

E-COMMERCE INVENTORY MANAGEMENT SYSTEM

CSE3001 – Software Engineering PROJECT BASED COMPONENT REPORT

By

G AADHITHYAA (19BCI0121)

VISHAL HASWANI (19BCI0181)

KABILAN V (19BCE0623)

School of Computer Science and Engineering



May 2021

DECLARATION

I hereby declare that the report entitled “**E-Commerce Inventory Management System**” submitted by me, for the CSE3001 Software Engineering (EPJ) to VIT is a record of bonafide work carried out by me under the supervision of Dr. Vengadeswaran S.

I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for any other courses in this institute or any other institute or university.

Place: Vellore

Date: 06 June, 2021

Vishal Haswani

Signature of the Candidate

CONTENTS

<u>TITLE</u>	<u>PAGE NO</u>
1. ABSTRACT	4
2. INTRODUCTION TO THE PROJECT	
• OBJECTIVE	4
• MOTIVATION	4
• PROJECT SCOPE	4
3. PROJECT DESCRIPTION AND GOALS	
• WORK BREAKDOWN STRUCTURE	5
• PROCESS MODEL	5
• SOFTWARE REQUIREMENTS SPECIFICATION	5
• SOFTWARE DESIGN DOCUMENT	8
4. MODULE DESCRIPTIONS	16
5. USER INTERFACE DESIGN	16
6. TEST CASES	18
7. SOFTWARE METRICS	23
8. CONCLUSION	24
9. APPENDIX	
• SCREEN SHOTS	24
• SAMPLE CODING	37

ABSTRACT

Inventory is a complete list of items that are present in stock. Inventory management is the method of creating and maintaining detailed records of goods, right from the time of purchase to sale of those goods. It keeps track of date and quantity of goods purchased from the supplier, the products present in stock and helps keep track of the orders that are received and processed. Inventory management helps prevent dead stock, easily identify products that are under-stocked and also set aside products for emergency situations. It is important for businesses to effectively manage inventory coming in and inventory going out to maximize profits in business operations. An inventory management system can help businesses strike the right balance between being under and over-stocked for optimal efficiency and profitability.

INTRODUCTION TO THE PROJECT

- **OBJECTIVE**

The objective of this project is to develop an E-Commerce Inventory Management System that helps E-Commerce retailers to maintain a track of their inventory and to check upon all orders they receive from all E-Commerce platforms they sell their products in.

- **MOTIVATION**

The mother of one of the developers of our team is in this very business. She was discussing about the problems they are facing and what issues can be resolved with a new dedicated system specifically for one retailer which motivated us to get on and try out this project.

- **PROJECT SCOPE**

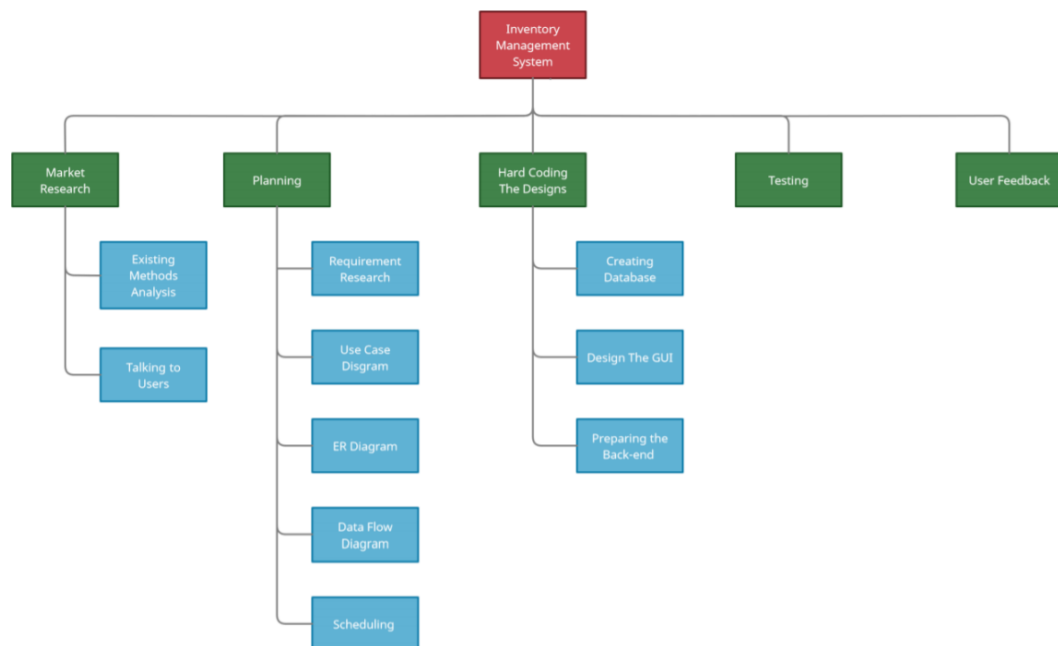
The software being specified in this document is an inventory management system that is being developed for small business owners who sell their products on e-commerce platforms like Amazon and Flipkart.

With the help of the system, the user will be able to keep a detailed record of his suppliers along with their contact info, the purchases that he has made from each supplier along with the current stock that is present in the inventory, detailed records of orders from the time they come in from different platforms till the time the product has been processed and sent for delivery and the money has been received from e-commerce platforms. We believe that this system will enable small business owners who sell their products on e-commerce platforms to effectively manage their inventory and strike the right balance between being under-stocked and over-stocked, which will in turn maximize their profits and also save them the time and work that they put in to manually write such records. The aim is to create such a system at minimum cost.

For this project, we will have a relational DB to store the supplier information, purchase information, stock and order details with a few triggers to automate certain processes. We will also be creating a clean and easy to understand GUI with the help of Python 3.9 and the functions to automate a few processes and the DB will be linked with the GUI also using Python. The project is to be done in three phases, designing the DB with its required components, the front-end module or the GUI using tkinter module from python and then preparing the back-end using Python.

PROJECT DESCRIPTION AND GOALS

- **WORK BREAKDOWN STRUCTURE**



- **PROCESS MODEL**

After a detailed analysis of all available process models, we have decided to follow the Waterfall model as the requirements are quite clear. This is a small project and does not require to be updated quite often. Which makes it our simplest choice. This also helps us to keep the user participation limited.

- **SOFTWARE REQUIREMENTS SPECIFICATION**

Functional Requirements

REQ-1: LOGIN MODULE

User should enter the password to access the features of the software. If they have an account already created, they can proceed by entering the password and logging in. Else they have to first register themselves by entering the password and then proceed with logging in.

REQ-2: VIEWING AVAILABLE STOCK

Once the user logs in, one among the many features of the software that they can use is the option to view available stock. Using this feature, the user can identify how much quantity of stock is available for each product and if there is any shortage, they can be noted down and orders can be placed for those stocks.

REQ-3: VIEWING AND UPDATING PRODUCT DETAILS

Since all products in the inventory will be out for sale on various E-Commerce platforms such as Amazon, Flipkart, etc., the user of the system should be able to view which stock is being sold in which E-Commerce platform and the details of the stock in the platform have to be updated in the inventory management system. Details can include Platform name, Selling Price etc. These details can be updated whenever there is a change in the details of the product in the particular E-Commerce platform.

REQ-4: VIEWING SUPPLIER DETAILS

All the stocks that are being stored and shipped out of the inventory will be purchased from particular suppliers in the market. The details of the supplier can be stored and viewed by the user whenever needed.

REQ-5: VIEWING ORDER DETAILS OBTAINED FROM E-COMMERCE PLATFORMS

Once any customer of an E-Commerce platform places an order for the products sold by the owner of the inventory and the system, the user will be able to view all the orders that were received from the E-Commerce platforms along with their details. Details include product and quantity, date of placing order, total amount of order etc.

REQ-6: UPDATION OF STOCK

The procedure of updating stock is one of the important functions of this system as it is responsible for the smooth functioning of the inventory. This procedure can occur in multiple situations:

- i. When the user purchases stocks from the supplier and the stock has reached the warehouse, the system should automatically update the amount of stock with the number what has been received from the supplier.
- ii. When an order from a customer has to be processed, first the user has to check whether he has enough stock to process. Once processed and packed, the amount of stock packed has to be automatically reduced from the inventory.
- iii. When an order sent out for delivery has been returned back, the amount of the stock that has been returned has to be updated automatically.

External Interface Requirements

1. SYSTEM INTERFACE:

Any popular OS which the user can use to run the designed system with the help of necessary applications such as Python 3.9, MySQL DB.

2. USER INTERFACE:

The GUI of the product shall be designed with the help of tkinter module in Python 3.9.

3. HARDWARE INTERFACE:

A secondary storage device is needed to backup the data in the system.

4. SOFTWARE INTERFACE:

A MySQL DB is needed to host the DB which will store the necessary details of the system

Non-functional Requirements

- **Performance Requirements**

The response time of the system and the DB should be in such a way that when the user wants to perform any value update in the DB, the DB should respond in the most minimal time possible so that the user can get his job done at the earliest. Since the system is run on the local system, the workload will be comparatively less than a central system.

- **Safety Requirements**

If in case the system of the user or the application gets crashed due to any unexpected emergencies, all the data that was stored in the DB should be backed up into the secondary storage device of the user system so that after reboot, no data is lost in the process and the work that was in process at that time can proceed from where it was left off.

- **Security Requirements**

The login details provided by the user which will be stored for faster access during the login process will be encrypted using the DES Algorithm to ensure extra layer of security to the user details and the algorithm should withstand any attacks from outside which may result in leak of sensitive user data.

- **Reliability Requirements**

The system shall be available 24 hours a day and 7 days a week for the user to perform operations and to maintain a smooth business.

- **Usability Requirements**

The user system should have all the necessary applications needed to run the system as mentioned in the System Interface section. The GUI that is being developed should be developed in such a way the user must get the feeling that it is easy to use

- **SOFTWARE DESIGN DOCUMENT**

- 1. Purpose**

The purpose of this document is to give a detailed description of the design for an inventory management system developed for small business owners who sell their products on e-commerce platforms like Amazon and Flipkart. The system is to be used by the user to manage his inventory by keeping track of his purchases from suppliers and also keeping records of orders that come in from various e-commerce platforms and monitor their status till it is sent for delivery. This document will illustrate in detail the complete design of the system, the constraints to develop the system and the scope of the project.

- 2. Intended Audience and Intended Use**

The SRS Document is intended to be used by the members of the development team to design the product E-commerce management system and to test that the designed system has met the specified requirements. The SRS document is also made available to our faculty, Mr. Vengadeswaran to verify that our system has met the specified requirements.

- 3. Project Scope**

The software being specified in this document is an inventory management system that is being developed for small business owners who sell their products on e-commerce platforms like Amazon and Flipkart.

With the help of the system, the user will be able to keep a detailed record of his suppliers along with their contact info, the purchases that he has made from each supplier along with the current stock that is present in the inventory, detailed records of orders from the time they come in from different platforms till the time the product has been processed and sent for delivery and the money has been received from e-commerce platforms. We believe that this system will enable small business owners who sell their products on e-commerce platforms to effectively manage their inventory and strike the right balance between being under-stocked and over-stocked, which will in turn maximize their profits and also save them the time and work that they put in to manually write such records. The aim is to create such a system at minimum cost.

For this project, we will have a relational DB to store the supplier information, purchase information, stock and order details with a few triggers to automate certain processes. We will also be creating a clean and easy to understand GUI with the help of Python 3.9 and the functions to automate a few processes and the DB will be linked with the GUI also using Python. The project is to be done in three phases, designing the DB with its required components, the front-end module or the GUI using tkinter module from python and then preparing the back-end using Python.

4. Definitions and Acronyms

Term	Definition
User	Person who will be using the system
DB	Database
GUI	Graphical User Interface
OS	Operating System
DES	Data Encryption Standard

5. Product Perspective

This product is an inventory management system specifically designed for the purpose of helping an E-commerce business owner. The software consists of an SQL DB, a frontend that enables the user to enter data through forms and a backend that joins the SQL DB and the forms.

6. User Needs

- a. People using the software:
 - i. Business Owner
 - ii. Warehouse Manager
- b. Keep track of the currently available product and their quantity.
- c. Keep track of the details of the order that comes and the order status, i.e., keep record of which order was packed, shipped, returned and whether the money was received or not.
- d. Keep track of the orders sent to the supplier.

7. Assumptions and Dependencies

- a. All the users using the software are trusted and there is no need to give separate privileges to users.
- b. The user is comfortable with filling all the information manually through forms in GUI.
- c. This is only useful for a small business receiving about 100-250 orders a day, as the only users are the owner and the manager and not the employees working under them.
- d. The systems have to be connected to the internet for sharing the common DB

8. Operating Environment

- a. Used by the business owner from any remote location with internet access.
- b. Used by the manager from the warehouse where all the products are going to be packaged.

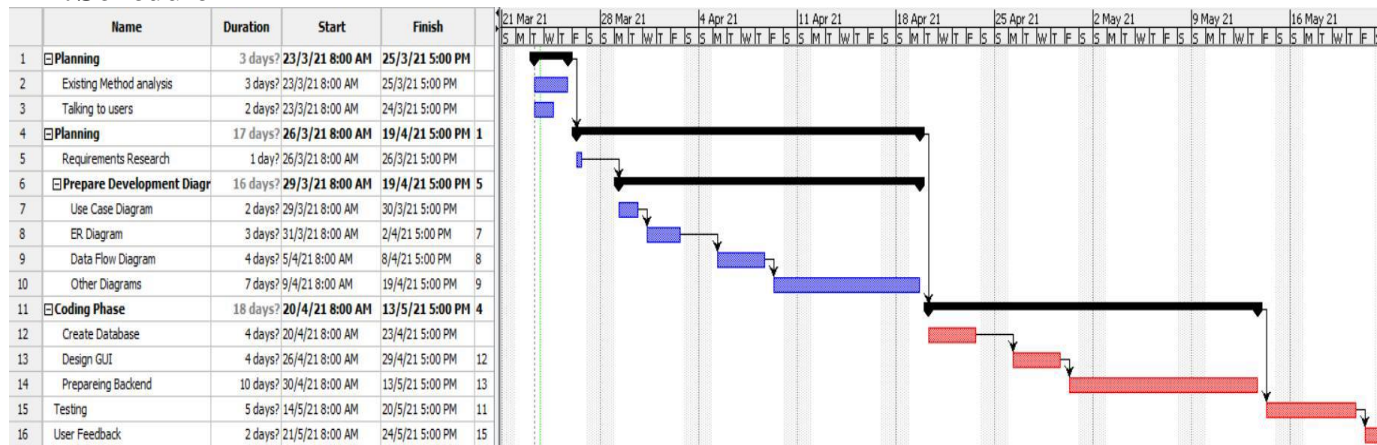
9. Design and Implementation Constraints

- The software must be completed within the stipulated time.
- The software runs on a computer that is connected to the internet.
- Python 3.9 or above must be installed.
- MySQL connector module should be installed.
- PyQt5 or Tkinter must be installed.

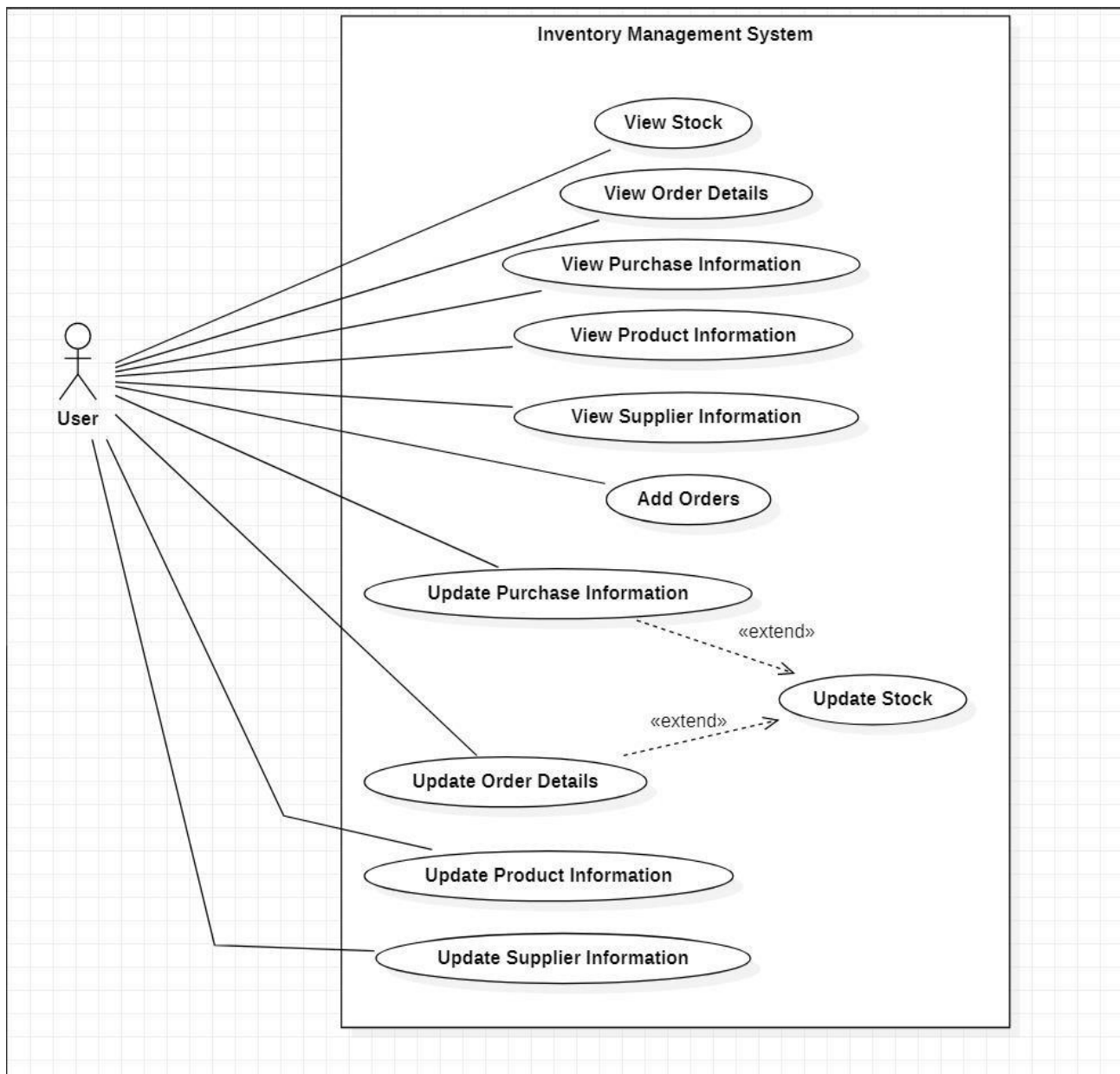
10. Milestones

- Database to be complete with all necessary tables and triggers.
- Designing part of GUI and Data Flow Diagram till Level-2
- Log In section has to be complete.
- Basic Inventory Management System has to be complete with the core functionalities.

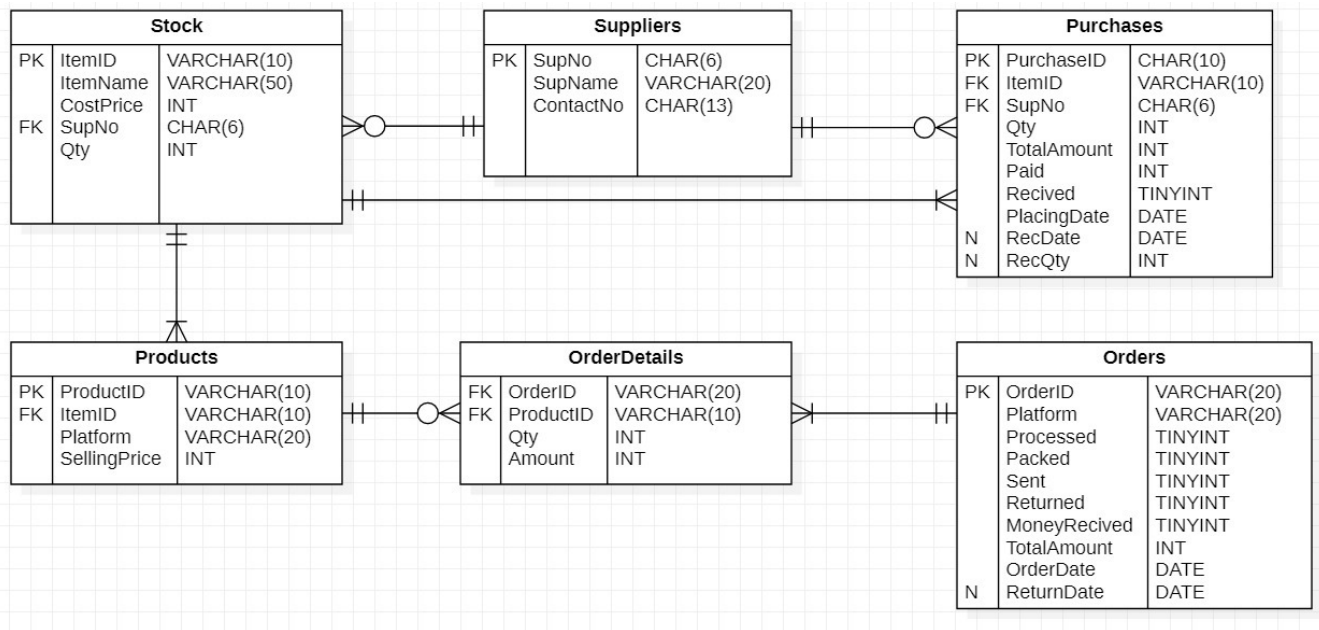
11. Schedule



12. USE CASE DIAGRAM

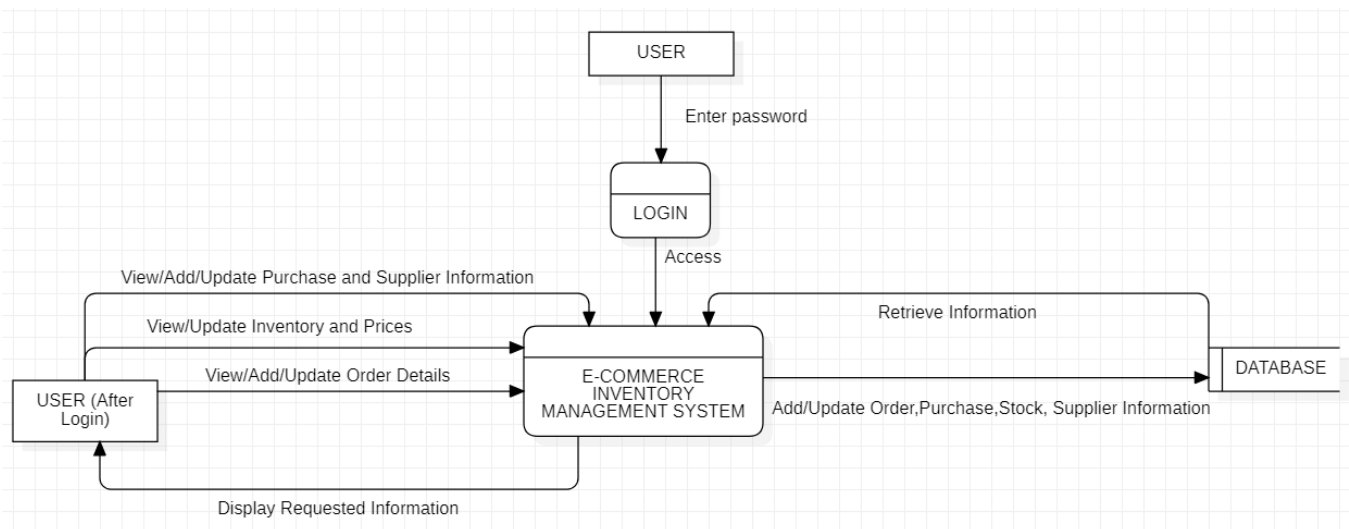


13. ER DIAGRAM

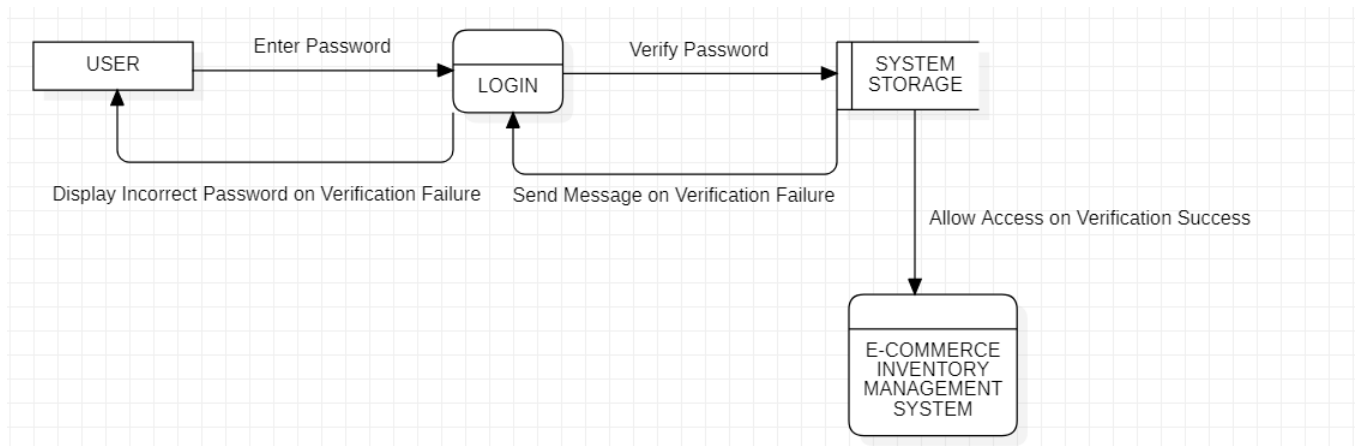


14. DATA FLOW DIAGRAMS

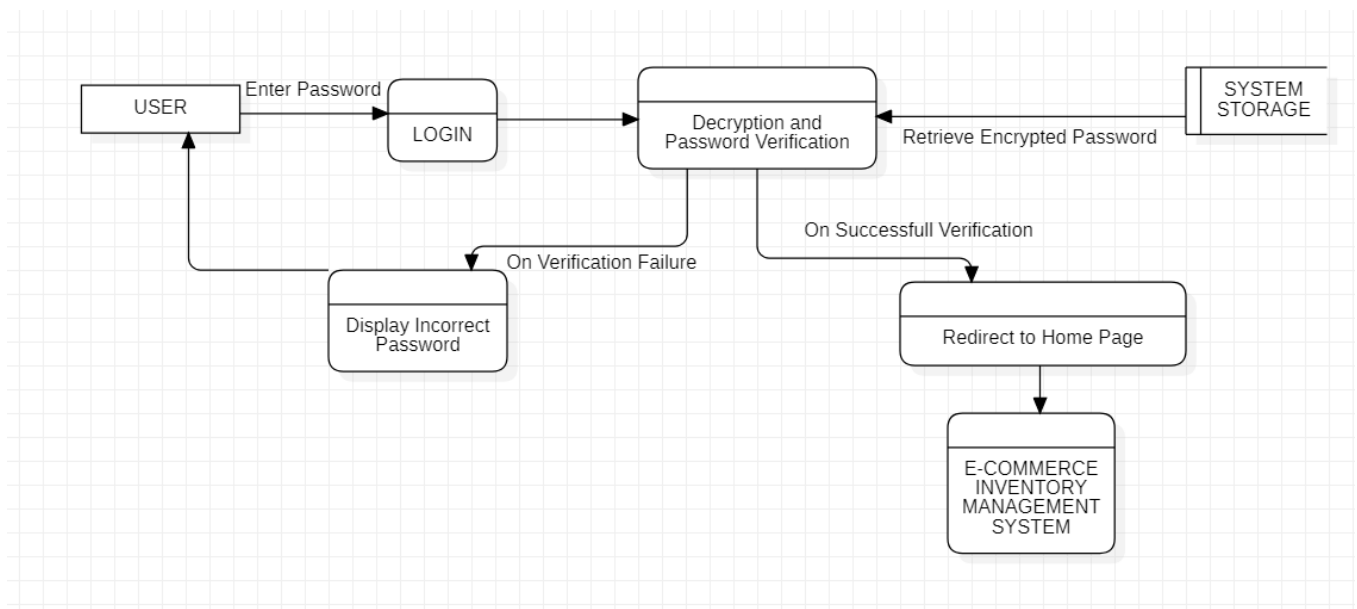
LEVEL 0 DFD



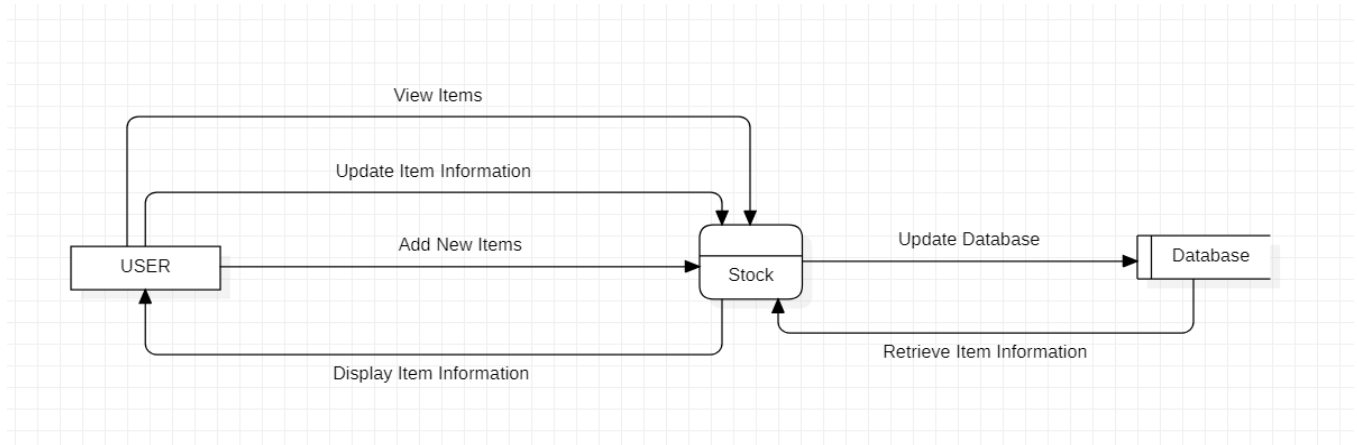
LEVEL 1 DFD FOR LOGIN



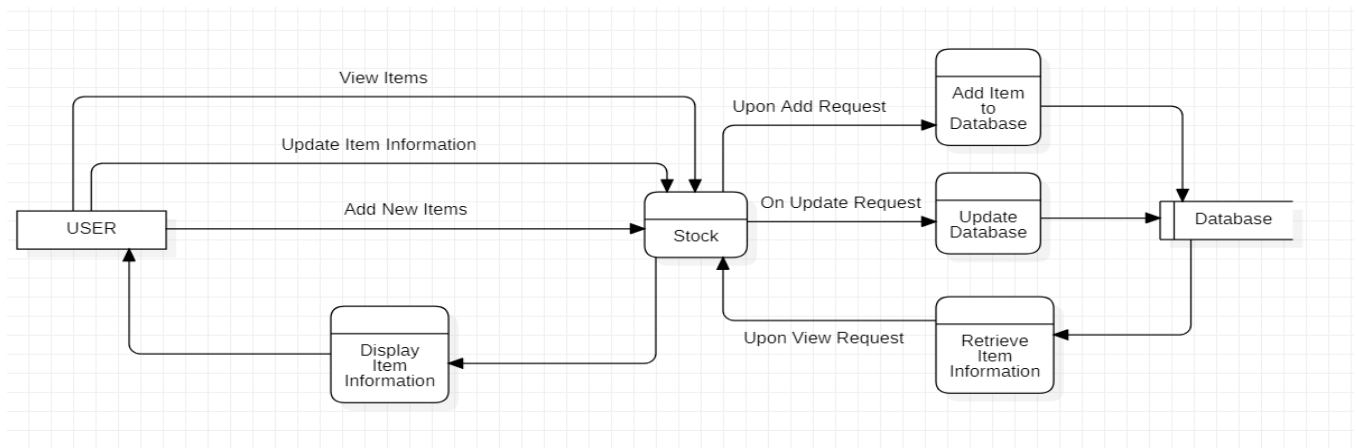
LEVEL 2 DFD FOR LOGIN



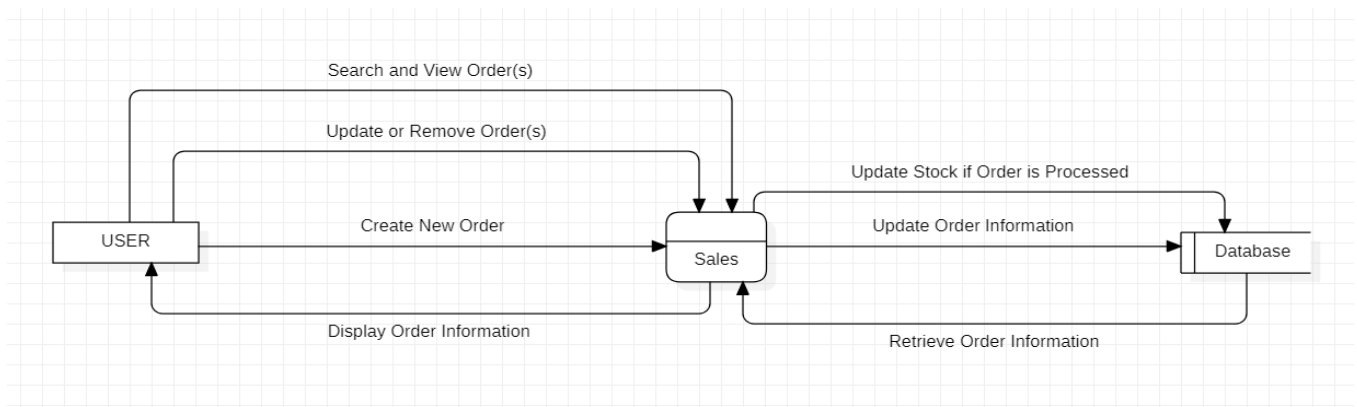
LEVEL 1 DFD FOR STOCK



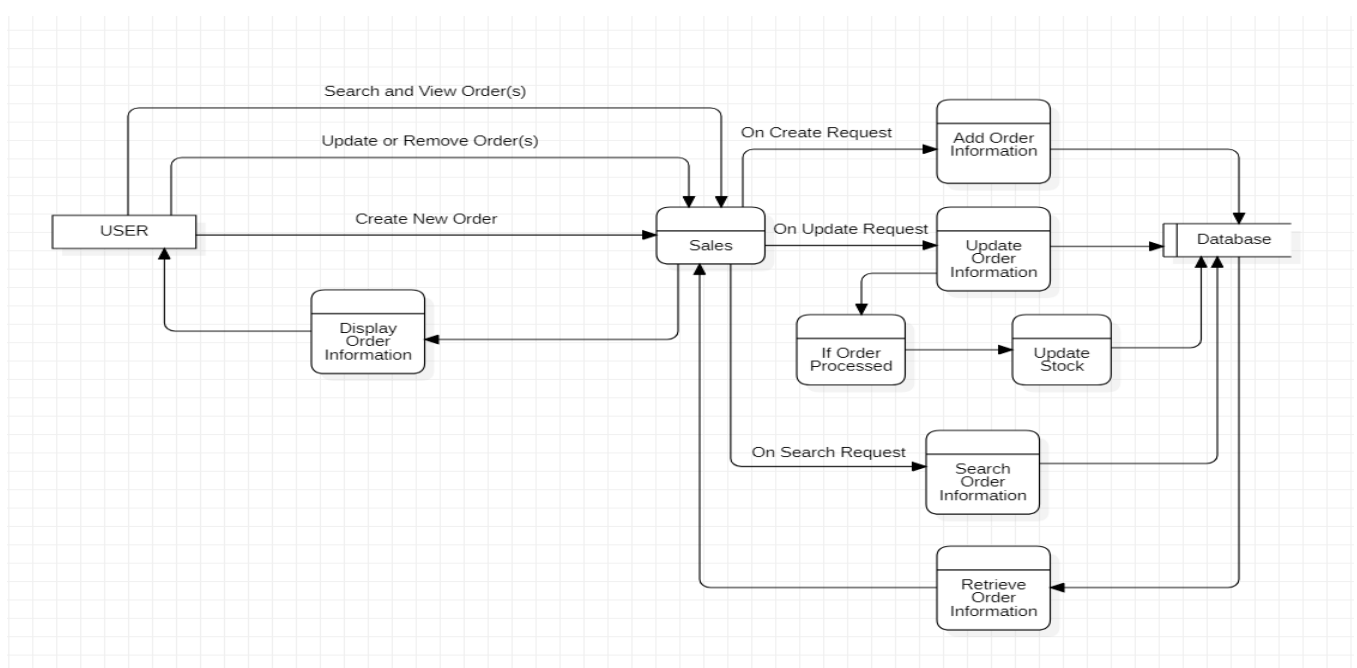
LEVEL 2 DFD FOR STOCK



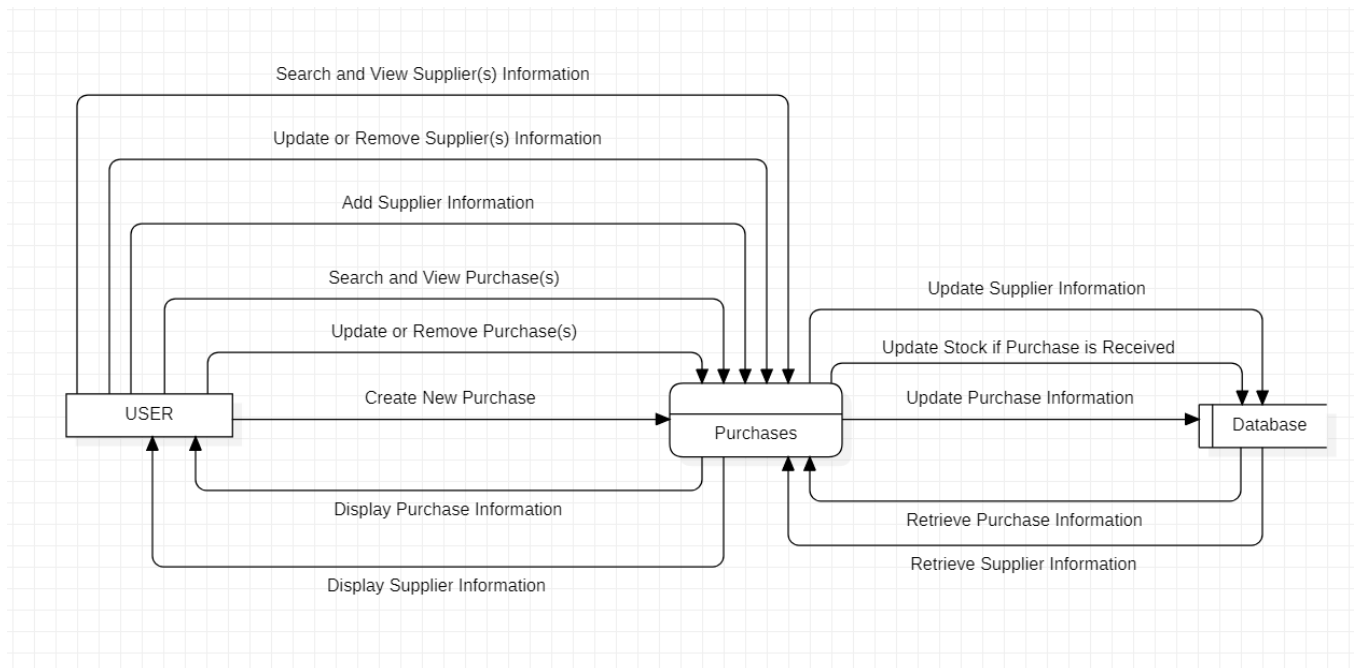
LEVEL 1 DFD FOR SALES



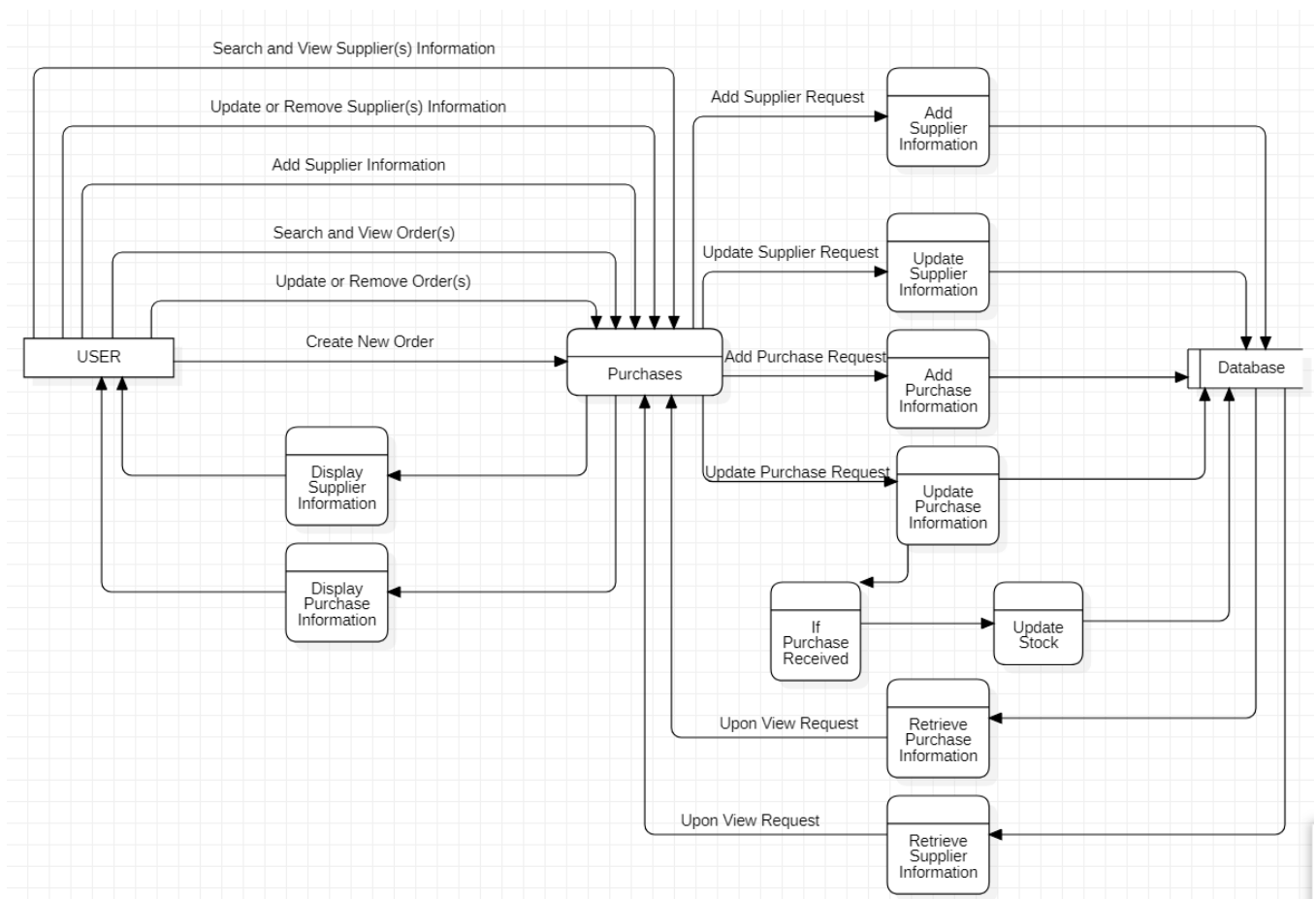
LEVEL 2 DFD FOR SALES



LEVEL 1 DFD FOR PURCHASES



LEVEL 2 DFD FOR PURCHASES



MODULE DESCRIPTIONS

1) Login Module:

The Login Module is included to ensure that the user alone is able to access the contents of the software. A password which is decided by the user will be used by them to enter into the home page of the system. The password will be stored in a file which will be encrypted using the Data Encryption Standard (DES) Algorithm to tighten the security provided to the credential of the user. If the password entered by the user matches with the password in the database, the user will be logged in and they will reach the home page of the system.

2) Sales Module:

The Sales Module has been developed keeping in mind the ease of having a check on the orders received from customers on these E-commerce platforms. This module is responsible for helping the user to add newly received orders, update the status of pending orders and remove cancelled orders. Another feature that is provided is to view the products that is being ordered by searching for the particular order.

3) Purchases Module:

The Purchases Module has been developed to help the user and owner of the system to keep track of the purchases they have done in the past for any product through its respective supplier. This module is responsible for adding new purchases, updating pending purchases and removing old purchases. It also has the facility of viewing the information of all suppliers who sell respective products. Adding new suppliers, updating information of suppliers and removing suppliers can also be done in this module.

4) Stock Module:

The Stock Module has been developed to help the user keep track of the amount of stock that is present for each product in the inventory. In case the user finds out that a particular stock is less in number, they can place an order to the supplier who sells that product. Once an order has been processed, this module is responsible to automatically reduce the number of stock available for that particular product and it is also responsible to automatically increase the number of stock available for that particular product once a purchase has been received.

USER INTERFACE DESIGN

A Python module named tkinter is used for the development of the user interface for all modules.

1) Login Module

-> A text field along with a “Login” button is given. The user must enter the correct password and then they will be taken to the home page.

-> The home page consists of six buttons namely Inventory to check the stock, Sales to update the order details, Purchases to update purchase details, Supplier to update

supplier details, Platform to update product details and Quit for quitting the application. The user can go to the module of their choice by clicking on any of the respective button.

2) Stock Module

-> In the Inventory Page, the user is provided with a form which has the respective details of an item and text fields in which the details can be entered. Four buttons namely Add, Update, Remove and Select are given to add, update, remove or select an item. A search bar along with a search button is also given to search for a particular item. A tree object is used to display the details of an item or multiple items. A back button is provided to reach the home page.

-> In the Platform Page, the user is provided with a form which has the respective details of a product and text fields in which the details can be entered. Four buttons namely Add, Update, Remove and Select are given to add, update, remove or select a product. A search bar along with a search button is also given to search for a particular product. A tree object is used to display the details of a product or multiple products. A back button is provided to reach the home page.

3) Sales Module

-> In the Sales Page, the user is provided with a form which has the respective details of an order and text fields along with checkboxes are provided in which the details can be entered. Four buttons namely Add, Update, Remove and Select are given to add, update, remove or select an order. A search bar along with a search button is also given to search for a particular order. A Search via Form button is also provided to search for orders based on details entered in the form. A tree object is used to display the details of an order or multiple orders. A button is provided to go to the Order Details page. A back button is provided to reach the home page.

-> In the Order Details Page, the user is provided with a form which has the respective details of a particular order namely Order ID, Product ID and Quantity and text fields have been provided in which the details can be entered. Three buttons namely Add, Remove and Select are given to add, remove and select a particular order and retrieve its details. A search button is also given to search for a particular order's details. A tree object is used to display the details of an order namely Product ID, Quantity and Amount. A back button is provided to reach the Sales page.

4) Purchases Module

-> In the Purchases Page, the user is provided with a form which has the respective details of a purchase and text fields along with checkboxes are provided in which the details can be entered. Four buttons namely Add, Update, Remove and Select are given to add, update, remove or select a purchase. A search bar along with a search button is also given to search for a particular purchase. A tree object is used to display the details of a purchase or multiple purchases. A back button is provided to reach the home page.

-> In the Supplier Page, the user is provided with a form which has the respective details of a particular supplier namely Supplier Number, Supplier Name and Contact Number and text fields have been provided in which the details can be entered. Four buttons namely Add, Update, Remove and Select are given to add, update, remove and select a particular supplier and retrieve their details. A search button is also given to search for

a particular supplier's details. A tree object is used to display the details of a supplier. A back button is provided to reach the home page.

TEST CASES

1) Login Module

Test Case ID	Test Case	Test Data	Expected Result	Actual Result	Comments
1	Enter Valid Password and Press Login Button	Valid Password	Login Successfully and go to Home Page	Login Successfully and go to Home Page	Success
2	Enter Invalid Password and Press Login Button	Invalid Password	Login Failed	Login Failed	Success

2) Stock Module

Test Case ID	Test Case	Test Data	Expected Result	Actual Result	Comments
1	Add new item with valid details	Valid Item Details	Item is added to database	Item is added to database	Success
2	Add new item with invalid details	Invalid Item Details	User is prompted to enter valid details	User is prompted to enter valid details	Success
3	Update item details with valid details	Valid Details	Item information is updated in database	Item information is updated in database	Success
4	Update item details with invalid details	Invalid Details	User is prompted to enter valid details	User is prompted to enter valid details	Success

5	Remove item with valid details	Valid Details	Item is removed from database	Item is removed from database	Success
6	Search for item with valid Item ID or valid Item Name	Valid Item ID or Valid Item Name	Item(s) with search value present are displayed	Item(s) with search value present are displayed	Success
7	Search for item with invalid Item ID or invalid Item Name	Invalid Item ID or Invalid Item Name	Nothing is displayed	Nothing is displayed	Success
8	Add new product with valid details	Valid Product Details	Product is added to database	Product is added to database	Success
9	Add new product with invalid details	Invalid Product Details	User is prompted to enter valid details	User is prompted to enter valid details	Success
10	Update product details with valid details	Valid Details	Product information is updated in database	Product information is updated in database	Success
11	Update product details with invalid details	Invalid Details	User is prompted to enter valid details	User is prompted to enter valid details	Success
12	Remove product with valid details	Valid Details	Product is removed from database	Product is removed from database	Success
13	Search for product with valid Item ID or valid Product ID	Valid Item ID or Valid Product ID	Product(s) with search value present are displayed	Product(s) with search value present are displayed	Success
14	Search for item with invalid Item	Invalid Item ID	Nothing is displayed	Nothing is displayed	Success

	ID or invalid Product ID	or Invalid Product ID			
--	--------------------------	-----------------------	--	--	--

3) Sales Module

Test Case ID	Test Case	Test Data	Expected Result	Actual Result	Comments
1	Add new order with valid details	Valid Order Details	Order is added to database	Order is added to database	Success
2	Add new order with invalid details	Invalid Order Details	User is prompted to enter valid details	User is prompted to enter valid details	Success
3	Update order details with valid details	Valid Details	Order information is updated in database	Order information is updated in database	Success
4	Update order details with invalid details	Invalid Details	User is prompted to enter valid details	User is prompted to enter valid details	Success
5	Remove order with valid details	Valid Details	Order is removed from database	Order is removed from database	Success
6	Search for order with valid Order ID or valid Platform	Valid Order ID or Valid Platform	Order(s) with search value present are displayed	Order(s) with search value present are displayed	Success
7	Search for order with invalid Order ID or invalid Platform	Invalid Order ID or Invalid Platform	Nothing is displayed	Nothing is displayed	Success

8	Add new product to an order with valid details	Valid Product Details	Product is added to the order in the database	Product is added to the order in the database	Success
9	Add new product to an order with invalid details	Invalid Product Details	User is prompted to enter valid details	User is prompted to enter valid details	Success
10	Remove products from an order with valid details	Valid Product Details	Product is removed from the order in the database	Product is removed from the order in the database	Success
11	Search for a product in an order with valid Order ID	Valid Order ID	Product with search value is displayed along with quantity and amount	Product with search value is displayed along with quantity and amount	Success
12	Search for a product in an order with invalid Order ID	Invalid Order ID	Nothing is displayed	Nothing is displayed	Success

4) Purchases Module:

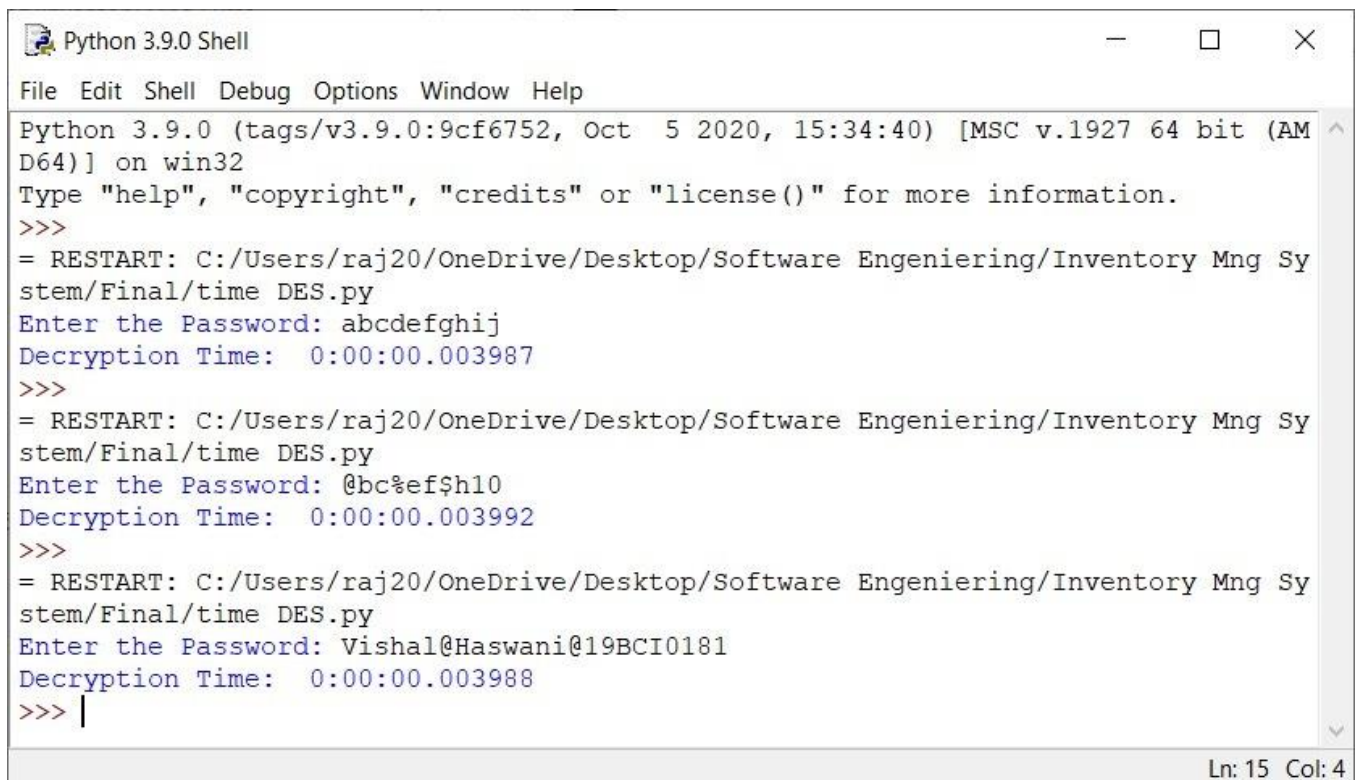
Test Case ID	Test Case	Test Data	Expected Result	Actual Result	Comments
1	Add new purchase with valid details	Valid Purchase Details	Purchase is added to database	Purchase is added to database	Success
2	Add new purchase with invalid details	Invalid Order Details	User is prompted to enter valid details	User is prompted to enter valid details	Success
3	Update purchase details with valid details	Valid Details	Purchase information	Purchase information	Success

			is updated in database	is updated in database	
4	Update purchase details with invalid details	Invalid Details	User is prompted to enter valid details	User is prompted to enter valid details	Success
5	Remove purchase with valid details	Valid Details	Purchase is removed from database	Purchase is removed from database	Success
6	Search for purchase with valid Purchase ID or valid Item ID or valid Supplier ID	Valid Purchase ID or valid Item ID or valid Supplier ID	Purchase(s) with search value present are displayed	Purchase(s) with search value present are displayed	Success
7	Search for purchase with invalid Purchase ID or valid Item ID or valid Supplier ID	Invalid Purchase ID or valid Item ID or valid Supplier ID	Nothing is displayed	Nothing is displayed	Success
8	Add new supplier information with valid details	Valid Supplier Details	Supplier is added to the database	Supplier is added to the database	Success
9	Add new supplier information with invalid details	Invalid Product Details	User is prompted to enter valid details	User is prompted to enter valid details	Success
10	Update supplier information with valid details	Valid Details	Supplier information is updated in database	Supplier information is updated in database	Success
11	Update supplier information with invalid details	Invalid Details	User is prompted to enter valid details	User is prompted to enter valid details	Success
12	Remove supplier information with valid details	Valid Supplier Details	Supplier is removed from the database	Supplier is removed from the database	Success

13	Search for a supplier with valid Supplier Name or Number	Valid Supplier Name or Number	Supplier with search value is displayed	Supplier with search value is displayed	Success
14	Search for a supplier with invalid Supplier Name or Number	Invalid Supplier Name or Number	Nothing is displayed	Nothing is displayed	Success

SOFTWARE METRICS

For a software, the metrics are considered to be the metrics that is obtained from any pre-defined algorithms used. In our software system, we have included a DES Algorithm in the Login Module to improve the security of the password that the user will use to enter to the home page. For the metrics of the DES Algorithm, it is going to be the time taken in microseconds for decryption of the password.



```

Python 3.9.0 Shell
File Edit Shell Debug Options Window Help
Python 3.9.0 (tags/v3.9.0:9cf6752, Oct 5 2020, 15:34:40) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/raj20/OneDrive/Desktop/Software Engeniering/Inventory Mng System/Final/time DES.py
Enter the Password: abcdefghij
Decryption Time: 0:00:00.003987
>>>
= RESTART: C:/Users/raj20/OneDrive/Desktop/Software Engeniering/Inventory Mng System/Final/time DES.py
Enter the Password: @bc%ef$h10
Decryption Time: 0:00:00.003992
>>>
= RESTART: C:/Users/raj20/OneDrive/Desktop/Software Engeniering/Inventory Mng System/Final/time DES.py
Enter the Password: Vishal@Haswani@19BCI0181
Decryption Time: 0:00:00.003988
>>> |
Ln: 15 Col: 4

```

As we can see from the above image, the time taken for decryption is within the range of 0.003987 seconds to 0.003992 seconds. One inference which we can take from this is that the length of the password will not have a huge effect on the decryption time and the system will be fast as it will be with smaller passwords.

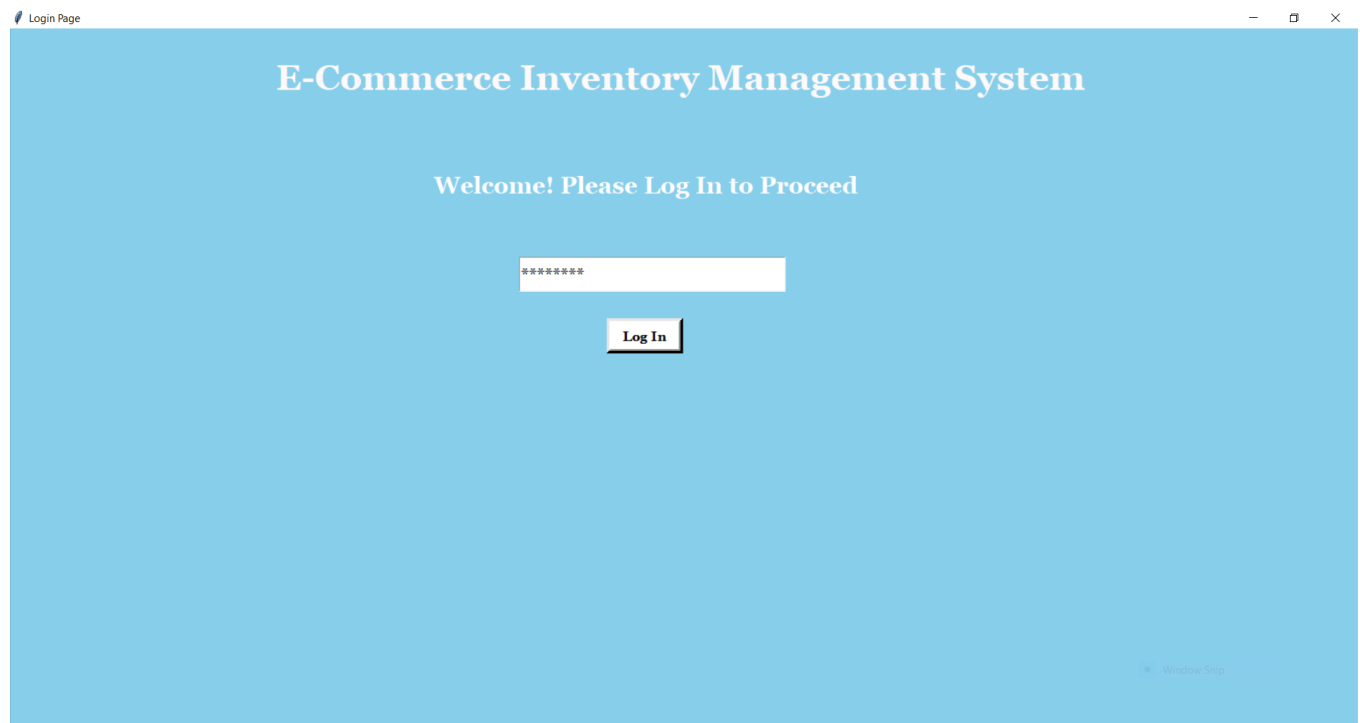
CONCLUSION

A software for E-Commerce Inventory Management has been developed using Python and MySQL. With the help of this software, the user will be able to keep a record of their supplier information, purchases made from different suppliers with the date and payment information. The user will also be able to keep a track of the items present in stock, their quantities and their selling prices on different E-Commerce platforms. In addition to this, the user will also be able to maintain records of the orders that come in from different platforms along with their payment information. Additionally, the user will be able to search, view and update any required information in any module. The system is designed in such a way that it automatically updates the stock when a purchase is received or an order is processed. The system also automatically updates the total amount for each order/ purchase when a new order/purchase is added or an existing order/purchase is modified. The User Interface of the system has been designed in such a way that it is easy to understand and use for the user. The final system was shown to one of the developers' mother, who is in this very business of selling products on E-Commerce platforms and she was satisfied with the system that has been developed.

APPENDIX

SCREEN SHOTS

1) Login Module

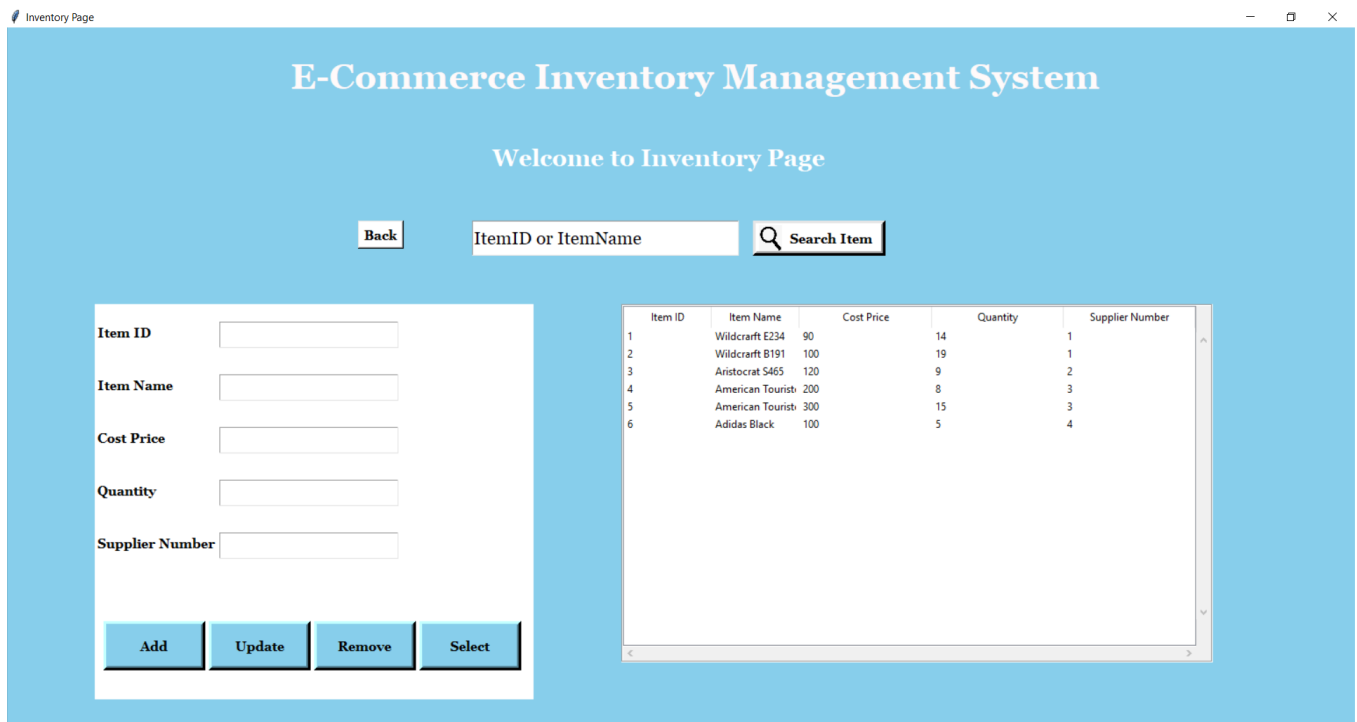


Login Screen



Home Page that is displayed after logging in

2) Stocks Module



Displaying existing stock upon clicking the search button

Inventory Page

E-Commerce Inventory Management System

Welcome to Inventory Page

Back

7

Search Item

Item ID

7

Item Name

Nike Air

Cost Price

500

Quantity

10

Supplier Number

4

Add

Update

Remove

Select

Item ID	Item Name	Cost Price	Quantity	Supplier Number
7	Nike Air	500	10	4

Adding an item and searching with Item ID

Inventory Page

E-Commerce Inventory Management System

Welcome to Inventory Page

Back

Search Item

Item ID

7

Item Name

Nike Air

Cost Price

500

Quantity

10

Supplier Number

4

Add

Update

Remove

Select

Item ID	Item Name	Cost Price	Quantity	Supplier Number
1	Wildcraft E234	90	14	1
2	Wildcraft B191	100	19	1
3	Aristocrat S465	120	9	2
4	American Tourist	200	8	3
5	American Tourist	300	15	3
6	Adidas Black	100	5	4

Removing the item by giving details in the form and pressing Remove

Inventory Page

E-Commerce Inventory Management System

Welcome to Inventory Page

Back

Search Item

Item ID

6

Item Name

Adidas Black

Cost Price

100

Quantity

6

Supplier Number

4

Add

Update

Remove

Select

Item ID	Item Name	Cost Price	Quantity	Supplier Number
1	Wildcraft E234	90	14	1
2	Wildcraft B191	100	19	1
3	Aristocrat S465	120	9	2
4	American Tourist	200	8	3
5	American Tourist	300	15	3
6	Adidas Black	100	6	4

Updating the Quantity of a particular stock and displaying

Products on Different Platform

E-Commerce Inventory Management System

Welcome to Inventory Page

Back

ItemID or ProductID or Platform

Search Item

Product ID

Item ID

Platform

Selling Price

Add

Update

Remove

Select

Product ID	Item ID	Platform	Selling Price
A1	1	Amazon	199
A2	2	Amazon	239
A3	3	Amazon	299
A4	4	Amazon	349
A5	5	Amazon	499
A6	6	Amazon	199
F1	1	Flipkart	199
F2	2	Flipkart	249
F3	3	Flipkart	299
F4	4	Flipkart	349
F5	5	Flipkart	499
F6	6	Flipkart	199

Displaying existing product details upon clicking the search button

E-Commerce Inventory Management System

Welcome to Inventory Page

Back

Myntra

Search Item

Product ID

Item ID

Platform

Selling Price

Add

Update

Remove

Select

Product ID	Item ID	Platform	Selling Price
M1	1	Myntra	199

Adding a product and searching with Platform Name

E-Commerce Inventory Management System

Welcome to Inventory Page

Back

Search Item

Product ID

Item ID

Platform

Selling Price

Add

Update

Remove

Select

Product ID	Item ID	Platform	Selling Price
A1	1	Amazon	199
A2	2	Amazon	239
A3	3	Amazon	299
A4	4	Amazon	349
A5	5	Amazon	499
A6	6	Amazon	199
F1	1	Flipkart	199
F2	2	Flipkart	249
F3	3	Flipkart	299
F4	4	Flipkart	349
F5	5	Flipkart	499
F6	6	Flipkart	199

Removing the product by giving details in the form and pressing Remove

Products on Different Platform

E-Commerce Inventory Management System

Welcome to Inventory Page

[Back](#)

Product ID:

Item ID:

Platform:

Selling Price:

Product ID	Item ID	Platform	Selling Price
A1	1	Amazon	249
A2	2	Amazon	239
A3	3	Amazon	299
A4	4	Amazon	349
A5	5	Amazon	499
A6	6	Amazon	199
F1	1	Flipkart	199
F2	2	Flipkart	249
F3	3	Flipkart	299
F4	4	Flipkart	349
F5	5	Flipkart	499
F6	6	Flipkart	199

Updating the Selling Price of a particular product and displaying

3) Sales Module

Sales Page

E-Commerce Inventory Management System

Welcome! Update Pending Order Details

[Back](#)

☐ Include CheckBoxes in Search

Order ID:

Platform:

Order Date:

☐ Processed
 ☐ Packed

☐ Sent
 ☐ Money Recieved

☐ Returned

Returned Date:

Order ID	Platform	Processed	Packed	Sent	Returned	Money Recieved	Total Amount	Order Date	Returned Date
1	Amazon	1	0	0	0	0	199	2021-02-04	None
2	Amazon	1	0	0	0	0	698	2021-02-05	None
3	Flipkart	1	0	0	0	0	249	2021-02-05	None
4	Flipkart	1	0	0	0	0	299	2021-02-06	None

Displaying existing orders upon clicking the search button

Sales Page

E-Commerce Inventory Management System

[Back](#) Welcome! Update Pending Order Details

☐ Include CheckBoxes in Search

[Search](#) [Search via Form](#) [Order Details](#)

Order ID:

Platform:

Order Date:

☒ Processed ☐ Packed

☐ Sent ☐ Money Recieved

☐ Returned

Returned Date:

[Add](#) [Update](#) [Remove](#) [Select](#)

Order ID	Platform	Processed	Packed	Sent	Returned	Money Recieved	Total Amount	Order Date	Returned Date
1	Amazon	1	0	0	0	0	199	2021-02-04	None

Displaying existing orders upon clicking the search via form button

Sales Page

E-Commerce Inventory Management System

[Back](#) Welcome! Update Pending Order Details

☐ Include CheckBoxes in Search

[Search](#) [Search via Form](#) [Order Details](#)

Order ID:

Platform:

Order Date:

☒ Processed ☐ Packed

☐ Sent ☐ Money Recieved

☐ Returned

Returned Date:

[Add](#) [Update](#) [Remove](#) [Select](#)

Order ID	Platform	Processed	Packed	Sent	Returned	Money Recieved	Total Amount	Order Date	Returned Date
5	Myntra	1	0	0	0	0	0	2021-02-08	None

Adding an order and searching with Order ID

Sales Page

E-Commerce Inventory Management System

[Back](#) Welcome! Update Pending Order Details

☐ Include CheckBoxes in Search

Order ID:

Platform:

Order Date:

☒ Processed ☐ Packed

☐ Sent ☐ Money Recieved

☐ Returned

Returned Date:

Order ID	Platform	Processed	Packed	Sent	Returned	Money Recieved	Total Amount	Order Date	Returned Date
1	Amazon	1	1	1	0	0	199	2021-02-04	None
2	Amazon	1	0	0	0	0	698	2021-02-05	None
3	Flipkart	1	0	0	0	0	249	2021-02-05	None
4	Flipkart	1	0	0	0	0	299	2021-02-06	None

Removing the order by giving details in the form and pressing Remove

Sales Page

E-Commerce Inventory Management System

[Back](#) Welcome! Update Pending Order Details

☐ Include CheckBoxes in Search

Order ID:

Platform:

Order Date:

☒ Processed ☒ Packed

☒ Sent ☐ Money Recieved

☐ Returned

Returned Date:

Order ID	Platform	Processed	Packed	Sent	Returned	Money Recieved	Total Amount	Order Date	Returned Date
1	Amazon	1	1	1	0	0	199	2021-02-04	None
2	Amazon	1	0	0	0	0	698	2021-02-05	None
3	Flipkart	1	0	0	0	0	249	2021-02-05	None
4	Flipkart	1	0	0	0	0	299	2021-02-06	None

Updating the order details and displaying

Orders Window

Back

Order ID **Search Order**

Product ID

Quantity

Add **Remove** **Select**

Product ID	Quantity	Amount
A4	2	698

Searching for details of an order by giving Order ID

Orders Window

Back

Order ID **Search Order**

Product ID

Quantity

Add **Remove** **Select**

Product ID	Quantity	Amount
A4	2	698
A3	2	598

Adding new products to an order

4) Purchases Module

Purchases Page

E-Commerce Inventory Management System

[Back](#) Welcome! Update Existing Purchase Details

Purchase ID

Item ID

Supplier Number

Quantity

Total Amount

Paid Amount

Placing Date

Received Date

Received Quantity

☐ Received

[Add](#) [Update](#) [Remove](#) [Select](#)

PurchaseID or ItemID or Supplier ID [Search](#)

Purchase ID	Item ID	Supplier Number	Quantity	Total Amount	Paid Amount	Received	Placing Date	Received Date	Received Quantity
1	1	1	15	1350	1350	1	2021-01-14	2021-01-31	15
2	2	1	20	2000	2000	1	2021-01-14	2021-01-31	20
3	3	2	10	1200	1200	1	2021-01-14	2021-01-30	10
4	4	3	10	2000	2000	1	2021-01-14	2021-01-28	10
5	5	3	15	4500	4500	1	2021-01-14	2021-01-28	15
6	6	4	5	500	500	1	2021-01-14	2021-01-27	5

Displaying existing purchases upon clicking search button

Purchases Page

E-Commerce Inventory Management System

[Back](#) Welcome! Update Existing Purchase Details

Purchase ID

Item ID

Supplier Number

Quantity

Total Amount

Paid Amount

Placing Date

Received Date

Received Quantity

☒ Received

[Add](#) [Update](#) [Remove](#) [Select](#)

[Search](#)

Purchase ID	Item ID	Supplier Number	Quantity	Total Amount	Paid Amount	Received	Placing Date	Received Date	Received Quantity
7	3	2	30	2700	2700	1	2021-02-10	2021-02-20	30

Adding a purchase and searching with Purchase ID

E-Commerce Inventory Management System

[Back](#)

Welcome! Update Existing Purchase Details

Purchase ID
Item ID
Supplier Number
Quantity
Total Amount
Paid Amount
Placing Date
Received Date
Received Quantity

☒ Received

Add

Update

Remove

Select

Purchase ID	Item ID	Supplier Number	Quantity	Total Amount	Paid Amount	Received	Placing Date	Received Date	Received Quantity
1	1	1	15	1350	1350	1	2021-01-14	2021-01-31	15
2	2	1	20	2000	2000	1	2021-01-14	2021-01-31	20
3	3	2	10	1200	1200	1	2021-01-14	2021-01-30	10
4	4	3	10	2000	2000	1	2021-01-14	2021-01-28	10
5	5	3	15	4500	4500	1	2021-01-14	2021-01-28	15
6	6	4	5	500	500	1	2021-01-14	2021-01-27	5

Removing the purchase by giving details in the form and pressing Remove

E-Commerce Inventory Management System

[Back](#)

Welcome! Update Existing Purchase Details

Purchase ID
Item ID
Supplier Number
Quantity
Total Amount
Paid Amount
Placing Date
Received Date
Received Quantity

☒ Received

Add

Update

Remove

Select

Purchase ID	Item ID	Supplier Number	Quantity	Total Amount	Paid Amount	Received	Placing Date	Received Date	Received Quantity
1	1	1	15	1350	1350	1	2021-01-14	2021-01-31	15
2	2	1	20	2000	2000	1	2021-01-14	2021-01-31	20
3	3	2	10	1200	1200	1	2021-01-14	2021-01-30	10
4	4	3	10	2000	2000	1	2021-01-14	2021-01-29	10
5	5	3	15	4500	4500	1	2021-01-14	2021-01-28	15
6	6	4	5	500	500	1	2021-01-14	2021-01-27	5

Updating the purchase details and displaying

Supplier Info

E-Commerce Inventory Management System

Welcome! Enter Supplier Details

[Back](#) [Search Item](#)

Supplier Number

Supplier Name

Contact Number

[Add](#) [Update](#) [Remove](#) [Select](#)

Supplier Number	Supplier Name	Contact Number
1	Rajashri Traders	+919123934346
2	Saikiran Agencies	+919287118909
3	AJB Wholesale Store	+919568478587
4	Jaishankar Traders	+919236542874

Displaying existing suppliers upon clicking search button

Supplier Info

E-Commerce Inventory Management System

Welcome! Enter Supplier Details

[Back](#) [Search Item](#)

Supplier Number

Supplier Name

Contact Number

[Add](#) [Update](#) [Remove](#) [Select](#)

Supplier Number	Supplier Name	Contact Number
5	Parvathy Suppliers	+919587401023

Adding a supplier and searching with Supplier Number

Supplier Info

E-Commerce Inventory Management System

Welcome! Enter Supplier Details

[Back](#) [Search Item](#)

Supplier Number

Supplier Name

Contact Number

[Add](#)
[Update](#)
[Remove](#)
[Select](#)

Supplier Number	Supplier Name	Contact Number
1	Rajashri Traders	+919123934346
2	Saikiran Agencies	+919287118909
3	AJB Wholesale Store	+919568478587
4	Jaishankar Traders	+919236542874

Removing the supplier by giving details in the form and pressing Remove

Supplier Info

E-Commerce Inventory Management System

Welcome! Enter Supplier Details

[Back](#) [Search Item](#)

Supplier Number

Supplier Name

Contact Number

[Add](#)
[Update](#)
[Remove](#)
[Select](#)

Supplier Number	Supplier Name	Contact Number
1	Rajashri Traders	+919123934887
2	Saikiran Agencies	+919287118909
3	AJB Wholesale Store	+919568478587
4	Jaishankar Traders	+919236542874

Updating the supplier details and displaying

SAMPLE CODING

The module wise lines of code have been given below:

Module	Lines of Code
Login	200
Inventory	350
Sales	500
Purchases	250
Extras	100

Code Snippets:

1) Sales Module

```
from tkinter import *
import tkinter.messagebox as messagebox
from tkinter import ttk
import mysql.connector as SQL
import datetime

root = Tk()
root.geometry("1920x1080")
root.title("Login Page")
folder_location = "C:\\Users\\raj20\\OneDrive\\Desktop\\Software Engeniering\\Inventory Mng System\\Final\\"
con = SQL.connect(host="127.0.0.1", user="root", password="asdfghjkl", database="IMS", port = 3306,
auth_plugin='mysql_native_password', autocommit=False)
cursor=con.cursor()

def put_in_table(tree_obj, data):
    for i in tree_obj.get_children():
        tree_obj.delete(i)
    for i in range(len(data)):
        tree_obj.insert(parent="", index=i, iid=i, values=data[i])
    return None

def sales():
    """
    topframe.place_forget()
    mainframe.place_forget()
    buttonframe.place_forget()
    """
    root.title("Sales Page")

def back():
    topframe3.place_forget()
    mainframe3.place_forget()
    formframe.place_forget()
    tableframe1.place_forget()
    searchframe.place_forget()
    start()

def search_order():
    query = "SELECT * FROM orders "
    if searchvar.get() != "":
```

```

        query += "WHERE OrderID LIKE \"% {} %\" OR Platform LIKE \"% {} %\";".format(searchvar.get(),
searchvar.get().capitalize())
        # print(query)
        cursor.execute(query)
        data = cursor.fetchall()
        put_in_table(tree, data)
        last_search = search_order
        return None

def search_using_form_data():
    # print(orderid.get(), platform.get(), orderdate.get(), "if check: ", include_checkbox_chk.get(),
processed_chk.get(), packed_chk.get(), sent_chk.get(), returned_chk.get(), money_recieved_chk.get(),
returndate.get())
    query = "SELECT * from orders "

    def check_where(Query):
        if "WHERE" != Query[21:26]:
            Query += "WHERE "
        return Query

    if orderid.get() != "" and orderid.get() != "Mandatory Entry":
        query += "WHERE OrderID LIKE \"% {} %\" AND ".format(orderid.get())

    if platform.get() != "" and platform.get() != "Mandatory Entry":
        query = check_where(query)
        query += "Platform LIKE \"% {} %\" AND ".format(platform.get().capitalize())

    if include_checkbox_chk.get() == 1:
        query = check_where(query)
        query += "Processed = { } AND Packed = { } AND Sent = { } AND Returned = { } AND MoneyRecieved
= { } AND "
        query = query.format(processed_chk.get(), packed_chk.get(), sent_chk.get(), returned_chk.get(),
money_recieved_chk.get())

    if orderdate.get() != "" and orderdate.get() != "YYYY-MM-DD":
        query = check_where(query)
        query += "OrderDate LIKE \"% {} %\" AND ".format(orderdate.get())

    if returned_chk.get() == 1 and include_checkbox_chk.get() == 1:
        if returndate.get() != "" and returndate.get() != "YYYY-MM-DD":
            query = check_where(query)
            query += "ReturnedDate LIKE \"% {} %\" AND ".format(returndate.get())

    if query[-4:] == "AND ":
        query = query[0:-4]
    # print(query)
    cursor.execute(query)
    data = cursor.fetchall()
    put_in_table(tree, data)
    last_search = search_using_form_data
    return None

last_search = search_order

def add_new_order():
    # print(orderid.get(), platform.get(), orderdate.get(), processed_chk.get(), packed_chk.get(), sent_chk.get(),
returned_chk.get(), money_recieved_chk.get(), returndate.get())
    query = "INSERT INTO orders VALUES ("
    if orderid.get() == "" or orderid.get() == "Mandatory Entry":
        orderid.set("Mandatory Entry")

```

```

        return None
    query += "\"" + orderid.get() + "\", "

    if platform.get() == "" or platform.get() == "Mandatory Entry":
        platform.set("Mandatory Entry")
        return None
    query += "\"" + platform.get().capitalize() + "\", "

    query += str(processed_chk.get()) + ", "
    query += str(packed_chk.get()) + ", "
    query += str(sent_chk.get()) + ", "
    query += str(returned_chk.get()) + ", "
    query += str(money_recieved_chk.get()) + ", "
    query += "0, " # TotalAmount

    if orderdate.get() == "" or orderdate.get() == "YYYY-MM-DD":
        orderdate.set(datetime.datetime.today().strftime('%Y-%m-%d'))
    query += "\"" + orderdate.get() + "\", "

    if returned_chk.get() == 1:
        if returndate.get() == "" or returndate.get() == "YYYY-MM-DD":
            returndate.set(datetime.datetime.today().strftime('%Y-%m-%d'))
        query += "\"" + returndate.get() + "\", "
    else:
        query += "NULL" + ");"

    # print(query)
    cursor.execute(query)
    last_search()
    return None

def update_order():
    query = "UPDATE orders SET "
    if orderid.get() in ["", "Mandatory Entry"]:
        orderid.set("Mandatory Entry")
        return None

    if platform.get() not in ["", "Mandatory Entry"]:
        query += "Platform = \"{}\", ".format(platform.get().capitalize())

    if orderdate.get() not in ["", "YYYY-MM-DD"]:
        query += "OrderDate = \"{}\", ".format(orderdate.get())

    query += "Processed = {}, Packed = {}, Sent = {}, Returned = {}, MoneyRecieved = {}, "
    query = query.format(processed_chk.get(), packed_chk.get(), sent_chk.get(), returned_chk.get(),
money_recieved_chk.get())

    if returned_chk.get() == 1:
        if returndate.get() in ["", "YYYY-MM-DD", "None"]:
            returndate.set("Mandatory Entry")
            return None
        else:
            query += "ReturnedDate = \"{}\", ".format(returndate.get())

    query = query[:-2] + " WHERE OrderID = \"{}\";".format(orderid.get())
    # print(query)
    cursor.execute(query)
    last_search()
    return None

```

```

def remove_order():
    query = "DELETE FROM orders WHERE OrderID = \"\"
    if orderid.get() in [ "", "Mandatory Entry"]:
        orderid.set("Mandatory Entry")
        return None
    query += orderid.get() + "\";"
    cursor.execute(query)
    last_search()
    return None

def double_click(event = None):

    def search_orderid():
        order_id.set(order_id.get())
        if order_id.get() in [ "", "Mandatory Entry"]:
            order_id.set("Mandatory Entry")
            return None
        query = "SELECT ProductID, Qty, Amount FROM order_details WHERE OrderID =
\"{}\";".format(order_id.get())
        cursor.execute(query)
        data = cursor.fetchall()
        put_in_table(tree2, data)
        return None

    def back():
        root1.destroy()
        return None

    def add_order_detail():
        # orderid.get(), productid.get(), quantity.get()
        if order_id.get() in [ "", "Mandatory Entry"]:
            order_id.set("Mandatory Entry")
            return None

        if productid.get() in [ "", "Mandatory Entry"]:
            productid.set("Mandatory Entry")
            return None

        if quantity.get() in [ "", "Mandatory Entry"]:
            quantity.set("Mandatory Entry")
            return None
        try:
            int(quantity.get())
        except:
            quantity.set("Only Number")
            return None
        query = "INSERT INTO order_details (OrderID, ProductID, Qty) VALUES (\"{}\", \"{}\", {});"
        query = query.format(order_id.get(), productid.get(), quantity.get())
        cursor.execute(query)
        search_orderid()
        return None

    def delete_product_from_detail():
        if order_id.get() in [ "", "Mandatory Entry"]:
            order_id.set("Mandatory Entry")
            return None

        if productid.get() in [ "", "Mandatory Entry"]:
            productid.set("Mandatory Entry")
            return None

```



```

query = "DELETE FROM order_details WHERE OrderID = \"{}\" AND ProductID = \"{}\";"
query = query.format(order_id.get(), productid())
cursor.execute(query)
return None

def select_record():
    if tree2.focus() != "":
        itm = tree2.focus()
        itm = tree2.item(itm, 'values')
        productid.set(itm[0])
        quantity.set(itm[1])
    return None

root1 = Toplevel()
root1.geometry("1050x500+200+200")
root1.title("Orders Window")

formframe = Frame(root1, width=500, height=350, bg="#FFFFFF")
formframe.place(x=20, y=80)

tableframe = LabelFrame(root1, width=600, height=400)
tableframe.place(x=550, y=40)

im = PhotoImage(file = folder_location + "images\\searchdesc.png", master=root1)
im = im.subsample(8, 8)
searchbtn = Button(formframe, text=" Search Order ", font="Georgia 11 bold", bg="white", bd=5,
image=im,compound=LEFT, command=search_orderid)
searchbtn.place(x=330, y=15, height=35)
scrollbarx = Scrollbar(tableframe, orient=HORIZONTAL)
scrollbary = Scrollbar(tableframe, orient=VERTICAL)
tree2 = ttk.Treeview(tableframe,columns=("Product ID", "Quantity", "Amount"),selectmode="browse",
height=18, yscrollcommand=scrollbary.set,xscrollcommand=scrollbarx.set)
column_width = {"#0": 0, "#1": 150, "#2": 150, "#3": 150}
column_heading=["Product ID", "Quantity", "Amount"]
for i in column_width:
    tree2.column(i, stretch=YES, minwidth=0, width=column_width[i])
for i in column_heading:
    tree2.heading(i, text=i, anchor=CENTER)
tree2.grid(row=1, column=0, sticky="W")
scrollbary.config(command=tree2.yview)
scrollbarx.grid(row=2, column=0, sticky="we")
scrollbarx.config(command=tree2.xview)
scrollbary.grid(row=1, column=1, sticky="ns", pady=30)
formframe.focus_set()

order_id = StringVar()
if tree.focus() != "":
    itm = tree.item(tree.focus(), "values")
    print(itm)
    itm = itm[0]
    order_id.set(itm)
productid = StringVar()
quantity= StringVar()
va = 20
column_heading1 = ["Order ID", "Product ID", "Quantity"]
for i in column_heading1:
    Label(formframe, text=i, font="Georgia 11 bold", bg="#FFFFFF").place(x=0, y=va)
    va += 60

```

```

Entry(formframe, textvariable=order_id, font="georgia 11 bold", bg="#FFFFFF", width=20).place(x=102,
y=20,height=30)
Entry(formframe, textvariable=productid, font="georgia 11 bold", bg="#FFFFFF", width=20).place(x=102,
y=80,height=30)
Entry(formframe, textvariable=quantity, font="georgia 11 bold", bg="#FFFFFF", width=20).place(x=102,
y=140,height=30)
Button(formframe, text="Add", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=add_order_detail).place(x=10, y=261)
Button(formframe, text="Remove", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=delete_product_from_detail).place(x=190, y=261)
Button(formframe, text="Select", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=select_record).place(x=370, y=261)
backbtn = Button(root1, text="Back", font="Georgia 11 bold", bg="white", command=back)
backbtn.place(x=20, y=40)

root1.config(bg="sky blue")
root1.mainloop()

return None

def select_order():
    itm = tree.focus()
    if itm == "":
        return None
    else:
        itm = tree.item(itm, "values")
        orderid.set(itm[0])
        platform.set(itm[1])
        processed_chk.set(itm[2])
        packed_chk.set(itm[3])
        sent_chk.set(itm[4])
        returned_chk.set(itm[5])
        money_recieved_chk.set(itm[6])
        orderdate.set(itm[8])
        returndate.set(itm[9])
    return None

topframe3 = LabelFrame(root, width=2000, height=140, bg="sky blue", borderwidth=0,
highlightthickness=0)
topframe3.place(x=0, y=0)
title = Label(topframe3, text='E-Commerce Inventory Management System', fg='snow', bg="sky
blue",anchor='center')
title.config(font="Georgia 30 bold")
title.place(x=320, y=30)

mainframe3 = LabelFrame(root, width=800, height=125, bg='sky blue', borderwidth=0, highlightthickness=0)
mainframe3.place(x=350, y=100)
welcome = Label(mainframe3, text='Welcome! Update Pending Order Details', fg='white', bg='sky
blue',anchor='center')
welcome.config(font="Georgia 20 bold")
welcome.place(x=200, y=30)
backbtn = Button(mainframe3, text="Back", font="Georgia 11 bold", bg="white", command=back)
backbtn.place(x=50, y=30)

formframe = Frame(root, width=500, height=480, bg="#FFFFFF")
formframe.place(x=50, y=285)

tableframe1 = LabelFrame(root, width=1300, height=400)
tableframe1.place(x=600, y=285)

```

```

searchframe=Frame(root,width=1000,height=70,bg="sky blue")
searchframe.place(x=530,y=200)
searchvar=StringVar()
searchentry=Entry(searchframe,textvariable=searchvar,font="georgia 15",width=25,bg="white")
searchentry.place(x=0,y=30,height=40)
im5 = PhotoImage(file=folder_location + "images\\vieworder.png", master=root)
im5 = im5.subsample(20, 20)
search_button = Button(searchframe, text="Search", font="Georgia 11 bold", bg="white",bd=5,
image=im5,compound=LEFT, command=search_order)
search_button.place(x=320, y=30, height=40)
search_from_form = Button(searchframe, text="Search via Form", font="Georgia 11 bold", bg="white",bd=5,
image=im5,compound=LEFT, command=search_using_form_data)
search_from_form.place(x=420, y=30, height=40)
order_detail_btn = Button(searchframe, text="Order Details", font="Georgia 11 bold", bg="white",bd=5,
image=im5,compound=LEFT, command=double_click)
order_detail_btn.place(x=760, y=30, height=40)
include_checkbox_chk = IntVar()
include_checkbox_chk.set(0)
cbut4 = Checkbutton(searchframe, text="Include CheckBoxes in Search", font="georgia 11 bold",
bg="white",variable=include_checkbox_chk).place(x=420, y=10, anchor="w")

scrollbarx = Scrollbar(tableframe1, orient=HORIZONTAL)
scrollbary = Scrollbar(tableframe1, orient=VERTICAL)
column_width = {"#0": 0, "#1": 70, "#2": 70, "#3": 70, "#4": 70, "#5": 70, "#6": 70, "#7": 100, "#8": 100,
"#9": 100, "#10": 100}
column_heading = ["Order ID", "Platform", "Processed", "Packed", "Sent", "Returned", "Money Recieved",
"Total Amount", "Order Date", "Returned Date"]
tree = ttk.Treeview(tableframe1, columns=column_heading, selectmode="browse", height=18,
yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)
for i in column_width:
    tree.column(i, stretch=NO, minwidth=0, width=column_width[i])
for i in column_heading:
    tree.heading(i, text=i, anchor=CENTER)
tree.grid(row=1, column=0, sticky="W")
scrollbary.config(command=tree.yview)
scrollbarx.grid(row=2, column=0, sticky="we")
scrollbarx.config(command=tree.xview)
scrollbary.grid(row=1, column=1, sticky="ns", pady=30)
tree.bind("<Double-1>", double_click)

formframe.focus_set()
orderid = StringVar()
platform = StringVar()
orderdate = StringVar(value = "YYYY-MM-DD")
returndate=StringVar(value = "YYYY-MM-DD")
va = 20
li = ["Order ID", "Platform", "Order Date"]
for i in range(3):
    Label(formframe, text=li[i], font="Georgia 11 bold", bg="#FFFFFF").place(x=0, y=va)
    va += 60
    Entry(formframe, textvariable=orderid, font="georgia 11 bold", bg="#FFFFFF",
width=20).place(x=142,y=20, height=30)
    Entry(formframe, textvariable=platform, font="georgia 11 bold", bg="#FFFFFF",
width=20).place(x=142,y=80,height=30)
    Entry(formframe, textvariable=orderdate, font="georgia 11 bold", bg="#FFFFFF",
width=20).place(x=142,y=140,height=30)
    processed_chk = IntVar()
    processed_chk.set(0)
    packed_chk = IntVar()
    packed_chk.set(0)

```

```

sent_chk = IntVar()
sent_chk.set(0)
returned_chk = IntVar()
returned_chk.set(0)
money_recieved_chk=IntVar()
money_recieved_chk.set(0)
cbut1 = Checkbutton(formframe, text="Processed", font="georgia 11 bold",
bg="white",variable=processed_chk).place(x=80, y=200, anchor="w")
cbut2 = Checkbutton(formframe, text="Packed", font="georgia 11 bold",
bg="white",variable=packed_chk).place(x=200, y=200, anchor="w")
cbut3 = Checkbutton(formframe, text="Sent", font="georgia 11 bold",
bg="white",variable=sent_chk).place(x=80, y=260, anchor="w")
cbut4 = Checkbutton(formframe, text="Returned", font="georgia 11 bold",
bg="white",variable=returned_chk).place(x=140, y=300, anchor="w")
cbut5 = Checkbutton(formframe, text="Money Recieved", font="georgia 11 bold",
bg="white",variable=money_recieved_chk).place(x=200, y=260, anchor="w")
Label(formframe,text="Returned Date",font="Georgia 11 bold", bg="#FFFFFF").place(x=0,y=340)
Entry(formframe, textvariable=returndate,font="georgia 11 bold", bg="#FFFFFF",
width=20).place(x=142,y=340,height=30)
Button(formframe, text="Add", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=add_new_order).place(x=10, y=401)
Button(formframe, text="Update", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=update_order).place(x=130, y=401)
Button(formframe, text="Remove", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=remove_order).place(x=250, y=401)
Button(formframe, text="Select", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=select_order).place(x=370, y=401)
root.config(bg="sky blue")
root.mainloop()

return None

sales()

```

2) Supplier Information

```

from tkinter import *
import tkinter.messagebox as messagebox
from tkinter import ttk
import mysql.connector as SQL

root = Tk()
root.geometry("1920x1080")
root.title("Login Page")
folder_location = "C:\\Users\\raj20\\OneDrive\\Desktop\\Software Engeniering\\Inventory Mng System\\Final\\"
con = SQL.connect(host="127.0.0.1", user="root", password="asdfghjkl", database="IMS", port = 3306,
auth_plugin='mysql_native_password', autocommit=False)
cursor=con.cursor()

def put_in_table(tree_obj, data):
    for i in tree_obj.get_children():
        tree_obj.delete(i)
    for i in range(len(data)):
        tree_obj.insert(parent="", index=i, iid=i, values=data[i])
    return None

def supplier_info():
    """
    topframe.place_forget()
    mainframe.place_forget()

```

```

buttonframe.place_forget()
"""
root.title("Supplier Info")

def back():
    topframe11.place_forget()
    mainframe11.place_forget()
    formframe.place_forget()
    tableframe.place_forget()
    start()
    pass

def add_new_supplier():
    if supno.get() in ["", "Mandatory Entry"]:
        cursor.execute("SELECT COUNT(*) FROM supplier;")
        supno.set(str(int(cursor.fetchall()[0][0] + 1)))

    if supname.get() in ["", "Mandatory Entry"]:
        supname.set("Mandatory Entry")
        return None

    if contactno.get() in ["", "Mandatory Entry"]:
        contactno.set("Mandatory Entry")
        return None
    try:
        int(contactno.get()[1:])
    except:
        contactno.set("+91<Numbers and No spaces>")
        return None

    query = "INSERT INTO supplier VALUES (\\"{}\\", \\"{}\\", \\"{}\\");".format(supno.get(), supname.get(),
contactno.get())
    cursor.execute(query)
    search_supplier()
    return None

def update_supplier():
    query = "UPDATE supplier SET "
    if supno.get() in ["", "Mandatory Entry"]:
        supno.set("Mandatory Entry")
        return None

    if supname.get() not in ["", "Mandatory Entry"]:
        query += "SupName = \\"{}\\", ".format(supname.get())

    if contactno.get() not in ["", "Mandatory Entry"]:
        try:
            int(contactno.get()[1:])
            query += "ContactNo = \\"{}\\", ".format(contactno.get())
        except:
            contactno.set("+91<Numbers and No spaces>")
            return None

    query = query[:-2] + " WHERE SupNo = \\"{}\\"".format(supno.get())
    cursor.execute(query)
    search_supplier()
    return None

def delete_supplier():
    if supno.get() in ["", "Mandatory Entry"]:

```

```

        supno.set("Mandatory Entry")
        return None

    query = "DELETE FROM supplier WHERE SupNo = '{}{}';".format(supno.get())
    cursor.execute(query)
    search_supplier()
    return None

def select_supplier():
    if tree.focus() != "":
        itm = tree.focus()
        itm = tree.item(itm, "values")
        supno.set(itm[0])
        supname.set(itm[1])
        contactno.set(itm[2])
    return None

def search_supplier():
    query = "SELECT * FROM supplier "
    if searchvar.get() not in ["", "SupName or SupNo"]:
        query += "WHERE SupNo LIKE '{}{}%' OR SupName LIKE '{}{}%';".format(searchvar.get(),
searchvar.get())
    cursor.execute(query)
    data = cursor.fetchall()
    put_in_table(tree, data)
    return None

    topframe11 = LabelFrame(root, width=2000, height=140, bg="sky blue", borderwidth=0,
highlightthickness=0)
    topframe11.place(x=0, y=0)
    title = Label(topframe11, text='E-Commerce Inventory Management System', fg='snow', bg="sky
blue", anchor='center')
    title.config(font="Georgia 30 bold")
    title.place(x=320, y=30)

    mainframe11 = Frame(root, width=800, height=125, bg='sky blue', borderwidth=0, highlightthickness=0)
    mainframe11.place(x=350, y=100)
    welcome = Label(mainframe11, text='Welcome! Enter Supplier Details', fg='white', bg='sky
blue', anchor='center')
    welcome.config(font="Georgia 20 bold")
    welcome.place(x=200, y=30)

    searchframe = Frame(root, width=720, height=70, bg="sky blue")
    searchframe.place(x=400, y=200)
    im = PhotoImage(file = folder_location + "images\\searchdesc.png", master=root)
    im = im.subsample(8, 8)
    backbtn = Button(searchframe, text="Back", font="Georgia 11 bold", bg="white", command=back)
    backbtn.place(x=0, y=20)
    searchvar = StringVar(value="SupName or SupNo")
    searchentry = Entry(searchframe, textvariable=searchvar, font="georgia 15", width=25, bg="white")
    searchentry.place(x=180, y=20, height=40)
    searchbtn = Button(searchframe, text=" Search Item ", font="Georgia 11 bold", bg="white", bd=5,
image=im, compound=LEFT, command=search_supplier)
    searchbtn.place(x=500, y=20, height=40)

    formframe = Frame(root, width=500, height=300, bg="#FFFFFF")
    formframe.place(x=100, y=275)

    tableframe = LabelFrame(root, width=1300, height=400)
    tableframe.place(x=700, y=275)

```

```

scrollbarx = Scrollbar(tableframe, orient=HORIZONTAL)
scrollbary = Scrollbar(tableframe, orient=VERTICAL)
column_width = {'#0': 0, '#1': 200, '#2': 200, '#3': 200}
column_heading = ["Supplier Number", "Supplier Name", "Contact Number"]
tree = ttk.Treeview(tableframe, columns=column_heading, selectmode="browse", height=18,
yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)
for i in column_width:
    tree.column(i, stretch=NO, minwidth=0, width=column_width[i])
for i in column_heading:
    tree.heading(i, text=i, anchor=CENTER)
tree.grid(row=1, column=0, sticky="W")
scrollbary.config(command=tree.yview)
scrollbarx.grid(row=2, column=0, sticky="we")
scrollbarx.config(command=tree.xview)
scrollbary.grid(row=1, column=1, sticky="ns", pady=30)
formframe.focus_set()
supno = StringVar()
supname = StringVar()
contactno = StringVar()
va = 20
for i in column_heading:
    Label(formframe, text=i, font="Georgia 11 bold", bg="#FFFFFF").place(x=0, y=va)
    va += 60
Entry(formframe, textvariable=supno, font="georgia 11 bold", bg="#FFFFFF", width=20).place(x=142,
y=20, height=30)
Entry(formframe, textvariable=supname, font="georgia 11 bold", bg="#FFFFFF",
width=20).place(x=142, y=80, height=30)
Entry(formframe, textvariable=contactno, font="georgia 11 bold", bg="#FFFFFF",
width=20).place(x=142, y=140, height=30)

Button(formframe, text="Add", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2, command=add_new_supplier).place(x=10, y=200)
Button(formframe, text="Update", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2, command=update_supplier).place(x=130, y=200)
Button(formframe, text="Remove", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2, command=delete_supplier).place(x=250, y=200)
Button(formframe, text="Select", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2, command=select_supplier).place(x=370, y=200)

root.config(bg="sky blue")
root.mainloop()

pass

supplier_info()

```

3) Inventory Module:

```

from tkinter import *
import tkinter.messagebox as messagebox
from tkinter import ttk
import mysql.connector as SQL

root = Tk()
root.geometry("1920x1080")
root.title("Login Page")
folder_location = "C:\\Users\\raj20\\OneDrive\\Desktop\\Software Engeniering\\Inventory Mng System\\Final\\"
con = SQL.connect(host="127.0.0.1", user="root", password="asdfghjkl", database="IMS", port = 3306,
auth_plugin='mysql_native_password', autocommit=False)
cursor=con.cursor()

```

```

def put_in_table(tree_obj, data):
    for i in tree_obj.get_children():
        tree_obj.delete(i)
    for i in range(len(data)):
        tree_obj.insert(parent="", index=i, iid=i, values=data[i])
    return None

def inventory_page():
    """
    topframe.place_forget()
    mainframe.place_forget()
    buttonframe.place_forget()
    """
    root.title("Inventory Page")

    def back():
        tableframe.place_forget()
        formframe.place_forget()
        topframe1.place_forget()
        mainframe1.place_forget()
        searchframe.place_forget()
        start()
        return None

    def searchprod():
        query = "SELECT * FROM Stock "
        if (searchvar.get() != "" and searchvar.get() != "ItemID or ItemName"):
            query += "WHERE ItemID LIKE '%" + searchvar.get() + "%' OR ItemName LIKE '%" + searchvar.get()
+ "%';"
            cursor.execute(query)
            reply = cursor.fetchall()
            put_in_table(tree, reply)
            return None

    def add_new_stock():
        # print(itemid.get(), itemname.get(), costprice.get(), quantity.get(), supplierinfo.get())
        query = "INSERT INTO stock VALUES (\\"

        if itemid.get() == "":
            cursor.execute("SELECT COUNT(*) FROM stock;")
            itemid.set(str(int(cursor.fetchall()[0][0]) + 1))
            query += itemid.get() + "\\"

        if itemname.get() == "" or itemname.get() == "Mandatory Input":
            itemname.set("Mandatory Input")
            return None
        else:
            query += ", \\" + itemname.get() + "\\"

        if costprice.get() == "":
            costprice.set("Mandatory Input")
            return None
        else:
            try:
                int(costprice.get())
            except:
                costprice.set("Only a Number")
                return None
            query += ", " + costprice.get()

```



```

if quantity.get() == "":
    quantity.set("0")
else:
    try:
        int(quantity.get())
    except:
        quantity.set("Only a Number")
        return None
query += ", " + quantity.get()

if supplierinfo.get() == "" and supplierinfo.get() == "Mandatory Input":
    supplierinfo.set("Mandatory Input")
    return None
else:
    query += ", \"\" + supplierinfo.get() + \"\";"
cursor.execute(query)
searchprod()
return None

def delproduct():
    if (itemid.get() == "" or itemid.get() == "Mandatory Entry"):
        itemid.set("Mandatory Entry")
    else:
        query = "DELETE FROM stock WHERE ItemID = \"\" + itemid.get() + \"\";"
        # print(query)
        cursor.execute(query)
        searchprod()
    return None

def update_product():
    query = "UPDATE stock SET "
    if (itemid.get() == "" or itemid.get() == "Mandatory Entry"):
        itemid.set("Mandatory Entry")
        return None

    if itemname.get() != "":
        query += "ItemName = \"\" + itemname.get() + \"\", "

    if costprice.get() != "":
        query += "CostPrice = " + costprice.get() + ", "

    if quantity.get() != "":
        query += "Qty = " + quantity.get() + ", "

    if supplierinfo.get() != "":
        query += "SupNo = " + supplierinfo.get() + ", "

    if (itemname.get() != "" or costprice.get() != "" or quantity.get() != "" or supplierinfo.get() != ""):
        query = query[:-2] + " WHERE ItemID = \"\" + itemid.get() + \"\";"
        print(query)
        cursor.execute(query)
        searchprod()
    return None

def select_record():
    itm = tree.focus()
    if itm == "":
        return None
    else:

```

```

        itm = tree.item(itm, "values")
        itemid.set(itm[0])
        itemname.set(itm[1])
        costprice.set(itm[2])
        quantity.set(itm[3])
        supplierinfo.set(itm[4])
    return None

topframe1 = LabelFrame(root, width=2000, height=140, bg="sky blue", borderwidth=0,
highlightthickness=0)
topframe1.place(x=0, y=0)
title = Label(topframe1, text='E-Commerce Inventory Management System', fg='snow', bg="sky
blue", anchor='center')
title.config(font="Georgia 30 bold")
title.place(x=320, y=30)

mainframe1 = LabelFrame(root, width=800, height=125, bg='sky blue', borderwidth=0, highlightthickness=0)
mainframe1.place(x=350, y=100)
welcome = Label(mainframe1, text='Welcome to Inventory Page', fg='white', bg='sky blue', anchor='center')
welcome.config(font="Georgia 20 bold")
welcome.place(x=200, y=30)

formframe = Frame(root, width=500, height=450, bg="#FFFFFF")
formframe.place(x=100, y=315)
tableframe = LabelFrame(root, width=1300, height=400)
tableframe.place(x=700, y=315)

searchframe = Frame(root, width=720, height=70, bg="sky blue")
searchframe.place(x=400, y=200)
im = PhotoImage(file = folder_location + "images\\searchdesc.png", master=root)
im = im.subsample(8, 8)
backbtn = Button(searchframe, text="Back", font="Georgia 11 bold", bg="white", command=back)
backbtn.place(x=0, y=20)
searchbtn = Button(searchframe, text=" Search Item ", font="Georgia 11 bold", bg="white", bd=5,
image=im, compound=LEFT, command=searchprod)
searchbtn.place(x=450, y=20, height=40)
searchvar = StringVar(value="ItemID or ItemName")
searchentry = Entry(searchframe, textvariable=searchvar, font="georgia 15", width=25, bg="white")
searchentry.place(x=130, y=20, height=40)

scrollbarx = Scrollbar(tableframe, orient=HORIZONTAL)
scrollbary = Scrollbar(tableframe, orient=VERTICAL)
tree = ttk.Treeview(tableframe, columns=("Item ID", "Item Name", "Cost Price", "Quantity", "Supplier
Number"), selectmode="browse", height=18, yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)
column_width = {"#0": 0, "#1": 100, "#2": 100, "#3": 150, "#4": 150, "#5": 150}
column_heading = ['Item ID', 'Item Name', 'Cost Price', 'Quantity', 'Supplier Number']
for i in column_width:
    tree.column(i, stretch=YES, minwidth=0, width=column_width[i])
for i in column_heading:
    tree.heading(i, text=i, anchor=CENTER)
tree.grid(row=1, column=0, sticky="W")
scrollbary.config(command=tree.yview)
scrollbarx.grid(row=2, column=0, sticky="we")
scrollbarx.config(command=tree.xview)
scrollbary.grid(row=1, column=1, sticky="ns", pady=30)
formframe.focus_set()
itemid = StringVar()
itemname = StringVar()
costprice = StringVar()

```

```

quantity = StringVar()
supplierinfo = StringVar()
va = 20
for i in range(5):
    Label(formframe, text=column_heading[i], font="Georgia 11 bold", bg="#FFFFFF").place(x=0, y=va)
    va += 60
    Entry(formframe, textvariable=itemid, font="georgia 11 bold", bg="#FFFFFF", width=20).place(x=142,
y=20,height=30)
    Entry(formframe, textvariable=itemname, font="georgia 11 bold", bg="#FFFFFF", width=20).place(x=142,
y=80,height=30)
    Entry(formframe, textvariable=costprice, font="georgia 11 bold", bg="#FFFFFF", width=20).place(x=142,
y=140,height=30)
    Entry(formframe, textvariable=quantity, font="georgia 11 bold", bg="#FFFFFF",
width=20).place(x=142,y=200,height=30)
    Entry(formframe, textvariable=supplierinfo, font="georgia 11 bold", bg="#FFFFFF",
width=20).place(x=142,y=260,height=30)
    Button(formframe, text="Add", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=add_new_stock).place(x=10, y=361)
    Button(formframe, text="Update", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=update_product).place(x=130, y=361)
    Button(formframe, text="Remove", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=delproduct).place(x=250, y=361)
    Button(formframe, text="Select", font="Georgia 11 bold", bg="sky blue", bd=5, width=10,
height=2,command=select_record).place(x=370, y=361)

root.config(bg="sky blue")
root.mainloop()

return None

inventory_page()

```