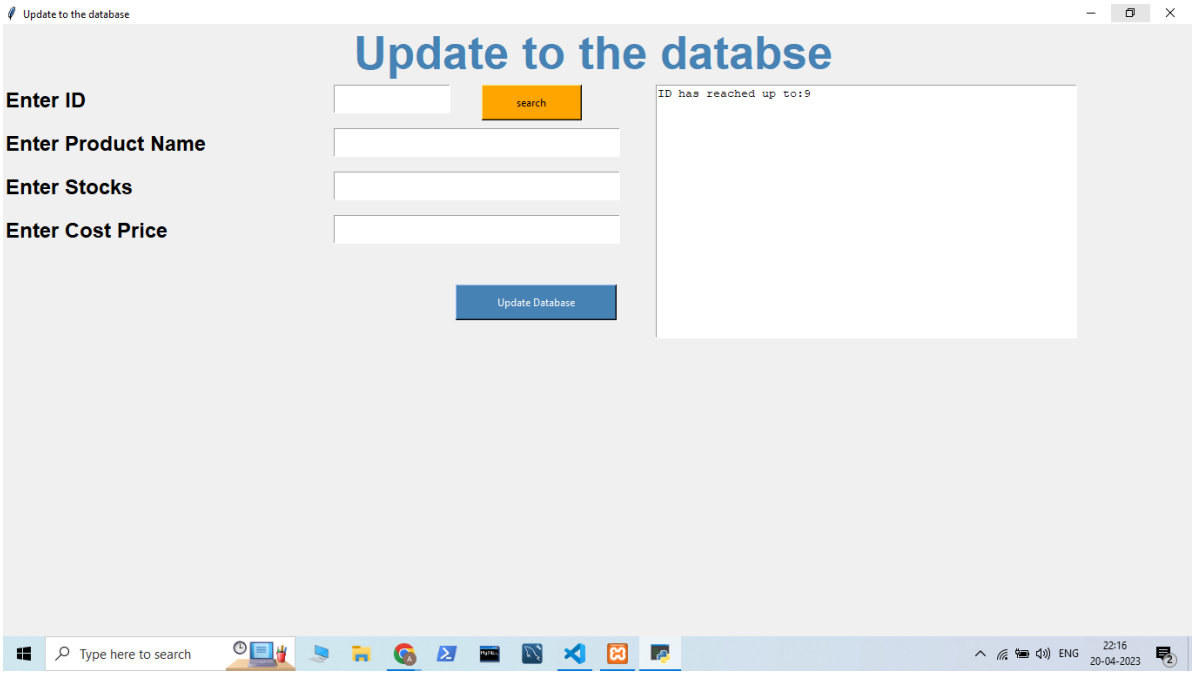
2) Add to Database all Output:-



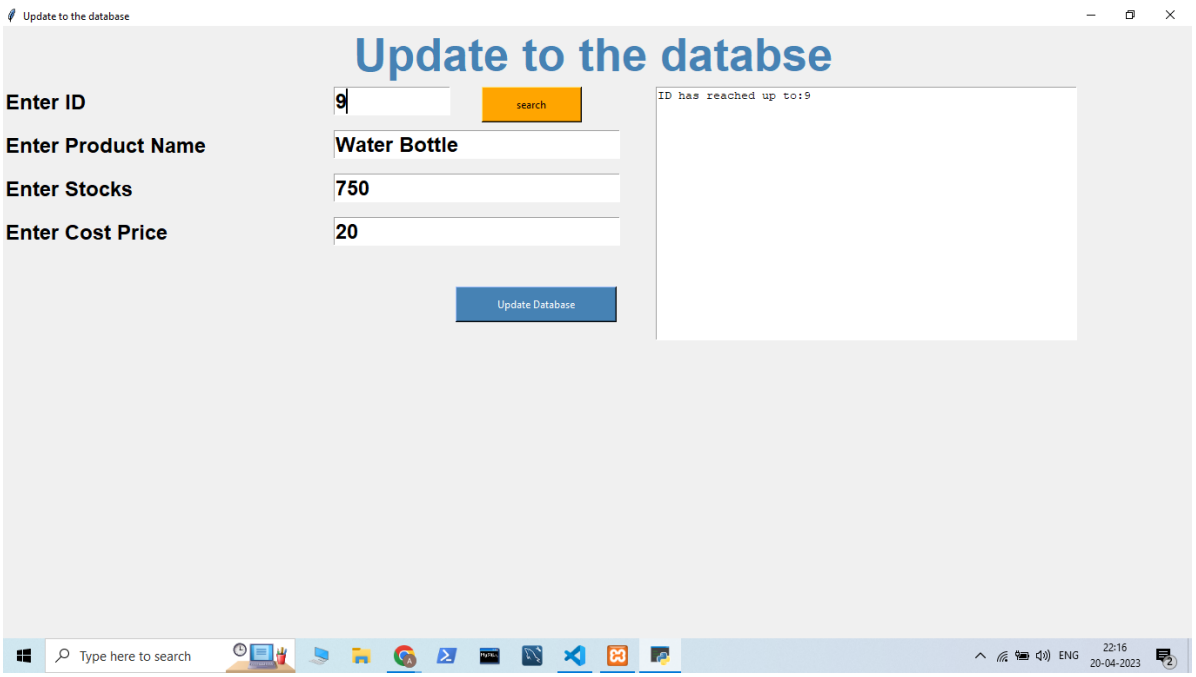
After Adding the Products-



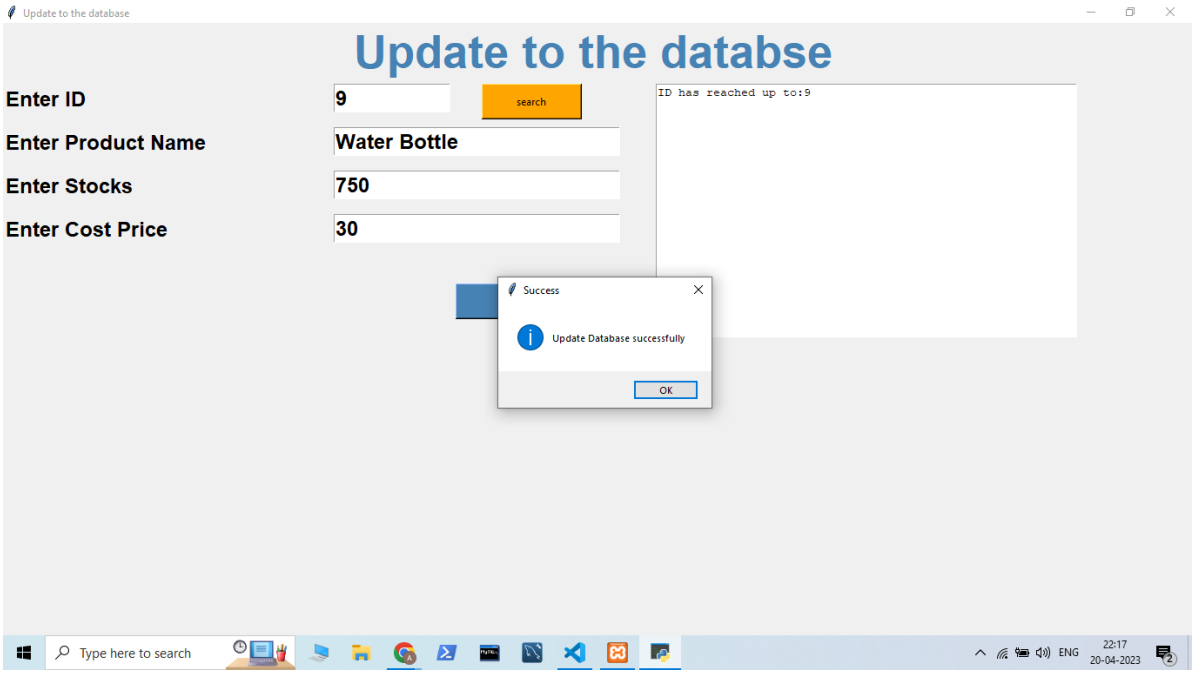
3) Update to the Database Output:-



After Searching-



Updated the product in Database Successfully-



3.5 Conclusion and Future work :

Conclusion:

In conclusion, the project inventory management system is an essential tool for any business that deals with inventory management. The system can help organizations manage their inventory efficiently, reduce costs, and increase profits. The experiment conducted on the prototype of the system revealed that the system has the potential to be an effective tool for managing inventory. User feedback, task completion rates, error rates, and overall satisfaction were all positive indicators that the system was well received by users. However, there may be areas of the system that need further improvement to make it more efficient and user-friendly.

Future work:

There are several areas of the project inventory management system that could be improved in future work. Some possible areas for improvement include:

1. Integration with other systems: The inventory management system could be integrated with other business systems, such as accounting or customer relationship management systems, to provide a more comprehensive view of the organization's operations.
2. Artificial intelligence and machine learning: The system could be enhanced by incorporating artificial intelligence and machine learning algorithms to automate certain inventory management tasks and provide insights into inventory trends.
3. Mobile compatibility: The system could be made compatible with mobile devices to allow users to access the system from anywhere, at any time.
4. User interface design: The system's user interface could be redesigned to make it more intuitive and user-friendly, reducing the learning curve for new users.
5. Advanced reporting: The system could be enhanced to provide more advanced reporting capabilities, such as predictive analytics, to help organizations make informed decisions about inventory management.

Overall, there is considerable scope for future work on the project inventory management system, and the improvements listed above could help to further increase the efficiency and effectiveness of the system