Centre NUMBER - 20184

Stock Control Mangement SYstem

Bilal Fawaz

AQA NEA 2019-20

ALevel

Table of Contents

[Analysis 3](#_Toc35871980)

[The Client – Samir Patel 3](#_Toc35871981)

[Background 3](#_Toc35871982)

[Interview with The Primary Client – Samir Patel 3](#_Toc35871983)

[Observations of The Current System 5](#_Toc35871984)

[The Stock List 5](#_Toc35871985)

[The Method Used 6](#_Toc35871986)

[Data Dictionary & Source and Destination Tables 6](#_Toc35871987)

[Current Data Dictionary Table 6](#_Toc35871988)

[Current Source and Destination Table 6](#_Toc35871989)

[Proposed and Updated Data Dictionary Table 7](#_Toc35871990)

[Proposed and Updated Source and Destination Tables 7](#_Toc35871991)

[Flow Charts 8](#_Toc35871992)

[Entity Relationship Diagram 9](#_Toc35871993)

[Limitations of the Current System 9](#_Toc35871994)

[Objectives 9](#_Toc35871995)

[Exemplary Stock Systems on the market 11](#_Toc35871996)

[Document Design 13](#_Toc35871997)

[What will I be using? 13](#_Toc35871998)

[Why I will be using it? 13](#_Toc35871999)

[Data I Will Be Using 13](#_Toc35872000)

[Data Assignments 13](#_Toc35872001)

[Login Page 14](#_Toc35872002)

[Data Dictionary for SQL Tables 14](#_Toc35872003)

[Brands Table 14](#_Toc35872004)

[Uses Table 14](#_Toc35872005)

[Product\_Info Table 14](#_Toc35872006)

[Place\_Order Table 15](#_Toc35872007)

[Supplier Table 15](#_Toc35872008)

[SQL Statements 15](#_Toc35872009)

[Table Creations 15](#_Toc35872010)

[SQL SELECT Statements 17](#_Toc35872011)

[SQL UPDATE Statements 19](#_Toc35872012)

[SQL DELETE STATEMENTS 19](#_Toc35872013)

[Entity Relationship Diagrams 19](#_Toc35872014)

[Flow Charts and Trees 21](#_Toc35872015)

[Windows Design 23](#_Toc35872016)

[Stock Management 26](#_Toc35872017)

[Add New Product 28](#_Toc35872018)

[Search 29](#_Toc35872019)

[Orders 30](#_Toc35872020)

[Supplier 33](#_Toc35872021)

[Pseudo and Python Code 34](#_Toc35872022)

[CSV File - Usernames and Passwords 34](#_Toc35872023)

[Validations 35](#_Toc35872024)

[Email for Placing Orders 35](#_Toc35872025)

[How I will be Testing 37](#_Toc35872026)

[Technical Solution 38](#_Toc35872027)

[Stock\_Management\_Database\_File 38](#_Toc35872028)

[Environmental Variables 41](#_Toc35872029)

[Main Programme 41](#_Toc35872030)

[Testing 148](#_Toc35872031)

[Evaluation/Appraisal 156](#_Toc35872032)

[Objectives Evaluated 156](#_Toc35872033)

[Feedback from the main client 157](#_Toc35872034)

[Analysis of Feedback 159](#_Toc35872035)

[Extension 159](#_Toc35872036)

[Conclusion of Project 160](#_Toc35872037)

[Bibliography 161](#_Toc35872038)

# Analysis

## The Client – Samir Patel

Name: Samir Patel

Business: Super Care Pharmacy

Intended Project: GUI Stock Management System

My end user for this project will be an old employer of mine during my work experience week for school, Samir Patel. Samir is a pharmacist who owns three pharmacies around the West Midlands. In this project I will be working mainly with employees and information regarding his latest pharmacy, Super Care Pharmacy. The Pharmacy opened recently in late 2014 and has since served thousands of patients and their prescription orders. They stock a wide range of medicine and store this alphabetically on the shelves just behind the front counter. Samir is excited to work with me to help develop a better system for his business, so that he can possibly implement it on his other two pharmacies.

## Background

I worked at the Super Care Pharmacy for a week in November 2016 as part of my school’s work experience week. My tasks involved preparing prescriptions for the pharmacies to then assess before it is sent out, organising pills for the mostly elderly patients for each day of the week, and providing satisfactory customer service on the pharmacy floor. However, the job that had most importance to me was stock management, because without the supply of medication we couldn’t help the hundreds if not thousands of people who come in each week.

When I first was assigned to the stock management and reordering task it was very long and tedious. I noticed they had a very old-fashioned list that you would tick off, which is problematic with the sheer amount of medicine in stock. A lot of the medicine were the same type but in different sizes and you had to make sure you were reading the same size. There was no technology integrated in this stock management system that could make it easy to identify what needs to be reordered, what stock is running dangerously low, and general information about the stock available. So, I believe a new digital system will make the old system redundant and be a more efficient shorter process, that is also more secure, and less frustrating to search for stock on a list. In addition, more time can be used to do the more important tasks such as consulting patients or preparing prescriptions for collection/delivery.

## Interview with The Primary Client – Samir Patel

Q: What’s your existing system for stock management and reordering?

A: Stock management and reordering is done manually. We do not use a programme. Based on

usage, a certain level of stock is kept, and this is topped up manually either daily or weekly

basis.

Q: What are its benefits?

A: Prices of medicines fluctuate, sometimes daily. We have up to six different suppliers. With this

system we are able to select appropriate suppliers

Q; What are its drawbacks?

A: Time consuming

Q: What features are most important to you?

A: Flexibility in being able to choose different suppliers, a list that makes it easier to search for the medication and edit.

Q: Which existing features do you find useful?

A: Being able to use different suppliers

Q: What would you like to see implemented into the system?

A: being able to know current stock balances

Q: How many members of staff will be using the new system?

A: 3

Q: How would you organise the medicine in terms of tables in databases, any specific groups of

medicines? i.e. Blood Thinners; (anticoagulants, antiplatelet drugs, and fibrinolytics)

A: Currently the medicines are organised alphabetically and that is important

Q: How important will the overall layout or design of the new system be to you?

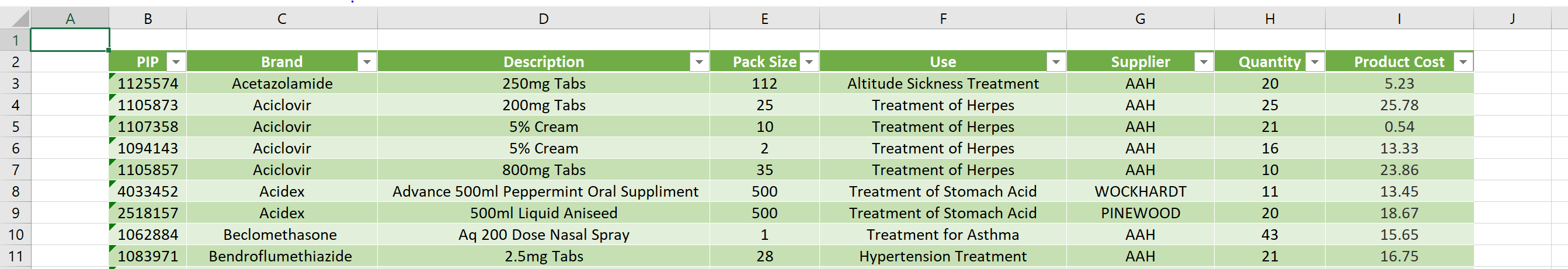
A: Layout will be important, it would have to be alphabetically

## Observations of The Current System

### The Stock List

**** At first look at the system it is quite old-fashioned pen and paper. The business needs a new more innovative way to take control of their stock. It can take some time to find the medication on the list and pencil everything in, its very time consuming and that time can be in better ways.

The current system has around 1,347 different variations and types of medications. The list includes the Product (Product Name), the Pack Size and PIP Code, which is the unique product code allocated to a specific product and includes its description. The list is not ordered alphabetically and there is no total quantity or pricing for the items, making it even harder to manage. At first glance the list was an eye sore with a lot of information displayed very simply.

Another issue I discovered, was that the Product column already had the pack size in (#), the dosage, and the supplier attached to the end. This was very disorientating to read and was just a complete mess as there was no need to have a pack size column if the pack size is already listed in the product name. Therefore, to organise the data in one easy to read table, I separated the data into columns so that there is individual data in the rows of each product. These included the PIP, Brand, Description, Pack Size, and Supplier. I also went into the pharmacy and counted the quantity of each item in the list as well as the current price, to add a Quantity and Product Cost columns. Moreover, with the pharmacists advise I labelled the uses and created a Use column for each product

The Pharmacy is in close partnership with a handful of suppliers for all the medication, and each product comes from only one supplier.

I displayed this new stock list to the owner for approval and he said that it was a “massive improvement”, that it was “easier to read”, and will “help employees with the specific information on each product, especially useful for new recruits”.

### The Method Used

The lists are printed every week, month, or when needed. A tick and a quantity is then added next to what needs to be ordered, and this process can take a whopping hour or more to go through each individual item of the 1,347, find it on the list and tick it off.

The data is not stored and rather anything low on stock is just reordered through suppliers’ booklets. Therefore, the amount of each product is not known until they do their weekly or monthly restocks, which can waste so much time.

## Data Dictionary & Source and Destination Tables

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| PIP Code | Stores the name of the player | Integer | Infinite | 1105857 | None |
| Product | Stores the product name, as well as the dosage, pack size, and supplier | String | Infinite | Paracetamol 500mg Tabs (20) AAH | None |
| Pack Size | Stores the pack size of each product | Integer | Infinite | 20 | None |

### Current Data Dictionary Table

### Current Source and Destination Table

|  |  |  |
| --- | --- | --- |
| **What is it** | **Source** | **Destination** |
| PIP Code | Pharmacist entering in when a new product is ordered | Excel Spread sheet |
| Product | Pharmacist entering the name of the product | Excel Spread sheet |
| Pack Size | Pharmacists entering the pack size of the product | Excel Spread sheet |

### Proposed and Updated Data Dictionary Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| Username | Stores the Username of each User | String | Infinite | Bfawaz1 | Unique |
| Password | Stores the Passwords of each Use, corresponding to a Username | String | At least 7 characters | Bones471£ | At least 7 characters long, including at least one; Capital Letter, Number, Special character from ($,#,@,!,\*,.,\_,&,!,£,^) |

#### For the Proposed Login System:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| PIP | Stores the name of the player | Integer | Infinite | 1105857 | Not Null |
| Brand | Stores the Brand Name | String | Infinite | Paracetamol | Not Null |
| Description | Stores the information about each product including the dosage | String | Infinite | 500mg Tabs | Not Null |
| Pack Size | Stores the pack size of each product | Integer | Infinite | 20 | Bigger than 0 |
| Use | Stores the potential use for each of the products | String | Infinite | Pain Killer | Not Null |
| Supplier | Stores the Supplier | String | Infinite | AAH | Not Null |
| Quantity | Stores the amount of each product on the store shelf | Integer | Infinite | 21 | Not Negative |
| Product Cost | Stores the cost on the product in British Pounds (£) | Float | Infinite | 2.50 | Not Null, two decimal places (because its money) |

#### For the Main Programme Stock Management System:

### Proposed and Updated Source and Destination Tables

#### For the Proposed Login System:

|  |  |  |
| --- | --- | --- |
| **What is it** | **Source** | **Destination** |
| Username | Admin Created | CSV File |
| Password | Admin Created to corresponding Username | CSV File |

#### For the Main Programme Stock Management System:

|  |  |  |
| --- | --- | --- |
| **What is it** | **Source** | **Destination** |
| PIP | From the individual product | SQL Data Base Table |
| Brand | The Brand of the product | SQL Data Base Table |
| Description | The label or information from a product | SQL Data Base Table |
| Pack Size | The label of the product | SQL Data Base Table |
| Use | The pharmacists advised use | SQL Data Base Table |
| Supplier | Supplier Sheet provided by the owner | SQL Data Base Table |
| Quantity | Counting the number of each products in shop | SQL Data Base Table |
| Product Cost | From the main supplier sheet | SQL Data Base Table |

## Flow Charts

**Current Stock List Process**

Go to the medicine shelves and find the corresponding product on the list

Print out a new list

Start

Finish

Yes

Add a cross (X) next to the product and list the quantity

Is it the last product on the list?

Is there a sufficient stock for that product?

Yes

No



No

Go to the next product on the list (ordered alphabetical)

Write the quantity needed, and add a Check (√)

## Entity Relationship Diagram

PIP Code

Product

Pack Size

One Product has one unique PIP code, and each product can come in or have many different pack sizes.

## Limitations of the Current System

The current system is a very outdated one, that must be updated to fit in with the environment of advance technology today. There are many limitations that can be eliminated by introducing a digital system.

The current limitations include the ability to search for a medication within a small period of given time. As the list is very unorganised and at times all over the place, it can make it harder to identify or recognise where the medication is from. Secondly the list does not provide what the medication is used for or categorise it in any way. If for example a new junior employee has started a job in the pharmacy, his knowledge of medicine is very limited in the beginning, providing a system that could categorise the medicine in terms of what it treats or how it is taken can be used to narrow down the search and save countless hours over a year. Furthermore

Filters for search and grouped categorically, admin,

## Objectives

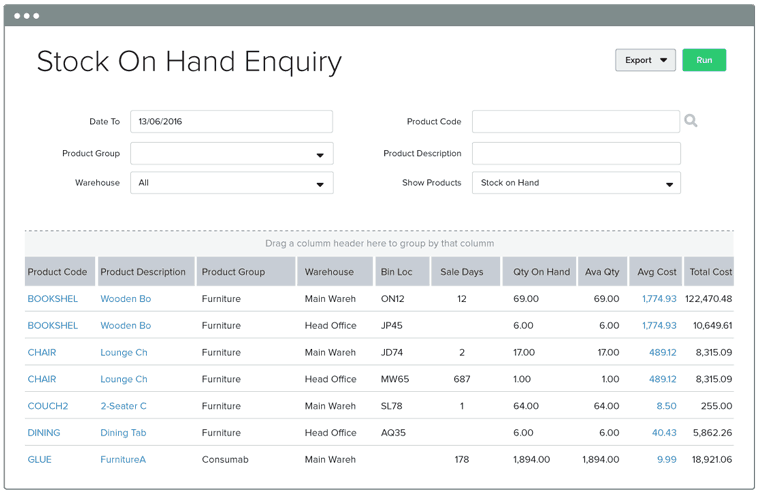
To help me decide on reasonable and realistic objectives to set for myself that also a line with my client’s vision for the programme I used the SMART method for each Proposed Objective. The SMART Method is an acronym for Specific, Measurable, Achievable, Relevant, and Time bound goals and I had to take this into account when creating my SMART Objectives.

The GUI Stock Management System I will be creating for this pharmacy will have restricted access, stock that is simply displayed and able to be filtered, a built-in place order function that sends a generated email straight to the supplier, and a simple GUI graphical layout which is easy to use. I will also incorporate a cancel order function that allows the user to pick a previously placed order and delete/cancel it by sending a cancelation email to the supplier. As well as a search engine that searches for specific brands, products, uses, or suppliers, and if a product is searched similar products are displayed to the user.

1. Restricted access – To secure the information of the pharmacy in terms of orders and times as well as employee personal information.
2. I will need to create a log in page for the employee or user to login in before gaining access to the information
3. An Admin Account
   1. which can control the information such as passwords acting as a master key, in case a new user needs to be added (where the Username is not taken and the password is strong), or an existing employee/user has forgotten their username
   2. that should also delete existing users, in case an employee leaves, so that they do not have access to the data of the pharmacy
4. Graphically simple and easy to understand or follow – Simple GUI
   1. Very simple design using Tkinter and laid out so employees can use with ease.
   2. The colour scheme must not be too distracting but simple and fit the theme of the pharmacy
   3. Easy to navigate through tabs and pages
   4. Fast responding, little to no delay or lag time and all widgets work accordingly
5. Sorted Data – Ordered lists for the products with different sorting techniques
   1. Stock Management Tab:
      1. Sorting stock by a click of a button, either by A-Z or Z-A, and a choice to then sort that by highest to lowest quantity or lowest to highest quantity
   2. Order History:
      1. Orders can be sorted either from newest or oldest placed date and time
6. Modifying Stock
   1. The ability for a specific product to be deleted, added, or quantity of the stock/product changed
7. Search Engine for the database
   1. A new tab with a search bar that allows the user to select “Search By” options via a drop-down menu so that the user can search by PIP, Brand, Use, or Supplier
   2. If a user selects the PIP therefore searching by specific products, the page will also display similar products (products with the same use).
8. Orders – To re-order existing products and check the order history
9. Built in separate tab/page that when clicked opens a page with two tabs for the user to decide, “Place Order” or “Order History”
10. When the “Place Order” tab is chosen
    1. The user has the option to select a product, brand, and quantity to be ordered from a drop-down menu. The total cost is then identified, and the order can be placed via a confirm order button which sends a generated email to the supplier of the product.
11. The previous orders placed is stored in “Order History”, this will include
    1. A cancel order button that when an OrderID is selected the information of that order is displayed and the user has the option to delete/cancel the order which will delete the order from the database and send a cancelation email to the supplier.
12. When an order number is clicked a new window opens displaying what has been ordered and the necessary information.

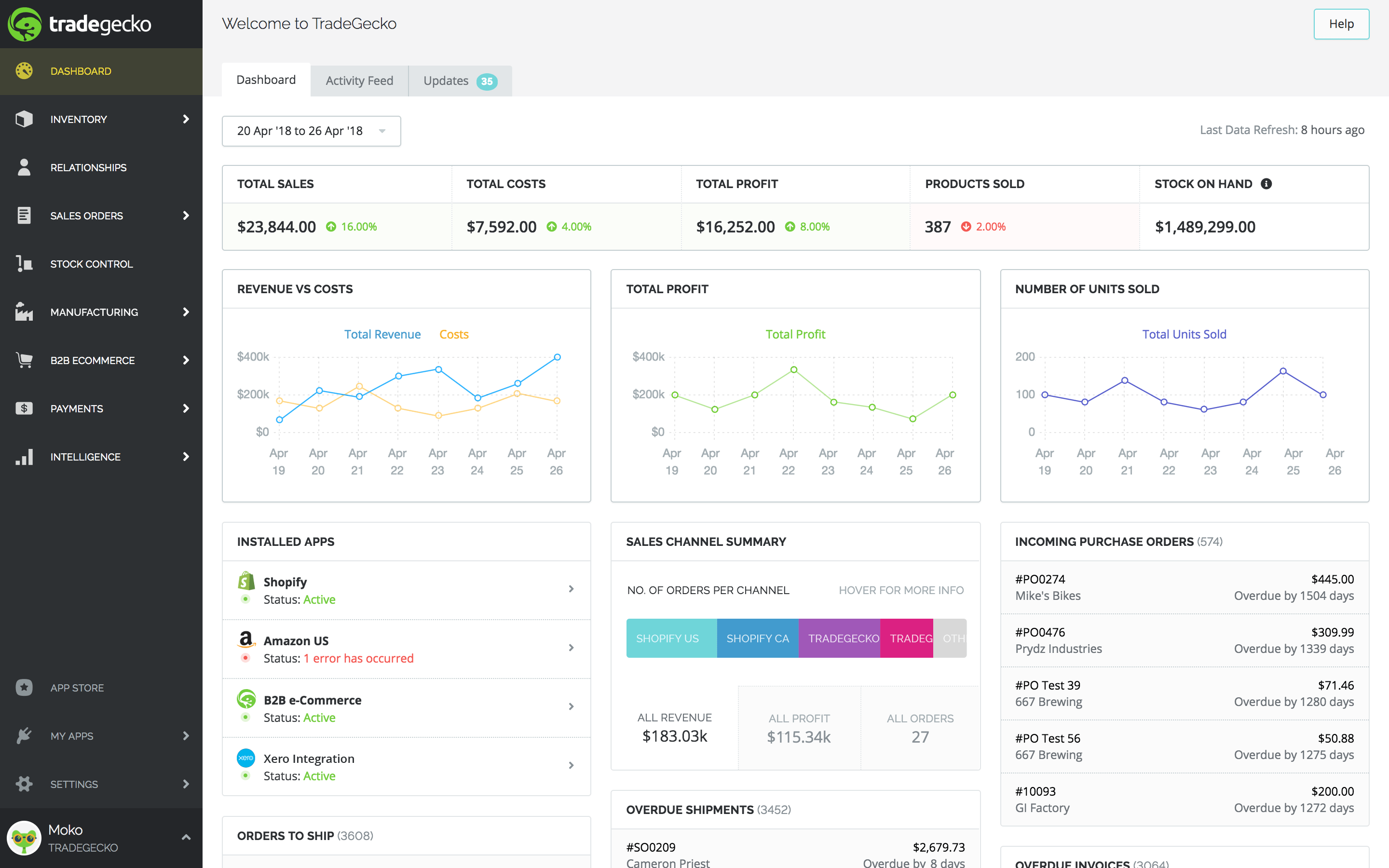
## Exemplary Stock Systems on the market

**Unleashed Software[[1]](#footnote-1)** is one example of a currently thriving stock management system. It is a cloud-based app that allows the freedom to enhance the management and movement of products. With Unleashed Software a business can track and control production, warehouses, suppliers, and sales.

The system allows its users to accurately track real-time stock across multiple locations. This allows the business to gain a visibility and new perspective into all stock management and transaction processes across stores or warehouses all over the world. Another reason to implement a digital system allows the businesses to possibly plan and make better data-driven decisions. Lastly Unleashed integrates well with many diverse types of financial software from eCommerce, point of sales, and even accounting. All of these key features provide for a much-desired end-to-end business

The overall interface and design of Unleashed Software allows for a comfortable experience navigating through the platform and find information faster and more efficiently than if it was done on paper. The colour schemes are reasonable and not too intense to the eye and the customer satisfaction with the software is around 95%.

**Trade Gecko[[2]](#footnote-2)** is a bigger more recognised name in the inventory and order management software industry. It is extremely powerful and built for high growth e-commerce, wholesale, and multinational brands such as Shopify, Xero, and Amazon. The platform has an automated track process for it sales of products, as well as its manufacturing and restocking information, even across multiple locations. It can boost business efficiency by syncing orders from different sales and inventory with the product. The system even tracks all the customers an supplier in one place, all with purchase histories to further improve customer-specific insights, all easily accessible through a click of a button. Trade Gecko’s Platform will even auto generate sales, inventory, and business reports, all to allow for a better business decision in real time.



In a similar way to Unleashed, Trade Gecko has a fantastic and oddly similar GUI. It is fast responding, organised, and simple. It also has a customer satisfaction of over 90%. They are both some of the most advanced Inventory/Stock Management Systems on the market, used by companies that handle millions if not billions of pounds.

Here is a **chart[[3]](#footnote-3)** comparing the top 5, two of which are Trade Gecko and Unleashed, with Gecko securing the number 1 spot and Unleashed just below. Trade Gecko has a greater Functionality and Ease of Use rating which is a big deciding factor in choosing an inventory/stock management system.

# Document Design

## What will I be using?

For this project I have decided to create the programme using python and its built-in functions, Tkinter for the GUI and SQL for the database.

Python is a high-level object orientated programming language, it is used worldwide by major corporations such as YouTube, Facebook, and Amazon. It one of the fastest growing programming languages, being the 3rd most popular programming language in 2018, just under C and Java.

Tkinter is a built-in function in python meaning it can be called and used without having to go through a separate application. The call function used is (import Tkinter), and this is called at the very being of the programme as python has a top to bottom approach. Tkinter is a type of GUI or graphical user interface, it provides a display in which the users of the programme can interact, giving a platform for the code to be displaying in a simpler fashion that we are more familiar in seeing.

SQL is another built in function in python, it is called using (import sqlite3). Coming from a language called MySQL it has a different type of syntax to that of python. SQL is mainly used for creating databases small and large, and it does this by creating tables with attributes, and linking them through primary and foreign keys.

## Why I will be using it?

Python is very dynamic and flexible, it is very simple to create functions using subroutines, and has a wide range of built in functions to choose from. It is the all in one type of programe, and the simplicity of it makes it easier for other users such as Samir to understand.

Tkinter is already a built-in function and has a wide range of GUI features that I believe will further enhance the look and feel of my programme. Some of these features includes the addition of drop-down menus, the ability to change the background and foreground colours, changing font and its size, and the ability to create multiple tabs and windows.

SQL is also already a built-in function. It is quite simple to create a database that is linked and intertwined. The use of the primary key as a unique identifier, can be turned into a foreign key in other tables to show a link. Modifying attribute, and data within tables are straight forward and everything is displayed in an easy to understand way.

## Data I Will Be Using

From the original 1,347 products I was given, I have chosen to narrow it down due to time constraints and practicability and in order to achieve my SMART objectives. I will be cutting down my SQL database to 149 products, however I have added room for a function to add new products, suppliers, and update prices and information. So that if I had time or needed to, I could add the missing data. As for now I am focusing on the program’s aesthetic, functionality, and safety.

## Data Assignments

For my Login Window I will be storing my Usernames and Passwords in a CSV file, which will be saved in the same folder as my main programme.

### Login Page

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Field Purpose** | **Field Type** | **Field Size** | **Example Data** | **Validation** |
| Username | Stores the Username of each User | String | Infinite | Bfawaz1 | Unique |
| Password | Stores the Passwords of each Use, corresponding to a Username | String | At least 7 characters | Bones471£ | At least 7 characters long, including at least one; Capital Letter, Number, Special character from ($,#,@,!,\*,.,\_,&,!,£,^) |

|  |  |  |
| --- | --- | --- |
| **What is it** | **Source** | **Destination** |
| Username | Admin Created | CSV File |
| Password | Admin Created to corresponding Username | CSV File |

## Data Dictionary for SQL Tables

### Brands Table

Brands(BrandID, Brand\_Name)

|  |  |
| --- | --- |
| Attribute | Data Type |
| Brand ID | INTEGER |
| Brand\_Name | TEXT |

### Uses Table

Uses(UseID, Use)

|  |  |
| --- | --- |
| Attribute | Data Type |
| UseID | INTEGER |
| Use | TEXT |

### Product\_Info Table

Product\_Info(PIP, BrandID, Product\_Discription, UseID, Pack\_Size, Product\_Cost, SupplyID, Quantity)

|  |  |
| --- | --- |
| Attribute | Data Type |
| PIP | INTEGER |
| BrandID | INTEGER |
| Product\_Discription | TEXT |
| UseID | INTEGER |
| Pack\_Size | TEXT |
| Product\_Cost | DECIMAL(6,2) |
| SupplyID | INTEGER |
| Quantity | INTEGER |

### Place\_Order Table

Place Order(OrderID, PIP, Order­\_Quantity, Total\_Cost, Date\_Placed)

|  |  |
| --- | --- |
| Attribute | Data Type |
| OrderID | INTEGER |
| PIP | INTEGER |
| Order\_Quantity | INTEGER |
| Total\_Cost | DECIMAL(6,2) |
| Date\_Placed | DATETIME |

### Supplier Table

Supplier(SupplyID, Supply\_Name, Phone, Email, Address)

|  |  |
| --- | --- |
| Attribute | Data Type |
| SupplyID | INTEGER |
| Supply\_Name | TEXT |
| Phone | INTEGER |
| Email | TEXT |
| Address | TEXT |

## SQL Statements

### Table Creations

To create a table, I will be using the DB Browser application, however I need to make sure that if the database is accidentally deleted the table and database are automatically regenerated. Therefore I will be using the “CREATE TABLE IF NOT EXISTS” statements to do this below, in addition I have also identified the data type and any validation such as “NOT NULL” and “UNIQUE”.

#### Brands

''' CREATE TABLE IF NOT EXISTS "Brands" (

"BrandID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

"Brand\_Name" TEXT NOT NULL UNIQUE

);'''

#### Place\_Order

'''CREATE TABLE IF NOT EXISTS "Place\_Order" (

"OrderID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

"PIP" INTEGER NOT NULL,

"Order\_Quantity" INTEGER NOT NULL,

"Total\_Cost" DECIMAL(6 , 2) NOT NULL,

"Date\_Placed" DATETIME NOT NULL,

FOREIGN KEY("PIP") REFERENCES "Product\_Info"("PIP")

);'''

#### Product\_Info

''' CREATE TABLE IF NOT EXISTS "Product\_Info" (

"PIP" INTEGER NOT NULL UNIQUE,

"BrandID" INTEGER NOT NULL,

"Product\_Discription" TEXT NOT NULL,

"UseID" INTEGER NOT NULL,

"Pack\_Size" TEXT NOT NULL,

"Product\_Cost" DECIMAL(6 , 2) NOT NULL,

"SupplyID" INTEGER NOT NULL,

"Quantity" INTEGER NOT NULL,

PRIMARY KEY("PIP"),

FOREIGN KEY("UseID") REFERENCES "Uses"("UseID"),

FOREIGN KEY("SupplyID") REFERENCES "Supplier"("SupplyID"),

FOREIGN KEY("BrandID") REFERENCES "Brands"("BrandID")

);'''

#### Supplier

''' CREATE TABLE IF NOT EXISTS "Supplier" (

"SupplyID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

"Supply\_Name" TEXT NOT NULL UNIQUE,

"Phone" VARCHAR(11) NOT NULL UNIQUE,

"Email" TEXT NOT NULL UNIQUE,

"Address" TEXT NOT NULL UNIQUE

);'''

#### Uses

''' CREATE TABLE IF NOT EXISTS "Uses"(

"UseID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

"Use" TEXT NOT NULL UNIQUE

);'''

### SQL SELECT Statements

My Selection Statements will extrapolated data from the tables where needed, I will be using this to populate all my list boxes that I have to display the data. Below is the SQL Statement I will be using to display the stocklist

#### Selecting Stock List Details

‘’’

SELECT PIP, Brand\_Name, Product\_Discription, Quantity

FROM "Product\_Info", "Brands"

WHERE Product\_Info.BrandID = Brands.BrandID

’’’

Along with the select statements I have used an “ORDER BY” statement to run as a command via a subroutine once a button is pressed. The way the lists are order by depends on the button pressed which will be clearly labelled with text on the button. Below are a couple of examples of how I will be using this.

#### Lowest Stock

‘’’

SELECT PIP, Brand\_Name, Product\_Discription, Quantity

FROM "Product\_Info", "Brands"

WHERE Product\_Info.BrandID = Brands.BrandID

ORDER BY Quantity

‘’’

#### Highest Stock

‘’’

SELECT PIP, Brand\_Name, Product\_Discription, Quantity FROM "Product\_Info", "Brands"

WHERE Product\_Info.BrandID = Brands.BrandID

ORDER BY Quantity desc

‘’’

#### Brand Name A-Z

‘’’

'SELECT PIP, Brand\_Name, Product\_Discription, Quantity FROM "Product\_Info", "Brands"

WHERE Product\_Info. BrandID = Brands.BrandID

ORDER BY Brand\_Name

‘’’

#### Brand Name Z-A

‘’’

'SELECT PIP, Brand\_Name, Product\_Discription, Quantity FROM "Product\_Info", "Brands"

WHERE Product\_Info. BrandID = Brands.BrandID

ORDER BY Brand\_Name desc

‘’’

Order

#### Newest Order

‘’’

SELECT OrderID, PIP, Order\_Quantity, Total\_Cost, Date\_Placed

FROM "Place\_Order"

ORDER BY Date\_Placed desc'

‘’’

#### Oldest Order

‘’’

SELECT OrderID, PIP, Order\_Quantity, Total\_Cost, Date\_Placed

FROM "Place\_Order"

ORDER BY Date\_Placed

‘’’

### SQL UPDATE Statements

To update current data within a table I will be using the “UPDATE” SQL statements to do this. The only update function will be for updating the stock/product quantity for specific products available in the pharmacy, as shown below.

#### Updating the Quantity a Specific Stock/Product

‘’’

UPDATE "Product\_Info" SET Quantity=?

WHERE PIP=?

‘’’,\

(Quantity\_Entered, PIP\_chosen))

### SQL DELETE STATEMENTS

#### Deleting a Product from The Database

‘’’

DELETE FROM "Product\_Info"

WHERE (PIP = (:PIP\_Selected))

‘’’,

{

'PIP\_Selected' : PIP\_Chosen

})

## Entity Relationship Diagrams

Supplier

Uses

Product Info

Brands

Place Order

All the tables have been normalised and have a one to one relationship as seen in the entity relationship diagram above. The suppliers are only UK based. I also kept fixed product costs and order costs for the product. One product per order.

## Flow Charts and Trees

#### Login Window

A close up of a map

Description automatically generated

A simple validation flow chart to make sure the Username and Passwords are within the CSV file and match. Once a user is logged in the main program starts. However if they enter the correct Username and Password for the Admin Account, then an Admin Window is Opened.

#### Admin New User Creation Flow Chart

A close up of a map

Description automatically generated

When a New User is created there will be three entry boxes; Username, Password, and Confirm Password. A Username is first entered and then checked to make sure it doesn’t exist in the CSV File already as a User, if it does the Admin tries again. After the Username is approved the password is created and then confirmed in the “Password” and “Confirm Password” boxes. First the Passwords must meet the criteria to be used as a valid password as displayed in the chart. The Passwords entered are then checked to see if they match. If they do then the Username and Password are now stored in the CSV file along with the other user’s login information.

#### Stock Management Tree

The main “Home Window” will include Tabs by using the Notebook function in tkinter, like an excel spread sheet it allows for a different page on the same window by selecting different tabs. These tabs include; Stock Management, Search, Orders ( which include Place Order and Order History tabs within it), and Suppliers

## Windows Design

Login WindowA screenshot of a cell phone

Description automatically generated

This LogIn window design is very simplistic. There are two entry boxes “Username” and “Password”, where the username and password for the existing user is entered. Then if the username matches with the password the person would be logged in after pressing the green “Log In” button. There is also a “Reset” button to reset anything entered into any of the boxes, as well as this there is an “Exit” button which destroys and quits the program. The LogIn page provides and strengthens protection and security of the pharmacy adding an extra digital layer to block out unauthorised access to confidential information.

A screenshot of a cell phone

Description automatically generated

When Invalid login details are entered a message box is displayed informing the user.

A screenshot of a cell phone

Description automatically generated

The Admin window is used to create new users, checking if the “Username” is not already registered, and using validation to make sure that the password meets a specific standard for security purposes. As well as this a “Confirm Password” entry box is also displayed that compares with the “Password” entered, to see if they match. I have also decided like the Login window to add a few buttons, the first being the “Create New User”, which so long that the criteria is met the User information is saved in the system, to be used. The other button is a reset button to erase all the contents of the entry boxes of “Username”, “Password”, and “Confirm Password”. There is a button called “Display Users”, and when clicked opens up a new window to display all users in the system. Lastly there is an “Exit” button that acts as a return button to go back and open the login window.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A message boxes are displayed on each of the validations for the new user creation

A screenshot of a cell phone

Description automatically generated

The Display window the users currently registered to the system and their passwords, in case they forget, in a horizontal and vertical scrollable listbox.

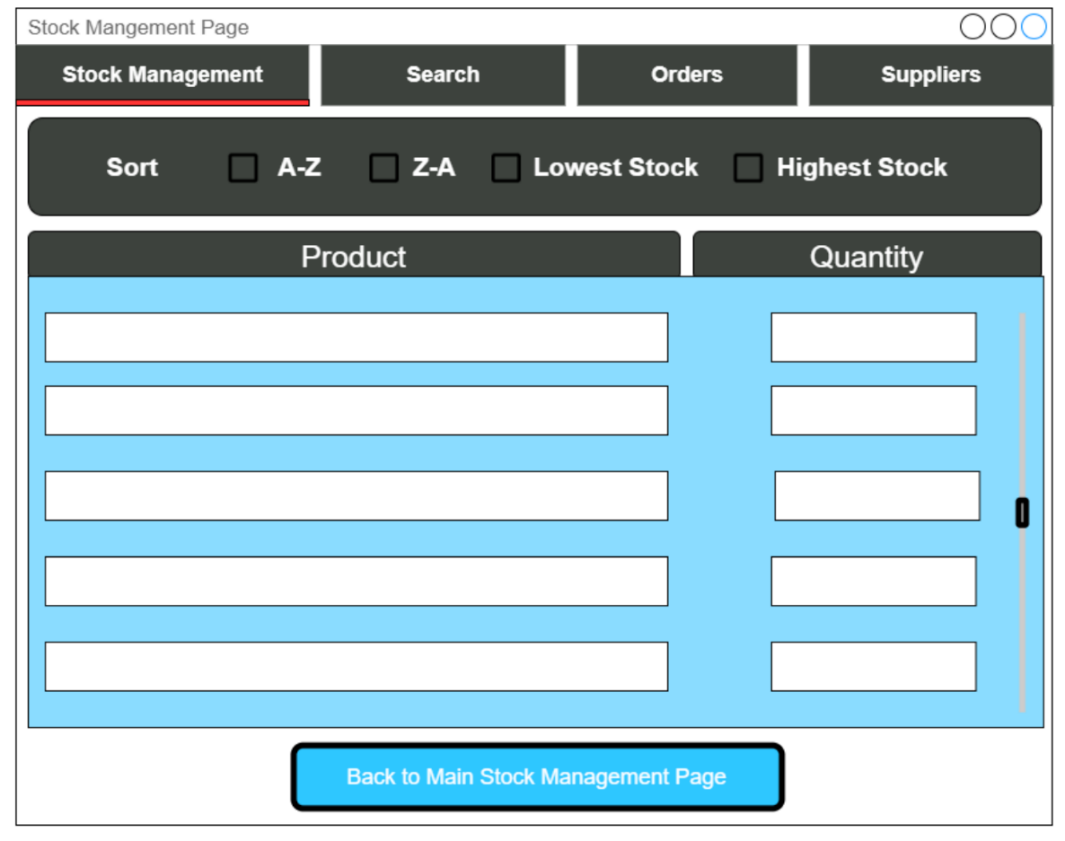
### Stock Management

Clicking on the Stock Management Tab brings the user to this window. Where the Product and the Quantity for each is listed in a scrollable list. The user also has the option to sort the list from alphabetical or reverse-alphabetical order by Product Name, or by lowest and highest stock by comparing quantity of product available. The Product Names are hyperlinks that open a new window when pressed, that displays the products information. There are also the buttons “Edit” and “Add More” at the bottom, which allows either the ability to edit the quantity or to add a new product.

#### Product Information

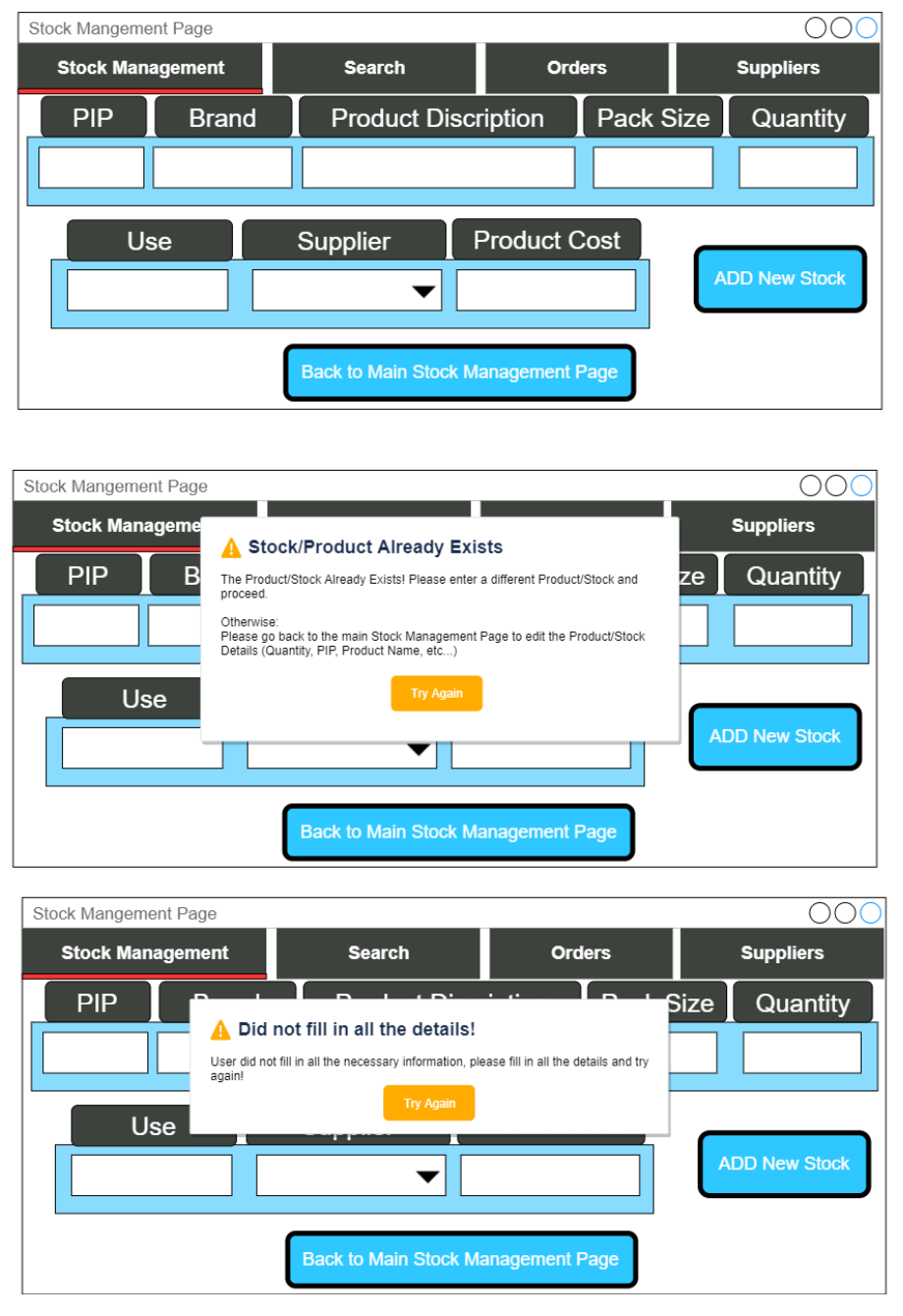
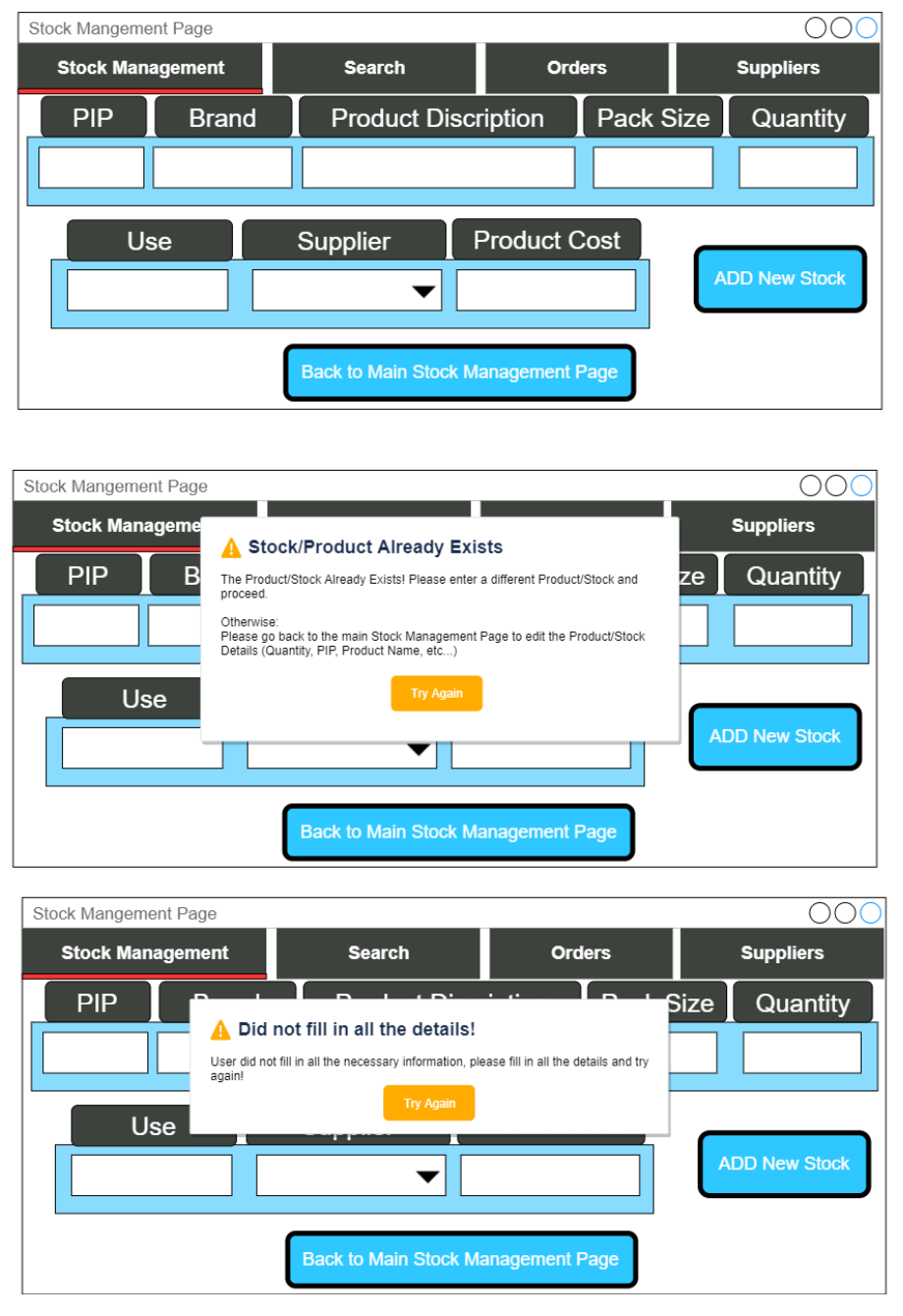
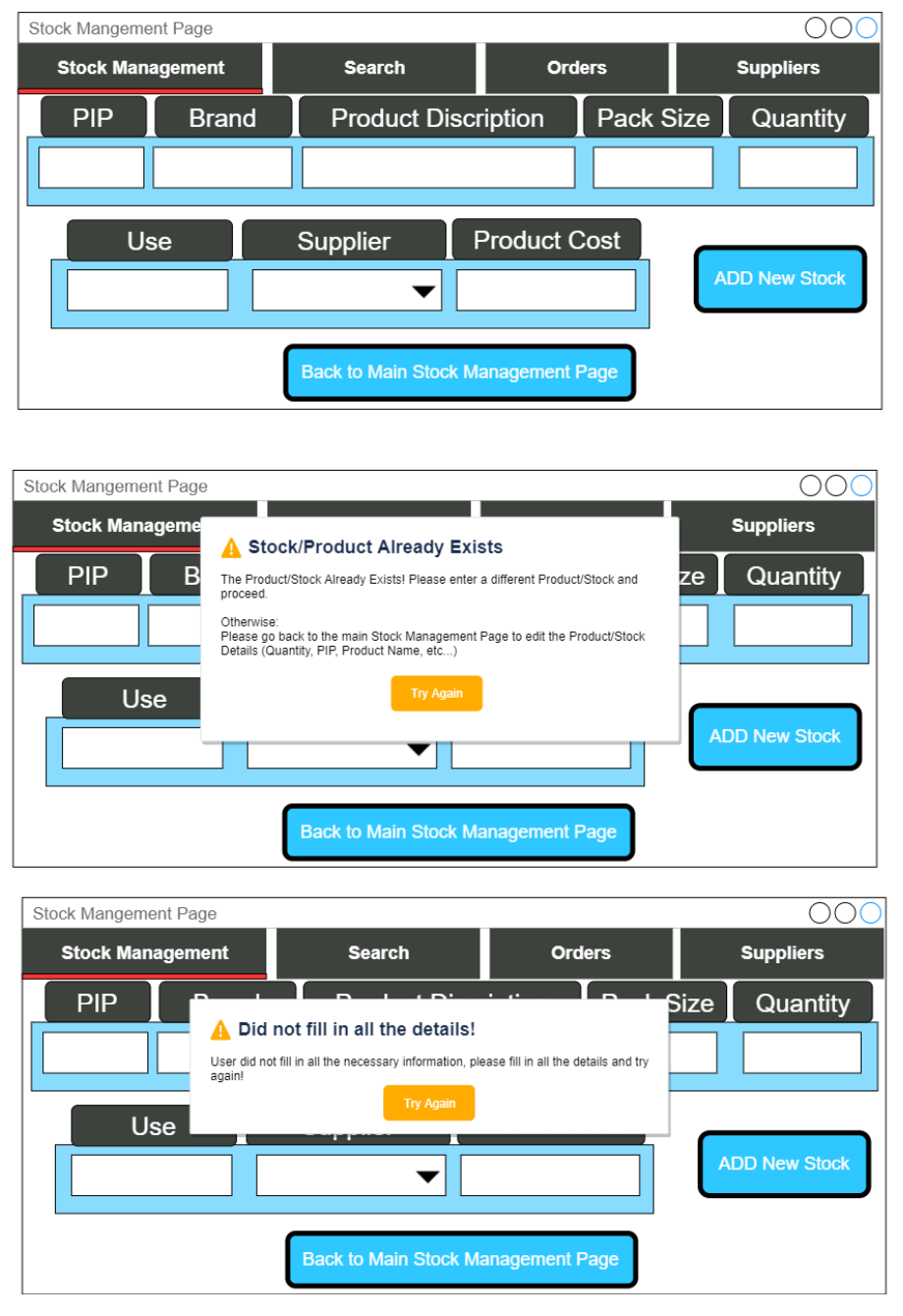
When a product is clicked on, this is new “Product Information” window is displayed. This Displays all the necessary relevant information needed of the product chosen. The supplier is also a hyper link that opens up the supplier’s information window. Lastly there return to the main stock management page

#### Edit Product



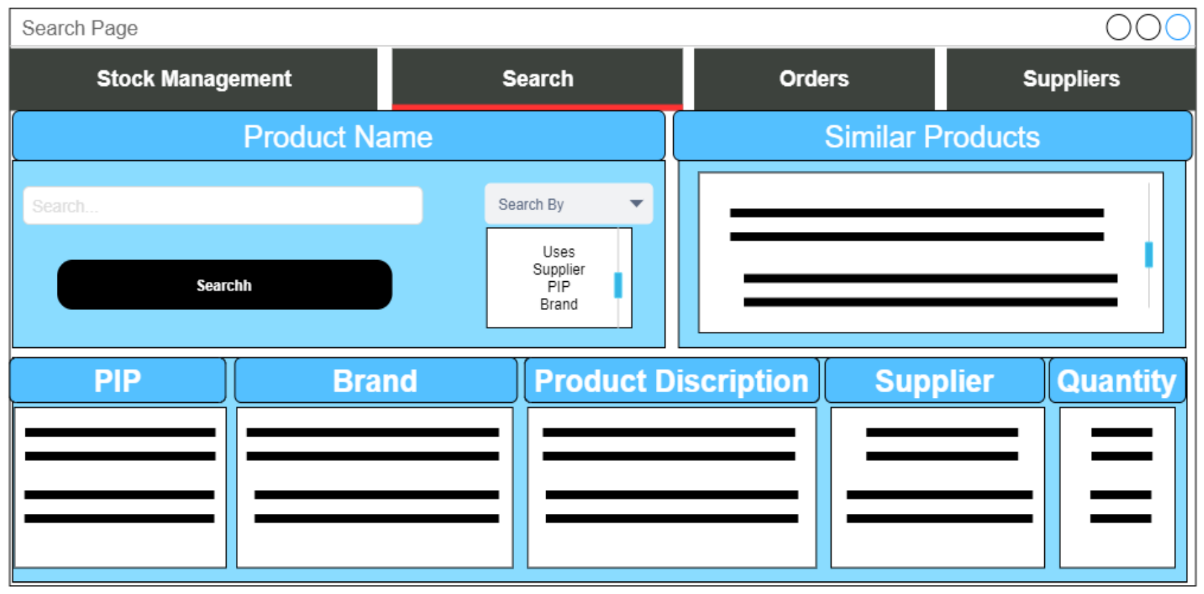
Allows the user to edit the quantity of the stock/product.

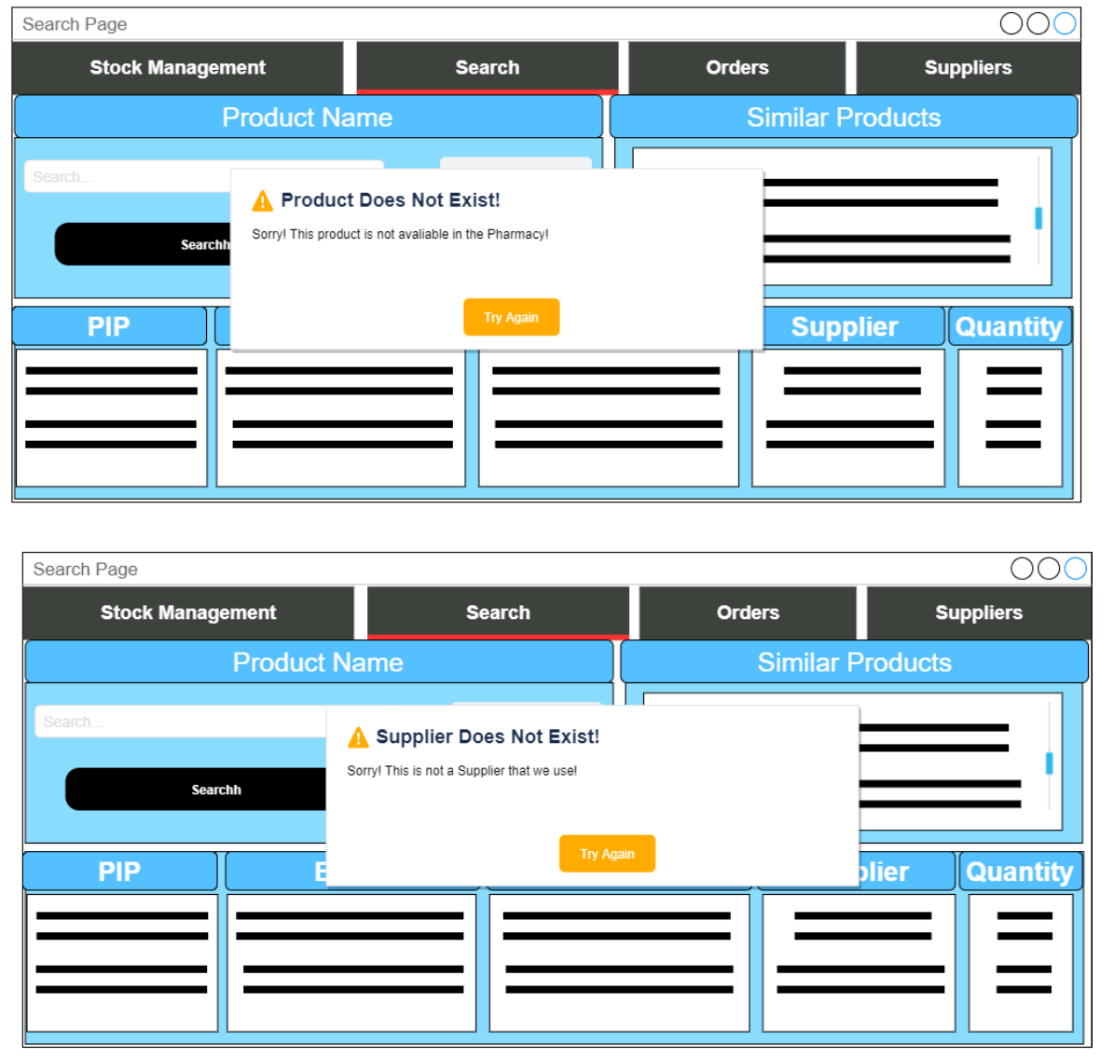
### Add New Product

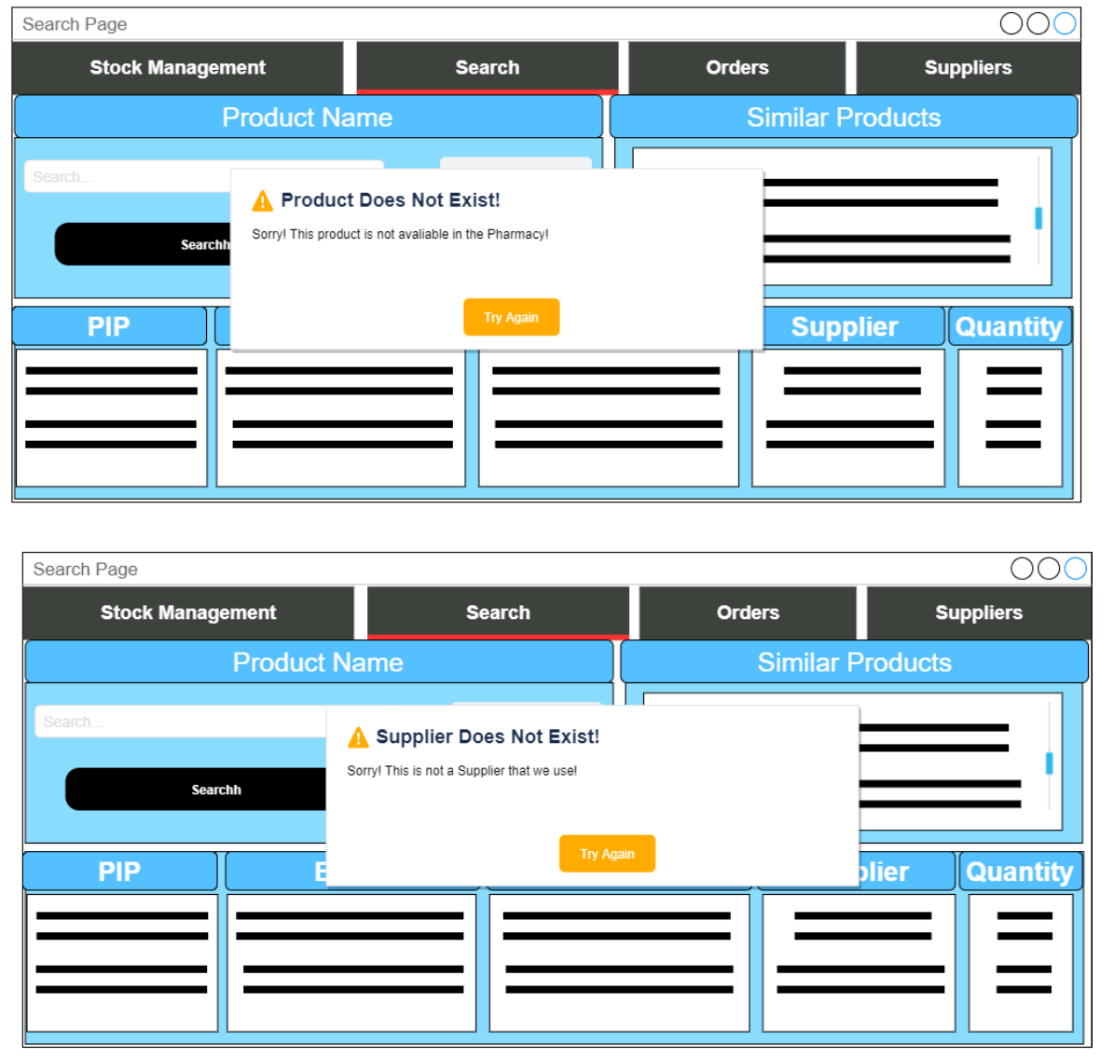


This New window allows the user to add a new product that doesn’t exist, and also include a validation as well as message box that pops up when a product already exists or not all the details are filled in.

### Search







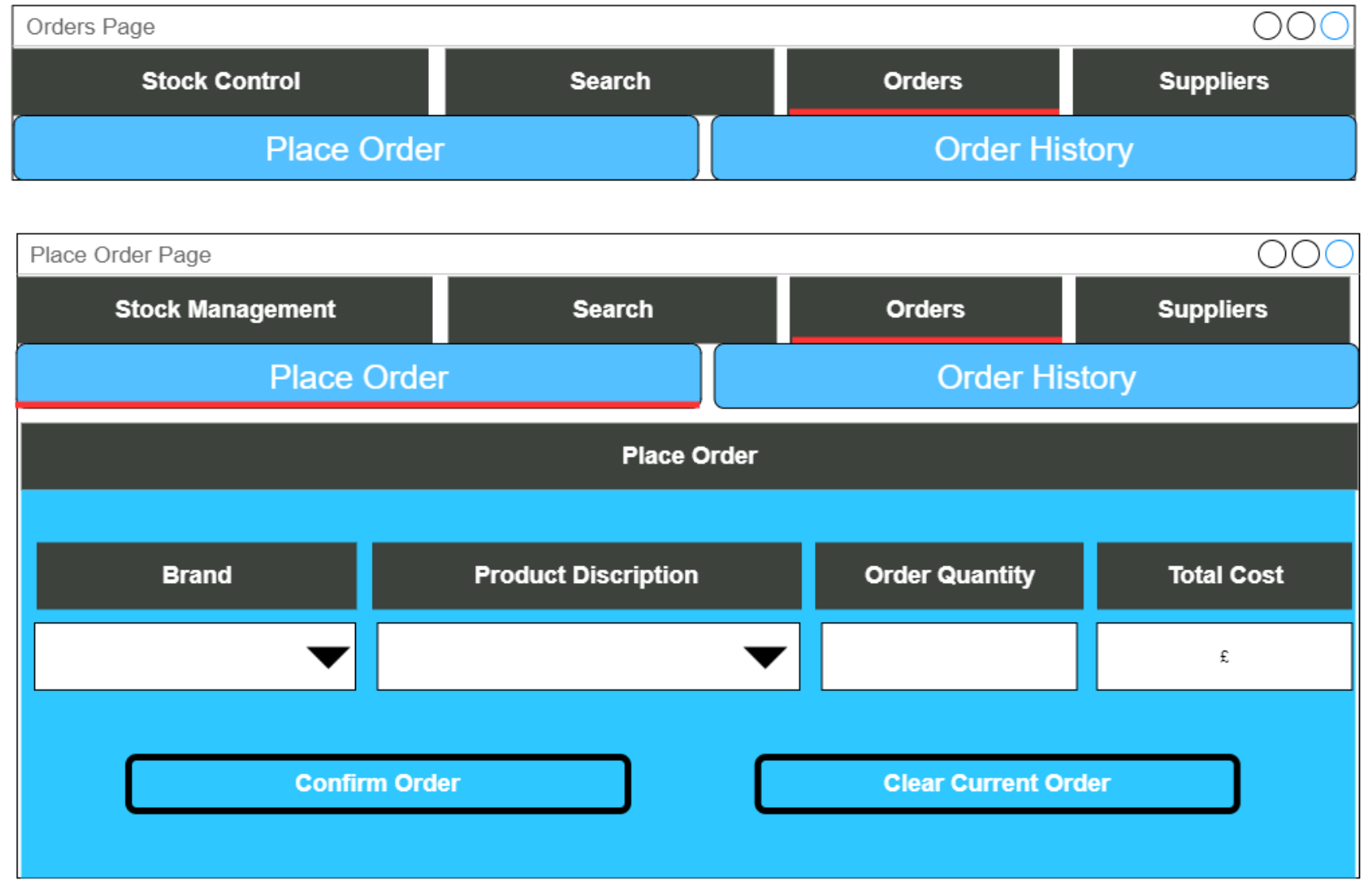


When the search page is chosen the user has the option to search by product use, PIP, Brand, Or Supplier. When either of the options is selected in the dropdown menu the user can then enter the details of the search and click the search button. If the details are in the system, the product or products are displayed in scrollable list boxes. As well as this, similar products are also displayed based on use in a listbox. If the details are not in the database, then an error message is then displayed to the user depending on what the search by selected is.

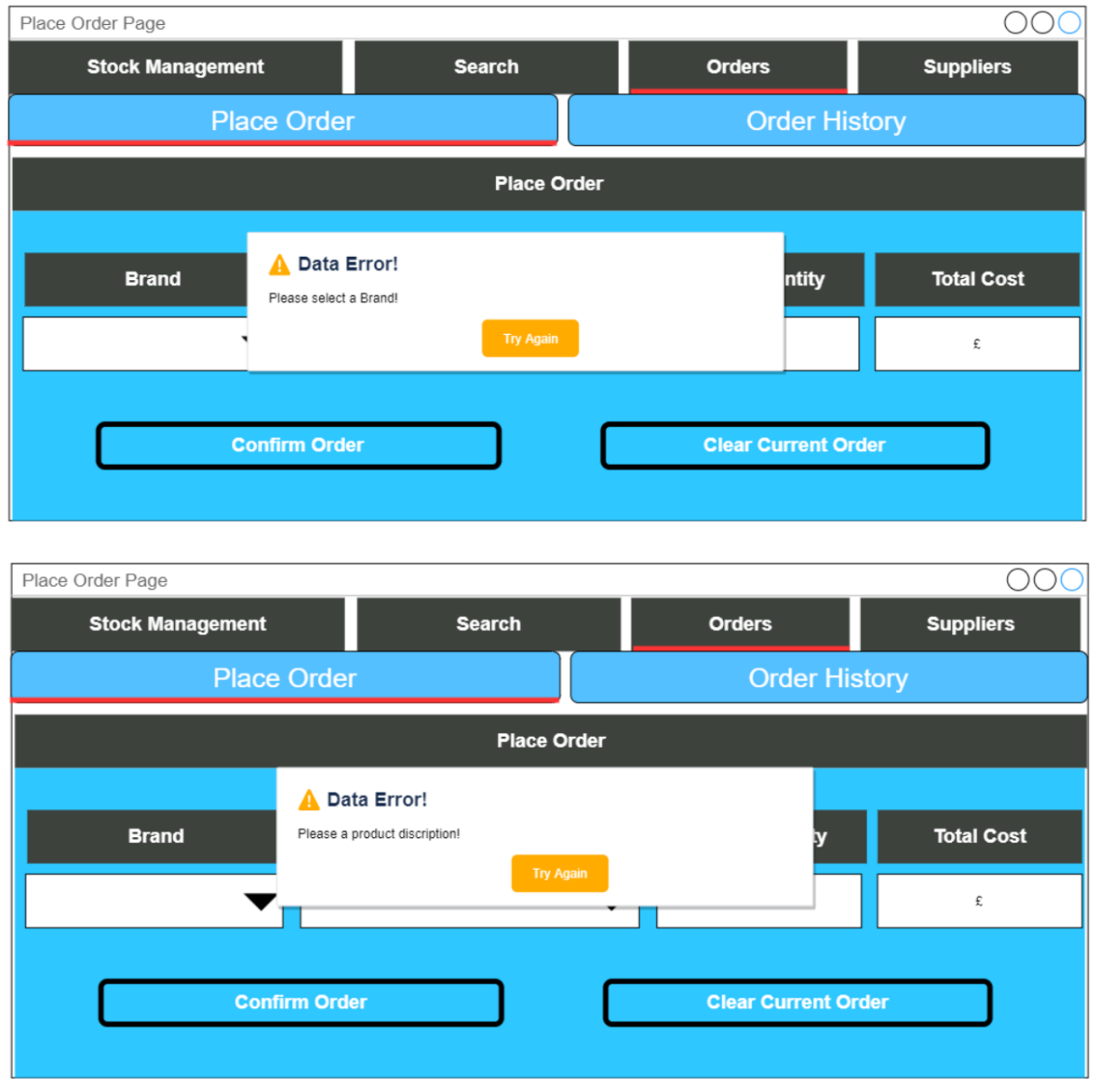
### A picture containing screenshot Description automatically generatedOrders

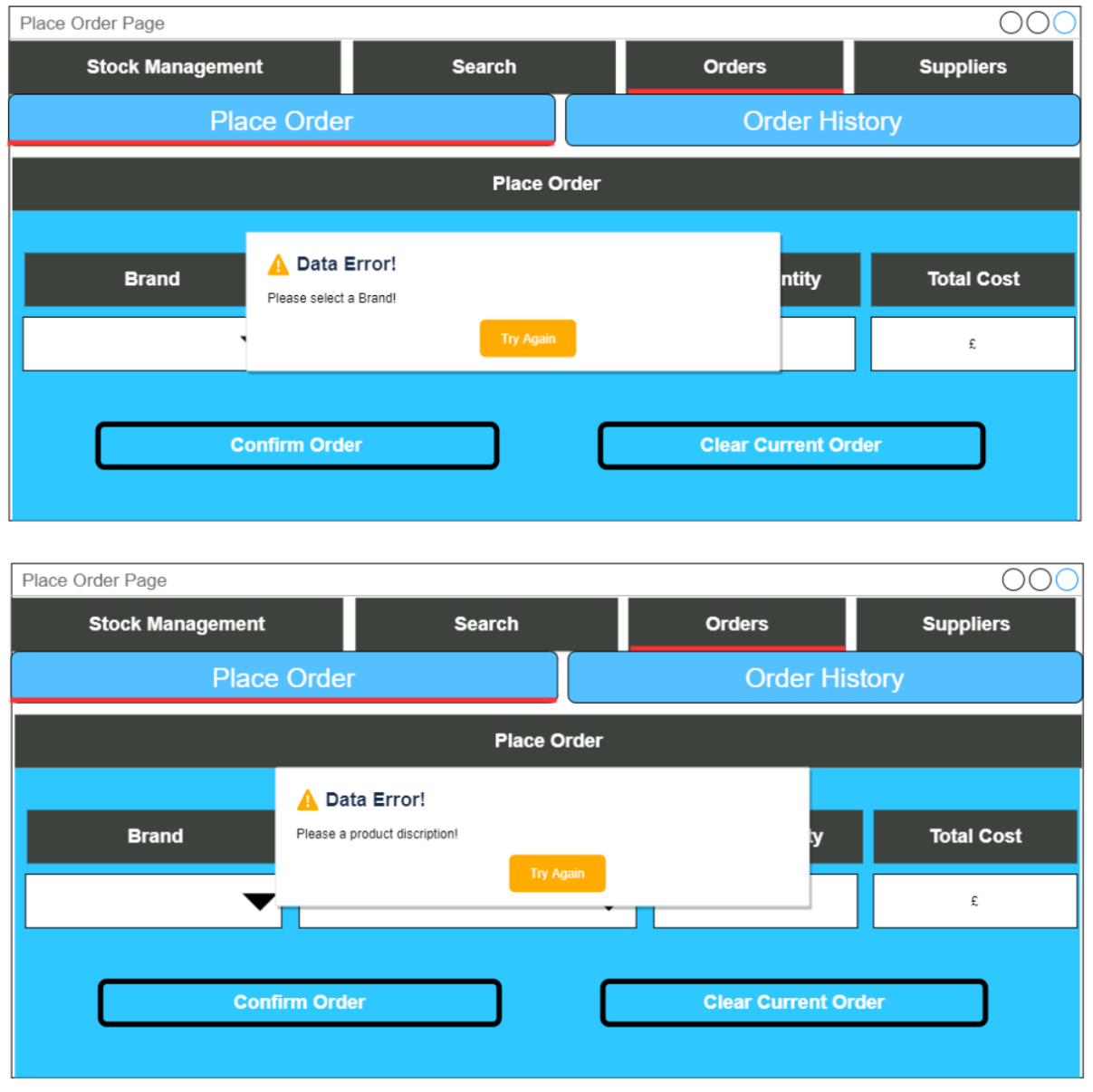
When opening the order tab two other tabs open below giving the user the choice between Placing a New Order or checking the Order History

#### Place Order

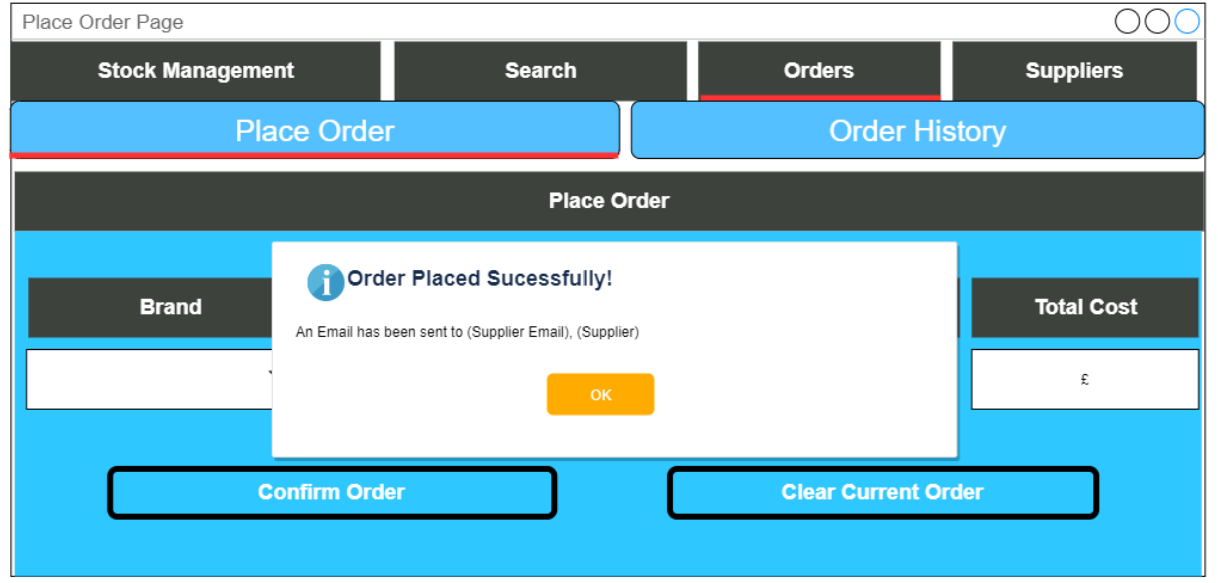


When placing a new order, you select the PIP number of the product and its corresponding quantity and the price is calculated. You can add more products by pressing add more and entering the details and finally a Total Price is calculated and the OrderID is generated. An email is also generated and sent directly to the supplier’s email so that the order is placed via Email.







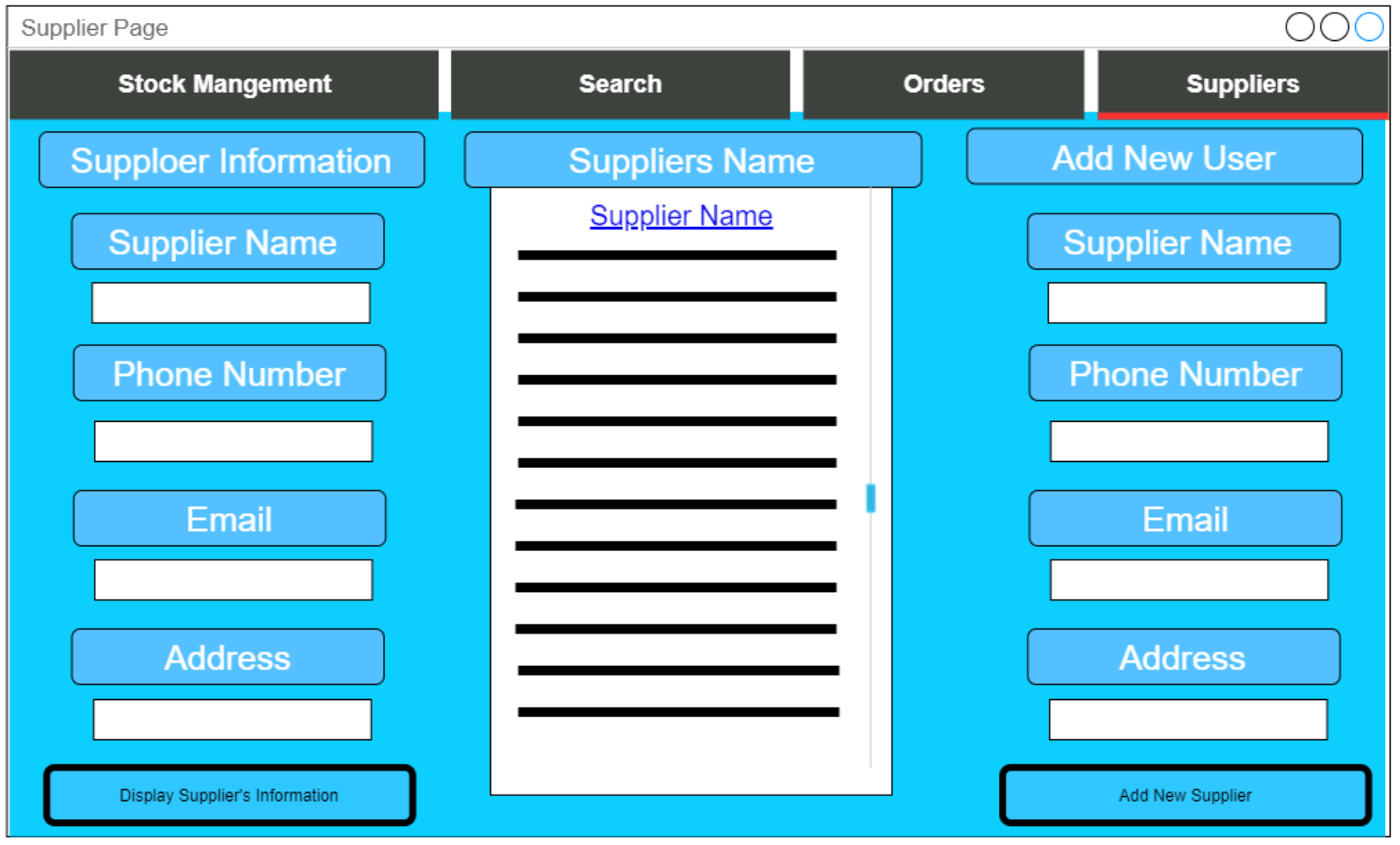


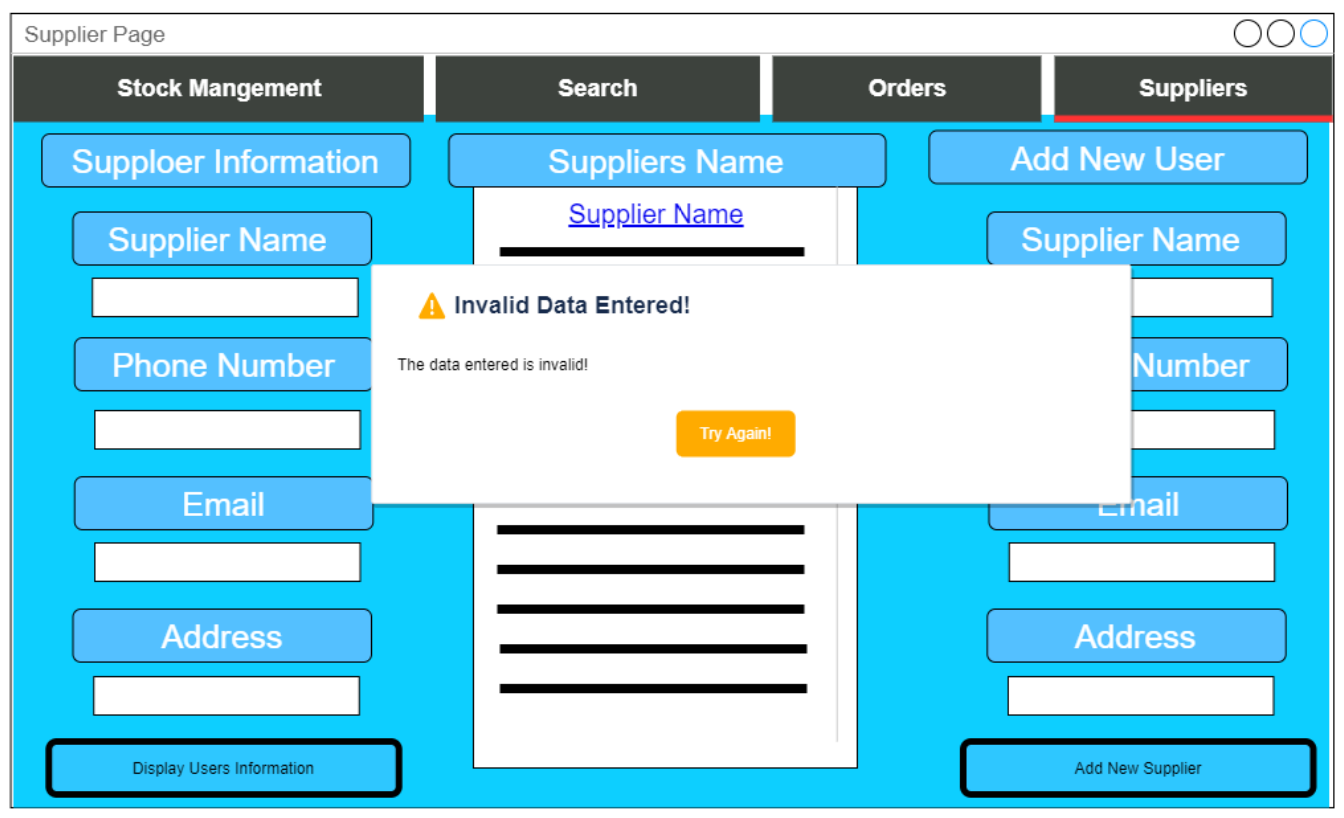
#### Order History



In the Order History Tab, the list of all Orders, including the OrderID, Date Purchased, and Total Price are displayed in a scrollable list. When the Order ID is pressed on, a window pops up with the information for that specific order, including the PIP, Supplier, and Quantity Ordered.

### Supplier



A List of suppliers are displayed in a scrollable list. The user has the option to either add a new supplier by entering the supplier’s details and clicking the “Add New Suppliers” Button. There is also a supplier’s information section where the user selects a supplier name from the list and clicks the “Display Supplier’s Information” Button to display all the suppliers information in the boxes.

If any data entered into the entry boxes for the add new user section is invalid then an eroor message is displayed to the user

## Pseudo and Python Code

### CSV File - Usernames and Passwords

I will be creating a CSV file to store all the Usernames and Passwords, and this file will have headers for the two rows Usernames and Passwords

with open('Users.csv', 'w') as Users:

filewriter = csv.writer(Users, delimiter=',')

filewriter.writerow(['Usernames', 'Passwords'])

filewriter.writerow(['Bfawaz1','Bones471'])

filewriter.writerow(['Samir007','h20medicine'])

filewriter.writerow(['Bettybo99','AQAcp2020'])

filewriter.writerow(['Paul101','123!Paul?321'])

filewriter.writerow(['ADMIN','100001'])

To extract the Usernames and Passwords so that they are corresponding and so that I skip the first row (The headers). I will be using a read csv file function to append the data from each row into a list so that the corresponding list index of both lists are the corresponding Username and Password.

User\_Names = []

Passes = []

with open('Users.csv', 'r') as f:

reader = csv.reader(f, delimiter=',', quotechar='"')

rowNr = 0

for row in reader:

if len(row) < 1:

continue

if rowNr >= 1:

User\_Names.append(row[0])

Passes.append(row[1])

rowNr = rowNr + 1

For the ADMIN to add a new User after all the necessary checks and validations are passed, the New Username and Password of the User is written as a new row in the User.csv file where everything is stored.

def append\_NewUser\_CSV(file\_name, list\_of\_elem):

with open(file\_name, 'a+', newline='') as write\_obj:

csv\_writer = writer(write\_obj)

csv\_writer.writerow(list\_of\_elem)

### Validations

For the validations in the programme I will be using a variety of loops, iterations, and selection statements so that the user has multiple chances to correct any data entered. If the data is in valid a message box will be displayed to inform the user why the data is erroneous and what they need to do so that it is acceptable. It also resets anything they have entered into the entry boxes to add the data.

if (len(Check\_Pass) < 8) or (re.search('[0-9]', Check\_Pass) is None) or (re.search('[A-Z]',Check\_Pass) is None) or (not any(c in Special\_Characters for c in Check\_Pass)):

messagebox.showwarning("Password Does Not Meet Criteria", """Please make sure your password:

\n 1) Is at least 8 Characters long

\n 2) Has at least one number in it

\n 3) Has a Capital letter

\n 4) Has at least one Special Characters from: ($,#,@,!,\*,.,\_,&,!,£,^)

\n Would you like to try again? """)

self.New\_Pass.set("")

self.Confirm\_Pass.set("")

return

else:

User\_Names.append(Check\_User)

Passes.append(Check\_Pass)

Added\_NewUser = [Check\_User, Check\_Pass]

append\_NewUser\_CSV('Users.csv', Added\_NewUser)

self.New\_User.set("")

self.New\_Pass.set("")

self.Confirm\_Pass.set("")

messagebox.showinfo("New User Added!","A New User Has Been Sucessfully Added!")

return

break

### Email for Placing Orders

I will need to first import (os) and (smtplib, ssl) to that you can use the simple mail transfer protocol (smtp) function and environmental variables for the email and email password. I will need to create environmental variables so that the email and email password are stored securely within the computer system.

I will be using port 465 of the smtp and this is how the email and email password is defined and used:

smtp\_server = "smtp.gmail.com"

sender\_email = os.environ.get('Pharmacy\_Email')

receiver\_email = str(Order\_Supplier\_Email[0])

password = os.environ.get('Pharmacy\_Password')

This will be how the email is generated when placing an order:

Email\_Message =("""From: Sam

To: %s

Subject: %s

To Whom It May Concern,

We would like to place an order to purchase %s packages of %s, %s. With PIP Code - %s.

Our Details are as follows

Pharmacy Name - Supercare Pharmacy

Phone Number - xxxxxxxxxxx

Fax - xxxxxxxxxxxx

Address - xxxxx, xxx xxxxxx, xxx xxxx

Many Thanks

Supercare Pharmacy

Sent via Python!

"""%(receiver\_email, Place\_PIP, Place\_Order\_Quantity\_Ordered, Order\_Brand\_Name\_chosen, Order\_Product\_Discription\_chosen, Place\_PIP))

As you can see the email will depend on the order placed as it depends on the product (PIP code, Brand Name, Product Discription), its quantity ordered, and the supplier of the product.

Along with this a try and exception statement will be used in the event of an error and a message will be displayed to the user if the email has sent or if the email has not been sent successfully.

try:

context = ssl.create\_default\_context()

with smtplib.SMTP\_SSL(smtp\_server, port, context=context) as server:

server.login(sender\_email, password)

server.sendmail(sender\_email, receiver\_email, Email\_Message)

messagebox.showinfo("Email","Successfully sent email to %s, %s"%(Order\_Supplier\_Email[0],Order\_Supplier\_Name[0]))

except Exception:

messagebox.showerror("Email Error!","Error: unable to send email")

Below is the generated email for “Order Cancelation” when a user wants to cancel an order previously placed.

Email\_Message =("""From: Sam

To: %s

Subject: %s

To Whom It May Concern,

We would like to CANCEL an order placed on %s, with OrderID - %s.

The order was to purchase %s packages of %s, %s. With PIP Code - %s.

Our details are as follows

Pharmacy Name - Supercare Pharmacy

Phone Number - xxxxxxxxxxx

Fax - xxxxxxxxxxxx

Address - xxxxx, xxx xxxxxx, xxx xxxx

Many Thanks

Supercare Pharmacy

Sent via Python!

"""%(receiver\_email,Cancel\_Order\_PIP,Cancel\_Order\_Date,Cancel\_OrderID,Order\_Quantity\_Chosen,Order\_Brand\_Name\_Chosen,Order\_Product\_Discription\_Chosen,Cancel\_Order\_PIP))

This email depend on the OrderID selected this email will include the date and time the order is place, the OrderID and what the placed order was.

## How I will be Testing

I will be testing my final programme by using a combination of a testing table and a screen capture video of the test being done. The testing table has a test number, description of what is being tested, the data type weather it is Typical (acceptable data) or Erroneous (invalid data), the data that will be tested, the expected result, weather the test has passed or failed, and final a cross reference link to the video and time in the video where the test takes place.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Number** | **Description** | **Data type** | **Tested Data** | **Expected Result** | **Pass/Fail** | **Cross Reference** |
| The test number | Description of what the test is looking for and how I will be preforming it | Typical or Erroneous. | The data that will be entered and tested | What I will be expecting as a result from the programme of the tested data | Did the test fail or pass? | link to the video of the test and a time range of when the test took place |

# Technical Solution

## Stock\_Management\_Database\_File

import sqlite3

from sqlite3 import Error

def create\_table(con, create\_sql\_table):

try:

c = con.cursor()

c.execute(create\_sql\_table)

con.commit()

except Error as e:

print(e)

def StockMangementData():

create\_Brands\_table = ''' CREATE TABLE IF NOT EXISTS "Brands" (

"BrandID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

"Brand\_Name" TEXT NOT NULL UNIQUE

);'''

creat\_Place\_Order\_table = '''CREATE TABLE IF NOT EXISTS "Place\_Order" (

"OrderID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

"PIP" INTEGER NOT NULL,

"Order\_Quantity" INTEGER NOT NULL,

"Total\_Cost" DECIMAL(6 , 2) NOT NULL,

"Date\_Placed" DATETIME NOT NULL,

FOREIGN KEY("PIP") REFERENCES "Product\_Info"("PIP")

);'''

create\_Product\_Info\_table = ''' CREATE TABLE IF NOT EXISTS "Product\_Info" (

"PIP" INTEGER NOT NULL UNIQUE,

"BrandID" INTEGER NOT NULL,

"Product\_Discription" TEXT NOT NULL,

"UseID" INTEGER NOT NULL,

"Pack\_Size" TEXT NOT NULL,

"Product\_Cost" DECIMAL(6 , 2) NOT NULL,

"SupplyID" INTEGER NOT NULL,

"Quantity" INTEGER NOT NULL,

PRIMARY KEY("PIP"),

FOREIGN KEY("UseID") REFERENCES "Uses"("UseID"),

FOREIGN KEY("SupplyID") REFERENCES "Supplier"("SupplyID"),

FOREIGN KEY("BrandID") REFERENCES "Brands"("BrandID")

);'''

create\_Supplier\_table = ''' CREATE TABLE IF NOT EXISTS "Supplier" (

"SupplyID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

"Supply\_Name" TEXT NOT NULL UNIQUE,

"Phone" VARCHAR(11) NOT NULL UNIQUE,

"Email" TEXT NOT NULL UNIQUE,

"Address" TEXT NOT NULL UNIQUE

);'''

create\_Uses\_table = ''' CREATE TABLE IF NOT EXISTS "Uses"(

"UseID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,

"Use" TEXT NOT NULL UNIQUE

);'''

con = sqlite3.connect("Stock\_Management\_Database.db")

if con is not None:

create\_table(con,create\_Brands\_table)

create\_table(con,creat\_Place\_Order\_table)

create\_table(con,create\_Product\_Info\_table)

create\_table(con,create\_Supplier\_table)

create\_table(con,create\_Uses\_table)

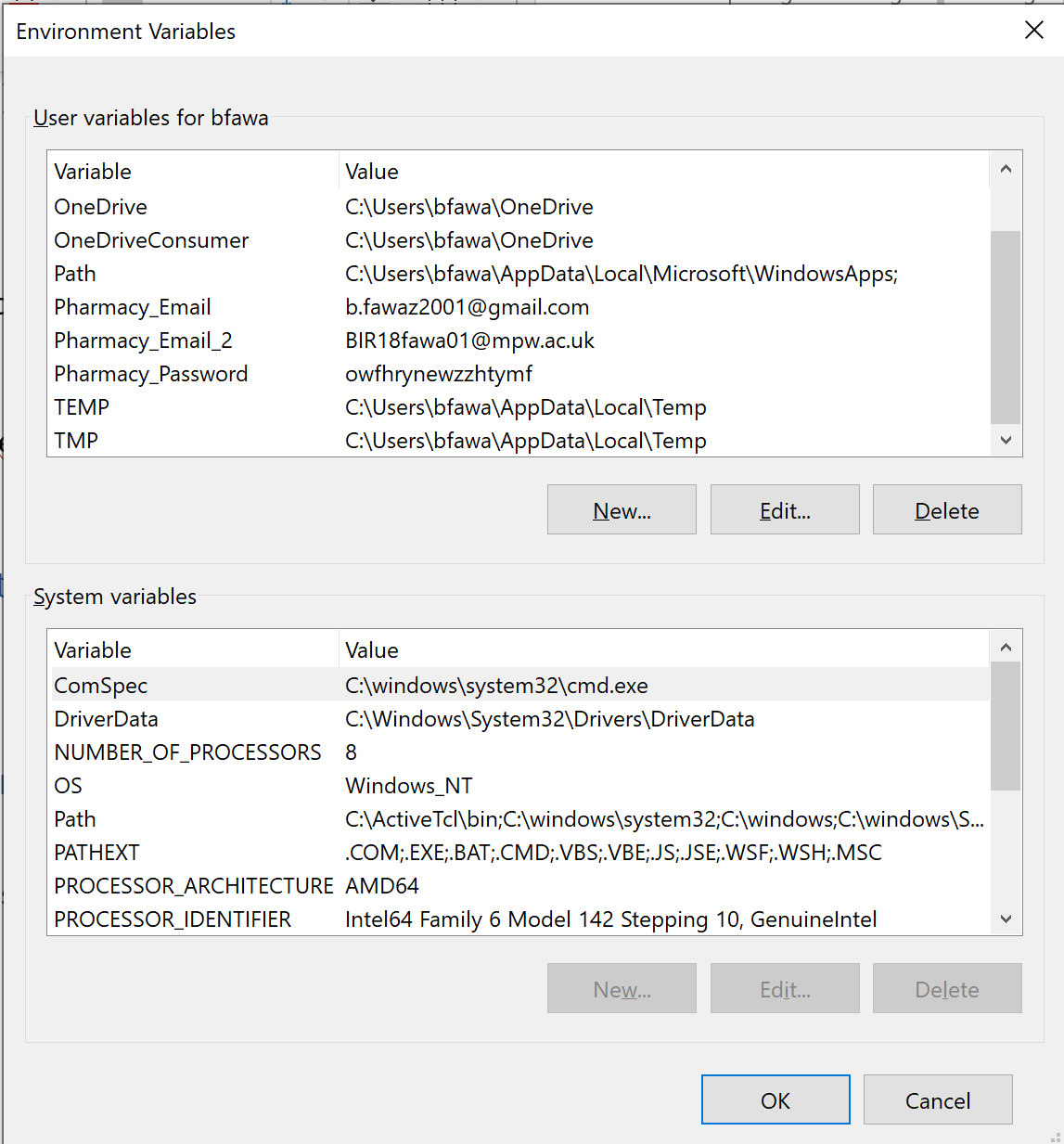
con.close()

else:

print("Error! Can't create the Database Connection!")

StockMangementData()

## Environmental Variables

[[4]](#footnote-4)



## Main Programme

import os

import smtplib, ssl

import tkinter

from tkinter import \*

from tkinter import messagebox

import tkinter.messagebox

from tkinter import Tk, StringVar, ttk

from tkinter import Toplevel

import tkinter.ttk as tkrtk

import tkinter as tkr

import csv

from csv import writer

import re

from datetime import datetime

import sqlite3

from sqlite3 import Error

import Stock\_Managemnt\_Database\_File

#-----------UserNames and Passwords File Creation--------------#

#with open('Users.csv', 'w') as Users:

#filewriter = csv.writer(Users, delimiter=',')

#filewriter.writerow(['Usernames', 'Passwords'])

#filewriter.writerow(['Bfawaz1','Bones471'])

#filewriter.writerow(['Samir007','h20medicine'])

#filewriter.writerow(['Bettybo99','AQAcp2020'])

#filewriter.writerow(['Paul101','123!Paul?321'])

#filewriter.writerow(['ADMIN','100001'])

#-------------Identifying User Names to Passwords--------------#

# create list holders for our data.

User\_Names = []

Passes = []

# open file

with open('Users.csv', 'r') as f:

reader = csv.reader(f, delimiter=',', quotechar='"')

# read file row by row

rowNr = 0

for row in reader:

if len(row) < 1:

continue

# Skip the header row

if rowNr >= 1:

User\_Names.append(row[0])

Passes.append(row[1])

# Increase the row number

rowNr = rowNr + 1

# Print data

#Zipped\_Lists = zip(User\_Names,Passes)

#Users\_and\_Passwords = (list(Zipped\_Lists))

#--------------------For Admins Eyes Only----------------------#

#def admin(self):

#with open('Users.csv', 'r') as a:

# reader = csv.reader(a)

# read file row by row

#for row in reader:

# Reveal.append(row)

def append\_NewUser\_CSV(file\_name, list\_of\_elem):

with open(file\_name, 'a+', newline='') as write\_obj:

csv\_writer = writer(write\_obj)

csv\_writer.writerow(list\_of\_elem)

#------------------------------Main Loop------------------------------#

def main():

global root

root = Tk()

app = Login(root)

#-----------------------------Login Window-----------------------------#

class Login:

def \_\_init\_\_(self, master):

self.master = master

self.master.title("User Login Page")

self.master.geometry("1350x750+0+0")

self.master.config(bg ='Salmon')

self.frame = Frame(self.master, bg='Salmon')

self.frame.pack()

self.Input\_UserName = StringVar()

self.Input\_Password = StringVar()

self.lblTitle = Label(self.frame, text = 'Login Window', font =('arial', 60, 'bold'),

bg='Salmon', fg='Cornsilk')

self.lblTitle.grid(row=0, column=0, columnspan=2, pady=20)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

self.LoginFrame1 = LabelFrame(self.frame, width=1350,height=300

,text="Login",font=('arial',20,'bold'), relief='ridge',bg='SlateBlue2', bd=40)

self.LoginFrame1.grid(row=1, column=0)

self.LoginFrame2 = LabelFrame(self.frame, width=1000,height=200

,font=('arial',20,'bold'), relief='ridge',bg='SlateBlue2', bd=40)

self.LoginFrame2.grid(row=2, column=0)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Inputs Entries UserName & Password~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

self.lblInput\_UserName = Label(self.LoginFrame1, text = 'Username', font =('arial',30,'bold'),bd=22,

bg = 'SlateBlue2', fg = 'Cornsilk')

self.lblInput\_UserName.grid(row=0,column=0)

self.txtInput\_UserName = Entry(self.LoginFrame1,font=('arial',30,'bold'),bd=7,textvariable=self.Input\_UserName,

width = 33)

self.txtInput\_UserName.grid(row=0,column=1,padx=88)

self.lblInput\_Password = Label(self.LoginFrame1,text='Password',font=('arial',30,'bold'),bd=22,

bg='SlateBlue2', fg = 'Cornsilk')

self.lblInput\_Password.grid(row=1,column=0)

self.txtInput\_Password = Entry(self.LoginFrame1, font =('arial',30,'bold'), show='\*',bd=7,textvariable=self.Input\_Password,

width=33)

self.txtInput\_Password.grid(row=1,column=1,columnspan=2,pady=30)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Buttons~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

self.btnLogin = Button(self.LoginFrame2, text='Login', width=15,font=('arial',30,'bold'),

bg='SlateBlue2', fg='Cornsilk', command=self.Check\_Login)

self.btnLogin.grid(row=3,column=0,pady=20,padx=8)

self.btnReset = Button(self.LoginFrame2, text='Reset', width=15,font=('arial',30,'bold'),

bg='SlateBlue2', fg='Cornsilk', command=self.Reset\_Com)

self.btnReset.grid(row=3,column=1,pady=20,padx=8)

self.btnExit = Button(self.LoginFrame2, text='Exit', width=15,font=('arial',30,'bold'),

bg='SlateBlue2', fg='Cornsilk', command=self.Exit\_Com)

self.btnExit.grid(row=3,column=2,pady=20,padx=8)

def Check\_Login(self):

Key = False

#Gets the username and password that has just been entered

try\_User = (self.Input\_UserName.get())

try\_Pass = (self.Input\_Password.get())

#Checks against the data base to see if there is a matching one in the lists.

while Key == False:

if (try\_User in User\_Names) and (try\_Pass in Passes) and (try\_User != "") and (try\_Pass != ""):

username\_index\_position = User\_Names.index(try\_User)

password\_index\_position = Passes.index(try\_Pass)

if username\_index\_position == password\_index\_position:

Key = True

if (try\_User == "ADMIN") and (try\_Pass == "100001"):

self.Admin\_window()

else:

self.Main\_window()

#Display LOGIN Sucessfull on a pop up window that disapears after 5 secs

#Also display on same window what user you are signed in with

else:

Key = False

messagebox.showerror("Invalid Login Details", "Something Went Wrong Please Try Again!")

self.Input\_UserName.set("") #Resets Username and Password Boxes

self.Input\_Password.set("")

return

else:

Key = False

messagebox.showerror("Invalid Login Details", "Incorrect Login Details! Try Again!")

self.Input\_UserName.set("") #Resets Username and Password Boxes

self.Input\_Password.set("")

return

def Reset\_Com(self):

self.Input\_UserName.set("")

self.Input\_Password.set("")

def Exit\_Com(self):

self.Exit\_Com = tkinter.messagebox.askyesno("\*!Quit System!\*", "Confirm Exit")

if self.Exit\_Com > 0:

self.master.destroy()

return

else:

return

def Main\_window(self):

self.master.destroy()

Tk().withdraw()

self.Home\_Win = Toplevel()

self.app = Main\_Window(self.Home\_Win)

def Admin\_window(self):

self.master.destroy()

Tk().withdraw()

self.Admin\_Window = Toplevel()

self.app = Admin(self.Admin\_Window)

#--------------------------------------------------Main/Home Window----=---------------------------------#

#-----------------------------------------------------------------------=--------------------------------#

PIP = []

Brand\_Name = []

Product\_Discription = []

Quantity = []

OrderID = []

Order\_PIP = []

Order\_Quantity = []

Total\_Cost = []

Dates = []

Suppliers\_Names = []

class Main\_Window:

#---------------------------------------------- Initialisation ----------------------------------------------------#

def \_\_init\_\_(self, master):

self.master = master

self.master.title("Main Page")

self.master.geometry("1350x800+0+0")

self.master.config(bg ='Peach Puff')

#self.main\_frame = Frame(self.master, bg='Peach Puff')

#self.main\_frame.pack()

#self.btnDestroy = Button(self.frame, text='Destroy Window', width=15,font=('arial',25,'bold'),

#bg='SlateBlue2', fg='Cornsilk', command=self.Destroy\_Window)

#self.btnDestroy.grid(row=3,column=0,pady=20,padx=8)

#a = tkrtk.Style()

#a.configure('TNotebook.Tab', font=('arial','38'), height = 10, foreground='green')

style = tkrtk.Style()

style.theme\_create( "Main\_Theme", parent="alt", settings={

"TNotebook": {"configure": {"tabmargins": [10, 5, 10, 0] } },

"TNotebook.Tab": {

"configure": {"padding": [130, 5], "background": "white" },

"map": {"background": [("selected", "red")],

"expand": [("selected", [1, 1, 1, 10])] } } } )

style.theme\_use("Main\_Theme")

Tabs = tkrtk.Notebook(self.master)

Stock\_Manage\_Frame = Frame(Tabs, bg='Peach Puff', width=1350, height=400)

global Search\_Frame

Search\_Frame = Frame(Tabs, bg='plum2', width=200, height=200)

Orders\_Frame = Frame(Tabs, bg='Thistle2', width=200, height=200)

Suppliers\_Frame = Frame(Tabs, bg='dark orchid', width=200, height=200)

Tabs.add(Stock\_Manage\_Frame, text=" Stock Managment " )

Tabs.add(Search\_Frame, text=" Search " )

Tabs.add(Orders\_Frame, text=" Orders " )

Tabs.add(Suppliers\_Frame, text=" Suppliers " )

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

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

#------------------------------------------- Stock Management Tab -----------------------------------------------------#

#------------------------------------------------- Frames -------------------------------------------------------------#

StockManage\_Frame0 = Frame(Stock\_Manage\_Frame, width=1350, height=25, bg = "Peach Puff")

StockManage\_Frame0.grid(row = 0, column =0, pady = 10)

StockManage\_Frame1 = Frame(Stock\_Manage\_Frame, width=1350, height=25, bg = "Peach Puff")

StockManage\_Frame1.grid(row = 1, column =0, padx = 100, pady = 10)

StockManage\_Frame2 = Frame(Stock\_Manage\_Frame, width=1350, height=400, bg = "Peach Puff")

StockManage\_Frame2.grid(row = 2, column =0, padx = 100)

StockManage\_Frame3 = Frame(Stock\_Manage\_Frame, width=1350, height=200, bg = "Peach Puff")

StockManage\_Frame3.grid(row = 3, column =0)

#--------------------------------------- Labels, Entry Boxes, and Buttons ---------------------------------------------#

StockManage\_Title = Label(StockManage\_Frame0, text = "Stock Management", font=('arial',25,'bold'), bd = 25, relief='ridge',

bg = 'Peach Puff', fg = 'Black')

StockManage\_Title.grid(row=0,column=0)

Order\_By\_Label = Label(StockManage\_Frame1, text = "Orber By", font=('arial',15,'bold'), bd = 10, relief='ridge',

bg = 'Peach Puff', fg = 'Black')

Order\_By\_Label.grid(row=0,column=0, padx = 5)

PIP\_Nummber\_Button = Button(StockManage\_Frame1,text='PIP', width=5,font=('arial',15,'bold'), bd = 5, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = self.Display\_Stock\_PIP)

PIP\_Nummber\_Button.grid(row = 0, column = 1, padx = 5)

Highest\_Stock\_Button = Button(StockManage\_Frame1,text='Highest Stock Quantity', width=20,font=('arial',15,'bold'), bd = 5, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = self.Display\_Stock\_From\_Highest)

Highest\_Stock\_Button.grid(row = 0, column = 2, padx = 5)

Lowest\_Stock\_Button = Button(StockManage\_Frame1,text='Lowest Stock Quantity', width=20,font=('arial',15,'bold'), bd = 5, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = self.Display\_Stock\_From\_Lowest)

Lowest\_Stock\_Button.grid(row = 0, column = 3, padx = 5)

Brand\_Name\_A2Z\_Button = Button(StockManage\_Frame1,text='Brand Name A to Z', width=15,font=('arial',15,'bold'), bd = 5, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = self.Display\_Stock\_A2Z)

Brand\_Name\_A2Z\_Button.grid(row = 0, column = 4, padx = 5)

Brand\_Name\_Z2A\_Button = Button(StockManage\_Frame1,text='Brand Name Z to A', width=15,font=('arial',15,'bold'), bd = 5, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = self.Display\_Stock\_Z2A)

Brand\_Name\_Z2A\_Button.grid(row = 0, column = 5, padx = 5)

Stock\_PIP\_Label = Label(StockManage\_Frame2, text = "PIP", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'Peach Puff', fg = 'Black')

Stock\_PIP\_Label.grid(row=0,column=0)

Brand\_Label = Label(StockManage\_Frame2, text = "Brand", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 15, bg = 'Peach Puff', fg = 'Black')

Brand\_Label.grid(row=0,column=1)

Stock\_Product\_Name\_Label = Label(StockManage\_Frame2, text = "Product Discription", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 25, bg = 'Peach Puff', fg = 'Black')

Stock\_Product\_Name\_Label.grid(row=0,column=2)

Stock\_Quantity\_Label = Label(StockManage\_Frame2, text = "Quantity", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 8, bg = 'Peach Puff', fg = 'Black')

Stock\_Quantity\_Label.grid(row=0,column=3)

self.scrollbar\_V = Scrollbar(StockManage\_Frame2)

self.scrollbar\_H\_PIP = Scrollbar(StockManage\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Brand\_Name = Scrollbar(StockManage\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Product\_Discription = Scrollbar(StockManage\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Quantity = Scrollbar(StockManage\_Frame2, orient=HORIZONTAL)

self.scrollbar\_V.grid(row = 1, column = 4, sticky=N+S+W)

self.scrollbar\_H\_PIP.grid(row = 2, column = 0, sticky=N+E+S+W)

self.scrollbar\_H\_Brand\_Name.grid(row = 2, column = 1, sticky=N+E+S+W)

self.scrollbar\_H\_Product\_Discription.grid(row = 2, column = 2, sticky=N+E+S+W)

self.scrollbar\_H\_Quantity.grid(row = 2, column = 3, sticky=N+E+S+W)

self.lbPIP = Listbox(StockManage\_Frame2, font=('arial',12,'bold'), width = 10, height = 20, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_PIP.set)

self.lbPIP.grid(row=1,column=0)

self.lbBrand\_Name = Listbox(StockManage\_Frame2, font=('arial',12,'bold'), width = 15, height = 20, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Brand\_Name.set)

self.lbBrand\_Name.grid(row=1,column=1)

self.lbProduct\_Discription = Listbox(StockManage\_Frame2, font=('arial',12,'bold'), width = 25, height = 20, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Product\_Discription.set)

self.lbProduct\_Discription.grid(row=1,column=2)

self.lbQuantity = Listbox(StockManage\_Frame2, font=('arial',12,'bold'), width = 8, height = 20, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Quantity.set)

self.lbQuantity.grid(row=1,column=3)

self.Display\_Stock\_PIP()

self.scrollbar\_V.config(command=self.yview)

self.scrollbar\_H\_PIP.config(command=self.lbPIP.xview)

self.scrollbar\_H\_Brand\_Name.config(command=self.lbBrand\_Name.xview)

self.scrollbar\_H\_Product\_Discription.config(command=self.lbProduct\_Discription.xview)

self.scrollbar\_H\_Quantity.config(command = self.lbQuantity.xview)

Edit\_Button = Button(StockManage\_Frame3,text='Edit Stock Quantity', width=15,font=('arial',15,'bold'), bd =10, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = self.edit\_stock)

Edit\_Button.grid(row = 0, column = 1, padx = 10, pady = 20)

AddNew\_Button = Button(StockManage\_Frame3,text='Add New Products', width=15,font=('arial',15,'bold'), bd = 10, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = self.New\_Product)

AddNew\_Button.grid(row = 0, column = 2, padx = 10, pady = 20)

Product\_Info\_Button = Button(StockManage\_Frame3,text='Product Info', width=15,font=('arial',15,'bold'), bd = 10, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = self.Product\_Info)

Product\_Info\_Button.grid(row = 0, column = 3, padx = 10, pady = 20)

Delete\_Button = Button(StockManage\_Frame3,text='Delete Product', width=15,font=('arial',15,'bold'), bd = 10, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = self.Delete\_Product)

Delete\_Button.grid(row = 0, column = 4, padx = 10, pady = 20)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

#-------------------------------------------------- Search Tab --------------------------------------------------------#

Search\_Frame0 = Frame(Search\_Frame, width=1350, height=30, bg = "plum2")

Search\_Frame0.grid(row = 0, column = 0, pady = 3)

Search\_Frame1 = Frame(Search\_Frame, width=1350, height=50, bg = "plum2",bd = 20, relief='ridge')

Search\_Frame1.grid(row = 1, column =0, padx = 350, pady = 3)

#--------------------------------------- Labels, Entry Boxes, and Buttons ---------------------------------------------#

Search\_Title = Label(Search\_Frame0, text = "Search", font=('arial',25,'bold'), width = 15, bd = 25, relief='ridge',

bg = 'Maroon3', fg = 'White')

Search\_Title.grid(row=0,column=0)

Search\_Label = Label(Search\_Frame1, text = "Search Bar", font=('arial',15,'bold'), bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Search\_Label.grid(row=0,column=0)

global Search\_Bar\_Entry

Search\_Bar\_Entry = StringVar()

global Search\_Bar

Search\_Bar = Entry(Search\_Frame1, font=('arial',20,'bold'),bd=7,textvariable = Search\_Bar\_Entry, width = 25, relief='ridge')

Search\_Bar.grid(row=0,column=1)

global Search\_By\_Menu

Search\_By\_Menu = StringVar()

Search\_By\_Menu.set("Search By")

Search\_Option = ["Use", "Supplier", "PIP", "Brand"]

Search\_By\_DropDownMenu = OptionMenu(Search\_Frame1, Search\_By\_Menu,\*Search\_Option)

Search\_By\_DropDownMenu.grid(row=0,column=2, padx = 15)

Search\_Button = Button(Search\_Frame1,text='Start Search', width=15,font=('arial',15,'bold'), bd = 10, relief='ridge',

bg='Maroon3', fg='White', command = self.Start\_Search)

Search\_Button.grid(row = 1, column = 1, pady = 5)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

#-------------------------------------------------- Orders Tab --------------------------------------------------------#

OrderTabs = tkrtk.Notebook(Orders\_Frame)

Place\_Order\_Tab = Frame(OrderTabs, bg='hotpink1', width=1000, height=500)

Order\_History\_Tab = Frame(OrderTabs, bg='indian red', width=1000, height=500)

OrderTabs.add(Place\_Order\_Tab, text=" Place Order " )

OrderTabs.add(Order\_History\_Tab, text=" Order History " )

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

#-------------------------------------------------- Place Orders Tab --------------------------------------------------------#

#-------------------------------------------------------- Frames ------------------------------------------------------------#

Place\_Order\_Frame0 = Frame(Place\_Order\_Tab, width = 1350, height=25, bg = "hotpink1")

Place\_Order\_Frame0.grid(row = 0, column =0, pady = 10)

Place\_Order\_Frame1 = Frame(Place\_Order\_Tab,width = 1350, height=25, bg = "hotpink1")

Place\_Order\_Frame1.grid(row =1, column =0, pady = 10)

Place\_Order\_Frame2 = Frame(Place\_Order\_Tab, width = 1350, height=25, bg = "hotpink1")

Place\_Order\_Frame2.grid(row = 2, column =0)

Place\_Order\_Frame3 = Frame(Place\_Order\_Tab, width = 1350, height=25, bg = "hotpink1")

Place\_Order\_Frame3.grid(row = 3, column =0)

#--------------------------------------------------------- List -------------------------------------------------------------#

Brands = [""]

Order\_BrandID = [""]

Order\_Product\_Discription = [""]

Order\_Supplier\_Name = [""]

Order\_Supplier\_Email = [""]

#-------------------------------------------- Place Order Tab Functions -----------------------------------------------------#

def List\_Brands():

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Brand\_Name FROM "Brands"')

Brands\_rows = cur.fetchall()

rowNr = 0

for row in Brands\_rows:

if len(row) < 1:

continue

if rowNr >= 0:

Brands.append(row[0])

rowNr = rowNr + 1

global Order\_Brand\_Menu

Order\_Brand\_Menu = StringVar()

Order\_Brand\_Menu.set("Pick a Brand")

Place\_Order\_Brand\_Label = Label(Place\_Order\_Frame1, text = "Brand", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'Thistle2', fg = 'Black')

Place\_Order\_Brand\_Label.grid(row=0,column=0)

global Brand\_Name\_DropDownMenu

Brand\_Name\_DropDownMenu = OptionMenu(Place\_Order\_Frame1, Order\_Brand\_Menu, \*Brands)

Brand\_Name\_DropDownMenu.grid(row=1,column=0)

Pick\_Choice = Button(Place\_Order\_Frame3,text='Confirm Brand', width=20,font=('arial',15,'bold'), bd =10, relief='ridge',

bg='Thistle2', fg='Black', command = BrandID)

Pick\_Choice.grid(row = 0, column = 0, pady = 10)

def BrandID():

if Order\_Brand\_Menu.get() == "Pick a Brand":

messagebox.showerror("No Brand Selected", "Please Select a Brand and Try Again")

else:

Order\_BrandID.clear()

global Order\_Brand\_Name\_chosen

Order\_Brand\_Name\_chosen = Order\_Brand\_Menu.get()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT BrandID FROM "Brands" WHERE Brand\_Name = (:Brand\_Name\_Selected)',

{

"Brand\_Name\_Selected" : Order\_Brand\_Name\_chosen

})

OrderID\_rows = cur.fetchall()

rowNr = 0

for row in OrderID\_rows:

if len(row) < 1:

continue

if rowNr >= 0:

Order\_BrandID.append(row[0])

rowNr = rowNr + 1

global Order\_BrandID\_chosen

Order\_BrandID\_chosen = Order\_BrandID[0]

List\_Product\_Discription()

def List\_Product\_Discription():

Order\_Product\_Discription.clear()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Product\_Discription FROM "Product\_Info" WHERE BrandID = (:BrandID\_Selected)',

{

"BrandID\_Selected" : Order\_BrandID\_chosen

})

Product\_Discription\_rows = cur.fetchall()

rowNr = 0

for row in Product\_Discription\_rows:

if len(row) < 1:

continue

if rowNr >= 0:

Order\_Product\_Discription.append(row[0])

rowNr = rowNr + 1

global Place\_Order\_Product\_Discription\_Label

Place\_Order\_Product\_Discription\_Label = Label(Place\_Order\_Frame1, text = "Product Discription", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 15, bg = 'Thistle2', fg = 'Black')

Place\_Order\_Product\_Discription\_Label.grid(row=0,column=1, padx = 10)

global Product\_Discription\_Menu

Product\_Discription\_Menu = StringVar()

Product\_Discription\_Menu.set("Pick a Product")

global Product\_Discription\_DropDownMenu

Product\_Discription\_DropDownMenu = OptionMenu(Place\_Order\_Frame1, Product\_Discription\_Menu, \*Order\_Product\_Discription)

Product\_Discription\_DropDownMenu.grid(row=1,column=1, padx = 10)

Pick\_Choice = Button(Place\_Order\_Frame3,text='Confirm Product', width=20,font=('arial',15,'bold'), bd =10, relief='ridge',

bg='Thistle2', fg='Black', command = Enter\_Quantity)

Pick\_Choice.grid(row = 0, column = 0, pady = 10)

def Enter\_Quantity():

if Product\_Discription\_Menu.get() == "Pick a Product" or (len(Product\_Discription\_Menu.get()) < 1):

messagebox.showerror("No Product Selected", "Please Select a Product and Try Again")

else:

global Place\_Order\_Quantity\_Label

Place\_Order\_Quantity\_Label = Label(Place\_Order\_Frame1, text = "Quantity", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'Thistle2', fg = 'Black')

Place\_Order\_Quantity\_Label.grid(row=0,column=2, padx = 10)

global Place\_Order\_Quantity

Place\_Order\_Quantity = StringVar()

global Place\_Order\_Quantity\_Entry

Place\_Order\_Quantity\_Entry = Entry(Place\_Order\_Frame1, font=('arial',20,'bold'),bd=7,textvariable = Place\_Order\_Quantity, width = 5)

Place\_Order\_Quantity\_Entry.grid(row=1,column=2, padx = 10)

Pick\_Choice = Button(Place\_Order\_Frame3,text='Confirm Quantity', width=20,font=('arial',15,'bold'), bd =10, relief='ridge',

bg='Thistle2', fg='Black', command = Total\_Cost\_Calculated)

Pick\_Choice.grid(row = 0, column = 0, pady = 10)

def Total\_Cost\_Calculated():

Total\_Cost = []

try:

global Order\_Quantity\_chosen

Order\_Quantity\_chosen = int(Place\_Order\_Quantity.get())

except ValueError:

messagebox.showerror("Invalid Data Entered!", "Please enter an integer value for the quantity! Try Again!")

Quantity\_Entry.set("")

if Order\_Quantity\_chosen < 1:

messagebox.showerror("No Quantity Entered!", "Please select a Quantity and Try Again")

else:

global Total\_Cost\_Label

Total\_Cost\_Label = Label(Place\_Order\_Frame2, text = "Total Cost", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'Thistle2', fg = 'Black')

Total\_Cost\_Label.grid(row=0,column=0, padx = 10)

global Order\_Total\_Cost

Order\_Total\_Cost = Listbox(Place\_Order\_Frame2, font=('arial',12,'bold'), width = 15, height = 1, bd = 10)

Order\_Total\_Cost.grid(row=1, column=0, padx = 600)

global Order\_Product\_Discription\_chosen

Order\_Product\_Discription\_chosen = Product\_Discription\_Menu.get()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Product\_Cost FROM "Product\_Info" WHERE BrandID = (:BrandID\_Selected) AND Product\_Discription = (:Product\_Discription\_Selected) ',

{

"BrandID\_Selected" : Order\_BrandID\_chosen,

"Product\_Discription\_Selected" : Order\_Product\_Discription\_chosen

})

Total\_Cost = cur.fetchone()

Final\_Total\_Cost = ((Order\_Quantity\_chosen) \* float(Total\_Cost[0]))

global Calculated\_Cost

Calculated\_Cost = round(Final\_Total\_Cost, 2)

Order\_Total\_Cost.insert(0, Calculated\_Cost)

global Place\_Order\_Choice

Place\_Order\_Choice = Button(Place\_Order\_Frame3,text='Place Order', width=20,font=('arial',15,'bold'), bd =10, relief='ridge',

bg='Thistle2', fg='Black', command = Place\_Order)

Place\_Order\_Choice.grid(row = 0, column = 1, padx = 10, pady = 10)

global Clear\_Order

Clear\_Order = Button(Place\_Order\_Frame3,text='Clear Current Order', width=20,font=('arial',15,'bold'), bd =10, relief='ridge',

bg='Thistle2', fg='Black', command = Clear\_Current)

Clear\_Order.grid(row = 0, column = 2, pady = 10)

def Clear\_Current():

Total\_Cost\_Label.destroy()

Order\_Total\_Cost.destroy()

Place\_Order\_Quantity\_Label.destroy()

Place\_Order\_Quantity\_Entry.destroy()

Place\_Order\_Product\_Discription\_Label.destroy()

Product\_Discription\_DropDownMenu.destroy()

Brand\_Name\_DropDownMenu.destroy()

Place\_Order\_Choice.destroy()

Clear\_Order.destroy()

List\_Brands()

def Place\_Order():

if (Order\_Brand\_Menu.get() == "Pick a Brand") and (Product\_Discription\_Menu.get() == "Pick a Proudct") or (len(Product\_Discription\_Menu.get()) < 1) and (Order\_Quantity\_chosen < 1):

messagebox.showerror("Data Missing!", "Please entered all data and Try Agan!")

else:

Place\_Order\_Brand = Order\_Brand\_Menu.get()

Place\_Order\_Product = Product\_Discription\_Menu.get()

Place\_Order\_Quantity\_Ordered = Order\_Quantity\_chosen

Current\_DateTime = (datetime.now()).strftime("%Y-%m-%d %H:%M:%S")

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP FROM "Product\_Info" WHERE BrandID = (:BrandID\_Selected) AND Product\_Discription = (:Product\_Discription\_Selected) ',

{

"BrandID\_Selected" : Order\_BrandID\_chosen,

"Product\_Discription\_Selected" : Order\_Product\_Discription\_chosen

})

Place\_Order\_PIP\_Fetched = cur.fetchone()

con.close()

Place\_PIP = int(Place\_Order\_PIP\_Fetched[0])

#-------------------------------- Send Supplier Email -------------------------------#

Order\_Supplier\_Name.clear()

Order\_Supplier\_Email.clear()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Supply\_Name, Email FROM "Product\_Info","Supplier" WHERE PIP = (:Order\_Supplier\_PIP) AND Product\_Info.SupplyID = Supplier.SupplyID',

{

"Order\_Supplier\_PIP" : Place\_PIP

})

OrderID\_rows = cur.fetchall()

rowNr = 0

for row in OrderID\_rows:

if len(row) < 1:

continue

if rowNr >= 0:

Order\_Supplier\_Name.append(row[0])

Order\_Supplier\_Email.append(row[1])

rowNr = rowNr + 1

port = 465

smtp\_server = "smtp.gmail.com"

sender\_email = os.environ.get('Pharmacy\_Email')

receiver\_email = str(Order\_Supplier\_Email[0])

password = os.environ.get('Pharmacy\_Password')

Email\_Message =("""From: Sam

To: %s

Subject: %s

To Whom It May Concern,

We would like to place an order to purchase %s packages of %s, %s. With PIP Code - %s.

Our details are as follows

Pharmacy Name - Supercare Pharmacy

Phone Number - xxxxxxxxxxx

Fax - xxxxxxxxxxxx

Address - xxxxx, xxx xxxxxx, xxx xxxx

Many Thanks

Supercare Pharmacy

Sent via Python!

"""%(receiver\_email, Place\_PIP, Place\_Order\_Quantity\_Ordered, Order\_Brand\_Name\_chosen, Order\_Product\_Discription\_chosen, Place\_PIP))

try:

context = ssl.create\_default\_context()

with smtplib.SMTP\_SSL(smtp\_server, port, context=context) as server:

server.login(sender\_email, password)

server.sendmail(sender\_email, receiver\_email, Email\_Message)

messagebox.showinfo("Email","Successfully sent email to %s, %s"%(Order\_Supplier\_Email[0],Order\_Supplier\_Name[0]))

except Exception:

messagebox.showerror("Email Error!","Error: unable to send email")

#------------------------------------------------------------------------------------#

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('INSERT INTO "Place\_Order"(PIP,Order\_Quantity,Total\_Cost,Date\_Placed) VALUES (:Order\_Place\_PIP\_Selected, :Order\_Place\_Order\_Quantity\_Selected, :Order\_Place\_Total\_Cost, :Order\_Place\_DateTime)',

{

"Order\_Place\_PIP\_Selected" : Place\_PIP,

"Order\_Place\_Order\_Quantity\_Selected" : Place\_Order\_Quantity\_Ordered,

"Order\_Place\_Total\_Cost" : Calculated\_Cost,

"Order\_Place\_DateTime" : Current\_DateTime

})

con.commit()

con.close()

messagebox.showinfo("New Order Placed!", "A new order has been placed sucessfully!")

Order\_Brand\_Menu.set("Pick a Brand")

#----------- Reset ---------#

Clear\_Current()

#------------------------------------------- Labels, Entry Boxes, and Buttons -----------------------------------------------#

Place\_Order\_Title = Label(Place\_Order\_Frame0, text = "Place Order", font=('arial',25,'bold'), bd = 25, relief='ridge',

bg = 'Thistle2', fg = 'Black')

Place\_Order\_Title.grid(row=0,column=0, pady = 50)

List\_Brands()

#-------------------------------------------------- Orders History Tab ------------------------------------------------------#

#-------------------------------------------------------- Frames ------------------------------------------------------------#

Order\_History\_Frame0 = Frame(Order\_History\_Tab, width=1350, height=25, bg = "indian red")

Order\_History\_Frame0.grid(row = 0, column =0, pady = 5)

Order\_History\_Frame1 = Frame(Order\_History\_Tab, width=1350, height=25, bg = "indian red")

Order\_History\_Frame1.grid(row = 1, column =0, padx = 200, pady = 5)

Order\_History\_Frame2 = Frame(Order\_History\_Tab, width=1350, height=400, bg = "indian red")

Order\_History\_Frame2.grid(row = 2, column =0, padx = 200, pady = 5)

Order\_History\_Frame3 = Frame(Order\_History\_Tab, width=1350, height=200, bg = "indian red")

Order\_History\_Frame3.grid(row = 3, column =0, pady = 5)

#------------------------------------------- Labels, Entry Boxes, and Buttons -----------------------------------------------#

Order\_History\_Title = Label(Order\_History\_Frame0, text = "Order History", font=('arial',25,'bold'), bd = 25, relief='ridge',

bg = 'gold', fg = 'Black')

Order\_History\_Title.grid(row=0,column=0)

Sort\_By\_Label = Label(Order\_History\_Frame1, text = "Sort By", font=('arial',15,'bold'), width = 15, bd = 10, relief='ridge',

bg = 'indian red', fg = 'White')

Sort\_By\_Label.grid(row=0,column=0, padx = 5)

Newest\_Sort\_Button = Button(Order\_History\_Frame1,text='Newest', width=10,font=('arial',15,'bold'), bd = 5, relief='ridge',

bg='gold', fg='Black', command = self.Display\_Newest)

Newest\_Sort\_Button.grid(row = 0, column = 1, padx = 5)

Oldest\_Sort\_Button = Button(Order\_History\_Frame1,text='Oldest', width=10,font=('arial',15,'bold'), bd = 5, relief='ridge',

bg='gold', fg='Black', command = self.Display\_Oldest)

Oldest\_Sort\_Button.grid(row = 0, column = 2, padx = 5)

OrderID\_Label = Label(Order\_History\_Frame2, text = "OrderID", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'gold', fg = 'Black')

OrderID\_Label.grid(row=0,column=0)

Order\_PIP\_Label = Label(Order\_History\_Frame2, text = "PIP", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'gold', fg = 'Black')

Order\_PIP\_Label.grid(row=0,column=1)

Order\_Quantity\_Label = Label(Order\_History\_Frame2, text = "Order Quantity", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 15, bg = 'gold', fg = 'Black')

Order\_Quantity\_Label.grid(row=0,column=2)

Order\_Total\_Cost\_Label = Label(Order\_History\_Frame2, text = "Order Cost", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'gold', fg = 'Black')

Order\_Total\_Cost\_Label.grid(row=0,column=3)

Order\_Date\_Label = Label(Order\_History\_Frame2, text = "Date & Time", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 25, bg = 'gold', fg = 'Black')

Order\_Date\_Label.grid(row=0,column=4)

self.scrollbar\_V = Scrollbar(Order\_History\_Frame2)

self.scrollbar\_H\_OrderID = Scrollbar(Order\_History\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Order\_PIP = Scrollbar(Order\_History\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Order\_Quantity = Scrollbar(Order\_History\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Total\_Cost = Scrollbar(Order\_History\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Date = Scrollbar(Order\_History\_Frame2, orient=HORIZONTAL)

self.scrollbar\_V.grid(row = 1, column = 5, sticky=N+S+W)

self.scrollbar\_H\_OrderID.grid(row = 2, column = 0, sticky=N+E+S+W)

self.scrollbar\_H\_Order\_PIP.grid(row = 2, column = 1, sticky=N+E+S+W)

self.scrollbar\_H\_Order\_Quantity.grid(row = 2, column = 2, sticky=N+E+S+W)

self.scrollbar\_H\_Total\_Cost.grid(row = 2, column = 3, sticky=N+E+S+W)

self.scrollbar\_H\_Date.grid(row = 2, column = 4, sticky=N+E+S+W)

self.lbOrderID = Listbox(Order\_History\_Frame2, font=('arial',12,'bold'), width = 5, height = 20, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_OrderID.set)

self.lbOrderID.grid(row=1,column=0)

self.lbOrder\_PIP = Listbox(Order\_History\_Frame2, font=('arial',12,'bold'), width = 10, height = 20, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Order\_PIP.set)

self.lbOrder\_PIP.grid(row=1,column=1)

self.lbOrder\_Quantity = Listbox(Order\_History\_Frame2, font=('arial',12,'bold'), width = 15, height = 20, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Order\_Quantity.set)

self.lbOrder\_Quantity.grid(row=1,column=2)

self.lbTotal\_Cost = Listbox(Order\_History\_Frame2, font=('arial',12,'bold'), width = 10, height = 20, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Total\_Cost.set)

self.lbTotal\_Cost.grid(row=1,column=3)

self.lbDate = Listbox(Order\_History\_Frame2, font=('arial',12,'bold'), width = 20, height = 20, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Date.set)

self.lbDate.grid(row=1,column=4)

self.Display\_Newest()

self.scrollbar\_V.config(command=self.yview)

self.scrollbar\_H\_OrderID.config(command=self.lbOrderID.xview)

self.scrollbar\_H\_Order\_PIP.config(command=self.lbOrder\_PIP.xview)

self.scrollbar\_H\_Order\_Quantity.config(command=self.lbOrder\_Quantity.xview)

self.scrollbar\_H\_Total\_Cost.config(command = self.lbTotal\_Cost.xview)

self.scrollbar\_H\_Date.config(command = self.lbDate.xview)

Edit\_Button = Button(Order\_History\_Frame3,text='Show Order Information', width=20,font=('arial',15,'bold'), bd =10, relief='ridge',

bg='gold', fg='Black', command = self.Show\_Order\_Info)

Edit\_Button.grid(row = 0, column = 2, padx = 10, pady = 10)

Cancel\_Order\_Button = Button(Order\_History\_Frame3,text='Cancel Order', width=20,font=('arial',15,'bold'), bd =10, relief='ridge',

bg='gold', fg='Black', command = self.Delete\_Order)

Cancel\_Order\_Button.grid(row = 0, column = 3, padx = 10, pady = 10)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

#------------------------------------------------- Suppliers Tab ------------------------------------------------------#

#----------------------------------------------------- Frames ---------------------------------------------------------#

Suppliers\_Frame0 = Frame(Suppliers\_Frame, width=150, height=25, bg = "dark orchid")

Suppliers\_Frame0.grid(row = 0, column = 0, pady = 35)

Suppliers\_Frame1 = Frame(Suppliers\_Frame, width=150, height=25, bg = "dark orchid")

Suppliers\_Frame1.grid(row = 1, column = 0, padx = 35, pady = 5)

Suppliers\_Frame2 = Frame(Suppliers\_Frame, width=150, height=25, bg = "dark orchid")

Suppliers\_Frame2.grid(row = 2, column = 0, padx = 35, pady = 5)

Suppliers\_Frame3 = Frame(Suppliers\_Frame, width=100, height=25, bg = "dark orchid")

Suppliers\_Frame3.grid(row = 0, column = 1, pady = 10)

Suppliers\_Frame4 = Frame(Suppliers\_Frame, width=100, height=50, bg = "dark orchid")

Suppliers\_Frame4.grid(row = 1, column = 1, padx = 35, pady = 5)

Suppliers\_Frame5 = Frame(Suppliers\_Frame, width=500, height=25, bg = "dark orchid")

Suppliers\_Frame5.grid(row = 0, column = 2, padx = 35, pady = 5)

Suppliers\_Frame6 = Frame(Suppliers\_Frame, width=500, height=25, bg = "dark orchid")

Suppliers\_Frame6.grid(row = 1, column = 2, padx = 35, pady = 5)

Suppliers\_Frame7 = Frame(Suppliers\_Frame, width=500, height=25, bg = "dark orchid")

Suppliers\_Frame7.grid(row = 2, column = 2, padx = 35, pady = 5)

#--------------------------------------- Labels, Entry Boxes, and Buttons ---------------------------------------------#

Suppliers\_Information\_Title = Label(Suppliers\_Frame0, text = "Supplier Information", font=('arial',25,'bold'), bd = 20, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Suppliers\_Information\_Title.grid(row=0,column=0)

Chosen\_Suppliers\_Name\_Label = Label(Suppliers\_Frame1, text = "Supplier Name", font=('arial',15,'bold'), width = 15, bd = 15, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Chosen\_Suppliers\_Name\_Label.grid(row = 0,column=0, padx = 5)

Chosen\_Suppliers\_Phone\_Label = Label(Suppliers\_Frame1, text = "Phone Number", font=('arial',15,'bold'), width = 15, bd = 15, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Chosen\_Suppliers\_Phone\_Label.grid(row = 2,column=0, padx = 5)

Choesen\_Suppliers\_Email\_Label = Label(Suppliers\_Frame1, text = "Email", font=('arial',15,'bold'), width = 20, bd = 15, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Choesen\_Suppliers\_Email\_Label.grid(row = 4,column=0, padx = 5)

Chosen\_Suppliers\_Address\_Label = Label(Suppliers\_Frame1, text = "Address", font=('arial',15,'bold'), width = 20, bd = 15, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Chosen\_Suppliers\_Address\_Label.grid(row = 6,column=0, padx = 5)

global Chosen\_Suppliers\_Name\_Display

global Chosen\_Suppliers\_Phone\_Display

global Choesen\_Suppliers\_Email\_Display

global Chosen\_Suppliers\_Address\_Display

Chosen\_Suppliers\_Name\_Display = Listbox(Suppliers\_Frame1, font=('arial',15,'bold'), width = 20, height = 1, bd = 10, relief='ridge',

bg = 'White', fg = 'Black')

Chosen\_Suppliers\_Name\_Display.grid(row = 1,column=0, padx = 5, pady = 10)

Chosen\_Suppliers\_Phone\_Display = Listbox(Suppliers\_Frame1, font=('arial',15,'bold'), width = 15, height = 1, bd = 10, relief='ridge',

bg = 'White', fg = 'Black')

Chosen\_Suppliers\_Phone\_Display.grid(row = 3,column=0, padx = 5, pady = 10)

Choesen\_Suppliers\_Email\_Display = Listbox(Suppliers\_Frame1, font=('arial',15,'bold'), width = 25, height = 1, bd = 10, relief='ridge',

bg = 'White', fg = 'Black')

Choesen\_Suppliers\_Email\_Display.grid(row = 5,column=0, padx = 5, pady = 10)

Chosen\_Suppliers\_Address\_Display = Listbox(Suppliers\_Frame1, font=('arial',15,'bold'), width = 30, height = 1, bd = 10,relief='ridge',

bg = 'White', fg = 'Black')

Chosen\_Suppliers\_Address\_Display.grid(row = 7,column=0, padx = 5, pady = 10)

Display\_Suppliers\_Informaton\_Button = Button(Suppliers\_Frame2,text='Display Supplier Information', width=25,font=('arial',20,'bold'), bd =10, relief='ridge',

bg='Cadet Blue', fg='Cornsilk', command = self.Supplier\_Details)

Display\_Suppliers\_Informaton\_Button.grid(row = 0, column = 0)

def Display\_Suppliers\_Name():

Suppliers\_Names.clear()

self.Suppliers\_Listbox.delete(0, END)

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Supply\_Name FROM "Supplier"')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Suppliers\_Names.append(row[0])

rowNr = rowNr + 1

self.Suppliers\_Listbox.insert("end", \*Suppliers\_Names)

Suppliers\_Title = Label(Suppliers\_Frame3, text = "List Of Suppliers", font=('arial',25,'bold'), bd = 25, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Suppliers\_Title.grid(row=0,column=0)

Suppliers\_Name\_Label = Label(Suppliers\_Frame4, text = "Suppliers", font=('arial',25,'bold'), bd = 25, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Suppliers\_Name\_Label.grid(row = 0,column=0)

scrollbar\_V = Scrollbar(Suppliers\_Frame4)

scrollbar\_H\_Suppliers = Scrollbar(Suppliers\_Frame4, orient=HORIZONTAL)

scrollbar\_V.grid(row = 1, column = 1, sticky=N+S+W)

scrollbar\_H\_Suppliers.grid(row = 2, column = 0, sticky=N+E+S+W)

self.Suppliers\_Listbox = Listbox(Suppliers\_Frame4, font=('arial',15,'bold'), width = 25

, height = 15, bg ='Cadet Blue', bd = 10, yscrollcommand=scrollbar\_V.set, xscrollcommand=scrollbar\_H\_Suppliers.set)

self.Suppliers\_Listbox.grid(row=1,column=0,pady = 5)

scrollbar\_V.config(command=self.Suppliers\_Listbox.yview)

scrollbar\_H\_Suppliers.config(command=self.Suppliers\_Listbox.xview)

Display\_Suppliers\_Name()

Add\_New\_Supplier\_Title = Label(Suppliers\_Frame5, text = "Add New Supplier", font=('arial',25,'bold'), bd = 25, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Add\_New\_Supplier\_Title.grid(row = 2,column=0, padx = 5)

Add\_Suppliers\_Name\_Label = Label(Suppliers\_Frame6, text = "Supplier Name", font=('arial',15,'bold'), width = 15, bd = 15, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Add\_Suppliers\_Name\_Label.grid(row = 0,column=0, padx = 5)

Add\_Suppliers\_Phone\_Label = Label(Suppliers\_Frame6, text = "Phone Number", font=('arial',15,'bold'), width = 15, bd = 15, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Add\_Suppliers\_Phone\_Label.grid(row = 2,column=0, padx = 5)

Add\_Suppliers\_Email\_Label = Label(Suppliers\_Frame6, text = "Email", font=('arial',15,'bold'), width = 20, bd = 15, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Add\_Suppliers\_Email\_Label.grid(row = 4,column=0, padx = 5)

Add\_Suppliers\_Address\_Label = Label(Suppliers\_Frame6, text = "Address", font=('arial',15,'bold'), width = 20, bd = 15, relief='ridge',

bg = 'Cadet Blue', fg = 'Black')

Add\_Suppliers\_Address\_Label.grid(row = 6,column=0, padx = 5)

global Add\_Suppliers\_Name

global Add\_Suppliers\_Phone

global Add\_Suppliers\_Email

global Add\_Suppliers\_Address

global Supplier\_Name\_Entered

global Phone\_Entered

global Email\_Entered

global Address\_Entered

Supplier\_Name\_Entered = StringVar()

Phone\_Entered = StringVar()

Email\_Entered = StringVar()

Address\_Entered = StringVar()

Add\_Suppliers\_Name = Entry(Suppliers\_Frame6, font=('arial',15,'bold'), textvariable = Supplier\_Name\_Entered, width = 20, bd = 10, relief='ridge',

bg = 'White', fg = 'Black')

Add\_Suppliers\_Name.grid(row = 1,column=0, padx = 5, pady = 10)

Add\_Suppliers\_Phone = Entry(Suppliers\_Frame6, font=('arial',15,'bold'), textvariable = Phone\_Entered, width = 15, bd = 10, relief='ridge',

bg = 'White', fg = 'Black')

Add\_Suppliers\_Phone.grid(row = 3,column=0, padx = 5, pady = 10)

Add\_Suppliers\_Email = Entry(Suppliers\_Frame6, font=('arial',15,'bold'), textvariable = Email\_Entered, width = 25, bd = 10, relief='ridge',

bg = 'White', fg = 'Black')

Add\_Suppliers\_Email.grid(row = 5,column=0, padx = 5, pady = 10)

Add\_Suppliers\_Address = Entry(Suppliers\_Frame6, font=('arial',15,'bold'), textvariable = Address\_Entered, width = 30, bd = 10,relief='ridge',

bg = 'White', fg = 'Black')

Add\_Suppliers\_Address.grid(row = 7,column=0, padx = 5, pady = 10)

Add\_New\_Supplier\_Button = Button(Suppliers\_Frame7,text='Add New Supplier', width=20,font=('arial',20,'bold'), bd =10, relief='ridge',

bg='Cadet Blue', fg='Cornsilk', command = self.Add\_New\_Supplier)

Add\_New\_Supplier\_Button.grid(row = 0, column = 0)

#--------------------------------------------------- Functions --------------------------------------------------------#

#------------------------------------------ Stock Management Tab Functions --------------------------------------------#

def Clear\_List(self):

self.lbPIP.delete(0, END)

self.lbBrand\_Name.delete(0, END)

self.lbProduct\_Discription.delete(0, END)

self.lbQuantity.delete(0, END)

def Display\_Stock\_PIP(self):

PIP.clear()

Brand\_Name.clear()

Product\_Discription.clear()

Quantity.clear()

self.Clear\_List()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP, Brand\_Name, Product\_Discription, Quantity FROM "Product\_Info", "Brands" WHERE Product\_Info.BrandID = Brands.BrandID')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

PIP.append(row[0])

Brand\_Name.append(row[1])

Product\_Discription.append(row[2])

Quantity.append(row[3])

rowNr = rowNr + 1

self.lbPIP.insert("end", \*PIP)

self.lbBrand\_Name.insert("end", \*Brand\_Name)

self.lbProduct\_Discription.insert("end", \*Product\_Discription)

self.lbQuantity.insert("end", \*Quantity)

def Display\_Stock\_From\_Lowest(self):

PIP.clear()

Brand\_Name.clear()

Product\_Discription.clear()

Quantity.clear()

self.Clear\_List()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP, Brand\_Name, Product\_Discription, Quantity FROM "Product\_Info", "Brands" WHERE Product\_Info.BrandID = Brands.BrandID ORDER BY Quantity')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

PIP.append(row[0])

Brand\_Name.append(row[1])

Product\_Discription.append(row[2])

Quantity.append(row[3])

rowNr = rowNr + 1

self.lbPIP.insert("end", \*PIP)

self.lbBrand\_Name.insert("end", \*Brand\_Name)

self.lbProduct\_Discription.insert("end", \*Product\_Discription)

self.lbQuantity.insert("end", \*Quantity)

def Display\_Stock\_From\_Highest (self):

PIP.clear()

Brand\_Name.clear()

Product\_Discription.clear()

Quantity.clear()

self.Clear\_List()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP, Brand\_Name, Product\_Discription, Quantity FROM "Product\_Info", "Brands" WHERE Product\_Info.BrandID = Brands.BrandID ORDER BY Quantity desc')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

PIP.append(row[0])

Brand\_Name.append(row[1])

Product\_Discription.append(row[2])

Quantity.append(row[3])

rowNr = rowNr + 1

self.lbPIP.insert("end", \*PIP)

self.lbBrand\_Name.insert("end", \*Brand\_Name)

self.lbProduct\_Discription.insert("end", \*Product\_Discription)

self.lbQuantity.insert("end", \*Quantity)

def Display\_Stock\_A2Z(self):

PIP.clear()

Brand\_Name.clear()

Product\_Discription.clear()

Quantity.clear()

self.Clear\_List()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP, Brand\_Name, Product\_Discription, Quantity FROM "Product\_Info", "Brands" WHERE Product\_Info.BrandID = Brands.BrandID ORDER BY Brand\_Name')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

PIP.append(row[0])

Brand\_Name.append(row[1])

Product\_Discription.append(row[2])

Quantity.append(row[3])

rowNr = rowNr + 1

self.lbPIP.insert("end", \*PIP)

self.lbBrand\_Name.insert("end", \*Brand\_Name)

self.lbProduct\_Discription.insert("end", \*Product\_Discription)

self.lbQuantity.insert("end", \*Quantity)

def Display\_Stock\_Z2A(self):

PIP.clear()

Brand\_Name.clear()

Product\_Discription.clear()

Quantity.clear()

self.Clear\_List()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP, Brand\_Name, Product\_Discription, Quantity FROM "Product\_Info", "Brands" WHERE Product\_Info.BrandID = Brands.BrandID ORDER BY Brand\_Name desc')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

PIP.append(row[0])

Brand\_Name.append(row[1])

Product\_Discription.append(row[2])

Quantity.append(row[3])

rowNr = rowNr + 1

self.lbPIP.insert("end", \*PIP)

self.lbBrand\_Name.insert("end", \*Brand\_Name)

self.lbProduct\_Discription.insert("end", \*Product\_Discription)

self.lbQuantity.insert("end", \*Quantity)

def edit\_stock(self):

PIP.clear()

Brand\_Name.clear()

Product\_Discription.clear()

Quantity.clear()

try:

PIP\_chosen = self.lbPIP.get(self.lbPIP.curselection())

except Exception:

messagebox.showerror("No Product Chosen!", "Please select a PIP number and try again!")

else:

Edit\_Quantity\_Window = Toplevel()

Edit\_Quantity\_Window.geometry("795x225")

Edit\_Quantity\_Window.config(bg ='MediumOrchid2')

Frame1 = Frame(Edit\_Quantity\_Window, bg = 'MediumOrchid2')

Frame1.grid(row = 0, column = 0)

Frame2 = Frame(Edit\_Quantity\_Window, bg = 'MediumOrchid2')

Frame2.grid(row = 1, column = 0)

Quantity\_Label = Label(Frame1, text = "Enter Quantity", font =('arial',30,'bold'),bd=20, relief='ridge',

bg = 'SlateBlue2', fg = 'Cornsilk')

Quantity\_Label.grid(row = 0 , column = 0)

Quantity\_Entry = StringVar()

New\_Quantity = Entry(Frame1, font =('arial',30,'bold'),bd=20, relief='ridge', textvariable = Quantity\_Entry,

bg = 'SlateBlue2', fg = 'Cornsilk')

New\_Quantity.grid(row = 0 , column = 1)

def Update\_Quantity():

try:

Quantity\_Entered = int(New\_Quantity.get())

except ValueError:

messagebox.showerror("Invalid Data Entered!", "Please enter an integer value for the quantity! Try Again!")

Quantity\_Entry.set("")

if (Quantity\_Entered > -1) and (type(Quantity\_Entered) == int):

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur = con.execute('UPDATE "Product\_Info" SET Quantity=? WHERE PIP=?',\

(Quantity\_Entered, PIP\_chosen))

con.commit()

con.close()

Edit\_Quantity\_Window.destroy()

self.Clear\_List()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP, Brand\_Name, Product\_Discription, Quantity FROM "Product\_Info", "Brands" WHERE Product\_Info.BrandID = Brands.BrandID')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

PIP.append(row[0])

Brand\_Name.append(row[1])

Product\_Discription.append(row[2])

Quantity.append(row[3])

rowNr = rowNr + 1

self.lbPIP.insert("end", \*PIP)

self.lbBrand\_Name.insert("end", \*Brand\_Name)

self.lbProduct\_Discription.insert("end", \*Product\_Discription)

self.lbQuantity.insert("end", \*Quantity)

else:

messagebox.showerror("Invalid data entered", "Please enter a valid quantity amount! Try Again!")

Quantity\_Entry.set("")

Update\_Quantity\_Button = Button (Frame2,text='Update Quantity', width=15,font=('arial',30,'bold'), bd = 10, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = Update\_Quantity)

Update\_Quantity\_Button.grid(row = 0 , column = 0, padx = 100, pady = 25)

def Product\_Info(self):

PIP\_Info = []

Brand\_Name\_Info = []

Product\_Discription\_Info = []

Pack\_Size\_Info = []

Product\_Cost\_Info = []

Use\_Info = []

Supply\_Name\_Info = []

Quantity\_Info = []

try:

PIP\_chosen = self.lbPIP.get(self.lbPIP.curselection())

except Exception:

messagebox.showerror("No Product Chosen!", "Please select a PIP number and try again!")

else:

PIP\_Info.clear()

Brand\_Name\_Info.clear()

Product\_Discription\_Info.clear()

Pack\_Size\_Info.clear()

Product\_Cost\_Info.clear()

Use\_Info.clear()

Supply\_Name\_Info.clear()

Quantity\_Info.clear()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP, Product\_Discription, Pack\_Size, Product\_Cost, Quantity FROM "Product\_Info" WHERE (PIP = (:PIP\_Selected))',

{

'PIP\_Selected' : PIP\_chosen

})

Product\_Info\_Rows = cur.fetchall()

con.close()

rowNr = 0

for row in Product\_Info\_Rows:

if len(row) == 0:

continue

if rowNr >= 0:

PIP\_Info.append(row[0])

Product\_Discription\_Info.append(row[1])

Pack\_Size\_Info.append(row[2])

Product\_Cost\_Info.append(row[3])

Quantity\_Info.append(row[4])

rowNr = rowNr + 1

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Brand\_Name,Use,Supply\_Name FROM "Product\_Info","Brands","Uses","Supplier" WHERE (PIP = (:PIP\_Selected)) AND Product\_Info.BrandID = Brands.BrandID AND Product\_Info.UseID = Uses.UseID AND Product\_Info.SupplyID = Supplier.SupplyID',

{

'PIP\_Selected' : PIP\_chosen

})

Other\_Rows = cur.fetchall()

con.close()

rowNr = 0

for row in Other\_Rows:

if len(row) < 1:

continue

if rowNr >= 0:

Brand\_Name\_Info.append(row[0])

Use\_Info.append(row[1])

Supply\_Name\_Info.append(row[2])

rowNr = rowNr + 1

Product\_Information = Toplevel()

Product\_Information.geometry("895x380")

Product\_Information.config(bg ='Cadet Blue')

Frame0 = Frame(Product\_Information, bg = "Cadet Blue")

Frame0.grid(row = 0, column = 0, pady = 5)

Frame1 = Frame(Product\_Information, bg = "Cadet Blue")

Frame1.grid(row = 1, column = 0, padx = 5, pady = 5)

Frame2 = Frame(Product\_Information, bg = "Cadet Blue")

Frame2.grid(row = 2, column = 0, pady = 5)

Frame3 = Frame(Product\_Information, bg = "Cadet Blue")

Frame3.grid(row = 3, column = 0, pady = 5)

Product\_Info\_Title = Label(Frame0, text = "Product Information", width = 30, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Product\_Info\_Title.grid(row = 0, column = 0)

PIP\_Label = Label(Frame1, text = "PIP", width = 15, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

PIP\_Label.grid(row = 0, column = 0)

PIP\_Display = Listbox(Frame1, width = 15, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

PIP\_Display.grid(row = 1, column = 0)

Brand\_Name\_Label = Label(Frame1, text = "Brand Name", width = 15, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Brand\_Name\_Label.grid(row = 0, column = 1)

Brand\_Name\_Display = Listbox(Frame1, width = 15, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Brand\_Name\_Display.grid(row = 1, column = 1)

Product\_Discription\_Label = Label(Frame1, text = "Product Discription", width = 20, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Product\_Discription\_Label.grid(row = 0, column = 2)

Product\_Discription\_Display = Listbox(Frame1, width = 20, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Product\_Discription\_Display.grid(row = 1, column = 2)

Quantity\_Label = Label(Frame2, text = "Quantity", width = 25, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Quantity\_Label.grid(row = 0, column = 0)

Quantity\_Display = Listbox(Frame2, width = 25, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Quantity\_Display.grid(row = 1, column = 0)

Product\_Cost\_Label = Label(Frame2, text = "Product Cost", width = 25, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Product\_Cost\_Label.grid(row = 0, column = 1)

Product\_Cost\_Display = Listbox(Frame2, width = 25, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Product\_Cost\_Display.grid(row = 1, column = 1)

Use\_Label = Label(Frame3, text = "Use", width = 25, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Use\_Label.grid(row = 0, column = 0)

Use\_Display = Listbox(Frame3, width = 25, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Use\_Display.grid(row = 1, column = 0)

Supply\_Name\_Label = Label(Frame3, text = "Supplier", width = 25, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Supply\_Name\_Label.grid(row = 0, column = 1)

Supply\_Name\_Display = Listbox(Frame3, width = 25, height = 1, font =('arial',20,'bold'), bd=5,

bg = 'White', fg = 'Black')

Supply\_Name\_Display.grid(row = 1, column = 1)

PIP\_Display.insert(0,PIP\_Info[0])

Brand\_Name\_Display.insert(0,Brand\_Name\_Info[0])

Product\_Discription\_Display.insert(0,Product\_Discription\_Info[0])

Quantity\_Display.insert(0,Quantity\_Info[0])

Product\_Cost\_Display.insert(0,Product\_Cost\_Info[0])

Use\_Display.insert(0,Use\_Info[0])

Supply\_Name\_Display.insert(0,Supply\_Name\_Info[0])

def New\_Product(self):

self.Clear\_List()

self.New\_Product = Toplevel()

self.app = New\_Product\_Window(self.New\_Product)

def Delete\_Product(self):

Brand\_Name\_Chosen = []

Product\_Discription\_Chosen = []

PIP\_Chosen = self.lbPIP.get(ACTIVE)

try:

PIP\_chosen = self.lbPIP.get(self.lbPIP.curselection())

except Exception:

messagebox.showerror("No Product Chosen!", "Please select a PIP number and try again!")

else:

Delete\_Product\_Win = Toplevel()

Delete\_Product\_Win.geometry("785x280")

Delete\_Product\_Win.config(bg ='Cadet Blue')

Frame0 = Frame(Delete\_Product\_Win, bg = "Cadet Blue")

Frame0.grid(row = 0, column = 0, pady = 5)

Frame1 = Frame(Delete\_Product\_Win, bg = "Cadet Blue")

Frame1.grid(row = 1, column = 0, padx = 5, pady = 5)

Frame2 = Frame(Delete\_Product\_Win, bg = "Cadet Blue")

Frame2.grid(row = 2, column = 0, pady = 5)

Title = Label(Frame0, text = "Delete Product", width = 25, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Title.grid(row = 0, column = 0)

Delete\_PIP\_Display\_Label = Label(Frame1, text = "PIP", width = 10, font =('arial',20,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Delete\_PIP\_Display\_Label.grid(row = 0, column = 0)

Delete\_Brand\_Display\_Label = Label(Frame1, text = "Brand", width = 10, font =('arial',20,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Delete\_Brand\_Display\_Label.grid(row = 0, column = 1)

Delete\_Product\_Discription\_Display\_Label = Label(Frame1, text = "Product Discription", width = 20, font =('arial',20,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Delete\_Product\_Discription\_Display\_Label.grid(row = 0, column = 2)

def Selected\_Product():

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Brand\_Name, Product\_Discription FROM "Product\_Info","Brands" WHERE (PIP = (:PIP\_Selected)) AND Product\_Info.BrandID = Brands.BrandID',

{

'PIP\_Selected' : PIP\_Chosen

})

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Brand\_Name\_Chosen.append(row[0])

Product\_Discription\_Chosen.append(row[1])

rowNr = rowNr + 1

Delete\_PIP\_Display.insert(0,PIP\_Chosen)

Delete\_Brand\_Name\_Display.insert(0,\*Brand\_Name\_Chosen)

Delete\_Product\_Discription\_Display.insert(0,\*Product\_Discription\_Chosen)

Delete\_PIP\_Display = Listbox(Frame1, width = 10, height = 1, font =('arial',15,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Delete\_PIP\_Display.grid(row = 1, column = 0)

Delete\_Brand\_Name\_Display = Listbox(Frame1, width = 15, height = 1, font =('arial',15,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Delete\_Brand\_Name\_Display.grid(row = 1, column = 1)

Delete\_Product\_Discription\_Display = Listbox(Frame1, width = 30, height = 1, font =('arial',15,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Delete\_Product\_Discription\_Display.grid(row = 1, column = 2)

Selected\_Product()

def Delete():

self.confirm = tkinter.messagebox.askyesno("Confirm Deletion of Data!", "Are you sure you want to delete this record?")

if self.confirm > 0:

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('DELETE FROM "Product\_Info" WHERE (PIP = (:PIP\_Selected))',

{

'PIP\_Selected' : PIP\_Chosen

})

con.commit()

con.close()

Delete\_Product\_Win.destroy()

PIP.clear()

Brand\_Name.clear()

Product\_Discription.clear()

Quantity.clear()

self.Clear\_List()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP, Brand\_Name, Product\_Discription, Quantity FROM "Product\_Info", "Brands" WHERE Product\_Info.BrandID = Brands.BrandID')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

PIP.append(row[0])

Brand\_Name.append(row[1])

Product\_Discription.append(row[2])

Quantity.append(row[3])

rowNr = rowNr + 1

self.lbPIP.insert("end", \*PIP)

self.lbBrand\_Name.insert("end", \*Brand\_Name)

self.lbProduct\_Discription.insert("end", \*Product\_Discription)

self.lbQuantity.insert("end", \*Quantity)

else:

Delete\_Product\_Win.destroy()

Delete\_Button = Button(Frame2,text='Delete Product', width=15,font=('arial',20,'bold'), bd = 10, relief='ridge',

bg='SeaGreen3', fg='Cornsilk', command = Delete)

Delete\_Button.grid(row = 0, column = 0)

#---------------------------------------------- Search Tab Functions --------------------------------------------------#

def Start\_Search(self):

Searched\_PIP = []

Searched\_Brand = []

Searched\_Product\_Discription = []

Searched\_Use = []

Searched\_Supplier = []

Searched\_Quantity = []

Similar\_PIP = []

Similar\_Brand = []

Similar\_Product\_Discription = []

Similar\_Use = []

Similar\_Supplier = []

Similar\_Quantity = []

def Clear\_Search\_Lists():

Searched\_PIP.clear()

Searched\_Brand.clear()

Searched\_Product\_Discription.clear()

Searched\_Use.clear()

Searched\_Supplier.clear()

Searched\_Quantity.clear()

Similar\_PIP.clear()

Similar\_Brand.clear()

Similar\_Product\_Discription.clear()

Similar\_Use.clear()

Similar\_Supplier.clear()

Similar\_Quantity.clear()

if Search\_By\_Menu.get() == "Search By":

messagebox.showerror("No Search Filter Selected!","Please select a Search By option to filter your search before countinuing! Try Again!")

else:

def Frames\_Labels\_and\_Lists():

Search\_Frame2 = LabelFrame(Search\_Frame, width = 500, height=150, text = "Search Results", font=('arial',25,'bold'), bd = 20, relief='ridge', bg = "plum2")

Search\_Frame2.grid(row = 2, column =0, pady = 5)

self.scrollbar\_V = Scrollbar(Search\_Frame2)

self.scrollbar\_H\_Search\_PIP = Scrollbar(Search\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Search\_Brand = Scrollbar(Search\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Search\_Product\_Discription = Scrollbar(Search\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Search\_Use = Scrollbar(Search\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Search\_Supplier = Scrollbar(Search\_Frame2, orient=HORIZONTAL)

self.scrollbar\_H\_Search\_Quantity = Scrollbar(Search\_Frame2, orient=HORIZONTAL)

self.scrollbar\_V.grid(row = 1, column = 6, sticky=N+S+W, padx = 5)

self.scrollbar\_H\_Search\_PIP.grid(row = 2, column = 0, sticky=N+E+S+W)

self.scrollbar\_H\_Search\_Brand.grid(row = 2, column = 1, sticky=N+E+S+W)

self.scrollbar\_H\_Search\_Product\_Discription.grid(row = 2, column = 2, sticky=N+E+S+W)

self.scrollbar\_H\_Search\_Use.grid(row = 2, column = 3, sticky=N+E+S+W)

self.scrollbar\_H\_Search\_Supplier.grid(row = 2, column = 4, sticky=N+E+S+W)

self.scrollbar\_H\_Search\_Quantity.grid(row = 2, column = 5, sticky=N+E+S+W)

Searched\_PIP\_Label = Label(Search\_Frame2, text = "PIP", font=('arial',15,'bold'), width = 15, bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Searched\_PIP\_Label.grid(row=0,column=0, padx = 5)

self.Searched\_PIP\_List = Listbox(Search\_Frame2, font=('arial',15,'bold'), height = 4, width = 15, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Search\_PIP.set)

self.Searched\_PIP\_List.grid(row=1, column=0, padx = 5)

Searched\_Brand\_Label = Label(Search\_Frame2, text = "Brand", font=('arial',15,'bold'), width = 15, bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Searched\_Brand\_Label.grid(row=0,column=1, padx = 5)

self.Searched\_Brand\_List = Listbox(Search\_Frame2, font=('arial',15,'bold'), height = 4, width = 15, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Search\_Brand.set)

self.Searched\_Brand\_List.grid(row=1, column=1, padx = 5)

Searched\_Product\_Discription\_Label = Label(Search\_Frame2, text = "Product Discription", width = 15, font=('arial',15,'bold'), bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Searched\_Product\_Discription\_Label.grid(row=0,column=2, padx = 5)

self.Searched\_Product\_Discription\_List = Listbox(Search\_Frame2, font=('arial',15,'bold'), height = 4, width = 15, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Search\_Product\_Discription.set)

self.Searched\_Product\_Discription\_List.grid(row=1, column=2, padx = 5)

Searched\_Use\_Label = Label(Search\_Frame2, text = "Use", font=('arial',15,'bold'), width = 15, bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Searched\_Use\_Label.grid(row=0,column=3, padx = 5)

self.Searched\_Use\_List = Listbox(Search\_Frame2, font=('arial',15,'bold'), height = 4, width = 15, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Search\_Use.set)

self.Searched\_Use\_List.grid(row=1, column=3, padx = 5)

Searched\_Supplier\_Label = Label(Search\_Frame2, text = "Supplier", font=('arial',15,'bold'),width =15, bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Searched\_Supplier\_Label.grid(row=0,column=4, padx = 5)

self.Searched\_Supplier\_List = Listbox(Search\_Frame2, font=('arial',15,'bold'), height = 4, width = 15, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Search\_Supplier.set)

self.Searched\_Supplier\_List.grid(row=1, column=4, padx = 5)

Searched\_Quantity\_Label = Label(Search\_Frame2, text = "Quantity", font=('arial',15,'bold'), width =10, bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Searched\_Quantity\_Label.grid(row=0,column=5, padx = 5)

self.Searched\_Quantity\_List = Listbox(Search\_Frame2, font=('arial',15,'bold'), height = 4, width = 5, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Search\_Quantity.set)

self.Searched\_Quantity\_List.grid(row=1, column=5, padx = 5)

self.scrollbar\_V.config(command=self.yview)

self.scrollbar\_H\_Search\_PIP.config(command=self.Searched\_PIP\_List.xview)

self.scrollbar\_H\_Search\_Brand.config(command=self.Searched\_Brand\_List.xview)

self.scrollbar\_H\_Search\_Product\_Discription.config(command=self.Searched\_Product\_Discription\_List.xview)

self.scrollbar\_H\_Search\_Use.config(command = self.Searched\_Use\_List.xview)

self.scrollbar\_H\_Search\_Supplier.config(command = self.Searched\_Supplier\_List.xview)

self.scrollbar\_H\_Search\_Quantity.config(command = self.Searched\_Quantity\_List.xview)

def Similar\_Labels\_and\_Lists():

Search\_Frame3 = LabelFrame(Search\_Frame, width=1350, height=150, text = "Similar Products", font=('arial',25,'bold'), bd = 20, relief='ridge', bg = "plum2")

Search\_Frame3.grid(row = 4, column =0, pady = 3)

self.scrollbar\_V = Scrollbar(Search\_Frame3)

self.scrollbar\_H\_Similar\_PIP = Scrollbar(Search\_Frame3, orient=HORIZONTAL)

self.scrollbar\_H\_Similar\_Brand = Scrollbar(Search\_Frame3, orient=HORIZONTAL)

self.scrollbar\_H\_Similar\_Product\_Discription = Scrollbar(Search\_Frame3, orient=HORIZONTAL)

self.scrollbar\_H\_Similar\_Use = Scrollbar(Search\_Frame3, orient=HORIZONTAL)

self.scrollbar\_H\_Similar\_Supplier = Scrollbar(Search\_Frame3, orient=HORIZONTAL)

self.scrollbar\_H\_Similar\_Quantity = Scrollbar(Search\_Frame3, orient=HORIZONTAL)

self.scrollbar\_V.grid(row = 1, column = 6, sticky=N+S+W, padx = 5)

self.scrollbar\_H\_Similar\_PIP.grid(row = 2, column = 0, sticky=N+E+S+W)

self.scrollbar\_H\_Similar\_Brand.grid(row = 2, column = 1, sticky=N+E+S+W)

self.scrollbar\_H\_Similar\_Product\_Discription.grid(row = 2, column = 2, sticky=N+E+S+W)

self.scrollbar\_H\_Similar\_Use.grid(row = 2, column = 3, sticky=N+E+S+W)

self.scrollbar\_H\_Similar\_Supplier.grid(row = 2, column = 4, sticky=N+E+S+W)

self.scrollbar\_H\_Similar\_Quantity.grid(row = 2, column = 5, sticky=N+E+S+W)

Similar\_PIP\_Label = Label(Search\_Frame3, text = "PIP", font=('arial',15,'bold'), width = 15, bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Similar\_PIP\_Label.grid(row=0,column=0, padx = 10)

global Similar\_PIP\_List

self.Similar\_PIP\_List = Listbox(Search\_Frame3, font=('arial',15,'bold'), height = 4, width = 15, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Similar\_PIP.set)

self.Similar\_PIP\_List.grid(row=1, column=0, padx = 10)

Similar\_Brand\_Label = Label(Search\_Frame3, text = "Brand", font=('arial',15,'bold'), width = 15, bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Similar\_Brand\_Label.grid(row=0,column=1, padx = 10)

global Similar\_Brand\_List

self.Similar\_Brand\_List = Listbox(Search\_Frame3, font=('arial',15,'bold'), height = 4, width = 15, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Similar\_Brand.set)

self.Similar\_Brand\_List.grid(row=1, column=1, padx = 10)

Similar\_Product\_Discription\_Label = Label(Search\_Frame3, text = "Product Discription", width = 15, font=('arial',15,'bold'), bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Similar\_Product\_Discription\_Label.grid(row=0,column=2, padx = 10)

global Similar\_Product\_Discription\_List

self.Similar\_Product\_Discription\_List = Listbox(Search\_Frame3, font=('arial',15,'bold'), height = 4, width = 15, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Similar\_Product\_Discription.set)

self.Similar\_Product\_Discription\_List.grid(row=1, column=2, padx = 10)

Similar\_Use\_Label = Label(Search\_Frame3, text = "Use", font=('arial',15,'bold'), width = 15, bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Similar\_Use\_Label.grid(row=0,column=3, padx = 10)

global Similar\_Use\_List

self.Similar\_Use\_List = Listbox(Search\_Frame3, font=('arial',15,'bold'), height = 4, width = 15, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Similar\_Use.set)

self.Similar\_Use\_List.grid(row=1, column=3, padx = 10)

Similar\_Supplier\_Label = Label(Search\_Frame3, text = "Supplier", font=('arial',15,'bold'),width =15, bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Similar\_Supplier\_Label.grid(row=0,column=4, padx = 10)

global Similar\_Supplier\_List

self.Similar\_Supplier\_List = Listbox(Search\_Frame3, font=('arial',15,'bold'), height = 4, width = 15, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Similar\_Supplier.set)

self.Similar\_Supplier\_List.grid(row=1, column=4, padx = 10)

Similar\_Quantity\_Label = Label(Search\_Frame3, text = "Quantity", font=('arial',15,'bold'), width =10, bd = 10, relief='ridge',

bg = 'Maroon3', fg = 'White')

Similar\_Quantity\_Label.grid(row=0,column=5, padx = 10)

global Similar\_Quantity\_List

self.Similar\_Quantity\_List = Listbox(Search\_Frame3, font=('arial',15,'bold'), height = 4, width = 5, bd = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Similar\_Quantity.set)

self.Similar\_Quantity\_List.grid(row=1, column=5, padx = 10)

self.scrollbar\_V.config(command=self.yview)

self.scrollbar\_H\_Similar\_PIP.config(command=self.Similar\_PIP\_List.xview)

self.scrollbar\_H\_Similar\_Brand.config(command=self.Similar\_Brand\_List.xview)

self.scrollbar\_H\_Similar\_Product\_Discription.config(command=self.Similar\_Product\_Discription\_List.xview)

self.scrollbar\_H\_Similar\_Use.config(command = self.Similar\_Use\_List.xview)

self.scrollbar\_H\_Similar\_Supplier.config(command = self.Similar\_Supplier\_List.xview)

self.scrollbar\_H\_Similar\_Quantity.config(command = self.Similar\_Quantity\_List.xview)

if Search\_By\_Menu.get() == "Use":

All\_Uses = []

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Use FROM "Uses"')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

All\_Uses.append(row[0])

rowNr = rowNr + 1

All\_Uses\_lower = [x.lower() for x in All\_Uses]

Search\_By\_Use = Search\_Bar\_Entry.get()

if Search\_By\_Use.lower() not in All\_Uses\_lower:

messagebox.showerror("No Use Exists!", "This Pharmacy does not have any products with this Use! Try Again!")

Search\_Bar\_Entry.set("")

else:

New\_Search\_By\_Use = All\_Uses[All\_Uses\_lower.index(Search\_By\_Use.lower())]

Frames\_Labels\_and\_Lists()

Clear\_Search\_Lists()

Search\_By\_Use = Search\_Bar\_Entry.get()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('''SELECT PIP, Brand\_Name, Product\_Discription, Use, Supply\_Name, Quantity FROM "Product\_Info", "Brands", "Uses", "Supplier"

WHERE Use = (:Searching\_Use) AND Product\_Info.BrandID = Brands.BrandID AND Product\_Info.UseID = Uses.UseID AND Product\_Info.SupplyID = Supplier.SupplyID''',

{

"Searching\_Use" : New\_Search\_By\_Use

})

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Searched\_PIP.append(row[0])

Searched\_Brand.append(row[1])

Searched\_Product\_Discription.append(row[2])

Searched\_Use.append(row[3])

Searched\_Supplier.append(row[4])

Searched\_Quantity.append(row[5])

rowNr = rowNr + 1

self.Searched\_PIP\_List.insert(0,\*Searched\_PIP)

self.Searched\_Brand\_List.insert(0,\*Searched\_Brand)

self.Searched\_Product\_Discription\_List.insert(0,\*Searched\_Product\_Discription)

self.Searched\_Use\_List.insert(0,\*Searched\_Use)

self.Searched\_Supplier\_List.insert(0,\*Searched\_Supplier)

self.Searched\_Quantity\_List.insert(0,\*Searched\_Quantity)

elif Search\_By\_Menu.get() == "Supplier":

All\_Suppliers = []

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Supply\_Name FROM "Supplier"')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

All\_Suppliers.append(row[0])

rowNr = rowNr + 1

All\_Suppliers\_lower = [x.lower() for x in All\_Suppliers]

Search\_By\_Suppliers = Search\_Bar\_Entry.get()

if Search\_By\_Suppliers.lower() not in All\_Suppliers\_lower:

messagebox.showerror("No supplier!", "This Pharmacy does not currently use this supplier! Try Again!")

Search\_Bar\_Entry.set("")

else:

New\_Search\_By\_Suppliers = All\_Suppliers[All\_Suppliers\_lower.index(Search\_By\_Suppliers.lower())]

Frames\_Labels\_and\_Lists()

Clear\_Search\_Lists()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('''SELECT PIP, Brand\_Name, Product\_Discription, Use, Supply\_Name, Quantity FROM "Product\_Info", "Brands", "Uses", "Supplier"

WHERE Supply\_Name = (:Searching\_Supplier) AND Product\_Info.BrandID = Brands.BrandID AND Product\_Info.UseID = Uses.UseID AND Product\_Info.SupplyID = Supplier.SupplyID''',

{

"Searching\_Supplier" : New\_Search\_By\_Suppliers

})

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Searched\_PIP.append(row[0])

Searched\_Brand.append(row[1])

Searched\_Product\_Discription.append(row[2])

Searched\_Use.append(row[3])

Searched\_Supplier.append(row[4])

Searched\_Quantity.append(row[5])

rowNr = rowNr + 1

self.Searched\_PIP\_List.insert(0,\*Searched\_PIP)

self.Searched\_Brand\_List.insert(0,\*Searched\_Brand)

self.Searched\_Product\_Discription\_List.insert(0,\*Searched\_Product\_Discription)

self.Searched\_Use\_List.insert(0,\*Searched\_Use)

self.Searched\_Supplier\_List.insert(0,\*Searched\_Supplier)

self.Searched\_Quantity\_List.insert(0,\*Searched\_Quantity)

elif Search\_By\_Menu.get() == "PIP":

All\_PIPs = []

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP FROM "Product\_Info"')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

All\_PIPs.append(row[0])

rowNr = rowNr + 1

try:

Search\_By\_PIP = int(Search\_Bar\_Entry.get())

except ValueError:

messagebox.showerror("PIP Does Not Exist!", "This Pharmacy does not currently have a product with that PIP Code! Try Again!")

Search\_Bar\_Entry.set("")

if Search\_By\_PIP not in All\_PIPs:

messagebox.showerror("PIP Does Not Exist!", "This Pharmacy does not currently have a product with that PIP Code! Try Again!")

Search\_Bar\_Entry.set("")

else:

Frames\_Labels\_and\_Lists()

Similar\_Labels\_and\_Lists()

Clear\_Search\_Lists()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('''SELECT PIP, Brand\_Name, Product\_Discription, Use, Supply\_Name, Quantity FROM "Product\_Info", "Brands", "Uses", "Supplier"

WHERE PIP = (:Searching\_PIP) AND Product\_Info.BrandID = Brands.BrandID AND Product\_Info.UseID = Uses.UseID AND Product\_Info.SupplyID = Supplier.SupplyID''',

{

"Searching\_PIP" : Search\_By\_PIP

})

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Searched\_PIP.append(row[0])

Searched\_Brand.append(row[1])

Searched\_Product\_Discription.append(row[2])

Searched\_Use.append(row[3])

Searched\_Supplier.append(row[4])

Searched\_Quantity.append(row[5])

rowNr = rowNr + 1

self.Searched\_PIP\_List.insert(0,\*Searched\_PIP)

self.Searched\_Brand\_List.insert(0,\*Searched\_Brand)

self.Searched\_Product\_Discription\_List.insert(0,\*Searched\_Product\_Discription)

self.Searched\_Use\_List.insert(0,\*Searched\_Use)

self.Searched\_Supplier\_List.insert(0,\*Searched\_Supplier)

self.Searched\_Quantity\_List.insert(0,\*Searched\_Quantity)

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('''SELECT PIP, Brand\_Name, Product\_Discription, Use, Supply\_Name, Quantity FROM "Product\_Info", "Brands", "Uses", "Supplier"

WHERE Use = (:Similar\_Use) AND Product\_Info.BrandID = Brands.BrandID AND Product\_Info.UseID = Uses.UseID AND Product\_Info.SupplyID = Supplier.SupplyID''',

{

"Similar\_Use" : Searched\_Use[0]

})

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Similar\_PIP.append(row[0])

Similar\_Brand.append(row[1])

Similar\_Product\_Discription.append(row[2])

Similar\_Use.append(row[3])

Similar\_Supplier.append(row[4])

Similar\_Quantity.append(row[5])

rowNr = rowNr + 1

self.Similar\_PIP\_List.insert(0,\*Similar\_PIP)

self.Similar\_Brand\_List.insert(0,\*Similar\_Brand)

self.Similar\_Product\_Discription\_List.insert(0,\*Similar\_Product\_Discription)

self.Similar\_Use\_List.insert(0,\*Similar\_Use)

self.Similar\_Supplier\_List.insert(0,\*Similar\_Supplier)

self.Similar\_Quantity\_List.insert(0,\*Similar\_Quantity)

elif Search\_By\_Menu.get() == "Brand":

All\_Brands = []

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Brand\_Name FROM "Brands"')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

All\_Brands.append(row[0])

rowNr = rowNr + 1

All\_Brands\_lower = [x.lower() for x in All\_Brands]

Search\_By\_Brand = Search\_Bar\_Entry.get()

if Search\_By\_Brand.lower() not in All\_Brands\_lower:

messagebox.showerror("Brand Does not Exist!", "This Pharmacy does not current have any Products of the selected Brand! Try Again!")

Search\_Bar\_Entry.set("")

else:

New\_Search\_By\_Brand = All\_Brands[All\_Brands\_lower.index(Search\_By\_Brand.lower())]

Frames\_Labels\_and\_Lists()

Clear\_Search\_Lists()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('''SELECT PIP, Brand\_Name, Product\_Discription, Use, Supply\_Name, Quantity FROM "Product\_Info", "Brands", "Uses", "Supplier"

WHERE Brand\_Name = (:Searching\_Brand) AND Product\_Info.BrandID = Brands.BrandID AND Product\_Info.UseID = Uses.UseID AND Product\_Info.SupplyID = Supplier.SupplyID''',

{

"Searching\_Brand" : New\_Search\_By\_Brand

})

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Searched\_PIP.append(row[0])

Searched\_Brand.append(row[1])

Searched\_Product\_Discription.append(row[2])

Searched\_Use.append(row[3])

Searched\_Supplier.append(row[4])

Searched\_Quantity.append(row[5])

rowNr = rowNr + 1

self.Searched\_PIP\_List.insert(0,\*Searched\_PIP)

self.Searched\_Brand\_List.insert(0,\*Searched\_Brand)

self.Searched\_Product\_Discription\_List.insert(0,\*Searched\_Product\_Discription)

self.Searched\_Use\_List.insert(0,\*Searched\_Use)

self.Searched\_Supplier\_List.insert(0,\*Searched\_Supplier)

self.Searched\_Quantity\_List.insert(0,\*Searched\_Quantity)

#------------------------------------------- Order History Tab Functions ----------------------------------------------#

def Clear\_Order\_History\_List(self):

self.lbOrderID.delete(0,END)

self.lbOrder\_PIP.delete(0,END)

self.lbOrder\_Quantity.delete(0,END)

self.lbTotal\_Cost.delete(0,END)

self.lbDate.delete(0,END)

def Display\_Newest(self):

OrderID.clear()

Order\_PIP.clear()

Order\_Quantity.clear()

Total\_Cost.clear()

Dates.clear()

self.Clear\_Order\_History\_List()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT OrderID, PIP, Order\_Quantity, Total\_Cost, Date\_Placed FROM "Place\_Order" ORDER BY Date\_Placed desc')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

OrderID.append(row[0])

Order\_PIP.append(row[1])

Order\_Quantity.append(row[2])

Total\_Cost.append(row[3])

Dates.append(row[4])

rowNr = rowNr + 1

self.lbOrderID.insert("end", \*OrderID)

self.lbOrder\_PIP.insert("end", \*Order\_PIP)

self.lbOrder\_Quantity.insert("end", \*Order\_Quantity)

self.lbTotal\_Cost.insert("end", \*Total\_Cost)

self.lbDate.insert("end", \*Dates)

def Display\_Oldest(self):

OrderID.clear()

Order\_PIP.clear()

Order\_Quantity.clear()

Total\_Cost.clear()

Dates.clear()

self.Clear\_Order\_History\_List()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT OrderID, PIP, Order\_Quantity, Total\_Cost, Date\_Placed FROM "Place\_Order"')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

OrderID.append(row[0])

Order\_PIP.append(row[1])

Order\_Quantity.append(row[2])

Total\_Cost.append(row[3])

Dates.append(row[4])

rowNr = rowNr + 1

self.lbOrderID.insert("end", \*OrderID)

self.lbOrder\_PIP.insert("end", \*Order\_PIP)

self.lbOrder\_Quantity.insert("end", \*Order\_Quantity)

self.lbTotal\_Cost.insert("end", \*Total\_Cost)

self.lbDate.insert("end", \*Dates)

def Show\_Order\_Info(self):

Chosen\_Order\_PIP = []

Chosen\_Order\_Brand\_Name = []

Chosen\_Order\_Product\_Discription = []

Chosen\_Order\_Quantity = []

Chosen\_Order\_Total\_Cost = []

Chosen\_Order\_Date = []

Chosen\_Order = self.lbOrderID.get(ACTIVE)

try:

OrderID\_Chosen = self.lbOrderID.get(self.lbOrderID.curselection())

except Exception:

messagebox.showerror("No Order Chosen!", "Please select an Order and try again!")

else:

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP, Order\_Quantity, Total\_Cost, Date\_Placed FROM "Place\_Order" WHERE OrderID = (:OrderID\_Selected)',

{

"OrderID\_Selected" : Chosen\_Order

})

Order\_rows = cur.fetchall()

con.close()

rowNr = 0

for row in Order\_rows:

if len(row) < 1:

continue

if rowNr >= 0:

Chosen\_Order\_PIP.append(row[0])

Chosen\_Order\_Quantity.append(row[1])

Chosen\_Order\_Total\_Cost.append(row[2])

Chosen\_Order\_Date.append(row[3])

rowNr = rowNr + 1

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Product\_Discription, Brand\_Name FROM "Product\_Info", "Brands" WHERE PIP = (:Order\_PIP\_Selected) AND Product\_Info.BrandID = Brands.BrandID',

{

"Order\_PIP\_Selected" : Chosen\_Order\_PIP[0]

})

PIP\_row = cur.fetchall()

con.close()

rowNr = 0

for row in PIP\_row:

if len(row) < 1:

continue

if rowNr >= 0:

Chosen\_Order\_Product\_Discription.append(row[0])

Chosen\_Order\_Brand\_Name.append(row[1])

rowNr = rowNr + 1

Order\_Information = Toplevel()

Order\_Information.geometry("950x380")

Order\_Information.config(bg ='Cadet Blue')

Frame0 = Frame(Order\_Information, bg = "Cadet Blue")

Frame0.grid(row = 0, column = 0, pady = 5)

Frame1 = Frame(Order\_Information, bg = "Cadet Blue")

Frame1.grid(row = 1, column = 0, padx = 5, pady = 5)

Frame2 = Frame(Order\_Information, bg = "Cadet Blue")

Frame2.grid(row = 2, column = 0, pady = 5)

Frame3 = Frame(Order\_Information, bg = "Cadet Blue")

Frame3.grid(row = 3, column = 0, pady = 5)

Order\_Info\_Title = Label(Frame0, text = "Order Information", width = 30, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Order\_Info\_Title.grid(row = 0, column = 0)

Chosen\_OrderID\_Label = Label(Frame1, text = "OrderID", width = 25, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Chosen\_OrderID\_Label.grid(row = 0, column = 1)

Chosen\_OrderID\_Display = Listbox(Frame1, width = 25, height = 1, font =('arial',20,'bold'), bd=5,

bg = 'White', fg = 'Black')

Chosen\_OrderID\_Display.grid(row = 1, column = 1)

Chosen\_PIP\_Label = Label(Frame2, text = "PIP", width = 15, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Chosen\_PIP\_Label.grid(row = 0, column = 0)

Chosen\_PIP\_Display = Listbox(Frame2, width = 15, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Chosen\_PIP\_Display.grid(row = 1, column = 0)

Chosen\_Brand\_Name\_Label = Label(Frame2, text = "Brand Name", width = 15, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Chosen\_Brand\_Name\_Label.grid(row = 0, column = 1)

Chosen\_Brand\_Name\_Display = Listbox(Frame2, width = 15, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Chosen\_Brand\_Name\_Display.grid(row = 1, column = 1)

Chosen\_Product\_Discription\_Label = Label(Frame2, text = "Product Discription", width = 20, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Chosen\_Product\_Discription\_Label.grid(row = 0, column = 2)

Chosen\_Product\_Discription\_Display = Listbox(Frame2, width = 20, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Chosen\_Product\_Discription\_Display.grid(row = 1, column = 2)

Chosen\_Quantity\_Label = Label(Frame3, text = "Quantity", width = 25, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Chosen\_Quantity\_Label.grid(row = 0, column = 0)

Chosen\_Quantity\_Display = Listbox(Frame3, width = 25, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Chosen\_Quantity\_Display.grid(row = 1, column = 0)

Chosen\_Total\_Cost\_Label = Label(Frame3, text = "Total Cost", width = 25, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Chosen\_Total\_Cost\_Label.grid(row = 0, column = 1)

Chosen\_Total\_Cost\_Display = Listbox(Frame3, width = 25, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Chosen\_Total\_Cost\_Display.grid(row = 1, column = 1)

Chosen\_Date\_Label = Label(Frame3, text = "Date and Time", width = 25, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Chosen\_Date\_Label.grid(row = 0, column = 0)

Chosen\_Date\_Display = Listbox(Frame3, width = 25, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Chosen\_Date\_Display.grid(row = 1, column = 0)

Chosen\_OrderID\_Display.insert(0,Chosen\_Order)

Chosen\_PIP\_Display.insert(0,Chosen\_Order\_PIP[0])

Chosen\_Brand\_Name\_Display.insert(0,Chosen\_Order\_Brand\_Name[0])

Chosen\_Product\_Discription\_Display.insert(0,Chosen\_Order\_Product\_Discription[0])

Chosen\_Quantity\_Display.insert(0,Chosen\_Order\_Quantity[0])

Chosen\_Total\_Cost\_Display.insert(0,Chosen\_Order\_Total\_Cost[0])

Chosen\_Date\_Display.insert(0,Chosen\_Order\_Date[0])

def Delete\_Order(self):

Order\_Brand\_Name\_Chosen = []

Order\_Product\_Discription\_Chosen = []

Order\_Quantity\_Chosen = []

Order\_Total\_Cost\_Chosen = []

Order\_Supplier\_Chosen = []

Order\_Date\_and\_Time\_Chosen = []

try:

OrderID\_Chosen = self.lbOrderID.get(self.lbOrderID.curselection())

except Exception:

messagebox.showerror("No Order Chosen", "Please select a OrderID and try again!")

else:

Order\_Brand\_Name\_Chosen.clear()

Order\_Product\_Discription\_Chosen.clear()

Order\_Quantity\_Chosen.clear()

Order\_Total\_Cost\_Chosen.clear()

Order\_Supplier\_Chosen.clear()

Order\_Date\_and\_Time\_Chosen.clear()

Delete\_Order\_Win = Toplevel()

Delete\_Order\_Win.geometry("1145x500")

Delete\_Order\_Win.config(bg ='Cadet Blue')

Frame0 = Frame(Delete\_Order\_Win, bg = "Cadet Blue")

Frame0.grid(row = 0, column = 0, pady = 5)

Frame1 = Frame(Delete\_Order\_Win, bg = "Cadet Blue")

Frame1.grid(row = 1, column = 0, padx = 5, pady = 5)

Frame2 = Frame(Delete\_Order\_Win, bg = "Cadet Blue")

Frame2.grid(row = 2, column = 0, pady = 5)

Frame3 = Frame(Delete\_Order\_Win, bg = "Cadet Blue")

Frame3.grid(row = 3, column = 0, pady = 5)

Title = Label(Frame0, text = "Cancel Order", width = 30, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Title.grid(row = 0, column = 0)

Delete\_OrderID\_Display\_Label = Label(Frame1, text = "Order ID", width = 10, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Delete\_OrderID\_Display\_Label.grid(row = 0, column = 0)

Delete\_Brand\_Display\_Label = Label(Frame1, text = "Brand", width = 10, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Delete\_Brand\_Display\_Label.grid(row = 0, column = 1)

Delete\_Product\_Display\_Label = Label(Frame1, text = "Product", width = 20, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Delete\_Product\_Display\_Label.grid(row = 0, column = 2)

Delete\_Quantity\_Display\_Label = Label(Frame1, text = "Quantity", width = 10, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Delete\_Quantity\_Display\_Label.grid(row = 0, column = 3)

Delete\_Total\_Cost\_Display\_Label = Label(Frame2, text = "Total Cost", width = 10, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Delete\_Total\_Cost\_Display\_Label.grid(row = 0, column = 0)

Delete\_Supplier\_Display\_Label = Label(Frame2, text = "Supplier", width = 15, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Delete\_Supplier\_Display\_Label.grid(row = 0, column = 1)

Delete\_Date\_Display\_Label= Label(Frame2, text = "Date Place", width = 20, font =('arial',25,'bold'), bd=15, relief='ridge',

bg = 'SeaGreen3', fg = 'Cornsilk')

Delete\_Date\_Display\_Label.grid(row = 0, column = 2)

def Selected\_Product():

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('''SELECT Brand\_Name, Product\_Discription, Order\_Quantity, Total\_Cost, Supply\_Name, Date\_Placed

FROM "Place\_Order","Product\_Info","Brands","Supplier"

WHERE (OrderID = (:Order\_Selected)) AND Place\_Order.PIP = Product\_Info.PIP AND Product\_Info.BrandID = Brands.BrandID AND Product\_Info.SupplyID = Supplier.SupplyID''',

{

'Order\_Selected' : OrderID\_Chosen

})

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Order\_Brand\_Name\_Chosen.append(row[0])

Order\_Product\_Discription\_Chosen.append(row[1])

Order\_Quantity\_Chosen.append(row[2])

Order\_Total\_Cost\_Chosen.append(row[3])

Order\_Supplier\_Chosen.append(row[4])

Order\_Date\_and\_Time\_Chosen.append(row[5])

rowNr = rowNr + 1

Delete\_OrderID\_Display.insert(0,OrderID\_Chosen)

Delete\_Brand\_Display.insert(0,\*Order\_Brand\_Name\_Chosen)

Delete\_Product\_Display.insert(0,\*Order\_Product\_Discription\_Chosen)

Delete\_Quantity\_Display.insert(0,\*Order\_Quantity\_Chosen)

Delete\_Cost\_Display.insert(0,\*Order\_Total\_Cost\_Chosen)

Delete\_Supplier\_Display.insert(0,\*Order\_Supplier\_Chosen)

Delete\_Date\_Display.insert(0,\*Order\_Date\_and\_Time\_Chosen)

Delete\_OrderID\_Display = Listbox(Frame1, width = 10, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Delete\_OrderID\_Display.grid(row = 1, column = 0)

Delete\_Brand\_Display = Listbox(Frame1, width = 10, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Delete\_Brand\_Display.grid(row = 1, column = 1)

Delete\_Product\_Display = Listbox(Frame1, width = 20, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Delete\_Product\_Display.grid(row = 1, column = 2)

Delete\_Quantity\_Display = Listbox(Frame1, width = 5, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Delete\_Quantity\_Display.grid(row = 1, column = 3)

Delete\_Cost\_Display = Listbox(Frame2, width = 10, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Delete\_Cost\_Display.grid(row = 1, column = 0)

Delete\_Supplier\_Display = Listbox(Frame2, width = 15, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Delete\_Supplier\_Display.grid(row = 1, column = 1)

Delete\_Date\_Display = Listbox(Frame2, width = 20, height = 1, font =('arial',20,'bold'), bd=5, relief='ridge',

bg = 'White', fg = 'Black')

Delete\_Date\_Display.grid(row = 1, column = 2)

Selected\_Product()

def Delete():

Cancel\_Order\_PIP = []

Cancel\_Order\_Supplier\_Name = []

Cancel\_Order\_Supplier\_Email = []

self.confirm = tkinter.messagebox.askyesno("Confirm Deletion of Data!", "Are you sure you want to cancel this order")

if self.confirm > 0:

Cancel\_Order\_PIP.clear()

Cancel\_Order\_Supplier\_Name.clear()

Cancel\_Order\_Supplier\_Email.clear()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT PIP FROM "Place\_Order" WHERE (OrderID = (:OrderID\_PIP))',

{

"OrderID\_PIP" : OrderID\_Chosen

})

Order\_PIP\_Fetched = cur.fetchone()

Cancel\_Order\_PIP.append(Order\_PIP\_Fetched[0])

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Supply\_Name, Email FROM "Place\_Order","Product\_Info","Supplier" WHERE (OrderID = (:OrderID\_Email)) AND (Place\_Order.PIP = Product\_Info.PIP) AND (Product\_Info.SupplyID = Supplier.SupplyID)',

{

"OrderID\_Email" : OrderID\_Chosen

})

OrderID\_rows = cur.fetchall()

rowNr = 0

for row in OrderID\_rows:

if len(row) < 1:

continue

if rowNr >= 0:

Cancel\_Order\_Supplier\_Name.append(row[0])

Cancel\_Order\_Supplier\_Email.append(row[1])

rowNr = rowNr + 1

Cancel\_Order\_Date = Order\_Date\_and\_Time\_Chosen[0]

Cancel\_OrderID = (OrderID\_Chosen)

port = 465

smtp\_server = "smtp.gmail.com"

sender\_email = os.environ.get('Pharmacy\_Email')

receiver\_email = str(Cancel\_Order\_Supplier\_Email[0])

password = os.environ.get('Pharmacy\_Password')

Email\_Message =("""From: Sam

To: %s

Subject: %s

To Whom It May Concern,

We would like to CANCEL an order placed on %s, with OrderID - %s.

The order was to purchase %s packages of %s, %s. With PIP Code - %s.

Our details are as follows

Pharmacy Name - Supercare Pharmacy

Phone Number - xxxxxxxxxxx

Fax - xxxxxxxxxxxx

Address - xxxxx, xxx xxxxxx, xxx xxxx

Many Thanks

Supercare Pharmacy

Sent via Python!

"""%(receiver\_email,Cancel\_Order\_PIP[0],Cancel\_Order\_Date,Cancel\_OrderID,int(Order\_Quantity\_Chosen[0]),Order\_Brand\_Name\_Chosen[0],Order\_Product\_Discription\_Chosen[0],Cancel\_Order\_PIP[0]))

try:

context = ssl.create\_default\_context()

with smtplib.SMTP\_SSL(smtp\_server, port, context=context) as server:

server.login(sender\_email, password)

server.sendmail(sender\_email, receiver\_email, Email\_Message)

messagebox.showinfo("Email","Successfully sent cancelation email to %s, %s"%(Cancel\_Order\_Supplier\_Email[0],Cancel\_Order\_Supplier\_Name[0]))

except Exception:

messagebox.showerror("Email Error!","Error: unable to send cancelation email")

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('DELETE FROM "Place\_Order" WHERE (OrderID = (:OrdeID\_To\_Delete))',

{

'OrdeID\_To\_Delete' : OrderID\_Chosen

})

con.commit()

con.close()

Delete\_Order\_Win.destroy()

Order\_Brand\_Name\_Chosen.clear()

Order\_Product\_Discription\_Chosen.clear()

Order\_Quantity\_Chosen.clear()

Order\_Total\_Cost\_Chosen.clear()

Order\_Supplier\_Chosen.clear()

Order\_Date\_and\_Time\_Chosen.clear()

self.Clear\_Order\_History\_List()

OrderID.clear()

Order\_PIP.clear()

Order\_Quantity.clear()

Total\_Cost.clear()

Dates.clear()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT OrderID, PIP, Order\_Quantity, Total\_Cost, Date\_Placed FROM "Place\_Order" ORDER BY Date\_Placed desc')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

OrderID.append(row[0])

Order\_PIP.append(row[1])

Order\_Quantity.append(row[2])

Total\_Cost.append(row[3])

Dates.append(row[4])

rowNr = rowNr + 1

self.lbOrderID.insert("end", \*OrderID)

self.lbOrder\_PIP.insert("end", \*Order\_PIP)

self.lbOrder\_Quantity.insert("end", \*Order\_Quantity)

self.lbTotal\_Cost.insert("end", \*Total\_Cost)

self.lbDate.insert("end", \*Dates)

else:

Delete\_Order\_Win.destroy()

Delete\_Button = Button(Frame3,text='Delete Product', width=15,font=('arial',15,'bold'), bd = 10, relief='ridge',

bg='SeaGreen3', fg='Cornsilk', command = Delete)

Delete\_Button.grid(row = 0, column = 0)

#-------------------------------------------- Suppliers Tab Functions -------------------------------------------------#

def Supplier\_Details(self):

Supplier\_Name\_Info = []

Supplier\_Phone = []

Supplier\_Email = []

Supplier\_Address = []

try:

Supplier\_chosen = self.Suppliers\_Listbox.get(self.Suppliers\_Listbox.curselection())

except Exception:

messagebox.showerror("No Supplier Chosen!", "Please select a Supplier and try again!")

else:

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Supply\_Name, Phone, Email, Address FROM "Supplier" WHERE (Supply\_Name = (:Supplier\_Selected))',

{

'Supplier\_Selected' : Supplier\_chosen

})

Supplier\_Info\_Rows = cur.fetchall()

con.close()

rowNr = 0

for row in Supplier\_Info\_Rows:

if len(row) < 1:

continue

if rowNr >= 0:

Supplier\_Name\_Info.append(row[0])

Supplier\_Phone.append(row[1])

Supplier\_Email.append(row[2])

Supplier\_Address.append(row[3])

rowNr = rowNr + 1

Chosen\_Suppliers\_Name\_Display.insert(0,Supplier\_Name\_Info[0])

Chosen\_Suppliers\_Phone\_Display.insert(0,Supplier\_Phone[0])

Choesen\_Suppliers\_Email\_Display.insert(0,Supplier\_Email[0])

Chosen\_Suppliers\_Address\_Display.insert(0,Supplier\_Address[0])

def Add\_New\_Supplier(self):

Continue = True

# or (not any(l in Email\_Domain for l in (Add\_Suppliers\_Email.get())))

Email\_Domain = ('.com','.ac.uk','.co.uk','.gov','.edu','.net','.org','.biz','.gov')

Email\_Domain\_re = re.compile("|".join(Email\_Domain))

if (len(Add\_Suppliers\_Name.get()) < 1):

messagebox.showerror("Information Error!","Please enter a Supplier Name! Try Again!")

Continue = False

Supplier\_Name\_Entered.set("")

try:

Phone\_Number = int(Add\_Suppliers\_Phone.get())

except ValueError:

messagebox.showerror("Invalid Data Entered!", "Please enter a valid Phone Number and try again!")

Continue = False

Phone\_Entered.set("")

if (len(Add\_Suppliers\_Email.get()) < 1) or ("@" not in Add\_Suppliers\_Email.get()) or (not Email\_Domain\_re.search(Add\_Suppliers\_Email.get())):

messagebox.showerror("Information Error!","Please enter a valid email address! Try Again!")

Continue = False

Email\_Entered.set("")

if (len(Add\_Suppliers\_Address.get()) < 1):

messagebox.showerror("Information Error!","Please select a Use before submiting data! Try Again!")

Continue = False

Address\_Entered.set("")

if Continue is True:

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('INSERT INTO "Supplier"(Supply\_Name,Phone,Email,Address) VALUES (:Supplier\_Name\_Value, :Phone\_Value, :Email\_Value, :Address\_Value)',

{

'Supplier\_Name\_Value' : Add\_Suppliers\_Name.get(),

'Phone\_Value' : Phone\_Number,

'Email\_Value' : Add\_Suppliers\_Email.get(),

'Address\_Value' : Add\_Suppliers\_Address.get()

})

con.commit()

con.close()

Supplier\_Name\_Entered.set("")

Phone\_Entered.set("")

Email\_Entered.set("")

Address\_Entered.set("")

Suppliers\_Names.clear()

self.Suppliers\_Listbox.delete(0, END)

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Supply\_Name FROM "Supplier"')

rows = cur.fetchall()

con.close()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Suppliers\_Names.append(row[0])

rowNr = rowNr + 1

self.Suppliers\_Listbox.insert("end", \*Suppliers\_Names)

#---------------------------------------------- Scrollbar Functions --------------------------------------------------#

def yview(self, \*args):

self.lbPIP.yview(\*args)

self.lbBrand\_Name.yview(\*args)

self.lbProduct\_Discription.yview(\*args)

self.lbQuantity.yview(\*args)

self.Searched\_PIP\_List.yview(\*args)

self.Searched\_Brand\_List.yview(\*args)

self.Searched\_Product\_Discription\_List.yview(\*args)

self.Searched\_Use\_List.yview(\*args)

self.Searched\_Supplier\_List.yview(\*args)

self.Searched\_Quantity\_List.yview(\*args)

try:

self.Similar\_PIP\_List.yview(\*args)

self.Similar\_Brand\_List.yview(\*args)

self.Similar\_Product\_Discription\_List.yview(\*args)

self.Similar\_Use\_List.yview(\*args)

self.Similar\_Supplier\_List.yview(\*args)

self.Similar\_Quantity\_List.yview(\*args)

except AttributeError:

return

self.lbOrderID.yview(\*args)

self.lbOrder\_PIP.yview(\*args)

self.lbOrder\_Quantity.yview(\*args)

self.lbTotal\_Cost.yview(\*args)

self.lbDate.yview(\*args)

def xview(self, \*args):

self.lbPIP.xview(\*args)

self.lbBrand\_Name.xview(\*args)

self.lbProduct\_Discription.xview(\*args)

self.lbQuantity.xview(\*args)

self.lbOrderID.xview(\*args)

self.lbOrder\_PIP.xview(\*args)

self.lbOrder\_Quantity.xview(\*args)

self.lbTotal\_Cost.xview(\*args)

self.lbDate.xview(\*args)

#-----------------------------------------------New Product WIndow-----------------------------------------------------#

#----------------------------------------------------------------------------------------------------------------------#

class New\_Product\_Window:

def \_\_init\_\_(self, master):

self.master = master

self.master.title(" Add New Product Page")

self.master.geometry("1050x650+0+0")

self.master.config(bg ='Cadet Blue')

self.frame = Frame(self.master, bg='Cadet Blue')

self.frame.pack()

PIP\_Entry = StringVar()

Product\_Discription\_Entry = StringVar()

Pack\_Size\_Entry = StringVar()

Product\_Cost\_Entry= StringVar()

Quantity\_Entry = StringVar()

#-----------------------------------------------------Frames-------------------------------------------------------------#

self.frame1 = Frame(self.frame, width=700,height=50 , relief='ridge',bg='MediumPurple2', bd=40)

self.frame1.grid(row=0, column=0)

self.frame2 = LabelFrame(self.frame, width=710,height=125

,font=('arial',20,'bold'), relief='ridge',bg='MediumPurple2', bd=40)

self.frame2.grid(row=1, column=0)

self.frame3 = LabelFrame(self.frame, width=710,height=125

,font=('arial',20,'bold'), relief='ridge',bg='MediumPurple2', bd=40)

self.frame3.grid(row=2, column=0)

self.frame4 = LabelFrame(self.frame, width=710,height=125

,font=('arial',20,'bold'), relief='ridge',bg='MediumPurple2', bd=40)

self.frame4.grid(row=3, column=0)

#------------------------------------------------------Label-------------------------------------------------------------#

Title\_label = Label(self.frame1, text = "Add New Product", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 20, bg = 'MediumPurple2', fg = 'White')

Title\_label.grid(row = 0, column = 0, pady = 10)

PIP\_label = Label(self.frame2, text = "PIP", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'MediumPurple2', fg = 'White')

PIP\_label.grid(row = 0, column = 0)

BrandID\_label = Label(self.frame2, text = "Brand Name", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 15, bg = 'MediumPurple2', fg = 'White')

BrandID\_label.grid(row = 0, column = 1)

Product\_Discription\_label = Label(self.frame2, text = "Product Discription", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 25, bg = 'MediumPurple2', fg = 'White')

Product\_Discription\_label.grid(row = 0, column = 2)

UseID\_label = Label(self.frame2, text = "Use", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'MediumPurple2', fg = 'White')

UseID\_label.grid(row = 0, column = 3)

Pack\_Size\_label = Label(self.frame3, text = "Pack Size", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'MediumPurple2', fg = 'White')

Pack\_Size\_label.grid(row = 0, column = 0)

Product\_Cost\_label = Label(self.frame3, text = "Product Cost", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 15, bg = 'MediumPurple2', fg = 'White')

Product\_Cost\_label.grid(row = 0, column = 1)

SupplyID\_label = Label(self.frame3, text = "Supplier", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'MediumPurple2', fg = 'White')

SupplyID\_label.grid(row = 0, column = 2)

Quantity\_label = Label(self.frame3, text = "Quantity", font=('arial',15,'bold'), bd = 10, relief='ridge',

width = 10, bg = 'MediumPurple2', fg = 'White')

Quantity\_label.grid(row = 0, column = 3)

#---------------------------------------------------------------Lists---------------------------------------------------------#

Brands = []

Uses = []

Suppliers = []

#------------------------------------------------------------Entry Boxes------------------------------------------------------#

PIP\_Entrybox = Entry(self.frame2, font=('arial',20,'bold'),bd=7,textvariable=PIP\_Entry, width = 10)

PIP\_Entrybox.grid(row=2,column=0)

def List\_Brands():

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Brand\_Name FROM "Brands"')

rows = cur.fetchall()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Brands.append(row[0])

rowNr = rowNr + 1

def List\_Uses():

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Use FROM "Uses"')

rows = cur.fetchall()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Uses.append(row[0])

rowNr = rowNr + 1

def List\_Suppliers():

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT Supply\_Name FROM "Supplier"')

rows = cur.fetchall()

rowNr = 0

for row in rows:

if len(row) < 1:

continue

if rowNr >= 0:

Suppliers.append(row[0])

rowNr = rowNr + 1

List\_Brands()

List\_Uses()

List\_Suppliers()

Brand\_Menu = StringVar()

Brand\_Menu.set("Pick a Brand")

Brand\_Name\_DropDownMenu = OptionMenu(self.frame2, Brand\_Menu, \*Brands)

Brand\_Name\_DropDownMenu.grid(row=2,column=1)

Product\_Discription\_Entrybox = Entry(self.frame2, font=('arial',20,'bold'),bd=7,textvariable = Product\_Discription\_Entry, width = 20)

Product\_Discription\_Entrybox.grid(row=2,column=2)

Use\_Menu = StringVar()

Use\_Menu.set("Pick a Use")

Use\_DropDownMenu = OptionMenu(self.frame2, Use\_Menu, \*Uses)

Use\_DropDownMenu.grid(row=2,column=3)

Pack\_Size\_Entrybox = Entry(self.frame3, font=('arial',20,'bold'),bd=7,textvariable = Pack\_Size\_Entry, width = 5)

Pack\_Size\_Entrybox.grid(row=2,column=0)

Product\_Cost\_Entrybox = Entry(self.frame3, font=('arial',20,'bold'),bd=7,textvariable = Product\_Cost\_Entry, width = 5)

Product\_Cost\_Entrybox.grid(row=2,column=1)

Supply\_Menu = StringVar()

Supply\_Menu.set("Pick a Supplier")

Supply\_Name\_DropDownMenu = OptionMenu(self.frame3, Supply\_Menu, \*Suppliers)

Supply\_Name\_DropDownMenu.grid(row=2,column=2)

Quantity\_Entrybox = Entry(self.frame3, font=('arial',20,'bold'),bd=7,textvariable = Quantity\_Entry, width = 5)

Quantity\_Entrybox.grid(row=2,column=3)

def Reset():

PIP\_Entry.set("")

Brand\_Menu.set("Pick a Brand")

Product\_Discription\_Entry.set("")

Use\_Menu.set("Pick a Use")

Pack\_Size\_Entry.set("")

Product\_Cost\_Entry.set("")

Supply\_Menu.set("Pick a Supplier")

Quantity\_Entry.set("")

def add\_new\_product():

Continue = True

def BrandID\_from\_Brand\_Name():

global Fetched\_BrandID

Check\_Brand\_Name = Brand\_Menu.get()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT BrandID FROM "Brands" WHERE Brand\_Name = (:Brand\_Name\_Dic)',

{

'Brand\_Name\_Dic' : Check\_Brand\_Name

})

Fetched\_BrandID = cur.fetchone()

def UseID\_from\_Uses():

global Fetched\_UseID

Check\_Uses = Use\_Menu.get()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT UseID FROM "Uses" WHERE Use = (:Uses\_Dic)',

{

'Uses\_Dic' : Check\_Uses

})

Fetched\_UseID = cur.fetchone()

def SupplyID\_from\_Supply\_Name():

global Fetched\_SupplyID

Check\_Supply\_Name = Supply\_Menu.get()

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('SELECT SupplyID FROM "Supplier" WHERE Supply\_Name = (:Supply\_Name\_Dic)',

{

'Supply\_Name\_Dic': Check\_Supply\_Name

})

Fetched\_SupplyID = cur.fetchone()

BrandID\_from\_Brand\_Name()

UseID\_from\_Uses()

SupplyID\_from\_Supply\_Name()

try:

PIP\_Entered = int(PIP\_Entry.get())

except ValueError:

messagebox.showerror("Invalid Data Entered!", "Please enter an integer value for the PIP number! Try Again!")

PIP\_Entry.set("")

Continue = False

try:

BrandID = Fetched\_BrandID[0]

except TypeError:

messagebox.showerror("Brand Name Error!", "Please select a Brand and try again!")

Continue = False

if (len(Brand\_Menu.get()) < 1):

messagebox.showerror("Information Error!","Please Select a Brand before submitting data! Try Again!")

Continue = False

Brand\_Menu.set("Pick a Brand")

if (len(Product\_Discription\_Entry.get()) < 1):

messagebox.showerror("Information Error!","Please fill in a valid discription before submitting data! Try Again!")

Continue = False

Product\_Discription\_Entry.set("")

try:

UseID = Fetched\_UseID[0]

except TypeError:

messagebox.showerror("Use Error!", "Please select a Use and try again!")

Continue = False

if (len(Use\_Menu.get()) < 1):

messagebox.showerror("Information Error!","Please select a Use before submitting data! Try Again!")

Continue = False

Use\_Menu.set("Pick a Use")

try:

Pack\_Size\_Entered = int(Pack\_Size\_Entry.get())

except ValueError:

messagebox.showerror("Invalid Data Entered!", "Please enter an integer value for the Pack Size! Try Again!")

Continue = False

Pack\_Size\_Entry.set("")

try:

Product\_Cost\_Entered = (Product\_Cost\_Entry.get())

integral, fractional = Product\_Cost\_Entered.split('.')

Product\_Cost\_Entered = float(Product\_Cost\_Entered)

if len(fractional) == 2 and Product\_Cost\_Entered > 0:

pass

else:

messagebox.showerror("Invalid Data Entered!", "Please enter a float value for the Product Cost rounded to two decimal places! Try Again!")

Product\_Cost\_Entry.set("")

except ValueError:

messagebox.showerror("Invalid Data Entered!", "Please enter a float value for the Product Cost rounded to two decimal places! Try Again!")

Continue = False

Product\_Cost\_Entry.set("")

try:

SupplyID = Fetched\_SupplyID[0]

except TypeError:

messagebox.showerror("Supplier Error!", "Please select a Supplier and try again!")

Continue = False

if (len(Supply\_Menu.get()) < 1):

messagebox.showerror("Information Error!","Please select a Supplier before submitting data! Try Again!")

Continue = False

Supply\_Menu.set("Pick a Supplier")

try:

Quantity\_Entered = int(Quantity\_Entry.get())

except ValueError:

messagebox.showerror("Invalid Data Entered!", "Please enter an integer value for the Quantity! Try Again!")

Continue = False

Quantity\_Entry.set("")

if Continue is True:

con = sqlite3.connect("Stock\_Management\_Database.db")

cur = con.cursor()

cur.execute('INSERT INTO "Product\_Info" VALUES (:PIP\_Value, :BrandID\_Value, :Product\_Discription\_Value, :UseID\_Value, :Pack\_Size\_Value, :Product\_Cost\_Value, :SupplyID\_Value, :Quantity\_Value)',

{

'PIP\_Value' : PIP\_Entry.get(),

'BrandID\_Value' : BrandID,

'Product\_Discription\_Value' : Product\_Discription\_Entry.get(),

'UseID\_Value' : UseID,

'Pack\_Size\_Value' : Pack\_Size\_Entry.get(),

'Product\_Cost\_Value' : Product\_Cost\_Entry.get(),

'SupplyID\_Value' : SupplyID,

'Quantity\_Value' : Quantity\_Entry.get()

})

con.commit()

con.close()

messagebox.showinfo("Entry Sucessful!", "The new product has been sucessfully entered!")

Add\_More = tkinter.messagebox.askyesno("Add More Data?", "Would you like to add another new product?")

if Add\_More > 0:

Reset()

else:

self.master.destroy()

Add\_New\_Data\_Button = Button(self.frame4, text = "Add New Product", width=15,font=('arial',20,'bold'), bd =10, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = add\_new\_product)

Add\_New\_Data\_Button.grid (row = 0, column = 0)

Reset\_Button = Button(self.frame4, text = "Reset", width=10,font=('arial',20,'bold'), bd =10, relief='ridge',

bg='SlateBlue2', fg='Cornsilk', command = Reset)

Reset\_Button.grid (row = 0, column = 1)

#--------------------------------------------------------------Fubctions---------------------------------------------------#

#-------------------------------------------------------------Admin Window-------------------------------------------------#

#------------------------------------------------------------------=-------------------------------------------------------#

class Admin:

def \_\_init\_\_(self, master):

self.master = master

self.master.title("Admin Page")

self.master.geometry("1350x750+0+0")

self.master.config(bg ='Cadet Blue')

self.frame = Frame(self.master, bg='Cadet Blue')

self.frame.pack()

self.New\_User = StringVar()

self.New\_Pass = StringVar()

self.Confirm\_Pass = StringVar()

self.lblTitle = Label(self.frame, text = 'Admin Page', font =('arial', 60, 'bold'),

bg='Cadet Blue', fg='Cornsilk')

self.lblTitle.grid(row=0, column=0, columnspan=2, pady=20)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Frames~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

self.AdminFrame1 = LabelFrame(self.frame, width=1350,height=500

,text="ADMIN EYES ONLY",font=('arial',20,'bold'), relief='ridge',bg='SlateBlue2', bd=40)

self.AdminFrame1.grid(row=1, column=0)

self.AdminFrame2 = LabelFrame(self.frame, width=1000,height=200

,font=('arial',20,'bold'), relief='ridge',bg='SlateBlue2', bd=40)

self.AdminFrame2.grid(row=2, column=0)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Inputs Entries UserName & Password~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

self.lblNew\_User = Label(self.AdminFrame1, text = 'Username', font =('arial',30,'bold'),bd=22,

bg = 'SlateBlue2', fg = 'Cornsilk')

self.lblNew\_User.grid(row=0,column=0)

self.txtNew\_User = Entry(self.AdminFrame1,font=('arial',30,'bold'),bd=7,textvariable=self.New\_User,

width = 30)

self.txtNew\_User.grid(row=0,column=1,padx=88)

self.lblNew\_Pass = Label(self.AdminFrame1,text='Password',font=('arial',30,'bold'),bd=22,

bg='SlateBlue2', fg = 'Cornsilk')

self.lblNew\_Pass.grid(row=1,column=0)

self.txtNew\_Pass = Entry(self.AdminFrame1, font =('arial',30,'bold'),bd=7,textvariable=self.New\_Pass,

width=30)

self.txtNew\_Pass.grid(row=1,column=1,columnspan=2,pady=30)

self.lblConfirm\_Pass = Label(self.AdminFrame1,text='Confirm Password',font=('arial',30,'bold'),bd=22,

bg='SlateBlue2', fg = 'Cornsilk')

self.lblConfirm\_Pass.grid(row=2,column=0)

self.txtConfirm\_Pass = Entry(self.AdminFrame1, font =('arial',30,'bold'),bd=7,textvariable=self.Confirm\_Pass,

width=30)

self.txtConfirm\_Pass.grid(row=2,column=1,columnspan=2,pady=30)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Buttons ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

self.btnUser\_Info = Button(self.AdminFrame2, text='Create New User', width=15,font=('arial',25,'bold'),

bg='SlateBlue2', fg='Cornsilk', command=self.Checks)

self.btnUser\_Info.grid(row=3,column=0,pady=20,padx=8)

self.btnReset = Button(self.AdminFrame2, text='Reset', width=10,font=('arial',25,'bold'),

bg='SlateBlue2', fg='Cornsilk', command=self.Reset\_Com)

self.btnReset.grid(row=3,column=1,pady=20,padx=8)

self.btnDisplay = Button(self.AdminFrame2, text='Display Users', width=15,font=('arial',25,'bold'),

bg='SlateBlue2', fg='Cornsilk', command=self.Display\_Users)

self.btnDisplay.grid(row=3,column=2,pady=20,padx=8)

self.btnExit = Button(self.AdminFrame2, text='Exit', width=10,font=('arial',25,'bold'),

bg='SlateBlue2', fg='Cornsilk', command=self.Exit\_Com)

self.btnExit.grid(row=3,column=3,pady=20,padx=8)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Subroutines ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

def Password\_Error\_Message(self):

self.Password\_Error\_Message = messagebox.showerror("Password Does Not Meet Criteria", """Please Make Sure your Password:

\n 1) Is at least 8 Characters long

\n 2) Has at least one number in it

\n 3) Has a Capital letter

\n 4) Has at least one Special Characters from: ($,#,@,!,\*,.,\_,&,!,£,^)

\n Would you like to try again? """)

return

def Checks(self):

Check\_User = (self.New\_User.get())

Check\_Pass = (self.New\_Pass.get())

Check\_Confirm = (self.Confirm\_Pass.get())

Special\_Characters = ['$', '#', '@', '!', '\*', ".", "\_", "&", "!", "£", "^"]

if Check\_User in User\_Names:

messagebox.showerror("Username Error!", "User Already Exists, Try again!")

self.New\_User.set("")

self.New\_Pass.set("")

self.Confirm\_Pass.set("")

else:

if Check\_Pass != Check\_Confirm:

messagebox.showerror("Password Error!", "Passwords Do Not Match! , Try again!")

self.New\_Pass.set("")

self.Confirm\_Pass.set("")

else:

while True:

if (len(Check\_Pass) < 8) or (re.search('[0-9]', Check\_Pass) is None) or (re.search('[A-Z]',Check\_Pass) is None) or (not any(c in Special\_Characters for c in Check\_Pass)):

messagebox.showerror("Password Does Not Meet Criteria", """Please make sure your password:

\n 1) Is at least 8 Characters long

\n 2) Has at least one number in it

\n 3) Has a Capital letter

\n 4) Has at least one Special Characters from: ($,#,@,!,\*,.,\_,&,!,£,^)

\n Please try again! """)

self.New\_Pass.set("")

self.Confirm\_Pass.set("")

return

else:

User\_Names.append(Check\_User)

Passes.append(Check\_Pass)

Added\_NewUser = [Check\_User, Check\_Pass]

append\_NewUser\_CSV('Users.csv', Added\_NewUser)

self.New\_User.set("")

self.New\_Pass.set("")

self.Confirm\_Pass.set("")

messagebox.showinfo("New User Added!","A New User Has Been Sucessfully Added!")

return

break

def Reset\_Com(self):

self.New\_User.set("")

self.New\_Pass.set("")

self.Confirm\_Pass.set("")

def Display\_Users(self):

self.master.destroy()

Tk().withdraw()

self.Display\_Users = Toplevel()

self.app = Display\_Window(self.Display\_Users)

def return\_to\_login(self):

self.master.destroy()

Tk().withdraw()

self.return\_to\_login = Toplevel()

self.app = Login(self.return\_to\_login)

def Exit\_Com(self):

self.Exit\_Com = tkinter.messagebox.askyesno("\*!Quit System!\*", "Confirm Exit")

if self.Exit\_Com > 0:

self.return\_to\_login()

return

else:

return

#------------------------------------------------Display Users Window---------------------------------------------#

#---------------------------------------------------------------=-------------------------------------------------#

class Display\_Window:

global User\_Names

global Passes

def \_\_init\_\_(self, master):

#Zipped\_Lists = zip(User\_Names,Passes)

#Users\_and\_Passwords = (list(Zipped\_Lists))

self.master = master

self.master.title("Display Users Page")

self.master.geometry("950x650+275+0")

self.master.config(bg ='Thistle2')

self.frame = Frame(self.master, bg='Thistle2')

self.frame.pack()

self.New\_User = StringVar()

self.New\_Pass = StringVar()

self.Confirm\_Pass = StringVar()

self.lblTitle = Label(self.frame, text = 'Display Users Page', font =('arial', 40, 'bold'),

bg='Thistle2', fg='Cadet Blue')

self.lblTitle.grid(row=0, column=0, columnspan=2, pady=5)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Frames~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

self.frame1 = LabelFrame(self.frame, width=1350,height=400

,text="UserNames and Passwords",font=('arial',20,'bold'), relief='ridge',bg='MediumPurple2', bd=40)

self.frame1.grid(row=1, column=0)

self.frame2 = LabelFrame(self.frame, width=500,height=150

,font=('arial',20,'bold'), relief='ridge',bg='MediumPurple2', bd=40)

self.frame2.grid(row=2, column=0)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Labels~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

self.scrollbar\_V = Scrollbar(self.frame1)

self.scrollbar\_H\_User = Scrollbar(self.frame1, orient=HORIZONTAL)

self.scrollbar\_H\_Password = Scrollbar(self.frame1, orient=HORIZONTAL)

self.scrollbar\_V.grid(row = 0, column = 4, sticky=N+S+W)

self.scrollbar\_H\_User.grid(row = 1, column = 1, sticky=N+E+S+W)

self.scrollbar\_H\_Password.grid(row = 1, column = 2, sticky=N+E+S+W)

def populateListbox(User,Pass):

self.lbUser.insert("end", \*User)

self.lbPassword.insert("end", \*Pass)

self.lbUser = Listbox(self.frame1, font=('arial',12,'bold'), width = 20, height = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_User.set)

self.lbUser.grid(row=0,column=1,pady=20,padx=8)

self.lbPassword = Listbox(self.frame1, font=('arial',12,'bold'), width = 20, height = 10, yscrollcommand=self.scrollbar\_V.set, xscrollcommand=self.scrollbar\_H\_Password.set)

self.lbPassword.grid(row=0,column=2,pady=20,padx=8)

self.scrollbar\_V.config(command=self.yview)

self.scrollbar\_H\_User.config(command=self.xview)

self.scrollbar\_H\_Password.config(command = self.lbPassword.xview)

populateListbox(User\_Names, Passes)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Buttons~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#

self.btnExit = Button(self.frame2, text='Exit', width=10,font=('arial',25,'bold'),

bg='MediumPurple2', fg='Cornsilk', command=self.Exit\_Com)

self.btnExit.grid(row=3,column=3,pady=20,padx=8)

#--------------------------------------------------- Functions -------------------------------------------------------------#

def yview(self, \*args):

self.lbUser.yview(\*args)

self.lbPassword.yview(\*args)

def xview(self, \*args):

self.lbUser.xview(\*args)

self.lbPassword.xview(\*args)

def return\_to\_admin(self):

self.master.destroy()

Tk().withdraw()

self.return\_to\_admin = Toplevel()

self.app = Admin(self.return\_to\_admin)

def Exit\_Com(self):

self.Exit\_Com = tkinter.messagebox.askyesno("\*!Quit System!\*", "Confirm Exit")

if self.Exit\_Com > 0:

self.return\_to\_admin()

return

else:

return

#----------------------------------------------------Main Loop------------------------------------------------------#

if \_\_name\_\_ == '\_\_main\_\_':

main()

# Testing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Number** | **Description** | **Data type** | **Tested** **Data** | **Expected result** | **Pass/Fail** | **Cross Reference** |
| 1 | I will be testing the function of logging into the Main Window via the Login Window | Typical | Username:  “Samir007”  Password:  “h20medicine%” | Logs into the main window of the stock management system | Pass | <https://youtu.be/N9rEDJ0rX84>  0:00 – 1:25 |
| 2 | I will be testing the function of logging into the Main Window via the Login Window | Erroneous | Username:  “Test1”  Password:  “Password£1” | Displays an error message  Saying invalid login details and resets the entry boxes. | Pass | <https://youtu.be/N9rEDJ0rX84>  1:25-2:36 |
| 3 | I will be testing the function of the “Reset” Button in the Login Window | Typical | Username:  “TestReset”  Password:  “TestReset1”,  “Reset” button Pressed | Resets the Username and Password entry boxes | Pass | <https://youtu.be/N9rEDJ0rX84>  2:36-3:23 |
| 4 | I will be testing the function of logging into the Admin Window | Typical | Username:  “ADMIN”  Password:  “100001” | Closes the Login Window and opens the Admin Window | Pass | <https://youtu.be/N9rEDJ0rX84>  3:23-4:28 |
| 5 | The creation of a new User must have a unique Username | Erroneous | Username:  “Bfawaz1” | Displays an error message is displayed explaining that the user already exists, and the entry box is reset | Pass | <https://youtu.be/LAtSAYlv3sw>  0:00-1:11 |
| 6 | The creation of a new User must have a unique Username | Typical | Username:  “Test001” | An error message is displayed telling the user to enter a valid password | Pass | <https://youtu.be/LAtSAYlv3sw>  1:11-2:18 |
| 7 | The creation of a new User must have a strong password that is valid | Erroneous | Password:  “Test01”  Confirm Password:  “Test01” | Error message is displayed telling the user that the password is invalid and what it should include to be valid. The password and confirm password entry boxes are reset. | Pass | <https://youtu.be/LAtSAYlv3sw>  2:18-3:13 |
| 8 | The creation of a new User must have the password and confirm password entered matching | Erroneous | Password:  “Test01!£”  Confirm Password:  “Test01!” | Error message displayed, telling the user that the passwords do not match. | Pass | <https://youtu.be/LAtSAYlv3sw>  3:13-4:02 |
| 9 | The creation of a new User must have a unique Username, Valid Passwords, and matching passwords | Typical | Password:  “Test01!£”  Confirm Password:  “Test01!£” | A message is displayed saying the User is successfully added. The new User’s Username and Password is added to the User CSV file in a new row. The entry boxes are reset. | Pass | <https://youtu.be/LAtSAYlv3sw>  4:02-5:20 |
| 10 | I will be testing the new User created Username and Password on the Login Window to login | Typical | Username:  “Test001”  Password:  “Test01!£” | The login window will close and the user will be logged into the main page/window of the stock management programme. | Pass | <https://youtu.be/LAtSAYlv3sw>  5:20-6:07 |
| 11 | I will be testing the function of the reset button on the entry boxes of the Admin Window | Typical | “Reset” button Pressed | The Username, Password, and Confirm Password entry boxes are reset | Pass | <https://youtu.be/LAtSAYlv3sw>  6:07-7:01 |
| 12 | When the “Display Users” button is pressed the current User’s information is displayed in a separate window | Typical | “Display Users” button Pressed | The Admin Window is closed, and a new Display Users Window is open with two list boxes populated with the current Usernames and Passwords in the system. | Pass | <https://youtu.be/LAtSAYlv3sw>  7:01-7:55 |
| 13 | I will be testing the Exit function of the Display Users, Admin and Login Windows | Typical | “Exit”  button is Pressed | The Display Users window is closed, and Admin window is open. Then the Admin window is closed, and the Login window is opened. Finally the Login window is closed | Pass | <https://youtu.be/LAtSAYlv3sw>  7:55-9:06 |
| 14 | I will be testing weather the “Order By” options, filter the stock lists to the correct descriptions of the buttons in the “Stock Management” Tab of the main page/window. As well as the function of the scrollbars on the lists | Typical | The “Brand Name Z to A”, “Brand Name A to Z”, ”Lowest Stock”, “Highest Stock”, and then “PIP” buttons are clicked in the “Order By” Frame. The scrollbar is used vertically and horizontally scroll | The List is first ordered by brand name from Z to A, then Brand Name A to Z, then quantity lowest to highest, then quantity highest to lowest. The Scrollbars are synced and works as expected. | Pass | <https://youtu.be/crhemIQFoLU>  0:00-2:01 |
| 15 | Making sure that a Product/PIP is chosen before the “Edit Stock Quantity” button is clicked | Erroneous | No PIP is selected  and the “Edit Stock Quantity” button is clicked | An error message is displayed asking the user to select a PIP code and try again | Pass | <https://youtu.be/crhemIQFoLU>  2:01-2:57 |
| 16 | I will be testing the function of editing the quantity of a stock/product | Typical | The PIP number “301416” is selected the new quantity entered is “120” | A new window is displayed for the user to enter a new quantity. Once the “Update Quantity” button is pressed the quantity of the stock/product is changed from 2 to 120 | Pass | <https://youtu.be/crhemIQFoLU>  2:57-3:43 |
| 17 | Making sure that when a new product is added all the validations are checked | Erroneous | PIP: “01tst0”, no Brand Name selected, Product Discription: “”, no Use is selected, Pack Size: ”£test1”, Product Cost: “-12.34”, Quantity: “10.89”, and no Supplier selected | An error message is sent along with each entry box or dropdown menu once the “Add New Product” button is clicked. Everything is also cleared and reset for the user to try again. | Pass | <https://youtu.be/crhemIQFoLU>  3:43-5:58 |
| 18 | A New Product is Added When all the data entered is valid | Typical | PIP: 1001,  Brand Name: E45  Product Discription:  “Test Discription 500mg”,  Use: “Pain Relief”, Pack Size: “50”, Product Cost: “2.50”, Quantity: “12”, Supplier: “AAH” | A message is displayed saying that the new product is successfully added, then another message is displayed asking the user if they would like to add a new user or if they are done. Finally the product is displayed in the stock list. | Pass | <https://youtu.be/crhemIQFoLU>  5:58-7:58 |
| 19 | Checking that the product Information is only displayed when a PIP number is chosen | Erroneous | No PIP number/code chosen and the “Show Product Information” button clicked | An error message is displayed asking the user to select a PIP and try again | Pass | <https://youtu.be/crhemIQFoLU>  7:58-8:55 |
| 20 | Seeing if the product information of a chosen product is displayed | Typical | “1001” PIP number/code chosen and the “Show Product Information” button clicked | Shows the information of the recently added product in a new window | Pass | <https://youtu.be/crhemIQFoLU>  8:55-9:34 |
| 21 | Testing the validation function of deleting a Stock/Product from the stock list | Erroneous | No PIP is selected | An error message is displayed asking the user to select a PIP and try again | Pass | <https://youtu.be/crhemIQFoLU>  9:34-10:28 |
| 22 | The function of deleting a product from the database | Typical | The PIP number “1001” selected and the “Delete Product” and confirm deletion buttons pressed | A new window opens on top of the main window and displays the details of the PIP selected. Once the confirm “Delete Product” button is clicked another information message pops up asking the user to confirm deletion. If “Yes” is clicked, the window closes, and the product is deleted from the database. If no is clicked the window is closed and nothing happens | Pass | <https://youtu.be/crhemIQFoLU>  10:28-11:52 |
| 23 | I will be testing the validation of the search engine and the search by option menu in the “Search” tab | Erroneous | No Search By option selected, and “Test Search” entered into the search bar | An error message is displayed showing that the user must select a “Search By” option from the dropdown menu. | Pass | <https://youtu.be/1yWc2A6Ql5Q>  0:00-1:06 |
| 24 | Testing the function of searching via the Use “Search By” Option | Erroneous | Use selected in the “Search By” drop down menu and “PTSD Treatment” entered into the search bar | An error message is displayed explaining there is no such product with that use and to try again. The search bar entry box is reset | Pass | <https://youtu.be/1yWc2A6Ql5Q>  1:06-1:59 |
| 25 | Testing the function of searching via the Use “Search By” Option | Typical | Use selected in the “Search By” drop down menu and “Pain Relief” entered into the search bar | A new frame pops up and displays the information of products with the use searched | Pass | <https://youtu.be/1yWc2A6Ql5Q>  1:59-3:12 |
| 26 | Testing the function of searching via the Use “Search By” Option. Using capitalised and non-capitalised words | Typical | Use selected in the “Search By” drop down menu and “pAiN reLieF” entered into the search bar | A new frame pops up and displays the information of products with the use searched | Pass | <https://youtu.be/1yWc2A6Ql5Q>  3:12-4:05 |
| 27 | Testing the function of searching via the Supplier “Search By” Option | Erroneous | Supplier selected in the “Search By” drop down menu and “ test Suppplier” entered into the search bar | An error message is displayed explaining there are no such product with that supplier and to try again. The search bar entry box is reset | Pass | <https://youtu.be/1yWc2A6Ql5Q>  4:05-4:57 |
| 28 | Testing the function of searching via the Supplier “Search By” Option | Typical | Supplier selected in the “Search By” drop down menu and “AAH” entered into the search bar | A new frame pops up and displays the information of products with the supplier searched | Pass | <https://youtu.be/1yWc2A6Ql5Q>  4:57-5:38 |
| 29 | Testing the function of searching via the Supplier “Search By” Option. Using capitalised and non-capitalised words | Typical | Supplier selected in the “Search By” drop down menu and “acTaVis” entered into the search bar | A new frame pops up and displays the information of products with the supplier searched | Pass | <https://youtu.be/1yWc2A6Ql5Q>  5:38-6:35 |
| 30 | Testing the function of searching via the PIP “Search By” Option | Erroneous | PIP selected in the “Search By” drop down menu and “1ff123” entered into the search bar | An error message is displayed explaining there is no such product with that PIP and to try again. The search bar entry box is reset | Pass | <https://youtu.be/1yWc2A6Ql5Q>  6:35-7:20 |
| 31 | Testing the function of searching via the PIP “Search By” Option | Typical | PIP selected in the “Search By” drop down menu and “959148” entered into the search bar | A new frame pops up and displays the information of products with the PIP searched. As well as another frame under that showing products with similar use as the PIP/Product searched | Pass | <https://youtu.be/1yWc2A6Ql5Q>  7:20-8:29 |
| 32 | Testing the function of searching via the Brand “Search By” Option | Erroneous | Brand selected in the “Search By” drop down menu and “Testing Brand” entered into the search bar | An error message is displayed explaining there is no such product with that brand and to try again. The search bar entry box is reset. | Pass | <https://youtu.be/1yWc2A6Ql5Q>  8:29-9:03 |
| 33 | Testing the function of searching via the Brand “Search By” Option | Typical | Brand selected in the “Search By” drop down menu and “Paracetamol” entered into the search bar | A new frame pops up and displays the information of products with the brand searched | Pass | <https://youtu.be/1yWc2A6Ql5Q>  9:03-9:51 |
| 34 | Testing the function of searching via the Brand “Search By” Option. Using capitalised and non-capitalised words | Typical | Brand selected in the “Search By” drop down menu and “paRaCetamoL ” entered into the search bar | A new frame pops up and displays the information of products with the brand searched | Pass | <https://youtu.be/1yWc2A6Ql5Q>  9:51-10:41 |
| 35 | I will be testing the Place Order function in the “Place Order” subtab of the “Orders” tab | Typical | Placing an Order for Brand: “Paracetamol”,  Product: “500mg Tabs Large Pack”, Order Quantity: “45” | Once the information is selected the quantity price should be “”. When the “Place Order” Button is clicked an email is sent directly to the correct supplier of the product. The Order is then also recorded in the database | Pass | <https://youtu.be/PVu8vEXAau0>  0:00-2:50 |
| 36 | Checking the validation of placing an order in the “Place Order” subtab of the “Orders” tab | Erroneous | Leaving the drop-down menus blank/not selected | An error message is displayed to the user to select an option and try again | Pass | <https://youtu.be/PVu8vEXAau0>  2:50-3:46 |
| 37 | Checking the validation of the order quantity in the “Place Order” subtab of the “Orders” tab | Erroneous | Entering “-12.7” | An error message is displayed to the user explaining that the quantity entered is invalid data, and the entry box is reset | Pass | <https://youtu.be/PVu8vEXAau0>  3:46-4:34 |
| 38 | Checking the function of resetting an order that is being placed to start over in the “Place Order” subtab of the “Orders” tab | Typical | Selecting Brand: “E45”, Product: “125g Cream”,  Order Quantity: “5”, and the “Clear Current Order” Button pressed | The labels and entry boxes are all removed or destroyed, and the options are reset to only be let with the starting position of choosing a brand | Pass | <https://youtu.be/PVu8vEXAau0>  4:34-5:57 |
| 39 | Showing that the Order History list can be filtered using the “Order By” Buttons, in the “Order History” subtab of the “Orders” tab | Typical | Selecting “Oldest Oder” button, then selecting “Newest Oder” Button | The Order History list is ordered from oldest to newest date and time placed first. Then ordered from Newest to oldest date and time placed | Pass | <https://youtu.be/PVu8vEXAau0>  5:57-7:03 |
| 40 | An OrderID must be selected before an order can be cancelled | Erroneous | No OrderID selected | An error message is displayed explaining that an OrderID must be selected | Pass | <https://youtu.be/PVu8vEXAau0>  7:03-7:46 |
| 41 | I will be testing the function of cancelling an existing order that is placed | Typical | OrderID: “7” selected | The details of the OrderID selected are displayed in a new window. When the “Delete Order” Button is pressed, a message pops up asking the user if they are sure that they want to cancel the order. If “Yes” is pressed then a generated email is sent to the supplier of the order, then the “Cancel Order” window is closed. The order is also removed from the database | Pass | <https://youtu.be/PVu8vEXAau0>  7:46-9:22 |
| 42 | The validation of making sure an order is selected before viewing the Order Information | Erroneous | No OrderID is selected | An error message is displayed asking the user to select an OrderID and try again | Pass | <https://youtu.be/PVu8vEXAau0>  9:22-10:07 |
| 43 | I will be testing the function of displaying a selected Order’s Information | Typical | OrderID: “2” is selected | The details of the selected Order/OrderID are displayed in a new window | Pass | <https://youtu.be/PVu8vEXAau0>  10:07-10:46 |
| 44 | I will be looking at the function of displaying current suppliers in the database | Typical | Nothing entered, the Supplier Tab selected | The Supplier Names are displayed in a scrollable list | Pass | <https://youtu.be/0tVLhXtBvwo>  0:00-0:58 |
| 45 | The validation of a phone number and email when adding a new supplier | Erroneous | Supplier Name: “Test Error”, Phone Number: “07test12345!”, Email: “error\_email”, Address: “AQA test centre, A01 2QA” | An error message will be displayed after each other explaining that the data entered for the Phone number and email are invalid, and the user should try again. The entry boxes are then reset. | Pass | <https://youtu.be/0tVLhXtBvwo>  0:58-2:38 |
| 46 | Testing the function of adding a New Supplier | Typical | Supplier Name: “Test Error”, Phone Number: “07123456789”, Email: “test\_email@gmail.com”, Address: “AQA test centre, A01 2QA” | The new supplier will be added into the database, and displayed in the “List of Suppliers” | Pass | <https://youtu.be/0tVLhXtBvwo>  2:38-4:46 |
| 47 | The validation of displaying a supplier’s information | Erroneous | No supplier name is selected | An error message is displayed to the user explaining that a supplier has not been chosen and to try again. | Pass | <https://youtu.be/0tVLhXtBvwo>  4:46-5:28 |
| 48 | I will be looking at the function of displaying a selected supplier | Typical | “Test Error” supplier name is selected | The information of the selected supplier is displayed in a new window | Pass | <https://youtu.be/0tVLhXtBvwo>  5:28-6:22 |

# Evaluation/Appraisal

## Objectives Evaluated

|  |  |  |
| --- | --- | --- |
| **Objective** | **Met?** | **Comment** |
| 1) a) | Yes | I created a fairly secure login system using a csv file to store the user’s information, and the validations work well in keeping unauthorised persons away. The login window/page also had other built in functions like resetting the entry boxes and exiting the programme |
| 1) b) i. | Yes | An Admin Login can be used to open up an Admin Window via the Login Window. This Admin account can be used to create a new user, given the username is not taken and meets and the password meets a certain criterion so that it is regarded as “a strong password”. The Admin also has the option to display the usernames and passwords in the Display Users Window. |
| 1) b) ii. | No | I ran out of time as we are limited to time in this coursework, so I could not implement a function to delete a user from the csv file |
| 2) a) | Yes | I used tkinter for my GUI and according to the feedback I received from my client and other independent tests with peers, they found it easy to use and very simple design yet quite “attractive”. |
| 2) b) | Yes | The colour scheme I used was very retro and plain not too colourful or bold, very simple colours that are close together in the colour wheel. It also somewhat resembles the colours of the pharmacy showing a bit more brand identity. |
| 2) c) | Yes | I created my tabs using the notebook widgets for tkinter on my main page/window. It is very easy to interact with and navigate to different sections of the programme. Even navigating through windows like where you can exit the Display Users Window to get to the Admin Window then the Login Window as seen in test number 13. |
| 2) d) | Yes | All the widgets work accordingly, there are little delays that are un-noticeable so we can disregard them, other wise the graphical layout is fairly smooth. The button widgets are used a lot to run procedures and functions or to open or escape new windows for more information to be displayed. |
| 3) a) i. | Yes | The stock lists are sorted easily via the order by buttons. Making it much easier to find the product or product you are looking for in a large list. |
| 3) b) i. | Yes | The orders are also able to be sorted from newest to oldest, using the date and time the order was placed in the programme. |
| 4) a) | Yes | A product/stock can be deleted, added, or quantity updated in the Stock Management tab. The process is easy to follow through and done through separate windows, as well as this the stock lists are automatically updated once anything is deleted, added, or updated |
| 5) a) | Yes | In the Search tab there is a search bar that only searches once a Search By option, unless an error message is displayed. The Search By options are Use, Supplier, PIP, Brand and if the searched item is not found an error message is displayed explaining why. |
| 5) b) | Yes | If the Search By option is PIP then if the PIP exists in the database information about the product is displayed, as well as similar product or products with the same use. |
| 6) a) | Yes | I have created the sub tabs Place Order and Order History in the tab Orders |
| 6) b) i. | Yes | The place order sub tab is very interactive, and the user must first select a brand from the brand list then confirm and only then will the Product Discription appear for that specific brand chosen. Once that product is confirmed the quantity entry box appears and after the user confirms the quantity the order cost is calculated using the product cost of the product and the quantity entered. Once the confirm order button is pressed an email is sent to the supplier with a generated email ordering the product chosen, and message boxes are displayed confirming the order being placed. |
| 6) c) i. | Yes | Orders can be cancelled through selecting an OrderID in the Order History sub tab and pressing the cancel order button. Once the cancelation is confirmed a generated email with the order information is then sent to the supplier’s email to cancel the order. |
| 6) d) i. | Yes | As an OrderID is selected and the Show Order Information is clicked a new window with the information about the order previously placed. |

## Feedback from the main client

Hello Bilal,

    Thank you for sending me the email with the program and all the necessary files attached. I had to install python to run the programme as you said and it ran smoothly. After testing and letting my colleagues have a go playing around with the program, my thoughts on the system are as follows:

1. I appreciated and enjoyed the simplicity of the design of the overall programme. It was easy to understand what things where and where I could interact or input data. I really liked the colour scheme used, it was very retro looking and my colleagues also agree and appreciated this aspect of the design as well.
2. I found it very easy to navigate through the windows when exiting or on the main page/window using the tabs and sub-tabs, just made it more of a familiar setting and was a really impressive piece of the programme that stood out.
3. The overall layout of the widgets and labels made the windows and tabs easy to understand and process the information on the screen.
4. The security of our information has now definitely been improved with the login page and the Admin window makes it very easy to add a new user if we receive a new employee. As well as this the Display Users Window accessed via the Admin Window helps remind us of the usernames and passwords that may be forgotten.
5. The features of filtering the list boxes for the stock list and order history list. This makes it easier to sort data and make it faster to display what stock is running low even just identify a specific product without needing to search, as well as having a history of orders previously placed in a changed order.
6. It was very easy to update the quantity of a product/stock and seeing the stock list being instantly updated was also a great touch to the programme and seemed very professional.
7. Viewing the information of a product was quite useful as well especially for the students that work on the weekends at the pharmacy, I can see it as being a tool they can use to help learn more about pharmacy.
8. The search engine is phenomenal, and I especially like that it is not case sensitive, and when a PIP is searched similar products are displayed. This makes it much nicer to use and search for products and can be a good educational tool for the university or work experience students to learn on the job as products with similar uses are identified.
9. The feature of placing an order was an interesting aspect of the programme as the dropdown menus where only available to be clicked once a brand was chosen and confirmed and I also liked the way the tab resets once the "Clear Current Order" button is pressed.
10. Having emails as an integrated feature of placing and canceling an order where the email is automatically generated based on the order being placed/cancelled by the programme, was an exceptional idea. This makes it so much easier for the pharmacy to place an order directly to the correct supplier and digitalizes this process, as in the past we had to manually call separate suppliers to place/cancel an order.
11. The supplier's tab was also a useful touch as we have a lot of all the suppliers we use, and we can just click on any of then and their information is displayed for us to use. Also, we can easily add a new supplier if we needed to.

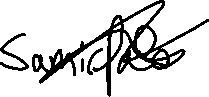
Some things I would've liked to see improved:

1. Deletion of User's data, if an employee is fired or no longer works with us.
2. Deletion of a supplier in case a supplier’s information is entered incorrectly
3. An error message displayed, or the programme doesn't allow you to add a product or supplier if you are adding one that already exists in the database.
4. An easy way to update the information of a product or supplier rather than going through the database to change it. I would like to update it through the programme.
5. The ability to have multiple suppliers for one product, and to have a product's cost constantly updated as it is in real life.
6. The search engine's ability improved to be able to search for similar data in the database and not just give an error if a minor mistake is made.
7. A feature that doesn't allow you to delete a placed order after a period of time or automatically updates the list today whether an order has arrived to then disable the function of cancelling an order placed.
8. I would have liked the ability to delete a supplier from our database if we no longer use them or they go out of business.

In all, I really enjoyed the programme and I was very impressed by the GUI, colour scheme and functionality of the programme, along with a proper database from our pharmacy. The programme worked better than I imagined, and this would be something that I will be implementing in my pharmacy once any minor bugs are fixed and it is at its final stage. A suggestion I would have looking forward to the future would be learning to connect the program to an electronic point of sales system (EPOS) so that the quantity of the stock is automatically updated after each purchase.

Many Thanks,

- Samir



Client’s Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Analysis of Feedback

From the overwhelming positive feedback, I received from not only my client but my peers, and family I can say that I am proud o what I programmed. I achieved most if not all my goals I set and add extra features along the way. I particularly noticed at during independent review and testing from third parties, people were impressed and just loved the functionality of the automated email sent via python when placing/cancelling an order.

Samir and his colleagues at the pharmacy used the programme for a few days and over the weekend. His university part time employees enjoyed the programme and noted it would be helpful as an educational tool for learning about medicine in a business environment. I was happy to see that my final product was functioning to a high standard as most people described and that it and be used in multiple ways.

My client also noted that he liked how easy it was to navigate and for the staff to understand how to use the programme. “The widgets were placed neatly” and “everything clearly labelled doing what it says it will do”, he said. He liked the addition of extra security with the login window, and it has made him think more about protecting his sensitive information better. Finally, the most that stood out from my conversations with Samir over the phone and inn person was the overall design, colour scheme and theme as well as the level of functionality given the time span.

From what my client Samir has said I believe he was satisfied and happy to implement my programme in his pharmacy once I “fix up” the code. This means making my code more efficient, debugging, and adding any extra features that my client and I have agreed upon. I have taken his criticism under consideration especially with having everything stored in one database rather than in a csv file as well and encrypting my information. I also want to learn more about EPOS and barcode scanners linked to them, so that I can potentially improve my programme through that.

## Extension

Somethings I would implement to my current programme or change are:

1. Storing the user’s information (Usernames and Passwords) in the database file instead of a csv file and encrypt the database as well. This will make the private sensitive information more secure as well as making it easier to store and manipulate the data for a user, so that I can create SQL queries to delete or update a User’s information. To do this I will just need to create a “Users” table in my database and store the current users there.
2. Add or use all the listed products from the spreadsheet onto my database, due to time constraints I was unable to, as there where over 1,300 products. I can now add these into my database through my programme, or I can research a more productive way of adding large information into databases and working with big databases.
3. As from Samir’s feedback, adding the minor features that I did not have time to add due to time constraints such as deleting a User’s information and Supplier information from the database. As wells as extra validation features such as not having the same supplier or product added into the database, and if a user tries to add it, then error messages are displayed. To delete the data, I would just need to create delete statements for the supplier in SQL and User in in CSV. For the validation I could check the database so that before a new supplier or product is entered, the system checks if it already exists in the database.
4. The ability for my search engine to search for similar results in my database depending on what “Search By” filter is selected, which my client Samir has also recommended I do in the future. To do this I can use I can try ApexSQL Search, which is a free SQL Server Management Studio (SSMS) built in, this makes it easier to create search queries for what I am aiming for.
5. My client’s suggestion of adding a feature to the Order History sub tab so that an order placed after a period of time cannot be cancelled. To do this I would simply get the Date and Time placed for that order selected and compare it to the Date and Time of when the Cancel Order Button was pressed.
6. Connecting my database to an EPOS system, would be another way I can further advance my programme. So that when an item is purchased or if an item is ordered and arrives it is automatically updated in the database and stock lists. To do this I would need to learn the workings of an EPOS system and how to link it to a database that can constantly be updated, then use that database in my programme.
7. Linking my product costs to real constantly updating supplier databases so that the cost is always correct, however I was limited by my knowledge at A level and access to large pharmaceutical suppliers’ databases and servers. To do this I have recently learned that the EPOS system is connected to the databases and servers of main suppliers in the UK, so if I can again, connect my database through the EPOS I would have a better functioning programme. So, nothing would be entered manually, and everything can be automated.
8. Having a low stock warning system in place would’ve been a crucial feature especially if it was connected to an EPOS, the system could be programmed to notify the user which stocks are critically low and a list of products to reorder and once one button is clicked all the products in the list are reordered simultaneously through generated emails. However, I would need to do more research into this so that I am able to work with the EPOS system and large databases.
9. Having multiple suppliers for one product making the programme more useful and realistic and so that I can then compare the supplier’s prices that are constantly updating to find the cheapest price or best deals. Also applying deals as you would have in catalogues when there are bulk buys in a specific product. To do this I would have needed to connect the database to a large database with the suppliers information and constantly updating costs for each product, then compare the cost for each product we have in the database to only use the best price for that day/month. However again due to my lack of knowledge of this at A level I could not have pursued this.

## Conclusion of Project

Overall, my client was happy with the finished product and so was the rest of the pharmacy. Samir had little criticism and was fairly harsh on my project so that I could further improve it for the possible future system of the pharmacy, also it was all up to his standards and goals that he set for me when we first had the talk about the project. I believe given the time I had and the little knowledge at A level I produced a better functioning system than I imagined.

I will be looking to further continue my research and development into this project in the summer. To do this I will be applying for computer science courses on databases and data handling as well as reading books about computer science in the business place so that I have a better insight into this.

# Bibliography

1. Unleashed - <https://www.softwareadvice.com/inventory-management/unleashed-profile/>
2. Trade Gecko - <https://www.capterra.co.uk/software/143110/tradegecko>
3. Capterra - <https://datainsights.capterra.com/p/inventory-management/143110/tradegecko/references?c=&c=&c=&sc=1158402>

1. <https://www.softwareadvice.com/inventory-management/unleashed-profile/> [↑](#footnote-ref-1)
2. <https://www.capterra.co.uk/software/143110/tradegecko> [↑](#footnote-ref-2)
3. <https://datainsights.capterra.com/p/inventory-management/143110/tradegecko/references?c=&c=&c=&sc=1158402> [↑](#footnote-ref-3)
4. Due to sensitive information I will not be showing my Pharmacy\_Password environment variable [↑](#footnote-ref-4)