
A level Computer Science NEA project – Stock Management System

By Rushabh Dharamshi

Contents

Contents

Initial Ideas.....	8
Justification of the project I am going to choose.....	11
1. Analysis	12
1.1. Problem Identification	12
1.2. Stakeholders	14
1.3. Research	14
• 1.3.1. Zoho Inventory System.....	15
• 1.3.2. Oracle NetSuite	17
• 1.3.3. Python Stock Management System.....	18
1.4. Essential features from analyzing existing software	21
1.5. Interview.....	22
• 1.5.1. Further Meeting with my client	26
1.6. Solution Requirements	28
1.7. Success Criteria.....	30
1.8. Limitations	35
2. Design	36
2.1. Decomposition	36
2.2. Justification of the Processes	37
2.3. Mini Interview.....	38
2.4. Defined Structure	39
2.5. GUI Design	40
• Login	40
• OTP code design	41
• Forget Password design	42
• Login Design Feedback part 1	42
• Database Login Design	45
• Database login Feedback	45
• Login table data dictionary.....	46
• Dashboard.....	47
• Dashboard Design Feedback part 1	48
• Supplier design.....	49

• Supplier Design Feedback part 1	51
• Supplier data dictionary.....	53
• Product	53
• Product Feedback Part 1	54
• Category database data dictionary.....	54
• Product database data dictionary.....	55
• Sales Design	56
• Sales data dictionary.....	56
• Sales Feedback Part 1	57
• Inventory Adjustment Design.....	58
• Inventory Adjustments design feedback Part 1	59
• Analysis design.....	60
• Analysis feedback part 1	60
• Stock-out Dashboard Button design	61
• Stock-out Feedback Part 1.....	62
• Reorder point design	62
• Reorder point design Feedback Part 1	63
2.6. Changes to be made in my design	63
• 2.6.1. Supplier	63
• 2.6.2. Inventory Adjustment design – No longer needed	64
2.7. Database Entity relationship Diagram	65
• Normalised database entity relationship diagram.....	67
2.8. Changes to be made in design part 2	68
• 2.8.1. Supplier Design	68
• 2.8.2. Product design	68
• 2.8.3. Sales design	69
• 2.8.4. Dashboard design.....	69
2.9. Final Data Dictionary	70
• 2.9.1. Final login Table Dictionary.....	70
• 2.9.2. Final supplier Table Dictionary.....	70
• 2.9.3. Final Product Table Dictionary	70
• 2.9.4. Final Category Table data dictionary	71
• 2.9.5. Final Sourcing Table data dictionary.....	71
• 2.9.6. Final Sales Table data dictionary	71
2.10. Algorithms	72

2.11.	Summary of Data structures to be used:.....	83
2.12.	Class Diagram.....	84
2.13.	Validation.....	85
2.14.	Test Plan	88
•	2.14.1. Post development data.....	88
•	2.14.2. Testing during development (Iterative data)	95
Development		106
Stage 1: Designing the Dashboard user interface		106
Stage 2: Supplier Window.....		114
•	Connecting the database	119
•	Testing the Supplier ADD function:.....	124
•	Testing: Populating supplier details into entry boxes from treeview.....	128
•	Testing: modify function	131
•	Testing: whether the ID can be updated	135
•	Testing: Delete function	136
•	Testing: Search function.....	141
Stage 3: Product Window		148
•	Testing: Add function in Category table	150
•	Testing: Delete function in Category table	154
•	Testing: Update function in Category table.....	155
•	Testing: Adding an existing category name	157
•	Testing: Adding a product with no user input in Product table	160
•	Testing: User adding an existing product (name) in the Product table	160
•	Testing when the user puts input only in the product entry box	161
•	Testing with only product and non-existing category name as input.....	162
•	Testing: Entering the wrong product data type	163
•	Testing: Entering an invalid reorder point and quantity (current)	165
•	Testing: Entering the wrong Product ID data type	166
•	Testing: Updating the quantity and reorder point of Product table	169
•	Testing: Updating the product table with a non-existing category name.....	172
•	Testing: Deleting an unselected record from the Product table	173
•	Testing: Deleting a selected record.....	174
•	Testing: Searching for a record with no input.....	176
•	Testing: When user selects an option but no input in the search bar	177
•	Testing: When user puts input into search bar but doesn't select an option	178

• Testing: When user puts all input (option and text)	179
• Testing: User input that doesn't match with any of the records.....	183
• Testing: User adding a non-existing supplier ID in Sourcing table	185
• Testing: User adds a non-existing Product ID in Sourcing table.....	186
• Testing: Adding 2 existing inputs (exists in Supplier and Product table)	188
• Testing: Adding a new supplier ID to a existing product ID in the sourcing table	190
• Testing: Whether the program allows one supplier to supply multiple products	190
• Testing: Updating source table to a product ID that doesn't exist in the product table	194
• Testing: Updating Supplier ID that doesn't exist in the Supplier Table	195
• Testing: Deleting a record from sourcing table.....	197
• Testing: Deleting a product with the wrong supplier ID but an existing Product ID in the sourcing table.....	199
Stage 4: Sales Window	202
• Creating the Sales CSV file.....	204
• Testing: Importing CSV file into the Sales table	206
• Testing: Make sure the date for the CSV sales file is in the correct format	209
• Testing: Updating the product levels correctly	213
• Testing: Sales Quantity Graph for last 7 days	221
• Testing: Value graph (£) for the last seven days	227
• Testing: Forecast of value made for tomorrow for a given Product ID.....	231
• Testing: Forecast of total quantity sold for a given Product ID for tomorrow	232
• Testing: Forecast total value (£) for next month for a given Product ID.....	234
• Testing: Forecasting the quantity sold next month for a given product ID	236
Stage 5: Analysis Window.....	237
• Testing: Graph showing top 5 products	239
• Testing: Shows top 5 products in descending order in treeview.....	242
• Testing: Graph to show worst 5 products	244
• Testing: Treeview showing worst 5 products in ascending order	245
Stage 6: Functionality of Dashboard.....	246
• Testing: Shows a number on the Reorder point button to show items less than reorder point.....	247
• Testing: Shows a number on the Stock Out button to show items less than reorder point	248
• Testing: Show total sales in value (£) for most recent date.....	249
• Testing: Messagebox showing the items out of stock or need reordering	252

Stage 7: Reorder Point window.....	254
• Getting an App password for Python	254
• Testing: Treeview showing products and suppliers when product quantity < reorder point	260
• Testing: Email being sent to suppliers.....	261
Stage 8: Stock-out window	263
• Testing: Treeview showing products and linked suppliers with 0 stock Quantity....	264
• Testing: Email should be sent to supplier.....	264
Stage 9: Login system – 2 factor authentication.....	266
• Testing: When the user enters then username and password correctly, OTP code should be sent.....	271
• Testing: Checking if OTP code matches the user's input	272
• Testing: OTP code validation when user forgets password	276
• Testing: Password being updated.....	278
• Testing: Password characters converted to *	280
Stage 10: Exit button	281
Evaluation	281
• Stakeholder testing.....	281
• Client response.....	281
• Client testing using post development data.....	291
• Evaluation of success criteria using Post development testing.....	334
• Usability features	346
• Limitations	360
• Maintenance Issues	361
Bibliography	362
Appendix	364
Code for creating tables and setting up database	364
Code for Dashboard window	364
Code for Supplier window	368
Code for Product window	373
Code for Sales window	384
Code for Analysis window	388
Code for Stock-Out window.....	392
Code for Reorder Point window.....	394
Code for Login	396

Centre number: 51427

Candidate number: 6117

Initial Ideas

Before starting my NEA, I need to decide what project I am going to do and why. There are so many factors to look at when considering what project to do such as the problem it is trying to solve or the benefit it provides, the complexity of the project, the programming language I am going to use, the features I can add to my program to make it complex and the time it will take to complete it.

I have 3 ideas in my mind:

- Quiz app
- AI chess
- Stock Management System

<u>Project</u>	<u>Complexity (1-9) (low-high)</u>	<u>Programming language</u>	<u>Features I can include</u>	<u>What problem am I solving?</u>	<u>Time it will take to complete</u>
Quiz app	7	Python	<p>Leaderboards – this can encourage people to use the app more so that they can make progress and move higher up in the leaderboard each time they answer a question correctly.</p> <p>Change the complexity of the questions based on the user – this can be done using AI that can make the questions harder as the user gets it correct.</p> <p>Include graphs that can compare with</p>	<p>This can help students revise for exams especially A levels when they are under a lot of pressure and need to learn the content quickly .</p> <p>By making the GUI, the software will look very interactive and friendly to use.</p>	<p>If the complex features are added, then developing the program can take roughly 2 months.</p> <p>However, to ensure that the user doesn't get asked too many repetitive questions, I will have to create lots and lots of questions manually or I could use web scraping to extract questions from other websites.</p> <p>This could take a while to do</p>

			another person – similar to leaderboard but this time data is being visualized making it easier to compare performance with other people.		which could extend the project by another month. Estimated time: 3-4 months
AI chess	9	Python	<p>Programming chess will be quite a challenge to do so as I will need to ensure that only valid chess moves can be played and that the program recognizes when pieces can or cannot be captured.</p> <p>Implementing AI- this could be by using the minimax algorithm or alpha-beta pruning. This will ensure that the computer plays the most optimum move and will give the user quite a challenge.</p> <p>Option to analyse your performance – this could be through rewinding the game and the program could</p>	<p>Top chess players can get bored of playing against other opponents as they aren't getting challenged enough.</p> <p>Incorporating AI will challenge the top chess players where they will be able to learn from their mistakes by analyzing the moves they played and hence improve their game.</p>	<p>This project is definitely complex, but it will take a lot of time to program all the chess pieces and ensure that each move is valid.</p> <p>Implementing AI will be quite challenge and time consuming. I will need to plan carefully, if I am going to do this project.</p> <p>Estimated time: 5-6 months</p>

			detect what the user's best move could be. I could use stacks to store the coordinates of the moves of the chess pieces where I can then use recursive backtracking to show the user's latest moves.		
Stock Management System	7	Python and SQL to retrieve data from tables within a database	<p>Graphs – this can show the performance of the products within a certain period. Graphs can help identify the most and least popular products based on the quantity sold.</p> <p>Forecasting – this can predict the sales of a product for future such as for tomorrow or the total sales for next month.</p> <p>Message alerts – this can be used to show alerts when product quantities are getting low and need reordering.</p>	<p>This can be used to help small businesses such as food stores keep track of the quantity of the products in the store. This ensures that they have enough products to meet the customer demands.</p> <p>Analysing performance of products helps owners boost sales by discarding products that have a poor sales performance and selling a lot more products that customers are</p>	<p>This project will not be too time-consuming once the database and all the tables are made. The features mentioned are not too challenging to program.</p> <p>The GUI should be made for this program so that it is easier for users to use and interpret data used within the software.</p> <p>Estimated time: 3-4 months</p>

				more likely to buy.	
--	--	--	--	---------------------	--

Justification of the project I am going to choose

Based on the table above, I have decided to do the Stock Management System as my NEA project. This is because this project has real-life application to businesses that sell goods and services to people. This will make me realise the importance of softwares used in retail and how they can massively aid businesses in making the right decisions. By choosing this project, I will be able to develop my Python and SQL skills and learn how to retrieve and manipulate data from tables within a database. This project is neither too complicated nor simple, which is suitable for the time given to complete this NEA.

Chess would have been a good project for complexity, but developing the complex features will take too much time in both coding the pieces and testing them out as there are so many scenarios that the chess pieces could be in. Developing an AI chess will require lots of research which again will be very time-consuming which could lead to the software not meeting the main criteria.

Making a quiz software would be beneficial as it will help me learn my A-level content faster as well. However, for it to be useful, there will need to be a lot of emphasis on the GUI to ensure users are attracted to the software and getting a dataset of questions will take a long time to do so.

In conclusion, I will be doing a Stock Management System as it can be completed in the time given to make this project and all the complexity features that could be added in the system such as forecasting have a real-life application which can help businesses make better decisions.

1. Analysis

1.1. Problem Identification



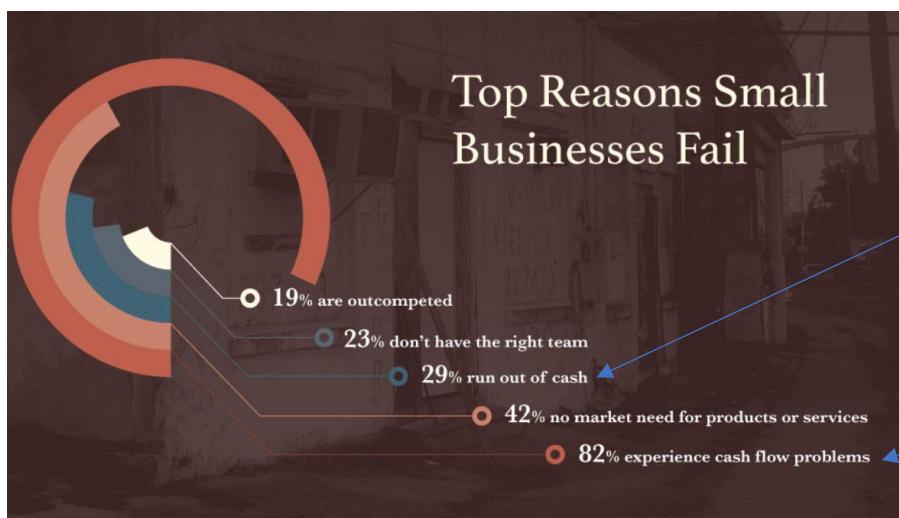
Retail business is challenging but it's even harder for small businesses as they lack the operations control and visibility. As you can see from this graph above, small businesses have an extremely small long-term survival rate with only 2.38% of businesses being successful after 10 years. My analysis has shown that typically small retail grocery businesses (known as corner shops) do not have the software to track their stocks and they end up relying on manual/visual methods to decide which products have less stock and place orders with suppliers. This manual approach is error-prone and results in either excess stock or lost customers as it is difficult to track inventory across hundreds of items. I would like to remove this manual approach and use a 'Stock Management' software program to manage business operations smoothly, helping to satisfy customers' demands while also ensuring that resources like money are spent efficiently on the right stocks. The software will have the capability to track inventory, analyze data and identify product demands, allowing ordering of the right products at right time. This ensures that the stocks are available when needed, helping to provide the end customers with the best services.

When new stock is purchased from suppliers, manual checks are made to ensure it is received in the correct quantity but once it is in the store (either in the backroom or on the shelf), the tracking of quantity is lost and there is a reliance on a visual check of the shelves to identify which product is running low on stock to place the next order with supplier. The user maintains a book where they manually enter which product has run out or is low on stock. With hundreds of products, it is an onerous task to keep tracking stock and can lead to errors. This lack of inventory visibility is a serious business issue as there are not only operational challenges (not buying product in time) impacting customer service but it also leads to cash-flow problems (buying products which are not needed).

Stock management challenge can be solved through computational methods since it requires several key components that must be coordinated for the software to fully function. For example, my software will need a database to keep a record of all the stocks available and provide visibility on customer demands. As small retail grocery businesses grow over time, they will have more customers and products causing more manual effort to maintain stock control and to create supplier orders within a certain time. Over time, they will have so

many demands that it will be impractical to handle all the operations manually since it will be time-consuming. Having a database allows you to hold the details of stocks concurrently such as the quantity available in the store. This can help to quickly identify if a product is available when a customer asks for it. By having a primary key such as a unique ID number, the exact product record can be identified and displayed to the user compared to the manual approach where the user will refer to stock ledger. In case, the business wants to adjust the inventory for a particular product (example: due to theft or damage), you can easily use the software to update records which will adjust the inventory and allow robustness within the software.

Additionally, a database can keep track of the company's sales, inventory holding costs, and profit margins which is a good indicator to the business about how well they are doing. This is significantly important as according to the chart below, 82% of businesses fail as they experience cash flow problems. Keeping track of profit margins will allow them to make decisions such as whether the price of some stocks should be raised higher or lower depending on the availability of some of the stocks and the demand for them. Having a data mining feature can look into historical sales data and help identify products with a higher or lower demand which can drive better business decisions about which products to continue selling whilst also discarding any low performing products. For a new business, analyzing it manually might be acceptable since there will be not much past data to work on but over time as you have more stock demands, it will be unsuitable to analyze huge amounts of data without any technology used.



Though there are many reasons for businesses to fail, keeping a stock management system with a database can allow businesses to not have to deal with any budgeting and cash flow problems.

Another computational method that could be used in my stock management system is abstraction. Abstraction is the technique of removing unnecessary details. In this case, I could remove any irrelevant product details such as storing the actual image. Though it does help visualize and may make the user identify the item quicker, an image will take a lot of data, which will not be needed. Instead, we could make the items a bit more descriptive by mentioning the colour of the item if some are similar or identical. For example, some bananas could be yellow, and some could be green. Instead of having visual images, just

mentioning the colour and the food item e.g. green bananas will help the user distinguish between similar products.

Decomposition can be used in my software which will help break a problem into different parts. For example, the stock management system will keep track of the stocks of the products. The businesses will then give us a file of their sales record showing the quantity of items sold for a certain product. This will be imported into my system. This will automatically update my stock levels for each item. If the stock levels go down to a certain point, a stock alert will be made which will allow the manager/person in charge to request orders from the supplier. This will then increment the stock levels of the products once the stocks are received from the supplier and the whole operational cycle begins again.

1.2. Stakeholders

The main stakeholders for this system will be businessmen and their employees who have a small store setup and currently rely on manual operations for stock management. Although my requirements have been gathered from grocery store perspective, the software is not restricted to grocery items. This means that the stakeholders will not be fairly representative of the whole population as I have collected and discussed the requirements only from a Retail grocery store but I believe it is a complex setup which can be applied to other businesses as well. This software can be applied to any business that deals with assets in terms of stocks – regardless of what the products are such as food, clothes, watches etc. Though businesses with large or good amounts of historical stock data are more likely to benefit from this software, new businesses can also use this software to try and adjust quickly to the new system that will help them in organizing all their operations in the first place.

More specifically my stakeholder is a local retail grocery shop called ‘One-Stop’, which is a small store run by the owner and his trusted manager. The store serves the needs of the local community by providing essential grocery items. The software will be made according to the requirements of the owner/manager of the food shop. I am planning on doing an interview with the manager and owner shortly and will ask for specific features that he/she would like in the software that would really improve his/her first operations. These stakeholders are the only ones with whom interview will be done.

Another stakeholder are the end-customers who will indirectly benefit from this new system. As the system is introduced to the retail shop, with better inventory management there will be fewer occurrences of stock-outs and the shop will be able to provide the best services to the customers. However, as there is neither direct interaction of end-customers with the software nor there is any involvement in the operations of the store, I have not considered end-customers as a stakeholder for my interview.

1.3. Research

As part of research, I investigated three existing software available in the market:

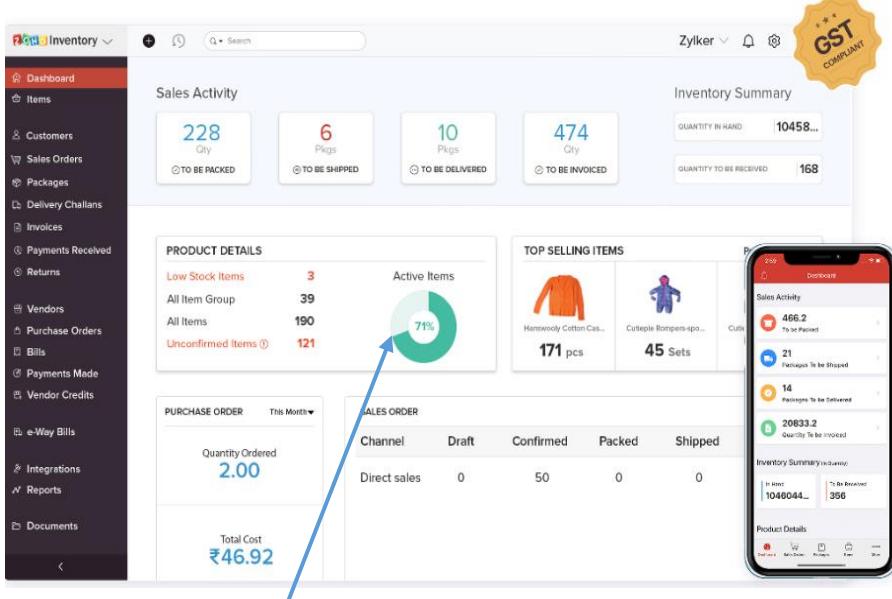
- Zoho Inventory System
- Oracle NetSuite
- Online Python Stock Management System

1.3.1. Zoho Inventory System

This is a very popular software that has many features that I can add to my system. The good thing about this system is that it doesn't have to be for a specific business such as a sports store. This system can be used for any retailer making it very versatile and easy to use.

The screenshot shows the Zoho Inventory System dashboard with the following sections:

- Sales Activity:** Displays four status boxes: "TO BE PACKED" (Qty: 0), "TO BE SHIPPED" (Pkgs: 0), "TO BE DELIVERED" (Pkgs: 0), and "TO BE INVOICED" (Qty: 0).
- Inventory Summary:** Shows "QUANTITY IN HAND" at 0 and "QUANTITY TO BE RECEIVED" at 0.
- PRODUCT DETAILS:** Shows "Low Stock Items" (0), "All Item Groups" (0), and "All Items" (0). A callout box notes: "The product details section gives us a quick insight into items that are low in stock so that they can be re-ordered again." An arrow points from this callout to the "Low Stock Items" section.
- TOP SELLING ITEMS:** Shows "No items were invoiced in this time frame". A callout box notes: "This immediately tells the user about the stocks that are doing the best and allows them to prioritize those stocks so that they never run out. Displaying it in the dashboard makes it easier to read and get a quick summary." An arrow points from this callout to the "TOP SELLING ITEMS" section.
- Overall Summary:** A callout box notes: "This dashboard includes a summary of the current state of the stock levels. This makes it really easy for people to get an overview of their current stocks and more stocks that are about to enter their inventory." An arrow points from this callout to the "QUANTITY IN HAND" and "QUANTITY TO BE RECEIVED" fields.



The dashboard also has a pie chart to show products that are active. Having a pie chart is a good way of visually representing the levels of active items making it easier for the user to understand. The drop-down arrow on Purchase Order helps to summarize for a range of days helping the clients find out how much money has gone out of their system allowing them to aid in financial decision making.

Here you can see actual data used in the software. The GUI has been made well as it looks attractive. Images have been shown in the top selling items to make it clear to the user the exact item.

The dashboard manages all the operations showing purchase orders, and the prices are also calculated. There is no dependency on other systems such as cashier machines to generate bills, making it very reasonable to use.

Features I like (Zoho):

- Leaderboard showing top selling items: This will be useful for my client as they can quickly analyze their stocks and make sure that their inventory levels for those stocks never go down. This will help satisfy customer demands and improve their reputation as a food store.
- GUI – This is what I will use in my software since it makes it so much easier to interpret and use the software itself. The GUI will also make the system very attractive to use and the different colours will make the software lively.
- I liked the idea of having a cashing machine that could potentially replace using an actual hardware. This would make it easier for my client, Parth to use, instead of relying on physically using the machine. I will have to ask the store manager/owner about this idea and whether its worth integrating this into my system. This software feature could potentially print out bills which can then be paid by the customer using the credit card machine (an external hardware provided by the food store).
- The layout – I like the idea of having the menu on the left-hand side and all the options in the menu. This makes it easier to go to different sections of the software like reports, bills and invoices, making the software very organized.

- There's a table showing sales order which has all the stock values. This makes it easier for the client to view any figures. I could use this in my software so that my client can have a direct view of information such as supplier details that could potentially be used to send emails in case some stocks are short.

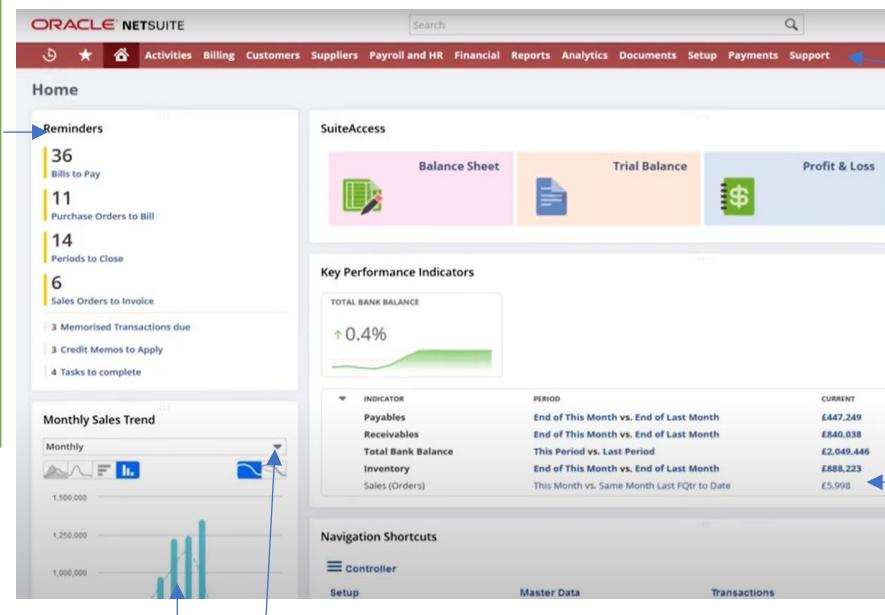
1.3.2. Oracle NetSuite

About this software: This can help businesses gain a real-time view of key suppliers and can manage inventory automatically. It helps keep track of safety stock, reorder points, inventory in multiple locations etc. It is a cloud-based system meaning that you can access the software provided you're connected to the Internet.

This is the financial management aspect of the Inventory software.

The reminders section seems quite useful as it allows the user to have a collection of all the tasks (such as paying the bills), making sure that all the processes are up to date.

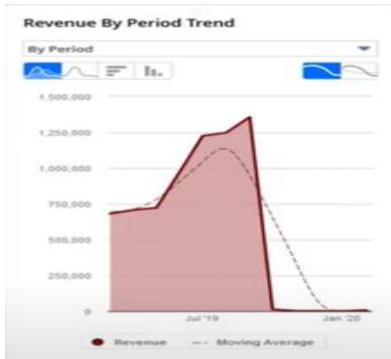
The reminders are linked so they take the user to the page where they can do their task. For example, the bills to pay reminder will take the user to show all the bills that need paying.



The monthly Sales trend is shown by a graphical representation (bar chart) just like Zoho. The down arrow indicates that you can see the sales trend for different periods providing flexibility to compare sales for different periods. This can allow users to make decisions of during what period are the number of sales the highest.

Here you have another dashboard, with all the menus being at the top unlike Zoho. The dashboard similarly to Zoho has some summary information such as the monthly sales trend.

The GUI looks great making the software seem very organized and friendly to use whilst also making it professional. The use of blue text helps to emphasize important information such as the total bank balance.



This is also part of the dashboard. This shows the revenue again by the time period. What I like about the graph, is that the graphical representation can be flexible – it can either be a line graph or a bar chart as show on the icons.

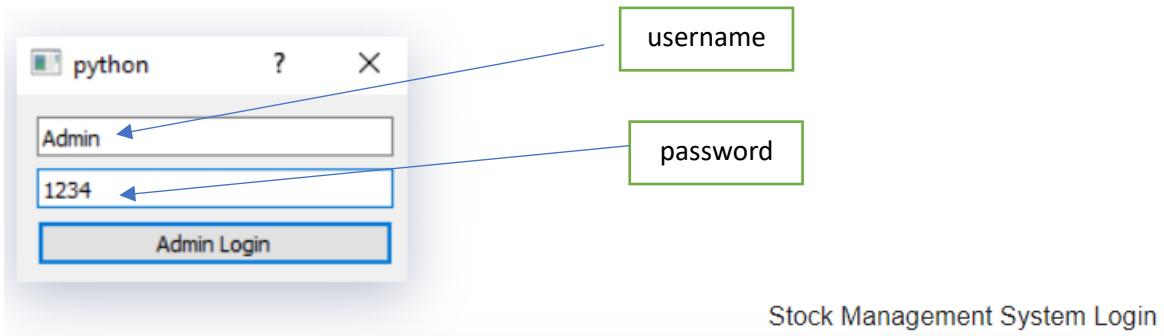
Features I like (Oracle Netsuite)

- I liked the reminders section showing all the tasks that have to be done. I can use this for my client so that he can have a list of tasks that will help any processes. This could be integrated with my re-order point so that the client can go on the reminders section and click one of the reminders that will tell him what requests he needs to make to the supplier.
- The GUI seems to look decent with all the sections being spaced out equally. This helps to ensure that the layout is proportional and uniform. I liked the use of contrasting colours, making it easier to visualize and see the text.
- The monthly sales trend shows the summary of sales for that period. This was like the Zoho software that instead showed the top selling products. I could integrate this into my system so that it shows a summary of the food store in the dashboard. However, since the software to be made, is for my client, my dashboard screen doesn't have to be really big.

1.3.3. Python Stock Management System

I found this stock management system online that uses Python and SQLite for the database. This will be really helpful for me as I can get an overview of how the system programmed in Python would look like.

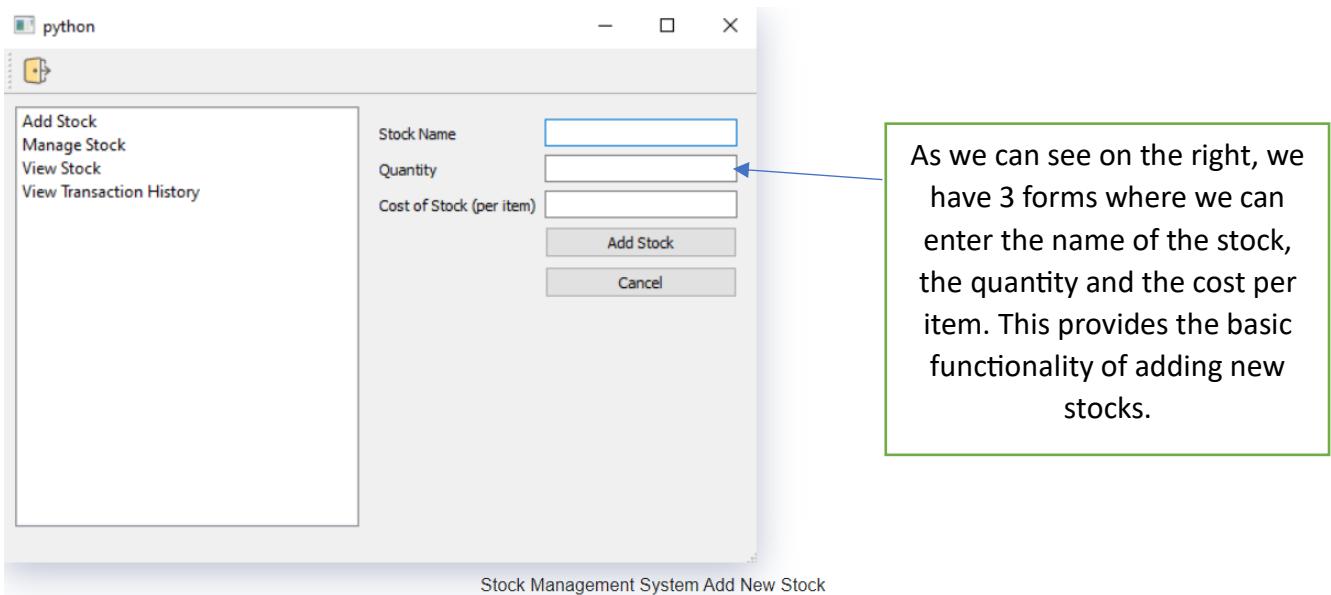
This is how the login system looks like.



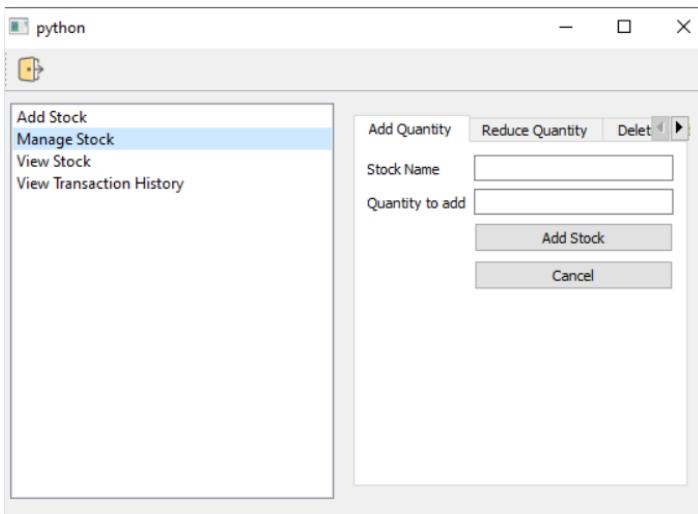
To access the entire system, the user has to login. This ensures that no third-party users can access the system, preventing them from accessing any sensitive information, that could put a business at a disadvantage.

The use of a login system is a great idea, though I feel like the one used here is a bit too basic. Though it does protect the system, there is a possibility of the person forgetting their password. Here it doesn't show an option for forget password, meaning that there is no flexibility in providing a new password.

This is the part of the software to add new stocks.

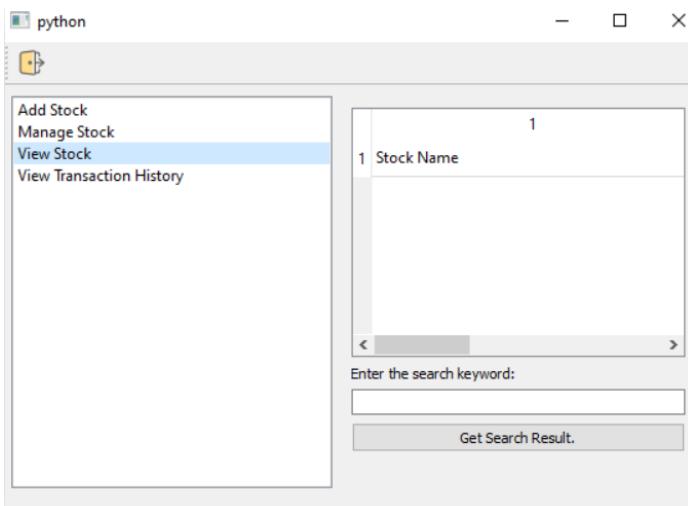


This part of the software manages new stocks.



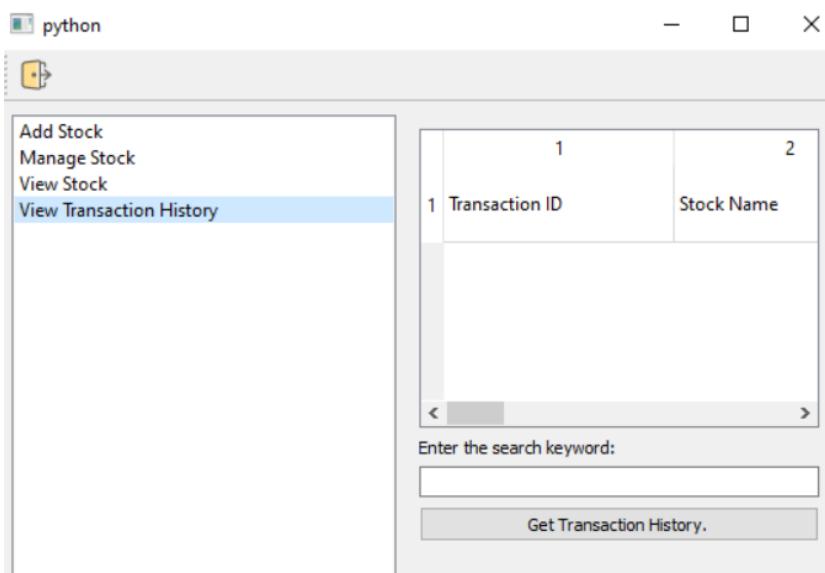
This feature provides the flexibility to add quantities of a certain stock once a supplier delivers the stocks to the store. This can update the inventory levels by adding or reducing quantity (if a customer buys a product). You can also delete stocks in case a stock is no longer needed.

This part is used to view stocks.



Here the user enters the name of the stock, which will then show the stock name and the quantity of that stock. This provides a summary of the status of the stock levels which is displayed on the right. The use of a search button makes it easier to find a particular stock item, assuming there are hundreds of records of stocks.

This is used to view the transaction history



Here, you can see the history of any transactions (such as adding, reducing quantity of a product) which allows the user to double check that they haven't made any mistakes in this process. The use of the Transaction ID serves as a primary key which can be used to identify a specific transaction.

Features that I like (Python stock Management)

- I like the use of a login system as it provides a layer of security and only allows authorized access to personal information such as sales figures, best selling products etc. I could use this in my software so that my client has to login to get access to the software. However unlike in this software, I will have a forgot password section where the user will receive a code on their phone that they can type into the system. This ensures that even if a stranger purposely tries to click on the forget password section, they would have to have the same contact details as the person they are intending to be, to gain access to the code. I might also add a 2-factor authentication to make the system more robust. This could be done by having another random code on their phone once they enter their username and password. I will clarify with my client and go by what he suggests will be appropriate for him to manage.
- The GUI – Though the system was basic, the GUI seemed decent with all the sections integrated together in one box. This makes it easier to manage the stock levels without having to move your cursor too far. I will try and make sure that in my software all the functionalities such as managing inventory, analysis are close together making them easier to access if they are in one place.
- The search section makes it easier to look for an item. I can definitely use this in my system so that the user can search for a particular product and look at their stock levels. This saves time if there are too many stocks available.
- No hardware requirements – Though the system was simple, without any charts or analysis of trends, it is quite easy to use. You don't need to buy any external hardware making price efficient. All you need is Python with either version 3.8 or 3.9 and SQLite3 which can be integrated with python to make a database.

1.4. Essential features from analyzing existing software

My solution will be a stock management system for my client's food store that will help him manage his food stock levels more efficiently. Given below is a list of features that I believe could be useful for my client based on the analysis of existing software.

<u>Essential features</u>	<u>Explanation</u>
1. Login system (I just have to confirm with the client about the 2-factor authentication)	This helps to make sure that only the manager and owner of the food store have access to the system. Others shouldn't be able to access the system as they may accidentally change some of the data figures which could affect any decisions that have to be made.
2. Reorder point alerts	Once the stocks go below a certain level, then there should be reorder point alerts that will point out that an

	item is low in stock. I should be able to view the items that are low in stock so that I can make a request to the supplier. If we don't have these alerts, then the client might forget to reorder something which could make the customers unsatisfied if they ask for that food product.
3. Dashboard	The dashboard will show a quick summary of the food store including the re-order points (showing the number of items lower than a certain level) as well as a stock out (where an item is out of stock). The dashboard will also contain a menu section that will have different parts such as an analysis section, suppliers' section etc.
4. Analysis	This will analyze the sales file that would be provided by the client. This will then be analyzed to find the most popular items and the least popular items. This would be displayed like a leaderboard showing the top 5 popular items and the top 5 least popular items. This can be used by my client to know the stocks that aren't really relevant to customers so that they can replace them with more useful stocks.
5. Supplier requests	This will be used to request the supplier once a stock is listed in the reorder points or worse stock-outs. An email would be sent to the supplier asking for the specific items that they want to be ordered.
6. Updating Inventory Levels	When the client imports the file on my software, it will automatically update the inventory levels. This is useful as though it isn't exactly real-time, the client can get a rough idea of the stock levels for any of the food products and estimate how many stocks they should have for a food item in their store.
7. Sales	Like in the Oracle Netsuite software, there was a bar chart showing the monthly sales trend. This will be used to show a graphical representation of the sales in terms of value and possibly quantity sold that will give my client an idea about how well his food store is going.

1.5. Interview

In this interview, I am going to ask the manager of the shop about any current systems that they use to keep track of their food stocks and any features that they think would benefit them greatly. This will give me a better insight into what features I should include when developing the software. The questions will be open-ended questions so that I can get a descriptive response which will clearly tell me what their requirements are.

Manager name: Parth

My questions for the manager are:

- 1. Describe the current operations done in the store from receiving the product from supplier to selling it to customer. Are you currently satisfied with all your operations that are running in your food store?**
- 2. What software are you currently using to manage inventory?**
- 3. How do you find out about the most popular food items?**
- 4. How do you know about least popular products?**
- 5. How do you know if a product is low on stock or out of stock?**
- 6. How do you analyze the store performance?**
- 7. What features would help aid your processes?**

Question 1 allows me to understand the entire flow of the product in the store and the overall process of handling their customers and operations. This will also give me an overview and help me put in context any further discussions and help in the development process.

Question 2 will tell me the exact system they are using as well as some details of how the system manages all the inventory-related operations.

Question 3 will tell me how the manager identifies the most popular food products that customers come to the store for. The objective is the store should always have stock of these items and they should be prioritized so that those stocks never run out.

Question 4 will tell me how the manager identifies the least popular food products. The objective is the store can have low stock of these items and they should not be prioritized. This can also be used by store manager to analyze and take decision if the item needs to be discontinued. This will be useful to understand how the software can be built to provide this analysis.

Question 5 will help me understand how the manager identifies low or out-of-stock products which can then be used to place supplier orders. This will aid in designing the software so that automatic alerts can be generated when stock needs to be ordered.

Question 6: Software needs to provide analysis around store inventory and performance. This question will help me to understand the current ways of analyzing performance and will help in deciding which elements can be built in the software.

Question 7 will tell me any features that their current system/business process does not have that I could potentially build into my software. This will help them adapt to any situation and manage the operations.

Parth's response

- 1. Describe the current operations done in the store from receiving the product from supplier to selling it to customer. Are you currently satisfied with all your operations that are running in your food store?**

Ans: "Orders are placed on supplier manually or sometimes we go to wholesale stores and buy. The decision to buy is done by visually checking in the store if stock is low or we are out of stock. When stock is received from supplier, we check the quantity and then put the products on the shelves. Once it is on the shelf, we do not track inventory – at any point in time, we do not know exactly how much quantity of a product we have in the store – unless we go and count on the shelf. As customers buy the products, it is recorded in our sales system. At the end of the day, a report is generated which gives the total sales value by category.

Broadly following categories are in the report:

- Bakery
- Fresh Vegetables
- Beers, Wine, and Spirit
- Grocery
- Dairy
- Cigarettes

Yes, I am very satisfied with the quality of our food products as well as the number of customers that come into our shop. We've given excellent customer service and all our prices seem to be very appropriate. We ensure that stocks match our demands, though sometimes we do have issues with no stock or over-stocks. E.g. Last week bread was in surplus."

Analysis: Their current process and sales system seem to be working fine but is heavily dependent on manual actions and it is not efficient and does not provide visibility.

Possible solution: Inventory management will help in making decisions around when and how much to order by analyzing the food product's data.

2. What software are you currently using to manage inventory?

Ans: "We do not have an inventory management system as the store is small. However, it will be very useful as many times customer asks for a product and we are not sure if we have it in stock. Also, this visibility can greatly help in placing or buying stock in time".

Analysis: There is a need for a system that can track inventory and help provide visibility along with alerts when to place the orders.

3. How do you find out about the most popular food items?

Ans: "The main food products that customers buy are milk, bread, butter, cheese, eggs, cookies, pancakes and yoghurt. Though there are many other products also in demand, these food items are the ones we've seen people usually buy. We know this by day-to-day experience and if needed we can look at the daily sales data".

Analysis: Although information is known by the manager and owner by experience, it is not available in any system. It will be useful to have this data so that these food items stock levels never go too down.

4. How do you know about least popular products?

Ans: "Again this is known by experience and generally when we see products not selling – stock not moving from the shelves, we would wait for few weeks and then either reduce the price if there is a sell-by-date or change the location of the product so that it is more visible to customers".

Analysis: Although information is known by the manager and owner by experience, it is not available in any system. Having this information provided by the system will help manager take appropriate actions in the store.

5. How do you know if a product is low on stock or out of stock?

Ans: "As our store is small, we know how much is on the shelf as we visually keep checking the shelves. Also, many times customer will ask for a product and we quickly check and confirm if it has become out of stock."

Analysis: This is currently a challenge as it completely depends on visual checks. Having a system to keep track of inventory will greatly help in identifying these cases.

6. How do you analyze the store performance?

Ans: "Every day at the end, the sales system generates a report which gives breakdown by category of how much sales have happened in value terms. As we know typical average sales, this helps us identify how the day's business was."

Date: 1 Sep 2022	
Category	Sales Value
Bakery	120
Fresh Vegetables	650
Beers, Wine, and Spirit	390
Grocery	480
Dairy	272
Cigarettes	148
TOTAL	2060

Analysis: This report is basic and provides a view of total sales each day split by categories. This can continue from their existing systems but my software can provide performance of the store from inventory perspective and give insights into top selling products while identifying products that need ordering.

7. What features would help aid your processes?

Ans: "Providing a store performance view across past weeks will be useful and also a way to identify which items need ordering will be great"

Analysis: It will be possible to generate a view of sales across past weeks and also I can add a re-order point which will suggest ordering an item as soon as its inventory falls below this level.

1.5.1. Further Meeting with my client

This meeting is done after I have had bit more thought on what is needed from the first interview to check if any additional features are needed.

In this meeting, I am going to ask my client about the following things:

- Cashing / Billing machine
- Reminder's section
- Type of Graphical Representation they like
- 2 factor authentication

My questions to him are:

1. What's your view of having a software that can produce a bill for what the client has purchased?

This question will allow me to know whether I should include a billing feature in my software. Though the client will import their external sales into my system, having a billing feature will allow the inventory levels to be updated real-time, rather than it being updated every night.

2. Do you think having a reminders section will make it easier for you to manage your tasks?

This question allows me to understand whether I should include a reminders section on my dashboard or not. The reminders section would include any reminders such as requesting a supplier for food stocks.

3. What types of graphical representations would be helpful for you to interpret data?

This will allow me to use a graphical representation that is easy for my client to interpret so that he can clearly identify any sales trends and the quantity of a particular item.

4. Do you want to use a 2-factor authentication to login to your system?

This question will tell me how complex my login system should be. Although a 2-factor authentication system would be helpful in increasing the security of any data, it is up to my clients choice of whether or not it is appropriate for him to login by.

Parth's response:

Q: What's your view of having a software that can produce a bill for what the client has purchased?

Response: "A cashing machine would be a good feature to add since it prevents from the cashier from physically using the machine making it much efficient to use. However, my system can already scan a product and tell the price automatically that gets added to a list of purchased items which is much faster than building a new billing software. Our current system also has the credit card machine integrated with this, so though the billing software may be easier to use, it will not be as time efficient as the system we currently have."

Analysis: Building the billing software isn't worth it since he already has a system for this. Instead, I will focus on the analysis part of my software that uses the sales data generated by my client.

Q: Do you think having a reminders section will make it easier for you to manage your tasks?

Response: "The reminders section will not be ideal since it is a small food store and so there would not be many tasks to do. However, having any notifications showing that a particular stock is getting low would help me inform whether I need to contact my supplier for more stocks."

Analysis: I will not add a reminders section, but I will make sure my stock alerts are effective enough for my client to see and act on it. Though my client may be interacting with the users, I will ensure that my alerts remain on the dashboard, until my client views it and clicks on the close button.

Q: What types of graphical representations would be helpful for you to interpret data?

Response: "Personally, a bar chart would be easier for me to compare between different sets of food data as it can help me identify how much value the food product has compared to another one."

Analysis: "When presenting my analysis in the form of a report, I will use bar charts as my way of representing data.

Q: Do you want to use a 2-factor authentication to login to your system?

Response: "Yes, I would like to have a 2-factor authentication system in order to login to the system. This will make my system more secure, as un-authorized users will not be able to use access the system.

Analysis: I will add a 2-factor authentication login system. As well as my client entering his username and password, I will ensure that a code is sent to his phone for him to type into the system.

Key Analysis from the interviews:

There is a need for inventory management system in the store that can not only provide visibility of inventory but also create alerts on stock-outs and for re-order points which help the store manager to raise orders. The analysis feature of showing the top sellers and least sellers can be hugely beneficial.

1.6. Solution Requirements

This section consists of the final client requirements that need to be built in the software, the corresponding hardware and software requirements.

1.6.1. Client requirements

After the interviews, the following is the list of requirements from the client that are needed in my software solution. These will also

Requirement Number	Requirement Details
R1	Provide a dashboard and easy menu icons to navigate
R2	Ability to search for a product and get the inventory for it
R3	Login ability with password
R4	Provide alerts for Stock Out
R5	Provide alerts for Reorder point so user knows when to place supplier order
R6	Ability to import sales data via a file
R7	View the sales data in Qty and Value in a bar chart for last 7 days
R8	Analysis screen to show top 5 items in terms of Qty sales over last 7 days – display information in a bar chart
R9	Analysis screen to show least 5 items in terms of Qty sales over last 7 days – display information in a bar chart
R10	New items to be manually added
R11	New Suppliers to be manually added and attached to new or existing items
R12	Ability to have a category against each item
R13	User should be able to update item name, Reorder point, Category

R14	User should be able to delete an item
R15	User should be able to update Supplier name or email
R16	User should be able to delete a Supplier
R17	User should be able to update inventory data manually if needed
R18	Ability to email the supplier with the item and qty needed
R19	Report showing yesterday's sales in value grouped by category

1.6.2. Hardware requirements

- A computer that can run the program.** – The computer doesn't have to be too computationally powerful since the food store doesn't have much historical data, but it should be able to go to different sections of my dashboard relatively quickly. A SSD is preferable for this software since my user will need access to data as fast as possible, and is much more sound efficient than HDD when processing large amounts of data. This will not disrupt my client's store and can easily gain access to any vital information.

The size of the computer display doesn't matter as I will ensure that the python windows are to a fixed dimension so that the GUI doesn't look out of proportion.

Minimum hardware specification

- Processor – A processor with a clock speed of at least 1GHz
 - RAM – at least 4 GB
 - Display – High definition at least 720p as it will be easier for my client to view any information
 - Storage – 64GB as over longer periods of time, more data will be stored in the system
- A phone** – This will be needed to receive the code for my 2-factor authentication login system. There are no requirements for the phone as long as it has the basic applications to receive emails such as messages, Outlook etc. The phone can have any operating system as long as my client can access their email as fast as possible. (Using a phone is not mandatory but makes it easier for the client to receive emails).
 - A keyboard** – This will be used to enter values into the system such as the supplier's details, the number of stock coming into the store etc. (Most laptops have an in-built keyboard).
 - A mouse** – This is used to move the cursor so that it click on the menu options to go on to a different feature in the dashboard.

1.6.3. Software requirements

- **Any operating system, Windows, Linux or Mac** – Python can work on any of these operating systems, so the OS used doesn't really matter.
- **PyCharm IDE** – This is where I will write my code. The good thing about this IDE is that is free to use and can work easily on Windows, Mac OS X and Linux. This IDE will also help me test and debug my code making it very sophisticated to write my program.
- **SQLite 3** – This software will be used to make a database and can be used to connect it with my python code. This will act as a data storage to store various data elements like items, sales history, etc.
- **DB Browser for SQLite** – Open-source tool that is used to add, and design database files and works well with SQLite.
- **Python packages** – I will be using various python packages:
 - A. **Tkinter** – This will be used to make the GUI for my software so that it is interactive and attractive to use for my client. This will be used to create user input forms as well as any messages such as error messages or updating record values messages.
 - B. **Matplotlib** – A plotting library in python that is used to visualize data. I will use this to plot graphs in my sales and analysis window.
 - C. **NumPy Pandas** - Used for data analysis and can allow csv files to be added to my database. Can be used to manipulate data by using SQL queries.
 - D. **SMTPlib** - Used to send emails to the suppliers when stocks are low or below the reorder point.

1.7. Success Criteria

Sr.	Success Criteria	Reference Requirement number	How to show Evidence
S1	<p>Client should be able to type in their username and password to access the system.</p> <p>Once the user enters their username and password correctly, they should get a code sent to their email that they would have to enter in order to finally gain access to the system.</p>	R3	I will take screenshots of my resulting outputs and will be in constant contact with my client to ensure that he

			is up to date with all the development tasks that I am making and approves of all my processes.
S2	The user should not be logged in to the system if he types an incorrect password or username.	R3	Same as above
S3	There should be a forget password section so that the user can still login even if they forget password.	R3	Same as above
S4	The OTP code sent to the email should pop up within 3 minutes (subject to internet connectivity). It shouldn't take too long or else it wouldn't be practical enough to use in my software.	R3	Same as above
S5	The stock management system must be GUI based.	R1	I will take screenshots of my GUI design and consistently be in touch with my client to check whether he likes the GUI features or not.
S6	The menu Icons should be big enough for my client to see clearly.	R1	Same as above
S7	There should be contrasting colours in the text which will be used to make the dashboard look lively.	R1	Same as above
S8	The form fields should be big enough to type or search for a food product.	R1	Same as above
S9	GUI should have buttons that go on to different sections such as the analysis menu displaying the analysis of the food products etc.	R1	Same as above
S10	If any of the item's inventory falls below a reorder point, then the reorder point button should be raised in the Dashboard. This button when clicked should show the items that are low in stock. The button should be big enough so that client can read it.	R5	I will take screenshots of my alerts and ask client whether the way the alert has been designed is

			effective in taking action.
S11	The alert should remain there until client clicks on the close button. This would imply that it has been read.	R5	Same as above
S12	If any of the items inventory becomes zero, then the stock out button should be raised in the Dashboard. This button when clicked should show the items that are out of stock. The button should be big enough so that client can read it.	R4	Same as above
S13	The alert should remain there until client clicks on the close button. This would imply that it has been read.	R4	Same as above
S14	User to have the ability to import sales files which gets imported into the database.	R6	Ensuring file format is correct, I will take screenshots showing file and then the same values in database table.
S15	The sales file should have the same structure as database.	R6	Same as above
S16	It shouldn't take too long to import the file.	R6	I will take screenshot of time required to upload the file.
S17	User should be able to view the sales data in Qty and Value in a bar chart for last 7 days	R7	I will take screenshots of my bar charts and check with the client.
S18	User should be able to find the top 5 least and most popular items in terms of Qty. This should be displayed in the form of a leaderboard that gets updated over time. The information should be presented in form of bar chart.	R8,R9	I will take screenshots of my bar charts and check with the client.
S19	User should be able to enter new items easily using the User Interface.	R10	Screenshot to show page where item is added.

S20	User should be able to add new supplier easily using the User interface	R11	Screenshot to show page where supplier is added
S21	User should be able to attach the supplier to item	R11	Screenshot to show page where item is linked to supplier
S22	User should be able to add a category for each item	R12	Screenshot showing page to add categories
S23	User should be able to update the category description	R12	Screenshot before and after the update.
S24	User should be able to update following attributes of item: Item name, description, Reorder point, Category	R13	Screenshot before and after the update
S25	User should be able to delete an item. There should be a message to confirm deletion. This action should delete data from all relevant tables.	R14	Screenshot of table before and after deletion.
S26	User should be able to update following attributes of Supplier: Name , email	R15	Screenshot of supplier before and after changes.
S27	User should be able to delete a supplier. There should be a message to confirm deletion. This action should delete data from all relevant tables.	R16	Screenshot of table before and after deletion.
S28	User should have the ability to update Inventory qty for any item.	R17	Screenshot of value entered in User Interface and the value in the table.
S29	System should allow user to email the supplier with the Item and Qty details needed.	R18	I will take screenshots of this and I will use one of my Gmail accounts to act as if I am the supplier. If I see an email

			from the client, then the process of sending email requests is working.
S30	System to generate a report that gives a summary of yesterday sales value (not Qty) grouped by categories.	R19	Screenshot of summary report.
S31	System menu should allow searching for a item and get the inventory for that item.	R2	I will take screenshot showing the search capability and results of inventory

I approve of all the processes so far. Below is the confirmation from my client of the software requirements that I will add into my program:



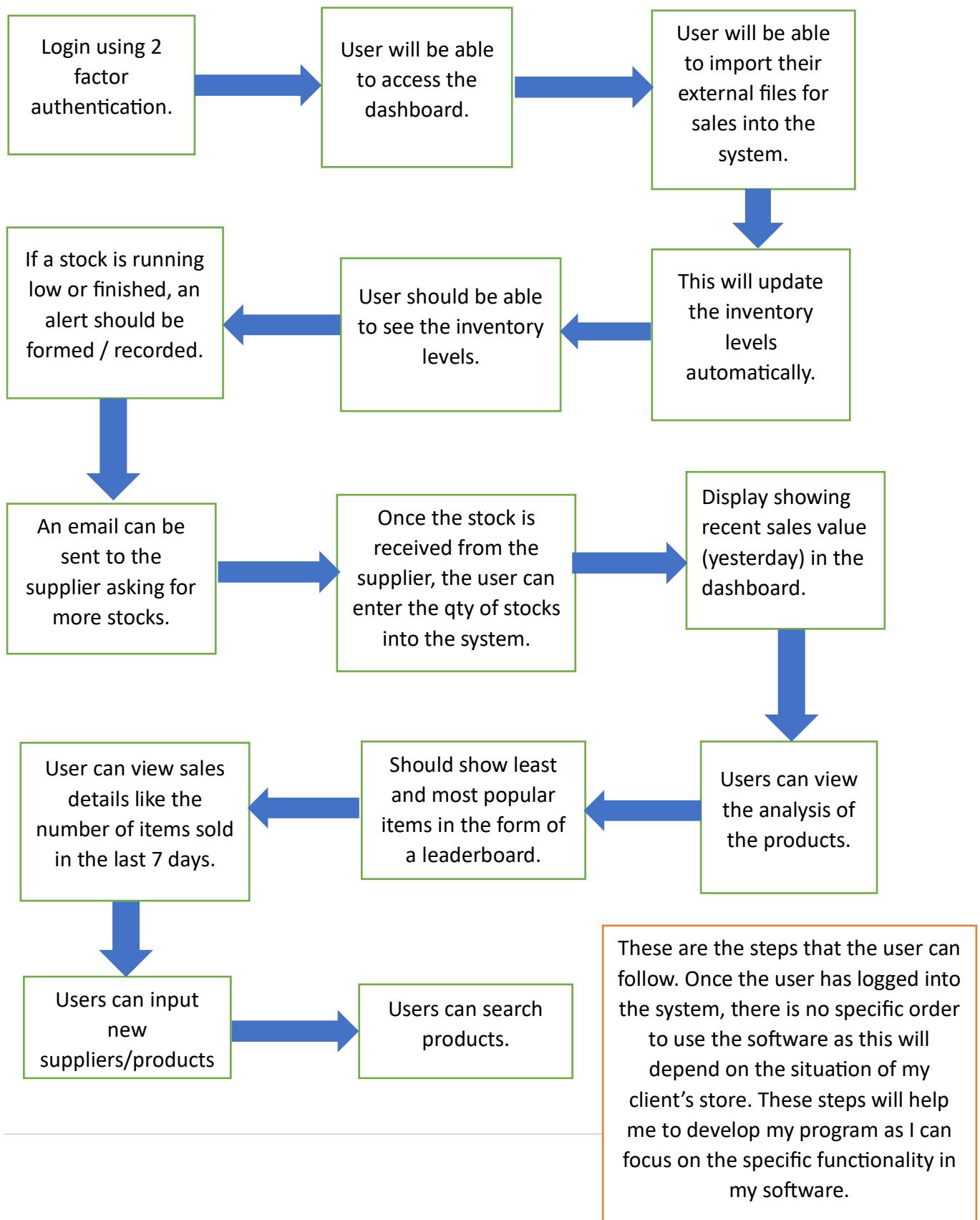
1.8. Limitations

- The main limitation of my software is that it is not in real-time since the sales file provided by the client's software will be imported into my solution at the end of the day or the next day. This means that any inventory levels will be updated only once the sales file is imported into my system.
- Another limitation is that my software does not use a barcode scanner so my client would have to manually count the number of items coming from supplier into the food store, and then enter it into the system. Unlike the first 2 existing solutions researched (1.3.1 and 1.3.2) that had a billing system with an inbuilt credit card machine, my software will not have those features so the client would have to depend on their hardware to perform any cashier to customer operations.
- Due to limited amount of time, I will not be able to make the system track how far the products are from the store. Most stock management systems will track the destination of the supplier goods and the estimated time, to get there. However, though this can help my client track where the supplier goods are, this will not necessarily help my client if he knows roughly how long each supplier takes to provide the goods.
- From the Hardware and Software requirements, my client cannot access the software on the phone as the software is not available on a browser/application. This means that all the related programs must be configured on their desktops/laptops. This makes the software not portable and so the client would not be able to access any data from devices that do not have all the software set up.
- Since my client will be sending emails to the suppliers, an internet connection will be required. This is also the case for logging in to the system where my client will receive the OTP code from their email. Without internet, my client would not be able to access the software.

2. Design

2.1. Decomposition

User Perspective:



2.2. Justification of the Processes

Login using 2 factor authentication

This is one problem that I will solve on its own. Without the login, the software should not be accessible so I will have to make sure that this process functions properly. Once this problem is solved, it doesn't really interlink with any other steps since the user verification is done. This process will work by the user entering their username and password - if it is correct, a code will be sent to their phone.

User will be able to access the dashboard

Once the user has logged in, they will be able to access the dashboard. The dashboard should contain 2 buttons – the stock-out and the stock alerts. When clicked, it should show the items that are out of stock or low in stock. The dashboard should also have the recent sales to view the most recent sales statistics. This will not be a button as I am planning on having a sales menu button that can show other information.

User will be able to import their external files into the system

This external file will contain sales recorded by my client itself. This file is in the form of Excel which can then be converted into a csv file that can be read by python. This will update the inventory levels as well as the leaderboards automatically. My client collects the sales data every day, so the file will be read by my program every night and updated every night. I just need to ask my client whether the Excel file would be the same file but updated or a completely new file each time.

An email should be sent to the supplier asking for more stocks

This email could be automated or the user could obviously do it manually. Once a stock is in the stock-alert or stock-out list, the user has to put the amount of stocks needed from a specific supplier. This would then be sent to the supplier's email asking them for new stock orders. Once the stocks have arrived, the user will update the inventory.

User can view sales details like the number of items sold for last 7 days.

This will be in the sales menu button where I can see the number of sales for the last 7 days such as the number of stocks sold yesterday, or the value of the sales made. This analysis will be done through the input of an external file from my client. For this process to happen, the external file should be successfully imported which is why I will focus on getting python to successful read the file. Once that happens, I can then do the sales analysis.

Users can view the analysis of the food products

This will be in the analysis section (menu) where the user can see the analysis of the food products such as the most performing products (most popular) and the least performing products. This will be in a leaderboard format and there will be a graph to help interpret the data much easier. Each time an external file is imported into the system, the leaderboard and the graphs should get updated if necessary to reflect the latest data available for input.

Users can input new suppliers/products.

This will be used to input any new suppliers by entering a primary key (supplier ID), their name and their email. From this email, we will be able to send messages in case we need more food stocks. This will be in the supplier's section which will make it easier for my client to remember any supplier details that he would like to contact in the future. New products can also be entered.

Users can search food products

This will be in my products section where my client can search for any food products that are available in the stock. In this section, I will also make sure that any new food products are entered into the system and database so that it can be analyzed.

Next steps:

Ask the client about how the files should be imported and whether the files containing the data would be updated on the same file or on a different file each time.

Top-down design

Start building GUI – Graphical User Interface

2.3. Mini Interview

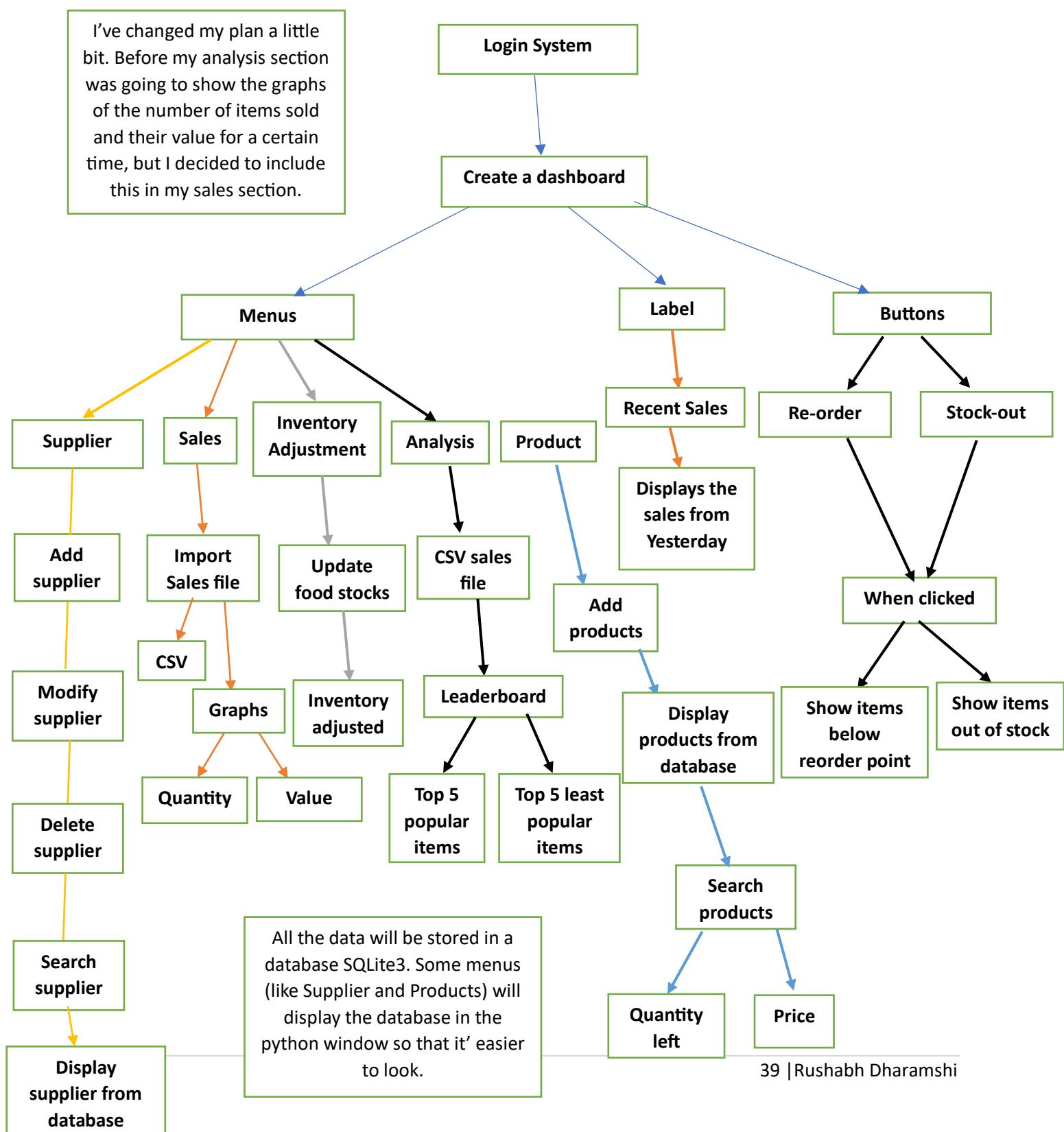
Q: Will the sales file that you give me be on the same excel file or would you provide us with a different file each time?

Ans: I will give you a different excel file each time, as I find it easier to record the sales for that day. This file will be given by the end of the day, and I will send it through your email.

Analysis: The excel file can be easily converted into a csv file which can then be added to my SQLite database which will then be updated.

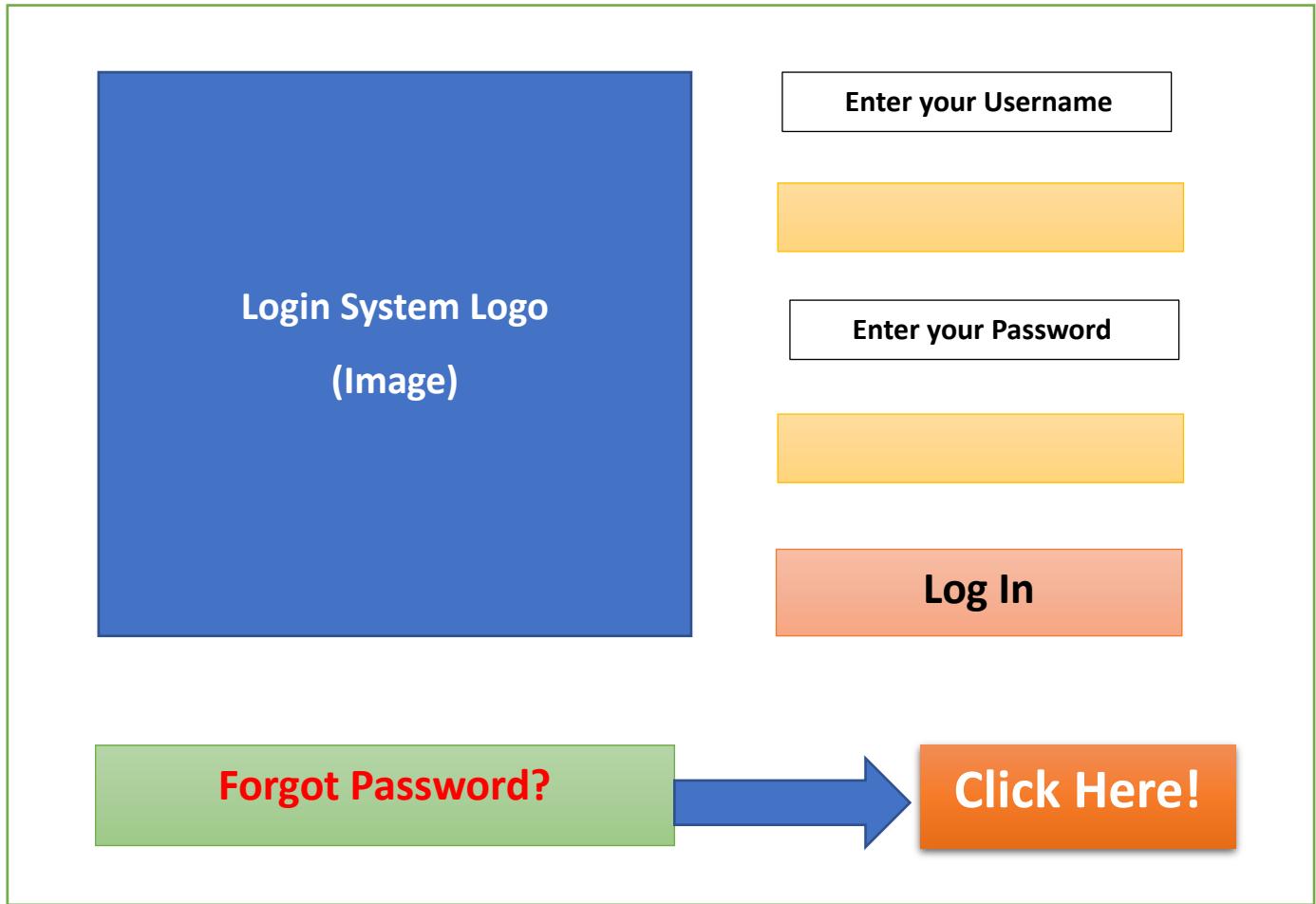
It would have been easier if the sales was recorded on the same file since I could just type the file name into Python and the same file would then be used in my program. However, since I will be using different files, I will have to create a button in Tkinter so that I can then choose the file to be imported.

2.4. Defined Structure



2.5. GUI Design

Login



This is my login system design. This is what the user will have to log into to access the Stock Management System program. The user will have to input their username and their password to get to the first stage of logging in.

Features:

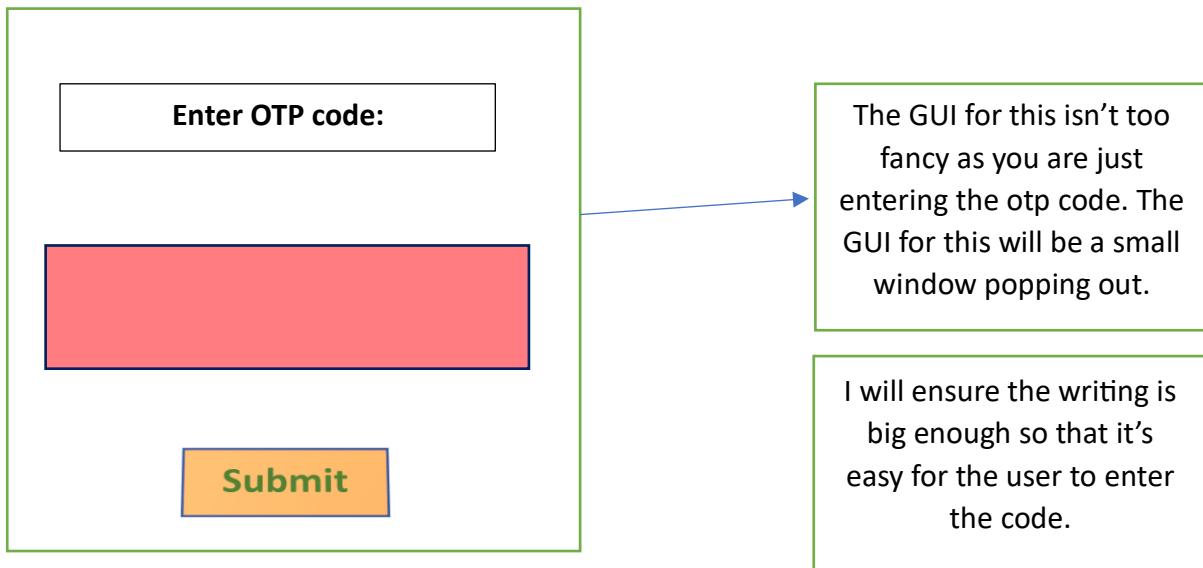
- **Logo** – This is used to make the login system look more attractive. I will have to talk to my client about what design will be appropriate for the logo.
- **Username entry** – This is used to allow the user to enter their username.
- **Password entry** – This is used to allow the user to enter their password.

Links to success criteria: I will ask the user to put in their username and password, so only users authorized can have control of the software. This ensures that no unwanted users cause any changes to the program.

- **Log in button** – This takes the user to another window that shows the op code form that they must put in. I will create the design for this later.
- **Forgot Password Label** – This directs the user to the Click Here button and allows robustness within the system in case the user forgets their password. I will create the design form for forget password later.
- **Click Here Button** – This will show the forgot password form where the user will be able to reset their password.

Links to Success Criteria: Having a forgot password ensures that in case the user forgets their password, they can still access their system as long as one of their details is provided. I will send an otp code to their email which they will have to enter to reset their password.

OTP code design



Links to Success Criteria: 2 factor authentication as the user will have to enter a code from their email to access the software. This ensures that even if an unwanted person finds out a person's username and password, they won't be able to access the system as they won't have access to the authorized person's email.

Forget Password design

Here, there are 3 forms – one for entering the OTP code found from the user's email, entering the new password and the confirmation of the new password. There is a change button at the end, to update the new password. Every time, the user forgets their password, a new OTP code will be sent on their email.

Links to Success Criteria – There is a forgot password section which allows a new password to be updated. Firstly, the OTP code will be entered to ensure that the person changing their password is authorized to access the system. This will help ensure that my client doesn't have to worry about any unwanted customers going into the system and changing the login details.

Next step:

- Ask client about where they want their username and password to be stored.
- Ask client for the design logos for the login system. I will need this during the development stage.
- Ask client about the designs of the login system

Q: Where would you like your username and password for your login system to be stored?

Ans: I would like my username and password to be stored in an external file or a database so that it is not visible whilst I am using the program.

Analysis of response: I could store the password and username in a different database and then run SQL query to get the password and username. This way, the username and password won't be written in the code itself and will prevent any third parties from getting access to the user's private information.

I decided not to ask about the logos yet as right now I need to ensure that I get feedback the client about the basic structure of the GUI, such as the rough positioning of the widgets like buttons, text boxes etc.

Login Design Feedback part 1

Since I have to show my designs to my client digitally, I decided to email him my designs and ask for feedback on what could be improved.

Here's the email:

"Hi Parth, I hope you are having a great day. Please find attached my below designs for the login system. I've ensured that all the buttons and text boxes are big enough so that it is easy to read and have used colours where necessary to make the user interface look colourful. Please tell me if there are any key features to be added or if the positioning or the size of the widgets have to be changed."

Kind regards,

Rushabh Dharamshi

This is Parth's reply:

"Hi Rushabh, these designs look really good and I like the use of large buttons and labels which makes it much easier to see. I love the variety of colours that you have used which definitely make the user interface look attractive. I especially like the forgot password design as the colours look well together and the boxes are equally spaced apart. The only changes I would like to see is the main login page, especially the Forgot Password label. I feel like there is a bit too much emphasis on that label and it can become a bit less fancy by making the label smaller. Usually the forgot password text would be small and would go below the login button, but then it would ruin the logo being the same height as the 4 boxes (2 label and 2 textboxes). Perhaps you could enlarge the logo and the 4 boxes whilst making the forgot password label and the click me button small. This way the height of the logo would be equal to the height of the 4 boxes.

As for the OTP code design, I think the design is perfect but the window for the design could be made a bit smaller and make sure the features are in proportion to each other. I just had one question – these designs you have made especially for the buttons and text labels, can this be easily made in Python?"

My reply back to Parth:

"Hi Parth, thank you for the feedback on the designs I created. The designs I am making will be made using Tkinter which is a very friendly and easy package to make user interfaces. Regardless I can make an exact design using other graphical softwares if needed, and then use it as a picture. Using Tkinter, this picture can then function as a button, which would have the functionality as any other button, so there is no need to worry about designing the actual widgets."

Summary of Path's feedback:

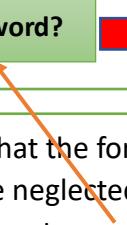
- Could reduce the size of the OTP design
- Could change the positioning as well as the size of the forgot password label
- Could change the positioning as well as the size of the click me button

Login System Logo (Image)

Enter your Username

Enter your Password

Log In

Forgot Password? 

Click Here

This is the design I have changed according to my client. Here I made sure that the forgot password was a bit less fancy, but at the same time I haven't made it too small for it to be neglected by the user. The height of the logo is roughly equivalent from the first label to the forgot password label.

Enter OTP code:

Label

I have scaled down the OTP design whilst making sure the labels, entry and buttons are in proportion to each other and the window itself.

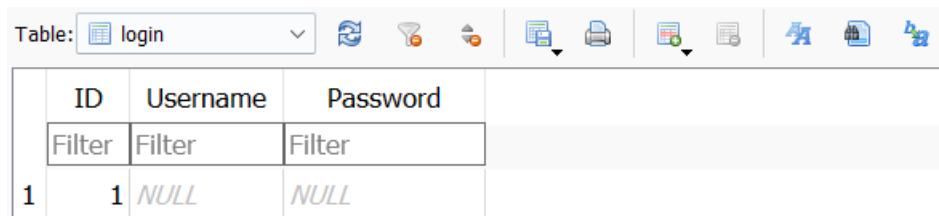
Submit

Button

I've shown Parth the changes and for now it seems to be fine. We will change the design at a later date if necessary.

Database Login Design

My database design will be in SQLite3 where there will be 3 fields – the ID, the username and the password. I will write the SQL query which will make the ID increment by one (autoincrement) each time a new record is added.



The screenshot shows a SQLite database table named 'login'. The table has three columns: 'ID', 'Username', and 'Password'. A single row is present with the value '1' in the 'ID' column, and 'NULL' in both the 'Username' and 'Password' columns. The table interface includes a toolbar with various icons for database management.

ID	Username	Password
Filter	Filter	Filter
1	1	NULL

This is how the database looks like. Here I have 3 fields with the ID acting as the primary key that autoincrements. The ID is incrementing by 1 each time, as you can see the number 1 on the ID column. If I didn't autoincrement the ID, then it would show the value NULL as shown on the Username and Password column where no input has been given.

The name of the Table is called login.

 DB Browser for SQLite - C:\Users\rusha\PycharmProjects\pythonProject5\sms.db

The overall name of the database is called sms – short for Stock Management System.

The database was designed in python where I wrote an SQL query. This is shown in the algorithms section. Here I had to connect my database to python and then I was able to make changes to the database using python and SQL queries.

My client's username and password can be written directly into the database or into the actual program itself. However, since my client asked me to not hard code his username and password, I will write it directly into the database itself which can then be fetched through my program.

Database login Feedback

I've sent my database design to Parth and explained it to him about how it will work. I also asked about how many people in his shop are going to use the login system and if so, will they have different usernames and password. This may sound pretty weird to ask but I just wanted to know about how many records I am likely to have in the database.

Email to Parth:

"Hi Parth, here is my database design below. The database will store your username and password which will not be written in the program. You will have to type your username and password onto the login designs I have made and once you have done so, the program will fetch the stored data from the database and compare whether the username and password you typed matches the ones stored in the database."

Response from Parth:

"Hi Rushabh, the database login design seems to be fine and I am happy with the process of how the login system is going to work. With regards to the number of people using the system, it will most likely be me but other members of the shop may use it in case I am unable to attend on that day. The username and password should be the same for each member, as in case a one of the members of the shop can't remember, they can always ask another member without trying to use the forgot password section.

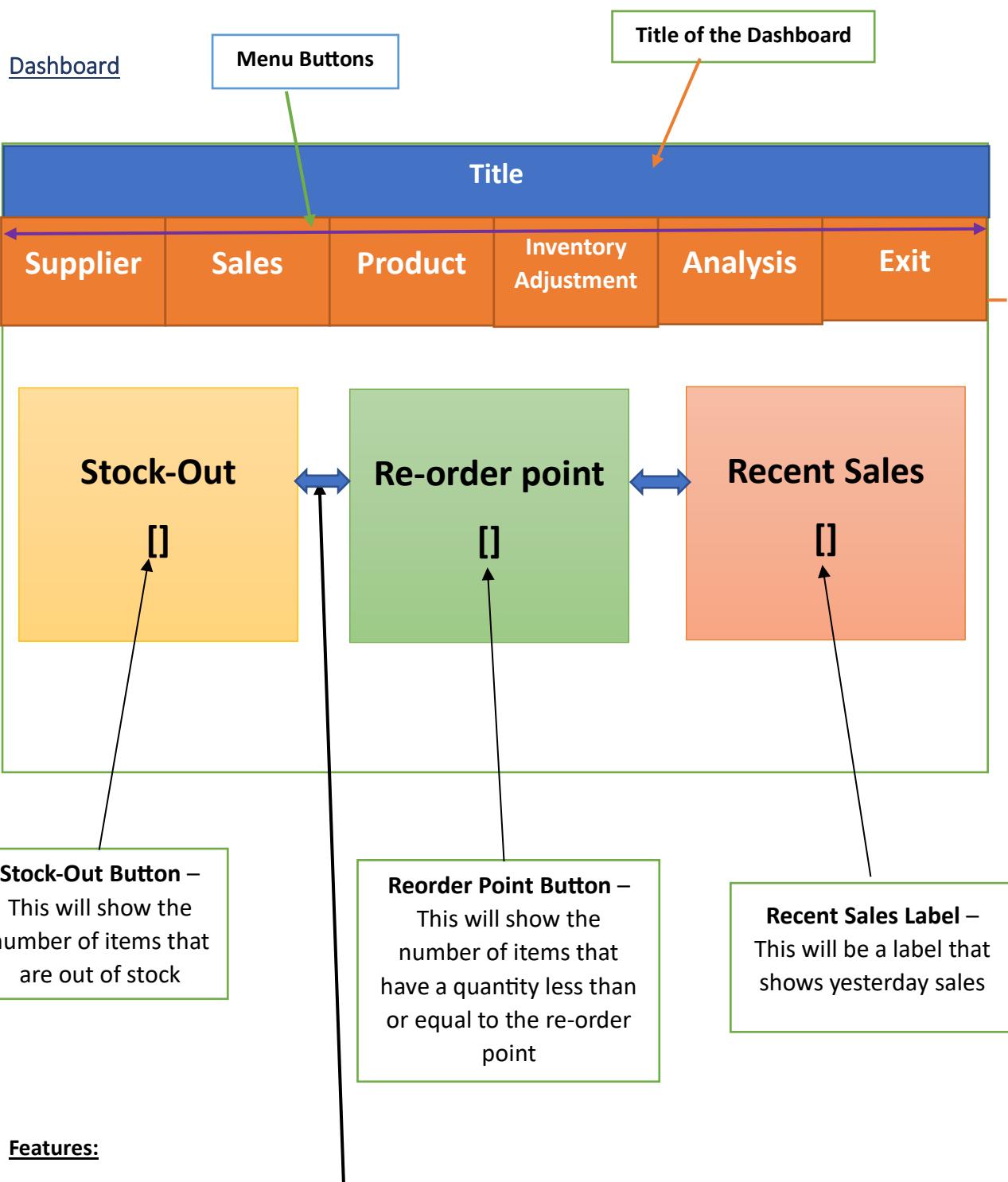
Login table data dictionary

Field	Data Type	Description	Example
ID	Integer	This is a primary key that autoincrements each time a new record is added. This will help to identify each new record using SQL query.	1, 2
Username	String	This is a username that will be used to login.	OneStop20
Password	String	This is used as the first line of verifying that the user is authorized to access the system.	GrocFoodStore
OTP code	Integer	This is used as the second line of verifying that the user is authorized to access the system. This will be sent on the user's email.	647357

I asked Parth about whether I should include the shop's email in the database or whether it can be included in the actual python program itself. He responded by saying that it would be perfectly fine if the shop's email was included in the actual python script itself, as long as the password remains untouched.

I approve of all the designs and changes made to the login system so far: (These aren't final designs for my login system but right now I just need an approval of roughly how the login design and process will work.)



**Features:**

- 2 buttons and 1 label equally spaced and the writing will be big enough so that it is easy to read by the user
- Title Section at the top of the dashboard window
- A horizontal Menu section that has 6 buttons which will produce a new window when clicked
- The menu buttons have the same dimensions so that the dashboard layout looks symmetrical

Dashboard Design Feedback part 1

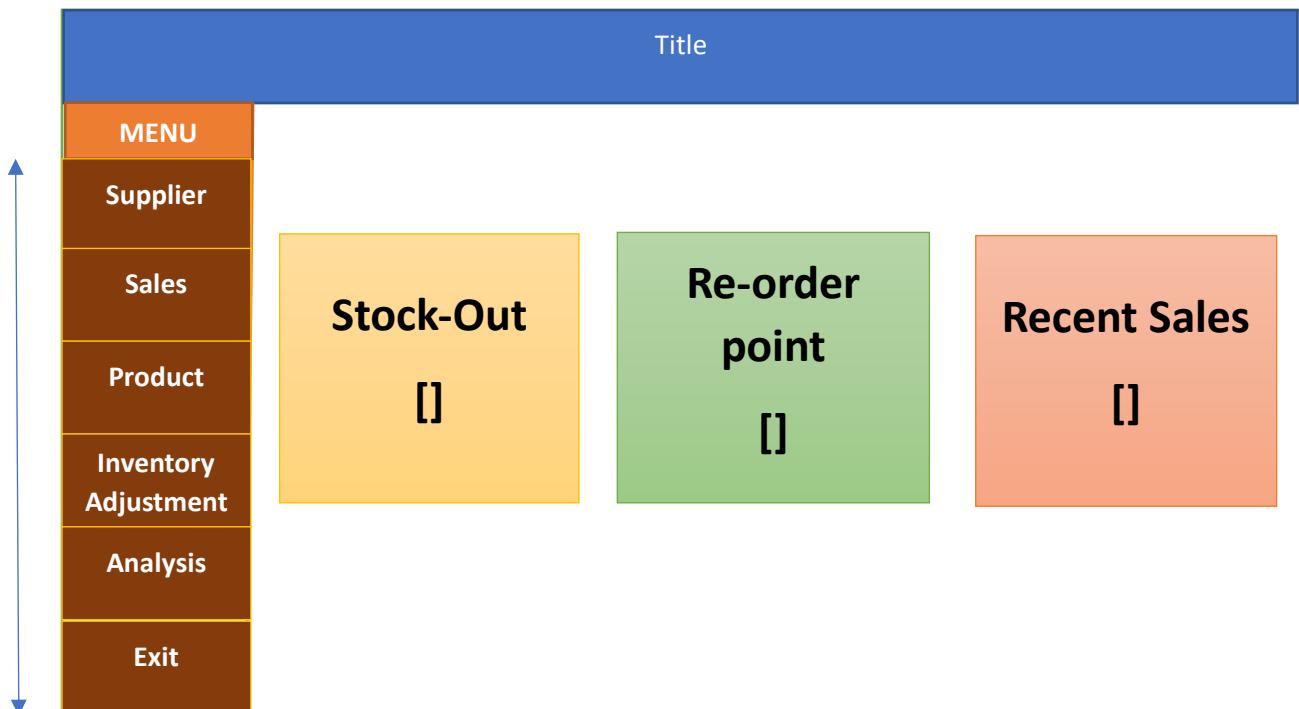
I've sent my main dashboard design to Parth.

Email to Parth:

"Hi Parth, here is the dashboard design. I have added menus on the top in a horizontal display and I have added 2 Buttons and 1 label in the middle of the dashboard window. The Buttons without clicked will show the number of items in stock-out or items that have a quantity less than or equal to the reorder point. I have ensured that the design will make it easy to access other parts of the system by making the text big enough to read.

Response:

"Hi Rushabh, I really like dashboard design especially having the 3 widgets to provide easy and quick information. What I would change is the orientation of the menu buttons so that it is vertical as it will be much easier to use. Also if you could make an orange label saying menu, it will be easier to see what menu buttons there are in the system.



Changes made:

- Orange menu label added
- Menu buttons in a vertical orientation
- I have changed the colour of the menu buttons so that there is a contrast in colours between the orange menu label and the menu buttons. I haven't made each menu button colour different as this would make the dashboard look a bit too fancy.

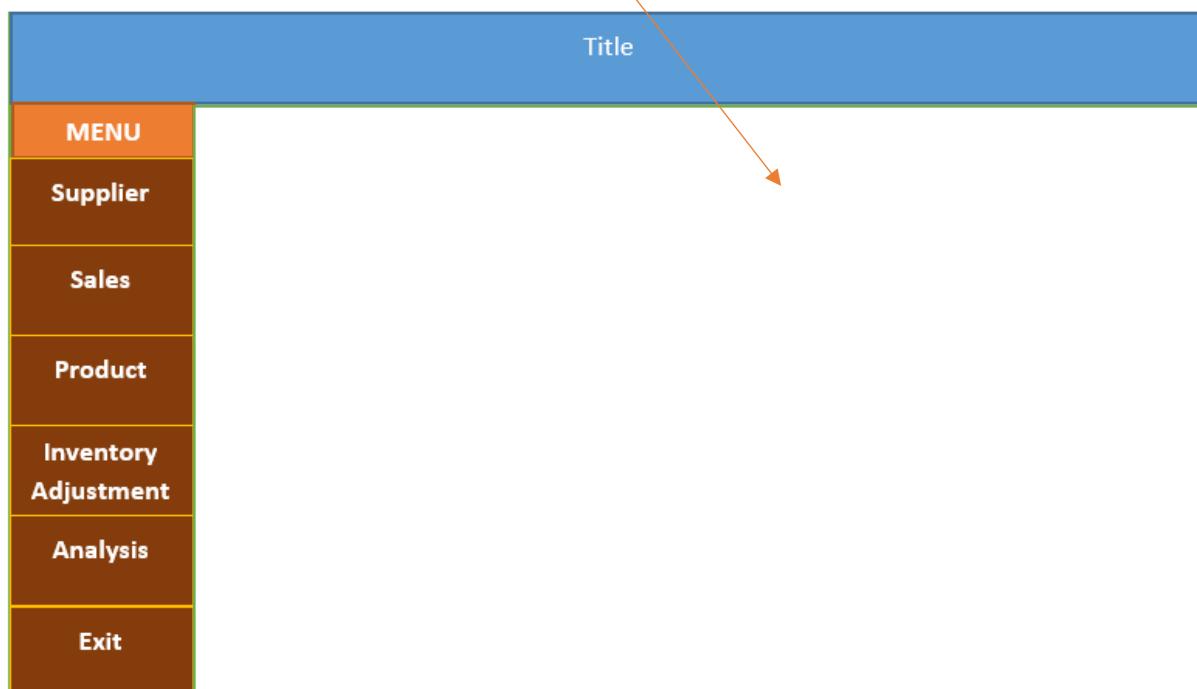
I've shown my client this new dashboard design and he very much likes the new layout of the dashboard with the menus being to the left of the window as well as the colour of the menu buttons.

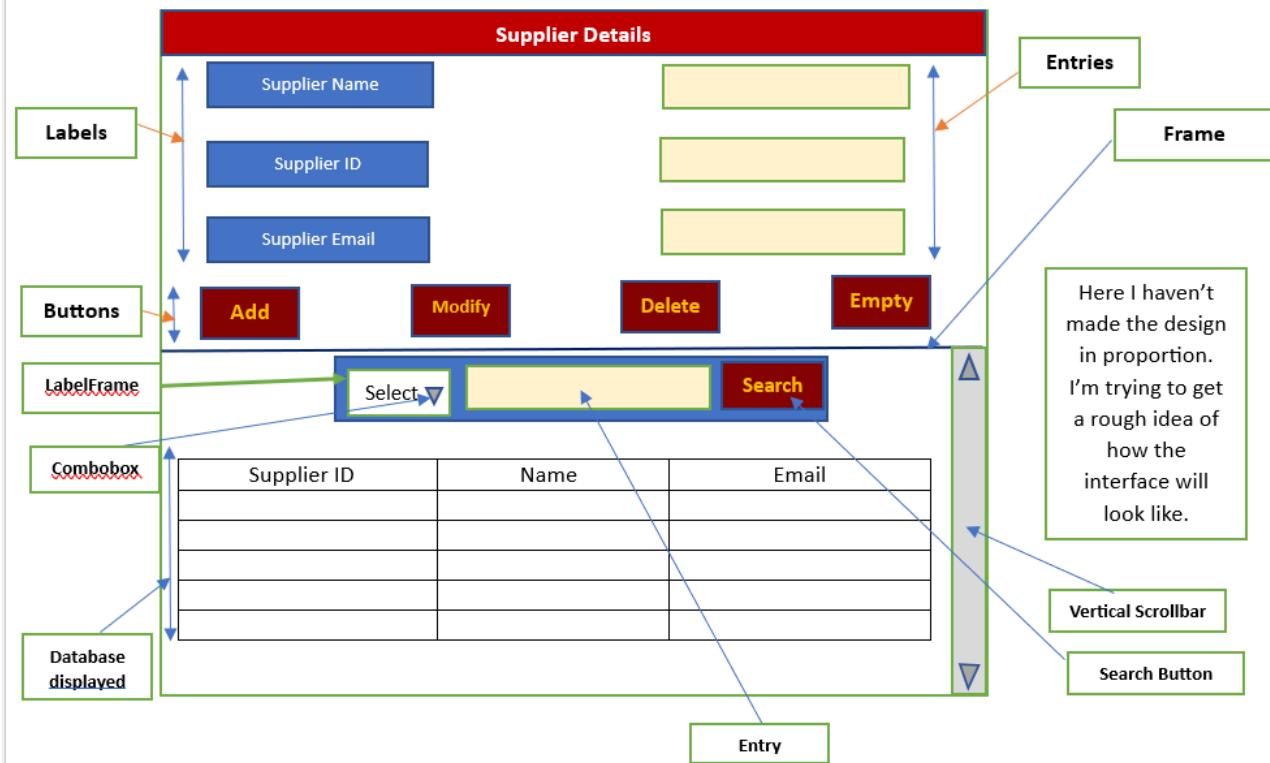
I approve of the changes made to the design of the dashboard:



Supplier design

When the Supplier Button is clicked, a new window will pop out that will cover roughly around 3/4th of the dashboard. The dimensions of the new window will be specified so that the title and menu buttons from the dashboard still remain on the screen when one of the menu buttons are clicked as show below.





Features in my Supplier Section:

- **3 Entry fields corresponded by their Labels:** There is a supplier name entry where the user enters the name of the supplier (firstname). The user inputs the supplier ID which is the primary key used to identify a unique supplier. Here, I haven't autoincremented the supplier ID which is why the user will have to input the ID unlike in my login system where the ID incremented by 1 each time a new record was added. I have also made a supplier email entry to ensure that there is a way to contact the supplier in case more food stocks need to be ordered.
 - **4 Buttons – Add, Modify, Delete and Empty.**

ADD – Once the user inputs the supplier details, pressing the add function will make a new record in the database with the inputs added. This database will also be displayed on the tkinter window itself using a widget called “Treeview”.

Modify - To use the modify button you have to select on a record by clicking on it from the Treeview widget database which will cause your inputs to automatically appear on the entries. After this, you can change the inputs and pressing modify will allow you to update and change the record.

Delete – To delete the record, you will select the record and then press the delete button. This will remove the record from the actual database and then the treeview will also automatically update to delete and remove the record.

Empty – If you want to unselect a record, then you can press the empty function to remove any records from appearing on the Entry forms.

- **Frame** – This is used to split the window into different parts and helps group widgets that work together for a certain process to occur. For example, in the top half of the window, the entries and the buttons work together to cause an output on the database and the treeview.
- **LabelFrame** – This is a container that can contain other widgets as shown in my design above where it contains a combobox, a button and an entry.
- **Combobox** – This is another widget that allows you to select one value out of a set of values that will be defined in the python code. The list of values will be Name, Email and ID. From this, the user will search something depending on what list of values they select. For example, if the user selects Name, then through the entry form he would type a supplier's name that he wants to search. If the user selects ID, then he would type a supplier's ID into the entry form, to search for that supplier. In most cases, the user would type the name since it is much easier than remembering the ID number of a supplier, though the ID is the primary key. Once a name/ID/Email is typed and the search button is clicked, the records will be filtered out showing only the records that fulfil the user's condition.
- **Vertical Scrollbar** – In case, more and more supplier records are added due to more suppliers providing the food store, then not all of the records will be seen through the treeview widget. Therefore, having a vertical scrollbar, allows you to see through all the supplier records without making the treeview take more space on the window.

Supplier Design Feedback part 1

I showed Parth my supplier design and if there are any designs to be changed.

Email to Parth:

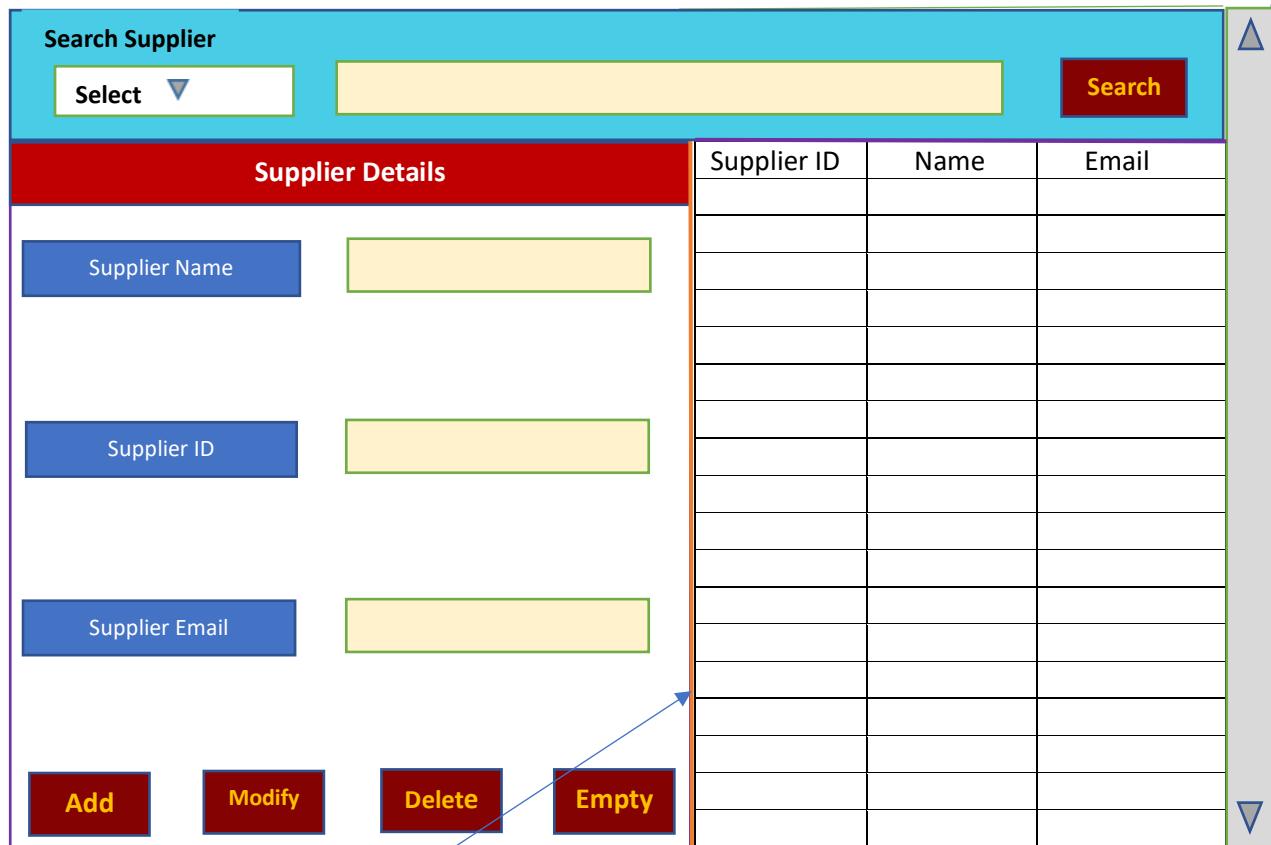
"Hi Parth, here is my initial supplier design. I've made the first half of the window where you can manually input any new suppliers and their details. The new records or any changes made are automatically updated to the Sqlite database itself. To ensure that you don't have to open the database, I have made a widget called Treeview where you can view what's stored in the database in the window itself. You can also search for suppliers and make changes where necessary."

Parth's response:

"Hi Rushabh, the functionality for the supplier section is perfect! The design also looks amazing and I love the contrasting colours between the button and its text. I also like the

fact that I can visually see the database in python itself making it very time-efficient as I wouldn't need to open the SQLite3 application. You've covered all the buttons so that I can add, delete and update any supplier records.

The only thing I would change is probably creating a vertical frame section so that it is easy to see the treeview and the input forms at once. That way, it will be easier to look at many records at once without using the scrollbar as much.



What I changed:

- Vertical frame – The treeview and the supplier details section are both vertically parallel
- Changed the background colour of the LabelFrame to light blue
- Extended the label frame
- Ensured that more records can fit on the window so that the use of the scrollbar is limited
- Label frame text called “Search Supplier” – This helps to direct the user of what they should be able to search in this section.

I've shown the changed design to Parth and he said that the layout is perfect which will help him use the functionalities of the design better. He also likes the light blue background of the labelframe and the fact that it has been elongated to cover the entire width of the window which in turn makes the supplier details and the treeview connected together to fulfil a task.

Supplier data dictionary

Field	Data Type	Description	Example
ID	Integer	This is a supplier's ID that serves as a primary key to identify the supplier. When new suppliers are added, the ID's will be checked to ensure that it is different.	1, 34, 456
Name	String	Name of the supplier. This will be a string variable and is not a primary key as the name of 2 or more suppliers could be same.	John, Tom, Bob
Email	String	Email of the supplier. This will allow the user (my client) to send emails to suppliers.	Tom.David@gmail.com

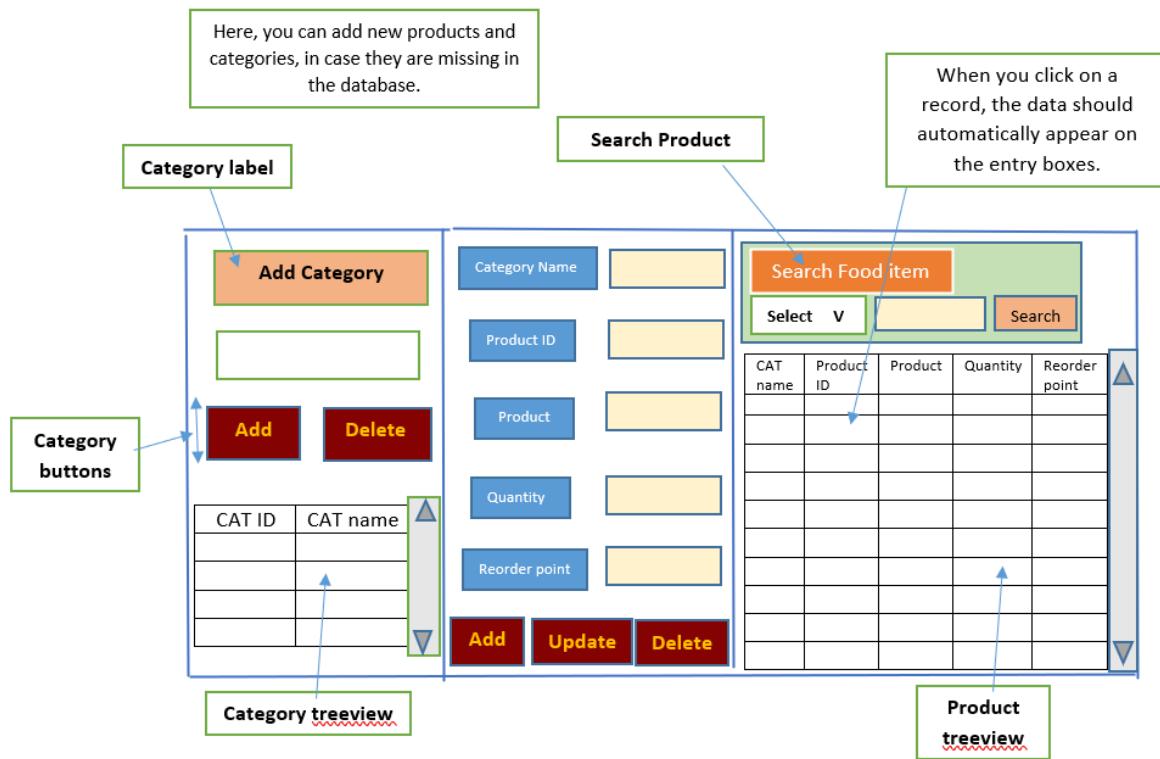
Product

Before I start designing my product window, I need to confirm some things with my client.

Q: For the products section, do you want to have generic food products such as milk, bread or do you want a categorized section where you can have several different food item names for the same category?

Response from Parth: I would like to have a categorized section since my food store has different types of items for the same food.

Analysis: I will create a category section in the products window where each food item will come from a category. For example, different types of bread will all come from the bread category section.



As usual, I've shown my product design (with the annotations) to my client:

Product Feedback Part 1

Email to Parth:

"Hi Parth, here is my product window design. Here, I have split the window into 3 sections with the first section showing different categories that can be added and deleted, the second section showing that you can add, update and delete the database on the right and the third section displaying the product details where you can also search for the products using one of the product information (Product ID or Product). "

Response:

"Hi Rushabh, the design is absolutely amazing. I liked how you utilized the entire window so that it has 3 different functionalities. Regarding the design, the layout is fantastic and all the functionalities have been added. Great Work!"

Category database data dictionary

Field	Data Type	Description	Example
CAT ID	Integer	Each time a new category is added, a new ID will automatically be generated	1, 2, 3, ,4, 5...

		(autoincremented). This will act as a primary key which can be used to identify a certain category.	
CAT name	String	This is the category name that can hold different items of the same category.	Bread – category Items – Baguette, Multigrain etc.

Product database data dictionary

Field	Data Type	Description	Example
CAT name	String	This is the category name that can hold different items of the same category.	Bread – category Items – Baguette, Multigrain etc.
Product ID	Integer	Primary key used to identify products belonging to the same category. This will be autoincremented.	1, 2, 3, 45, 56
Product	String	Name of the product – (not the category)	Category – Pasta Products – Ravioli, Farfalle
Quantity	Integer	This stores the amount of that product available in the shop	34, 56, 100
Reorder point	Integer	This is a value used to determine at what quantity, more of those stocks should be reordered. This may be a rough or calculated value depending on past sales	50, 100

Approval of the client for my product designs: (This will be changed at a further date in my development section if necessary)

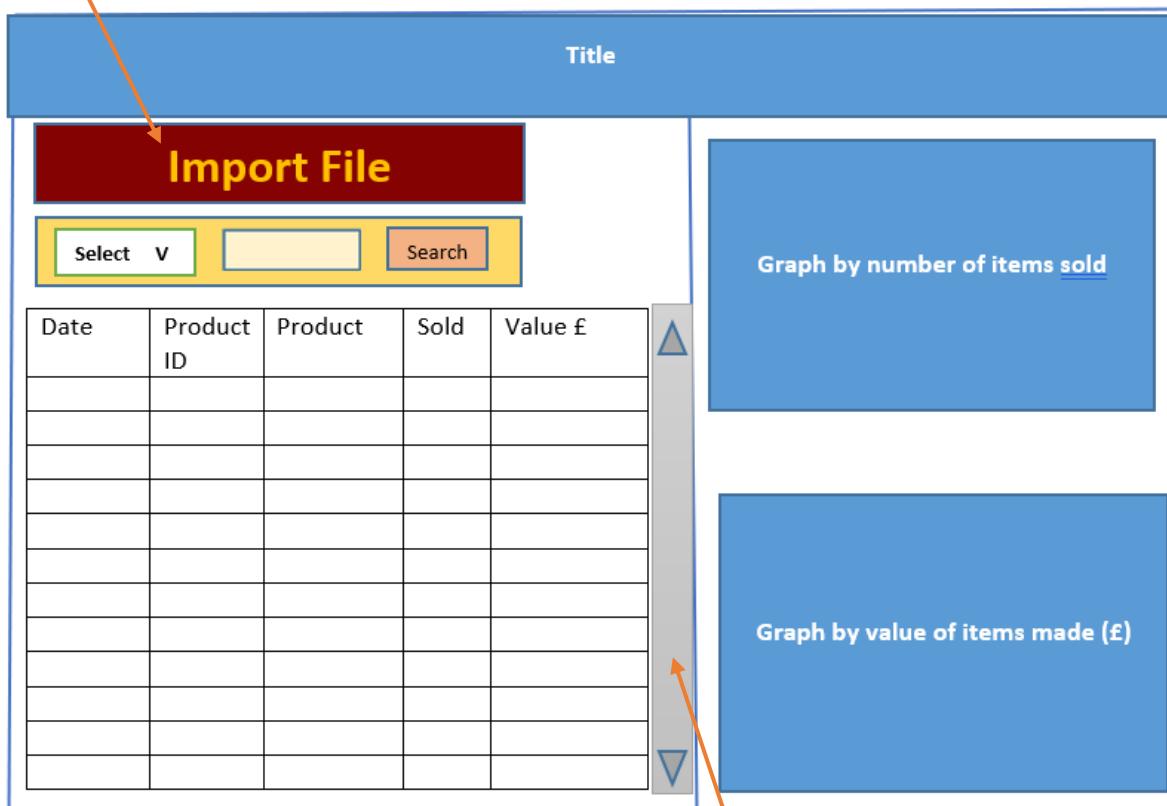


Import file button – this will be used to update the database inventory levels and provides us with sales data from which we can produce the 2 graphs

Centre number: 51427

Candidate number: 6117

Sales Design



When a product is searched, all the other products will be filtered. The treeview will only show the products searched. There will be multiple records showing the same product but for different time periods. The graph will then show the number of items sold and their value throughout time.

Vertical scrollbar – to search through various records

Sales data dictionary

Field	Data type	Description	Example
Date	Integer	Stores the date, month and year which will be needed in my graph	28/01/2022
Product ID	Integer	This is a foreign key used from the product database to identify a certain product	1,2,3

Product	String	name of the product from a certain category – this will be used in my graphs to show the amount of that product sold over a period of time	Kit-kat
Sold	Integer	Stores the number of items sold for a certain product	50
Value	Integer	Stores the value of selling those number of items	£50 – assuming each kit-kat is £1

Here, the date and the product ID can act as a composite primary key in case the user wants to find the sales data for a certain product sold on a certain date.

Sales Feedback Part 1

I showed my sales design with the annotations to Parth.

Email to Parth: “Hi Parth, here is my sales design where 2 time series graphs are displayed for a product searched showing the quantity sold and its value in pounds. The sold section will automatically update the inventory levels in the product table and will show stock-alerts if the quantity is below a certain point.”

Email response: “This sounds great and I feel like the design has got all the adequate widgets to make the whole system function. I like the idea of showing 2 visual graphs that change when a different product is searched. The use of the treeview to show sales records of the product seems perfect as long as time taken to load the graphs is not compensated.”

Analysis of response: The client seems to like my design idea and how the sales window will function. I will just ensure that having too many records will not delay any of the processes that the user may need to carry out.

I approve of the sales design and how it's going to work. (Remember, this isn't the final design for my sales window – I'm just providing an overview of how the sales window should function and what other databases it will affect. – like the product quantity field).

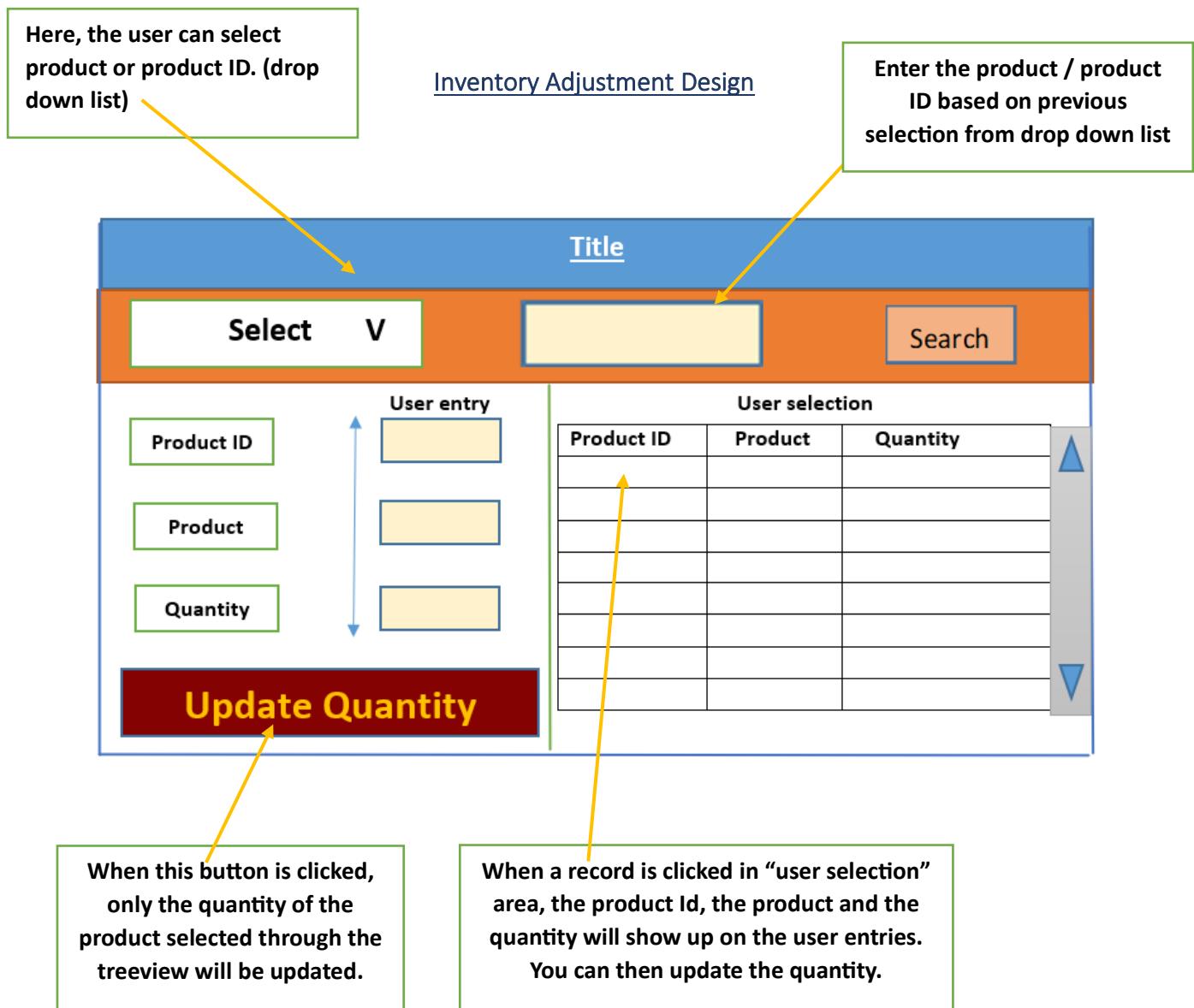


Initial idea of dealing with the import file

Since my client will be using different excel files per day to record the sales data, I decided to create one big excel file where I will combine the excel files that the client will email so that I can directly convert it into a csv file which can be scripted into the python code, rather than manually importing a different csv file each time. Each time a new data is recorded, I will update it into the big excel file and update it into the csv file. This will update all the records where necessary.

Developed idea of dealing with the import file

The problem with my initial idea is that it will display the entire sales data not just for that day, but the day before and so on. This would decrease the inventory levels impractically by using all the sold quantities that have been recorded. I have decided to instead create an import function that will browse through my file where I can select a file that I would like to upload into my software. This way it will properly change the inventory levels just by using the sales data for that day itself and also allows provides flexibility to my client who can just import the saved excel file into the system.



No new fields are created, so I haven't created a data dictionary.

The user will go on the Inventory Adjustments section, only in cases where a manual adjustment is needed and this is typically seen when there are incidents of theft, damage and wastage.

Inventory Adjustments design feedback Part 1

Email to Parth: "Hi Parth, here is my inventory adjustments design section where you can manually change the quantity of the products in case anything happens."

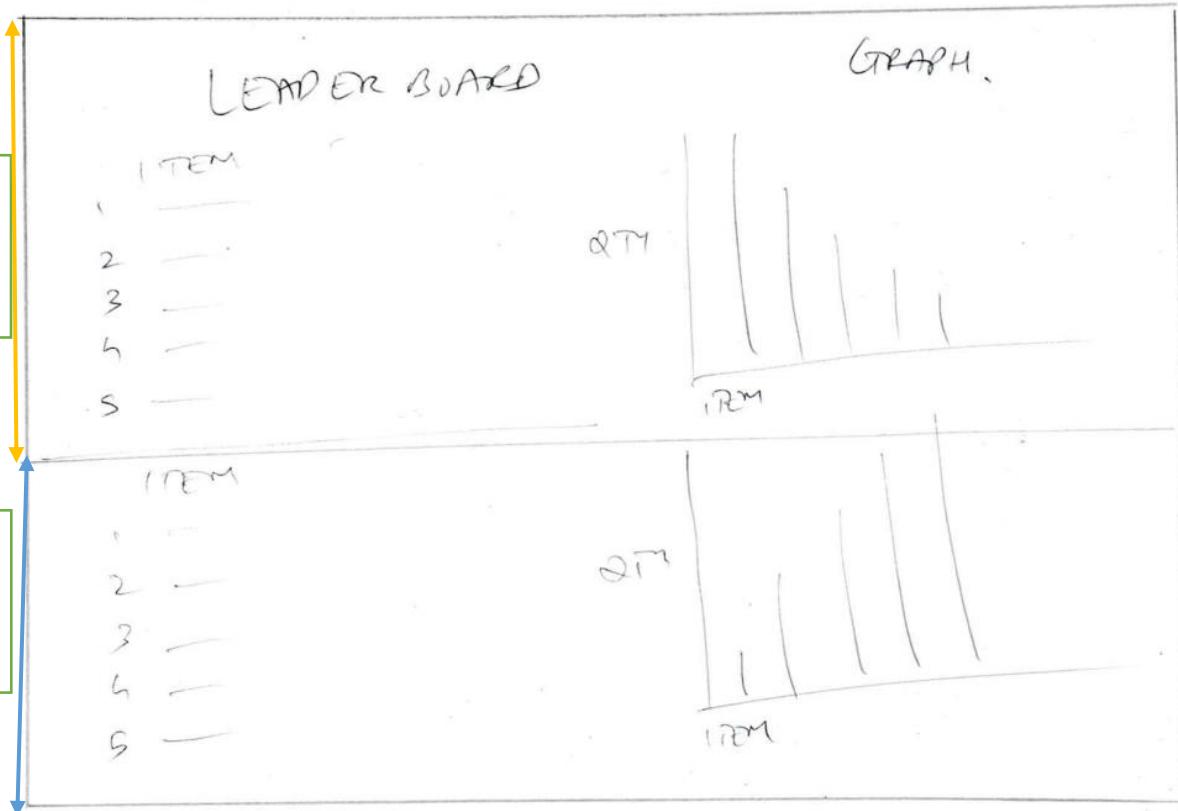
Response from Parth: "Hi Rushabh, the design seems good and once again all the functionalities have been added to ensure that I can change the stock levels of any of my products. The only question I have is that when I am searching for a product, I will not be able to remember its product ID which is why when I am entering the product or characters of the product name will I be able to see all the records of the products that contain the characters sequentially. For example, if I type chips, will the treeview show all the product records that contain the word "chip"?"

Email Back to Parth: "Hi Parth, that is a good question to ask and yes, the treeview should show all the records that have those sequential characters in the product. For example, if you type chips, all the records with the product containing the word "chips" such as Banana chips will show up.

Analysis design

ANALYSIS DESIGN

TOP 5 & LEAST 5 (QTY OVER LAST 7 DAYS)



Here, I have not decided the colours of my bar charts, so I have hand-drawn my design for the analysis section. I have split the window into 2 horizontal sections using a frame. The top half section will show the Top 5 performing products in the form of a leaderboard and a bar chart where you can see the number of products sold within the week and the lower half shows the top 5 least performing products. Using the bar chart will allow my user to compare any products based on the number sold.

Top 5 performing products leaderboard - rank 1 = Best product meaning that most quantity sold

Top 5 least performing products leaderboard – rank 1 = Worst product meaning that least quantity is sold

The leaderboard and the graphs will update so that it shows the data for the past 1 week.

Analysis feedback part 1

Email to Parth:

"Hi Parth, here is my design for the analysis section. Here I have added a bar chart as you wanted and have displayed a leaderboard showing the top 5 worst and best performing products."

Response from Parth:

"Hi Rushabh, I really like the design and having both the leaderboard and the bar chart will help visualize the performance of the products."

Stock-out Dashboard Button design

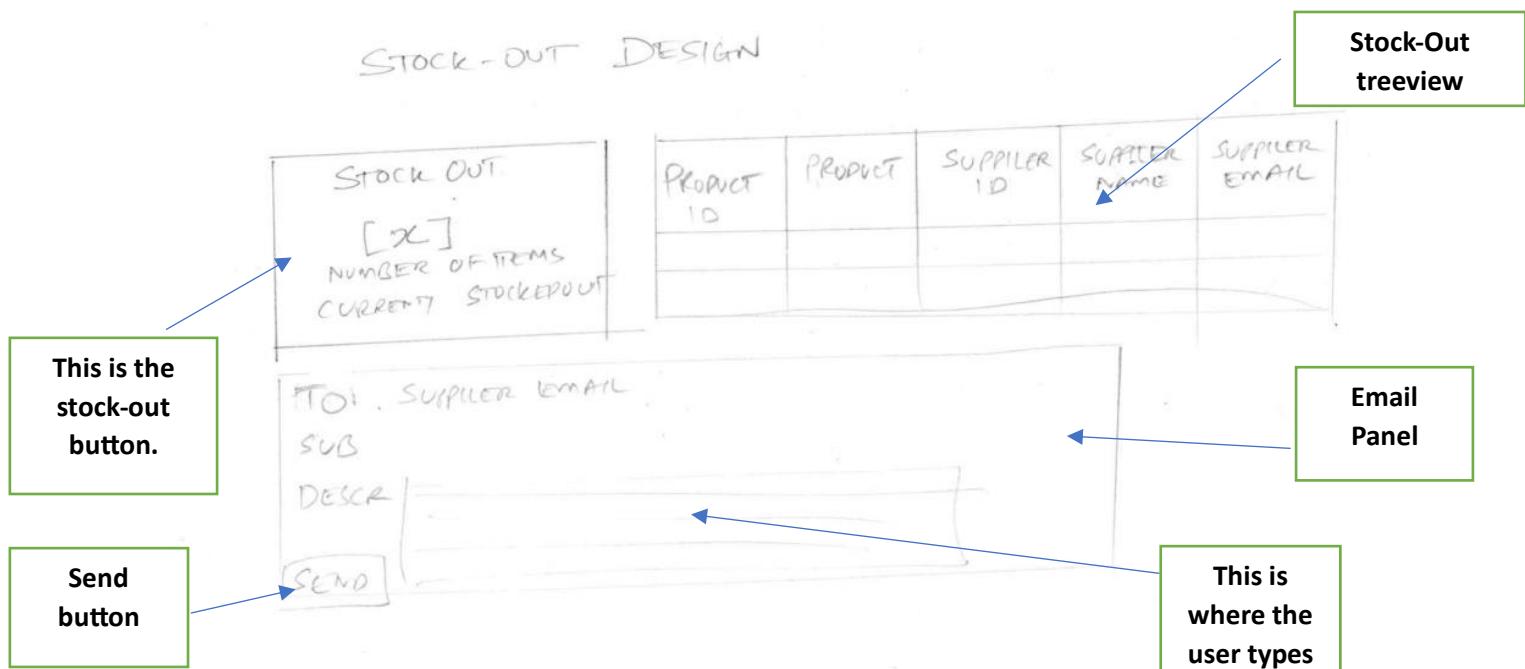
Q: Would you like the email to be automated or sent manually?

(Though it doesn't change my design, I want to ensure that the functionality of the design is appropriate for my client).

Ans: I would like the email to be manual so that I can write a description of what stocks I need and how much. The products sold will vary on a day-to-day basis so I will use my rough judgement and the sales graphs on deciding how many stocks I would need.

Q: Does each product have multiple suppliers?

Ans: No, in my shop, one product will have only 1 supplier supplying it. However, the same supplier can supply multiple items.

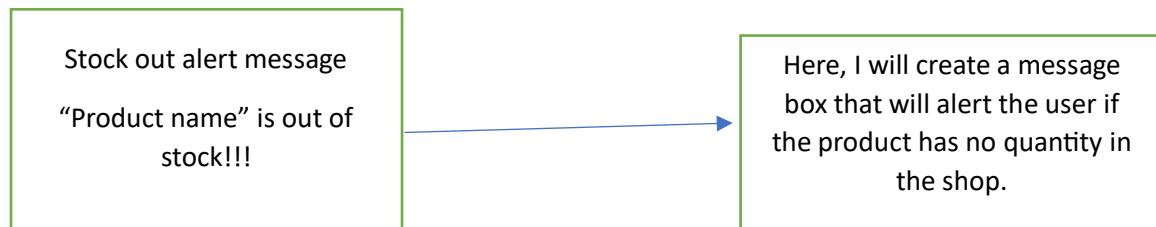


When the stock-out button is clicked, the Stock-Out treeview and the email panel will show up.

Here I have added the 3 supplier fields so that I can link the Supplier details table with this stock-out treeview. This will allow me to get the name of the person who is supplying a

given product to the shop. I haven't added quantity field, as if the product is out of stock, then the product's quantity is obviously 0.

For the supplier table to be linked to this Stock-Out treeview, I will add a product ID field into the supplier database which will be derived from the products database.



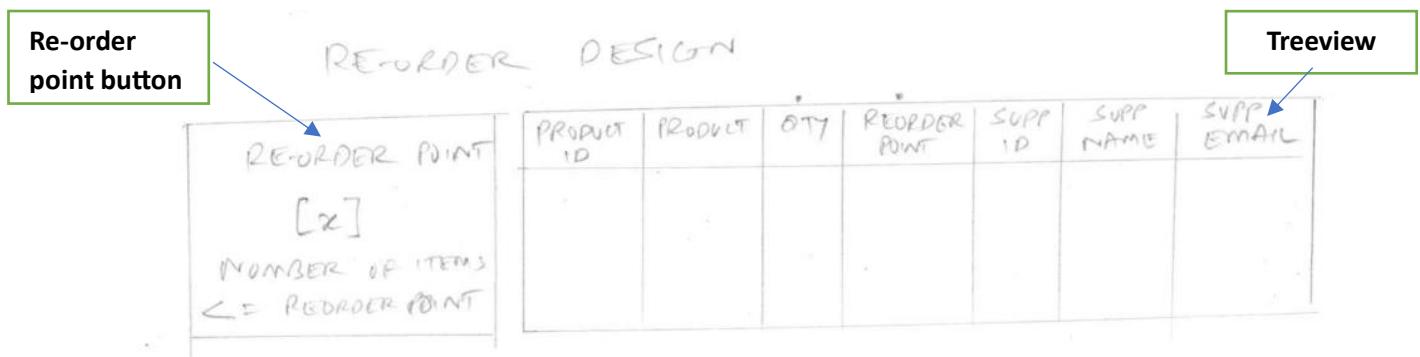
Stock-out Feedback Part 1

I showed my design to Parth

Response from Parth (without emailing him): The design is nice, and all the features have been added such as the email panel to email the suppliers and the table (treeview) that displays the products that are out of stock. I like the use of the alert message that will help identify which product is out of stock. As a user, I want to ensure that this alert message box is impactful, so probably add some extra aesthetics when developing this alert.

Reorder point design

The reorder point design is similar to the stock-out design, as it has the email panel. The only difference is that the reorder point treeview will contain products that have a quantity less than or equal to the reorder point. This means that the products in the stock-out treeview will also appear in the reorder point treeview as well.



Email panel

Centre number: 51427

Candidate number: 6117



Re-order point alert message

Product name needs reordering!!

This will produce an alert message showing the product name where the quantity is less than or equal to the reorder point

Reorder point design Feedback Part 1

I showed this design to Parth who gave similar feedback compared to the Stock-out design.

2.6. Changes to be made in my design

2.6.1. Supplier

As said in my stock-out design, I need to add a product ID field to my supplier table. This means I will have to create a product ID form and an entry label. I will ensure that the product ID cannot be changed as it is primary key from the products table, that will be used as a foreign key in the supplier's table.

The screenshot shows a 'Search Supplier' interface. On the left, there is a 'Supplier Details' section with fields for 'Supplier Name', 'Supplier ID', 'Supplier Email', and 'Product ID'. Below these are buttons for 'Add', 'Modify', 'Delete', and 'Empty'. On the right is a table with columns: Supplier ID, Name, Email, and Product ID. A green box labeled 'Product ID field' points to the 'Product ID' column header. Another green box labeled 'Product ID entry' points to the 'Product ID' field in the 'Supplier Details' section. A green box labeled 'Product ID label' points to the 'Product ID' field in the 'Supplier Details' section. Arrows from the 'Product ID entry' and 'Product ID label' boxes also point to the 'Product ID' field in the 'Supplier Details' section.

Having the Product ID field in the suppliers table will allow me to connect tables from the Stock-out and the Reorder point.

2.6.2. Inventory Adjustment design – No longer needed

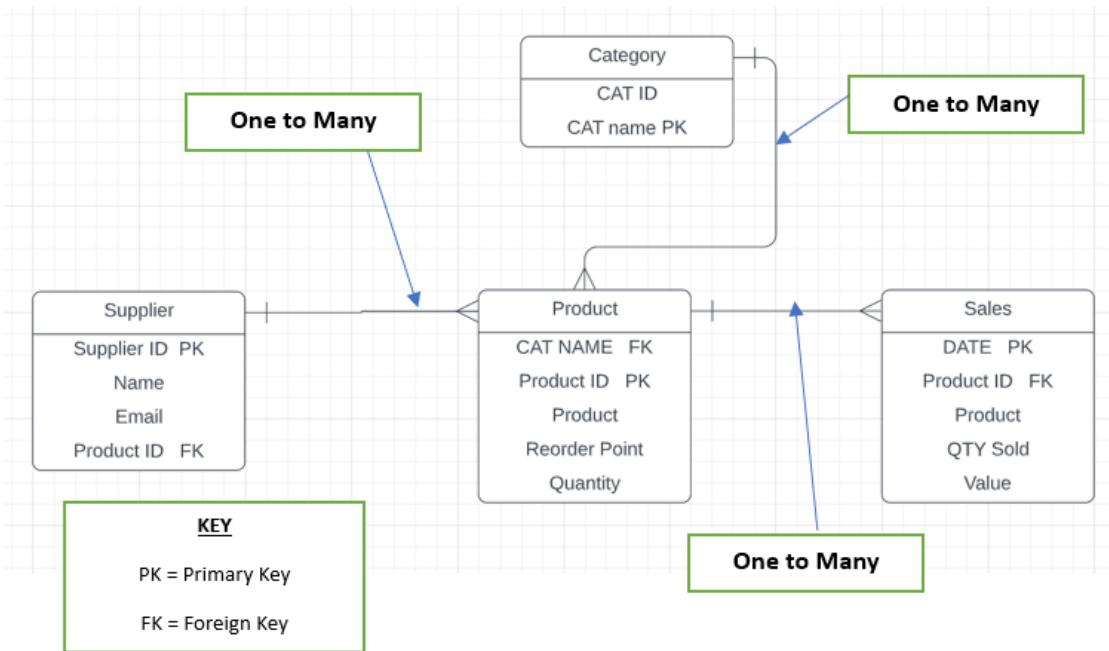
The fields in my Inventory adjustment Product ID, Product and Quantity are contained in the Products design table as well. Since the whole purpose of the Inventory Adjustments is to update stocks in case any incidents happen, this in fact, can be done using the products section as well. Therefore there is no point repeating the same functionality in repeating windows, which is why I am no longer going to have the Inventory Adjustment section as it doesn't have a unique functionality.

I've talked about this with my client and he is happy with the changes I have made as long as no functionalities have been compensated (which they haven't).

Client approval of the changes:

A handwritten signature in black ink, appearing to read "Rushabh".

2.7. Database Entity relationship Diagram



Here I have changed some of the functionalities of the fields. For example, in the Category diagram, I have made the CAT name as the primary key since I am using that field to link the product table with the category field. I have also decided that in the sales record, since some of the records will have the same product ID but at different times, the DATE and Product ID fields will be the composite key. That way, the user can look at the sales data for a particular product at a particular time making that record chosen unique.

Here, my Supplier to Product is a one-to-many relationship, since my client said that a supplier can supply multiple products to the store, but a product cannot be supplied by more than 1 supplier. Though in big retail shops, the relationship for Supplier and Product would be many to many, here it is an exception due to the nature of my client's shop. In the supplier table, the fields Supplier ID and Product ID will become a composite key since as stated before, one supplier can supply multiple products so therefore to find a unique record in the supplier table, I will have to make use of using those 2 fields as my primary key.

My category to Product relationship is one-to-many relationship as a category can consist of many products, but 1 product can only belong to one category. For example, in an Icecream category, there can be different brands of icecream and their flavours.

As you can see on my diagram, there are no Reorder points and Stock-Outs, as there is no table in the database for these. There is a treeview to display the products fulfilling the conditions, but this treeview will contain fields that are derived from other existing tables as shown in the diagram above.

Normalisation of Database

Since I am using a relational database (multiple tables connected together), I need to ensure that my database is designed as efficiently as possible. There are 3 forms of normalization:

1NF – First normal form

- Contains no repeating attributes
- Contains no groups of attributes

How to make it 1NF:

I will ensure that when adding data to my tables like in my Product table, one category will not contain multiple Product IDs, Product, Reorder Points and Quantity in the same row. Instead, there would be different records of data despite having the same category name.

In the sales table, I can get rid of the Product attribute as it's already repeating in the Product table. Instead, I could just use the product ID attribute that is a foreign key in the sales Table to link the 2 tables together and get the product name for that product ID.

Sales
DATE PK
Product ID FK
QTY Sold
Value

This is the changed version which removes redundancy in data so that any inefficiencies and inconsistencies in data are prevented. For example, in case I want to change the name of the product for a given Product ID, it only needs to be updated once in only the products table and not anywhere else.

2NF – Second normal form

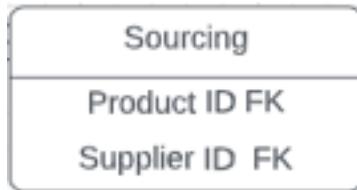
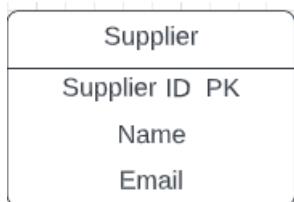
- It's in first normal form.
- Contains no partial dependencies. – This means that one or more attributes depends on a part of the composite key and not all the fields of the composite key.

How to make it 2NF:

Supplier
Supplier ID PK
Name
Email
Product ID FK

Here from the last page, the Supplier ID and the Product ID act as a composite key. Here, the supplier's name and email depend only on the Supplier ID and not the Product ID.

Instead, I could make another table called "Sourcing" where the fields will have the Product ID and the Supplier ID. Since, one product ID will have only one Supplier, therefore a unique matching supplier ID, I can easily derive the suppliers' details from the sourcing table for that given product ID.



Here, in this table, the Product ID will be connected to the Product table and the Supplier ID will be connected to the Supplier table.

3NF – Third normal form

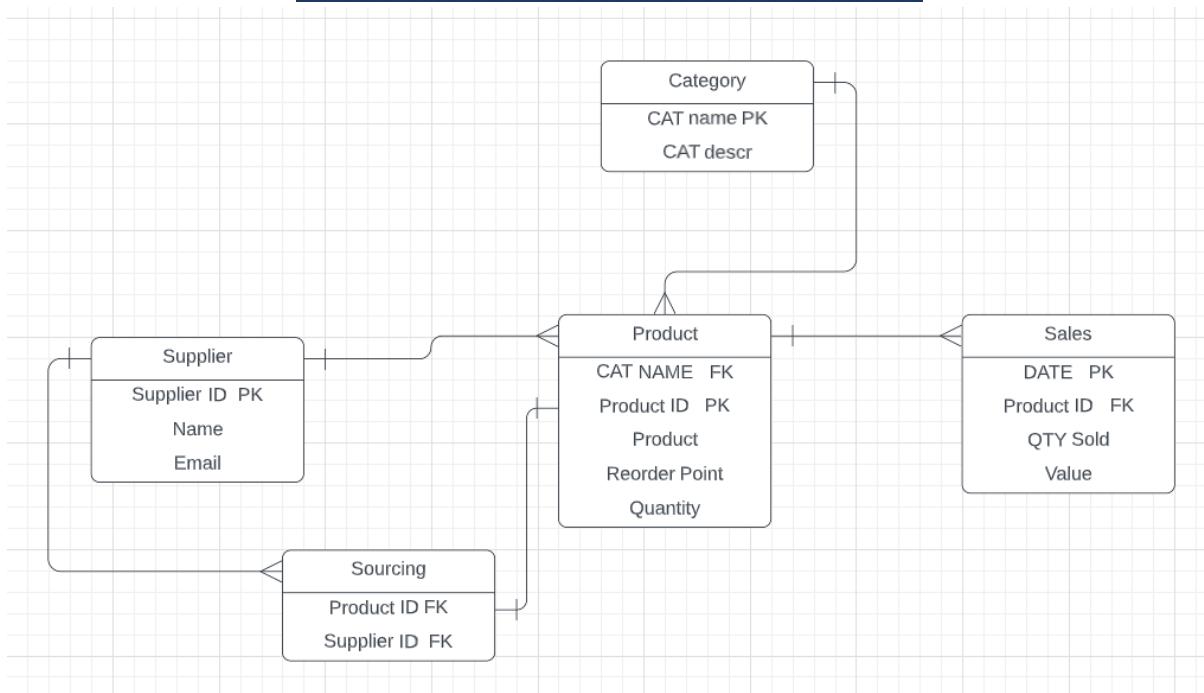
- It's in second normal form
- Contains no non-key dependencies – this means that the value of an attribute should not be dependent on another attribute that is not part of the key.

How to make it 3NF: No changes needed

In the Sales table, the Date and the Product ID form the composite key. Here it is already in 3NF as the Quantity Sold and the Value depend on the Product ID and the time. In my sales table, for that particular date there would only be one record of a particular product as the sales data (Quantity sold and Value) is being accumulated for that unique product and time.

Here, the Value is not dependent on the Quantity sold as I am not calculating it from the Quantity sold field. This sales data is being imported into the system, so calculating the value of that product sold is already done before.

Normalised database entity relationship diagram



Here, I have changed the Category table so that there is a Category description. The Category name is the primary key but it is not a primary key in the Product table as a unique Product ID will only have one Category name.

2.8. Changes to be made in design part 2

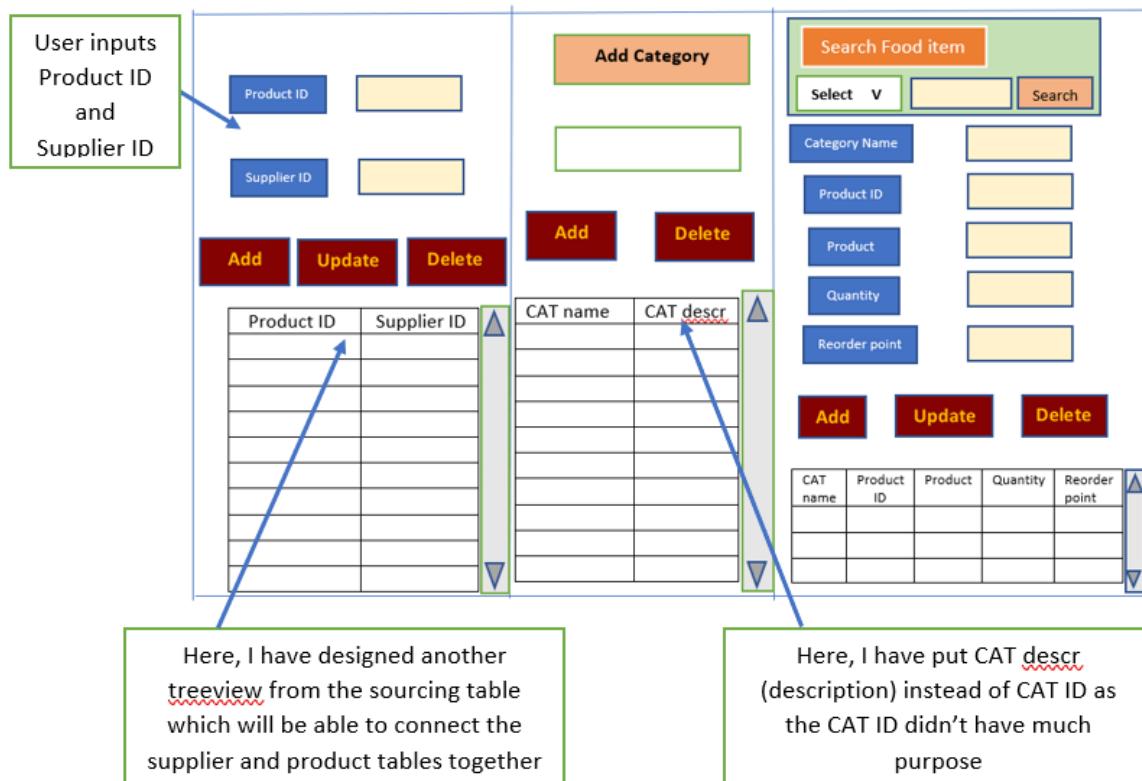
2.8.1. Supplier Design

I have normalized my database so there will no longer be a product ID field in the supplier table. So the supplier table design will now look like this:

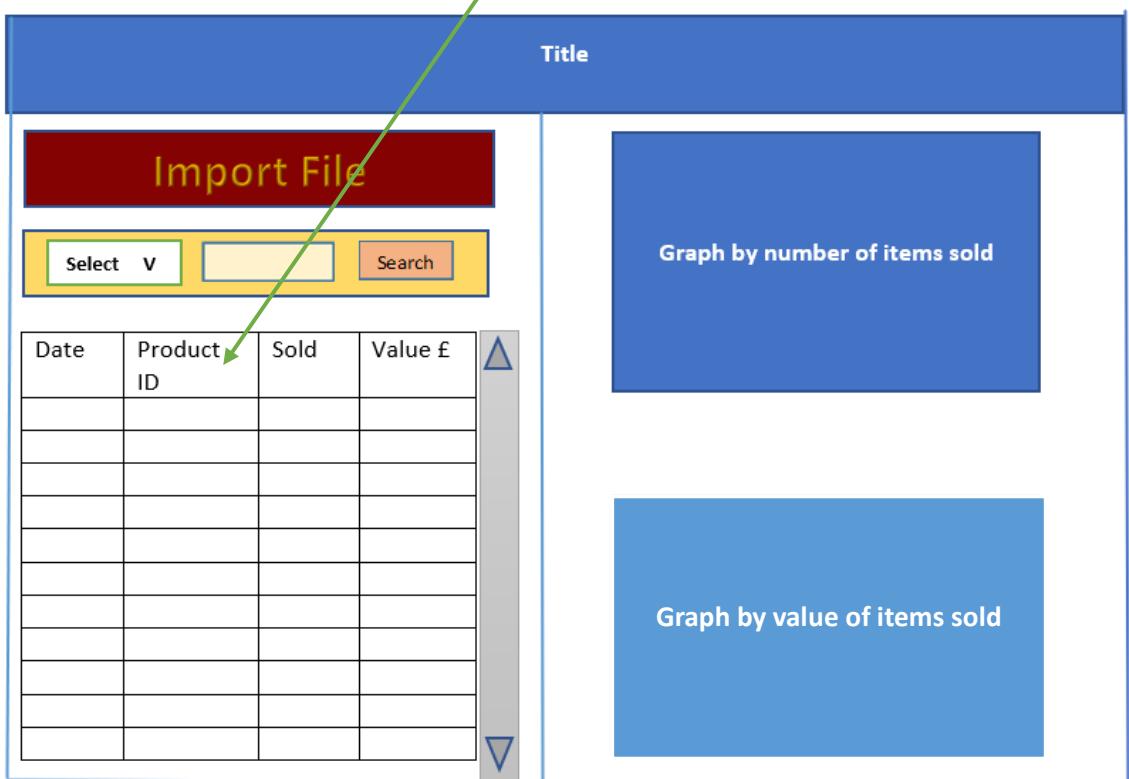
Search Supplier		
Select	<input type="text"/>	Search
Supplier Details		
Supplier Name	<input type="text"/>	Supplier ID Name Email
Supplier ID	<input type="text"/>	
Supplier Email	<input type="text"/>	
Add	Modify	Delete Empty

Here, I have got rid of Product ID

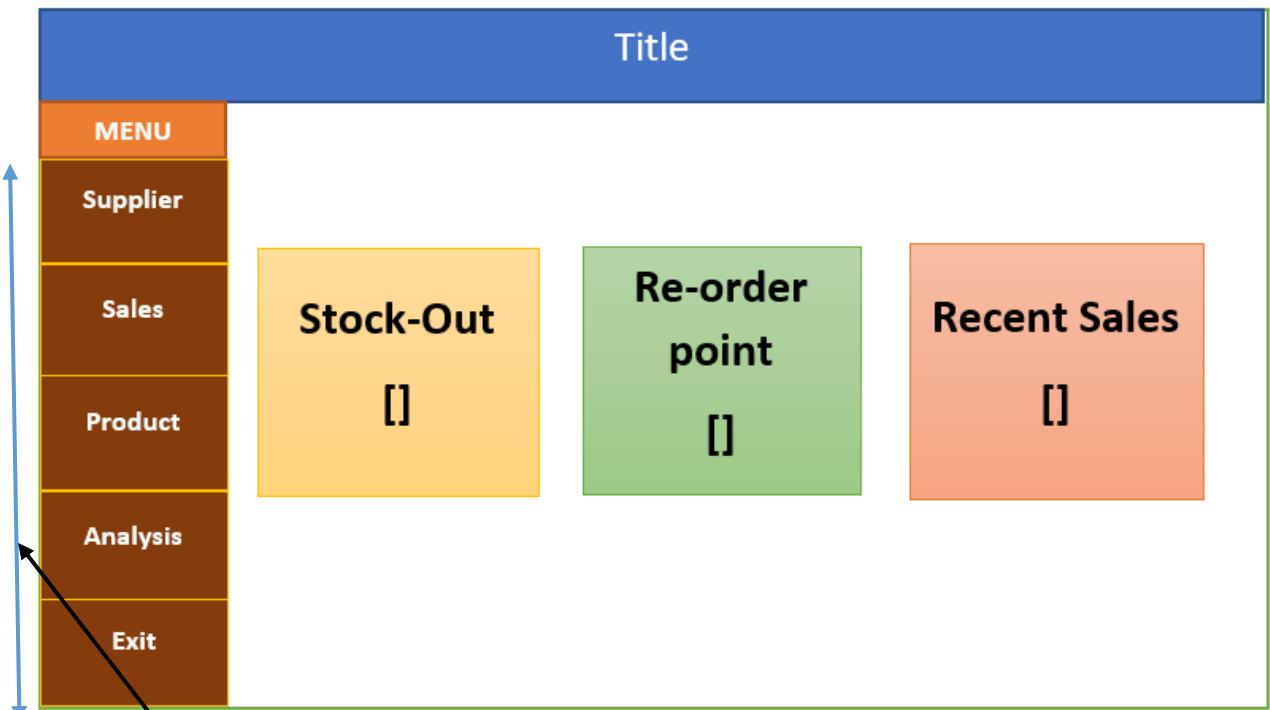
2.8.2. Product design



2.8.3. Sales design



2.8.4. Dashboard design



Here, the section inventory adjustment has been removed. It is no longer in the menu section.

2.9. Final Data Dictionary

2.9.1. Final login Table Dictionary

The data dictionary for the login system is the same as before:

Login table data dictionary

2.9.2. Final supplier Table Dictionary

The data dictionary for the supplier section is the same as before:

Supplier data dictionary

2.9.3. Final Product Table Dictionary

Field	Data type	Description	Example
CAT name	String	This will act as a foreign key to connect the category and product table together.	Milk
Product ID	Integer	Primary key used to identify products belonging to the same category. This will be autoincremented.	1, 2, 3, 45, 56
Product	String	Name of the product – (not the category)	Category – Pasta Products – Ravioli, Farfalle
Reorder Point	Integer	This stores the amount of that product available in the shop	34, 56, 100
Quantity	Integer	This is a value used to determine at what quantity, more of those stocks should be reordered. This may be a rough or calculated value depending on past sales	50, 100

2.9.4. Final Category Table data dictionary

Field	Data type	Description	Example
CAT name	String	This is the primary key used to identify multiple products belonging to that category	Bread – category
CAT descr	String	This is used to describe the category or tell its full form if abbreviated	Dry cat food Wet cat food

2.9.5. Final Sourcing Table data dictionary

Field	Data type	Description	Example
Product ID	Integer	Acts as a foreign key from the Product table.	1,2,3,4
Supplier ID	Integer	Acts as a foreign key from the supplier table.	1,2,3,4

2.9.6. Final Sales Table data dictionary

Field	Data type	Description	Example
Date	Integer	Stores the date, month and year which will be needed in my graph. This is the primary key that is part of the composite key used to identify a certain sales record.	28/01/2022
Product ID	Integer	This is a foreign key used from the product database to identify a certain product. This is part of the composite key used to identify a certain sales record	1,2,3
QTY Sold	integer	Stores the number of items sold for a certain product	50
Value	integer	Stores the value of selling those number of items	£50 – assuming each kitkat is £1

2.10. Algorithms

The main algorithms will help create a relational database and analyze product performances and their sales value for the last 1 week. The main result is that:

Once sales file has been imported the software should:

- **Update the inventory levels (quantity) of the products (product section)**
- **Produce any alerts if necessary (Reorder and Stock-out buttons)**
- **Store different product sales record for that date (Sales Section)**
- **Update any graphs and leaderboards where necessary (Sales and Analysis section)**

For these bullet points to happen, I will need to first link the tables together. Using the Database diagram, I will select the fields used to connect the different tables together.

Connecting the database

```
import sqlite3
conn = sqlite3.connect('sms.db')
cur = conn.cursor()
```

Creating tables

```
function create_dtbse():
    cur.execute("Create Table Supplier
                Supplier_ID integer PRIMARY KEY NOT NULL,
                Name text NOT NULL,
                Email text NOT NULL
                );
    con.commit()

    cur.execute("Create Table login
                ID integer PRIMARY KEY AUTOINCREMENT,
                Username text NOT NULL,
                Password text NOT NULL
                );
    con.commit()

    cur.execute("CREATE Table category
                CAT_NAME text PRIMARY KEY NOT NULL,
                CAT_DESCR text NOT NULL
                );
    con.commit()

    cur.execute("CREATE Table Product
                Product_ID integer PRIMARY KEY AUTOINCREMENT NOT NULL,
                CAT_NAME NOT NULL,
                Product text NOT NULL,
                Reorder_Point integer NOT NULL,
                Quantity integer NOT NULL,
                FOREIGN KEY (CAT_NAME) REFERENCES category (CAT_NAME)
                );
    con.commit()
```

```

    cur.execute("CREATE Table Sourcing
                Product_ID integer NOT NULL,
                Supplier_ID integer NOT NULL,
                foreign key (Product_ID) references Product(Product_ID)
                foreign key (Supplier_ID) references Supplier(Supplier_ID)
                primary key (Product_ID, Supplier_ID)
            );
    con.commit()

    cur.execute("CREATE Table Sales
                Date integer NOT NULL,
                Product_ID integer NOT NULL,
                QTY_sold integer NOT NULL,
                Value integer NOT NULL
                foreign key (Product_ID) references Product(Product_ID)
                primary key (Date, Product_ID)
            );
    con.commit()

endfunction

```

Adding a new record

To add a new record in whichever table, I will need to check that no existing same primary key exists. I've written pseudocode for adding a new record in the Supplier's table.

```

supplier_ID = IntVar() # initialising the Supplier_ID variable
supplier_name = StringVar() # initialising the supplier_name variable
supplier_email = StringVar() # initialising the supplier_email variable

select * from supplier where ID =? (supplier_ID.get())
row = cur.fetchone # fetches next row of data from table
if row != None:
    error alert message
elseif
    cur.execute("Insert into Supplier(ID, Name, Email) values(?, ?, ?)
                supplier_ID.get()
                supplier_name.get()
                supplier_email.get()
                # getting the value stored for these variables

```

This line ensures that the new record for the primary key inserted is new

Error alert message – I will cover this in my validation section

Updating a new record

```

select * from supplier where ID =? (supplier_ID.get())
row = cur.fetchone # fetches next row of data from table
if row == None:
    error alert message
elseif
    cur.execute("Update supplier SET Name = ?, Email = ?, WHERE ID=?"
                supplier_ID.get()
                supplier_name.get()
                supplier_email.get()

```

This line ensures that the record only gets updated if there is an existing primary key

Deleting a new record

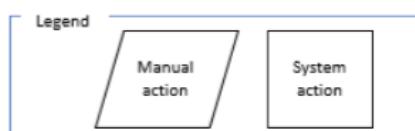
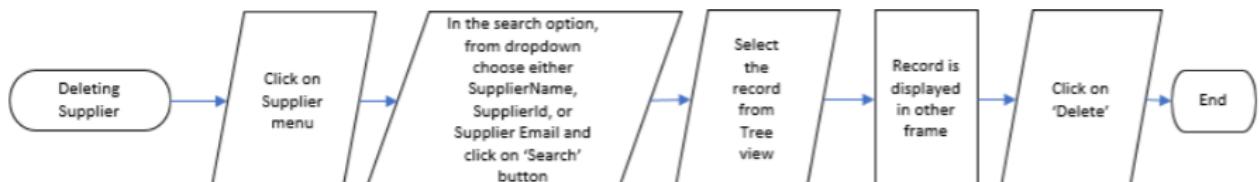
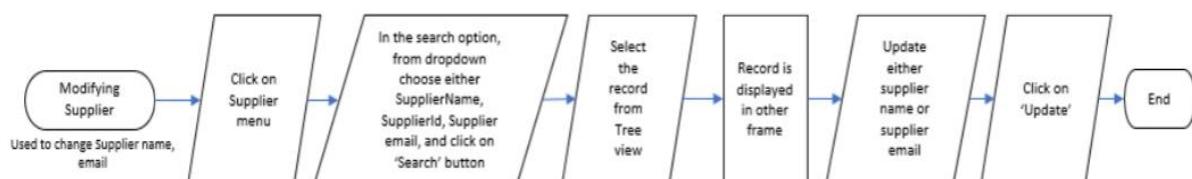
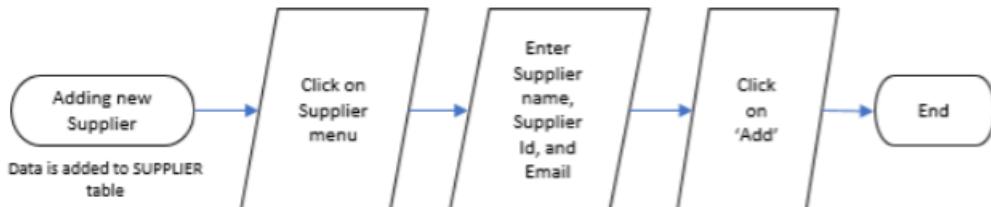
```

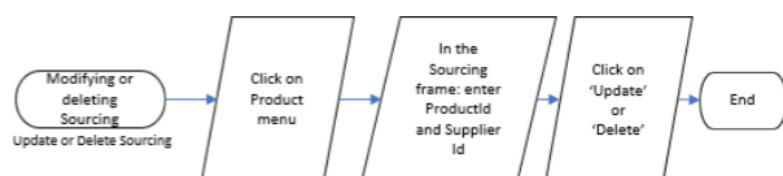
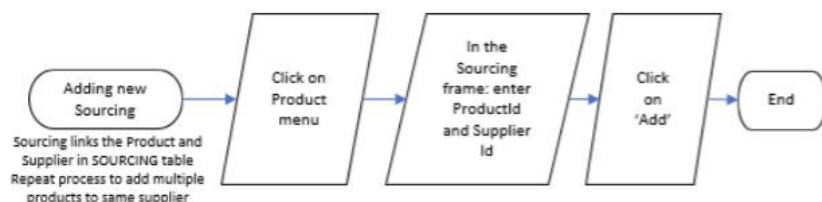
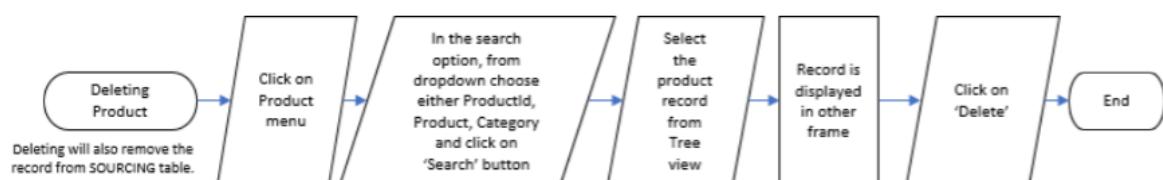
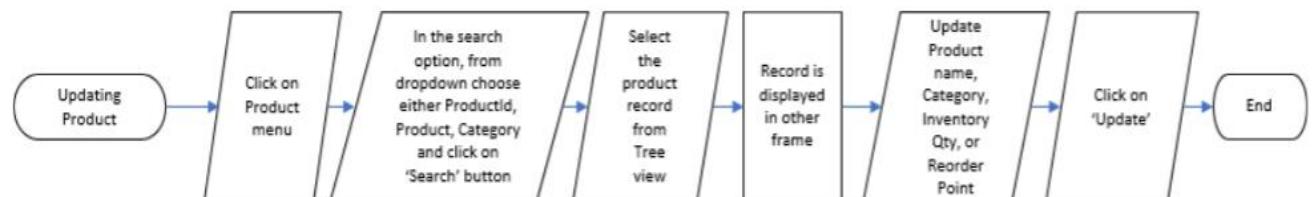
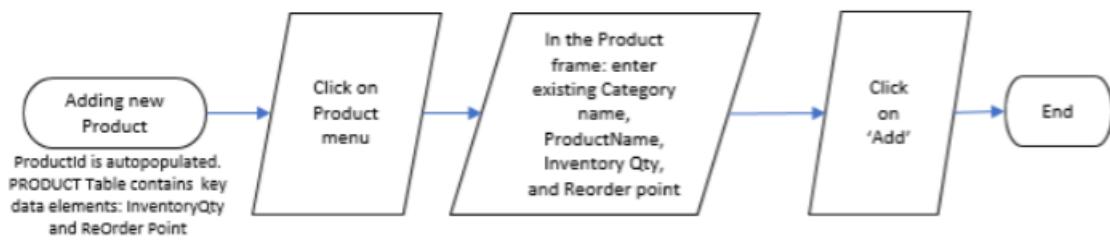
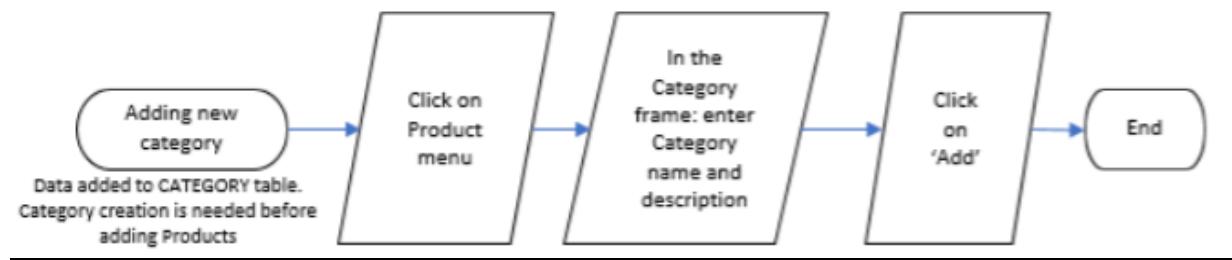
select * from supplier where ID =? (supplier_ID.get())
row = cur.fetchone # fetches next row of data from table
if row == None:
    error alert message
else:
    ask = messagebox confirmation wanting to delete the record
    if ask == True
        cur.execute("Delete from supplier WHERE ID=? " (supplier_ID.get()))
    ...

```

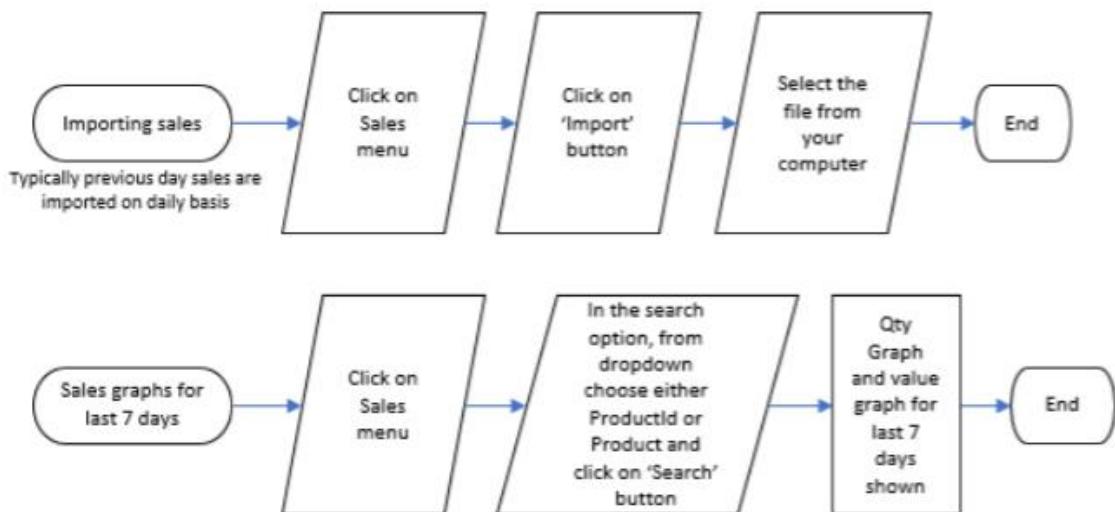
Only deletes existing records

Here, I have put a message box confirmation in case the user accidentally clicks on the delete button

FlowchartsSupplier

Product

Sales



```

import_button = Button(text = "Import file", command = import_csv)
function import_csv
    x = [] # time
    y = [] # quantity
    with open(file link) as csvfile then
        readCSV = csv.reader(csvfile, delimiter = ',')
        for row in readCSV then
            x.append(row[0])
            y.append(int(row[1]))
    plt.bar(x,y)
endfunction

```

This is for making the sales graph

Import file button

Here, this algorithm allows a graph to be formed from reading the csv file. I will need to build this algorithm more so that it can produce a graph for the last seven days when an item is searched in the sales window.

The csv file should be added to the current sales table and should also update the quantity field in the product table.

To get records for the last 7 days:

```

SELECT QTY_sold, Date FROM Sales
WHERE Date >= DATEADD(day,-7, GETDATE())

```

Using this algorithm, I will need to produce a graph based on what the client will search in the search section.

Value (£) graph in Sales Section

```

function value_graph():
    user_input = input("Enter Product_ID")
    select_items_sold = cursor.execute("Select Date, Value FROM Sales
    where Date > (SELECT DATE('now', '-7 day')) and Product_ID =?, (user_input)")
    dates = []
    value = []
    for row in select_items_sold:
        dates.append(row[0])
        value.append(row[1])
    plt.bar(dates, value, color = 'g', width = 0.72, label="Value £")
    plt.xlabel('Date')
    plt.ylabel('Value')
    plt.title("The value of food sold (£)")
    plt.show()

```

Using the algorithm from the previous page, I have used it within the SQL query to fetch the Date and the Sales where the Date is within one week.

I have also created a variable that stores the Product ID that the user has input so that the corresponding Sales in terms of Value graph can be created. This is done by using 'user_input' in the WHERE clause so that the corresponding dates and the Value generated can be stored in 2 separate lists and be used to make the graph.

Sales graph in Sales Section

```

function sales_graph():
    connection = sqlite3.connect(r'sms.db')
    cursor = connection.cursor()
    select_items_soldx = cursor.execute("Select Date, Sold FROM Sales
    WHERE Date > (SELECT DATE('now', '-7 day')) and Product_ID =?", (user_input))
    dates = []
    sold = []
    for row in select_items_soldx:
        dates.append(row[0])
        sold.append(row[1])
    plt.bar(dates, sold, color='c', width=0.72, label="Quantity sold")
    plt.xlabel('Date')
    plt.ylabel('Sold')
    plt.title('Quantity of food sold')
    plt.legend()
    plt.show()

```

The same logic applies here, but instead we are using the field Sold (in terms of Quantity) and the y labels will change so that it shows the quantity of that product sold within one week.

I have still used the variable 'user_input' which will only be input once by the user to show both the Value and Sales (Quantity) graph.

Creating a search button

To create the search button, I will need to initialize 2 variables – one variable that searches a record using a certain field and another variable that searches the actual data. Here, I have used the supplier section as the algorithm.

```
searchuse = StringVar()
searchtxt = StringVar()
Search_box = LabelFrame(text = "Search Supplier")
Search_box.place(x coordinates, y coordinates, width, height)

block = ttk.Combobox(Search_box, searchuse, values = "Select", "Name", "Email", "ID")
block.place(x coordinate, y coordinate, width, height)
```

Here searchuse is a variable in the supplier section that is used to search through records using either Name, Email or ID. The option to select a value to search through is done using the Tkinter Combobox.

The Combobox is within the Search box and its coordinates can be specified.

The algorithm below is an SQL query that is used to get the field of what was used to search (Name, Email or ID) and also gets the value searched. Here, I have used the LIKE operator that is part of my where clause to find any records that match the specific pattern. Here my pattern is that if the user searches a fraction of the value, then all records that contain what the user searched in any order will be shown. I have specified this using the percentage symbols on both sides of "search.get() " method.

```
cur.execute("select * from supplier where + searchuse.get() + LIKE % + searchtxt.get() + %")
```

Any other records that do not fit with the criteria are temporarily removed but are still stored in the database.

Treeview

```
supplyTable = ttk.Treeview(columns = (ID, Name, Email)
supplyTable.heading("ID")
supplyTable.heading("Name")
supplyTable.heading("Email")
supplyTable.column(ID, (width can be specified here), (stretch (takes in a boolean value)))
supplyTable.column(Name, (width can be specified here), (stretch (takes in a boolean value)))
supplyTable.column(Email, (width can be specified here), (stretch (takes in a boolean value)))
```

This algorithm is used to introduce the treeview that will display the supplier table in the supplier window itself. That way, it is easier for the client to refer to this rather than going on the actual SQLite application.

Temporarily removing records from Treeview that don't match the criteria

```

cur.execute("select * from supplier where + searchuse.get() + LIKE % + searchtxt.get() + %")
rows=cur.fetchall()
if length of rows does not equal to 0 then
    supplyTable.delete(*SupplyTable.getChildren())
    for row in rows:
        insert in SupplyTable (values = row)

```

Here once the entire data stored in the treeview has been deleted, the new values that match the criteria are added on different rows using the for loop. Here, the algorithm is checking that if the table is not empty, then it will temporarily delete any existing data on the treeview and will show the records that match the criteria.

Importing sales file algorithm

The sales file will be imported to the system showing the cumulative sales and value for a certain product at a particular date.

```

function select_file:
    file = open("sales.csv")
    contents = csv.reader(file)
    insert_records = Insert into sales(Date, Product_ID, Sold, Value) VALUES(?, ?, ?, ?)"
    cursor.executemany(insert_records, contents)
    select_all = "Select * from Sales where Date > Date - 7
    rows = cursor.execute(select_all).fetchall

```

Here, I have created an object called file that opens the csv file. When this file is opened, we can then read it using csv.reader that will iterate through the csv lines. We can then add this to the sales table by using the sql command “insert” to add sales records in the given order as shown.

The csv file is imported to the sales table by specifying 2 parameters – the variable named “insert_records” which stores the SQL command to add the records into the table and the contents variable that is an object used to read the csv file.

The last 2 lines enable the treeview to show only the Sales records for the last seven days. Having a treeview showing all the sales records will not be of much use, as it is very unlikely that the user will look at hundreds of sale data.

The variable ‘row’ stores the SQL command of fetching all the records. This is not just one record, but all the records that fulfil the criteria of the records being within one week.

```

for r in rows:
    Sales_Table.insert('', End, values = r)

```

Since there are multiple records, I have iterated through using the for loop and for each value of r, it is inserted into the sales table.

Here my client would need the excel file to have the Date, Product_ID, sold (in terms of quantity) and value in the correct order.

2023-02-03	1	10	100
2023-02-03	2	10	100
2023-02-03	3	5	100
2023-02-03	4	25	100

For example, here the data in the excel spreadsheet has been written in the correct order. The dates are allowed to repeat as it part of the composite key along with the product ID.

Updating stock levels

This function to update the stock levels for products will only happen once the user has imported the file into the sales table. This is because the update stock function below relies on using sql queries to fetch today's records from the sales table.

```

function updateStock()
    sales_file = open(Sales.csv)
    select_today = "SELECT Sold FROM Sales where Date == (SELECT(DATE('now')) ORDER BY rowid"
    rows = cursor.execute(select_today).fetchall()
    select_product_quantity = "SELECT Quantity FROM Product"
    current_stock = cursor.execute(select_product_quantity).fetchall()
    current_stock_lst = []
    sales_today_lst = []
    for r in rows:
        sales_today_lst.append(r)
    for x in current_stock:
        current_stock_lst.append(x)

    final_updated_stock = []

    updated_stock = current_stock_lst - sales_today_lst
    for r in updated_stock:
        if r < 0:
            r = 0
        final_updated_stock.append(r)

    for i, value in enumerate(final_updated_stock)
        cursor.execute("Update Product SET Quantity = ? WHERE rowid = ?", (value, i+1))

```

Here, the function uses SQL query to fetch today's records using the date field and is ordered through the row id. Here, the row id or the product ID could be used to order the sales records as they both store the same value and will be auto-incremented.

The function also fetches all the current levels of the stock (Quantity) that will be in the Products table. Since they are stored in a variable, I have created 2 lists where the variables storing today's sales records and the current levels of the stocks will be iterated and added to the corresponding lists. By having 2 lists, I will be able to get the same index of the 2 lists showing the current levels of that product (for example: index 1 will show the current stock for product ID 2) and the quantity of that product sold (i.e. index 1 will show the quantity of product ID 2 sold). This will allow me to directly subtract the 2 lists together to get the updated stock for each product.

The updated stock for each product will be stored in another list called final_updated_stock. This will be the result of the 2 lists mentioned previously subtracted together.

For testing purposes, I have ensured that even if one of the values in the list is negative (which will not happen as you can't sell more than the current quantity of the product), it will be turned to 0.

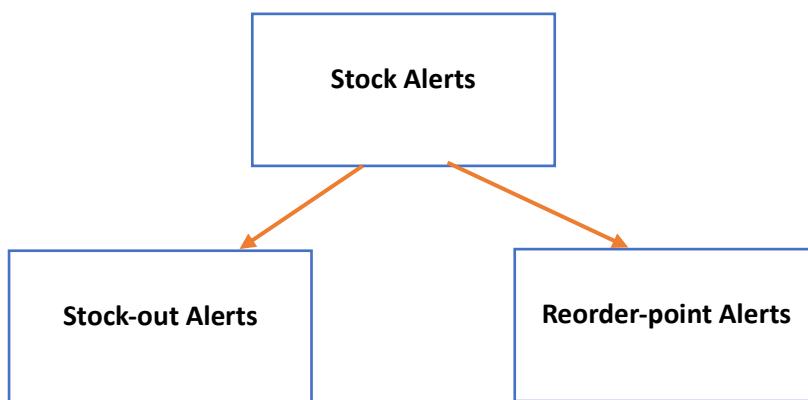
To update the products through SQL, I have iterated through the list showing the updated stock levels and have used the row id to get the unique record from the product table.

Here the value stores the actual contents of the list and the variable “i” is used to get the index of the list. I have ensured that the quantity is set to the value of the variable ‘value’ and that the row id is $i + 1$. The row id has to be one more than the value of i as SQLite starts with row id 1 and not 0. This contrasts with a list that starts with index 0.

Stock Alerts

I will create a function called update_label that will cause any alerts to be formed based on 2 factors:

- If quantity of product < reorder point
- Quantity of product = 0 = stock out



```

stockout = Button()
reorder_point = Button()
function update_lbl()
    cur.execute("select * from Product where Quantity < reorder_point")
    reorder_point = cur.fetchall()
    reorder_point.config(text = str(len(reorder_point)))

    cur.execute("select * from Product where Quantity = 0")
    stock_out = cur.fetchall()
    stockout.config(text = str(len(stock_out)))
  
```

Here, I have created 2 buttons (buttons not labels as the user will click on these buttons to see the products that are stocked out or are below the reorder-point) that can be used to overwrite the content of what is displayed on the button.

Recent Sales

I have used SQL query to fetch all the records that are then stored in a variable. The variable holds the records of all the products from the product table that fulfil the condition. Since the buttons should show the number of items out of stock and below the reorder point, I have ensured that the button text shows the number / length of records fulfilling the conditions using the len() function.

```

sales_recent = Label()
cur.execute("Select total(Value) FROM Sales where Date = (select MAX(Date) From Sales)")
recent_sales = cur.fetchall()
sales_recent.config(text = str(recent_sales))

```

Here, I have created a recent sales label that shows the total sales in (£) made from the most recent date stored in the Sales table. The label text is then updated to show the value of the recent sales.

Top 5 performing products Bar Chart – Analysis section

```

function top_5_products()
    cursor.execute("Select Product_ID, SUM(Sold) AS total_revenue from Sales WHERE Date > (SELECT(DATE('now',-7)) GROUP by Product_ID ORDER BY SUM(sold))")
    show_top_5 = cursor.fetchall()
    top_5 = (show_top_5[:-5])
    product_ID = []
    sold = []
    for row in top_5:
        product_ID.append(row[0])
        sold.append(row[1])
    plt.bar(product_ID, sold, color = 'c', width = 0.72, label = "Quantity sold")
    plt.xlabel('Product_ID')
    plt.ylabel('Sold')
    plt.title('Top 5 performing products')
    plt.legend()
    plt.show()

```

This function here fetches the all the Product ID's and all of its total sales for one week. This is done using the sum function that adds up all the quantity of the stocks sold.

I have then extracted the top 5 products by getting the last 5 rows that were stored in the variable show_top_5.

I have created 2 lists Product ID and Sold so that the graphs can be plotted as respective x and y coordinates.

Matplotlib is used here to create the bar chart.

Top 5 least performing Products Bar Chart – Analysis section

```

function worst_5_products()
    cursor.execute("Select Product_ID, SUM(Sold) AS total_revenue from Sales WHERE Date > (SELECT(DATE('now',-7)) GROUP by Product_ID ORDER BY SUM(sold))")
    show_worst_5 = cursor.fetchall()
    worst_5 = (show_worst_5[:5])
    product_ID = []
    sold = []
    for row in worst_5:
        product_ID.append(row[0])
        sold.append(row[1])
    plt.bar(product_ID, sold, color = 'c', width = 0.72, label = "Quantity sold")
    plt.xlabel('Product_ID')
    plt.ylabel('Sold')
    plt.title('Worst 5 performing products')
    plt.legend()
    plt.show()

```

Here, this function works similarly to the previous one, however here we are selecting the first 5 elements as the Product ID is arranged from the least sales to the highest sales. This will give us the worst 5 performing products.

Leaderboard showing top 5 performing products (Treeview) – Analysis section

```

function top_5_products_leaderboard():
    top_5 = "SELECT Sales.Product_ID, Product.Product, SUM(Sold)
    AS total_revenue from Sales INNER JOIN Product ON Product.Product_ID = Sales.Product_ID
    WHERE Date > (SELECT DATE('now', '-7 day')) GROUP BY Sales.Product_ID ORDER BY SUM(Sold) DESC LIMIT 5"
    rows = cursor.execute(top_5).fetchall()
    for r in rows:
        self.best_5_table.insert('', END, values=r)
    ...

```

This function shows the Product ID, Product and Quantity Sold for the top 5 performing products. Here, I have joined the Sales and Product table using the common field of Product ID to fetch the name of the product using SQL. These are the stored in a variable that is iterated and inserted into the treeview.

Leaderboard showing worst 5 performing products (Treeview) – Analysis section

```

function display_worst_5_products_leaderboard(self):
    show_worst_5 = "SELECT Sales.Product_ID, Product.Product, SUM(Sold) AS total_revenue
    from Sales INNER JOIN Product ON Product.Product_ID = Sales.Product_ID WHERE Date > (SELECT DATE('now', '-7 day'))
    GROUP BY Sales.Product_ID ORDER BY SUM(Sold) LIMIT 5"
    rows = cursor.execute(show_worst_5).fetchall()
    print(rows)
    connection.commit()
    for r in rows:
        self.least_5_table.insert('', END, values=r)
    ...

```

This function shows the Product ID, Product and Quantity Sold for the worst 5 performing products. Here, I have again joined the Sales and Product table using the common field of Product ID to fetch the name of the product using SQL. These are the stored in a variable that is iterated and inserted into the treeview.

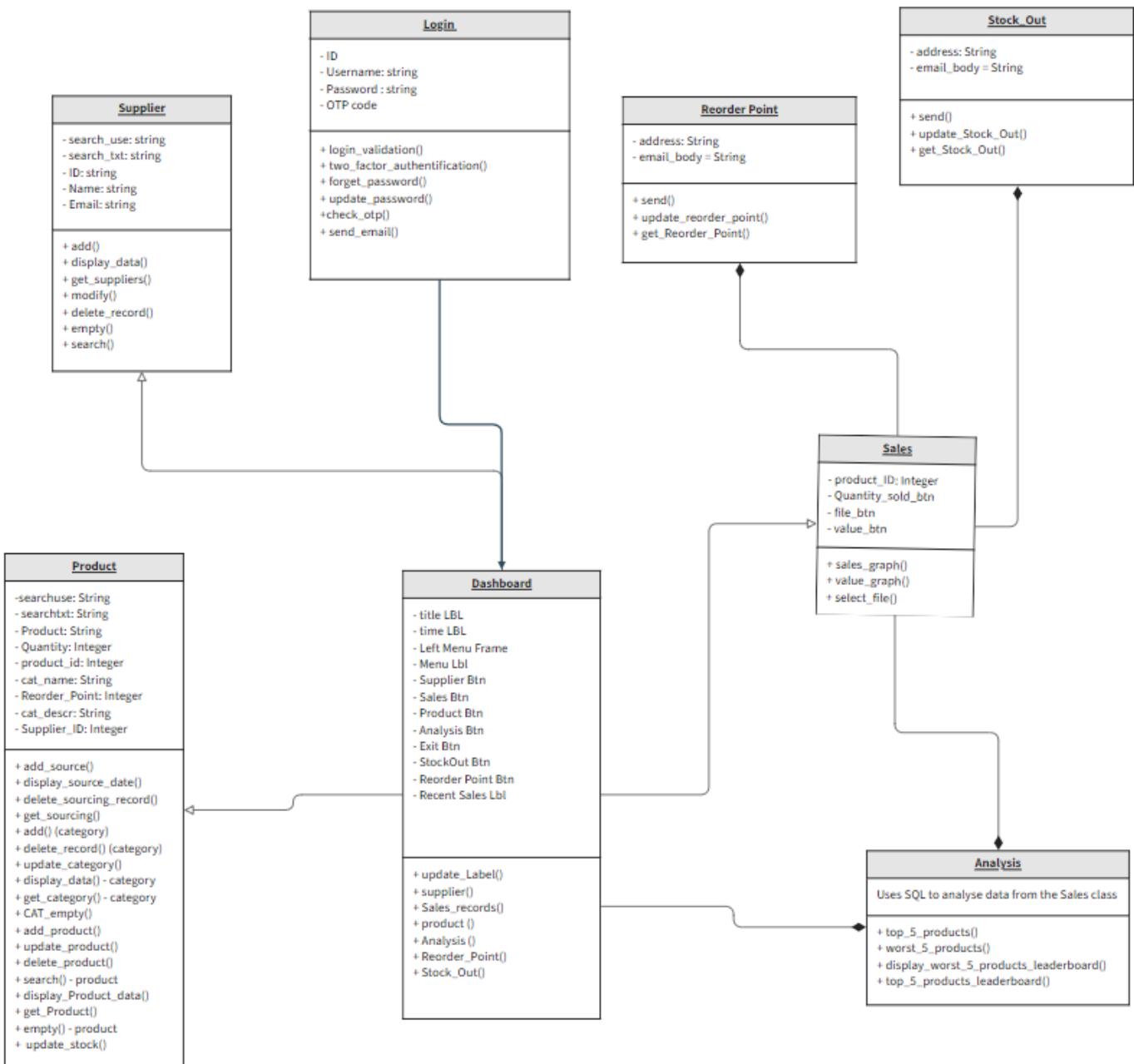
In both functions, I have used the LIMIT clause so that only a set number of records can be fetched. Here, in this case, I have used LIMIT 5 so that only 5 records that fulfil the criteria are fetched and shown on the leaderboard.

2.11. Summary of Data structures to be used:

<u>Data Structures</u>	<u>Reason why it's used</u>
<ul style="list-style-type: none"> Database 	<ul style="list-style-type: none"> Can store lots of data Data can be maintained Data relationships can be made

	<ul style="list-style-type: none"> • Data can be easily accessed through SQL
<ul style="list-style-type: none"> • CSV (comma-separated values) file 	<ul style="list-style-type: none"> • Used to transfer data to a system • Easy to import • Easy to use as you can input different data types separated by a comma • Can be used with SQL
<ul style="list-style-type: none"> • lists 	<ul style="list-style-type: none"> • Used to store records of data fetched from SQL queries so that they can be used to produce graphs

2.12. Class Diagram



Here, I have created a class diagram so that each button in the dashboard is a completely separate class. This will help me to initialize attributes for each class.

The dashboard is the main window from which other windows can be accessed by clicking on the corresponding buttons.

The diagram above shows the variables and the subroutines for each class.

2.13. Validation

Since my program largely revolves around the user inputting data into the system through the use of entry boxes, most of my validation will be largely focused towards the user's input.

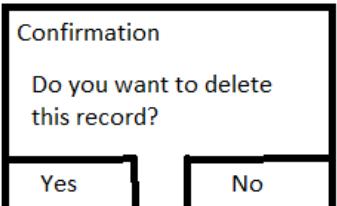
Login validation

<u>User input</u>	<u>Validation</u>
• Entering username and password	• Program should check that if they are valid, then an opt code should be sent via email. Otherwise, an error should occur saying that one of your details is not correct
• Entering OTP code (once username and password is correct)	• Program should check if the OTP code is correct or not. If correct, it should take the user to the dashboard or else an error message should appear.
• Entering OTP code (for forgotten password)	• Program should check that OTP code matches the OTP code generated.
• Entering Password and confirm Password again	• Program should check that the confirm password value is exactly the same as the new password value. If not, then the new password should not be updated into the login table.

Supplier Validation

<u>User input</u>	<u>Validation</u>
• Adding supplier details	<ul style="list-style-type: none"> Ensuring that the same existing supplier ID is not added as Supplier ID is a primary key. Checks to see that the email address of the supplier has the correct format such as having "@". If this is not in the right format, then the

	email sent by the user will not be received by the supplier.
• Updating supplier details	<ul style="list-style-type: none"> Ensuring that an existing supplier ID is being updated otherwise any new supplier ID's can be added using the add button. Supplier ID cannot be changed as it is part of the SQL WHERE condition
• Deleting supplier records	<ul style="list-style-type: none"> Ensuring that an existing record is being deleted. Message box asking the user to confirm the record they delete. This should be included just in case the user accidentally selects a record and presses the delete button.



This is how the message box should look like when the user presses the delete button.

Product Validation

<u>User Input</u>	<u>Validation</u>
• Adding Category records	• Ensuring that they are not integer data types
• Adding Product records	• Ensuring that the category exists in the category table
• Changing or adding stock quantity or reorder points	• Ensuring that the reorder point and the stock quantity fields are not in string data type. If they are, then other parts of the program won't work such as the label buttons in the dashboard that show the number of items out of stock or are below the reorder point

<ul style="list-style-type: none"> Updating Product records 	<ul style="list-style-type: none"> Making sure that the category if changed is existing in the category table. Making that the reorder point and the stock levels is in an integer format (same reason as above) Making sure that the Product name is not repeating
<ul style="list-style-type: none"> Adding source records 	<ul style="list-style-type: none"> Ensuring that the added Supplier ID and Product ID exist in their respective tables otherwise the program will not be able to join the 2 tables together successfully. This would result in the program not being able to detect the supplier's Email address for a given Product ID and so emails won't be able to be sent to the suppliers.
<ul style="list-style-type: none"> Updating source records 	<ul style="list-style-type: none"> Ensuring that the Product ID cannot be updated as it is part of the WHERE condition for updating records. Ensuring that the Supplier ID is existing (same reason as above in adding source records)

Sales Validation

User Input	Validation
<ul style="list-style-type: none"> Creating the sales CSV file content 	<ul style="list-style-type: none"> Ensuring that the date is in correct format (YYYY-MM-DD) or else an error message should occur Making sure that the data entered is in the order of the Sales table. (Date, Product ID, Sold, Value (£)) or else it will affect other parts of the program such as the product table where the stock levels will get adjusted depending on the quantity sold from my each record in the csv file. It will also affect the graphs produced in my sales and analysis windows.

2.14. Test Plan

2.14.1. Post development data

Test plan is created based on the [success criteria](#) such that each criterion has corresponding test id. There can be multiple test ids for a success criterion.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
1.	Login to the system with a username and password and check if an OTP number is received on mobile phone.	Enter username: User1 Enter password: Inventory#1 Click on 'Log In'	OTP received on the registered phone number of User1. OTP should be received within 3 minutes.	S1, S4
2.	Login to the system with a username and password to receive an OTP number on registered mobile phone. Once OTP is entered, access is allowed to system.	Enter username: User1 Enter password: Inventory#1 Click on 'Log In' Enter OTP and then click on 'Submit'	Once OTP is entered, access is allowed to system.	S1
3.	Login to the system with an incorrect username	Enter username: User10 Enter password: Inventory#1 Click on 'Log In'	Message displayed on screen saying 'Invalid username or password'. There should be no OTP received on phone.	S2
4.	Login to the system with an incorrect password	Enter username: User1 Enter password: Inventory1 Click on 'Log In'	Message displayed on screen saying 'Invalid username or password.' There should be no OTP received on phone.	S2
5.	Login to the system with valid username and password but an incorrect OTP	Enter username: User1 Enter password: Inventory#1 Click on 'Log In'	Message displayed on screen saying 'Invalid OTP.' Access to system is not allowed.	S2

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
		Do not enter the OTP received on phone but put a different OTP number.		
6.	User forgets password and is allowed to reset it.	Enter username: User1 Click on 'Click Here' User enters the OTP received to allow the new password to be updated. User must enter the password again to confirm the new password	OTP is received on Email And user can enter a new password.	S3
7.	GUI testing – When user logs in, they see the dashboard with menu icons on left and Stock Out, Reorder Point, and Yesterdays sales.	Enter username: User1 With a valid password to login.	Menu icons on the left for Supplier, Sales, Product, Inventory Adjustment, Analysis, and Exit are shown. Stock Out button shows the number of products currently stocked out. Reorder Point button shows the number of products which are below reorder Point and need new ordering. Sales label shows the total value of yesterday's sales in £ for all products.	S5, S6
8.	Menu icons and labels are easy to read and have contrasting colours	Enter username: User1 With a valid password to login.	Menu icons and labels are easy to read and have contrasting colours	S6, S7
9.	Form fields allow searching of records easily	Enter username: User1 With a valid password to login.	It should be easy to enter text in the fields.	S8

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
		Click on the menu icon 'Supplier' and search for a supplier.		
10.	Menu buttons should navigate to different areas of the software	<p>Enter username: User1 With a valid password to login.</p> <p>1]Click on the menu icon 'Supplier'. 2]Click on the menu icon 'Sales' 3]Click on the menu icon 'Product' 4]Click on the menu icon 'Analysis' 5]Click on the menu icon 'Exit'</p>	<p>After logging in successfully,</p> <p>1] Supplier screen is displayed which allows searching for a supplier</p> <p>2] Sales screen is displayed where import function is available</p> <p>3] Product screen is displayed which allows searching for a product.</p> <p>4] Inventory Adjustment screen is displayed</p> <p>5] Analysis screen is displayed.</p> <p>6] User exists from the software.</p>	S9
11.	Reorder point on Dashboard providing count and further details. Ref: <u>Reorder point design</u>	<p>1] Login to the application (Ensure there are some products which have inventory less than Reorder point).</p> <p>2] On the Dashboard, the Reorder point label should show the count of products currently having inventory below Reorder point.</p> <p>3] Click on the count and it should open a new window showing the details of products</p> <p>4] Click on one of the products from the details view to populate the supplier</p>	<p>1] User logged into application.</p> <p>2] The count of products displayed on dashboard is correct.</p> <p>3] On clicking the count, it opens a new window showing the list of products. This should contain the Product, current inventory Qty, Supplier details.</p> <p>4] After clicking on the product in the details panel, automatically the email panel below should get populated.</p> <p>5] Enter additional text – e.g. Supply 2 cases</p>	S10

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
		details in the email panel below. 5]For the product chosen in #4 above, User can enter any free text in the email panel and send the email to supplier.	and click on the 'Send' button to send email to supplier.	
12.	Reorder point alert Ref: <u>Reorder point design</u>	Test data to have inventory below Reorder point. When user logs in, an alert is shown for every product that is under Reorder point. This alert can be on any page (not restricted to dashboard).	Alert correctly shows the products that are having inventory below Reorder point. User needs to click on the 'X' to close the alert window. Alert should remain until it is closed by user.	S11
13.	Stock-out on Dashboard providing count and further details. Ref: <u>Stock-out Dashboard Button design</u>	1]Login to the application (Ensure there are some products which have inventory as zero). 2]On the Dashboard, the Stock-out label should show the count of products currently having zero inventory. 3]Click on the count and it should open a new window showing the details of products 4]Click on one of the products from the details view to populate the supplier details in the email panel below. 5]For the product chosen in #4 above, User can enter any	1] User logged into application. 2]The count of products displayed on dashboard is correct. 3]On clicking the count, it opens a new window showing the list of products. This should contain the Product, Supplier details. 4]After clicking on the product in the details panel, automatically the email panel below should get populated. 5]Enter additional text – e.g. Supply 2 cases and click on the 'Send' button to send email to supplier.	S12

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
		free text in the email panel and send the email to supplier.		
14.	Stock-out alert	Test data to have inventory as zero for some products. When user logs in, an alert is shown for every product that has zero inventory. This alert can be on any page (not restricted to dashboard).	Alert correctly shows the products that are stocked-out. User needs to click on the 'X' to close the alert window. Alert should remain until it is closed by user.	S13
15.	Import Sales data Ref: <u>Sales design</u>	Create a file with sales data from previous day. File name is 'Sales' and contains Date, Item, Qty. Login to system and click on Import.	Records from file are imported and visible in the Tree view after import.	S14
16.	Import Sales data Ref: <u>Sales design</u>	Create a file with sales data from previous day. This is an Erroneous test – Name the File as 'Sales_'. Login to system and click on Import.	Due to the file name having an underscore in the name, it should not be imported and error message provided to user.	S15
17.	Import Sales data Ref: <u>Sales Design</u>	Create a file with sales data from previous day. This is an Erroneous test. File name is 'Sales' and contains Item, Qty, Date (expected format was Date, Item, Qty). Login to system and click on Import.	File should not be imported and error message provided to user.	S15
18.	Import Sales data Ref: <u>Sales design</u>	Create a file with sales data from previous day. File name is 'Sales' and contains Date, Item, Qty.	Check the time required to complete the import. It should not be more than 1 minute.	S16

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
		Login to system and click on Import.		
19.	Import Sales data Ref: <u>Sales design</u>	Login to system and click on the 'Graph by number if items sold'	Graph is displayed with last 7 days of sales data in Qty	S17
20.	Import Sales data Ref: <u>Sales design</u>	Login to system and click on the 'Graph by value'	Graph is displayed with last 7 days of sales value	S17
21.	Analysis	Login to system and 1] click on 'Analysis' tab from the menu. 2] Close the top five graph 3] Close the least five graph	1] The top five items are displayed in a graph 2] Next graph of least five items is displayed 3] Tree view showing both top and least items displayed.	S18
22.	Add new Product	Login to system and 1] click on 'Product' tab from the menu. 2] Enter category name 3] Enter product name 4] Enter any existing Inventory Qty (if any) 5] Enter Reorder point	Product Id is created automatically, and new Product is stored with its category, inventory and reorder qty.	S19
23.	Add new Supplier	Login to system and 1] click on 'Supplier' tab from the menu. 2] Enter Supplier Id 3] Enter Supplier name 4] Enter Supplier Email	Supplier is created	S20
24.	Attaching Product to Supplier	Login to system and 1] click on 'Product' tab from the menu. 2] In the 'Source' section, enter the Product Id and Supplier Id. Click on 'Add'.	Product and Supplier are linked and visible in the Tree view	S21

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
25.	Add Category	1] click on 'Product' tab from the menu. 2]In the 'Category' section, enter the Category Name and Category Description. Click on 'Add'.	Category is added and visible in the Tree view	S22

26.	Update Category Description	1] click on 'Product' tab from the menu. 2]In the 'Category' section, from the TreeView, select the specific category that needs update. 3] The selected values will be visible in the value box above. Enter the new description and click on 'Update'	Category description is updated and visible in the Tree view.	S23
27.	User should be able to update following attributes of item: Item description, Reorder point, Category	1] click on 'Product' tab from the menu. 2]From the Tree view showing list of products, select the specific product that needs update. 3] In the section below, the values will be populated. Enter a new reorder point, Product description, Category click on 'Update'	For a specific product, the following attributes are updated: <ul style="list-style-type: none">• Reorder Point• Product description• Category	S24
28.	Delete Product	1] click on 'Product' tab from the menu. 2]From the Tree view showing list of products, select the specific product that needs delete. 3] click on 'Delete'	Product is deleted and no longer visible in Tree view. Before Deletion a confirmation message is displayed to user.	S25

29.	Update Supplier Name or Email	Login to system and 1] click on 'Supplier' tab from the menu. 2] From the Tree view showing list of Suppliers, select the specific Supplier that needs update. 3] Enter Supplier name 4] Enter Supplier Email 5] Click on 'Update'	Supplier Name and Email is updated.	S26
30.	Delete Supplier	Login to system and 1] click on 'Supplier' tab from the menu. 2] From the Tree view showing list of Suppliers, select the specific Supplier that needs delete. 3] Click on 'Delete'	Supplier is deleted and no longer visible in Tree view. Before Deletion a confirmation message is displayed to user.	S27
31.	Update Inventory of a product	1] click on 'Product' tab from the menu. 2] From the Tree view showing list of products, select the specific product that needs update or viewing. 3] In the section below, the values will be populated. Enter new Inventory Qty and click on 'Update'	For a specific product, the Inventory is visible and can be updated.	S28, S31

2.14.2. Testing during development (Iterative data)

Here, I will use the test data below to test that my program works while developing it.

I am using 3 different types of data:

- Normal – Valid data that the program should accept
- Boundary: This is still valid but it just falls within the boundary or the specific range of the data
- Erroneous: Invalid data that the program should not accept

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>

1.	Adding supplier details	Supplier Name: Jack ID : 7 Email: Jack.Smith @gmail.com	Name: H ID: 1000 Email: Jack@gmail.com	Name: 24 ID: Supplier Email: X	The Normal and Boundary data should be added to the supplier table but not the Erroneous data
2.	Ensuring that the same Supplier ID is not added	N/A	N/A	Existing supplier ID's	The program should not allow existing ID's to be added – hence an error should be created
3.	If I click on one of the treeview records, it should show up on the entry forms.	Middle record from the treeview	First or last treeview record	None of the records	If the user clicks on existing records, then it should show up in the corresponding entry boxes. If none of the records are clicked, then nothing should show up.
4.	Updating supplier details	Name: Jack changed to John Email: John@gmail.com	Name: J Email: J@gmail.com	Name: 2 Email: 2@gmail.com	The normal and boundary data will get updated but not the erroneous data
5.	Updating supplier ID	N/A	N/A	Adding a different supplier ID	Message box with an error – since this is not an existing ID
6.	Deleting supplier ID	Deleting one of the	Deleting the last record	Selecting a record out of range	Existing supplier records that

		middle records			have been selected, should get deleted.
7.	Message error should occur if the value of the options to select is "select" (the default option).	N/A	N/A	Not selecting the 3 options (ID, Name and Email)	Error asking for an option to be selected
8.	Message error should occur if nothing has been typed into the search bar	N/A	N/A	Nothing	Error asking the user to type something
9.	Testing the search bar	Name: Test (Full name of the supplier) ID: 5 Email: Test@gmai l.com	Name: 's' ID: 5 Email: T	Name: TrialTest ID: 78 Email:TrialTest@gmail .com	For the normal data, there should only be 1 unique record found. For the boundary data, there might be more than 1 record that fulfils a partial input by the user. For the erroneous data, there should be no records found.
10.	Adding a new record in the category table	Name: Hersheys Category Description : Chocolate	Name: H Category Description: C	Name: 20 Category Description: 25	Program should deny the erroneous data
11.	Deleting a record from	Deleting one of the	Deleting the last record	Selecting a record out of range	Program should delete

	the category table	middle records			the existing category records selected
12.	Updating a record from the category table	Updating one of the middle records	Updating the last record	Selecting a record out of range	Program should update the existing records selected
13.	Adding an existing category name	N/A	N/A	Adding existing category name	Should get an error
14.	Adding a product record with no user input	N/A	N/A	No user input	An error should occur
15.	Adding a record with existing product	N/A	N/A	User input of an existing product	An error should occur
16.	Adding a new product record with only product as the input	Product: Pasta	N/A	N/A	Error should occur asking for the category to be added
17.	Adding a new product record with only product and non-existing category name as the input	N/A	N/A	Category: Error Product: Pasta	An error should occur as Test is not an existing category
18.	Entering the wrong product data type	N/A	N/A	Product: 25 and 25r Category: Junk	Program should reject the input
19.	Entering the wrong quantity	N/A	N/A	Reorder point: x Quantity: Y	Program should reject the input due

	and reorder point data type				to wring data type used
20.	Entering the wrong product ID data type	N/A	N/A	X	Program should ignore the user input as the product ID is being auto-incremented
21.	Updating the product table	Updating one of the middle records	Updating the last record	Updating nothing with no record from treeview selected	Product record should successfully update the normal and boundary data
22.	Updating the product table with a non-existing category name	N/A	N/A	Category name: Error	The program should show an error saying that the category doesn't exist so the record cannot be updated.
23.	Deleting an unselected product	N/A	N/A	No user selection of record	Error should occur saying it can't delete
24.	Deleting a selected record	One of the middle records	The last record	N/A	Will delete any existing record regardless of the position of the record
25.	Searching for a product record with no input	N/A	N/A	N/A – no input from user	Program should tell the user to select one of the options
26.	Searching for a product record with an option selected but no user text	N/A	N/A	No user input on search bar	Message asking for the user to type what they want to search for

	input on search bar				
27.	Searching for a product record with input in the search bar but no option selected	N/A	N/A	No option selected	Message asking for the user to select an option
28.	Searching for a product record with all inputs from the user	Option: Product / ID Text: Carrots / 14	Option: Product/ID Text: C/1	N/A	Should show all records that match the pattern of the user's input
29.	User putting an input that doesn't match any of the product records	N/A	N/A	Product: ABC ID: 296	Message showing no records found
30.	Adding a non-existing supplier ID (not in supplier table) to the sourcing table	N/A	N/A	Supplier ID: 27 Product ID: 6 (existing)	Error message stating that the supplier ID doesn't exist (invalid)
31.	Adding a non-existing product ID (not in product table) to the sourcing table	N/A	N/A	Product ID: 50 Supplier ID: 3 (existing)	Error stating that the product ID doesn't exist (invalid)
32.	Adding 2 existing inputs (exist in supplier	Supplier ID: 3 Product ID: 5	N/A	Supplier ID: three Product ID: five	Successfully adds normal data but not erroneous

	and Product table)				
33.	Adding a new supplier ID to a product ID that exists in the sourcing table	N/A	N/A	Product ID: 4 Supplier ID: 3	Should show an error as one product can have only 1 supplier
34.	Adding a repeating supplier ID (that already exists in the sourcing and supplier table) to a new Product ID (doesn't exist in sourcing table)	Supplier ID: 1 Product ID: 14	N/A	N/A	Program should allow existing supplier ID (in both supplier and sourcing table) to supply to multiple products
35.	Updating sourcing table	Supplier ID: changed from 1 to 3 Product ID: 14	Same as normal	Supplier ID: 1 to three Product ID: 14	Should update the normal data and ignore the erroneous data
36.	Updating sourcing table to a product ID that doesn't exist in the product table	N/A	N/A	Supplier ID: 1 – exists Product ID: 100	An error should occur asking the user to enter an existing product ID
37.	Updating the sourcing table to a supplier ID that doesn't exist in the	N/A	N/A	Product ID: 5 Supplier ID: 100	An error should occur asking the user to update to an existing Supplier ID

	supplier table				(that exists in the Supplier table)
38.	Deleting a record from the sourcing table	Deleting one of the middle records	Deleting the last record	Not selecting a record	Will delete any record as long as the Product ID in the entry box is not blank and exists in the sourcing table
39.	Deleting a product with the wrong supplier ID but an existing Product ID in the sourcing table	N/A	N/A	Product ID: 1 Supplier ID: 70	The user will be able to delete the record with the Product ID 1 not matter what the supplier ID is put as
40.	CSV file should be imported to the Sales table	Importing CSV file	N/A	N/A	In the Sales table, the data from the CSV file should be shown as records
41.	If the date is in the wrong format	N/A	N/A	07/03/2023	Error message should occur asking for the date to be in the correct format
42.	Updating the Product levels once CSV file has been imported	N/A	N/A	N/A	Program should successfully reduce the stock levels for multiple products correctly
43.	Produces Sales graphs	Existing Product ID	Existing Product ID	Non-existing Product ID	Produces graph for the

	by quantity and by value for the last 7 days when a user enters product ID into the entry box		(same as normal data)		last 7 days for existing Product ID's
44.	Graph showing top 5 performing products in Analysis window	N/A	N/A	N/A	Graph should show top 5 products by quantity sold
45.	Graph showing worst 5 performing products in Analysis window	N/A	N/A	N/A	Graph should show worst 5 performing products
46.	Treeview showing top 5 products in descending order	N/A	N/A	N/A	Treeview should show top 5 products in descending order with the first record showing the highest quantity sold for a product
47.	Treeview showing worst 5 products in ascending order	N/A	N/A	N/A	Treeview should show worst 5 products in ascending order based on quantity sold
48.	Reorder Point button should show	N/A	N/A	N/A	Number should be shown on top

	the number of products that are below the reorder point				of the reorder point button
49.	Stock-out button should show the number of products where the quantity is 0	N/A	N/A	N/A	Number should be shown on top of the stock-out button
50.	Recent sales showing the total amount of sales in value (£) for all the products for the most recent date	N/A	N/A	N/A	Number showing the total amount of value made (£) for all products sold on the most recent date
51.	Message box being displayed to show the items that are out of stock or need reordering	N/A	N/A	N/A	Message box should show correctly the names of products that are below the reorder point or have no quantity in the shop. This should be shown once.
52.	Treeview showing products and linked suppliers when product quantity is below reorder point	N/A	N/A	N/A	Treeview containing products and linked suppliers that are below reorder point

53.	User should be able to send email to the suppliers (same test for stock_out window as well)	N/A	N/A	N/A	Suppliers' should get an email sent by the user (client who owns the store)
54.	Treeview should show products and linked suppliers when product quantity is 0	N/A	N/A	N/A	Treeview containing products and linked suppliers where quantity is 0
55.	Random OTP number is generated and sent via email	N/A	N/A	N/A	6-digit random number generated each time
56.	Check if OTP code is correct	Correct OTP code	N/A	Invalid OTP code	No error message appearing
57.	Password being updated	N/A	N/A	N/A	Password should be updated in the login table. The next time the user logs in, the new password should be used.
58.	Password should not be visible in English	N/A	N/A	N/A	When user types in their password, it should be converted into asterisks.

Development

Stage 1: Designing the Dashboard user interface

Dashboard

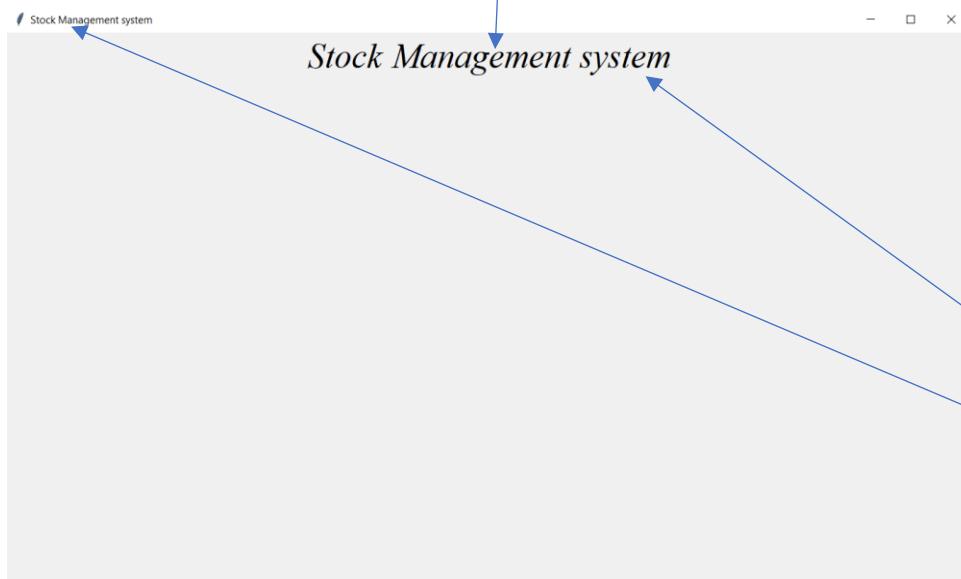
This is the code to create the structure for my dashboard. Here I have used Tkinter which allows me to create a GUI for my stock management system. Tkinter can cross-platform meaning that it can work with macOs, Windows and Linux. This adds flexibility to the system as it can be used on any computer regardless of the operating system used.

Here I have created a class called SMS (short for Stock Management System) which acts like a blueprint for creating objects.

```

1  from tkinter import * # this is used to create the GUI of my stock management system
2  class SMS:
3      def __init__(self, wind): # short for window
4          self.wind = wind
5          self.wind.geometry("1400x800+0+0") # this sets the dimensions of the window
6          self.wind.title("Stock Management system") # this is the title of the window
7          title = Label(self.wind, text = "Stock Management system", font = ("times new roman", 30, "italic")).place(x=0, y=0, relwidth= 1, height = 70)
8
9      wind = Tk()
10     obj = SMS(wind)
11     wind.mainloop()

```



This is the output I get from the code.

Here the structure works as we can see a big title with all the features like italics, size, font etc. We can also see a small title at the top left corner.

Next step: I will add some design to this interface by adding some colours and creating labels using Tkinter that will allow some sort of interactivity that will introduce to other sections of the stock management system like the sales section, suppliers etc.

```

from tkinter import *      # this is used to create the GUI of my stock management system
class SMS:
    def __init__(self, wind): # short for window
        self.wind = wind
        self.wind.geometry("1400x800+0+0")    # this sets the dimensions of the window
        self.wind.title("Stock Management system")      # this is the title of the window
        title = Label(self.wind, text = "Stock Management system",
                      font = ("arial", 30, "bold"), bg = "#FFFFE4", fg = "blue").place(x=0, y=0, relwidth= 1, height = 70 )
wind = Tk()
obj = SMS(wind)
wind.mainloop()

```

Here I have changed the colour of the text as well as the background of the label.

In this code, the background colour is given in colour codes in hexadecimal. This hexadecimal code corresponds to the colour bisque1 which is a creamish colour.

Note: Here I have used the colour codes from Tkinter so the bisque hexadecimal code might be different from another website.

Output:



Here we can see that I have added some aesthetics. The label title “Stock Management system” has been emphasized by adding some colours and by changing the text style from being italic to bold and from times new roman to arial.

Note: Though I have set the dimensions of the window, I can enlarge the screen so that the window takes the entire screen. If I wanted to have a fixed window dimension that cannot be enlarged/resizable, then I would have to write a resizable function where I would have to write the Boolean value false, false for both the height and width parameters.

For now, I have not written this code and will later decide whether I should make the window non-resizable.

Logout Button

```
butn_log = Button(self.wind, text = "Log off", font = ("arial", 10, "bold"), command = wind.destroy) # creating a log off button
butn_log.place(x = 1300, y = 10) # this places the log off button depending on its position
```

This is the code for my logout button. Here the text is “Log off” and have positioned it so that it goes to the top-right corner of the window. This is done by using the place function which allows you to place the button wherever you specify.

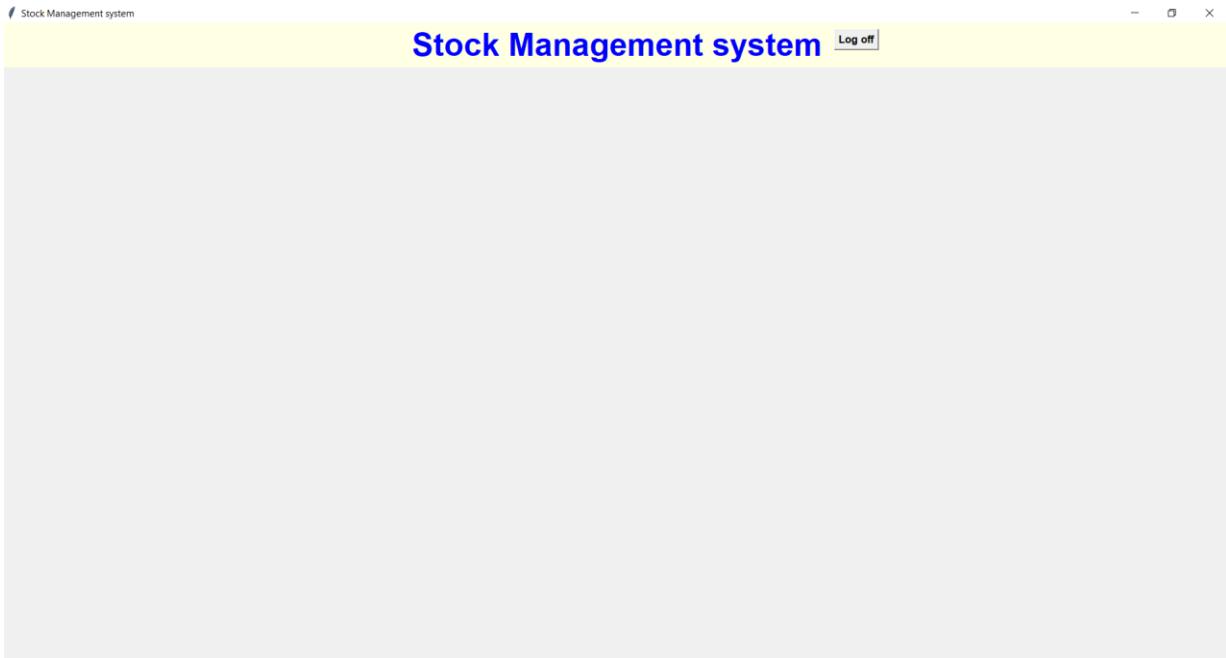
Here I have used the destroy function() to close the Tkinter window if the button is clicked.

The only problem is that my window dimensions are smaller than the size of the screen. So, using my window dimensions, the button goes to the top right. However, if I make the window resizable then, the button goes to the close right of the big title “Stock Management System”.



This is the size of the window using its dimensions. Here you can see that the Log off button is on the top-right corner. The button positioning looks better when it is at the corner and you will see on most sites that the log-off button is always in the corner.

For the log-off button, I was able to change the appearance of the text by making it bold and by setting the font to Arial. I didn't make it too visually attractive, as it is not the main feature of my dashboard.



This is what happens when I make the window full screen. Here is the log-off button which is still towards the right end of the window but it's a bit too close to the title.

So I will change the dimensions of the window so that it takes the entire screen and I will also make it non-resizable or else the positioning of all the buttons to be added will change.

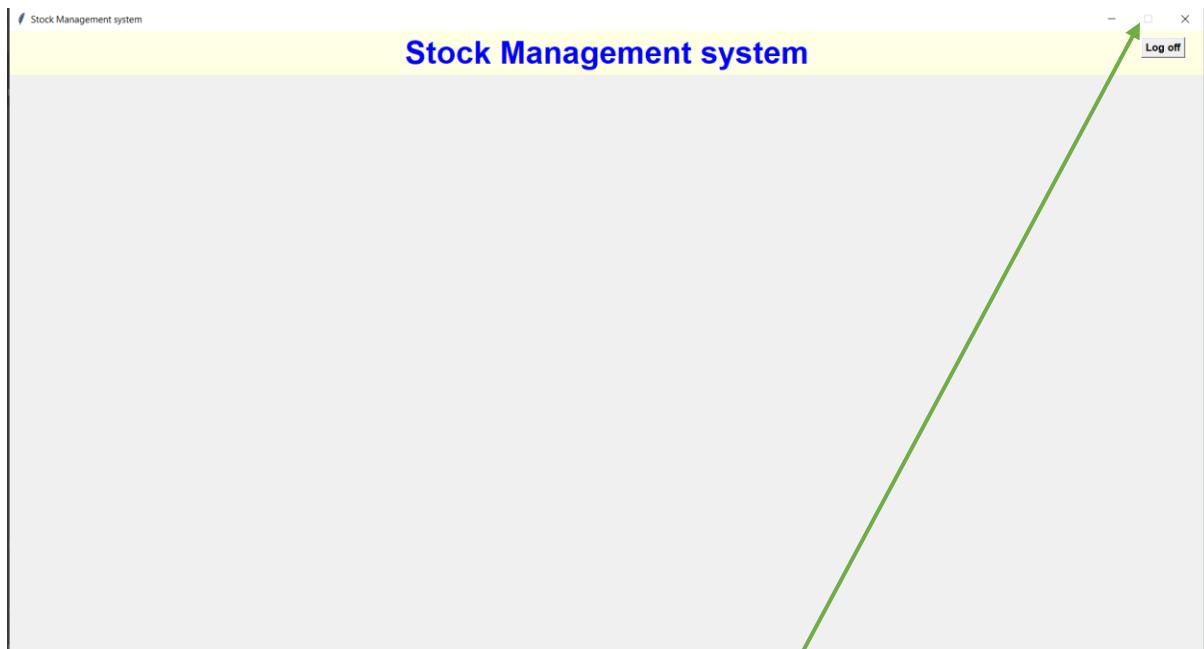
```

1  from tkinter import *      # this is used to create the GUI of my stock management system
2  class SMS: # creating a class called SMS - short for Stock Management System
3      def __init__(self, wind):    # short for window
4          self.wind = wind
5          self.wind.geometry("1900x990+0+0")    # this sets the dimensions of the window
6          self.wind.title("Stock Management system")        # this is the title of the window
7          title = Label(self.wind, text = "Stock Management system")
8          title['font'] = ("arial",30, "bold"), title['bg'] = "#FFFFE4", title['fg'] = "blue"
9          title.place(x=0, y=0, relwidth= 1,height = 70)
10         self.wind.resizable(False, False) # you can't resize the screen
11
12         btn_log = Button(self.wind, text = "Log off", font = ("arial", 10, "bold"), command = wind.destroy) # creating a log off button
13         btn_log.place(x = 1800, y = 10) # this places the log off button depending on its position
14
15
16         wind = Tk()
17         obj = SMS(wind)
18         wind.mainloop()

```

Here in line 10, I have added the resizable function so that the dimensions of the window cannot be resized. I have also changed the dimensions of my window to 1900 and 990 for the width and height of my window. The button's x coordinate has been changed from 1300 to 1800 so that it is in the top-right corner.

The dimensions for my window are not exactly the same as the screen's dimensions as I needed some space to have the entire window on the screen. Otherwise, I would have to drag the window to recentre it to see some of the buttons like the cross sign for closing.



This is what it looks like after modifying some of the values. Here the log-off button is in the right-hand corner, and you can also see that there is a faint line on the maximising button. This shows that the window cannot be maximised any further.

Next step: I will add some images and more content with colour to start filling the dashboard.

Since I am going to use an image, I decided I will need to speak with my client to decide which image they would like to stick with. I have gathered 3 images and will ask for their feedback.



"I really like the image. It reflects exactly to what the customers will do in the shop. As the trolley is full, it shows that the shop has a variety of items and shows how good the shop is performing."



"This image is great since it reflects on how many stocks the shop has. The stocks are neatly organised into shelves which is the basic structure of organising the food items".



"I really like this image as it combines the concepts of the previous 2 images." It shows the stocks being arranged on shelves and the customers' behaviour when they enter the food store."

Summary: The client likes the last image so I will use this in my dashboard.

I have saved this file as a png image – called “grocery store.png”. I will use this identifier in my code to get the image displayed on the Tkinter window.

```
self.image_title = PhotoImage(file = r"C:\Users\rusha\PycharmProjects\pythonProject5\grocery_store.png") # path showing where my image is
```

I have mentioned the file path name here but this will be different for my client. When my client uses the software, I will ensure that the photo should be stored on their computer and the file should be typed into the code directly.

When I run the program, I get an error.

```
File "C:\Users\rusha\AppData\Local\Programs\Python\Python39\lib\tkinter\_init__.py", line 4064, in __init__
    Image.__init__(self, 'photo', name, cnf, master, **kw)
File "C:\Users\rusha\AppData\Local\Programs\Python\Python39\lib\tkinter\_init__.py", line 4009, in __init__
    self.tk.call('image', 'create', imgtype, name,) + options)
_tkinter.TclError: couldn't recognize data in image file "C:\Users\rusha\PycharmProjects\pythonProject5\grocery_store.png"
```

The format of the file is fine. The problem could be that the image file is corrupted.

```
# path showing where my image is
self.image_title = PhotoImage(file = r"C:\Users\rusha\PycharmProjects\pythonProject5\grocery_store_image.png")
```

I have changed the name of the image and have run the program again. This time the program works.

Creating the Menu Frame

```
LeftMenu = Frame(self.wind, bd=2, bg = "lightblue", relief=GROOVE, cursor = "circle") # this changes the cursor to circle when its on the frame
LeftMenu.place(x=0, y=170, width=295, height=1000) # this is used to adjust the placement of the frame as well as its dimensions
```

To create the frame, I have specified in which window it will go in, so here I have used "self.wind" so that the frame goes into this window only.

Here, I am using the Tkinter Frame widget to create a frame to show where the Menu buttons will be.

Creating the Menu buttons and label

```
#####
-----These sections will go under the left menu-----
#####

menu_sect = Label(LeftMenu, text = "Menu", font = ("times new roman", 50, "bold"), bg = "orange").pack(side = TOP, fill = X) # this is a label

butn_supplier = Button(LeftMenu, text = "Supplier", command = self.supplier,
                      font = ("times new roman", 44, "bold"), bg = "white", bd = 3, cursor = "plus").pack(side = TOP, fill = X) # supplier button

butn_sales = Button(LeftMenu, text="Sales", command = self.Sales_records,
                     font= ("times new roman", 44, "bold"), bg="white", bd=3, cursor="plus").pack(side=TOP, fill=X) # sales button

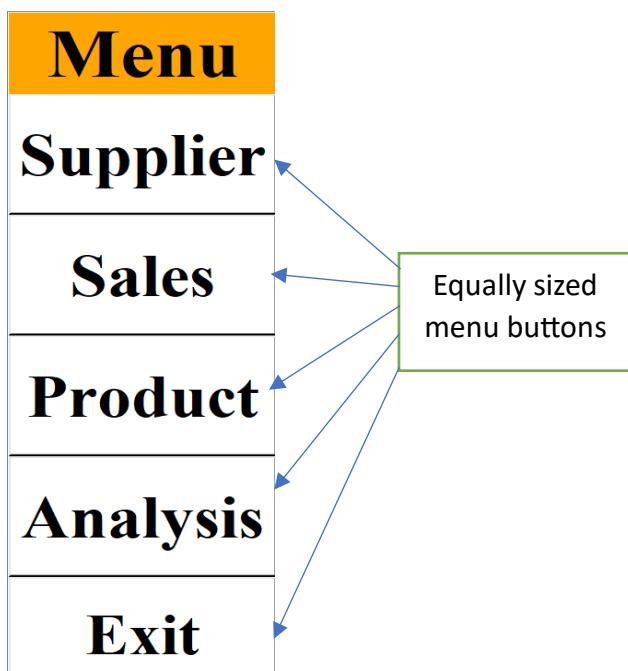
butn_product = Button(LeftMenu, text="Product", command = self.product,
                      font= ("times new roman", 44, "bold"), bg="white", bd=3, cursor="plus").pack(side=TOP, fill=X) # product button

butn_analysis = Button(LeftMenu, text = "Analysis", command= self.Analysis,
                       font = ("times new roman", 44, "bold"), bg = "white", bd = 3, cursor = "plus").pack(side = TOP, fill = X) # analysis button

butn_exit = Button(LeftMenu, text="Exit", font= ("times new roman", 42, "bold"), bg="white", bd=3, cursor="plus").pack(side=TOP, fill=X) # exit button
```

For the buttons and the label to be added to the LeftMenu frame, we need to add the name of the frame as one of the arguments.

Here, I have added a command that will cause a function to be processed when clicked. I will create the functions later. To execute the code, I will get rid of these commands as I haven't defined the functions yet.



Here, the Menu is a label and the others are buttons that when clicked will show a different Tkinter window. I have ensured that the texts are large enough for my client to read and that the dimensions of the buttons are equal to ensure that the GUI dashboard looks friendly.

When creating these labels and menus, I can specify what the text should be, the size and font types used.

```

menu_sect = Label(LeftMenu, text = "Menu", font = ("times new roman", 50, "bold"), bg = "orange").pack(side = TOP, fill = X) # this is a label

butn_supplier = Button(LeftMenu, text = "Supplier", command = self.supplier,
                      font = ("times new roman", 44, "bold"), bg = "white", bd = 3, cursor = "plus").pack(side = TOP, fill = X) # supplier button

butn_sales = Button(LeftMenu, text="Sales", command = self.Sales_records,
                     font = ("times new roman", 44, "bold"), bg = "white", bd = 3, cursor = "plus").pack(side = TOP, fill = X) # sales button

```

Creating the Reorder Point, Stock Out Button and the Recent Sales Label

```

self.lbl_stockout = Button(self.wind, text = "Stock-Out \n[0]", bg = "green", command = self.Stock_Out, relief = "groove", fg = "white", font = ("arial", 20, "bold"))
self.lbl_stockout.place(x = 400, y = 250, height = 250, width = 350)

self.lbl_reorder = Button(self.wind, text = "Reorder-Point \n[0]", bg = "blue", command = self.Reorder_Point, relief = "groove", fg = "white", font = ("arial", 20, "bold"))
self.lbl_reorder.place(x = 900, y = 250, height = 250, width = 350)

self.lbl_sales_recent = Label(self.wind, text = "Recent Sales \n[0]", bg = "red", relief = "groove", fg = "white", font = ("arial", 20, "bold"))
self.lbl_sales_recent.place(x = 1400, y = 250, height = 250, width = 350)

```

This code below creates 2 buttons – the reorder point and the stock out button and 1 label that displays the most recent sales in £. Here, I have used the place() method to explicitly set the coordinates of the widgets across the entire Tkinter window.

Here, I have used the text method to store the corresponding values so that it can be seen from the Dashboard window. By default, there are all set to 0 and can be later updated as shown in my algorithms.

Again, for the buttons – when clicked they will cause the respective functions to be performed.

Creating a time label

```

self.time = Label(self.wind, text = "Welcome To The Stock Management system!\n\t Date: DD-MM-YYYY\n\t Time: HH:MM:SS",
                  compound = LEFT, font = ("arial", 15, "bold"), bg = "yellow", fg = "green")
self.time.place(x = 0, y = 140, relwidth = 1, height = 30)

```

This code shows the title of the software on the dashboard and will show the current time. Again, here I have adjusted the properties and have placed it so that the label goes to the top of the window.

```

def update_label(self):
    time_ = time.strftime("%H:%M:%S")
    date_ = time.strftime("%d-%m-%Y")
    self.time.config(text = f"Welcome to Stock Management System\n\t Date: {str(date_)}\n\t Time: {str(time_)}") # this ensures that the labels change in real time
    self.time.after(200, self.update_label)

```

This function above updates the time. Here I have used the config method to overwrite the contents of the label.

The first 2 lines use the ‘strftime’ method of the time module to retrieve the current time and date in the format specified by the arguments passed in this method. “%H:%M:%S” specifies the time format as hours:minutes:seconds, while %d-%m-%Y specifies the date format as day-month-year.

The after method of the time widget is called to allow the update_label method to be called after 200 milliseconds. This causes the label to be updated with the current time and date every 200 milliseconds.

This is how the top of the Tkinter dashboard window should look like:

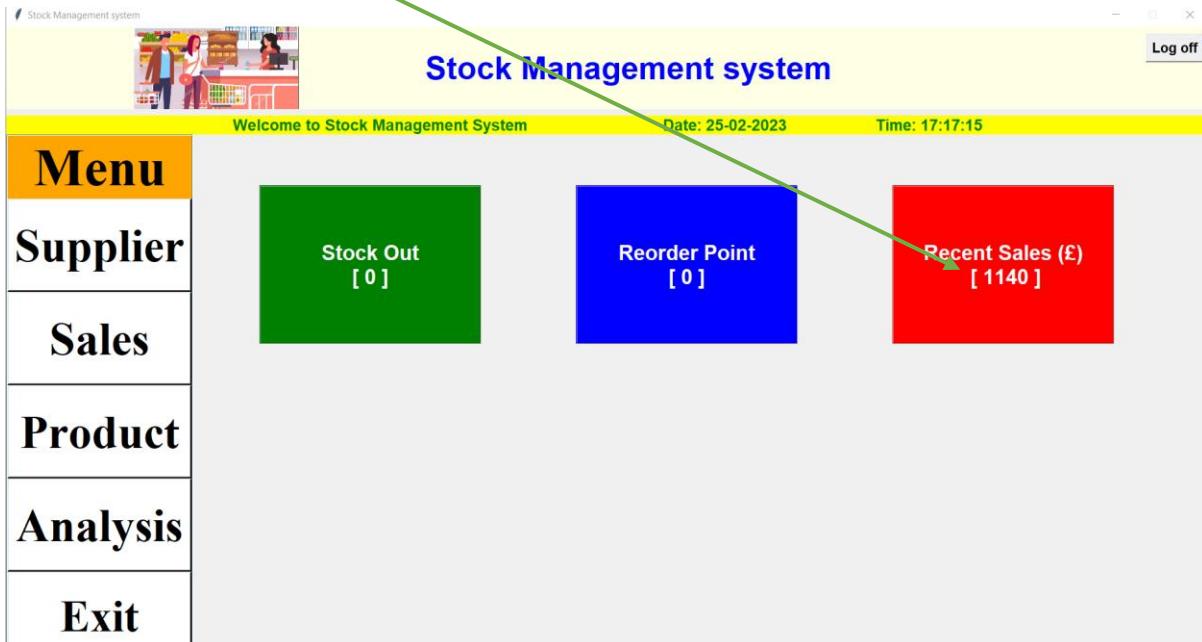


Here, the date and the time are updated showing the current date and time of the day.

Overall, with all the features in the dashboard, it should look like this:

Here, I've just manually entered a random Sales value (1140) onto the code itself to see if it shows up on the window.

```
self.lbl_sales_recent = Label(self.wind, text="Recent Sales \n[1140]", bg="red", relief="groove", fg="white", font=("arial", 20, "bold"))
self.lbl_sales_recent.place(x=1400, y=250, height=250, width=350)
```



Stage 2: Supplier Window

Initialising the variables

```

from tkinter import *      # this is used to create the GUI of my stock management system
from PIL import Image, ImageTk # in case we want any pictures
from tkinter import ttk, messagebox # this will show any errors or any info when giving input
import sqlite3
import re

class SupplierClass: # creating a class called SupplierClass
    def __init__(self, wind):    # short for window
        self.wind = wind
        self.wind.title("Supplier details") # this is the title of the window

        # supplier variables
        # StringVar() stores string data and allows you to manage the value

        self.var_searchuse = StringVar()
        self.var_searchtxt = StringVar()
        self.var_supplier_id = IntVar()
        self.var_supplier_name = StringVar()
        self.var_supplier_email = StringVar()

```

Here, I have created a Supplier class from which all my features will be added into. I have initialized the variables that I will be using for specific operations.

Here, I have used self to represent the instance of the class. By using self, I can get access to the variables and methods of the instance of the class.

- self.var_searchuse: This variable is used to store the user's choice of what they will use (ID, Name or Email) to search through the supplier treeview
- self.var_searchtxt : Stores the input of what the user searches on the search bar

The other 3 variables store the ID, Name and the Email of the suppliers.

```

# adjusting the window so that it fits into the homepage
window_width = 1550
window_height = 750
screen_width = wind.winfo_screenwidth()
screen_height = wind.winfo_screenheight()
center_x = int(130+(screen_width / 2 - window_width / 2)) # centre x
center_y = int(50+(screen_height / 2 - window_height / 2)) # centre y

# set the position of the window to the center of the screen
self.wind.geometry(f'{window_width}x{window_height}+{center_x}+{center_y}')

```

This code above is used to set the dimensions of the supplier window. Since this is a top-level widget, this will be on top of the dashboard window. I have ensured that the entire dashboard window is not covered, so I have set the dimensions accordingly.

I have gone back to the dashboard window and have added this code below. This function will create the new top-level window (the dimensions set from the previous code) and passes the self.wind object as the parent window.

```
def supplier(self):
    self.supplier_1 = Toplevel(self.wind)
    self.new_supp = SupplierClass(self.supplier_1)
```

A new instance of a class called “SupplierClass” is created which has been passed as an argument. Note that to create this function, I would need to define the SupplierClass – why is why I started coding for the Supplier window first.

For this function to work, it must be connected with the Supplier button so that when it is clicked by my client, this new window can be created.

```
butn_supplier = Button(LeftMenu, text = "Supplier", command = self.supplier,
                      font = ("times new roman", 44, "bold"), bg = "white", bd = 3, cursor = "plus").pack(side = TOP, fill = X) # supplier button
```

Here, command = self.supplier means that when the button is clicked, it will go to the function supplier and run it. Here, I have used “self”.supplier so that we can access the method (function) of the instance of the class.

```
if __name__ == "__main__":
    wind = Tk()
    obj = SupplierClass(wind)
    wind.mainloop() # this executes Tkinter
```

This code on the left needs to be added so that the GUI window can be opened when the user inputs it.

This code will be added to other classes as well.

```
from supplier import SupplierClass
```

This code here allows the user to use the ‘SupplierClass’ in the dashboard program. By using the import function, the user can use it easily without having to copy the class definition to every module by importing it. In the dashboard, I will import all the classes so that the other programs from different classes can be used from the dashboard. This is much easier than going on individual class files and executing them one by one.

```
Search_box = LabelFrame(self.wind, text = "Search Supplier", bg = "lightblue", font = ("arial", 12, "bold"))
Search_box.place(x = 30, y = 20, width = 650, height = 70)

# options to search from - Name, Email or ID

option_block = ttk.Combobox(Search_box, textvariable= self.var_searchuse, values = ("select", "Name", "Email", "ID"),
                           state= "readonly", justify = CENTER,
                           font = ("times new roman", 15))
option_block.place(x = 10, y= 5, width = 200, height = 30)
option_block.current(0)
```

This code above generates a "Search box" label frame. Additionally, it adds a combobox inside the label frame called "option block" with the search choices "select," "Name," "Email," and "ID."

The combobox is in the "read-only" state, which prevents the user from entering their own options and only allows them to be selected from a drop-down list. The combo box's default value is "select".

Here a text variable is an optional feature that allows a Tkinter widget to be set up with a value. By using the 'textvariable', the value of the widget can be fetched or set using the variable. This variable "searchuse" will store the user's option that they choose and that will be used to search through the records. For example, if the user clicks on the dropdown list with the value "Name", then the user will search the supplier by typing their name or part of their name.

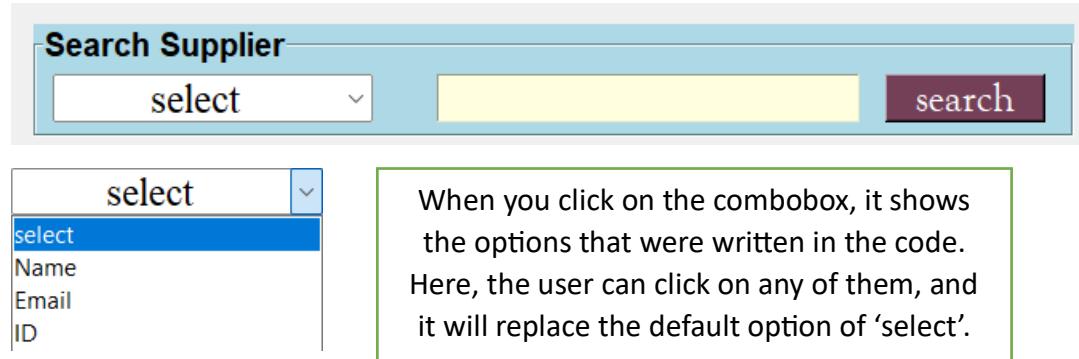
```
txt_block = Entry(Search_box, textvariable= self.var_searchtxt ,font = ("goudy old style", 15), bg = "lightyellow").place(x= 250, y = 5, height = 30)
btn_block = Button(Search_box, text = "search", command = self.search, font = ("goudy old style", 15), bg = "#734058",
fg = "white").place(x= 530, y = 5, width = 100, height = 30)
```

Here, there are 2 widgets being created – an entry and a button. The entry widget creates a form where the user can type in something they want to search. Here, the text variable is being used to store the user's input which will be used by the search function.

The button widget will allow the search function to be executed. Here, we haven't made the search function so temporarily, I will delete the command part.

Both the widgets are inside the search_box as it has been passed here as an argument.

This is how it should look like:



```

lbl_suppl_id = Label(self.wind, text = "Supplier ID",
                     font = ("Rosewood Std Regular",15), bg = "#4cafaf", fg = "black").place(x=30, y = 350, height = 60)

lbl_suppl_name = Label(self.wind, text = "Supplier Name",
                       font = ("Rosewood Std Regular",15), bg = "#4cafaf", fg = "black").place(x=30, y = 150, height = 60)

lbl_suppl_email = Label(self.wind, text = "Supplier Email", font = ("Rosewood Std Regular",15),
                        bg = "#4cafaf", fg = "black").place(x = 30, y = 550, height = 60)

txt_suppl_id = Entry(self.wind, textvariable= self.var_supplier_id, font = ("Rosewood Std Regular",15),
                      bg = "lightyellow", fg = "black").place(x=340, y = 350, height = 60)

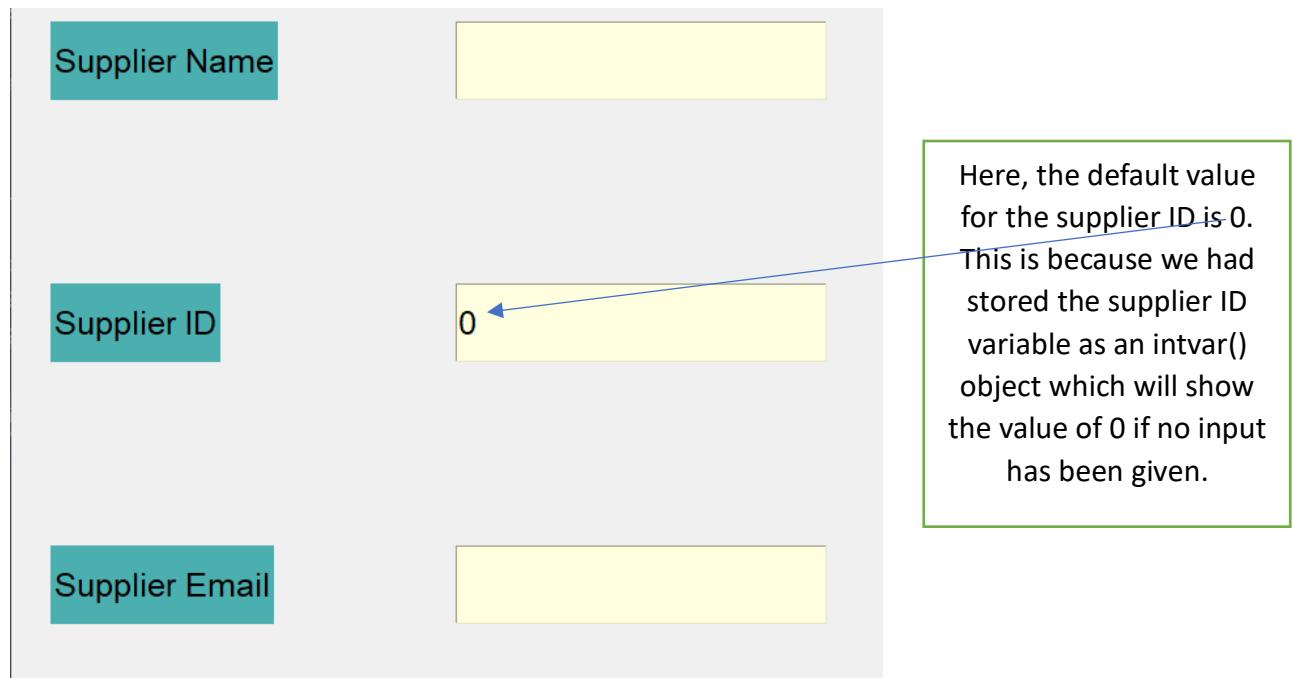
txt_suppl_name = Entry(self.wind, textvariable= self.var_supplier_name, font = ("Rosewood Std Regular",15),
                       bg = "lightyellow", fg = "black").place(x=340, y = 150, height = 60)

txt_suppl_email = Entry(self.wind, textvariable= self.var_supplier_email, font = ("Rosewood Std Regular",15),
                        bg = "lightyellow", fg = "black").place(x = 340, y = 550, height = 60)

```

Here, I have created 3 labels and 3 entry boxes from which the user can enter the supplier's details. Once again, the text variable has been used to link the variables used to store the supplier details with the entry boxes. Once the user puts input into those boxes, the value of those input is stored in the corresponding variables. Here, all these widgets use the parameter (self.wind) to ensure that those widgets lie on that corresponding Tkinter window.

The output should look like this:



```

##### Adding button
Add_btn = Button(self.wind, text = "Add", command = self.add, bg = "#734058",
                  font = ("OCR A Std", 15), fg = "yellow").place(x = 25, y = 670, height = 55, width = 100)

##### Modifying Button (updates)
Mdfy_btn = Button(self.wind, text="Modify", command = self.modify, bg="#734058",
                  font=("OCR A Std", 15), fg="yellow").place(x=205, y=670, height= 55, width = 100)

##### Deleting Button
Dlt_btn = Button(self.wind, text="Delete", command = self.delete_record, bg="#734058",
                  font=("OCR A Std", 15), fg="yellow").place(x=385, y=670, height=55, width = 100)

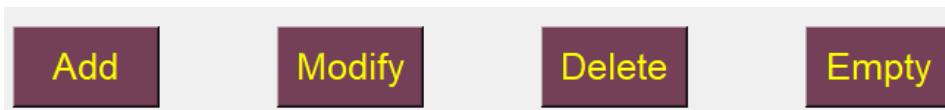
##### Empty Button
Empty_btn = Button(self.wind, text = "Empty", command = self.empty, bg = "#734058",
                  font = ("OCR A Std", 15), fg = "yellow").place(x = 565, y = 670, height = 55, width = 100)

```

This code above is for creating the buttons that will allow us to do the basic operations such as add, modify and delete. The empty button here is to remove any values that are currently in the entry boxes.

Once again, these buttons are linked to functions that when clicked, the respective function will be executed. To define the functions, we need to first create a SQLite database which I will be doing soon.

The buttons look like this:



Connecting the database

```

import sqlite3
def create_dtbse():
    con = sqlite3.connect(database=r"sms.db")
    cur = con.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS supplier(ID integer PRIMARY KEY,Name text,Email text)") # supplier table
    con.commit()
create_dtbse()

```

This function here creates a SQLite database called ‘sms.db’.

Here, the ID is the primary key which will be used to identify individual records in the supplier table.

By importing the SQLite module, it provides a connection between Python and SQLite. A connection has been made and there is a cursor function which is used to create a cursor object that will be used for SQL queries.

All these changes have been committed using the commit method.

sms

25/02/2023 19:03

Data Base File

If I go on DB Browser and then open database, I can see the sms file created for me already.



Here, a supplier table has been made from the function above. Here, I can add data about different suppliers to the supplier table.

```
#### Having a vertical frame section

Sply_frame = Frame(self.wind, bd = 7, relief = GROOVE, highlightcolor="red", bg = "#734058", cursor = "circle" )
Sply_frame.place(x = 670, y = 100, height = 900, relwidth = 1 )

#### vertical scrollbar
myscrolly = Scrollbar(self.wind, orient= VERTICAL)
myscrollx = Scrollbar(Sply_frame, orient = HORIZONTAL)

##### Treeview #####
self.SupplyTable = ttk.Treeview(Sply_frame, columns=("ID", "Name", "Email"), yscrollcommand= myscrolly.set, xscrollcommand= myscrollx.set)

myscrollx.config(command = self.SupplyTable.xview)
myscrolly.config(command = self.SupplyTable.yview)

myscrollx.pack(side = BOTTOM, fill = X)
myscrolly.pack(side = RIGHT, fill = Y)

self.SupplyTable.heading("ID", text="Supplier ID")
self.SupplyTable.heading("Name", text = "Name")
self.SupplyTable.heading("Email", text="Email")
self.SupplyTable["show"] = "headings"

self.SupplyTable.column("ID", width =280, stretch=NO) # fixed width
self.SupplyTable.column("Name", width = 280, stretch=NO) # fixed width
self.SupplyTable.column("Email", width =280, stretch= NO) # fixed width

self.SupplyTable.pack(fill=BOTH, expand=1)
```

This code here allows the treeview for showing the values of the supplier table to be created. Here, the width of each of the 3 columns are the same and the order of the columns are made so that they match the order of the supplier table – ID, Name and Email.

This treeview has been created so that it is within the Frame that has been created.

	Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com	
2	Rushabh	rushabh.is.co028@gmail.com	
3	Bob	wateraid2.0@gmail.com	
4	Nijan	nijankannathasan@gmail.com	

This is how the whole supplier window should look like. Here, I have added values directly from the DB Browser to check whether the supplier details are being displayed on the treeview.

Add button function

```
def add(self): # creating a function to add supplier details to the database
    con = sqlite3.connect(database= r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:

        if self.var_supplier_id.get() == "" or self.var_supplier_email.get() == "" or self.var_supplier_name.get() == "":
            messagebox.showerror("Incorrect Input", "Please fill in all the details", parent = self.wind) # validation. Cannot add supplier detail without supplier ID
        else:
            cur.execute("Select * from supplier where ID=?", (self.var_supplier_id.get(),))
            row = cur.fetchone() # fetches the next row of data
            if row != None:
                messagebox.showerror("Error", "This supplier ID already exists", parent = self.wind)
            else:
```

Here, the `get()` method is used to retrieve the variable's value from the entry box. I have ensured that the user cannot leave any of the entry boxes empty by producing a message box that tells the user to fill in all the forms. If the user has filled these in, then the inputs are checked to ensure that there are no existing supplier ID's since it is a primary key. If there is already an existing ID, then an error is produced.

```
        cur.execute("Insert into supplier(ID, Name, Email) values(?, ?, ?)", (
            self.var_supplier_id.get(),
            self.var_supplier_name.get(),
            self.var_supplier_email.get(),

        ))
        con.commit() # commit any changes
        messagebox.showinfo("Added!!", "Successfully Added:" + ' ' + str(self.var_supplier_name.get()), parent = self.wind)
        self.empty()

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent = self.wind)
```

If there is no existing supplier ID, then the user's input is added to the database. A message box is displayed showing that it has been successfully added. If it's not added due to the wrong data type being used, then an error is produced.

The problem occurs that if I add a number to the supplier's name and I do not have the right email address, the program would still add it to the database. For example, below the program is allowing the Supplier's name to be 45 and the email to be 4. This is not correct and it can cause issues in other parts of the program such as the reorder point and stock out windows, where I will be relying on the Supplier table to send emails to the supplier.

Supplier details			
Supplier Name	45	Supplier ID	1 2 3
		 Added!!	X
		 Successfully Added: 45	OK
Supplier ID	5		
Supplier Email	4		
Add	Modify	Delete	Empty

The code is adding the details perfectly to the supplier table but it is unable to detect if they are valid or not.



	Supplier ID	Name	Email
1		Sam	rushlovesgames28@gmail.com
2		Rushabh	rushabh.is.cool28@gmail.com
3		Bob	wateraid2.8@gmail.com
4		Nijan	nijankannathasan@gmail.com
5	45		4

```

else:

    if not self.var_supplier_name.get().isalpha():
        messagebox.showerror("Incorrect Input", "Please enter a valid name", parent=self.wind)
        return

    email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
    if not re.match(email_pattern, self.var_supplier_email.get()):
        messagebox.showerror("Incorrect Input", "Please enter a valid email address", parent=self.wind)
        return

    cur.execute("Insert into supplier(ID, Name, Email) values(?, ?, ?)", (
        self.var_supplier_id.get(),
        self.var_supplier_name.get(),
        self.var_supplier_email.get(),
    ))

```

Here, I have added more code so that the name and the email address can be validated. For the name, I have ensured that it only contains letters by using the `isalpha()` method that returns True if all the characters in the variable are letters and False if not. Here, I have used the negation ‘not’ so that if the expression is True, then a message box is displayed with an error. If it is false, then the execution of the code is continued.

The screenshot shows a Python application window with three tabs: 'Supplier Name', 'Supplier ID', and 'Supplier Email'. The 'Supplier Name' tab is active, displaying the value '1' in a text input field. A tooltip box points to this field with the text: 'Incorrect data type used to input the supplier's name'. Below the tabs, a message box titled 'Incorrect Input' displays the error message 'Please enter a valid name'. A tooltip box points to this message box with the text: 'This code validates the user's input and ensures that no integer values can be accepted as the Supplier's name.' In the 'Supplier ID' tab, the value '5' is entered in the text input field. A tooltip box points to this field with the text: 'Here, the supplier name is alphabetical but the supplier Email is in an integer format.' In the 'Supplier Email' tab, the value '4' is entered in the text input field. A tooltip box points to this message box with the text: 'When I click on add, it shows me an error telling me to add a valid email address.'

Testing the Supplier ADD function:

From the design section, I had created several types of test data – normal, boundary and erroneous.

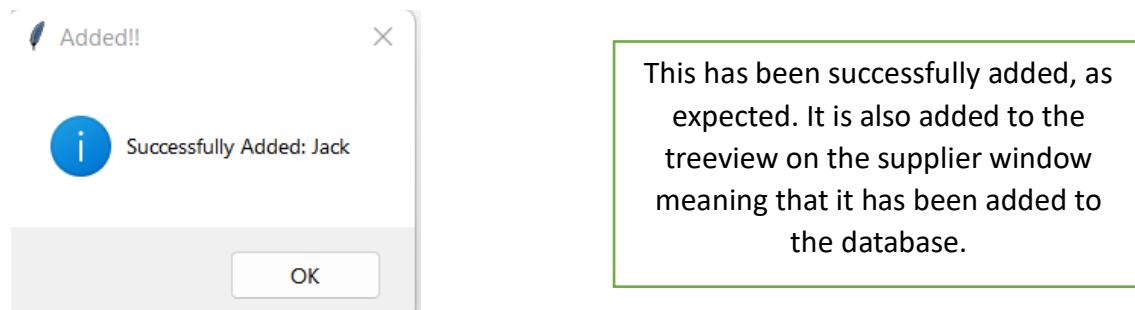
I will be using this to check that my add function really works.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>
1.	Adding supplier details	Supplier Name: Jack ID : 7 Email: Jack.Smith@gmail.com

Supplier Name	Jack
Supplier ID	7
Supplier Email	Jack.Smith@gmail.com

From the design section, I expect this to be successfully added.

Result using normal data:



Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
3	Bob	wateraid2.8@gmail.com
4	Nijan	nijankannathasan@gmail.com
7	Jack	Jack.Smith@gmail.com

This code below shows how to display the data from the supplier table to the treeview. Here, I have used SQL query to fetch all the records from the supplier table which are then added to the treeview using the for loop.

```

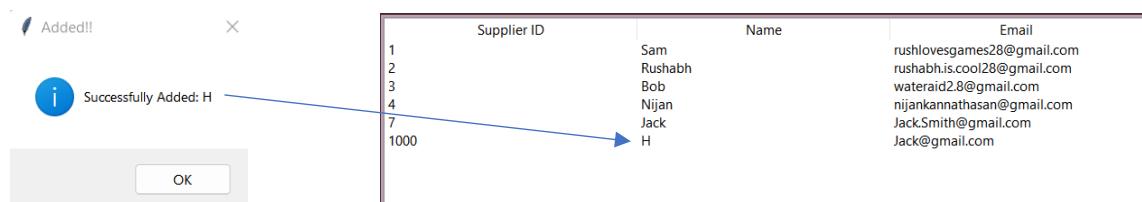
def display_data(self): # display the data stored in the database
    con = sqlite3.connect(database = r'sms.db') # connecting it to the database
    cur = con.cursor() # database cursor
    try:
        cur.execute("select * from supplier") # sql query
        rows = cur.fetchall() # get all the records
        self.SupplyTable.delete(*self.SupplyTable.get_children())
        for row in rows: # for loop
            self.SupplyTable.insert('', END, values = row)

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

Using the boundary data:

<u>Boundary</u>		
Name: H	Supplier Name	H
ID: 1000	Supplier ID	1000
Email: Jack@gmail.com	Supplier Email	Jack@gmail.com



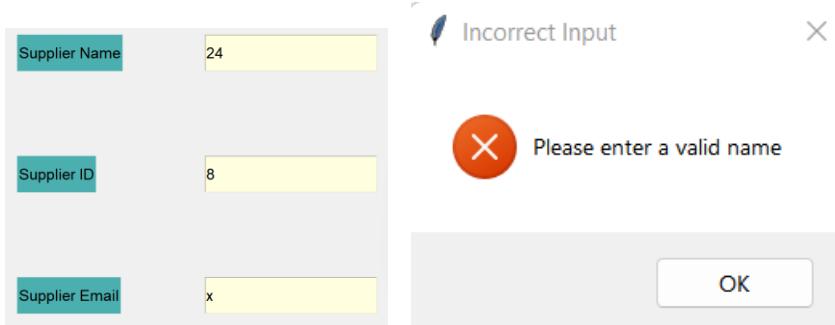
Erroneous

The program works for the boundary data as expected.

Supplier Name	24
Supplier ID	Supplier
Supplier Email	x

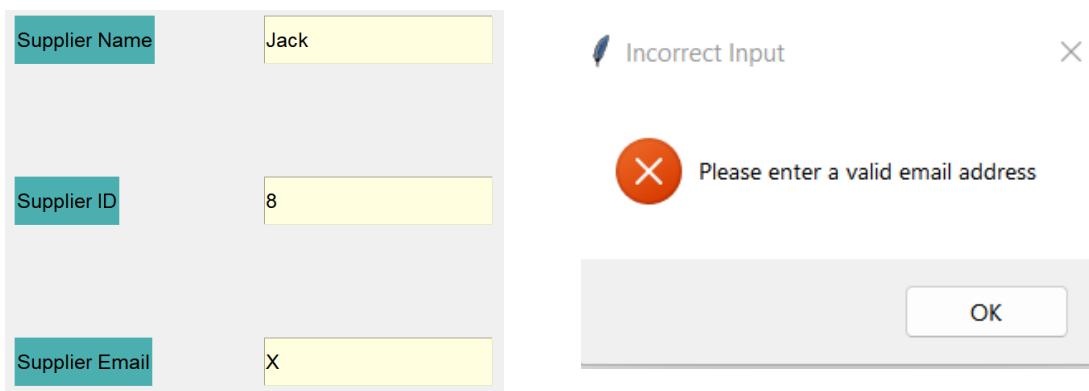
The program doesn't allow this input which is exactly what we wanted. Here, the ID is in the wrong format so it cannot be accepted.

If the ID was in the right format:



Here, the Supplier ID is in the right format, but the Supplier Name is in the wrong data type so a message is shown.

If the Supplier Name and the ID are in the right data type but the email is wrong, then it should show a message saying the email format is wrong.



Here, it shows a message telling us to enter a valid email address.

All of the tests for the add function have worked.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual results</u>
1.	Adding supplier details	Supplier Name: Jack ID : 7 Email: Jack.Smith@gmail.com	Name: H ID: 1000 Email: Jack@gmail.com	Name: 24 ID: Supplier Email: X	The Normal and Boundary data should be added to the supplier	Pass

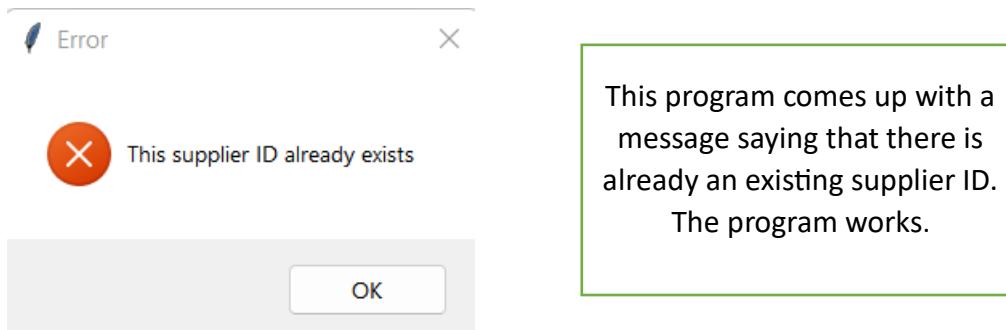
					table but not the Erroneous data	
--	--	--	--	--	-------------------------------------------	--

2.	Ensuring that the same Supplier ID is not added	N/A	N/A	Existing supplier ID's	The program should not allow existing ID's to be added – hence an error should be created
----	-------------------------------------------------	-----	-----	------------------------	-------------------------------------------------------------------------------------------

The screenshot shows a 'Supplier details' form on the left and a database table on the right. The form fields are: Supplier Name (Jack), Supplier ID (7), and Supplier Email (Jack@gmail.com). Below the form are buttons: Add, Modify, Delete, and Empty. A cursor points from the 'Delete' button towards the database table. The database table has columns: Supplier ID, Name, and Email. It contains the following data:

Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
3	Bob	wateraid28@gmail.com
4	Nijan	nijankannathasan@gmail.com
5	Jack	JackSmith@gmail.com
6	H	Jack@gmail.com
7		

As you can see above, I am entering the ID to be as 7 – which already exists.



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual results</u>
--------------------	-------------------------	---------------	-----------------	------------------	-------------------------	-----------------------

2.	Ensuring that the same Supplier ID is not added	N/A	N/A	Existing supplier ID's	The program should not allow existing ID's to be added – hence an error should be created	Pass – Comes up with a message showing there is already an existing supplier ID
----	-------------------------------------------------	-----	-----	------------------------	-------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------

```
#####
select function #####
def get_suppliers(self, suppliers): # this gets the row of data
    self.focus = self.SupplyTable.focus()
    self.data = self.SupplyTable.item((self.focus))
    row = self.data['values']
    self.var_supplier_id.set(row[0]), # indexing starts from 0
    self.var_supplier_name.set(row[1]),
    self.var_supplier_email.set(row[2])
```

This code above allows us to retrieve the selected row in the treeview and allows it to be populated on to the corresponding entry forms. Here, the focus method gets the focused item in the table that has been selected which then gets the values of those records.

The set method has been used to update the values of the variables with the values of the record that the user has selected.

To ensure that the mouse and the program work together, we have to use a binding method as show below.

```
self.SupplyTable.bind("<ButtonRelease-1>", self.get_suppliers)
```

This code links the function (get_suppliers) with the left button – so that when the user releases the left button, the function will be called. This function can be called many times as long as the user clicks on one of the records in the treeview and release the left button of the mouse.

Testing: Populating supplier details into entry boxes from treeview

3.	If I click on one of the treeview records, it	Middle record from the treeview	First or last treeview record	None of the records	If the user clicks on existing records, then it
----	-----------------------------------------------	---------------------------------	-------------------------------	---------------------	-------------------------------------------------

	should show up on the entry forms.			should show up in the corresponding entry boxes. If none of the records are clicked, then nothing should show up.
--	------------------------------------	--	--	-------------------------------------------------------------------------------------------------------------------

Normal Data

Supplier details		Supplier ID	Name	Email
Supplier Name	Bob	1	Sam	rushlovesgames28@gmail.com
		2	Rushabh	rushabh.is.cool28@gmail.com
Supplier ID	3	3	Bob	wateraid2.8@gmail.com
		4	Nijan	nijankannathasan@gmail.com
Supplier Email	wateraid2.8@gmail.com	7	Jack	Jack.Smith@gmail.com
		1000	H	Jack@gmail.com

The entry forms are successfully filled up with the values of one of the middle records selected.

Boundary data

Supplier details		Supplier ID	Name	Email
Supplier Name	Sam	1	Sam	rushlovesgames28@gmail.com
		2	Rushabh	rushabh.is.cool28@gmail.com
Supplier ID	1	3	Bob	wateraid2.8@gmail.com
		4	Nijan	nijankannathasan@gmail.com
Supplier Email	rushlovesgames28@gma	7	Jack	Jack.Smith@gmail.com
		1000	H	Jack@gmail.com

Supplier details	
Supplier Name	H
Supplier ID	1000
Supplier Email	Jack@gmail.com

	Supplier ID	Name	Email
1		Sam	rushlovesgames28@gmail.com
2		Rushabh	rushabh.is.cool28@gmail.com
3		Bob	wateraid2.8@gmail.com
4		Nijan	nijankannathasan@gmail.com
7		Jack	Jack.Smith@gmail.com
1000	H		Jack@gmail.com

When the first and the last records have been clicked, they successfully show up on the entry fields.

Erroneous

Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
3	Bob	wateraid2.8@gmail.com
4	Nijan	nijankannathasan@gmail.com
7	Jack	Jack.Smith@gmail.com
1000	H	Jack@gmail.com

If I click on a non-existing record but within the treeview, I get an error:

```
self.var_supplier_id.set(row[0]), # indexing starts from 0
IndexError: string index out of range
```

This means that the record I have clicked is not within the range of the existing records and so the entry fields are empty.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual results</u>
3	If I click on one of the treeview records, it should	Middle record from the treeview	First or last treeview record	None of the records	If the user clicks on existing records, then it should show up in	Pass – entry boxes show details when

	show up on the entry forms.				the corresponding entry boxes. If none of the records are clicked, then nothing should show up.	existing record in treeview is selected
--	-----------------------------	--	--	--	-------------------------------------------------------------------------------------------------	-----------------------------------------

Modify function

```
#####
# modifying function #####
def modify(self): # creating a function to change the supplier details to the database
    con = sqlite3.connect(database= r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        if self.var_supplier_id.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter an existing supplier ID", parent=self.wind)
            # validation. Cannot modify supplier details without supplier ID
        else:
            cur.execute("Select * from supplier where ID=?", (self.var_supplier_id.get(),))
            row = cur.fetchone() # fetches the next row of data
            if row == None: # will only update if I have an existing supplier ID
                messagebox.showerror("Error", "This supplier ID is invalid") # validation
            else:
                # this updates the supplier details
                cur.execute("Update supplier SET Name=?, Email=? WHERE ID=?", # we need a where statement for our query
                           (self.var_supplier_name.get(),
                            self.var_supplier_email.get(),
                            self.var_supplier_id.get(),
                           ))
                con.commit() # commit any changes
                messagebox.showinfo("Updated!!", "Successfully updated Supplier!!", parent=self.wind)
    
```

This code here is used to modify the record of an existing supplier. Here, only the Name or the Email can be changed as the Supplier ID is the primary key which should not be changed.

The program checks that if there is an existing supplier ID, then the record is allowed to be modified. If not, then a message is displayed showing that the supplier ID is invalid.

When updating the records, I have used the ID field as part of the where clause since it is a primary key and will be able to identify unique supplier records.

Testing: modify function

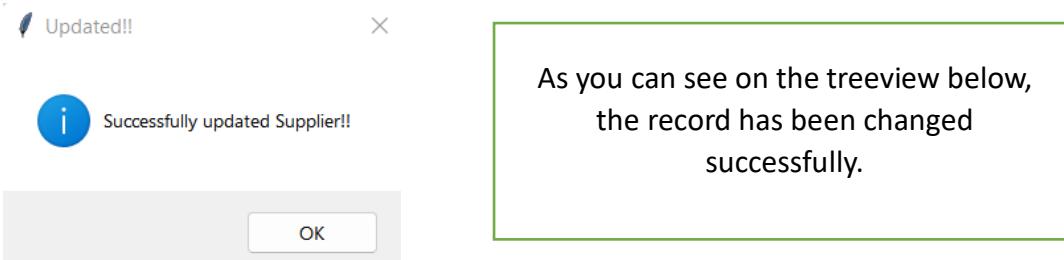
Normal

4.	Updating supplier details	Name: Jack changed to John Email: John@gmail.com	Name: J Email: J@gmail.com	Name: 2 Email: 2@gmail.com	The normal and boundary data will get updated but not the erroneous data
----	---------------------------	-----------------------------------------------------	-------------------------------	-------------------------------	--------------------------------------------------------------------------

Supplier details	
Supplier Name	John
Supplier ID	7
Supplier Email	John@gmail.com

Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
3	Bob	wateraid2.8@gmail.com
4	Nijan	nijankannathasan@gmail.com
7	Jack	JackSmith@gmail.com
1000	H	Jack@gmail.com

Here, I have selected a record, and have changed the name to John and the email to John@gmail.com



Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
3	Bob	wateraid2.8@gmail.com
4	Nijan	nijankannathasan@gmail.com
7	John	John@gmail.com
1000	H	Jack@gmail.com

Boundary

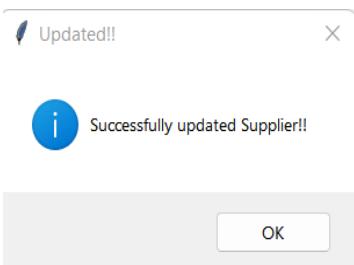
Supplier details	
Supplier Name	H
Supplier ID	1000
Supplier Email	Jack@gmail.com

Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
3	Bob	wateraid2.8@gmail.com
4	Nijan	nijankannathasan@gmail.com
7	John	John@gmail.com
1000	H	Jack@gmail.com

Here, I have changed the Name and the Email of the Supplier.

Supplier Name	J
Supplier ID	1000
Supplier Email	J@gmail.com

	Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com	
2	Rushabh	rushabh.is.cool28@gmail.com	
3	Bob	wateraid28@gmail.com	
4	Nijan	nijankannathasan@gmail.com	
7	John	John@gmail.com	
1000	J	J@gmail.com	



This also works as the valid data has been successfully updated.

Erroneous data:

Since the supplier name cannot be a number, the program should return an error.

Supplier Name	2
Supplier ID	1000
Supplier Email	2@gmail.com

i Successfully updated Supplier!!

But instead, the program allows it.

I have looked back at the code and noticed that there is no validation used for the modify function. Since we are updating the function and not adding, I need to ensure that there is validation for not just the add function.

```
def modify(self): # creating a function to change the supplier details to the database
    con = sqlite3.connect(database='r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        if self.var_supplier_id.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter an existing supplier ID", parent=self.wind)
            # validation. Cannot modify supplier details without supplier ID
        else:
            cur.execute("Select * from supplier where ID=?", (self.var_supplier_id.get(),))
            row = cur.fetchone() # fetches the next row of data
            if row == None: # will only update if I have an existing supplier ID
                messagebox.showerror("Error", "This supplier ID is invalid") # validation
                # checks that the supplier name only has alphabets
            if not self.var_supplier_name.get().isalpha():
                messagebox.showerror("Incorrect Input", "Please enter a valid name", parent=self.wind)
                return

            # ensures that the email format follows a specific pattern
            email_pattern = r'^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
            if not re.match(email_pattern, self.var_supplier_email.get()):
                messagebox.showerror("Incorrect Input", "Please enter a valid email address", parent=self.wind)
                return
```

I have added the code for the email and the name validation to the modify function.

The screenshot shows a Windows application window titled "Supplier details". It contains three input fields: "Supplier Name" (containing "2"), "Supplier ID" (containing "1000"), and "Supplier Email" (containing "2@gmail.com"). To the right of the application is a screenshot of a SQLite database table with columns "Supplier ID", "Name", and "Email". The table has rows numbered 1 to 7. Row 1000 is highlighted in blue, matching the value in the "Supplier ID" field of the application. A validation message box titled "Incorrect Input" is displayed, stating "Please enter a valid name".

When I select that record, and press modify again it shows me a message box displaying that the name needs to be valid.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual results</u>
4.	Updating supplier details	Name: Jack changed to John Email: John@gmail.com	Name: J Email: J@gmail.com	Name: 2 Email: 2@gmail.com	The normal and boundary data will get updated but not the	At first it failed, but then I wrote the code for validating the supplier name

					erroneous data	and email and it worked.
--	--	--	--	--	----------------	--------------------------

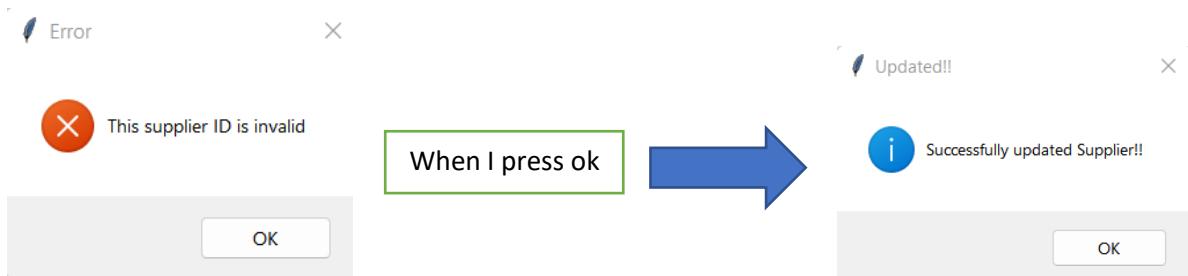
Testing: whether the ID can be updated

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>
5.	Updating supplier ID	N/A	N/A	Adding a different supplier ID	Message box with an error – since this is not an existing ID

Supplier ID	Name	Email
1	Sam	rushilovesgames28@gmail.com
2	Rushabh	rushabhiscool28@gmail.com
3	Bob	wateraid28@gmail.com
4	Nijan	nijankannathasan@gmail.com
7	jack	2@gmail.com
1000	2	2@gmail.com

Here, I have selected a record from the treeview and have changed the supplier ID from 7 to 8.

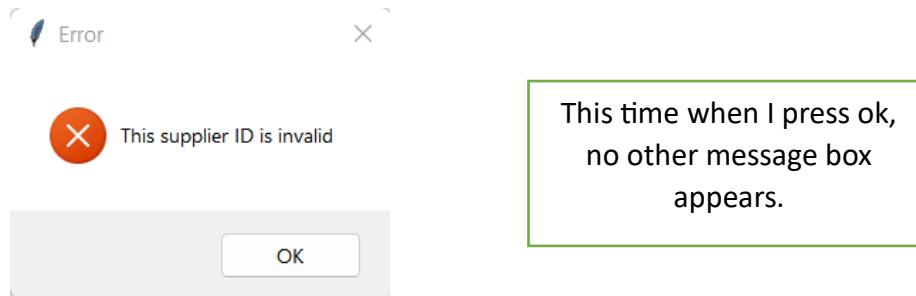
Output



The supplier ID doesn't get updated in the supplier table but the message box shows that the Supplier has been updated.

To get rid of this I have added a return statement – this is used to exit the function if the supplier ID entered by the user is not found in the supplier table. That way, only the message box showing the error will appear.

```
cur.execute("Select * from supplier where ID=?", (self.var_supplier_id.get(),))
row = cur.fetchone() # fetches the next row of data
if row == None: # will only update if I have an existing supplier ID
    messagebox.showerror("Error", "This supplier ID is invalid") # validation
    return
```



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>5.</u>	Updating supplier ID	N/A	N/A	Adding a different supplier ID	Message box with an error – since this is not an existing ID	Pass – though a message box showing that the supplier was updated was shown. I have amended my code by adding the return statement.

Testing: Delete function

```
#####
    Delete function #####
def delete_record(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_supplier_id.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter an existing supplier ID", parent=self.wind)
        else:
            cur.execute("Select * from supplier where ID = ?", (self.var_supplier_id.get(),))
            row = cur.fetchone()
            if row == None:
                messagebox.showerror("Error", "This supplier ID is invalid.")
            else:
                ask = messagebox.askyesno("Confirmation", "Do you want to delete this record?", parent = self.wind)
                if ask == True:
                    cur.execute("Delete from supplier where ID=?",(self.var_supplier_id.get(),)) # the ID is the primary key
                    con.commit()
                    messagebox.showinfo("Delete", "Successfully deleted", parent = self.wind)
                    self.empty() # calls the empty function
    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)
```

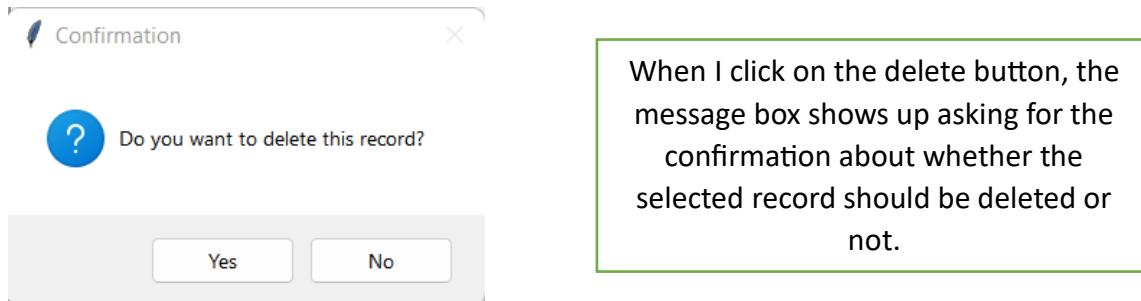
This is the function to delete a supplier record. Here, the user will select a record and click on the delete button. A message box will appear asking for the confirmation of that record to be deleted. If the input is yes, then the SQL query to delete the selected record is performed.

6.	Deleting supplier ID	Deleting one of the middle records	Deleting the last record	Selecting a record out of range	Existing supplier records that have been selected, should get deleted.
-----------	----------------------	------------------------------------	--------------------------	---------------------------------	------------------------------------------------------------------------

	Supplier ID	Name	Email
1		Sam	rushlovesgames28@gmail.com
2		Rushabh	rushabh.is.cool28@gmail.com
3		Bob	wateraid2.8@gmail.com
4		Nijan	nijankannathasan@gmail.com
7		jack	2@gmail.com
1000		2	2@gmail.com

Normal data:

Here I have selected the 5th record from the supplier table.



If I press yes, then it will delete the record. If I press no, it will not delete the record

Here below, I have deleted the record by pressing yes on the messagebox.



Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
3	Bob	wateraid2.8@gmail.com
4	Nijan	nijankannathasan@gmail.com
1000	2	2@gmail.com

As you can see here, the record that has supplier ID 7 has been deleted from the table as it is no longer exists in the treeview.

Boundary data:

Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
3	Bob	wateraid2.8@gmail.com
4	Nijan	nijankannathasan@gmail.com
1000	2	2@gmail.com

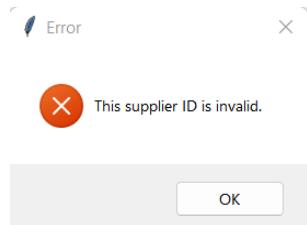


Once again, this message box appears, showing that the code works regardless of what position of the supplier record is selected.

Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
3	Bob	wateraid2.8@gmail.com
4	Nijan	nijankannathasan@gmail.com

Note that the record that was selected had a supplier name of 2 which isn't valid. It stayed in the table as I added validation after I updated the record.

Erroneous:



If I don't select anything, then it shows me with a message box error saying that the supplier ID is invalid. This is because by default, the supplier ID variable will be equal to 0 as it is an IntVar and since there is no supplier ID with 0, it will show an error.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
6.	Deleting supplier ID	Deleting one of the middle records	Deleting the last record	Selecting a record out of range	Existing supplier records that have been selected, should get deleted.	Pass: Deletes existing selected supplier records

Search function

This code here allows me to search through a supplier record using the Tkinter combobox. Here, I have used two variables – self.var_searchuse() which stores the variable of what is being used to search through the record and self.var_searchtxt() which stores the value of the user's input.

If the value of the self.var_searchuse is "select" which is the default option, then a message error appears telling the user to select one of the options.

Here, for the options, though the supplier ID will give us the unique record instantaneously, I have added the other 2 values as they are much easier to remember than the supplier ID by the user hence acting as a secondary key.

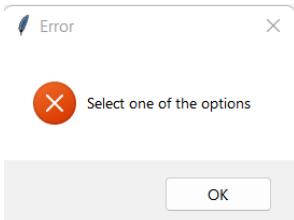
```
#####
# Search function #####
def search(self):
    con = sqlite3.connect(database=r'sms.db') # connecting it to the database
    cur = con.cursor() # database cursor
    try:
        if self.var_searchuse.get() == "select": # the user should select either name, email or ID
            messagebox.showerror("Error", "Select one of the options", parent = self.wind) # error due to incorrect search
        elif self.var_searchtxt.get() == "":
            messagebox.showerror("Error", "Please type what you want to search", parent=self.wind)
        else:
            cur.execute("select * from supplier where "+self.var_searchuse.get()+" LIKE '%"+self.var_searchtxt.get()+"%'") # whatever we search can be in any position
            rows = cur.fetchall() # get all the records
            if len(rows)!=0:
                self.SupplyTable.delete(*self.SupplyTable.get_children())
                for row in rows:
                    self.SupplyTable.insert('', END, values=row)

            else:
                messagebox.showerror("Error", "Nothing found", parent = self.wind)

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # this gives us any errors that may occur when coding
```

Here, I have not selected an option as it shows the default option of "select".

Output when I click the search button:



This code successfully works as the program cannot filter out the records based on no option being selected.

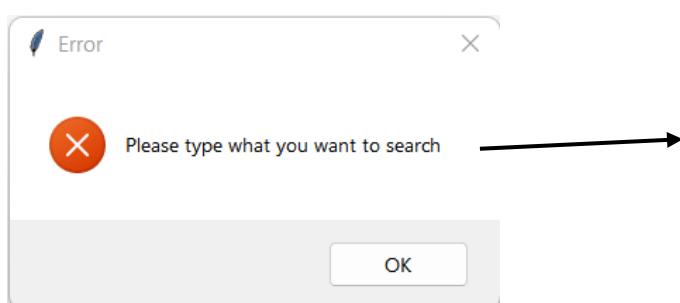
<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
7.	Message error should occur if the value of the options to select is "select" (the default option).	N/A	N/A	Not selecting the 3 options (ID, Name and Email)	Error asking for an option to be selected	Pass – error asking for an option to be selected

```
try:
    if self.var_searchuse.get() == "select": # the user should select either name, email or ID
        messagebox.showerror("Error", "Select one of the options", parent = self.wind) # error due to incorrect search
    elif self.var_searchtxt.get() == "":
        messagebox.showerror("Error", "Please type what you want to search", parent=self.wind)
```

These if statements from the search function mean that if an option is selected, but the user doesn't input what they want to search for, then another error should occur.

Test:

Here, I have selected the name as the option.



This part of the program has worked as it ensures that the user has given input to the text otherwise the program won't have any pattern that the records should have. For example, typing just the letter 'e' in the search bar with the option as the name means that the supplier should have the letter 'e' within their name.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
8.	Message error should occur if nothing has been typed into the search bar	N/A	N/A	Nothing	Error asking the user to type something	Pass – error asking for the user to type what they want to search

```

else:
    cur.execute("select * from supplier where "+self.var_searchuse.get() +" LIKE '%" +self.var_searchtxt.get()+"%'") # whatever we search can be in any position
    rows = cur.fetchall() # get all the records
    if len(rows)!=0:
        self.SupplyTable.delete(*self.SupplyTable.get_children())
        for row in rows:
            self.SupplyTable.insert('', END, values=row)

    else:
        messagebox.showerror("Error", "Nothing found", parent = self.wind)

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # this gives us any errors that may occur when coding

```

```
cur.execute("select * from supplier where "+self.var_searchuse.get() +" LIKE '%" +self.var_searchtxt.get()+"%'")
```

This line of code above executes a query to search for the supplier records that match the condition set by the user. Here, I have used the LIKE operator to search for records that contain specific characters.

The self.var_searchtxt.get() method gets the text input from the user which is then matched with records that contain those text or partial texts in the selected column.

The self.var_searchuse.get() method fetches the column name to search for. This is concatenated with the 'LIKE' operator and the characters to be searched, so that the SQL query is made at run-time based on user input. This provides versatility as the column to look for and the characters to search for are not hard-coded, but instead they vary based on the user's input.

Testing: Search function

9.	Testing the search bar	Name: Test (Full name of the supplier) ID: 5 Email: Test@gmai l.com	Name: 's' ID: 5 Email: T	Name: TrialTest ID: 78 Email:TrialTest@gmail .com	For the normal data, there should only be 1 unique record found. For the boundary data, there might be more than 1 record that fulfils a partial input by the user. For the erroneous data, there should be no records found.
-----------	------------------------	------------------------------------------------------------------------------------	--------------------------------	------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Search Supplier

Name	Test	search
------	------	--------

	Supplier ID	Name	Email
1		Sam	rushlovesgames28@gmail.com
2		Rushabh	rushabh.is.cool28@gmail.com
3		Bob	wateraid2.8@gmail.com
4		Nijan	nijankannathasan@gmail.com
5		Test	Test@gmail.com

Normal data

	Supplier ID	Name	Email
5	Test		Test@gmail.com

When I clicked the search button, it has removed all the other records that don't fulfil the condition.

Here, only the records that have the name 'Test' are shown.

The code through checking if there are any rows returned that match the user's conditions. If true, then it clears all the contents of the treeview and then loops through the returned rows and inserts them into each row.

Checking with ID:

Search Supplier

ID	5	search
----	---	--------

Supplier ID	Name	Email
5	Test	Test@gmail.com

The program works for the ID as well.

Checking with email:

Search Supplier

Email	Test@gmail.com	search
-------	----------------	--------

Supplier ID	Name	Email
5	Test	Test@gmail.com

The code gives the same record regardless of what is being used as the input.

Boundary data

Search Supplier

Name	s	search
------	---	--------

Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
5	Test	Test@gmail.com

Here, as you can see there is more than 1 record as each of the supplier names have the letter 's'.

Search Supplier

ID	5	search
----	---	--------

	Supplier ID	Name	Email
5	Test		Test@gmail.com
15	Trial		Trial@gmail.com

Here, the number 5 is common in both records for the ID column, so both of them are visible.

Search Supplier

Email	<input type="text" value="T"/>	<input type="button" value="search"/>
-------	--------------------------------	---------------------------------------

	Supplier ID	Name	Email
3	Bob		wateraid2.8@gmail.com
4	Nijan		nijankannathasan@gmail.com
5	Test		Test@gmail.com
15	Trial		Trial@gmail.com

Here, all the emails have the letter 't' so they are shown.

Search Supplier

Email	<input type="text" value="tr"/>	<input type="button" value="search"/>
-------	---------------------------------	---------------------------------------

	Supplier ID	Name	Email
15	Trial		Trial@gmail.com

However, if I put one or more letter, then the program will take those letters consecutively, and search for records that have those consecutive letters. Above, the input is 'tr' and there is only one record that has 'tr' consecutively.

Erroneous

Search Supplier

Name	<input type="text" value="TrialTest"/>	<input type="button" value="search"/>
------	----------------------------------------	---------------------------------------

	Supplier ID	Name	Email
1		Sam	rushlovesgames28@gmail.com
2		Rushabh	rushabh.is.cool28@gmail.com
3		Bob	wateraid2.8@gmail.com
4		Nijan	nijankannathasan@gmail.com
5		Test	Test@gmail.com
15		Trial	Trial@gmail.com

Error

Nothing found

Here, when I press the search button, then I get an error saying there are no such records that fulfil the condition. Notice that the treeview remains the same and none of the records are cleared.

Search Supplier

ID

search

	Supplier ID	Name	Email
1		Sam	rushlovesgames28@gmail.com
2		Rushabh	rushabh.is.cool28@gmail.com
3		Bob	wateraid2.8@gmail.com
4		Nijan	nijankannathasan@gmail.com
5		Test	Test@gmail.com
15		Trial	Trial@gmail.com

Error

Nothing found

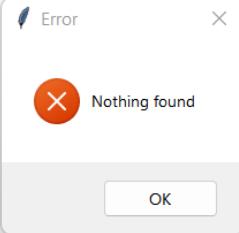
The same result occurs when I enter a non-existent supplier ID.

Search Supplier

Email

search

	Supplier ID	Name	Email
1		Sam	rushlovesgames28@gmail.com
2		Rushabh	rushabh.i.s.cool28@gmail.com
3		Bob	wateraid2.8@gmail.com
4		Nijan	nijankannathasan@gmail.com
5		Test	Test@gmail.com
15		Trial	Trial@gmail.com



Though the input has the words Trial and Test which form part of 2 different emails, because my input has put those words consecutively, it needs to find an email that follows this pattern – and none of the records match the criteria fully.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
9.	Testing the search bar	Name: Test (Full name of the supplier) ID: 5 Email: Test@gmail.com	Name: 's' ID: 5 Email: T	Name: TrialTest ID: 78 Email:TrialTest@gmail.com	For the normal data, there should only be 1 unique record found. For the boundary data, there might be more than 1 record that fulfils a partial input by the user. For the erroneous data, there should be	Pass – got the expected results.

					no records found.	
--	--	--	--	--	-------------------------	--

Client feedback:

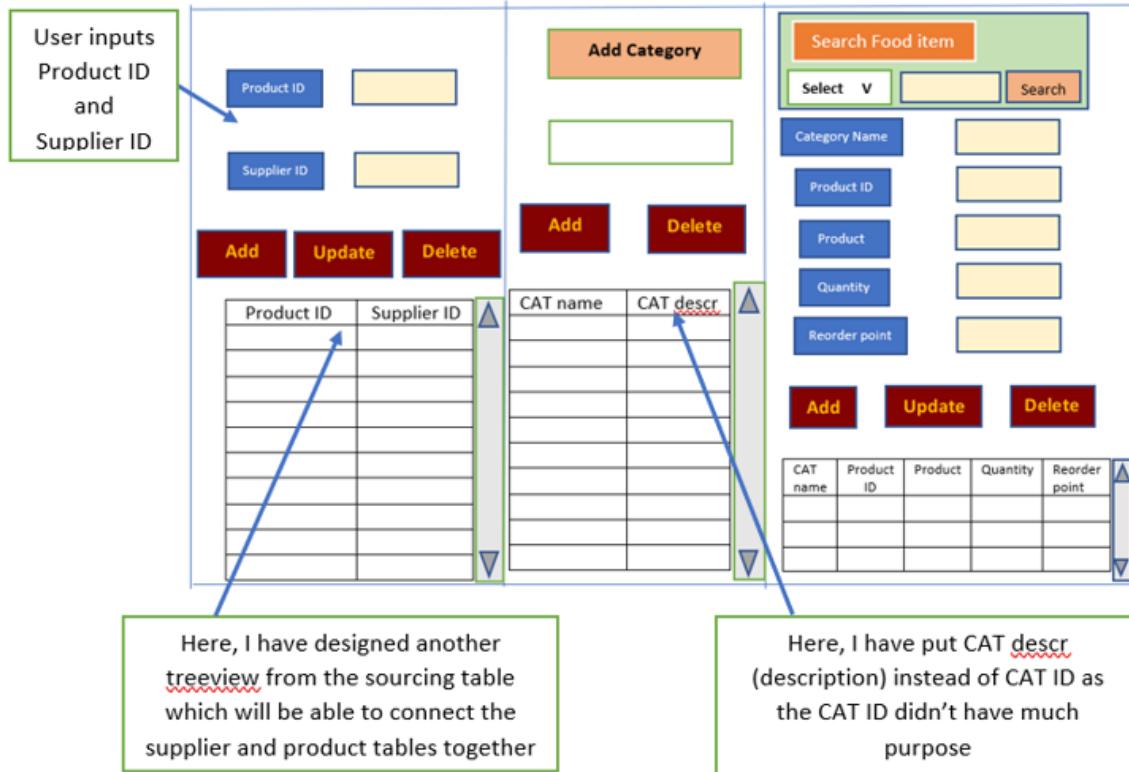
I have shown this Supplier Window to Parth to get any feedback from him.

Parth: "This looks really well organised – I really like the features that make this supplier window easy to use and overall the Supplier Window is user friendly. The tests that you have done have helped me understand the validations and what can data types should be used when using this window".

Comment: The client has approved of this Supplier Window development so I can move on to the Product Stage.

Stage 3: Product Window

For this stage, I will not be going in to depth about some of the features covered from the Supplier window such as the add, delete and update buttons and also the search bar.



This is from the design section. Before coding this window, I have decided to add an Update button to the category section so that the category description can be updated. I have decided to put the product search bar in the middle and the product treeview at the top. This will allow me to quickly glance over the 2 different Tkinter widgets.

Client feedback: I mentioned this to Parth who doesn't mind how the widgets are distributed as long as it is easy to use.

Creating the table

This is used to create the Category table. Here the CAT NAME is the primary key.

Here, I am creating a Product table where the Product ID is autoincremented. This means that each time a new record is being added, no matter what the user inputs as the Product ID, the system will increment the highest product ID number by 1.

The CAT_NAME from the Category table is a foreign key in the Product table which links the 2 tables together. Here, I have mentioned the CAT_NAME to be the foreign key and have referenced from which table it belongs in (category table).

```
cur.execute("CREATE TABLE IF NOT EXISTS Sourcing(Product_ID integer, Supplier_ID integer, foreign key (Product_ID) references Product(Product_ID), "
           "foreign key (Supplier_ID) references supplier(ID), primary key (Product_ID, Supplier_ID)) ",)
con.commit()
```

Here, I have created a Sourcing table to link the supplier table and the product table together. This table has a composite key that is made from 2 foreign keys as shown above – (Product_ID and the Supplier_ID).

```
# defining the variables
self.var_searchuse = StringVar()
self.var_searchtxt = StringVar()
self.var_product = StringVar() # name of product
self.var_quantity = IntVar()
self.var_product_id = IntVar()

self.var_cat_name = StringVar()
self.var_Reorder_Point = IntVar()
self.var_cat_descr = StringVar()

self.var_supplier_ID = IntVar()
```

The supplier ID variable will be part of the sourcing table.

The other variables above are for the Product and the Category table. Similarly, in the supplier window, these variables will store the input from the user when adding, updating, deleting and searching records.

```
#####
# Category Treeview #####
#####

myscrolly = Scrollbar(cat_frame, orient=VERTICAL)
myscrollx = Scrollbar(cat_frame, orient=HORIZONTAL)

self.Cat_Table = ttk.Treeview(cat_frame, columns=("CAT_NAME", "CAT_DESCR"), yscrollcommand=myscrolly.set, xscrollcommand=myscrollx.set)

myscrolly.pack(side=BOTTOM, fill=X)
myscrolly.pack(side=RIGHT, fill=Y)
myscrollx.config(command=self.Cat_Table.xview)
myscrolly.config(command=self.Cat_Table.yview)

self.Cat_Table.heading("CAT_NAME", text="CAT Name")
self.Cat_Table.heading("CAT_DESCR", text = "CAT Descr")
self.Cat_Table["show"] = "headings"

self.Cat_Table.column("CAT_NAME", width =140, stretch=NO) # fixed width
self.Cat_Table.column("CAT_DESCR", width = 140, stretch=NO) # fixed width

self.Cat_Table.pack(fill=BOTH, expand=1)
self.Cat_Table.bind("<ButtonRelease-1>", self.get_category)

self.display_data()
```

This code above makes the category treeview.

```

def add(self): # creating a function to add category details to the database
    con = sqlite3.connect(database=r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        if self.var_cat_name.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter a category",
                                 parent=self.wind) # validation.
    if not self.var_cat_name.get().isalpha():
        messagebox.showerror("Incorrect Input", "Please enter a valid category name", parent=self.wind)
        return
    else:
        cur.execute("Select * from category where CAT_NAME=?", (self.var_cat_name.get(),))
        row = cur.fetchone() # fetches the next row of data
        if row != None:
            messagebox.showerror("Error", "The Category already exists", parent = self.wind)
            # parent = self.wind ensures that even if an error occurs it will not return to the dashboard window
        else:
            cur.execute("Insert into category(CAT_NAME, CAT_DESCR) values(?,?)", (
                self.var_cat_name.get(),
                self.var_cat_descr.get()
            ))
            con.commit() # commit any changes
            messagebox.showinfo("Added!!", "Successfully Added Category!!", parent=self.wind)
            self.display_data()

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

This code above is used to add category names and the category description into the category table. Here, I have used the same validation rules as I did with the supplier window when adding new records.

Testing: Add function in Category table

10.	Adding a new record in the category table	Name: <u>Hersheys</u> Category: Chocolate	Name: H Category: C	Name: 20 Category: 25	Program should deny the erroneous data
-----	-------------------------------------------	----------------------------------------------	------------------------	--------------------------	----------------------------------------

Normal data

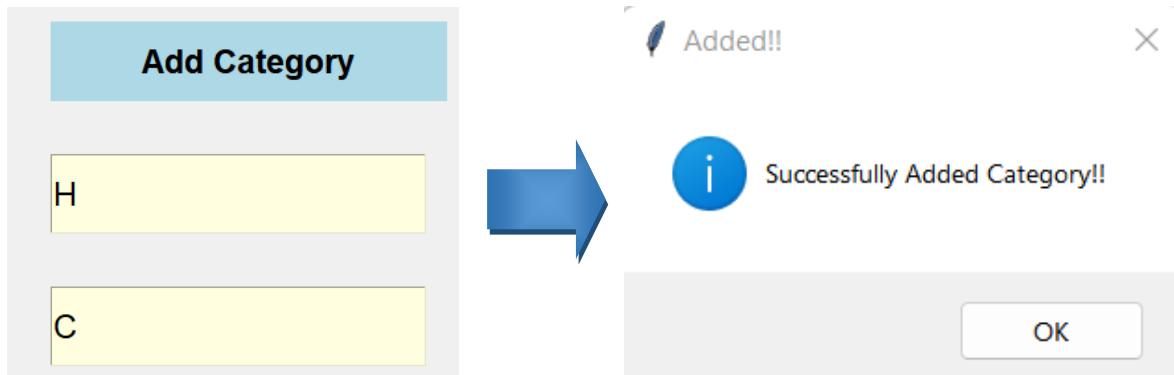


CAT Name	CAT Descr
Cheese	Dairy
Junk	Unhealthy
Vitamin_A	nutrition
Vitamin_B	nutrition
Vitamin_C	nutrition
Vitamin_D	nutrition
Vitamin_E	nutrition
Protein	nutrition
Cake	Dessert
Hersheys	Chocolate

Here the user's input is entered into the category table and the category treeview.

Notice that when there are lots of records added, you can use the scrollbars.

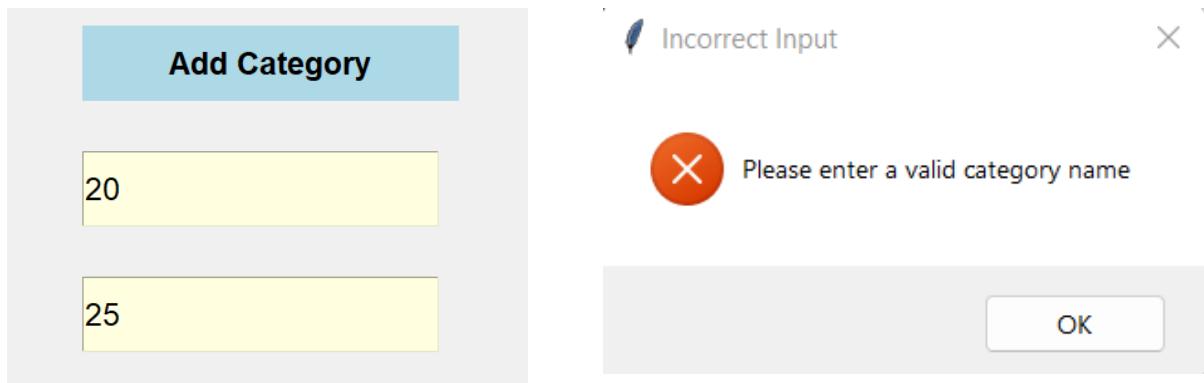
Boundary Data



CAT Name	CAT Descr
Junk	Unhealthy
Vitamin_A	nutrition
Vitamin_B	nutrition
Vitamin_C	nutrition
Vitamin_D	nutrition
Vitamin_E	nutrition
Protein	nutrition
Cake	Dessert
Hersheys	Chocolate
H	C

The boundary data is successfully added.

Erroneous data:



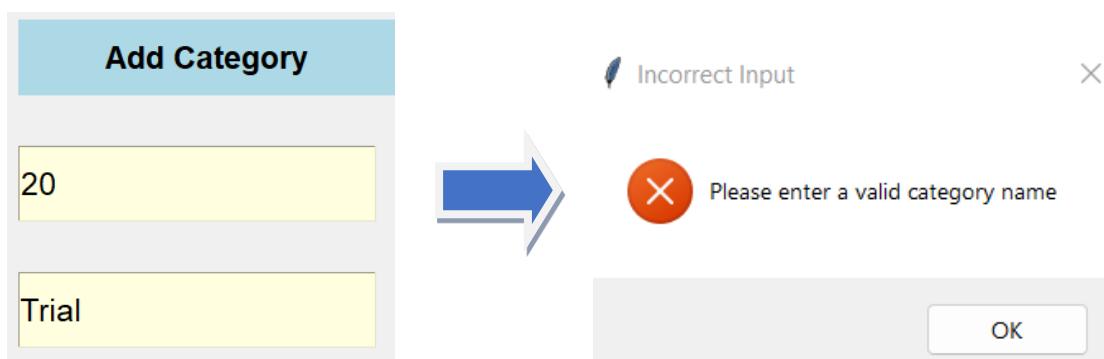
```
def add(self): # creating a function to add category details to the database
    con = sqlite3.connect(database=r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        if self.var_cat_name.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter a category",
                                 parent=self.wind) # validation.
        if not self.var_cat_name.get().isalpha():
            messagebox.showerror("Incorrect Input", "Please enter a valid category name", parent=self.wind)
            return

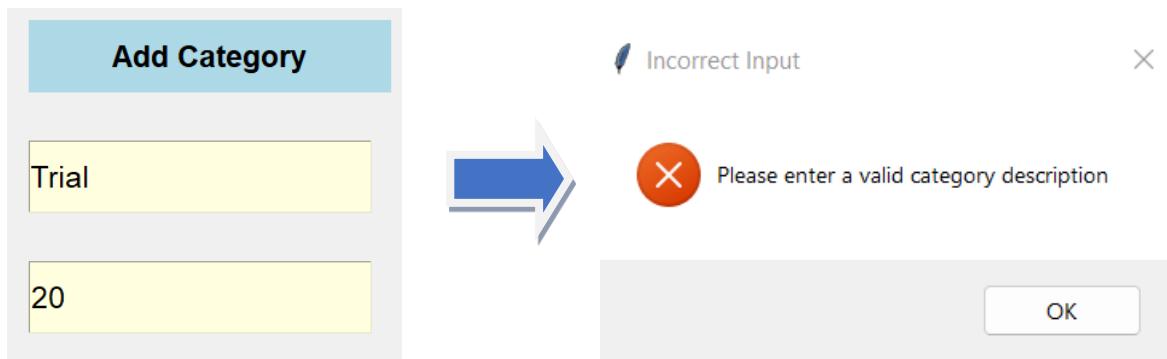
        if not self.var_cat_descr.get().isalpha():
            messagebox.showerror("Incorrect Input", "Please enter a valid category description", parent=self.wind)
            return
    
```

Above, I have added the validation for the category description so that it has to be a string.

```
else:
    cur.execute("Select * from category where UPPER(CAT_NAME)=?", (self.var_cat_name.get().upper(),))
    row = cur.fetchone() # fetches the next row of data
    if row != None:
        messagebox.showerror("Error", "The Category already exists", parent = self.wind)
        # parent = self.wind ensures that even if an error occurs it will not return to the dashboard window
        return
    else:
```

Here, I have used the upper function to convert the user input to Capitals as well as what is being used with the WHERE clause. This way, if the user enters the word junk and a category called 'Junk' with a capital J exists, the program will recognise that they are the same name as both category names will be converted to capitals so 'JUNK' is the same as 'JUNK'.





<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
10.	Adding a new record in the category table	Name: Hersheys Category: Chocolate	Name: H Category: C	Name: 20 Category: 25	Program should deny the erroneous data	Same as expected results

Category delete function

11.	Deleting a record from the category table	Deleting one of the middle records	Deleting the last record	Selecting a record out of range	Program should delete the existing category records selected
-----	-------------------------------------------	------------------------------------	--------------------------	---------------------------------	--------------------------------------------------------------

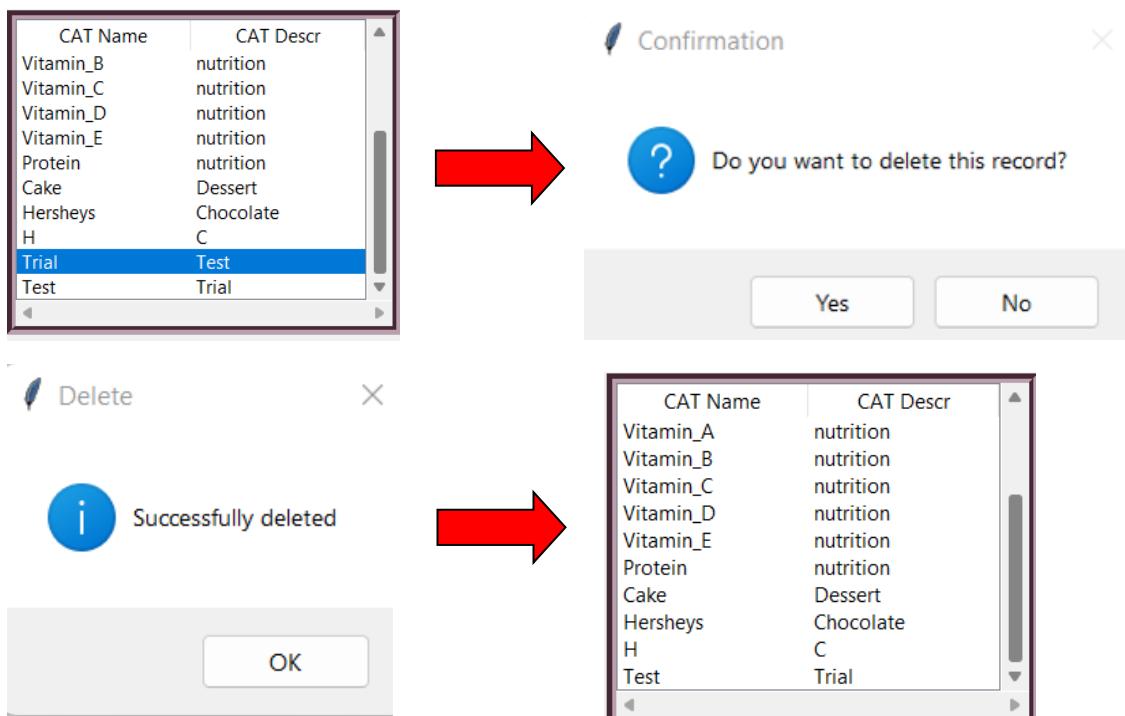
```

def delete_record(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_cat_descr.get() == "" and self.var_cat_name == "":
            messagebox.showerror("Incorrect Input", "Please choose a category and its description", parent=self.wind)
        else:
            cur.execute("Select * from category where CAT_NAME = ? ", (self.var_cat_name.get(),))
            row = cur.fetchone()
            if row == None:
                messagebox.showerror("Error", "This Category name is invalid")
            else:
                ask = messagebox.askyesno("Confirmation", "Do you want to delete this record?", parent = self.wind)
                if ask ==True:
                    cur.execute("Delete from category where CAT_NAME=? ", (self.var_cat_name.get(),)) # the ID is the primary key
                    con.commit()
                    messagebox.showinfo("Delete", "Successfully deleted", parent = self.wind)
                    self.CAT_empty()

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)
    
```

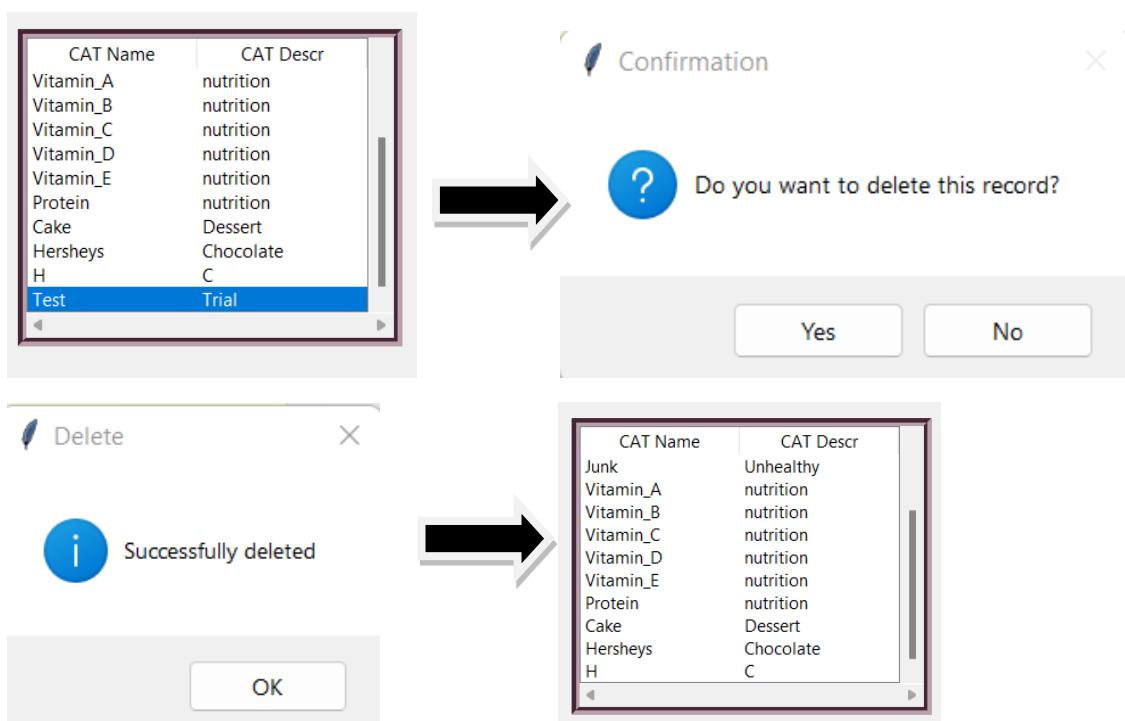
Testing: Delete function in Category table

Normal:



The normal data is being successfully deleted.

Boundary:



The last record from the category table has been deleted successfully.

Erroneous data:



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
11.	Deleting a record from the category table	Deleting one of the middle records	Deleting the last record	Selecting a record out of range	Program should delete the existing category records selected	Same as the expected results

Update Category function

```
def update_category(self):
    con = sqlite3.connect(database=r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        if self.var_cat_name.get() == "" and self.var_cat_descr.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter an existing Category", parent=self.wind)
        else:
            cur.execute("Select * from category where Upper(CAT_NAME)=?", (self.var_cat_name.get().upper(),))
            row = cur.fetchone() # fetches the next row of data
            if row == None:
                messagebox.showerror("Error", "This CAT name is invalid. Only existing category names can be updated", parent = self.wind) # validation
            else:
                cur.execute("Update category SET CAT_DESCR=? WHERE CAT_NAME=?", ( # we need a where statement for our query
                    self.var_cat_descr.get(),
                    self.var_cat_name.get()
                ))
                con.commit()
                messagebox.showinfo("Updated!!", "Successfully updated Category Description!!", parent=self.wind)
                self.display_data() # this helps to display the data. Once the update has been made, the changed data will be displayed automatically

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent = self.wind)
```

This code here is used to update the category description, based on the category selected.

Testing: Update function in Category table

12.	Updating a record from the category table	Updating one of the middle records	Updating the last record	Selecting a record out of range	Program should update the existing records selected
-----	-------------------------------------------	------------------------------------	--------------------------	---------------------------------	-----------------------------------------------------

Normal

The screenshot shows a table on the left with columns 'CAT Name' and 'CAT Descr'. The rows contain data such as Vitamin_B (nutrition), Vitamin_C (nutrition), Vitamin_D (nutrition), Vitamin_E (nutrition), Protein (nutrition), Cake (Dessert), Hersheys (Chocolate), H (C), Trial (Trial), and Test (error). The row for 'Trial' is highlighted with a blue selection bar. To the right, there is an 'Add Category' dialog box with two input fields: 'Trial' in the top field and 'test' in the bottom field.

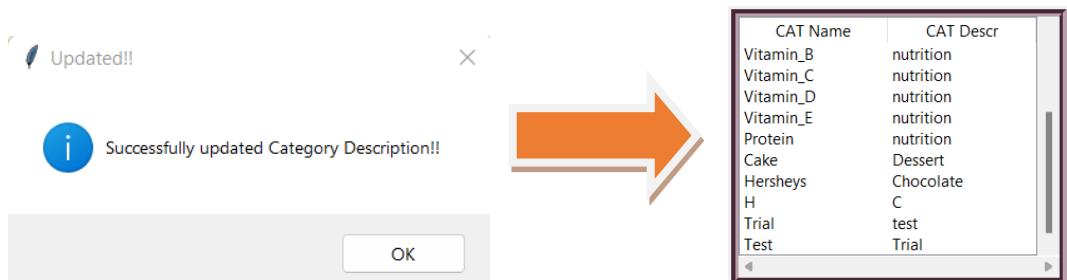
Here, I have changed the category description from Trial to Test.

The screenshot shows a success message 'Updated!!' with a blue exclamation mark icon. Below it is another message 'Successfully updated Category Description!!' with an information icon. A large blue arrow points from the original state to the new state. To the right, a table shows the updated data: 'Trial' now has 'test' as its description, while all other entries remain the same.

Boundary

The screenshot shows a table on the left with columns 'CAT Name' and 'CAT Descr'. The rows contain data such as Vitamin_B (nutrition), Vitamin_C (nutrition), Vitamin_D (nutrition), Vitamin_E (nutrition), Protein (nutrition), Cake (Dessert), Hersheys (Chocolate), H (C), Trial (test), and Test (error). The row for 'Test' is highlighted with a blue selection bar. To the right, there is an 'Add Category' dialog box with two input fields: 'Test' in the top field and 'Trial' in the bottom field.

Here, I have changed the name of the description of the category named 'Test' from error to Trial.



As you can see on the treeview above, the changes have been successfully reflected.

Erroneous



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
12.	Updating a record from the category table	Updating one of the middle records	Updating the last record	Selecting a record out of range	Program should update the existing records selected	Same as the expected results

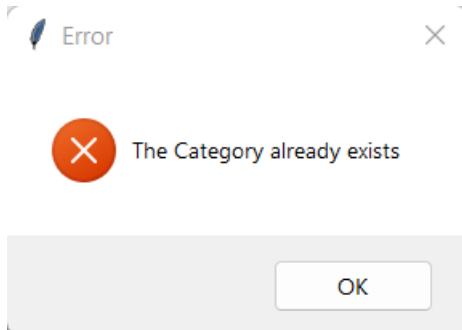
Testing: Adding an existing category name

13.	Adding an existing category name	N/A	N/A	Adding existing category name	Should get an error
-----	----------------------------------	-----	-----	-------------------------------	---------------------

The screenshot shows a software interface. On the left, a modal window titled "Add Category" contains a text input field with the value "Test" and a message box below it with the text "error". On the right, a table view displays a list of categories with columns "CAT Name" and "CAT Descr". The table includes rows for Vitamin_B, Vitamin_C, Vitamin_D, Vitamin_E, Protein, Cake, Hersheys, H, Trial, and Test. The row for "Test" has "test" in the "CAT Descr" column.

CAT Name	CAT Descr
Vitamin_B	nutrition
Vitamin_C	nutrition
Vitamin_D	nutrition
Vitamin_E	nutrition
Protein	nutrition
Cake	Dessert
Hersheys	Chocolate
H	C
Trial	test
Test	Trial

Here, I am adding an existing category name called “Test”. According to my program, I should receive an error saying this category name has already been taken.



Here, the program has successfully recognised that it cannot add an existing category. The program will allow the modification of the record of a given category but it will not allow the name to be category name to be changed, as it is a primary key for the category table.

To ensure that the program is not case sensitive, I have modified the code:

```

else:
    cur.execute("Select * from category where UPPER(CAT_NAME)=?", (self.var_cat_name.get().upper(),))
    row = cur.fetchone() # fetches the next row of data
    if row != None:
        messagebox.showerror("Error", "The Category already exists", parent = self.wind)
        # parent = self.wind ensures that even if an error occurs it will not return to the dashboard window
        return
    else:

```

Product table

```

def add_product(self): # creating a function to add product details to the table
    con = sqlite3.connect(database=r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        # if no input is found in the product entry box
        if self.var_product.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter a Product",
                                 parent=self.wind)
            return

        # checks whether the product input is in alphabets or not
        if not self.var_product.get().isalpha():
            messagebox.showerror("Incorrect Input", "Please enter the correct data type for product", parent=self.wind)
            return

        else:
            cur.execute("Select * from Product where Product =?", (self.var_product.get(),))
            row = cur.fetchone() # fetches the next row of data
            if row != None:
                messagebox.showerror("Error", "This Product already exists", parent = self.wind)
                return
            cur.execute("Select CAT_NAME from category where Upper(CAT_NAME) = ?",(self.var_cat_name.get().upper(),))
            fetch = cur.fetchone()
            if fetch == None:
                messagebox.showerror("Error", "Please add an existing category", parent = self.wind)

```

This part of the product function is used to add a record to the product table. Here, I have programmed all the errors that could occur.

Since the product table uses the category field from the category table, I have made sure that the program checks that there is an existing category before the user enters a record into the table.

```

            cur.execute("Select CAT_NAME from category where Upper(CAT_NAME) = ?",(self.var_cat_name.get().upper(),))

```

Here, I have ensured that the program is not case sensitive by converting the category fetched and the category out by the user into capitals.

```

        else:
            cur.execute(
                "Insert into Product( CAT_NAME, Product, Reorder_Point, Quantity) values(?, ?, ?, ?)",
                (
                    self.var_cat_name.get(),
                    self.var_product.get(),
                    self.var_Reorder_Point.get(),
                    self.var_quantity.get(),
                )
            )

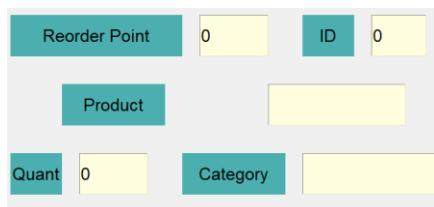
            con.commit() # commit any changes
            messagebox.showinfo("Added!!", "Successfully Added Product!!", parent=self.wind)
            self.display_Product_data()

        except Exception as ex:
            messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) #### shows an error

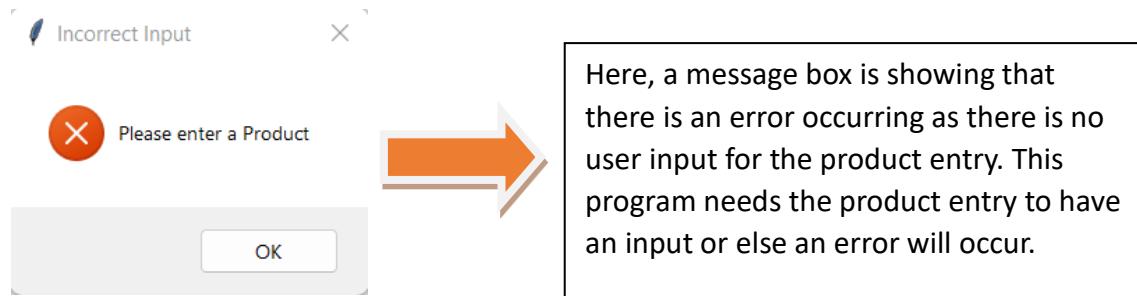
```

If the user's input is validated, then it is ready to be inserted into the table. Once again, I am using the .get() method to get the current values of the variables from the entry boxes which are then inserted in the corresponding order of the table.

14.	Adding a product record with no user input	N/A	N/A	No user input	An error should occur
------------	--------------------------------------------	-----	-----	---------------	-----------------------

Testing: Adding a product with no user input in Product table

Here, there is no user input, but just the default values for the entry boxes where the variables are set to Int type.



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
14.	Adding a product record with no user input	N/A	N/A	No user input	An error should occur	An error occurs

Testing: User adding an existing product (name) in the Product table

I have amended the code so that the program is not case-sensitive.

15.	Adding a record with existing product	N/A	N/A	User input of an existing product	Error should occur
------------	---------------------------------------	-----	-----	-----------------------------------	--------------------

Here, I have selected 'burger' as the existing product.

When I click on the add button:

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50
2	Junk	Doughnut	100	110
3	Junk	Pizza	30	50
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	Protein	Tofu	30	50
12	Protein	Almonds	50	60

Search Product

select

Add Update Delete

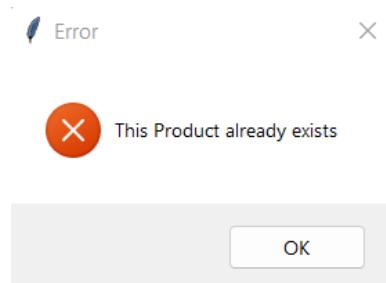
Reorder Point: 50 ID: 1

Product: Burger

Quant: 50 Category: Junk

Here, I have selected 'burger' as the existing product.

When I click on the add button:



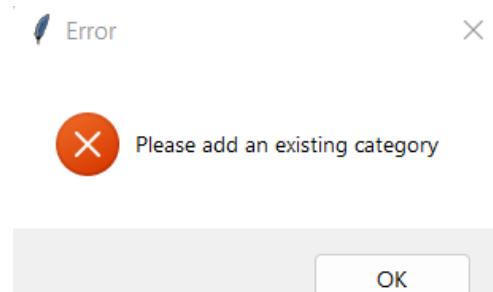
I get a message saying that the product already exists meaning that the function works.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>15.</u>	Adding a record with existing product	N/A	N/A	User input of an existing product	An error should occur	An error occurs

Testing when the user puts input only in the product entry box

16.	Adding a new product record with only product as the input	Product: Pasta	N/A	N/A	Error should occur asking for the category to be added
-----	------------------------------------------------------------	----------------	-----	-----	--------------------------------------------------------

	Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50	
2	Junk	Doughnut	100	110	
3	Junk	Pizza	30	50	
4	Milk	Cocunut	30	35	
5	Vitamin_A	Carrots	190	200	
6	Cheese	Cheddar	60	80	
7	Cheese	Mozzarella	40	50	
10	Milk	Paneer	30	0	
11	Protein	Tofu	30	50	
12	Protein	Almonds	50	60	



Here, the program asks me to add an existing category as expected in my test table.

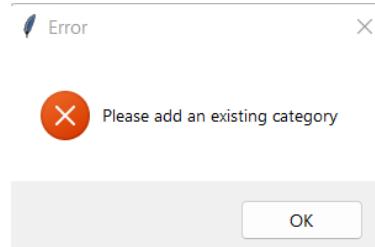
Test Number	What I'm testing	Normal	Boundary	Erroneous	Expected Results	Actual outcome
<u>16.</u>	Adding a new product record with only product as the input	Product: Pasta	N/A	N/A	Error should occur asking for the category to be added	An error occurs as expected - pass

Testing with only product and non-existing category name as input

	Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50	
2	Junk	Doughnut	100	110	
3	Junk	Pizza	30	50	
4	Milk	Cocunut	30	35	
5	Vitamin_A	Carrots	190	200	
6	Cheese	Cheddar	60	80	
7	Cheese	Mozzarella	40	50	
10	Milk	Paneer	30	0	
11	Protein	Tofu	30	50	
12	Protein	Almonds	50	60	

Here, I have entered the category "Error" and the Product to be pasta. The program should not accept this input due to the Error category not existing in the category table.

Result:



This message box shows that the program can check whether the user has input an existing category or not.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>17.</u>	Adding a new product record with only product and non-existing category name as the input	N/A	N/A	Category: Error Product: Pasta	An error should occur as Test is not an existing category	Pass – the program can check whether the user has input an existing category or not

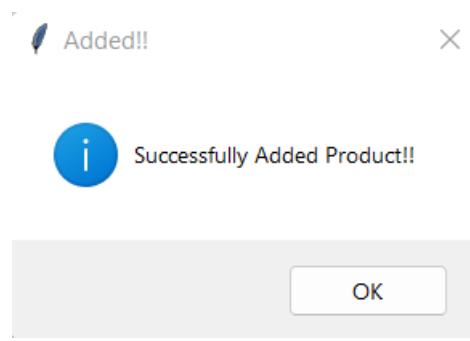
Testing: Entering the wrong product data type

<u>18.</u>	Entering the wrong product data type	N/A	N/A	Product: 25 Category: Junk	Program should reject the input
------------	--------------------------------------	-----	-----	-------------------------------	---------------------------------

The screenshot shows a software interface for managing products. At the top, there is a table with columns: Product ID, Category, Product, Reorder Point, and Stock Quantity. The data includes items like Pizza, Milk, and Carrots. Below the table is a search bar labeled 'Search Product' with a dropdown menu set to 'select'. Underneath the search bar are three buttons: 'Add', 'Update', and 'Delete'. The 'Add' button is highlighted. Below these buttons is a form with several fields. The 'Product' field contains the value '25'. Other visible fields include 'Reorder Point' (set to 0), 'ID' (set to 20), 'Quant' (set to 0), and 'Category' (set to 'Junk').

Here, I have entered the Product to be 25 which is an integer value and not a string. The program should not accept this input from the user.

Result:



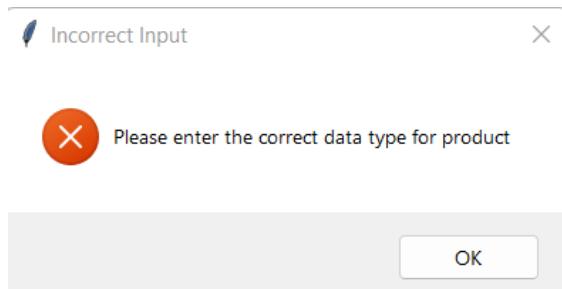
Product ID	Category	Product	Reorder Point	Stock Quantity
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	Protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60
22	Junk	25	0	0

Here, the program has added the invalid product data type to the product table and hence the product treeview as shown above.

```
def add_product(self): # creating a function to add product details to the table
    con = sqlite3.connect(database=r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        # if no input is found in the product entry box
        if self.var_product.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter a Product",
                                 parent=self.wind)
        return

        # checks whether the product input is in alphabets or not
        if not self.var_product.get().isalpha():
            messagebox.showerror("Incorrect Input", "Please enter the correct data type for product", parent=self.wind)
    return
```

Here, I have amended the code by adding a validation that checks whether the value that is stored in the product variable (self.var_product) has letters or not.



I have manually deleted the record from the DB browser. When I add the same input, I get the message shown on the left.

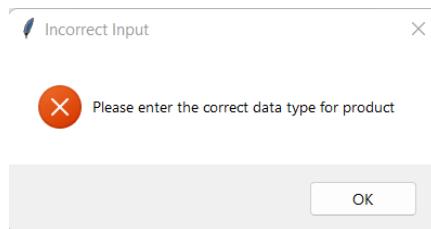
The added code helps validate the data type of the Product entered.

Product ID	Category	Product	Reorder Point	Stock Quantity
3	Junk	Pizza	30	50
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	Protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60

Search Product		
select	<input type="text"/>	search
Add	Update	Delete
Reorder Point	<input type="text" value="0"/>	ID <input type="text" value="0"/>
Product	<input type="text" value="25r"/>	
Quant	<input type="text" value="0"/>	Category <input type="text" value="Junk"/>

Here, I have put a mixture of an integer and a string to check whether the added code works for a mixture of different data types or not.

When I click the add button:

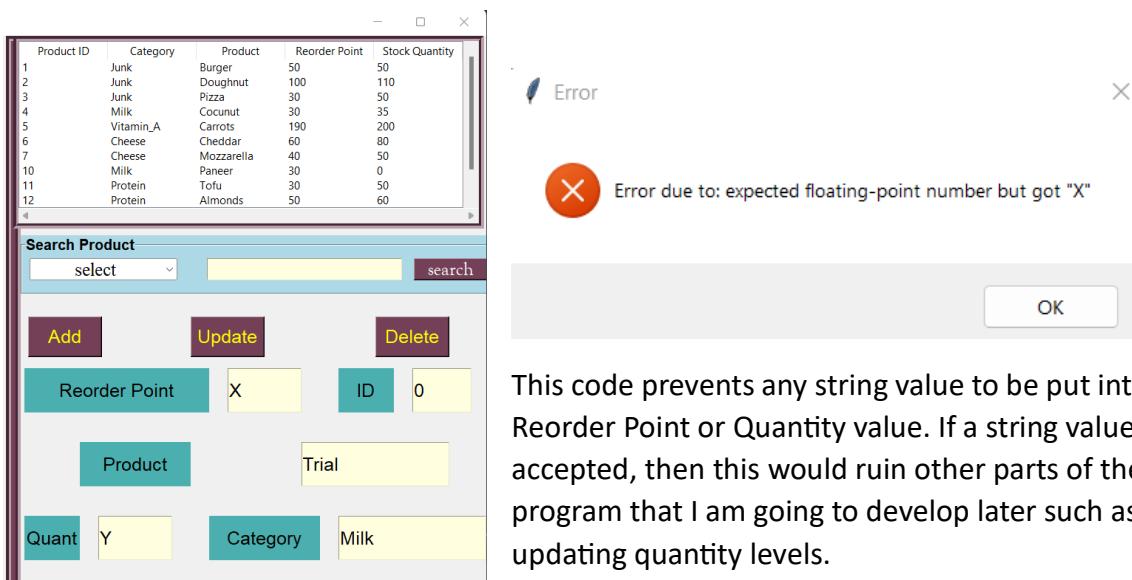


It shows me the same message as before meaning that the program will not allow a single integer character to be accepted as the product value.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
18.	Entering the wrong product data type	N/A	N/A	Product: 25 and 25r Category: Junk	Program should reject the input	Initially it failed as I didn't include validation for the product data type. The program now has validation for the product (name) entered.

Testing: Entering an invalid reorder point and quantity (current)

19.	Entering the wrong quantity and reorder point data type	N/A	N/A	Reorder point: x Quantity: Y	Program should reject the input due to wrong data type used
-----	---------------------------------------------------------	-----	-----	---------------------------------	-------------------------------------------------------------



This code prevents any string value to be put into the Reorder Point or Quantity value. If a string value was accepted, then this would ruin other parts of the program that I am going to develop later such as updating quantity levels.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>19.</u>	Entering the wrong quantity and reorder point data type	N/A	N/A	Reorder point: x Quantity: Y	Program should reject the input due to wrong data type used	Pass

Testing: Entering the wrong Product ID data type

20.	Entering the wrong product ID data type	N/A	N/A	X	Program should ignore the user input as the product ID is being auto-incremented
------------	-----------------------------------------	-----	-----	---	----------------------------------------------------------------------------------

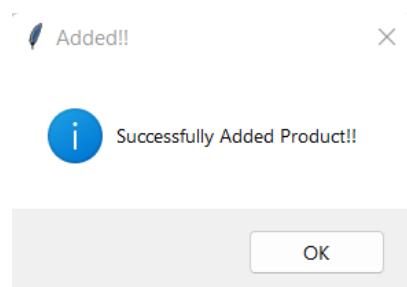
The screenshot shows a product management interface. On the left is a treeview of products:

Product ID	Category	Product	Reorder Point	Stock Quantity
3	Junk	Pizza	30	50
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	Protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60

The main area has a 'Search Product' bar with dropdowns for 'select' and 'search'. Below it are buttons for 'Add', 'Update', and 'Delete'. An 'Add' dialog is open, showing fields for 'Reorder Point' (0), 'ID' (x), 'Product' (Trial), 'Quant' (0), and 'Category' (Bread). The 'ID' field is highlighted in yellow.

Here, I have put the Product ID as x. According to the nature of the Product ID, it should ignore the input as it is being auto-incremented.

Result:



The treeview now includes a new entry at the bottom:

Product ID	Category	Product	Reorder Point	Stock Quantity
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	Protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60
25	Bread	Trial	0	0

Here, the treeview has the new product record despite an invalid data type input for the Product ID.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
20.	Entering the wrong product ID data type	N/A	N/A	X	Program should ignore the user input as the	Pass

					product ID is being auto-incremented	
--	--	--	--	--	--------------------------------------	--

Updating the product table

```
def update_product(self):
    con = sqlite3.connect(database=r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        if self.var_product_id.get() == "": # if the product ID selected is not reflected in the entry field
            messagebox.showerror("Incorrect Input", "Please select an existing Product ID", parent=self.wind)
            return

        if self.var_product.get() == "": # if there is no input in the product entry
            messagebox.showerror("Incorrect Input", "Please select an existing Product ", parent=self.wind)
            return

        if self.var_cat_name.get() == "": # if there is no category in the category name entry
            messagebox.showerror("Incorrect Input", "Please select a existing Category ", parent=self.wind)
            return

        # checks whether the product input is in alphabets or not
        if not self.var_product.get().isalpha():
            messagebox.showerror("Incorrect Input", "Please enter the correct data type for product", parent=self.wind)
            return

        cur.execute("Select CAT_NAME from category where CAT_NAME = ?", (self.var_cat_name.get(),))
        fetch = cur.fetchone()
        if fetch == None:
            messagebox.showerror("Error", "Please update to an existing category", parent=self.wind)

    else:
```

```
        cur.execute("Select * from Product where Product_ID=?", (self.var_product_id.get(),))
        row = cur.fetchone() # fetches the next row of data
        if row == None: # will only update if I have the Product ID
            messagebox.showerror("Error", "This Product ID is invalid") # validation
```

The code above is used to update a product record in the table. Here, I have added validation to ensure that the product name will only be changed to a value that only contains alphabets. I have also ensured that the program fetches the current record using the Product ID as this will not change.

```
else:
    cur.execute("Update Product SET CAT_NAME=?, Product=?, Reorder_Point=?, Quantity=? WHERE Product_ID=?", ( # we need a where statement for our query
        self.var_cat_name.get(),
        self.var_product.get(),
        self.var_Reorder_Point.get(),
        self.var_quantity.get(),
        self.var_product_id.get(),
    ))
    con.commit()
    messagebox.showinfo("Updated!!!", "Successfully updated Product!!", parent=self.wind)
    self.display_Product_data() # this helps to display the data. Once the update has been made, the changed data will be displayed automatically

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent = self.wind)
```

I have amended the code:

```

cur.execute("Select CAT_NAME from category where upper(CAT_NAME) = ?",
fetch = cur.fetchone()
if fetch == None:
    messagebox.showerror("Error", "Please update to an existing category", parent=self.wind)
    return
else:
    cur.execute("Select * from Product where Product_ID=?",
    (self.var_product_id.get(),))
    row = cur.fetchone() # fetches the next row of data
    if row == None: # will only update if I have the Product ID
        messagebox.showerror("Error", "This Product ID is invalid") # validation
        return

```

Once again, I have ensured that the update function should not be case sensitive by converting to capitals and should treat any 2 words that have the same spelling as the same.

If all the validation features are accepted, then the program will update the records.

Testing: Updating the quantity and reorder point of Product table

21.	Updating the product table	Updating one of the middle records	Updating the last record	Updating nothing with no record from treeview selected	Product record should successfully update the normal and boundary data
-----	----------------------------	------------------------------------	--------------------------	--------------------------------------------------------	------------------------------------------------------------------------

Product ID	Category	Product	Reorder Point	Stock Quantity
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paner	30	0
11	Protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60
25	Bread	Trial	0	0
26	Bread	Test	0	0

This is the record I have selected.

The screenshot shows a product management application. At the top is a table with columns: Product ID, Category, Product, Reorder Point, and Stock Quantity. The table contains 21 rows of data. Row 25, which has a Product ID of 25, a Category of Bread, a Product of Trial, a Reorder Point of 0, and a Stock Quantity of 0, is highlighted with a blue selection bar. Below the table is a search bar labeled 'Search Product' with a dropdown menu set to 'select'. Underneath the search bar are four buttons: 'Add', 'Update', 'Delete', and 'Cancel'. The 'Update' button is highlighted with a dark purple background. Below these buttons is a form with fields: 'Reorder Point' (set to 0), 'ID' (set to 25), 'Product' (set to Trial), 'Quant' (set to 0), 'Category' (set to Bread). The entire interface has a light blue header bar.

Here, the reorder point and the quantity is 0. I will change these numbers and will change the name of the Product from Trial to Trials.

Product ID	Category	Product	Reorder Point	Stock Quantity
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	Protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60
25	Bread	Trial	0	0
26	Bread	Test	0	0

Search Product

select search

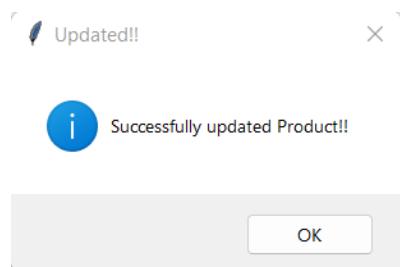
Add Update Delete

Reorder Point ID

Product Trials

Quant Category Bread

Here, I have changed to Reorder point to 25 and the Quantity to 50.



As you can see below, the record has been successfully changed. The name changes to Trials and the reorder point and the quantity change to 25 and 50 respectively.

Product ID	Category	Product	Reorder Point	Stock Quantity
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	Protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60
25	Bread	Trials	25	50
26	Bread	Test	0	0

Boundary data: Updating the last record from the product table

Updated!!

Successfully updated Product!!

OK

Product ID	Category	Product	Reorder Point	Stock Quantity
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	Protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60
25	Bread	Trials	25	50
26	Bread	Test	0	0

Add Update Delete

Reorder Point: 100 ID: 26

Product: Test

Quant: 100 Category: Bread

As you can see on the last record of the treeview, both the Reorder Point and the Stock Quantity have changed to 100 starting initially from 0.

Product ID	Category	Product	Reorder Point	Stock Quantity
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	Protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60
25	Bread	Trials	25	50
26	Bread	Test	100	100

Erroneous data: No record selected

Incorrect Input

Please select an existing Product

OK

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50
2	Junk	Doughnut	100	110
3	Junk	Pizza	30	50
4	Milk	Cocount	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Add Update Delete

Reorder Point: 0 ID: 0

Product:

Quant: 0 Category:

Here, I have not selected any record as the entries for the Product and Category are empty and none of the records in the treeview are highlighted in blue.

When I press the update button:

The message indicates that the program doesn't allow an empty product to be updated.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>21.</u>	Updating the product table	Updating one of the middle records	Updating the last record	Updating nothing with no record from treeview selected	Product record should successfully update the normal and boundary data	Pass

Testing: Updating the product table with a non-existing category name

Product ID	Category	Product	Reorder Point	Stock Quantity
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60
25	Bread	Trials	25	50
26	Bread	Test	100	100

Search Product

Add Update Delete

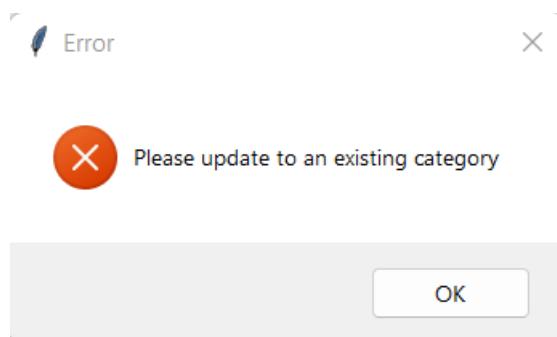
Reorder Point: 100 ID: 26

Product: Test

Quant: 100 Category: Error

For this test, I have put the category as "Error" which doesn't exist in the category table.

When I click on update:



This program ensures that the category name that is being updated to exists in the category table.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>22.</u>	Updating the product table with a non-existing category name	N/A	N/A	Category name: Error	The program should show an error saying that the category doesn't	Pass – only updates category if it exists in the category table

					exist so the record cannot be updated.	
--	--	--	--	--	----------------------------------------	--

Since this update function uses the same validation code as the add function, I have not carried out any more tests for the update function.

Deleting a record from the Product table

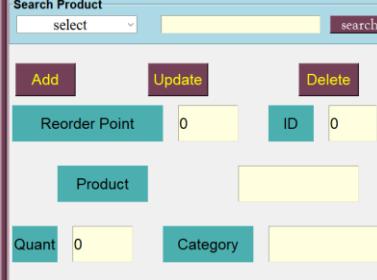
```
def delete_product(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_product_id.get() == "": # if entry box for product ID is blank
            messagebox.showerror("Incorrect Input", "Please choose a Product ID to be deleted", parent=self.wind)
        else:
            cur.execute("Select * from Product where Product_ID = ?", (self.var_product_id.get(),)) # uses product id to fetch the existing record
            row = cur.fetchone()
            if row == None:
                messagebox.showerror("Error", "This Product ID is invalid", parent = self.wind)
            else:
                # confirmation of deletion
                ask = messagebox.askyesno("Confirmation", "Do you want to delete this record?", parent = self.wind)
                if ask == True:
                    cur.execute("Delete from Product where Product_ID=?", (self.var_product_id.get(),)) # the ID is the primary key
                    con.commit()
                    messagebox.showinfo("Delete", "Successfully deleted", parent = self.wind)
                    self.display_Product_data()

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)
```

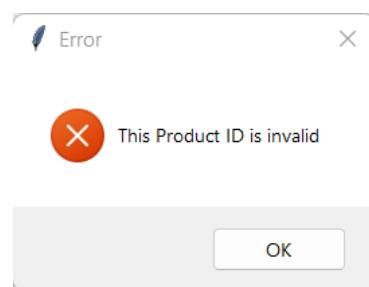
Testing: Deleting an unselected record from the Product table

23.	Deleting an unselected product	N/A	N/A	No user selection of record	Error should occur saying can't delete
-----	--------------------------------	-----	-----	-----------------------------	----------------------------------------

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50
2	Junk	Doughnut	100	110
3	Junk	Pizza	30	50
4	Milk	Coconut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60



Here, nothing has been selected as none of the records are highlighted in blue. When I press delete:



The reason it shows this message error is that the default value for an Int variable is 0, and since there is no Product ID that has the value 0, an error occurs saying the product ID is invalid.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>23.</u>	Deleting an unselected product	N/A	N/A	No user selection of record	Error should occur saying it can't delete	Pass – only deletes selected records

Testing: Deleting a selected record

24.	Deleting a selected record	One of the middle records	The last record	N/A	Will delete any existing record regardless of the position of the record
-----	----------------------------	---------------------------	-----------------	-----	--------------------------------------------------------------------------

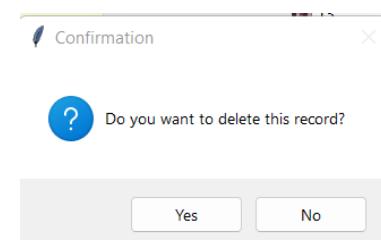
Normal:

The screenshot shows a Windows application window titled "Search Product". Inside, there is a table with columns: Product ID, Category, Product, Reorder Point, and Stock Quantity. The table contains 26 rows of data. Row 25, which has a blue background, is highlighted. Below the table are three buttons: "Add", "Update", and "Delete". A confirmation dialog box titled "Confirmation" is overlaid on the window, asking "Do you want to delete this record?".

Product ID	Category	Product	Reorder Point	Stock Quantity
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60
25	Bread	Trials	25	50
26	Bread	Test	100	100

Here, I have selected the second last row as the record I want to delete.

When I press delete:



I get a message box asking whether I want to delete the record or not.

If I press yes:

The screenshot shows the same application window after the deletion. The confirmation dialog is closed, and a message box at the bottom left says "Successfully deleted". The table now has 25 rows, and the last row (Product ID 26) is highlighted. The "Delete" button is still visible below the table.

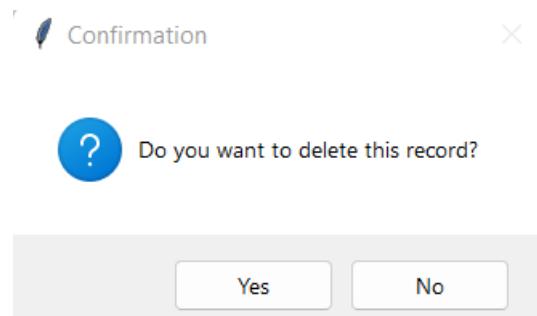
As you can see on the treeview above, there is no record that has the Product ID 25.

Boundary data:

Product ID	Category	Product	Reorder Point	Stock Quantity
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60
26	Bread	Test	100	100

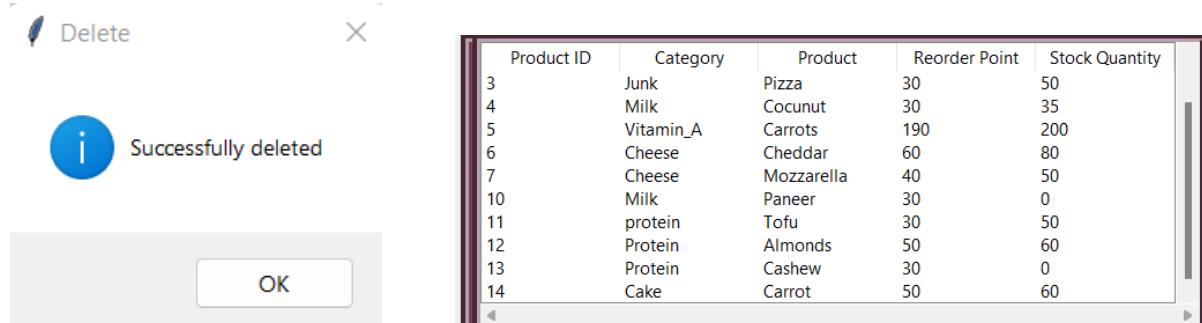
Here, I have selected the last record of the product table.

When I press delete:



Once again. I get a confirmation message confirming whether I want to delete the chosen record.

When I press yes:



Here, as you can see on the treeview, the record containing product ID 26 no longer exists.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>24.</u>	Deleting a selected record	One of the middle records	The last record	N/A	Will delete any existing record regardless of the position of the record	Successfully deletes a selected product

Searching for a Product based on Product ID or Product

```

def search(self):
    con = sqlite3.connect(database=r'sms.db') # connecting it to the database
    cur = con.cursor() # database cursor
    try:
        if self.var_searchuse.get() == "select": # the user should select either product name or ID
            messagebox.showerror("Error", "Select one of the options", parent = self.wind) # error due to incorrect search
        elif self.var_searchtxt.get() == "":
            messagebox.showerror("Error", "Please type what you want to search", parent=self.wind)
        else:
            cur.execute("select * from Product where "+self.var_searchuse.get() + " LIKE '%" +self.var_searchtxt.get()+"%'") # whatever we search can be in any position
            rows = cur.fetchall() # get all the records
            if len(rows)!=0:
                self.Product_Table.delete(*self.Product_Table.get_children())
                for row in rows:
                    self.Product_Table.insert('', END, values=row)

            else:
                messagebox.showerror("Error", "Nothing found", parent = self.wind)

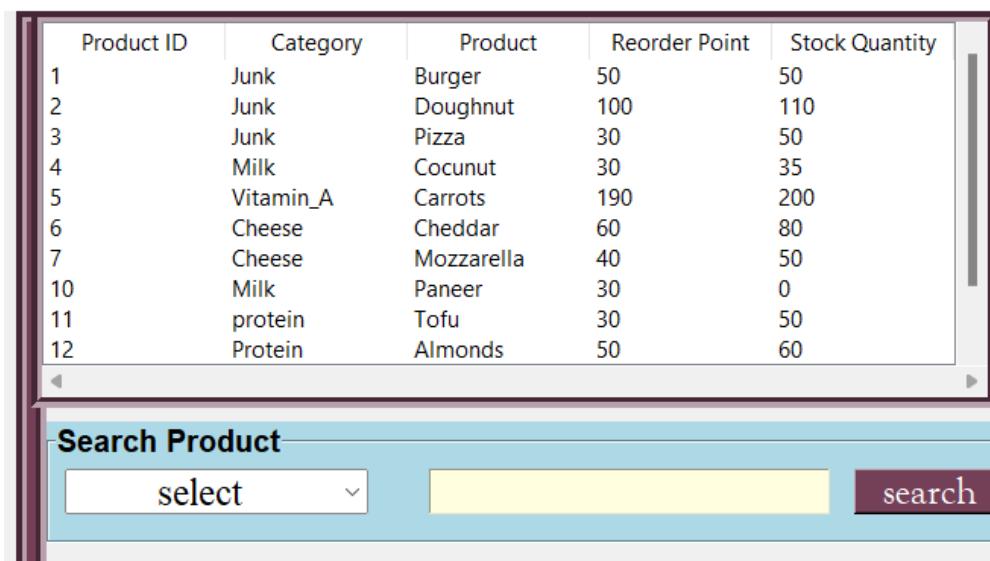
    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # this gives us any errors that may occur when coding

```

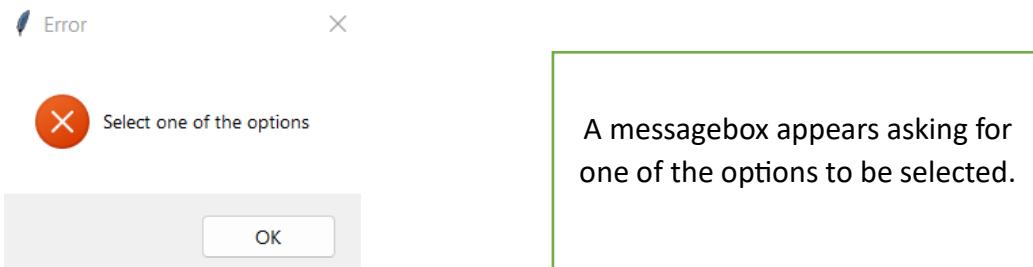
Similar to the Supplier search section, this code above uses a Tkinter combobox to allow a user to search for a product record based on the option that they want to search using (Product or ID) and the input they put into the search bar.

Testing: Searching for a record with no input

25.	Searching for a record with no input	N/A	N/A	N/A – no input from user	Program should tell the user to select one of the options
------------	--------------------------------------	-----	-----	--------------------------	-----------------------------------------------------------



When the user presses the search button:



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
25.	Searching for a record with no input	N/A	N/A	N/A – no input from user	Program should tell the user to select one of the options	Pass – a message asking for the user to select an option

Testing: When user selects an option but no input in the search bar

26.	Searching for a record with an option selected but no user text input on the search bar	N/A	N/A	No user input on the search bar	A message asking for the user to type what they want to search for
------------	-----------------------------------------------------------------------------------------	-----	-----	---------------------------------	--------------------------------------------------------------------

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50
2	Junk	Doughnut	100	110
3	Junk	Pizza	30	50
4	Milk	Cocnut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Search Product

Product search

Add Update Delete

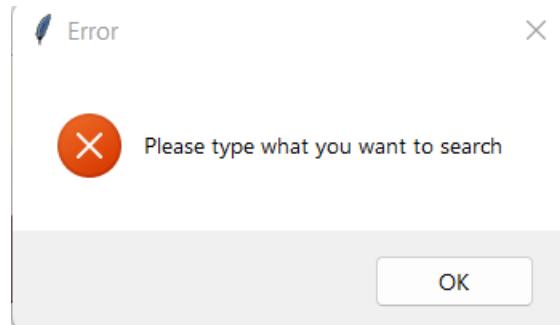
Reorder Point ID

Product

Quant Category

Here, I have selected the option “Product” but have not put any input in the search bar.

When I press search:



This message above asks for the user to type something in the search bar as I expected.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>26.</u>	Searching for a record with an option selected but no user text input on search bar	N/A	N/A	No user input on search bar	Message asking for the user to type what they want to search for	Pass – gets message asking for the user to type something into the search bar

Testing: When user puts input into search bar but doesn't select an option

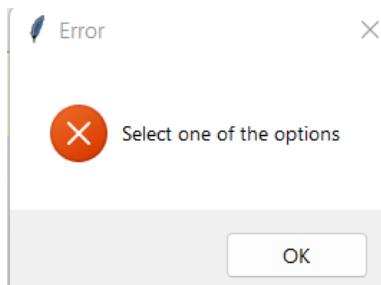
<u>27.</u>	Searching for a record with input in the search bar but no option selected	N/A	N/A	No option selected	Message asking for the user to select an option
------------	----------------------------------------------------------------------------	-----	-----	--------------------	-------------------------------------------------

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50
2	Junk	Doughnut	100	110
3	Junk	Pizza	30	50
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Search Product

select xyz search

Here, I have put the input as "xyz" but haven't selected an option. The program should give us an error message.



Here, the program does show an error message asking for one of the options to be selected. This is because if no options are selected, then the program doesn't know what field it should traverse through to match the input on the search bar.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>27.</u>	Searching for a record with input in the search bar but no option selected	N/A	N/A	No option selected	Message asking for the user to select an option	Pass – gets a message asking for the user to select an option

Testing: When user puts all input (option and text)

28.	Searching for a record with all inputs from the user	Option: Product / ID Text: Carrots / 14	Option: Product / ID Text: C / 1	N/A	Should show all records that match the pattern of the user's input
-----	------------------------------------------------------	--------------------------------------------	-------------------------------------	-----	--------------------------------------------------------------------

Normal**Normal: Using Product Name**

The screenshot shows a software application window. At the top, there is a table with the following data:

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50
2	Junk	Doughnut	100	110
3	Junk	Pizza	30	50
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Below the table is a search interface with the following elements:

- A title "Search Product" in bold black font.
- A dropdown menu labeled "Product" with a small arrow icon.
- A text input field containing the word "Carrots".
- A solid blue rectangular button labeled "search" in white text.

When I press the search button:

The screenshot shows the same software application window after the search button was pressed. The results table now contains only one row, which is:

Product ID	Category	Product	Reorder Point	Stock Quantity
5	Vitamin_A	Carrots	190	200

This output above shows the records that match the user's criteria. Here, as you can see that the user can easily search for a record as the program will temporarily hide all the unmatching records.

This is useful if the product is right at the bottom of the treeview and database, and there are lots of product records to scroll through.

Here, there is only one record that matches the criteria, as I have entered the full name of the product and not partial characters. The program doesn't allow any repeating product names to be entered.

Normal: Using Product ID

The screenshot shows a software application window. At the top is a table with columns: Product ID, Category, Product, Reorder Point, and Stock Quantity. The data includes various items like Junk, Milk, Vitamin_A, Cheese, and Protein categories with their respective products and stock levels. Below this is a search interface titled "Search Product". It has a dropdown menu set to "Product ID", a text input field containing "14", and a dark blue "search" button.

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50
2	Junk	Doughnut	100	110
3	Junk	Pizza	30	50
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

When I press the search button:

The screenshot shows the same software interface after pressing the search button. The results table now contains only one row, which corresponds to the product with Product ID 14. This row shows the details: Category "Cake", Product "Carrot", Reorder Point "50", and Stock Quantity "60".

Product ID	Category	Product	Reorder Point	Stock Quantity
14	Cake	Carrot	50	60

Here, only one record shows with the Product ID as 14.

Note that if there was a product with ID number containing 14, that would show up as well. In this case, I haven't added a lot of products so I wanted to include this in my normal data to show only one record being matched.

Boundary data

Boundary: Using Product name

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50
2	Junk	Doughnut	100	110
3	Junk	Pizza	30	50
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Search Product

Product

When I press the search button:

All the records that don't have the letter c or C (doesn't matter if its capital or not) are filtered out and as you can below all the records that match the criteria are shown.

Here, all the products that start with the letter c or C are shown. In this case, each product has 'c' as their first letter. If there was a product with the letter c in middle of the word, that record would also be shown too.

Product ID	Category	Product	Reorder Point	Stock Quantity
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60

Search Product

Product

Boundary: Using the Product ID

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50
2	Junk	Doughnut	100	110
3	Junk	Pizza	30	50
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Search Product

Product ID

When I press the search button:

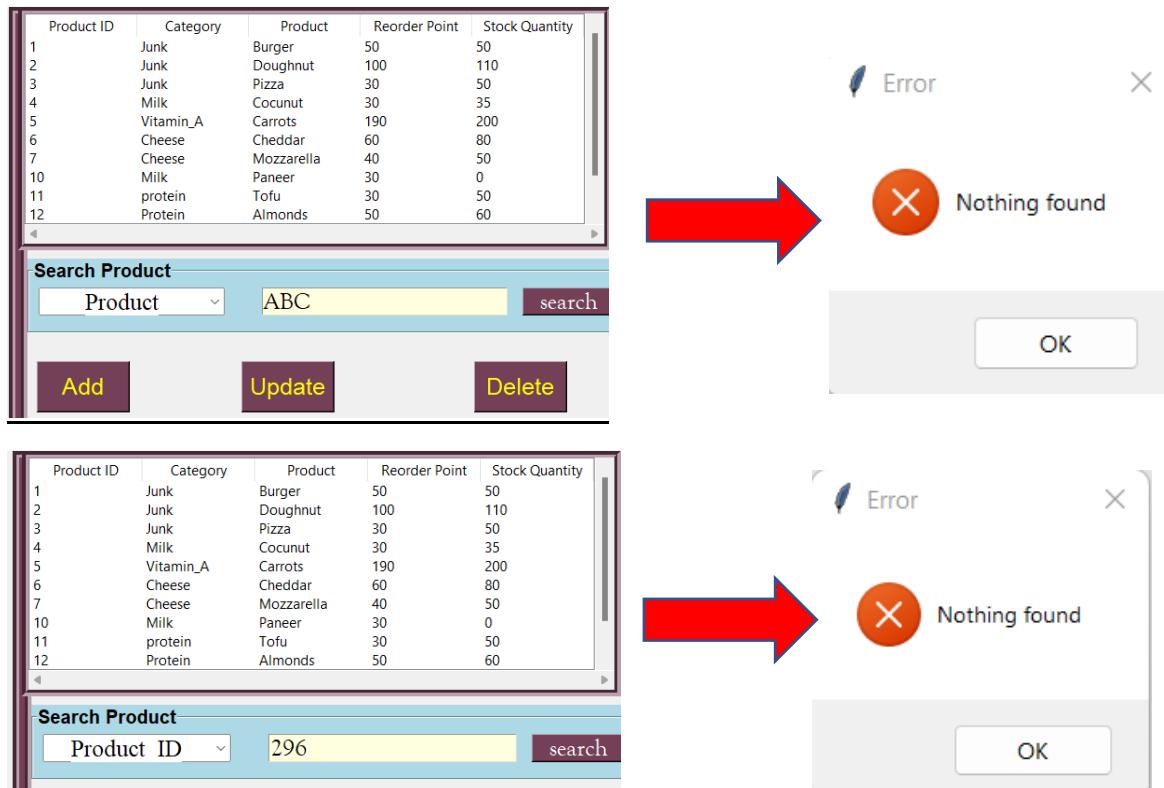
Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	0
14	Cake	Carrot	50	60

Here, all the records that have the Product ID containing 1 are shown.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
28.	Searching for a record with all inputs from the user	Option: Product / ID Text: Carrots / 14	Option: Product/I D Text: C/1	N/A	Should show all records that match the pattern of the user's input	Pass – all the records that match the criteria set by the user are shown

Testing: User input that doesn't match with any of the records

29.	User putting an input that doesn't match any of the records	N/A	N/A	Product: ABC ID: 296	Message showing no records found
------------	-------------------------------------------------------------	-----	-----	-------------------------	----------------------------------



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
29.	User putting an input that doesn't match any of the records	N/A	N/A	Product: ABC ID: 296	Message showing no records found	Pass – message showing nothing found

Sourcing table

This part is made once the supplier and the product tables have been made and fully tested.

This will allow the user to link the Supplier ID and the Product ID together.

```
#####
##### Sales Treeview #####
source_scrolly = Scrollbar(sourcing_frame, orient=VERTICAL)
source_scrollx = Scrollbar(sourcing_frame, orient=HORIZONTAL)
self.sourcing_table = ttk.Treeview(sourcing_frame, columns=("Product_ID", "Supplier_ID"), yscrollcommand=source_scrolly.set, xscrollcommand=source_scrollx.set)
source_scrolly.pack(side=BOTTOM, fill=Y)
source_scrollx.pack(side = RIGHT, fill=X)
source_scrollx.config(command=self.sourcing_table.xview)
source_scrolly.config(command=self.sourcing_table.yview)
self.sourcing_table.heading("Product_ID", text="Product ID")
self.sourcing_table.heading("Supplier_ID", text="Supplier ID")
self.sourcing_table["show"] = "headings"

self.sourcing_table.column("Product_ID", width=120, stretch=NO)
self.sourcing_table.column("Supplier_ID", width=120, stretch=NO)
self.sourcing_table.pack(fill=BOTH, expand=1)
self.sourcing_table.bind("<ButtonRelease-1>", self.get_sourcing)
self.display_source_data()
```

Below is the treeview for the sourcing section.

Add function:

```
def add_source(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_product_id.get() == "" or self.var_supplier_ID.get() == "":
            messagebox.showerror("Error", "Please fill in all the forms", parent = self.wind)
        else:
            cur.execute("Select * from Sourcing where Product_ID=? ", (self.var_product_id.get(),))
            row = cur.fetchone()
            if row != None:
                messagebox.showerror("Error", "This Product ID is invalid", parent = self.wind)

            ##### This checks to see if the Product ID is existing or not #####
            cur.execute("Select Product_ID from Product where Product_ID=? ",(self.var_product_id.get(),))
            fetch_product_ID = cur.fetchone()
            if fetch_product_ID == None:
                messagebox.showerror("Error", "This is not an existing Product ID", parent = self.wind)

            cur.execute("SELECT ID from supplier where ID=? ",(self.var_supplier_ID.get(),))
            fetch_Supplier_ID = cur.fetchone()
            if fetch_Supplier_ID == None:
                messagebox.showerror("Error", "This is not an existing Supplier ID", parent= self.wind)
```

Here, I have added validation to ensure that the user inputs an existing supplier ID and Product ID. If not, then this will affect SQL queries that will be using these variables for fetching supplier and product details.

```
else:
    cur.execute("Insert into Sourcing(Product_ID,Supplier_ID) values(?,?)",
               (self.var_product_id.get(),
                self.var_supplier_ID.get()))
    con.commit()
    messagebox.showinfo("Successfully added", "Successfully linked supplier and product", parent = self.wind)
    self.display_source_data()

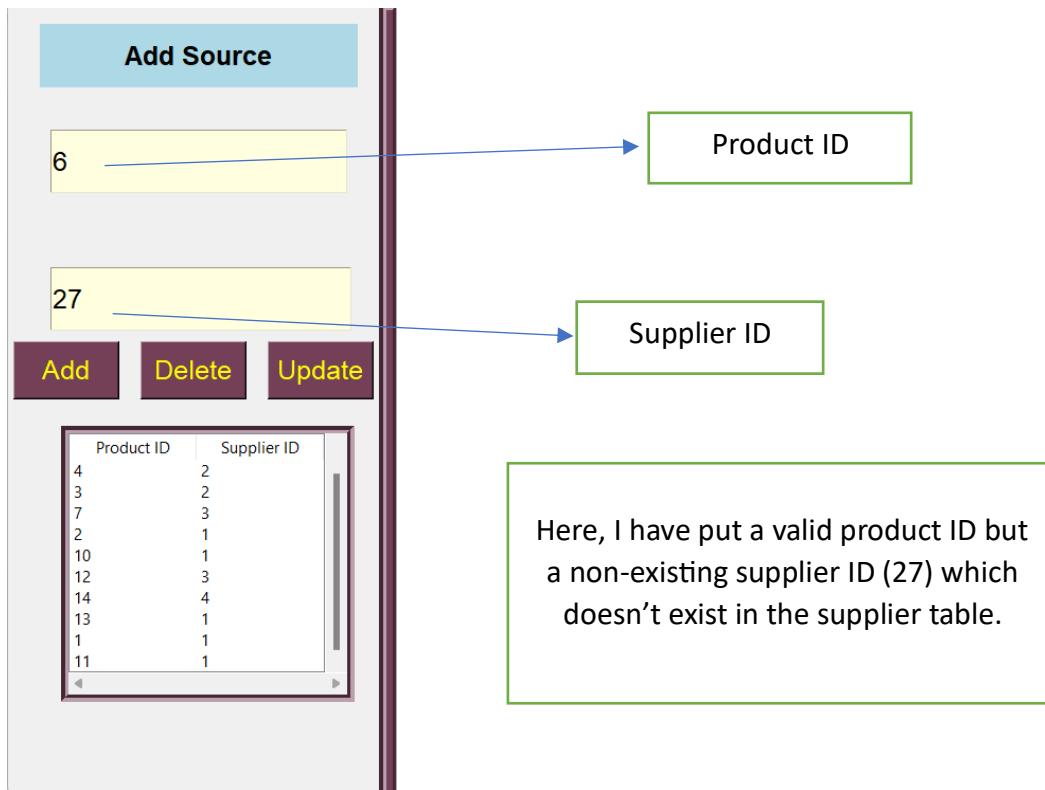
except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent= self.wind)
```

This code above is executed if all the user inputs are valid - and then they are inserted into the sourcing table.

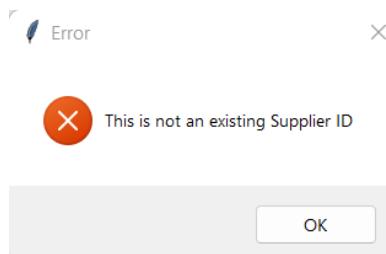
Testing: User adding a non-existing supplier ID in Sourcing table

30.	Adding a non-existing supplier ID to the sourcing table	N/A	N/A	Supplier ID: 27	Error message stating that the supplier Id doesn't exist (invalid)
------------	---------------------------------------------------------	-----	-----	-----------------	--------------------------------------------------------------------

```
cur.execute("SELECT ID from supplier where ID=? ",(self.var_supplier_ID.get(),))
fetch_Supplier_ID = cur.fetchone()
if fetch_Supplier_ID == None:
    messagebox.showerror("Error", "This is not an existing Supplier ID", parent= self.wind)
```



When I press add:

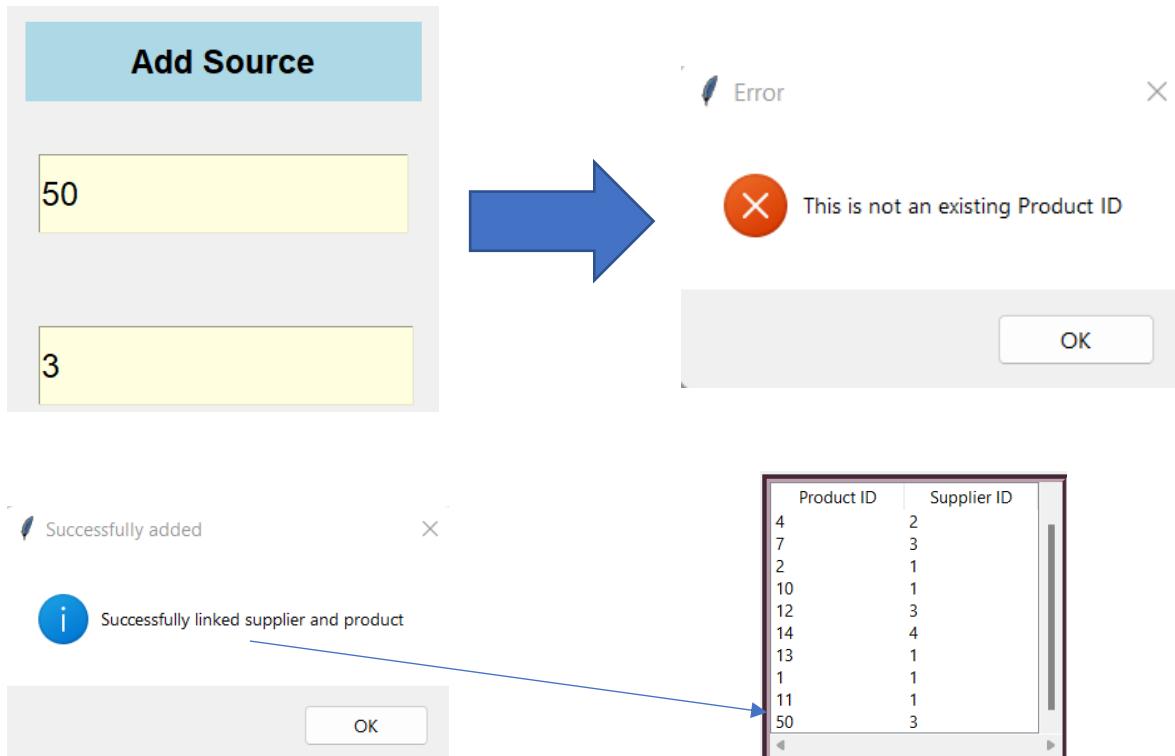


A message occurs as expected displaying that this is not an existing supplier.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>30.</u>	Adding a non-existing supplier ID to the sourcing table	N/A	N/A	Supplier ID: 27	Error message stating that the supplier Id doesn't exist (invalid)	Pass – error appears telling the user that the supplier ID they input is not existing

Testing: User adds a non-existing Product ID in Sourcing table

31.	Adding a non-existing product ID to the sourcing table	N/A	N/A	Product ID: 50 Supplier ID: 3	Error stating that the product ID doesn't exist (invalid)
-----	--------------------------------------------------------	-----	-----	----------------------------------	-----------------------------------------------------------



When I press the add button, at first it shows an error message, then it shows a message saying that it has successfully linked supplier and product when the Product ID doesn't exist. It also gets added to the sourcing table as shown in the treeview above.

I have checked the code below and noticed I haven't used the return statement when giving conditions.

```

if self.var_product_id.get() == "" or self.var_supplier_ID.get() == "":
    messagebox.showerror("Error", "Please fill in all the forms", parent = self.wind)
else:
    cur.execute("Select * from Sourcing where Product_ID=?", (self.var_product_id.get(),))
    row = cur.fetchone()
    if row != None:
        messagebox.showerror("Error", "This Product ID is invalid", parent = self.wind)

#####
# This checks to see if the Product ID is existing or not #####
cur.execute("Select Product_ID from Product where Product_ID=?",(self.var_product_id.get(),))
fetch_product_ID = cur.fetchone()
if fetch_product_ID == None:
    messagebox.showerror("Error", "This is not an existing Product ID", parent = self.wind)

cur.execute("SELECT ID from supplier where ID=?",(self.var_supplier_ID.get(),))
fetch_Supplier_ID = cur.fetchone()
if fetch_Supplier_ID == None:
    messagebox.showerror("Error", "This is not an existing Supplier ID", parent=self.wind)

```

Amended code:

```

if self.var_product_id.get() == "" or self.var_supplier_ID.get() == "":
    messagebox.showerror("Error", "Please fill in all the forms", parent = self.wind)
    return
else:
    # since supplier to product is a one to many relationship
    # no product should have more than 1 supplier
    cur.execute("Select * from Sourcing where Product_ID=?", (self.var_product_id.get(),))
    row = cur.fetchone()
    if row != None:
        messagebox.showerror("Error", "This Product ID is invalid", parent = self.wind)
        return

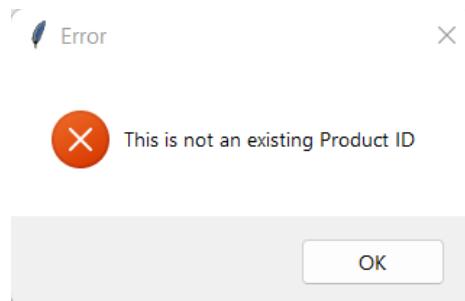
    ##### This checks to see if the Product ID is existing or not #####
    cur.execute("Select Product_ID from Product where Product_ID=?", (self.var_product_id.get(),))
    fetch_product_ID = cur.fetchone()
    if fetch_product_ID == None:
        messagebox.showerror("Error", "This is not an existing Product ID", parent = self.wind)
        return

    cur.execute("SELECT ID from supplier where ID=?", (self.var_supplier_ID.get(),))
    fetch_Supplier_ID = cur.fetchone()
    if fetch_Supplier_ID == None:
        messagebox.showerror("Error", "This is not an existing Supplier ID", parent = self.wind)
        return

```

Here, I have added return statements so that no more code is executed after an error occurs.

This time when I press add:



I only get this error message and nothing else. The treeview doesn't update.

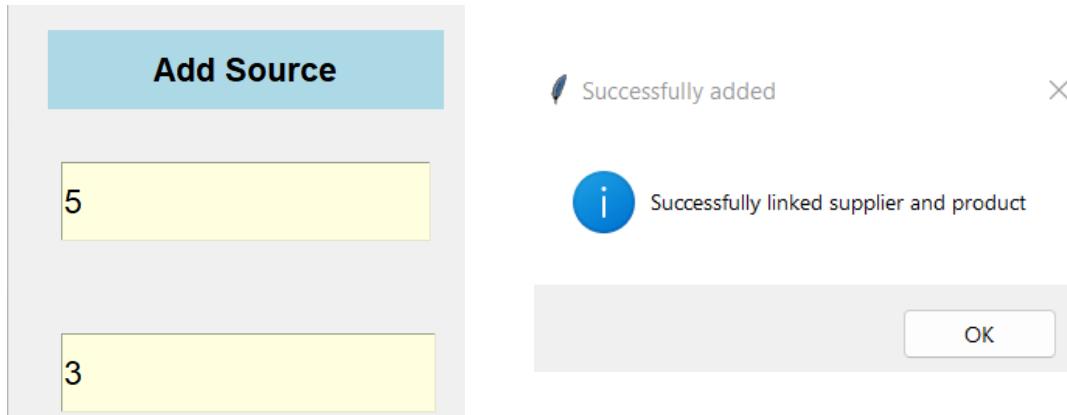
<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>31.</u>	Adding a non-existing product ID to the sourcing table	N/A	N/A	Product ID: 50 Supplier ID: 3 (existing)	Error stating that the product ID doesn't exist (invalid)	Failed initially but have amended the errors

Testing: Adding 2 existing inputs (exists in Supplier and Product table)

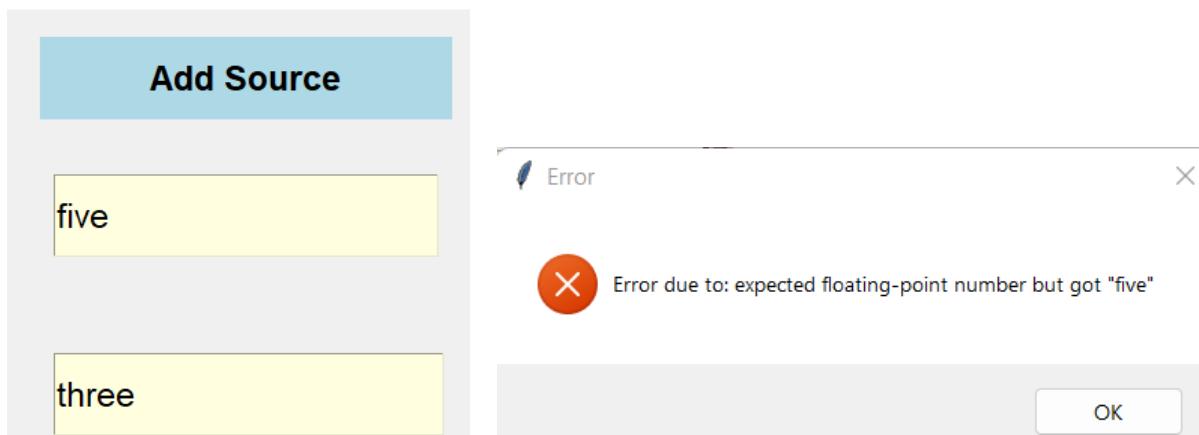
<u>32.</u>	Adding 2 existing inputs (exist in supplier	Supplier ID: 3 Product ID: 5	N/A	Supplier ID: three Product ID: five	Successfully adds normal data but not erroneous
------------	---------------------------------------------	---------------------------------	-----	----------------------------------------	-------------------------------------------------

	and Product table)				
--	--------------------------	--	--	--	--

Normal:



Erroneous:

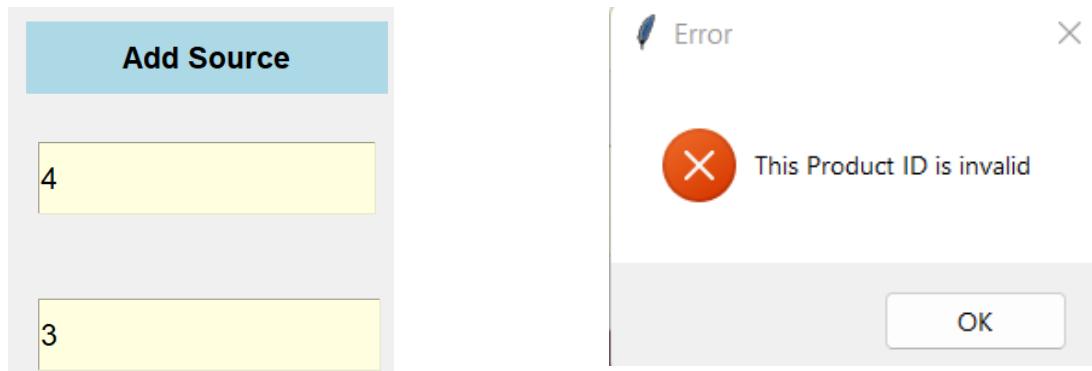


The program works perfectly, as it rejects inputs of a different data type.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>32.</u>	Adding 2 existing inputs (exist in supplier and Product table)	Supplier ID: 3 Product ID: 5	N/A	Supplier ID: three Product ID: five	Successfully adds normal data but not erroneous	Pass

Testing: Adding a new supplier ID to a existing product ID in the sourcing table

33.	Adding a new supplier ID to a product ID that exists in the sourcing table	N/A	N/A	Product ID: 4 Supplier ID: 3	Should show an error as one product can have only 1 supplier
------------	----------------------------------------------------------------------------	-----	-----	---------------------------------	--------------------------------------------------------------



Here we get a message showing that the product ID is invalid. This is because there is already a Product ID 4 record in the sourcing table which cannot be repeated due to the relationship between Product and Supplier (Many to One).

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
33.	Adding a new supplier ID to a product ID that exists in the sourcing table	N/A	N/A	Product ID: 4 Supplier ID: 3	Should show an error as one product can have only 1 supplier	Pass – program doesn't allow a repeating product ID in the sourcing table

Testing: Whether the program allows one supplier to supply multiple products

34.	Adding a repeating supplier ID (that already exists in the sourcing and supplier table) to a new Product ID (doesn't exist in sourcing table)	Supplier ID: 1 Product ID: 14	N/A	N/A	Program should allow existing supplier ID (in both supplier and sourcing table) to supply to multiple products
------------	-----------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------	-----	-----	----------------------------------------------------------------------------------------------------------------

Product ID	Supplier ID
4	2
7	3
2	1
10	1
12	3
13	1
1	1
11	1
3	3
6	3

As you can see here, I have manually added multiple product ID's to the same supplier ID.

I will test this feature using the actual program now and not through the DB browser:

Add Source

Here, I have entered the Product ID as 14 and the supplier ID as 1. When I press the add button:

Successfully added x

i Successfully linked supplier and product

Product ID	Supplier ID
2	1
10	1
12	3
13	1
1	1
11	1
3	3
6	3
5	3
14	1

The message box shows that the supplier and the product have been successfully linked. This means that from Product ID 14 I could fetch the details of the Supplier through SQL queries.

The treeview on the right shows that they have been successfully added as the last record shows the user input.

No.	What I'm testing	Normal	Boundary	Erroneous	Expected Results	Actual outcome
-----	------------------	--------	----------	-----------	------------------	----------------

<u>34.</u>	Adding a repeating supplier ID (that already exists in the sourcing and supplier table) to a new Product ID (doesn't exist in the sourcing table)	Supplier ID: 1 Product ID: 14	N/A	N/A	Program should allow existing supplier ID (in both supplier and sourcing table) to supply to multiple products	Pass – program allows one supplier to supply multiple products
------------	---------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------	-----	-----	----------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------

Updating sourcing table

```
def update_sourcing_record(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_product_id.get() == "" or self.var_supplier_ID.get() == "": # if fields are empty
            messagebox.showerror("Incomplete", "Please select a record from the treeview", parent = self.wind)
            return
        else:
            # this checks if there is an existing product ID in the product table
            cur.execute("Select * from Sourcing where Product_ID = ? ",(self.var_product_id.get(),))
            row = cur.fetchone()
            if row == None:
                messagebox.showerror("Error", "Please choose an existing Product ID", parent = self.wind)
                return
            # this checks if there is an existing Supplier ID in the supplier table
            cur.execute("Select * from Sourcing where Supplier_ID =? ",(self.var_supplier_ID.get(),))
            row = cur.fetchone()
            if row == None:
                messagebox.showerror("Error", "Please choose an existing Supplier ID", parent = self.wind)
                return
    
```

This code above checks if there is an existing supplier ID and product ID in their respective tables. This validation is necessary to ensure that if any supplier records or product records have been deleted, they should reflect to other programs that use SQL queries to gain information from those tables.

```
else:
    cur.execute("Update Sourcing SET Supplier_ID=? WHERE Product_ID=? ", ( # we need a where statement for our query
        self.var_supplier_ID.get(),
        self.var_product_id.get()
    ))
    con.commit()
    messagebox.showinfo("Success", "Successfully updated", parent = self.wind)
    self.display_source_data()

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # shows the error
```

This code updates the sourcing table by updating the Supplier ID for a given product ID. Note that the product ID cannot be updated as it is being used as part of the WHERE clause.

Testing: Updating the sourcing table

35.	Updating sourcing table	Supplier ID: changed from 1 to 3 Product ID: 14	Same as normal	Supplier ID: 1 to three Product ID: 14	Should update the normal data and ignore the erroneous data
------------	-------------------------	----------------------------------------------------	----------------	-------------------------------------------	-------------------------------------------------------------

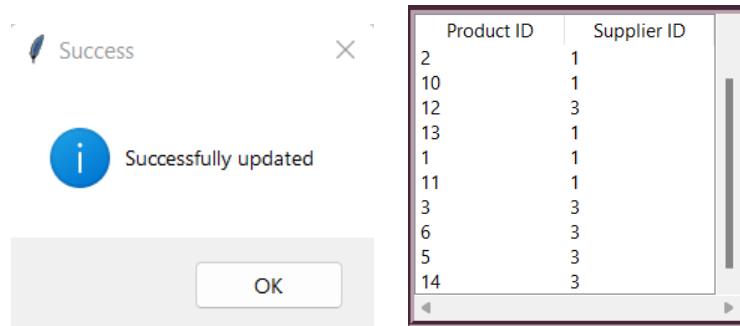
Normal data

Product ID	Supplier ID
2	1
10	1
12	3
13	1
1	1
11	1
3	3
6	3
5	3
14	1

Here, I have selected the last record and have changed the supplier ID from 1 to 3.

According to the logic in my code, the program should allow the Supplier ID to be changed to 3.

When I press the add button:



The message showing that the record has been successfully updated is shown and as you can see the treeview above has been updated as the supplier ID has now been changed to 3.

Erroneous data:

Add Source

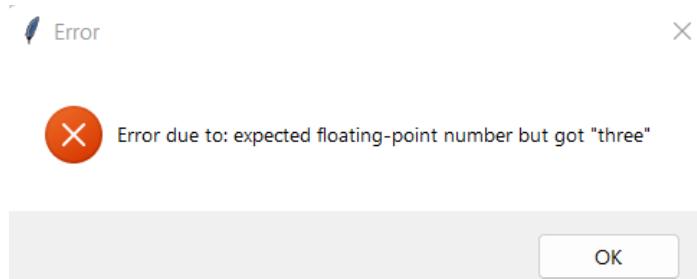
14
three

Add Delete Update

Product ID	Supplier ID
2	1
10	1
12	3
13	1
1	1
11	1
3	3
6	3
5	3
14	1

Here, I have changed the supplier ID from 1 to three. The reason I am using the wrong data type is to check if the program allows a wrong data type to be entered or not.

When I press the update button:



Here, I get an error saying that the data type is wrong and so it is not added to the sourcing table.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>35.</u>	Updating sourcing table	Supplier ID: changed from 1 to 3 Product ID: 14	Same as normal	Supplier ID: 1 to three Product ID: 14	Should update the normal data and ignore the erroneous data	Pass – an error message occurs when the wrong data type is used

Testing: Updating source table to a product ID that doesn't exist in the product table

<u>36.</u>	Updating sourcing table to a product ID that doesn't exist in the product table	N/A	N/A	Supplier ID: 1 – exists Product ID: 100	An error should occur asking the user to enter an existing product ID
------------	---------------------------------------------------------------------------------	-----	-----	--------------------------------------------	-----------------------------------------------------------------------

Add Source

Here, I have kept the input supplier ID as 1 and the Product ID as 100. When I press update:

It tells me to choose an existing product ID.

This test is really important as I will be using SQL queries in other parts of the program to fetch the product and corresponding supplier. If the product ID doesn't exist, then the program will not be able to fetch the product details which could affect the functionality of my program.

Regardless, the program should not change the Product ID as it is part of the where clause of the SQL query.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>36.</u>	Updating sourcing table to a product ID that doesn't exist in the product table	N/A	N/A	Supplier ID: 1 – exists Product ID: 100	An error should occur asking the user to enter an existing product ID	Pass – doesn't allow the user to update to a product ID that doesn't exist in the product table

Testing: Updating Supplier ID that doesn't exist in the Supplier Table

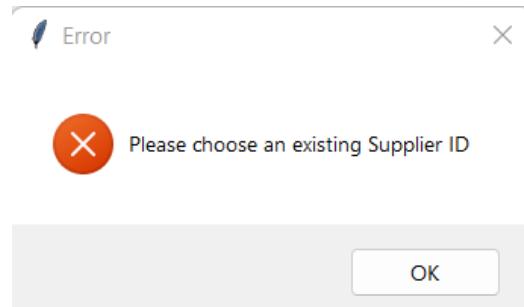
<u>37.</u>	Updating the sourcing table to a supplier ID that doesn't exist in the supplier table	N/A	N/A	Product ID: 5 Supplier ID: 100	An error should occur asking the user to update to an existing Supplier ID (that exists in the Supplier table)
------------	---------------------------------------------------------------------------------------	-----	-----	-----------------------------------	----------------------------------------------------------------------------------------------------------------

Add Source

5		
100		
Add	Delete	Update

Here, I have put the Supplier ID as 100 which doesn't exist in the Supplier Table.

When I press the update button:



The message box shows that the program validates the user's input as it doesn't allow the sourcing table to be updated. Again, this test is really important as I will be using SQL queries for my reorder point and stock-out windows where it will be able to join the Product and Supplier Table together using the Sourcing table.

If the program allowed the user to update to a non-existing supplier ID, then the SQL queries won't be able to fetch the Supplier details for a given product ID and so the user won't be able to send emails as they won't have the supplier's Email Address.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
37.	Updating the sourcing table to a supplier ID that doesn't exist in the supplier table	N/A	N/A	Product ID: 5 Supplier ID: 100	An error should occur asking the user to update to an existing Supplier ID (that exists in the Supplier table)	Pass – doesn't allow the user to update to a supplier ID that doesn't exist in the supplier table

Deleting a record from the Sourcing table

```

def delete__sourcing_record(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_product_id.get() == "": # if product entry box is empty
            messagebox.showerror("Incorrect Input", "Please enter an existing Product ID", parent=self.wind)
            return
        else:
            cur.execute("Select * from Sourcing where Product_ID = ? ",(self.var_product_id.get(),)) # using product id as part of the where clause
            row = cur.fetchone()
            if row == None: # if product ID doesn't exist
                messagebox.showerror("Error", "This Product ID is invalid")
            else:
                ask = messagebox.askyesno("Confirmation", "Do you want to delete this record?", parent = self.wind) # confirmation
                if ask ==True:
                    cur.execute("Delete from Sourcing where Product_ID=? ",(self.var_product_id.get(),)) # the ID is the primary key
                    con.commit()
                    messagebox.showinfo("Delete", "Successfully deleted", parent = self.wind)
                    self.display_source_data() # automatically deletes it without refreshing the tkinter window

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # shows the error

```

This code is used to delete a record from the sourcing table. Here, I have used product ID as part of the where clause. Note that if the user enters an existing product ID but the wrong supplier ID, the program will delete the record as long as the product ID put in the entry box is valid.

Testing: Deleting a record from sourcing table

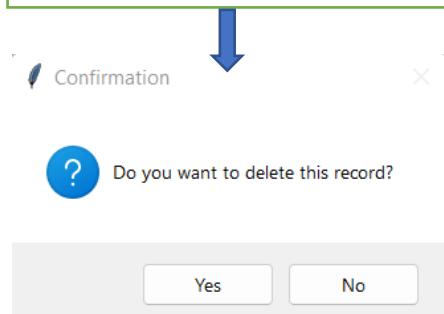
38.	Deleting a record from the sourcing table	Deleting one of the middle records	Deleting the last record	Not selecting a record	Will delete any record as long as the Product ID in the entry box is not blank and exists in the sourcing table
------------	-------------------------------------------	------------------------------------	--------------------------	------------------------	-----------------------------------------------------------------------------------------------------------------

Normal data

The screenshot shows a Tkinter application window. At the top, there are three buttons: 'Add' (yellow), 'Delete' (blue), and 'Update' (green). Below the buttons is a table with two columns: 'Product ID' and 'Supplier ID'. The table contains the following data:

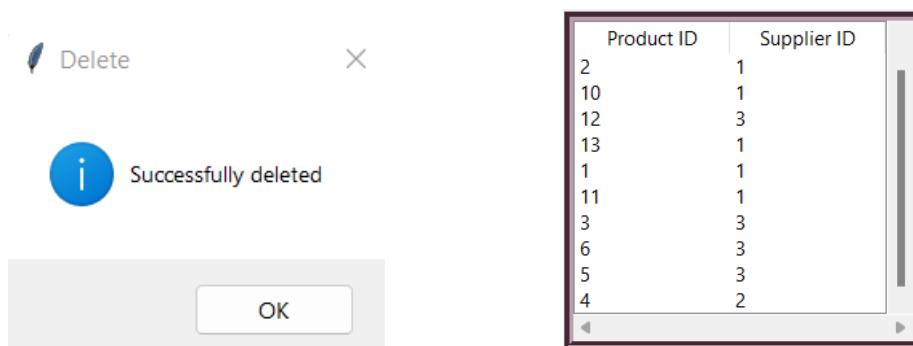
Product ID	Supplier ID
10	1
12	3
13	1
1	1
11	1
3	3
6	3
5	3
14	1
4	2

Here, I have selected the second last record which gets populated into the respective entry fields. When I press the delete button:



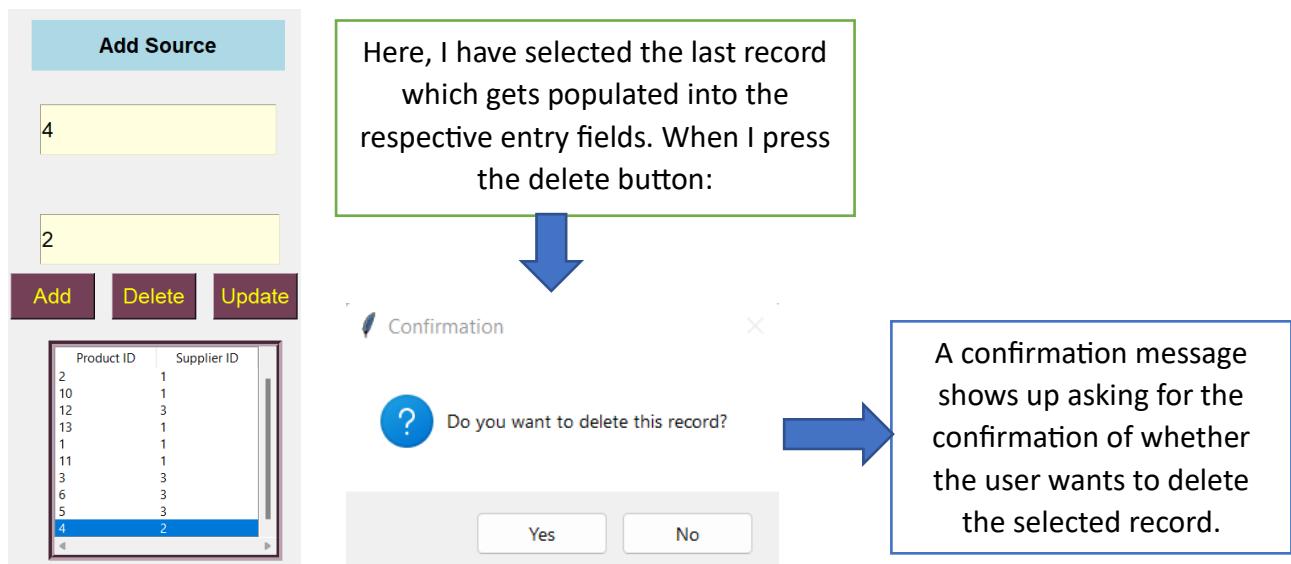
A confirmation message shows up asking for the confirmation of whether the user wants to delete the selected record.

If they press yes:

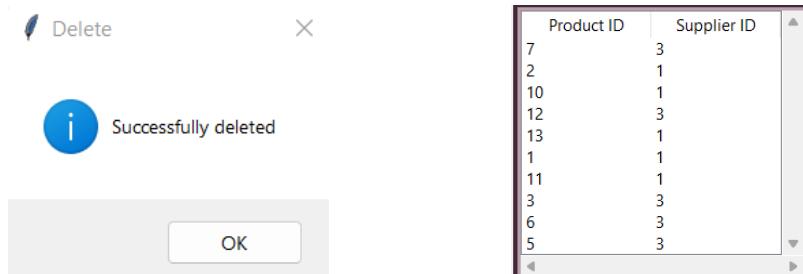


As you can see in the treeview, the record with product ID 14 no longer exists.

Boundary:



If they press yes:



As you can see in the treeview, the last record has been deleted, as there is no record in the sourcing table with the product ID of 4.

Erroneous data:

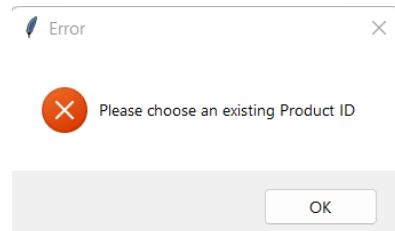
Add Source

Add **Delete** **Update**

Product ID	Supplier ID
7	3
2	1
10	1
12	3
13	1
1	1
11	1
3	3
6	3
5	3

When the user selects no record, the default values are 0 as the supplier ID and the product Id variable are both integer variables.

When the user presses update:



A message box appears asking for the user to choose an existing product ID to delete.

This program successfully tells whether the Product ID is existing or not. If it is not existing, the program doesn't know which record to delete and hence an error will occur.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
38.	Deleting a record from the sourcing table	Deleting one of the middle records	Deleting the last record	Not selecting a record	Will delete any record as long as the Product ID in the entry box is not blank and exists in the sourcing table	Pass – doesn't allow the user to delete a record if there is no existing Product ID in the sourcing table

Testing: Deleting a product with the wrong supplier ID but an existing Product ID in the sourcing table

39.	Deleting a product with the wrong supplier ID but an existing Product ID	N/A	N/A	Product ID: 1 Supplier ID: 70	The user will be able to delete the record with the Product ID 1 not matter what
------------	--------------------------------------------------------------------------	-----	-----	----------------------------------	----------------------------------------------------------------------------------

	in the sourcing table				the supplier ID is put as
--	-----------------------	--	--	--	---------------------------

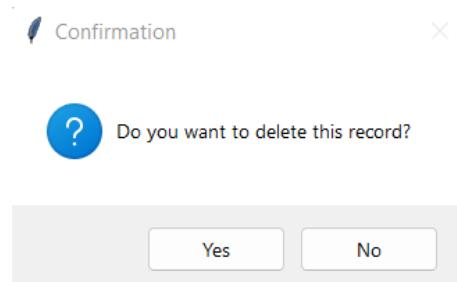
Add Source

1
70
Add Delete Update

Product ID	Supplier ID
7	3
2	1
10	1
12	3
13	1
11	1
3	3
6	3
5	3
1	1

Here, I have selected the record with Product ID 1 and have changed the supplier ID to 70. Hopefully, when the user presses the delete button, the record with product ID 1 should be deleted regardless of the input in the supplier ID entry box as supplier ID isn't part of the WHERE condition.

When the user presses delete:



This message shows asking the confirmation of the record with the product ID that is in the entry box

If the user presses yes:

Delete

Successfully deleted

OK

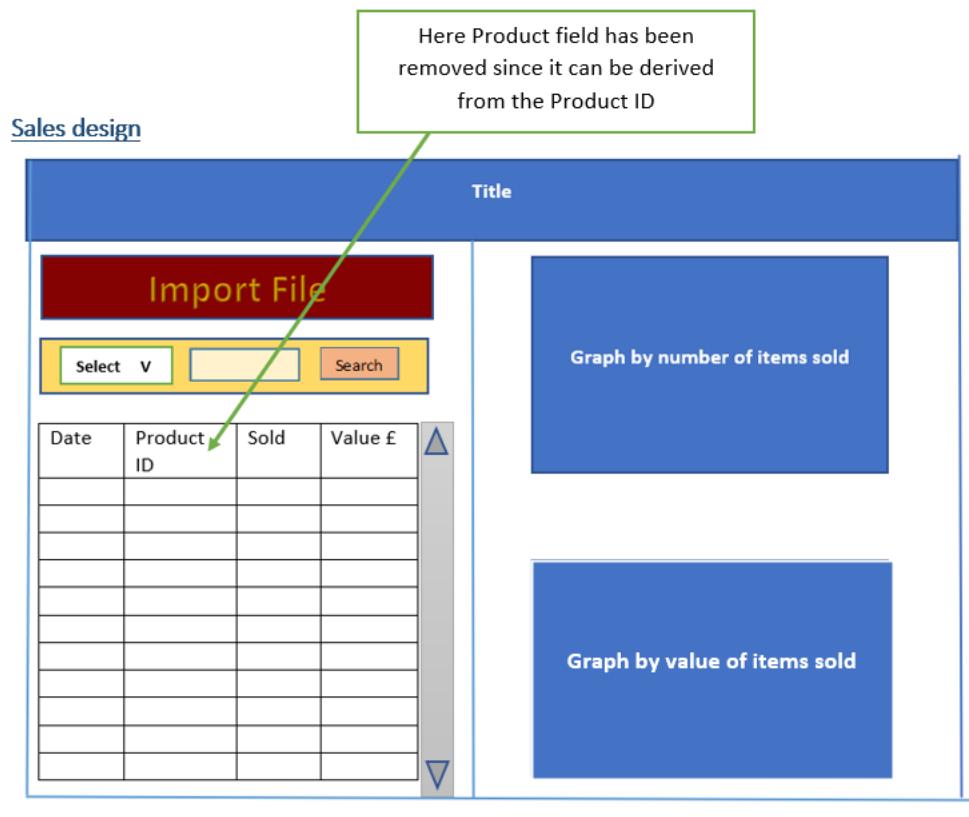
Product ID	Supplier ID
7	3
2	1
10	1
12	3
13	1
11	1
3	3
6	3
5	3

As you can see in the treeview on the left, there is no longer a record with the Product ID of 1.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>39.</u>	Deleting a product with the wrong supplier ID but an existing Product ID in the sourcing table	N/A	N/A	Product ID: 1 Supplier ID: 70	The user will be able to delete the record with the Product ID 1 not matter what the supplier ID is put as	Pass – as long as the product ID exists in the sourcing table, that record will be deleted regardless of the supplier ID put.

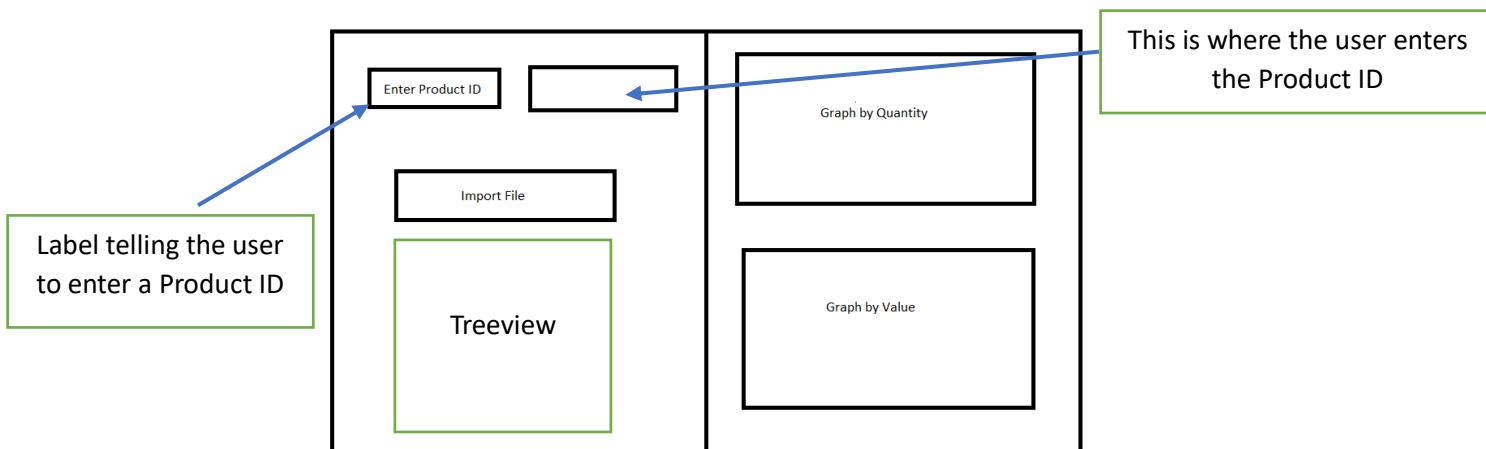
Client Review: I have shown this development to Parth who is happy with the Product window and all the validations used to ensure the user enters valid data. Parth asked why there are no labels for the sourcing and category sections in the Product window used to enter data. I replied by telling him that the order you enter it is the order of the Category and Source table. For example, the top entry box for the sourcing section refers to the first column of the Sourcing table.

Stage 4: Sales Window



Here, I have decided to not include the search bar in the Sales window as it has no purpose. Though the user can select the treeview which will then produce the graph accordingly for the chosen product ID, the user doesn't know what the product name is making it hard to interpret the data.

Instead, what the user could do is go on the Product window and fetch the Product ID using the product search bar when they type the name of the Product. They can then use that Product ID and put it into an entry box that I will create which can then produce the Sales by Quantity and Value Graphs. I have reviewed this with my client Parth who is happy with the design as shown below.



```
cur.execute("CREATE TABLE IF NOT EXISTS Sales(Date text, Product_ID integer, Sold integer, Value integer, foreign key (Product_ID) references "
           "Product(Product_ID), primary key(Date, Product_ID))",
con.commit()
```

This code here creates the Sales table where the Product ID and the Date act as a composite key.

The Product ID is defined as the foreign key in the Sales table and is referenced to the Product table where it acts as a primary key.

```
import os
from tkinter import *      # this is used to create the GUI of my stock management system
import numpy as np
from PIL import Image, ImageTk
from tkinter import ttk, messagebox
import sqlite3
import csv
import matplotlib.pyplot as plt
from product import *
from tkinter import filedialog as fd
from tkinter.messagebox import showinfo
import pandas as pd
from product import ProductClass
import datetime
```

Here, I have imported all the modules that I am going to use such as a “csv” to import csv files into the system, matplotlib to create graphs for a given product ID entered.

```
self.var_product_ID = IntVar() # variable that I am going to use

product_id_label = Label(self.wind, text = "Enter Product ID",
                        font = ("Rosewood Std Regular", 15), bg = "#4cafaf", fg = "black").place(x=30, y = 50, height = 60, width = 250)

txt_product = Entry(self.wind, textvariable= self.var_product_ID,
                     font = ("Rosewood Std Regular", 15), bg = "lightyellow", fg = "black").place(x = 400, y = 50, height = 60)
```

Here, I have created an integer variable called “self.var_product_ID” which will store the value of what the user will input into the entry box.

The code here creates a label saying to “Enter Product ID” and an entry form has been created where the user can enter the Product ID to produce the corresponding graphs.

```
#####
self.Sales_Table = ttk.Treeview(sales_frame,
                                columns=("Date", "Product_ID", "Sold", "Value"),
                                yscrollcommand=scrolly.set, xscrollcommand=scrollx.set)

scrollx.pack(side=BOTTOM, fill=X)
scrolly.pack(side=RIGHT, fill=Y)
scrollx.config(command=self.Sales_Table.xview)
scrolly.config(command=self.Sales_Table.yview)

self.Sales_Table.heading("Date", text="Date")
self.Sales_Table.heading("Product_ID", text="Product ID")
self.Sales_Table.heading("Sold", text="Sold")
self.Sales_Table.heading("Value", text="Value £")
self.Sales_Table["show"] = "headings"

self.Sales_Table.column("Date", width=170, stretch=NO) # fixed width
self.Sales_Table.column("Product_ID", width=170, stretch=NO) # fixed width
self.Sales_Table.column("Sold", width=170, stretch=NO)
self.Sales_Table.column("Value", width=170, stretch=NO)

self.Sales_Table.pack(fill=BOTH, expand=1)
self.Sales_Table.pack(fill=BOTH, expand=1)
```

This code above creates the Sales treeview that will be visible when the user imports the sales csv file.

Creating the Sales CSV file

The CSV file can be created in notepad or in excel. If it's created in notepad then it automatically gets converted into an excel file.

Here, I have created a sample csv file that I will be testing:

	A	B	C	D
1	2023-03-06	13	60	160
2	2023-03-06	1	35	140
3	2023-03-06	2	20	100
4	2023-03-06	3	90	180
5				

Pre-condition: Here, the data format cannot be changed otherwise it can't be used by SQL to manipulate data.

The date needs to be in YYYY-MM-DD.

Note that the order of the records have to be correct as it will be entered into the sales table in the order in which the headings are at.

Here the order is Date, Product ID, Quantity (number of items sold) and Value(£).

Importing a CSV file into the Sales table

```
#####
# importing file function #####
def select_file(self):
    connection = sqlite3.connect(r'sms.db') # connecting to the database
    cursor = connection.cursor() # creating a cursor object
    file = open("sales.csv") # opening the name of the csv file
    contents = csv.reader(file) # csv reader object - reading the contents of the csv file
    insert_records = "INSERT INTO Sales (Date, Product_ID, Sold, Value) VALUES(?, ?, ?, ?)" # this imports the file data into the table
    cursor.executemany(insert_records, contents)
    select_last_7_days = "SELECT * FROM Sales WHERE Date > (SELECT DATE('now', '-7 day'))" # Fills treeview with sales data of least seven days
    rows = cursor.execute(select_last_7_days).fetchall() # fetches all the records that fulfil the select_all query above
    connection.commit()
    for r in rows:# iterates through rows
        self.Sales_Table.insert('', END, values=r) # inserts records
```

This code above opens the CSV file “sales.csv” and reads the data stored in it using the csv.reader object. I have created a variable called insert_records that contains the SQL query to insert records.

I have then used the cursor.executemany() method that will insert multiple records of data into the Sales table. This method takes 2 arguments – the SQL query and the variable storing the csv reader object. This ensures that the program reads the contents of the csv file and inserts it into the Sales table using the insert_records SQL query variable.

Once this is done, to ensure that the treeview shows the data for only the last 7 days, I have created a variable called “select_last_7_days” that uses SQL query to fetch records within the last 7 days.

This is then stored in the variable “rows” which is then looped through using a for loop that then inserts it into the Sales treeview each record at a time.

```
file_btn = Button(self.wind, text="Import file", bg = "#734058", command = self.select_file,
                  font = ("OCR A Std", 15), fg = "yellow").place(x = 190, y = 320, height = 55, width = 370) # buttons that will import the file
```

Import file button

Import file

Sales Treeview

Date	Product ID	Sold	Value £

Testing: Importing CSV file into the Sales table

40.	CSV file should be imported to the Sales table	CSV file	N/A	N/A	In the Sales table, the data from the CSV file should be shown as records
------------	------------------------------------------------	----------	-----	-----	---------------------------------------------------------------------------

Here, when I press the Import file button:

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Users\rusha\AppData\Local\Programs\Python\Python39\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "C:\Users\rusha\PycharmProjects\pythonProject5\Sales_records.py", line 138, in select_file
    cursor.executemany(insert_records, contents)
sqlite3.IntegrityError: UNIQUE constraint failed: Sales.Date, Sales.Product_ID
```

I get an error saying that the combination of my Date and Product ID (as they are composite keys) are wrong.

I have checked my database, and saw that there was repeating combinations for my Date and Product ID.

	Date	Product_ID	Sold	Value
	Filter	Filter	Filter	Filter
100	2023-02-23	1	45	135
101	2023-02-23	11	45	135
102	2023-02-23	7	45	135
103	2023-02-24	7	45	135
104	2023-02-24	11	45	135
105	2023-02-25	1	50	150
106	2023-02-25	2	60	180
107	2023-02-25	4	70	350
108	2023-02-25	3	10	50
109	2023-02-25	5	25	100
110	2023-02-25	11	50	150
111	2023-02-25	13	60	160
112	2023-02-26	13	60	160
113	2023-03-06	13	60	160
114	2023-03-06	1	35	140
115	2023-03-06	2	20	100
116	2023-03-06	3	90	180

Here, the last 4 records are the same as my Sales records which is why I am getting an integrity error as SQLite doesn't allow duplicate records (Same Product ID and Date combinations) to be added to the table.

This can create problems, if my user wants to update the Sales or Value record for a given Product ID after they have pressed the “Import File” button.

The only way they can update the changes, is by going on the DB browser and manually deleting the records and adding them back again. This is bad practice and so I will need to change the code so that it is flexible to import an updated Sales file.

Changing the import Sales file code

```

def import_files(self):
    connection = sqlite3.connect('sms.db') # connecting to the database
    cursor = connection.cursor() # creating a cursor object
    with open("sales.csv") as file:
        contents = csv.reader(file) # csv reader object - reading the contents of the csv file
        insert_records = "INSERT INTO Sales (Date, Product_ID, Sold, Value) VALUES(?, ?, ?, ?)" # this imports the file data into the table
        for row in contents:
            print(row)
            try:
                datetime.datetime.strptime(row[0].strip(), "%Y-%m-%d") # converts the first column in the sales csv file to a datetime object
            except ValueError:
                result = messagebox.showerror("Error", "Please check date is formatted properly", parent = self.window) # this checks that the date is in the correct format
                if result == "ok":
                    messagebox.destroy() # if user presses ok, then the messagebox should disappear
            continue

```

This code here reads the contents of the CSV file called “Sales” and then inserts it into the Sales table. Here, I have converted the first column of the sales csv file into a datetime object so that it is in the correct format.

I have added a date validation to show a message if the date is in the incorrect format so that it shouldn’t be added to the sales table, or else this will affect other parts of the program without the user knowing it.

The .strip() method removes any unnecessary white spaces at the beginning and at the end of the date. This makes the program more flexible as it doesn’t matter the positioning of the date within the excel file or notepad, as long as it is the first column and the date format is correct.

```

# Check if a record already exists for the current Date and Product_ID
select_existing = "SELECT * FROM Sales WHERE Date=? AND Product_ID=?"
existing_record = cursor.execute(select_existing, (row[0], row[1])).fetchone()
if existing_record is not None:
    # Record already exists, so update the Sold and Value columns
    update_record = "UPDATE Sales SET Sold=?, Value=? WHERE Date=? AND Product_ID=?"
    cursor.execute(update_record, (row[2], row[3], row[0], row[1]))
else:
    # Record does not exist, so insert a new record
    cursor.execute(insert_records, row)
select_all = "SELECT * FROM Sales WHERE Date > (SELECT DATE('now', '-7 day'))" # sql statement to select all the fields in the sales table within the last 7 days
rows = cursor.execute(select_all).fetchall()
for r in rows:# using the for loop to iterate through the values stored in the row variable
    self.Sales_Table.insert('', END, values=r)# inserting into sales treeview
connection.commit() # commit changes to the database

```

This code above creates a variable called “select_existing” that contains the SQL statement to select records in the sales using the Date and Product ID as part of the WHERE condition as they are a composite key. Another variable called “existing_record” is created that has the SQL query to search for records that have the same Product ID and Date combination after the contents of the csv file have been added to the sales table.

If there are existing records, then the program will update the fields for that record so that there are no duplicate records making the data much more accurate to process and manipulate.

If there are no existing records, then the program will use the “inserts variable” to insert new sales records into the sales table. The function ends with an SQL query to fetch all the records from the Sales table for the last 7 days which are then shown in the Sales treeview.

Testing: Same test as before

When I press the import file button:

The treeview with the sales records for the last 7 days is shown.

Import file				
Date	Product ID	Sold	Value £	
2023-03-06	1	35	140	
2023-03-06	2	20	200	
2023-03-06	3	90	180	
2023-03-06	13	60	160	

This code is much better than the previous code as it allows flexibility and instead of integrity errors occurring it allows the values to be updated.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>40.</u>	CSV file should be imported to the Sales table	CSV file	N/A	N/A	In the Sales table, the data from the CSV file should be shown as records	CSV file content successfully added to the sales table after changing the code so that it accepts existing records

The problem with the updated code is that it will completely update the Sold and Value fields. This means that if a user sells 50 items of Product ID 1, and then changes the CSV file so that the sold items for Product ID is 1 (after they have pressed the import file button for the first time), then though the stock levels for Product ID 1 will be adjusted correctly, the Sales table will store the Sold field for Product ID as 1 and not add it to the existing value of the number of items sold. This means that the Sold value will not be 51 but instead be 1. This can affect other programs such as the Sales graphs that use SQL queries to produce graphs for a particular product for the last 7 days and also the Analysis graphs that show the most popular and least popular items based on the quantity of items sold.

To make sure that the Sold and Value(£) data are aggregated for the existing records, I have added this code:

```
update_record = "UPDATE Sales SET Sold=Sold+?, Value=Value+? WHERE Date=? AND Product_ID=?"
```

This code here ensures that the updated value is the sum of the new Sold and Value data and the existing Sold and Value data for an existing record.

Date	Product ID	Sold	Value £
2023-03-06	1	630	2520
2023-03-06	2	360	3600
2023-03-06	3	1620	3240
2023-03-06	13	282	2880
2023-03-07	1	70	280
2023-03-07	2	40	400
2023-03-07	3	180	360
2023-03-07	13	300	1000

Here, I have entered the sales for today's date. (7th March)

The sold value for Product ID 13 is 300. If I change the CSV file so that the sold value for Product ID is 100, the treeview should show the total of the existing and the new sold quantity which is 400.

	A	B	C	D
1	2023-03-07	13	100	500
2	2023-03-07	1	35	140
3	2023-03-07	2	20	200
4	2023-03-07	3	90	180
5				

When I click the import file button again:

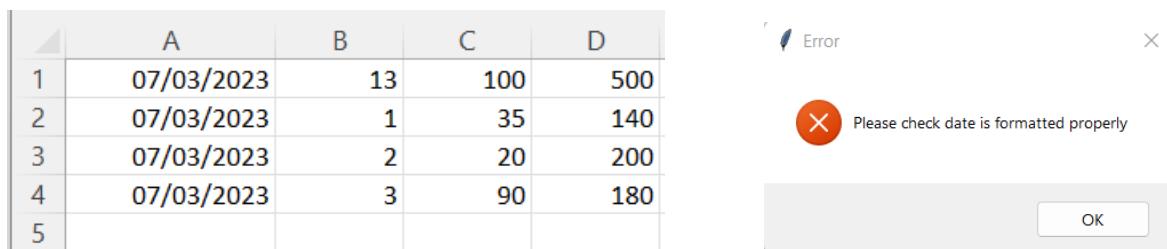
Date	Product ID	Sold	Value £
2023-03-07	3	180	360
2023-03-07	13	300	1000
2023-03-06	1	630	2520
2023-03-06	2	360	3600
2023-03-06	3	1620	3240
2023-03-06	13	282	2880
2023-03-07	1	105	420
2023-03-07	2	60	600
2023-03-07	3	270	540
2023-03-07	13	400	1500

As you can see on the last record, the Product ID for the sold value is 400.

The reason I get multiple records of the same Date and Product ID combination in this treeview here is because I haven't closed the sales window and then reopened the window and pressed the import file button. Here instead, I have pressed on the import file button multiple times. This though will not affect other parts of the program.

Testing: Make sure the date for the CSV sales file is in the correct format

<u>41.</u>	If the date is in the wrong format	N/A	N/A	07/03/2023	Error message should occur asking for the date to be in the correct format
------------	------------------------------------	-----	-----	------------	----------------------------------------------------------------------------



The screenshot shows a CSV file with five columns: Date, Sold, Price, and Stock Level. The data is as follows:

	A	B	C	D
1	07/03/2023	13	100	500
2	07/03/2023	1	35	140
3	07/03/2023	2	20	200
4	07/03/2023	3	90	180
5				

A message box titled 'Error' is displayed, containing the text 'Please check date is formatted properly' with an 'OK' button.

Here, the date is not in the right format, which is why I am getting a messagebox asking for the user to check the date is formatted properly.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>41.</u>	If the date is in the wrong format	N/A	N/A	07/03/2023	Error message should occur asking for the date to be in the correct format	Pass – an error occurs asking for the date to be in the correct format

Updating Stock levels

To update the stock levels based on the sales data in the CSV file, I have written the function in the Product Class which will then be called from the Sales class function.

```
def update_stock():
    connection = sqlite3.connect(r'sms.db')
    cursor = connection.cursor() # creates a cursor object
    sales_file = open("sales.csv") # open the csv file
    #
    select_today = "SELECT Sold FROM Sales where Date == (SELECT DATE('now')) ORDER BY rowid" #
    rows = cursor.execute(select_today).fetchall() # fetches all of today's sales records

    select_product_quantity = "SELECT Quantity FROM Product" # selects all the quantity of products from the product table
    current_stock = cursor.execute(select_product_quantity).fetchall()

    current_stock_lst = [] # list of current levels of the product
    sales_today_lst = [] # list of sales for products
```

Here, I have created a function that creates 2 lists called current_stock_list (the current levels of each product) and the Sales_today_lst (quantity of a product sold).

These lists will store a wide range of data based on the 2 SQL queries shown. One variable called “select_today” is used to store the SQL query to select the Sold values for the given Product IDs and order them by the rowID. Another variable called ‘select_product_quantity’ selects the current quantity levels for all the products.

```

for r in rows: # iterating the values stored in rows
    sales_today_lst.append(r) # appends the value of r to the sales_today_lst list
print(sales_today_lst) # just to show all the data the list contains

for x in current_stock: # iterating the values stored in current_stock
    current_stock_lst.append(x) # appends the value of x to the current_stock_lst list
print(current_stock_lst) # just to show all the data the list contains

```

This code here iterates through all the records stored in the 2 variables ‘rows’ and ‘current_stock’ and adds them individually to their respective lists.

```

#####
##### makes sure that only the correct amount of stocks are updated #####
#####

while len(sales_today_lst) < len(current_stock_lst):
    current_stock_lst.pop() # removes the last element

updated_stock = tuple(numpy.subtract((current_stock_lst),(sales_today_lst))) # subtracts the two lists to find the remaining levels of stocks

updated_stock = list(updated_stock) # converts to list

final_updated_stock = [] ##### list of all the current inventory levels #####

```

This code is used to ensure that the length of the 2 lists are the same so that the numpy.subtract() method can be used to calculate the difference between them.

The while loop compares the length of the 2 lists. If the length of the sales_today_lst is less than the length of the current_stock_lst, then the last element of the current_stock_lst is continuously removed until it is the same size as the sales_today_lst using the pop() method.

The numpy.subtract() method returns a tuple which is converted to a list() using the list() function and assigned to the updated_stock variable.

I have defined a list called final_updated_stock.

```

for r in updated_stock: # iterates through the values in updated_Stock
    connection = sqlite3.connect(r'sms.db') # connecting to the sms database
    cursor = connection.cursor() # creating a cursor object
    r = int(r)
    if r < 0: # if the stock remaining quantity is less than 0, it should be converted to 0
        r = 0 # converted to 0
    final_updated_stock.append(r) # append the value of r to the final_updated_stock list

```

This section of code above iterates through the values stored in ‘updated_stock’ and then checks if the remaining stock level is less than 0. If it is less than 0, then it is converted to the

value 0 as you cannot have a negative quantity of a Product. The values stored in the variable ‘r’ are then appended one by one into the final_updated_stock list.

```
conn = sqlite3.connect(database=r'sms.db') # connection to database
cursor = conn.cursor() # creates a cursor object to perform SQL queries

for i, value in enumerate(final_updated_stock): # enumerate function() is used to get the index of each value in the list
    cursor.execute("UPDATE Product SET Quantity = ? WHERE Product_ID= ?", (value, i+1))
# i+1 variable to ensure that each product's quantity is updated to the correct Product ID which starts with 1 and not 0 which is why it is i + 1
conn.commit()
conn.close()
```

I have used a cursor.execute() method to update the quantity with the new inventory levels for each product.

The enumerate function() returns a tuple containing the value and index of each item in the final_updated_stock list. The for loop iterates over these tuples and assigns the index to the variable ‘i’ and the value to the ‘value’ variable.

If the final_updated_stock is ‘[20,50,60]’, the loop would iterate over the tuples: ‘(0,20)’, ‘(1,50)’, ‘(2, 60)’.

The problem with this code though is that it needs to ensure that Sales data in the CSV file are ordered based on their Product ID. This means that the first sales data should have Product ID 1 and there should not be any skipped Product ID’s in both the Product table and the Sales CSV file. This is not suitable if my client has made sales in only 2 products (Product ID’s 1 and 134) as it would mean that my client has to include all the Products with ID’s ranging from 2-133 inclusively in the csvCSVle even though no sales for those product ID’s have been made.

If the user deletes a Product record in the Product Table, then the next time the user adds a new record, SQLite will not autoincrement the Product ID by one. For example, if the user deletes the record with ID 7, and then the user adds a new record, - that new record will not have the Product ID 7 but instead it will be assigned to 8. This can cause stock levels to be changed incorrectly, as the value of variable ‘i’ increments by one each iteration whereas the Product ID can be skipped to the next unused ID if a record has been deleted. This problem occurs as both the Product ID and the variable ‘i+1’ are part of the WHERE condition.

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	65
2	Junk	Doughnut	100	0
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Here, as you can see in the first column, there is no Product ID 8 and 9 as these records have been deleted and so the same ID cannot be used. This is the nature of SQLite, and so I must change the logic of my code to update the stock levels correctly no matter how the sales data is ordered in my CSV file.

Initially the value of ‘i’ is 0. During the first iteration, the SQL query will update the quantity for product ID ‘i+1’ which is Product ID 1. This is an issue if the first sales data in the CSV file doesn’t have the Product ID = 1. This issue adds up through each iteration done.

If the Product ID wasn't autoincremented, the this would not be an issue as the user could enter the Product ID manually themselves and so the regular sequence of Product ID's could have been followed. However, it is necessary to change the logic of the code to prevent such inflexibility for my client when they are entering the sales figures (data) into the CSV file.

Testing: Updating the product levels correctly

42.	Updating the Product levels once CSV file has been imported	N/A	N/A	N/A	Program should successfully reduce the stock levels for multiple products correctly
------------	-------------------------------------------------------------	-----	-----	-----	-------------------------------------------------------------------------------------

	A	B	C	D
1	2023-03-08	13	100	500
2	2023-03-08	1	35	140
3	2023-03-08	2	20	200
4	2023-03-08	3	90	180

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	65
2	Junk	Doughnut	100	0
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Product ID	Category	Product	Reorder Point	Stock Quantity
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	35
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	300
14	Cake	Carrot	50	60

This is the current levels of all the Products in the table.

When I press the import file button:

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	0
2	Junk	Doughnut	100	0
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	0
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Product ID	Category	Product	Reorder Point	Stock Quantity
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	0
5	Vitamin_A	Carrots	190	200
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	50
10	Milk	Paneer	30	0
11	protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	300
14	Cake	Carrot	50	60

Here as you can see the code doesn't update the stock levels correctly.

For example, for Product ID 1, the initial quantity of stocks was 65 and the quantity of those stocks sold was 35. (Reference to the screenshot of the excel file). So according to the logic of my software, the product table or treeview should show the updated Stock Quantity for Product ID 1 as 30.

However, as you can see above, the treeview shows the stock quantity for Product ID 1 to be 0. This is because the program has subtracted the sold value for the first record in the CSV file (100) from 65 giving us a value of 0 quantity left in the store.

Amending the code

```
def update_stock_levels():
    connection = sqlite3.connect(r'sms.db') # connecting to the database
    cursor = connection.cursor() # creating a cursor object

    with open('sales.csv', 'r') as f: # opening the csv file and reading its contents
        reader = csv.reader(f)
        for row in reader: # iterating through each row in the csv file
            product_id = row[1] # getting the Product ID from the csv file and storing it in a variable
            sold_quantity = row[2] # getiting the sold quantity from the CSV file and storing it in a variable
            if not sold_quantity: # if there is no sold quantity, then it should skip to the next row
                continue
            sold_quantity = int(sold_quantity) # converting the sold quantity to an integer type

            # gets the current stock quantity from the PROduct table based on the Product ID in the CSV file
            cursor.execute("SELECT Quantity FROM Product WHERE Product_ID = ?", (product_id,))
            current_stock = cursor.fetchone()[0]
            updated_stock = max(current_stock - sold_quantity, 0) # subtraction to get the updated stock levels for a product
            # updated stock stores the updated quantity value remaining for a product
            # product ID stores the Product ID from the CSV file
            # Below is the query to update the product stock quantity using the Product ID as part of the where condition
            cursor.execute("UPDATE Product SET Quantity = ? WHERE Product_ID = ?", (updated_stock, product_id))

    connection.commit() # commit the changes
    connection.close()
```

Here, I have created a much more efficient and shorter function that updates the Product stock quantity using the Product ID as part of the SQL where condition.

When subtracting values to get the updated stock, I have used a max function() to ensure that the updated stock quantity is never less than 0. If (as part of testing) the quantity sold is greater than the current stock levels for a particular product, then the updated quantity will be equal to 0.

Testing this function

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	300
2	Junk	Doughnut	100	150
3	Junk	Pizza	30	175

This is a screenshot of the first 3 products in the product treeview.

I will change the sales data so that it has the sales for these 3 products but in a different order so that it doesn't start and end with Product ID 1 and 3 respectively.

```
ProductClass.update_stock_levels()
```

I have written this code at the end of the import_files() function in the SalesClass which will call the "update_stock_levels" which is in the Product Class.

	A	B	C	D
1	2023-03-08	3	100	500
2	2023-03-08	1	35	140
3	2023-03-08	2	20	200

When I press the import button:

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	265
2	Junk	Doughnut	100	130
3	Junk	Pizza	30	75

As you can see, the Stock Quantity column in the treeview has been changed.

Product ID 1: Initial Quantity = 300, Quantity sold: 35, Updated Quantity as shown above = 265

This is correctly changed as $300 - 35 = 265$.

Product ID 2: Initial Quantity = 150, Quantity sold: 20, Updated Quantity as shown above = 130

This stock level has been changed correctly as $150 - 20 = 130$.

Product ID: 3: Initial Quantity = 175, Quantity sold: 100, Updated Quantity as shown above = 75

This stock level has been changed correctly as $175 - 100 = 75$

Comment: This function works no matter what order the sales records in the CSV file are put in. It also works if Product records have been deleted.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
42.	Updating the Product levels once the CSV file has been imported	N/A	N/A	N/A	Program should successfully reduce the stock levels for multiple products correctly	Failed initially as the logic of the function required the order of the sales data in the CSV file to be specific. Have changed the function so that the program uses the Product ID as part of the SQL where condition and doesn't need to rely on the order of the csv file sales data.

Creating graphs

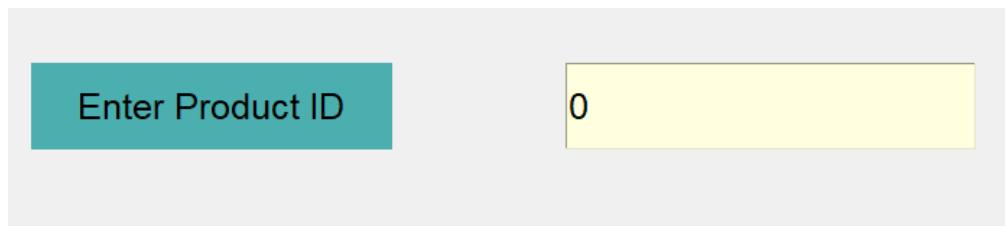
```
self.var_product_ID = IntVar() # variable that I am going to use

product_id_label = Label(self.wind, text = "Enter Product ID", # label for telling the user where to enter the product ID
                        font = ("Rosewood Std Regular" ,15), bg = "#4cafaf" , fg = "black").place(x=30, y = 50, height = 60, width = 250) # positioning of labels

txt_product = Entry(self.wind, textvariable= self.var_product_ID , # entry box to enter the Product ID
                     font = ("Rosewood Std Regular" ,15), bg = "lightyellow" , fg = "black").place(x = 400, y = 50, height = 60)
```

This code here creates a Product ID variable that takes an integer data type. This will store the value of the Product ID entered by the user in the entry box.

Here, I have also created an entry box called 'txt_product' that creates an entry box for the user to type in their Product ID that they would like to see the sales graphs for.



This is how it looks like. Once again, the default value in the entry box is '0' as the Product ID variable in the Sales Class is an integer variable.

Producing the Sales graph by Quantity sold for last 7 days

```
#####
def sales_graph(self):
    connection = sqlite3.connect(r'sms.db') # connecting to the database
    cursor = connection.cursor() # creating a cursor object
    select_items_soldx = cursor.execute("Select Date, Sold FROM Sales WHERE Date > (SELECT DATE('now', '-7 day')) and "
                                         "Product_ID =? ", (self.var_product_ID.get(),)).fetchall() # SQL query to fetch all the Date, sold records
    # within 7 days for a given Product ID entered by the user
    dates = [] # stores all the date values
    sold = [] # stores all the sold values
    for row in select_items_soldx:
        dates.append(row[0]) # adds date value to the dates list
        sold.append(row[1]) # adds sold value to the sold list

    plt.bar(dates, sold, color='c', width=0.72, label="Quantity sold")
    plt.xlabel('Date') # x axis label
    plt.ylabel('Sold') # y axis label
    plt.title('Quantity of food sold') # title
    plt.legend()
    plt.show()
```

This code here creates a variable called “select_items_soldx” that stores the SQL statement to fetch the Date and Value fields from the Sales table for a given product where the date is within the last 7 days.

I have used the variable “self.var_product_ID” as part of the where condition that will retrieve within the last 7 days for that Product ID entered by the user.

Here, I have created 2 lists called dates and sold. These will store the dates and quantity sold when the program iterates through the values stored in the variable ‘select_items_soldx’.

Since in my SQL statement, I have selected the Date first and then Sold, each time the variable is being iterated using ‘row’, the first value will be a dates value and the second value will be the sold value.

These values stored in ‘row’ are added to their corresponding lists from which a graph is made using Matplotlib.

Producing the Sales graph by Value (£) for last 7 days

```
#####
# Graph by value of a Product for last seven days #####
#####

def value_graph(self):
    connection = sqlite3.connect(r'sms.db') # connecting to the database
    cursor = connection.cursor() # creating a cursor object
    select_value = cursor.execute("Select Date, Value FROM Sales WHERE Date > (SELECT DATE('now','-7 day')) and "
                                "Product_ID =? ",(self.var_product_ID.get(),)).fetchall() # SQL query to fetch all the Date, Value records
    dates = [] # within 7 days for a given Product ID entered by the user
    value = [] # stores all the Value made in (£)
    for row in select_value:
        dates.append(row[0]) # adds date to the list
        value.append(row[1]) # adds value to the list

    plt.bar(dates, value, color='g', width=0.72, label="Value £")
    plt.xlabel('Date') # x axis label
    plt.ylabel('Value') # y axis label
    plt.title('The value of food sold (£)') # title
    plt.legend(loc="lower right")
    plt.show()
```

This code here has a very similar pattern to the previous code with the only difference that here instead of sold we are using the value field in the sales table.

The code seems fine but according to the design, the 2 graphs should only be produced once the user has entered a product ID. The function to create these graphs should have a button that the user can press it to view the 2 graphs. Additionally, this code doesn't allow the graphs to be embedded into the tkinter window so I will need to change this code.

I will discuss this with my client:

Me: I am thinking of adding 2 buttons that will produce the 2 sales graphs when the user clicks on it.

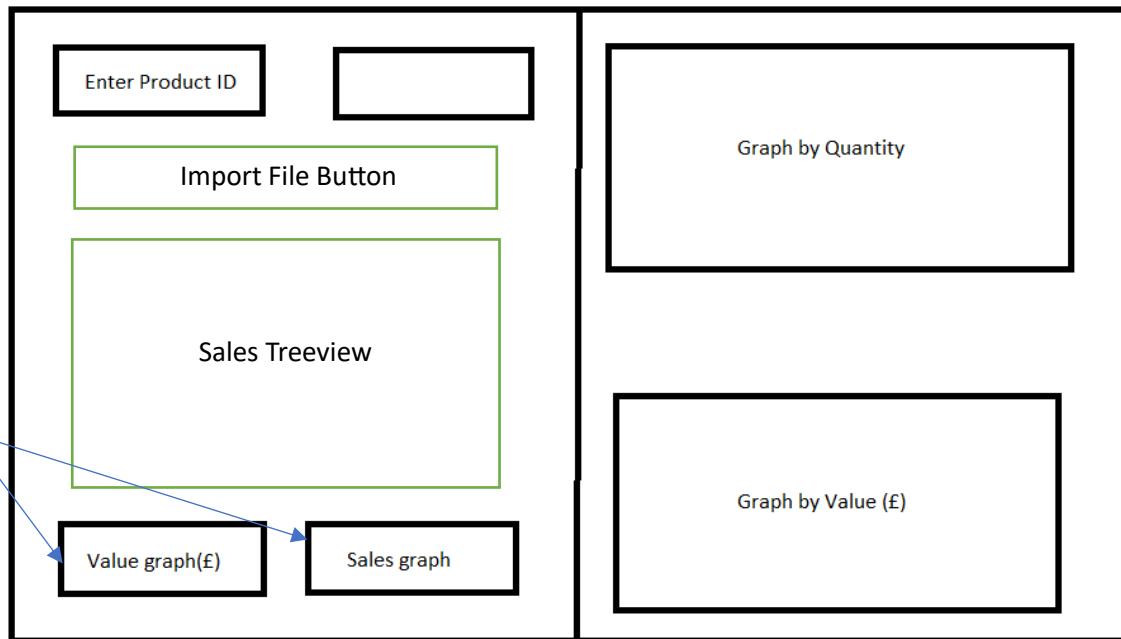
Parth: So how does the program know when it should produce the graphs?

Me: The user will enter the Product ID that they want to search for and then they can press on the buttons to produce the graphs.

Parth: Ok, so the user has control of what graphs they would like to see without the program instantly generating the graphs?

Me: Yes, the program will only generate the graphs when the user clicks on either of the 2 buttons.

Parth: Ok, got it. I am perfectly fine with this change.



I have shown Part where the buttons will be placed.

Part is fine with the design as long as both the buttons are of the same size and the same with the graphs to ensure the window looks symmetrical.

To ensure that the graphs are created within the Tkinter window, I have added this code at the beginning:

```
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg # provides a canvas that can display graph in tkinter window
```

This is a module that provides a canvas that can display the Matplotlib graphs in the Tkinter window.

Producing a value graph for a given product ID entered by the user for the last 7 days

```
def display_value_graph(self):
    # create a Matplotlib figure with a size of 5*5 inches
    fig = plt.Figure(figsize=(5, 5), dpi=100)
    ax = fig.add_subplot(111)

    # fetch data from the database and add it to the figure
    connection = sqlite3.connect(r'sms.db') # creating a database connection
    cursor = connection.cursor()
    select_value = cursor.execute("Select Date, Value FROM Sales WHERE Date > (SELECT DATE('now', '-7 day')) and "
                                  "Product_ID = ? ", (self.var_product_ID.get(),)).fetchall() # getting data within the last 7 days

    dates = [] # dates list
    value = [] # value list
    for row in select_value:
        dates.append(row[0]) # adding to dates list
        value.append(row[1]) # adding to value list
    # create a bar chart of the data
    ax.bar(dates, value, color='g', width=0.72, label="Value £")
    ax.set_xlabel('Date')
    ax.set_ylabel('Value')
    ax.set_title('The value of food sold (£)')
    ax.legend(loc="lower right")
```

```
# create a Tkinter canvas that can display the Matplotlib figure
canvas = FigureCanvasTkAgg(fig, master=self.wind)
canvas.draw()
canvas.get_tk_widget().place(x=900, y=20) # this sets the position of the graph in the Tkinter window
```

This code is similar to the previous code but it creates the Tkinter canvas widget that can display the graph.

The ‘FigureCanvasTkAgg’ function takes the first function as the Matplotlib figure object and the second argument is the main widget that contains the canvas which is self.wind.

The ‘canvas.draw()’ function updates the canvas to display the matplotlib figures.

The ‘canvas.get_tk_widget()’ function returns the Tkinter widget object for the canvas.

Producing a sold quantity graph for Product ID entered by the user for last 7 days

```
def display_sold_quantity_graph(self):
    # create a Matplotlib figure with a size of 5x5 inches
    fig = plt.Figure(figsize=(5, 5), dpi=100)
    ax = fig.add_subplot(111)

    # fetch data from the database and add it to the figure
    connection = sqlite3.connect('sms.db')
    cursor = connection.cursor()
    select_value = cursor.execute("Select Date, Sold FROM Sales WHERE Date > (SELECT DATE('now', '-7 day')) and "
                                  "Product_ID =? ", (self.var_product_ID.get(),)).fetchall() # getting the data for last 7 days
    dates = [] # dates list
    sold = [] # sold list
    for row in select_value:
        dates.append(row[0]) # adding to dates list
        sold.append(row[1]) # adding to sold list
    # create a bar chart of the data
    ax.bar(dates, sold, color='g', width=0.50, label="Sold in terms of Quantity")
    ax.set_xlabel('Date')
    ax.set_ylabel('Sold by Quantity')
    ax.set_title('The quantity of food sold')
    ax.legend(loc="lower right")
```

This code here produces the bar chart for sold quantity over a given time period with the graph embedded into the Tkinter window.

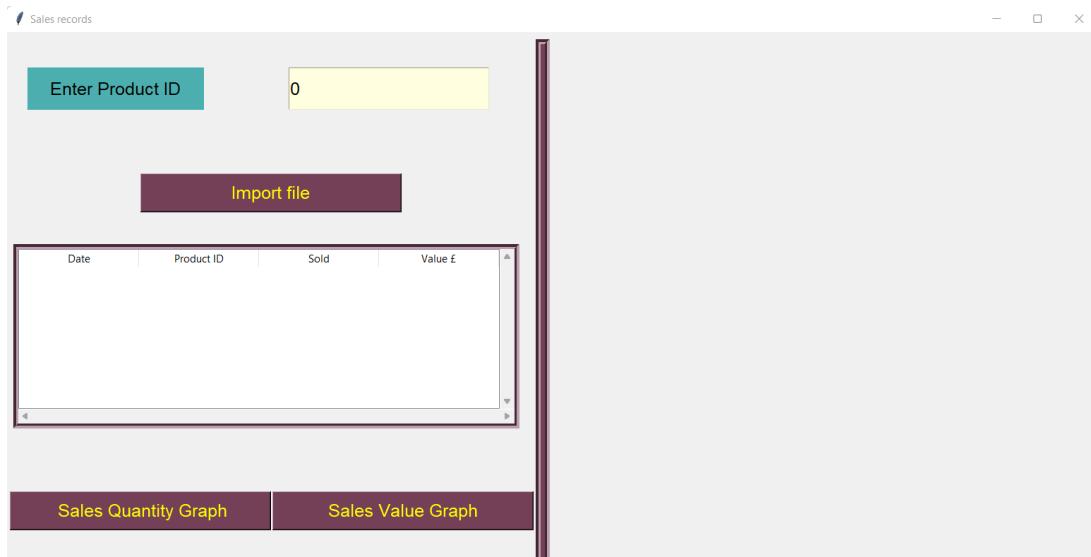
Creating the buttons

```
##### Quantity sold button #####
Quantity_sold_btn = Button(self.wind, text = " Sales Quantity Graph", command = self.display_sold_quantity_graph,
                           bg = "#734058", font = ("OCR A Std", 15), fg = "yellow").place(x = 5, y = 650, height = 55, width = 370)

##### Import file button #####
file_btn = Button(self.wind, text="Import file",bg = "#734058", command = self.import_files,
                  font = ("OCR A Std", 15), fg = "yellow").place(x = 190, y = 200, height = 55, width = 370) # buttons that will import the file

##### Value (£) button #####
value_btn = Button(self.wind, text = "Sales Value Graph", command = self.display_value_graph,
                   bg = "#734058", font = ("OCR A Std", 15), fg = "yellow").place(x = 377, y = 650, height = 55, width = 370)
```

This code here creates the 2 buttons – the quantity sold button and the value (£) button that are linked to their corresponding functions. I have adjusted the placement for the import file button so that it is placed higher in the Tkinter window so that the Sales Treeview can be positioned higher up as well. This allows space for the 2 buttons to be placed.

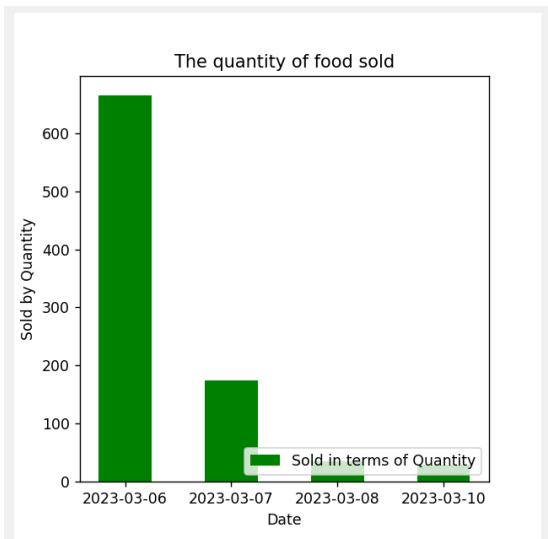


Here, I have placed the Quantity and Value graph to be on the same position in the window. This is to test the suitability of the size of the graph.

Testing: Sales Quantity Graph for last 7 days

43.	Produces Sales graphs by quantity and value for the last 7 days when a user enters product ID into the entry box	Existing Product ID	Existing Product ID (same as normal data)	Non-existing Product ID	Produces graph for the last 7 days for existing Product ID's
------------	------------------------------------------------------------------------------------------------------------------	---------------------	-------------------------------------------	-------------------------	--------------------------------------------------------------

When I click on the Quantity graph:

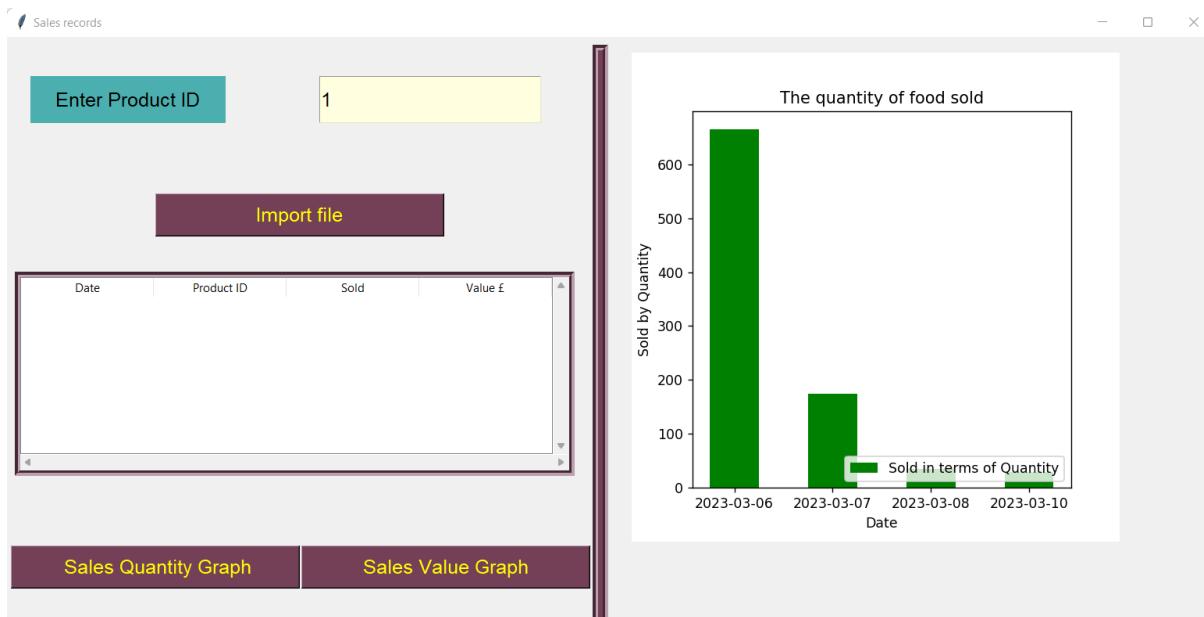


Dates in ascending order

Here, the graph shows all the values for the last 7 days. The reason there isn't 7 separate bar charts is because I haven't imported the sales data every day and so not all 7 charts are shown.

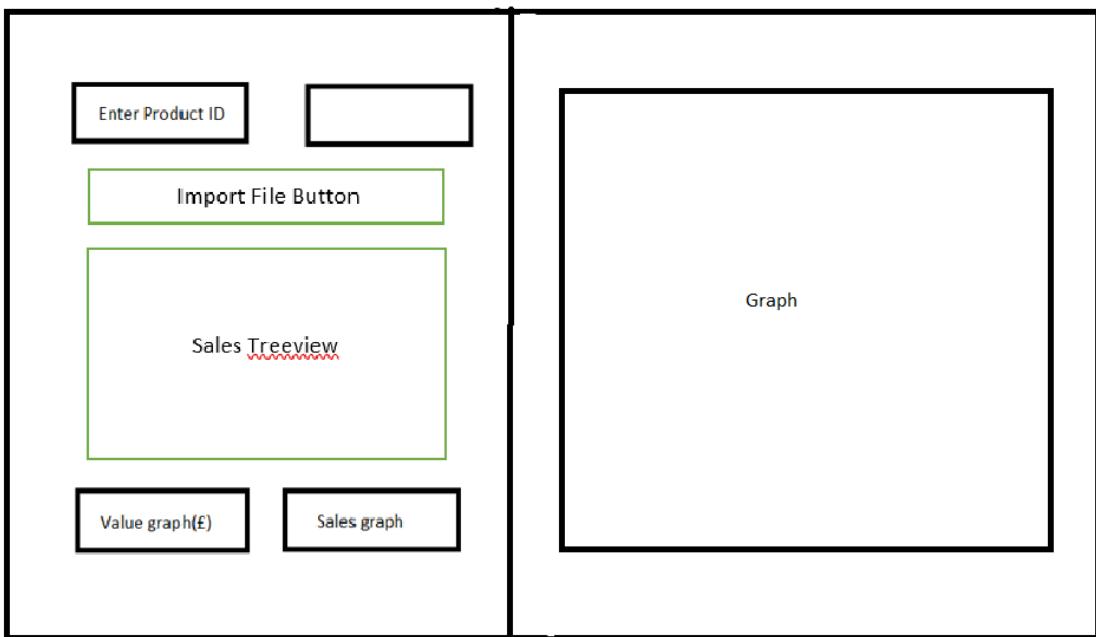
Here, each bar is equally spaced to ensure that the bars look consistent

Seeing the bar chart on the Tkinter Sales Window:



Here, as you can see only 4 bar charts have been shown and already the graph is taking a lot of space. I will do 2 things:

1. Ensure that there is only 1 graph shown on the window
2. Reduce the size of the 2 sales buttons and the treeview so that a graph with 7 charts can fit in



Here, I have changed the layout so that there is only 1 graph. If 2 graphs were included, then the graphs would be really small and it will be very difficult for my client to interpret the charts.

Client review

I have showed this to Parth:

Me: I have changed the design as shown above as the 2 graphs cannot fit on the window

Parth: Can't the 2 graphs be scaled down?

Me: The graphs can be scaled down, but the dates on the x-axis will overlap which can cause lot of confusion when interpreting the graphs.

Parth: Apart from the graphs, are there any other changes?

Me: No major changes. I will reduce the width of each column in the treeview and I will shorten the text on the button so that less space is wasted.

Date	Product ID	Sold	Value £
2023-03-07	13	700	3000
2023-03-08	1	35	140
2023-03-08	2	20	200
2023-03-08	3	100	500
2023-03-10	1	30	90
2023-03-10	2	65	130
2023-03-10	3	100	500
2023-03-10	4	20	200
2023-03-10	5	100	500
2023-03-10	7	35	140

Here, as you can see, there is a lot of blank space and so I will be reducing the width of the treeview columns.

Sales Quantity Graph

Sales Value Graph

As you can see here, a lot of space is being wasted due to the text of the button. I will reduce the text so that in both buttons the word “Sales” is removed.

Parth: I am fine with the changes in the button since I can easily distinguish between the two.

So what happens if I click on the Quantity graph and then the Value graph for the same Product ID?

Me: When you click on the Quantity graph button, that graph will automatically appear embedded into the Tkinter window. When you click on the Value graph, the first graph will be replaced with the value graph exactly as the same position in the window.

Parth: Ok, that seems fine to me. Can you change the title of the 2 graphs so that it mentions the name of the Product?

Me: Yes absolutely, I can use SQL query to retrieve the name of the Product through the Product table.

```
self.Sales_Table.column("Date", width=130, stretch=NO) # fixed width
self.Sales_Table.column("Product_ID", width=130, stretch=NO) # fixed width
self.Sales_Table.column("Sold", width=130, stretch=NO)
self.Sales_Table.column("Value", width=130, stretch=NO)
```

Here, I have changed the width of each treeview column from 170 to 130.

Date	Product ID	Sold	Value £
2023-03-06	1	665	2660
2023-03-06	2	360	3600
2023-03-06	3	1620	3240
2023-03-06	13	282	2880
2023-03-07	1	175	700
2023-03-07	2	120	1200
2023-03-07	3	540	1080
2023-03-07	13	700	3000
2023-03-08	1	35	140
2023-03-08	2	20	200

Compared to before, the treeview looks much less empty and so more space in the window has been created for the graph to fit in.

```
border_middle_frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=2000, width=20)
border_middle_frame.place(x=695, y=10) # places the middle frame into a specific coordinate
```

Here, I have changed the x position of the middle frame to 695 which is a shift to the left from its original position.

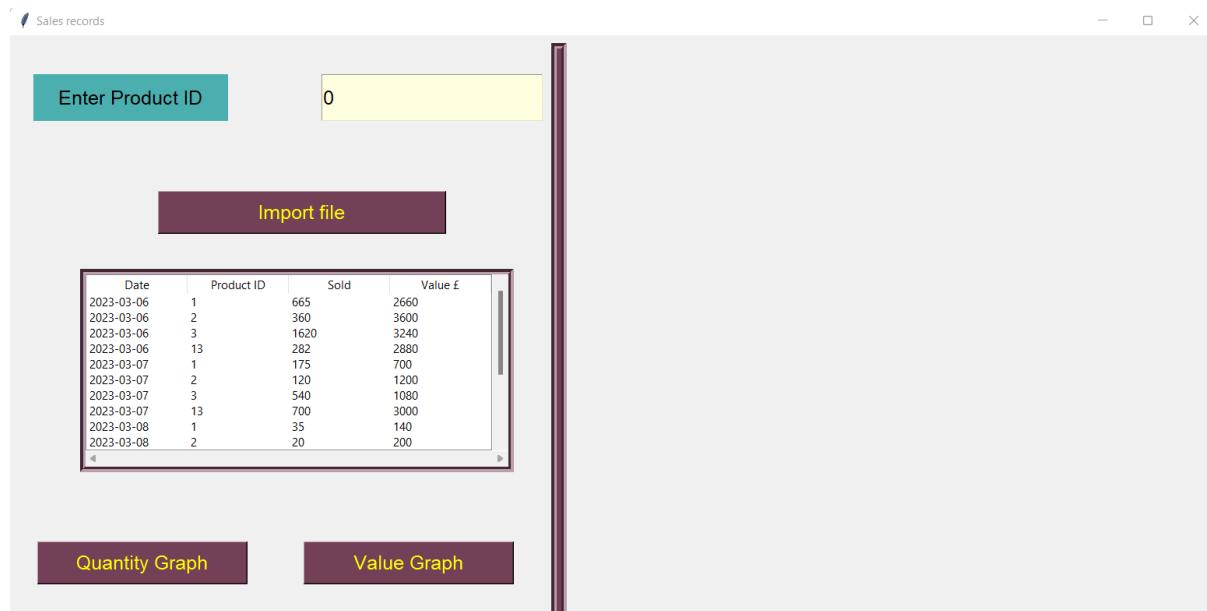
```
#####
# Quantity sold button #####
Quantity_sold_btn = Button(self.wind, text = "Quantity Graph", command = self.display_sold_quantity_graph,
                           bg = "#734058", font = ("OCR A Std", 15), fg = "yellow").place(x = 35, y = 650, height = 55, width = 270)
```

Here, I have changed the text on the Quantity graph button to “Quantity Graph” and have also reduced the width of the button to 270.

```
#####
# Value (£) button #####
value_btn = Button(self.wind, text = "Value Graph", command = self.display_value_graph,
                   bg = "#734058", font = ("OCR A Std", 15), fg = "yellow").place(x = 377, y = 650, height = 55, width = 270)
```

Same with the value button but the text is changed to “Value graph”.

Changed layout of sales window:



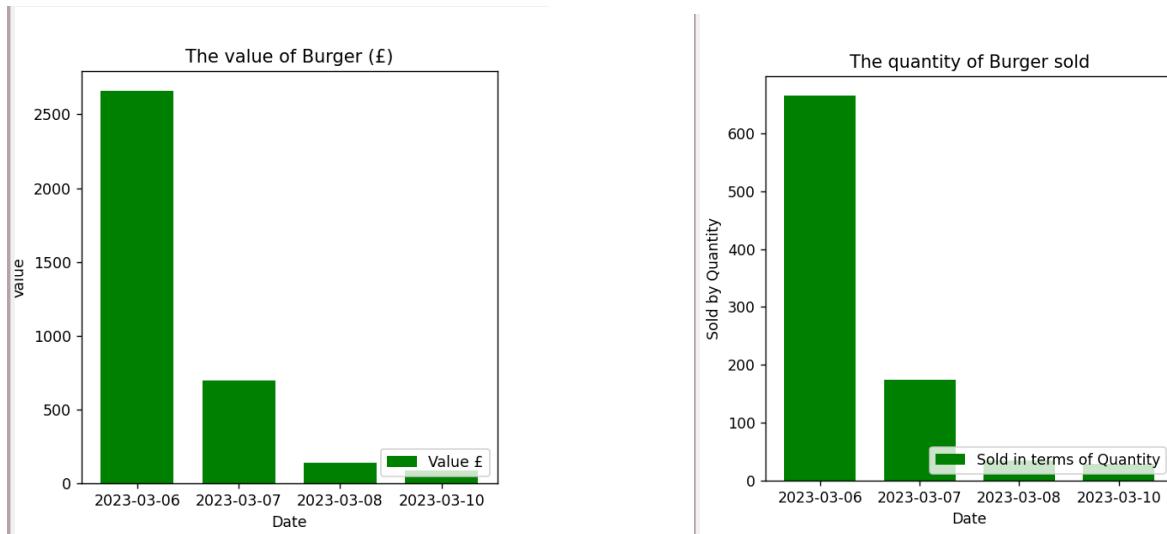
Here, as you can see there is much more space on the right section of the window to fit our graph.

```
# get the name of the Product from the Product table
name_of_product = cursor.execute("Select Product FROM Product WHERE Product_ID = ?", (self.var_product_ID.get(),)).fetchone()[0]
ax.set_title(f'The value of {name_of_product} (£)') # title with name of product
```

To include the name of the Product in the title, I have used SQL query to get the name of the Product based on the Product ID entered by the user on the entry box. This is stored in a variable which is then formatted to include the name of the Product in the title.

I have done the same for the Quantity graph title:

```
# get the name of product from the product table
name_of_product = cursor.execute("Select Product FROM Product WHERE Product_ID = ?".format(self.var_product_ID.get()),).fetchone()[0]
ax.set_title(f'The quantity of {name_of_product} sold') # title with name of product
```



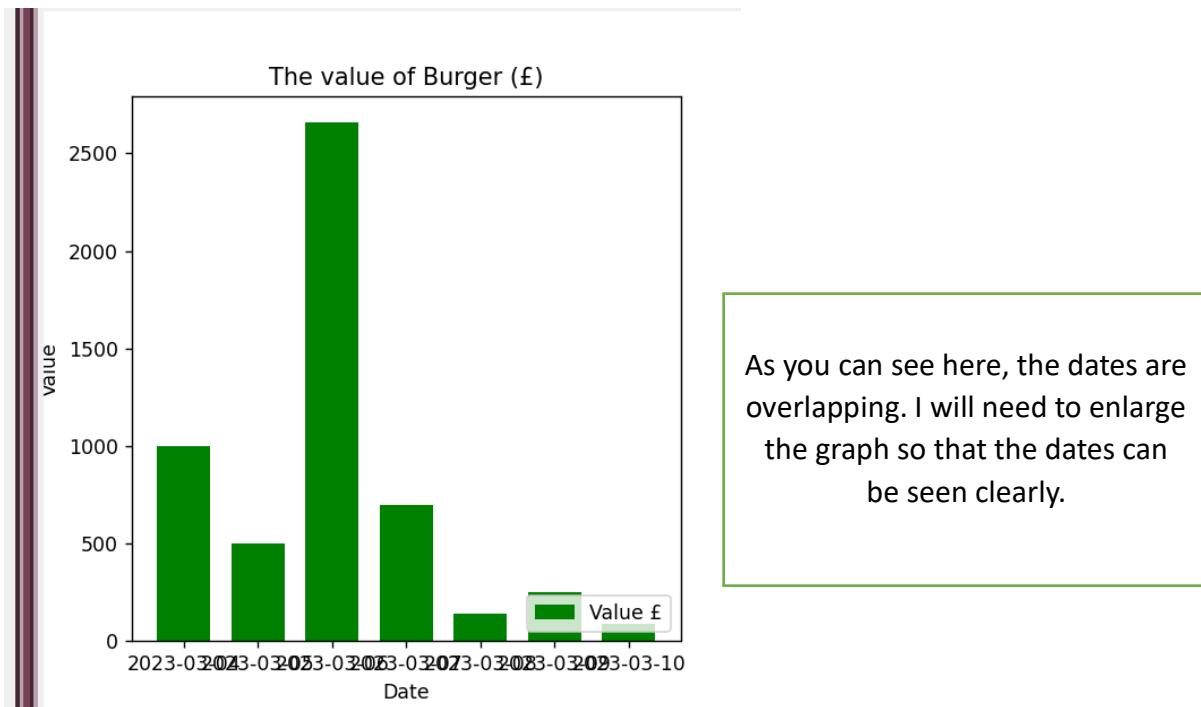
As you can see here, the name of the Product ("Burger") appears in both the graphs making it easier for my client to know which product data these graphs are being made.

The screenshot shows a software window titled "Sales records". It includes a text input field "Enter Product ID" with the value "1", a button "Import file", a table view of sales data, and two buttons at the bottom: "Quantity Graph" and "Value Graph". To the right of the table is a bar chart titled "The value of Burger (£)".

Date	Product ID	Sold	Value £
2023-03-06	1	650	2600
2023-03-07	1	180	700
2023-03-08	1	10	150
2023-03-10	1	5	100

In this screenshot, the first 4 charts fit easily. I will manually add sales data (Product ID: 1) to the database for the 4th, 5th and 9th of March.

130	2023-03-04	1	100	1000
131	2023-03-05	1	50	500
132	2023-03-09	1	25	250



I have changed the height and width of the graph by adjusting the 'figsize' parameter.

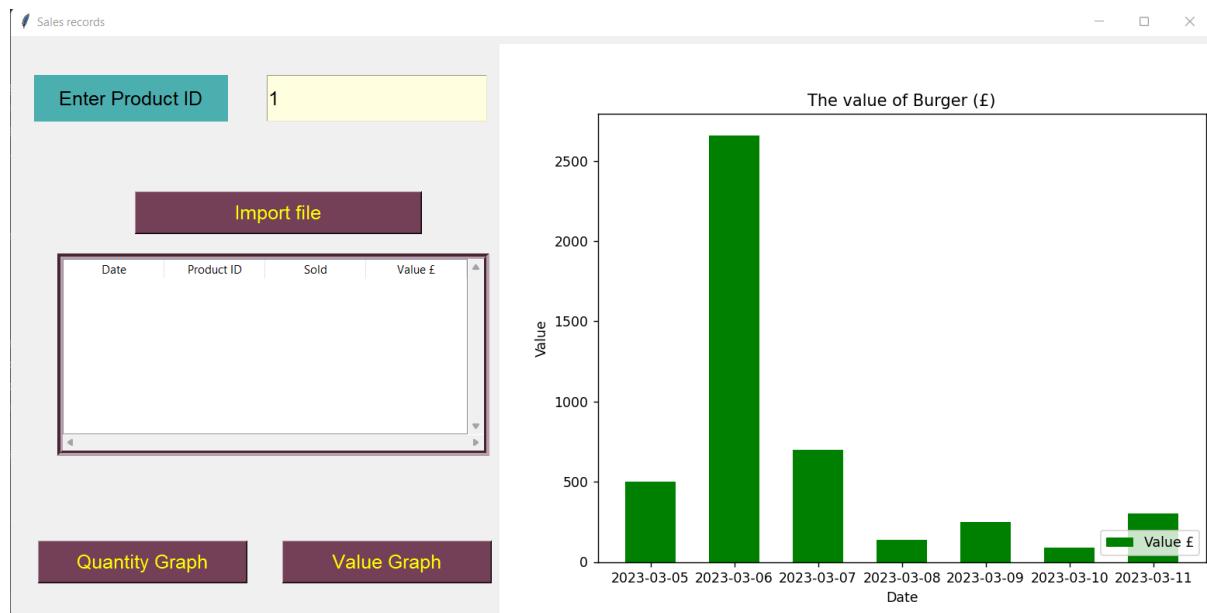
```
def display_value_graph(self):
    # create a Matplotlib figure with a size of 8.1* 5 inches
    fig = plt.Figure(figsize=(8.1, 6), dpi=100)
```

Here the width is 8.1 inches, and the height is 6 inches

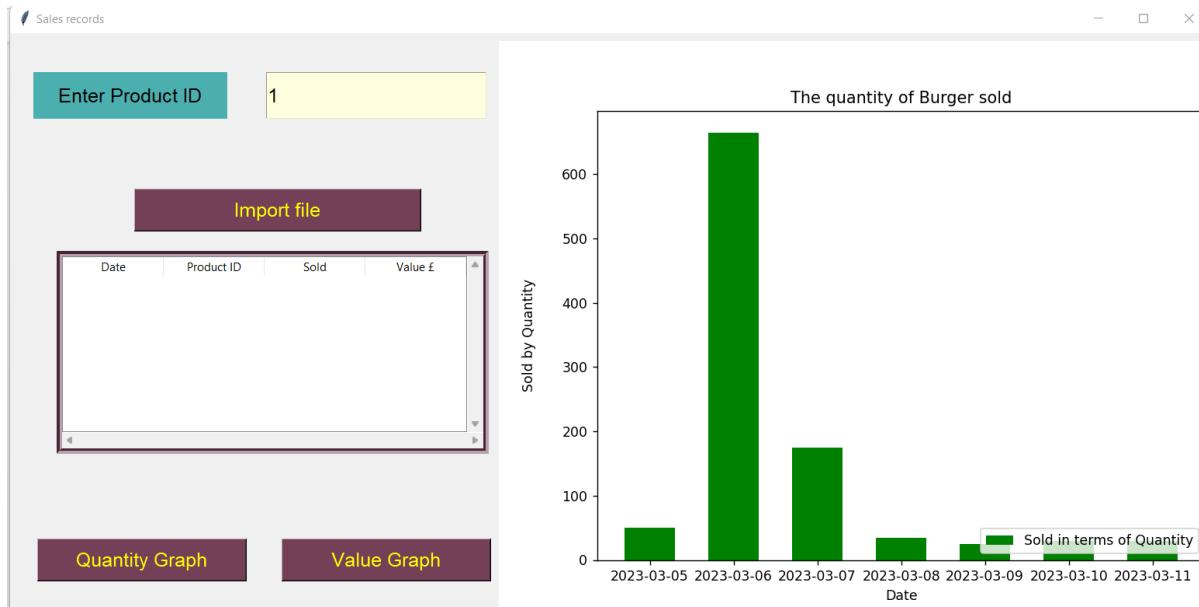
Testing: Value graph (£) for the last seven days

Normal data

Have used existing Product ID as 1. Here, I get 7 individual bars that are within 1 week.



I have adjusted the same height and width for the quantity graph as well. When the user presses the Quantity graph:

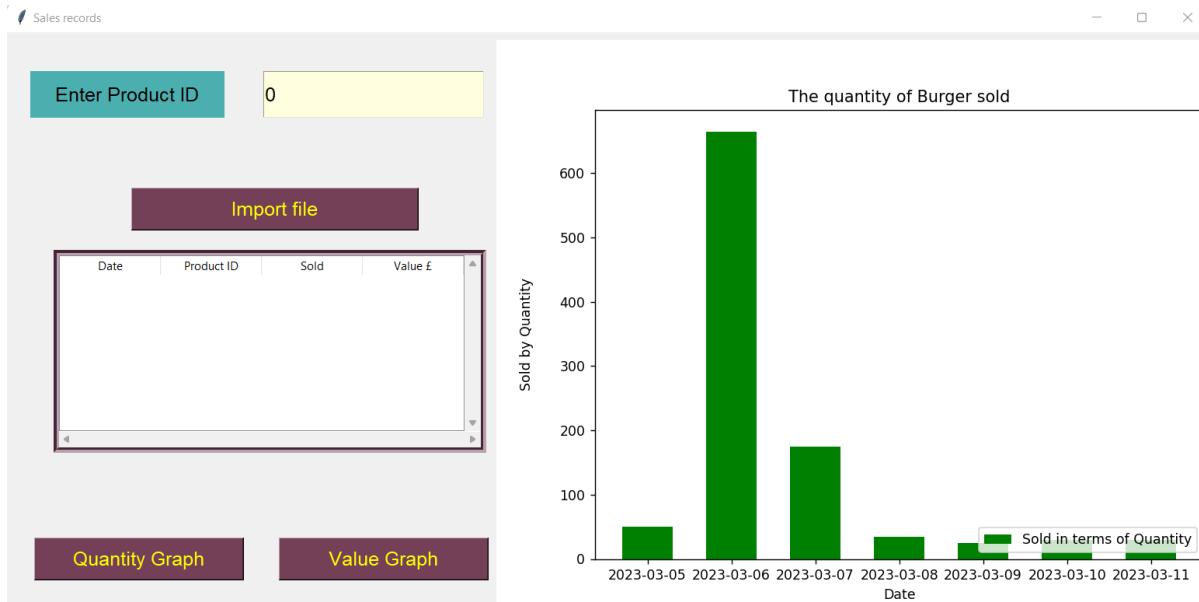


The graph gets exactly replaced with no change in position.

Here, all the 7 dates can be seen clearly with no overlaps at all. The graph is perfectly made for the last 7 days as there are 7 individual bars.

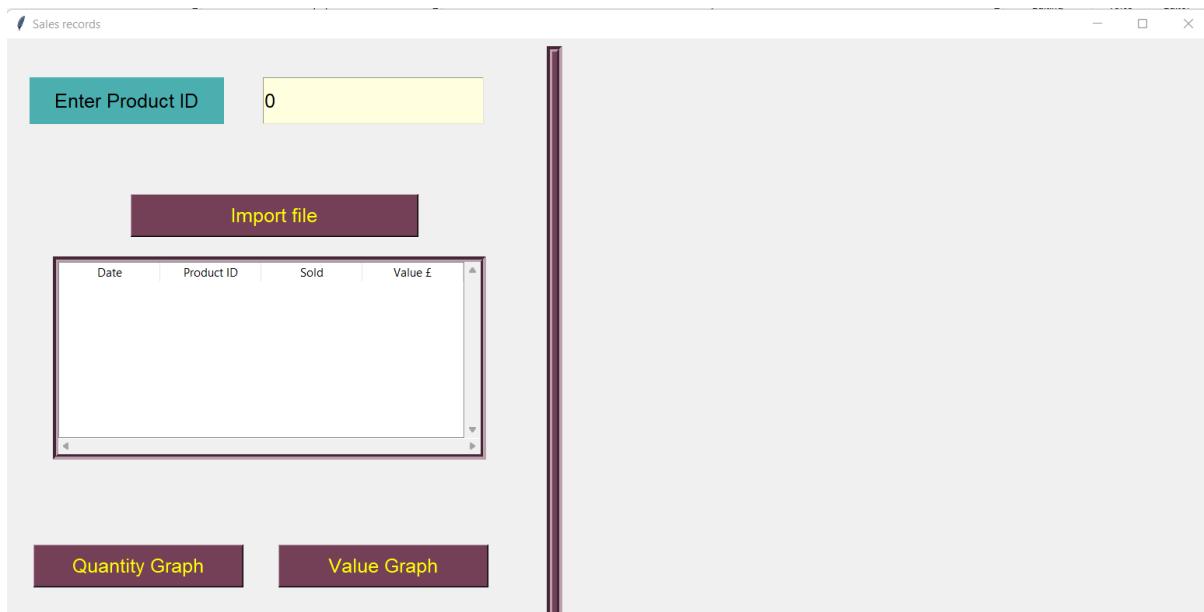
Erroneous data

If the user enters a non-existing Product ID such as 0, then no graph should be made. The graph will not change if there is an existing graph on the window already.



Here, as you can see, I have entered Product ID 0 which does not exist in the Product table.

If I enter Product ID 0 with no previous graphs embedded into the window, I should get no graph produced as shown below.



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
43.	Produces Sales graphs by quantity and value for the last 7 days when a user enters product ID into the entry box	Existing Product ID	Existing Product ID (same as normal data)	Non-existing Product ID	Produces graph for the last 7 days for existing Product ID's	Passed – Graph produced for existing Product ID's. No graph / no new graph produced for non-existing Product ID's

Client review: I have shown my sales window program to Parth who is happy with the functionality.

Parth: I really like the Sales window and the graphs made look really nice and easy to read. Can you create a messagebox that can forecast sales for the next day (Quantity and Value for a given Product ID) and also for the next month (total sales for the next month).

Me: Yes, this is possible but it will take some time. I can create a messagebox that can be displayed after the user clicks on one of the graph buttons. For example, if the user clicks on the Quantity graph, after the graph is displayed, the messagebox showing the forecast for tomorrow and the next month can be displayed one after the other.

Parth: Yes, this is perfect. As long as I can get a rough estimate of the forecast.

Me: The only problem is that this forecast program may not give a reliable output as there is not much data to work from.

Parth: I don't really mind as long as I can get a predicted value. I will also use my interpretation to check if my estimated value matches with the program's predicted value for a given Product ID.

ARIMA (Autoregressive Integrated Moving Average model)

Based on the last interaction with my client, I have decided to use a ARIMA model which is a time-series forecasting technique used to predict future sales based on past data.

Due to limited time, I haven't got much in depth but have made sure that I have a good understanding of how this model works.

There are some main features of this model:

- Autoregression – The ideas that future values are linked to the past values. If the past values are high, the predicted value is also going to be high.
- Moving Average – Uses the relationship between the current values and the error terms (the difference between the actual value and the predicted value) of the previous data in the time series

Importing new libraries

```
from statsmodels.tsa.arima.model import ARIMA # allows ARIMA model to be used
```

This allows the ARIMA model to be used for forecasting.

Code to forecast Value(£) for Tomorrow

```
def forecast_value_in_pounds_tomorrow(self):
    conn = sqlite3.connect(r'sms.db') # connect to the database
    # sql statement to fetch the Date, and value records that are then ordered by their date
    query = "SELECT strftime('%Y-%m-%d', Date) AS Date, SUM(Value) AS Total_Value FROM Sales WHERE Product_ID =? GROUP by Date ORDER BY Date"
    # params argument used to pass the Product ID obtained from its variable as a parameter to the query
    df_sales = pd.read_sql_query(query, conn, params=[self.var_product_ID.get()])
```

Here, I have created an SQL statement to fetch all the Date and Value data for the given Product ID which is then ordered by its date.

Here, I have used the ‘pd.read_sql_query()’ function to execute the SQL query and fetch the records in the form of a pandas DataFrame object.

The ‘strftime()’ function is used to format the ‘Date’ column in the form of ‘YYYY-MM-DD’.

The ‘params’ argument is used to pass the value of ‘self.var_product_ID.get()’ as a parameter to the Query so that the data related to that Product ID entered by the user is fetched.

```

df_sales.index = pd.to_datetime(df_sales['Date']) # sets the index of the dataframe to the Date column
df_sales = df_sales.drop(columns=['Date']) # drops date column as it is no longer needed
df_sales = df_sales.asfreq('D') # sets the frequency of the data to daily
df_sales.fillna(0, inplace=True) # fills missing values with 0

# using an arima model for the sales data
model = ARIMA(df_sales, order=(1, 1, 1), freq='D') # p=1, d=1, q=1
result = model.fit() # fits the model

```

Here the index of the dataframe is set to the Date column and then the 'Date' column is dropped from the dataframe as it is no longer needed for analysis. The frequency of the data is set to daily using the asfreq method. Any missing values or sales that haven't occurred for that day are filled with 0 using the fillna method.

An ARIMA model is created by passing 3 different values:

- P: which tells the program how many previous data point you would like to consider when making the next predicted data point.
- d: order of differencing – When the sales data is non-stationary, we can difference the data until it is stationary. d = 1 means it has been differenced once. Differencing is performed by subtracting the previous observation from the current observation.
- q: represents the number of past error terms (the differences between the observed values and the predicted values) that are used in the model.

The model is then fit using the 'fit' method. Here, I have called this function after the value graph button has been clicked.

```

# create a Tkinter canvas that can display the Matplotlib figure
canvas = FigureCanvasTkAgg(fig, master=self.wind)
canvas.draw()
canvas.get_tk_widget().place(x=630, y=10) # this sets the position of the graph in the Tkinter window
self.forecast_value_in_pounds_tomorrow() # calls the function to forecast value for tomorrow

```

```

# Forecast future sales
forecast = result.forecast(steps=1, typ='levels') # forecast for the next day
message = f"The total value (£) forecast for Product ID {self.var_product_ID.get()} for the next day is: {forecast[0]:.2f}"
messagebox.showinfo("Value Forecast", message, parent=self.wind) # messagebox showing forecast in value for tomorrow

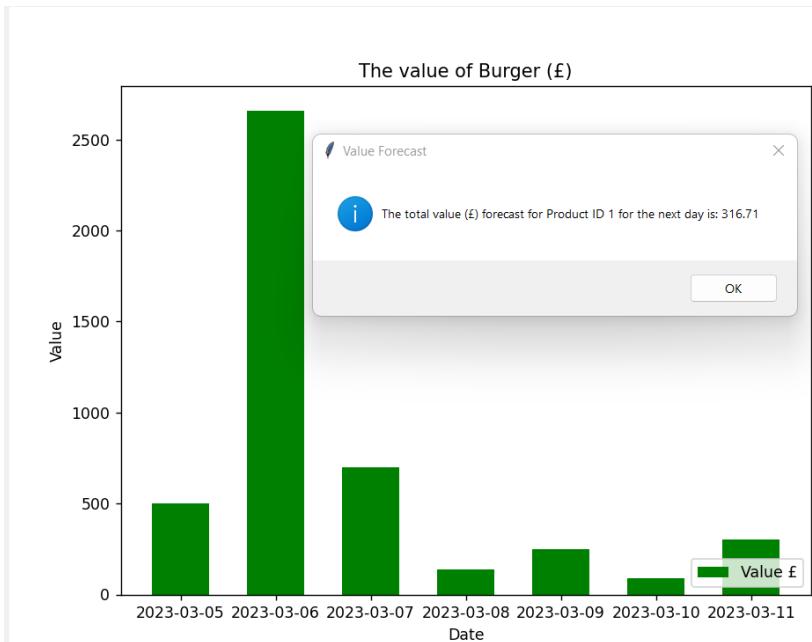
```

The 'forecast' method is used to generate predictions of future values based on the past data given. Steps = 1 tells the program that we want to generate the forecast for tomorrow.

'typ = levels' specifies that we want to forecast the sales value in pounds rather than the differences between data.

Testing: Forecast of value made for tomorrow for a given Product ID

When I press the value button:



Here, a message box successfully appears showing the forecast value in (£) for tomorrow for Product ID 1 (burger). Here, it has predicted the value for burger for tomorrow to be £316.71 meaning that this is how much money will be made by my client for that product tomorrow.

Code to forecast Quantity sold for Tomorrow

```
def forecast_quantity_sold_tomorrow(self):
    conn = sqlite3.connect(r'sms.db') # connect to the database
    # sql statement to fetch the Date, and Sold records that are then ordered by their date
    query = "SELECT strftime('%Y-%m-%d', Date) AS Date, SUM(Sold) AS Total_Sold FROM Sales WHERE Product_ID =? GROUP BY Date ORDER BY Date"
    # params argument used to pass the Product ID obtained from its variable as a parameter to the query
    df_sales = pd.read_sql_query(query, conn, params=[self.var_product_ID.get()])

    df_sales.index = pd.to_datetime(df_sales['Date'])# sets the index of the dataframe to the Date column
    df_sales = df_sales.drop(columns=['Date'])# drops date column as it is no longer needed
    df_sales = df_sales.asfreq('D')# sets the frequency of the data to daily
    df_sales.fillna(0, inplace=True)# fills missing values with 0

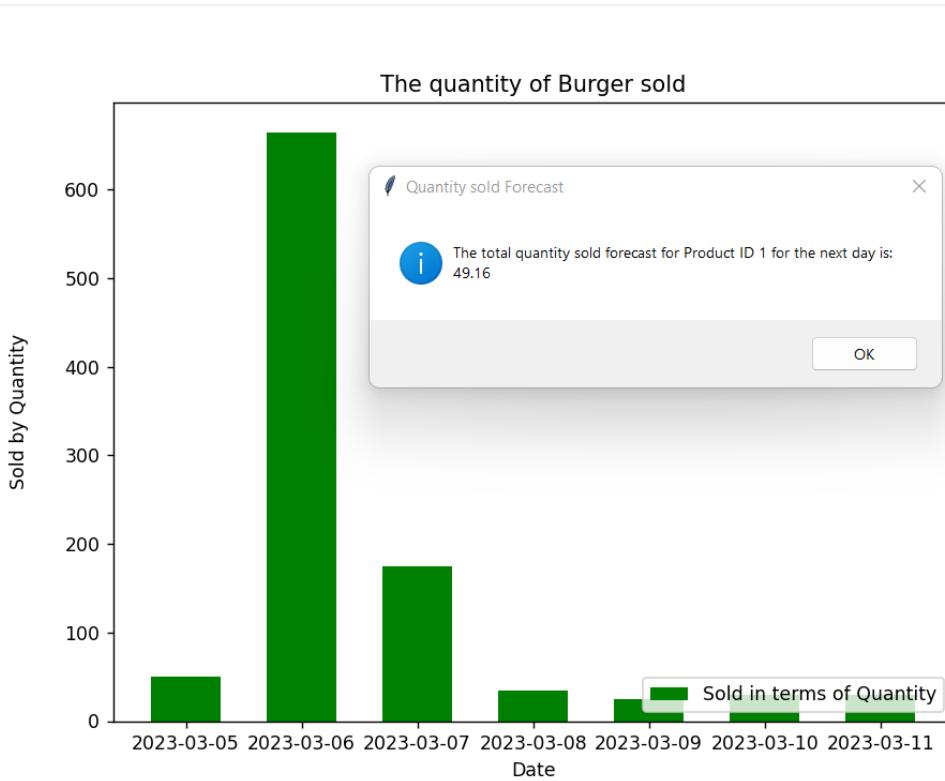
    # using an arima model for the sales data
    model = ARIMA(df_sales, order=(1, 1, 1), freq='D') # p=1, d=1, q=1
    result = model.fit()# fits the model

    # Forecast future sales
    forecast = result.forecast(steps=1, typ='levels') # forecast for the next day
    message = f"The total quantity sold forecast for Product ID {self.var_product_ID.get()} for the next day is: {forecast[0]:.2f}"
    messagebox.showinfo("Quantity sold Forecast", message, parent=self.wind) # messagebox showing forecast in Quantity sold for tomorrow
```

Here, the same structure has been used for the previous code but instead of value it is sold.

Testing: Forecast of total quantity sold for a given Product ID for tomorrow

When I click the Quantity graph button:



The program shows a message box showing the total quantity sold for tomorrow for Product ID 1 to be around 49. This means that around 49 burgers should be sold by my client tomorrow according to the model.

Code to forecast total value(£) made for next month for a given Product ID

```
def forecast_value_in_pounds_for_next_month(self):
    conn = sqlite3.connect(r'sms.db') # connecting to the database
    # Date and sum of all value made for a given product ID per month
    query = "SELECT strftime('%Y-%m', Date) AS Month, SUM(Value) AS Total_Value FROM Sales WHERE Product_ID =? GROUP BY Month ORDER BY Month"
    df_sales = pd.read_sql_query(query, conn, params=[self.var_product_ID.get()])
    df_sales.index = pd.to_datetime(df_sales['Month']) # sets index of the dataframe to the Month column
    df_sales = df_sales.drop(columns=['Month']) # drops month column as it is no longer needed
    df_sales = df_sales.asfreq('MS') # sets frequency of the data to monthly
    df_sales.fillna(0, inplace=True) # if data is blank, fill it with 0

    model = ARIMA(df_sales, order=(1, 1, 1), freq='MS') # p=1, d=1, q=1
    result = model.fit() # fit the model with the sales data

    # Forecast future sales
    forecast = result.forecast(steps=1, typ='levels')

    message = f"The total value (£) forecast for Product ID {self.var_product_ID.get()} next month is: {forecast[0]:.2f}"
    messagebox.showinfo("Value Forecast", message, parent=self.wind) # message to forecast total value for next month for the given product ID
```

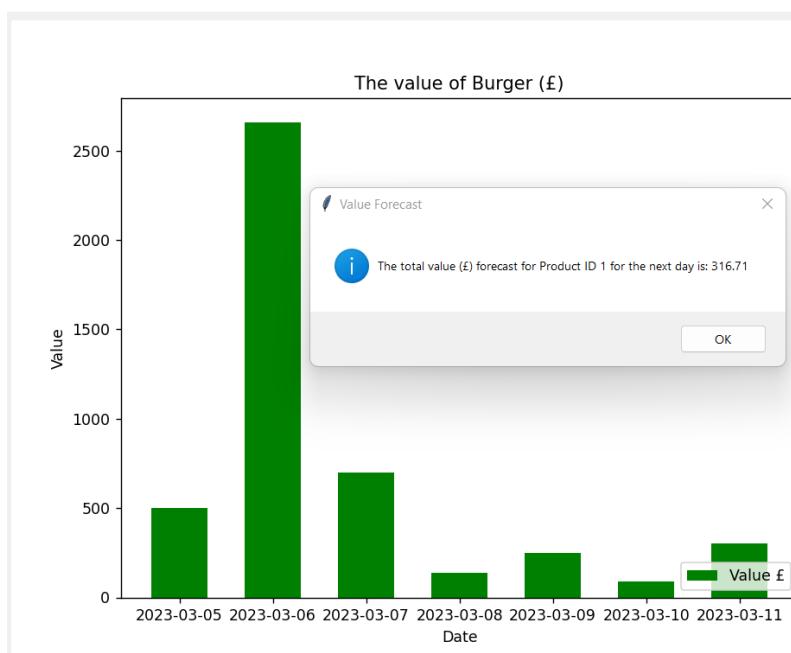
This function forecasts the total value in (£) for a given Product ID for the next month. Again the code is very similar, but instead the SQL statement adds up all the value for the Product ID by Months which is then used to forecast the total value for next month. Here, the frequency of the data has been set to monthly as the data used in the ARIMA model is by each month.

```
# create a Tkinter canvas that can display the Matplotlib figure
canvas = FigureCanvasTkAgg(fig, master=self.wind)
canvas.draw()
canvas.get_tk_widget().place(x=630, y=10) # this sets the position of the graph in the Tkinter window
self.forecast_value_in_pounds_tomorrow() # calls the function to forecast value for tomorrow
self.forecast_value_in_pounds_for_next_month() # calls the function to forecast value for next month
```

I have called the function after the program calls the function to forecast the value for tomorrow. For the second function to work, the first function will run and then when the user presses the close button on the message box, that is when the second function will run.

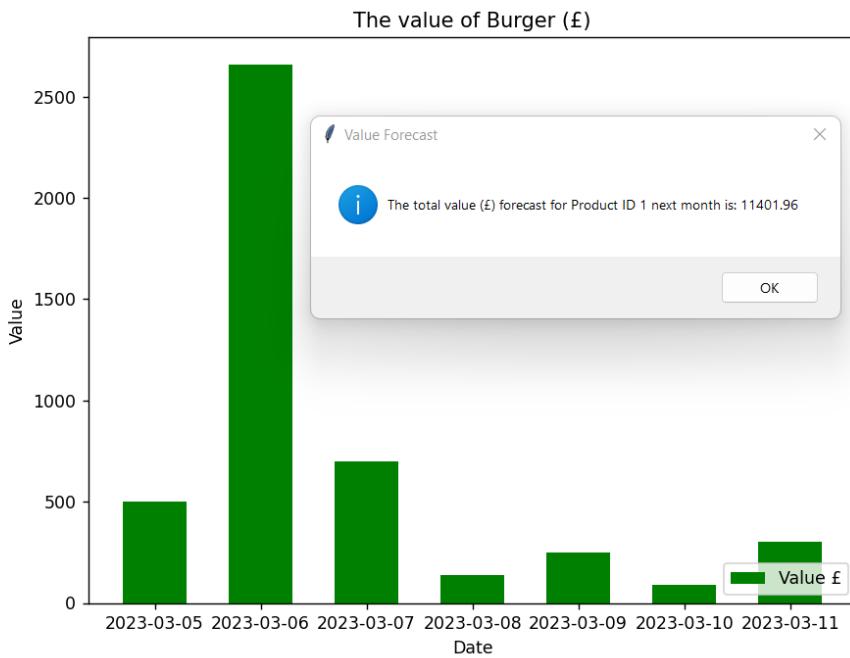
Testing: Forecast total value (£) for next month for a given Product ID

When I press the value graph button:



Here, the first message box appears showing the forecast for tomorrow.

When I press the close button or the ok button:



Here, a message is being displayed showing the total forecast in value for Product ID 1 for next month to be around £11400.

Code to forecast total quantity sold for next month for a given Product ID

```
def forecast_Quantity_sold_for_next_month(self):
    conn = sqlite3.connect('sms.db') # connecting to database
    # Date and sum of all sold quantity within that month for a given product ID
    query = "SELECT strftime('%Y-%m' , Date) AS Month, SUM(Sold) AS Total_Sales FROM Sales WHERE Product_ID = ? GROUP by Month ORDER BY Month"
    df_sales = pd.read_sql_query(query, conn, params=[self.var_product_ID.get()])
    df_sales.index = pd.to_datetime(df_sales['Month']) # sets the index of the dataframe to the Month column
    df_sales = df_sales.drop(columns=['Month']) # drops month column as it is no longer needed
    df_sales = df_sales.asfreq('MS') # sets frequency of the data to monthly
    df_sales.fillna(0, inplace=True) # if data is blank, fill it with 0

    model = ARIMA(df_sales, order=(1,1,1), freq='MS') # p=1, d=1, q=1
    result = model.fit() # fit the model with the sales data

    # Forecast future sales
    forecast = result.forecast(steps=1, typ='levels') # forecast for the next month

    message = f"The total sales (in terms of quantity) forecast for Product ID {self.var_product_ID.get()} next month is: {forecast[0]:.2f}"
    messagebox.showinfo("Sales Forecast", message, parent=self.wind) # message to forecast total quantity sold for the given Product ID
```

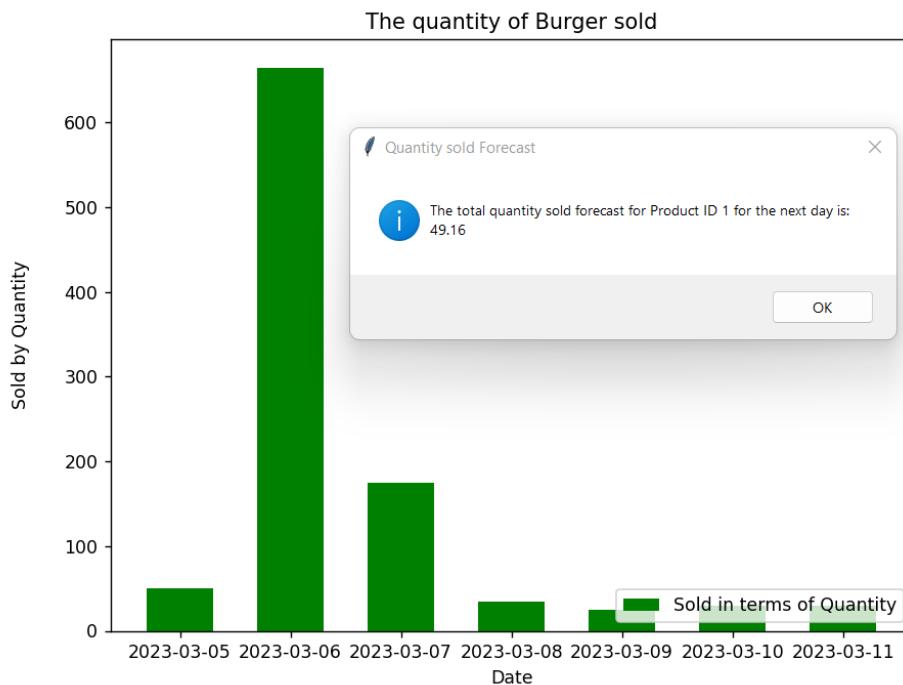
This code is similar to the previous one but this time instead of value it is total quantity sold.

This code forecasts the total quantity sold next month for a given product ID.

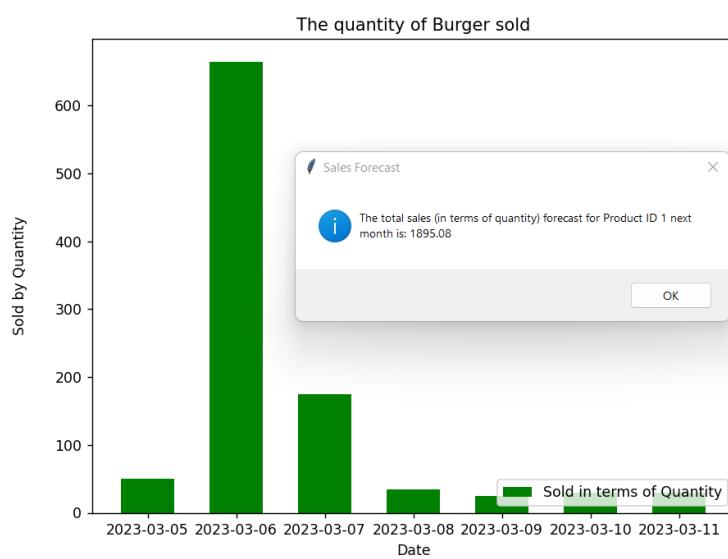
```
# create a Tkinter canvas that can display the Matplotlib figure
canvas = FigureCanvasTkAgg(fig, master=self.wind)
canvas.draw()
canvas.get_tk_widget().place(x=630, y=10) # this sets the position of the graph in the Tkinter window
self.forecast_quantity_sold_tomorrow() # calls function
self.forecast_Quantity_sold_for_next_month() # calls function
```

Again, the first function to forecast the quantity sold tomorrow will run first, and then once the user clicks on the close button for the message box, the second function will run.

Testing: Forecasting the quantity sold next month for a given product ID



When the user closes the first message box:



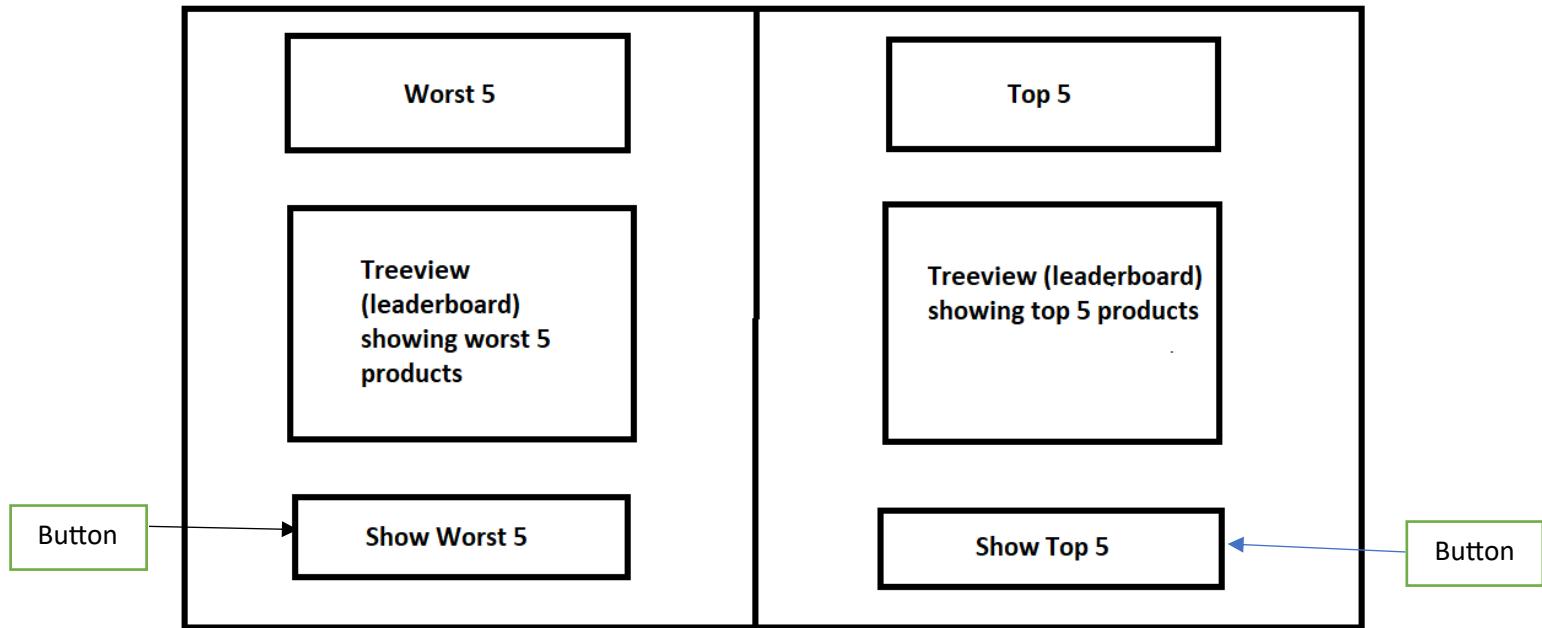
The second function runs and the second message box appears showing the forecast for the total quantity sold for Product ID 1 next month to be around 1895.

Client review:

I have shown this to Parth who is satisfied with the forecasting programs.

Stage 5: Analysis Window

Based on how much space the graph took in the Sales window, I have decided to change the design of my Analysis window so that instead of the 2 graphs (top5 and worst 5 product graphs) being embedded into the Tkinter window, I have decided to use buttons that will show the graphs in a completely different window.



Client

I have shown this to Parth who is happy with the changes.

Parth: Will the treeview show the top 5 products and worst 5 products automatically?

Me: Yes, when you click on the Analysis section, the treeview will show the worst and top 5 products automatically without being dependent on any buttons to be clicked.

```
# Label showing worst 5 products
worst_5_lbl = Label(self.wind, text = "Worst 5 Products", font = ("Rosewood Std Regular",30), bg = "#4cafaf", fg = "black")\
.place(x=70, y = 80, height = 100, width = 500)

# Label showing top 5 products
top_5_lbl = Label(self.wind, text = "Top 5 Products", font = ("Rosewood Std Regular",30), bg = "#4cafaf", fg = "black")\
.place(x=870, y = 80, height = 100, width = 500)

# This frame will contain the treeview showing the best 5 products
top_5_frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=2000, width=2000)
top_5_frame.place(x=850, y=250) # placing the frame

# This frame will contain the treeview showing the worst 5 products
worst_5_frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=2000, width=2000)
worst_5_frame.place(x=60, y=250) # placing the frame

##### this is a frame to split the entire window into half #####
Middle_Frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=3000, width=20)
Middle_Frame.place(x=750, y=5)
```

Here, I have created 3 frames and 2 labels:

- worst_5_lbl – This shows the section of the window that produces the graph and leaderboard for the worst 5 products
- top_5_lbl – This shows the section of the window that produces the graph and leaderboard for the top 5 products
- top_5_frame – This contains the treeview to show leaderboard of top 5 products
- worst_5_frame – This contains the treeview to show leaderboard of worst 5 products
- Middle_Frame - This splits the window into half – one half showing top 5 products and the other half showing the worst 5 products

```
#####
# Treeview for top 5 products #####
#####

rightscrolly = Scrollbar(top_5_frame, orient=VERTICAL),# vertical scrollbar
rightscrollx = Scrollbar(top_5_frame, orient=HORIZONTAL),# horizontal scrollbar

# defining the treeview
self.best_5_table = ttk.Treeview(top_5_frame, columns=("Product_ID", "Product", "Sold"), yscrollcommand=rightscrolly.set, xscrollcommand=rightscrollx.set)
rightscrollx.pack(side=BOTTOM, fill=X)
rightscrolly.pack(side=_RIGHT, fill=Y)
rightscrollx.config(command=_self.best_5_table.xview)
rightscrolly.config(command=_self.best_5_table.yview)

self.best_5_table.heading("Product_ID", text="Product_ID")
self.best_5_table.heading("Product", text=_"Product")
self.best_5_table.heading("Sold", text=_"Sold")

self.best_5_table["show"] = "headings"

self.best_5_table.column("Product_ID", width=170, stretch=NO),# fixed width
self.best_5_table.column("Product", width=170, stretch=NO),# fixed width
self.best_5_table.column("Sold", width=170, stretch=NO),# fixed width
```

This code above creates the treeview for the top 5 products.

```
#####
# Treeview for the worst 5 products #####
#####

scrollly = Scrollbar(worst_5.frame, orient=VERTICAL),# vertical scrollbar
scrolllx = Scrollbar(worst_5.frame, orient=HORIZONTAL),# horizontal scrollbar

# least 5 treeview
self.least_5_table = ttk.Treeview(worst_5.frame, columns=("Product_ID", "Product", "Sold"), yscrollcommand=scrollly.set, xscrollcommand=scrollx.set)
scrollx.pack(side=BOTTOM, fill=X)
scrolly.pack(side=RIGHT, fill=Y)
scrollx.config(command=self.least_5_table.xview)
scrolly.config(command=self.least_5_table.yview)

self.least_5_table.heading("Product_ID", text="Product_ID")
self.least_5_table.heading("Product", text=_"Product")
self.least_5_table.heading("Sold", text="Sold")

self.least_5_table["show"] = "headings"

self.least_5_table.column("Product_ID", width=170, stretch=NO),# fixed width
self.least_5_table.column("Product", width=170, stretch=NO),# fixed width
self.least_5_table.column("Sold", width=170, stretch=NO),# fixed width
```

This code above creates the treeview for the worst 5 products.

Creating graph to show top 5 products within last 7 days.

```

def top_5_products(self):
    connection = sqlite3.connect(r'sms.db') # connect to the database
    cursor = connection.cursor()
    # this uses the Product ID and sums up all the quantity sold within one week. It is ordered in ascending order
    cursor.execute("SELECT Product_ID, SUM(Sold) AS total_revenue from Sales WHERE Date > (SELECT DATE('now', '-7 day')) "
                  "GROUP BY Product_ID ORDER BY SUM(Sold) ")
    show_top_5 = cursor.fetchall() # this fetches all the data from the sql query
    top_5 = (show_top_5[-5:]) # this shows the last 5 products from the retrieved data
    product_ID = [] # creates a list to store multiple Product ID's
    sold = [] # creates a list to store multiple sold data
    for row in top_5: # iterates through the top 5 values
        product_ID.append(row[0]) # adds to the list
        sold.append(row[1]) # adds to the list

```

This code here uses an SQL query to fetch all the Product ID's and add up all the quantity sold for ach product ID within one week. This is ordered by the sum sold in ascending order.

My code then takes the last 5 products from the 'show_top_5' variable and then adds them to 2 different lists that store the Product ID and the total quantity sold for a product ID within one week using a for loop.

```

# Set the width and gap between bars in the bar chart
width = 0.8
gap = 0.05

# Compute the positions of each bar
positions = range(1, len(top_5) + 1)
positions = [pos - width / 2 - gap for pos in positions]

# create the bar chart
plt.bar(positions, sold, width=width, color='c', label="Quantity sold")
plt.xticks(range(1, len(top_5) + 1), product_ID)

plt.xlabel('Product_ID') # x axis title
plt.ylabel('Sold') # y axis title
plt.title('Top 5 performing products') # title
plt.tight_layout()
plt.legend()
plt.show() # display the bar chart

```

This code here defines the features of the graphs and ensures that there is no gap between a wide range of products. For example, if Product IID 1 and 100 are part of the top 5 products, then there shouldn't be a massive gap in the graph. Each bar will spaced evenly from each other no matter how large the range of Product ID's.

```

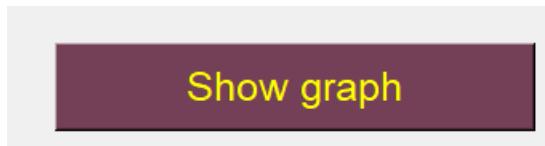
# Button that shows the top 5 products in a graph format when clicked
top_5_btn = Button(self.wind, text="Show graph", command=self.top_5_products, bg="#734058",
                    font=("OCR A Std", 15), fg="yellow").place(x=1000, y=670, height=55, width=300)

```

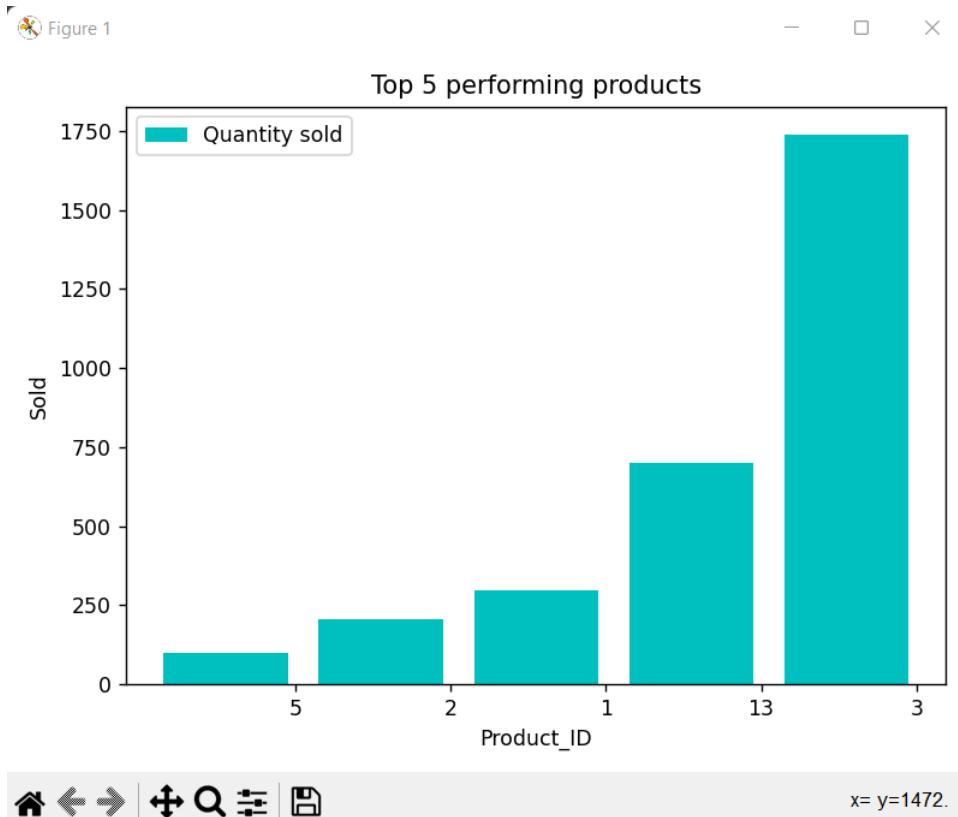
This code here creates a button that links backs to the function above.

Testing: Graph showing top 5 products

This is how the button looks like:



When I click on the button:



Here, I get a new window showing the graph from the 2 lists. As you can see, the graph orders the bar in ascending order making it easier for my client to quickly interpret the graph.

However, I have changed the code to make it more efficient. Instead of using indexing to get the top 5 products, I could use the LIMIT clause to limit the program to fetch only the first 5 records. This way, I reduce the number of lines needed in my code.

```
def top_5_products(self):
    connection = sqlite3.connect(r'sms.db') # connect to the database
    cursor = connection.cursor()
    # this uses the Product ID and sums up all the quantity sold within one week. It is ordered in descending order
    cursor.execute("SELECT Product_ID, SUM(Sold) AS total_revenue FROM Sales WHERE Date > (SELECT DATE('now', '-7 day')) "
                  "GROUP BY Product_ID ORDER BY SUM(Sold) DESC LIMIT 5") # limit 5 means only the first 5 products are fetched
    show_top_5 = cursor.fetchall() # this fetches all the first (top) 5 products
    product_ID = [] # creates a list to store multiple Product ID's
    sold = [] # creates a list to store multiple sold data
    for row in show_top_5: # iterates through the top 5 values
        product_ID.append(row[0]) # adds to the list
        sold.append(row[1]) # adds to the list
```

Here, I have ordered the total quantity sold in descending order to get the highest sold product as the first data. This way, I can use the LIMIT clause to only fetch 5 products.

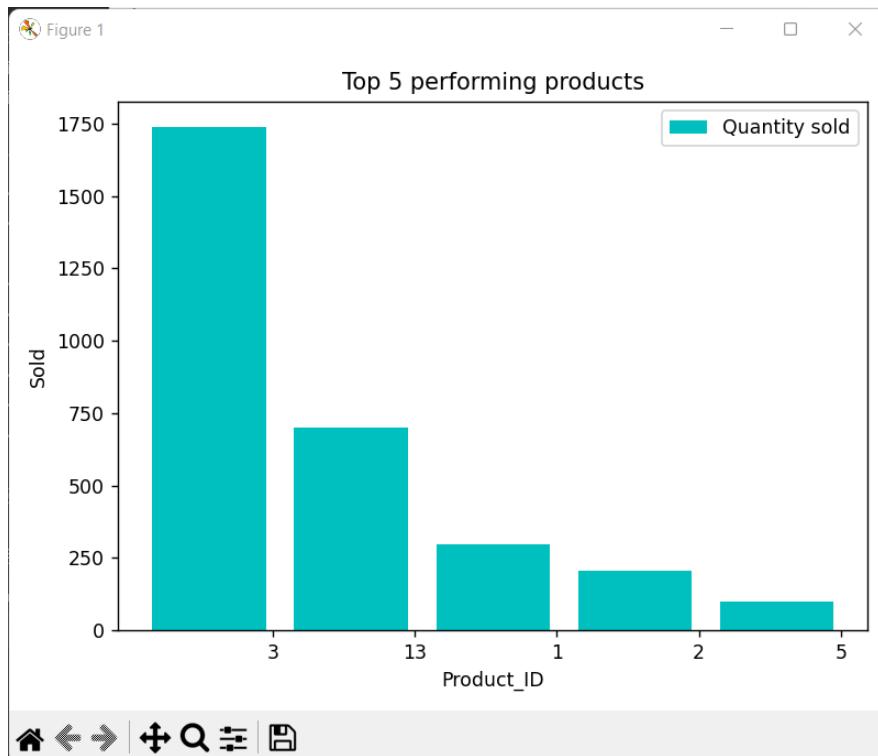
```
# Set the width and gap between bars in the bar chart
width = 0.8
gap = 0.05

# Compute the positions of each bar
positions = range(1, len(show_top_5) + 1)
positions = [pos - width / 2 - gap for pos in positions]

# Create the bar chart
plt.bar(positions, sold, width=width, color='c', label="Quantity sold")
plt.xticks(range(1, len(show_top_5) + 1), product_ID)

plt.xlabel('Product_ID') # x axis title
plt.ylabel('Sold') # y axis title
plt.title('Top 5 performing products') # title
plt.tight_layout()
plt.legend()
plt.show() # display the bar chart
```

When I press the button:



Here, the graph is in descending order – a contrasting result compared to the previous result. However, it makes it easier for my client as the treeview will also be in descending order and so the client can quickly identify the name of the Product for Product ID 3 etc.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>44.</u>	Graph showing top 5 performing products	N/A	N/A	N/A	Graph should show top 5 products	Passed – Graph shows top 5 products based on

	in Analysis window				by quantity sold	quantity sold
--	--------------------	--	--	--	------------------	---------------

Code to show top 5 products in treeview in descending order

```
#####
##### Leaderboard showing top 5 products #####
#####

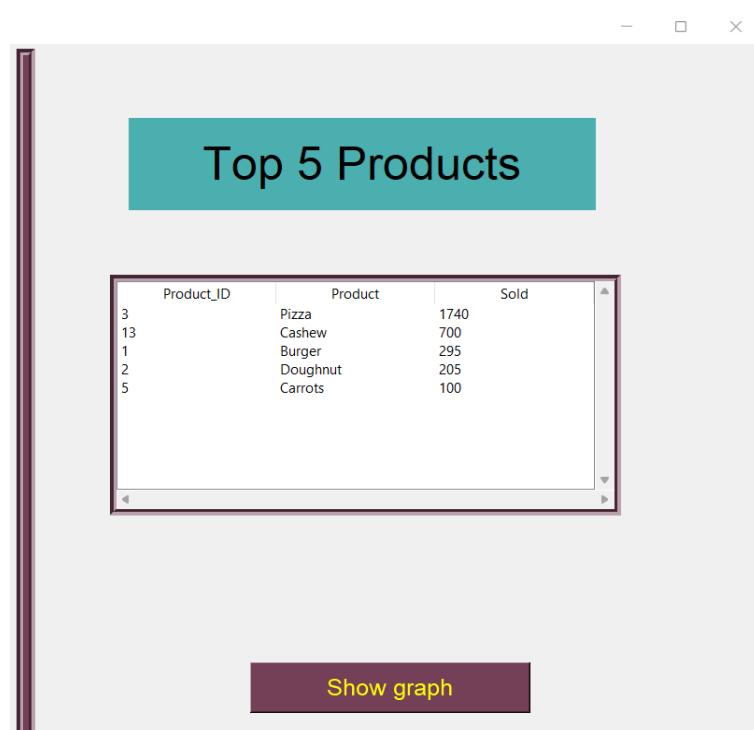
def top_5_products_leaderboard(self):
    connection = sqlite3.connect(r'sms.db') # connect to database
    cursor = connection.cursor()
    # here I have joined the Product and Sales table through their Product field as it is common in both tables
    # This helps to fetch the name of the Product from the Product table by joining the 2 tables together
    top_5 = "SELECT Sales.Product_ID, Product.Product, SUM(Sold) AS total_revenue from Sales" \
        " INNER JOIN Product ON Product.Product_ID = Sales.Product_ID WHERE Date > (SELECT DATE('now', '-7 day'))" \
        " GROUP BY Sales.Product_ID ORDER BY SUM(Sold) DESC LIMIT 5"
    # LIMIT 5 means only 5 products will be shown on the treeview
    rows = cursor.execute(top_5).fetchall()
    connection.commit()
    for r in rows:# iterates through the values stored in rows
        self.best_5_table.insert('', END, values=r)# inserts into best 5 treeview
```

This code here uses the inner join method to link the Product and Sales tables together using the Product ID as the common field. This allows us to fetch the name of the Product instead of the user having to go back to the Product table to search for it.

Here, the SQL query fetches the first 5 products that have sold the most in descending order which is then iterated through using a 'for' loop and inserted into the best 5 treeview.

Testing: Shows top 5 products in descending order in treeview

When I open the sales window:



Here, as you can see the treeview shows the Product ID, Product (name) and the total quantity sold within one week in descending order.

The program works as 1740 > 700 > 295 > 205 > 100 which is in descending order

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>45.</u>	Treeview showing top 5 products in descending order	N/A	N/A	N/A	Treeview should show top 5 products in descending order with the first record showing the highest quantity sold for a product	Passed – Treeview showing top 5 products in descending order

Code to show graph of worst 5 products

This code below uses SQL query to fetch the first 5 products based on total quantity sold within one week in ascending order. These are then added to their corresponding lists using a for loop.

Here row[0] refers to the Product_ID and row[1] refers to the sold quantity.

```
#####
# Graph showing worst 5 products #####
#####

def worst_5_products(self):
    connection = sqlite3.connect(r'sms.db') # connects tp database
    cursor = connection.cursor()
    # fetches the product id and sum of sold quantity within one week in ascending order and retrieves first 5 products
    cursor.execute("SELECT Product_ID, SUM(Sold) AS total_revenue FROM Sales WHERE Date > (SELECT DATE('now', '-7 day')) "
                  "GROUP BY Product_ID ORDER BY SUM(Sold) LIMIT 5")
    show_worst_5 = cursor.fetchall()
    product_id = [] # list
    sold = [] # list
    for row in show_worst_5:# for loop
        product_id.append(row[0])# add to list
        sold.append(row[1])# add to list
    width = 0.8
    gap = 0.05
    # Compute the positions of each bar
    positions = range(1, len(show_worst_5) + 1)
    positions = [pos - width / 2 - gap for pos in positions]

    plt.bar(positions, sold, width=width, color='c', label="Quantity sold")
    plt.xticks(range(1, len(show_worst_5) + 1), product_id)
```

```
# features of the graph
plt.xlabel('Product_ID')
plt.ylabel('Sold')
plt.title('Worst 5 performing products')
plt.tight_layout()
plt.legend()
plt.show()
```

Features of the graph

Testing: Graph to show worst 5 products

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>46.</u>	Graph showing worst 5 performing products in Analysis window	N/A	N/A	N/A	Graph should show worst 5 performing products	Passed – Graph shows worst 5 products in ascending order

Code to show worst 5 products in treeview in ascending order

```
#####
##### Leaderboard showing worst 5 products #####
#####
def display_worst_5_products_leaderboard(self):
    connection = sqlite3.connect('sms.db') # connecting to database
    cursor = connection.cursor()
    ###### Connecting Sales and Product tables to fetch Product field #####
    ### shows products in ascending order based on total quantity sold within one week
    show_worst_5 = "SELECT Sales.Product_ID, Product.Product, SUM(Sold) AS total_revenue from Sales" \
                  " INNER JOIN Product ON Product.Product_ID = Sales.Product_ID WHERE Date > (SELECT DATE('now', '-7 day'))" \
                  " GROUP BY Sales.Product_ID ORDER BY SUM(Sold) LIMIT 5" # limit 5 means only first 5 products are fetched
    rows = cursor.execute(show_worst_5).fetchall()
    connection.commit()
    for r in rows:# iterating through values stored in rows
        self.least_5_table.insert('', END, values=r)# adding it to least_5 treeview
```

This code here again joins the Product and Sales table together and fetches the first 5 products that are ordered in ascending order based on total quantity sold within one week. These are then stored in a variable which is then iterated through and added to the least_5 treeview.

Testing: Treeview showing worst 5 products in ascending order

Product_ID	Product	Sold
4	Cocunut	20
7	Mozzarella	35
2	Doughnut	85
5	Carrots	100
1	Burger	120

This program shows the worst 5 products in ascending order as $20 < 35 < 85 < 100 < 120$

Test Number	What I'm testing	Normal	Boundary	Erroneous	Expected Results	Actual outcome
47.	Treeview showing worst 5 products in ascending order	N/A	N/A	N/A	Treeview should show worst 5 products in ascending order based on quantity sold	Passed – Treeview shows the worst 5 products in ascending order

How the entire Analysis window looks like: (The treeview showing the top 5 products is different from the screenshots shown above as this screenshot was taken on a different day).

The screenshot shows the 'Performance of Products' analysis window. It features two main sections: 'Worst 5 Products' on the left and 'Top 5 Products' on the right. Each section contains a table with columns: Product_ID, Product, and Sold. In the 'Worst 5 Products' section, the data is:

Product_ID	Product	Sold
4	Cocunut	20
7	Mozzarella	35
2	Doughnut	85
5	Carrots	100
1	Burger	120

In the 'Top 5 Products' section, the data is:

Product_ID	Product	Sold
3	Pizza	1200
1	Burger	120
5	Carrots	100
2	Doughnut	85
7	Mozzarella	35

At the bottom of each section is a 'Show graph' button.

Client review

Parth: The program looks really well made and the graphs look really nice and it is easy to find the names of the Product by referring to the treeview.

Comment: The client is happy with my Analysis section and so I can move to the next stage.

Stage 6: Functionality of Dashboard

```
def update_label(self):
    connection = sqlite3.connect(r'sms.db')
    cur = connection.cursor()
    try:
        ##### Reorder point #####
        cur.execute("Select * from Product WHERE Quantity < Reorder_Point")
        reorder_point = cur.fetchall() # fetches all the records where the quantity is less than the reorder point
        self.lbl_reorder.config(text=f'Reorder Point\n[ {str(len(reorder_point))} ]') # displays the number of items that fulfil the condition

        ##### Stock Out #####
        cur.execute("Select * from Product WHERE Quantity = 0") # select all record where quantity is 0
        stock_out = cur.fetchall()
        self.lbl_stockout.config(text=f'Stock Out\n[ {str(len(stock_out))} ]') # gets the number of products that have 0 quantity and adds it to label
    
```

This code here creates a function called “update_label” where it displays the text on 2 buttons – the Reorder Point and the Stock Out.

Here, I have used 1 SQL query to fetch all the records where the Product’s current quantity is less than the Reorder Point by using the Product table which is stored in a variable. I have then used the len() function to get the number of products that match the criteria which are then added to the reorder point label text.

Another SQL query fetches all the records where the Product’s current quantity is 0. I have again used the len() function to get the number of Products with quantity 0. This number is then added to the Button text.

```
#####
Recent Sales #####
cur.execute("SELECT Sum(Value) FROM Sales where Date = (SELECT MAX(Date) FROM Sales)") # gets the total sum of sales from the most recent date
recent_sales = cur.fetchall()

array_recent_sales = np.array(recent_sales) # creates a numpy array from the database query
array_recent_sales_new = str(array_recent_sales) # converting to string

most_recent_sales = array_recent_sales_new[2:-2] # this removes any brackets

self.lbl_sales_recent.config(text=f' Recent Sales (£) \n [ {str(most_recent_sales)} ]') # adda it to the label

time_ = time.strftime("%H:%M:%S") # time
date_ = time.strftime("%d-%m-%Y") # date
self.time.config(text_=f"Welcome to Stock Management System\t\t Date: {str(date_)}\t\t Time: {str(time_)}") # this ensures that the labels change in real time
self.time.after(200, self.update_label)

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)
```

This code here adds up all the value (£) for the most recent date for all the Products using an SQL query that uses the Sum() function. The result of the query is stored in a variable named ‘recent_sales’. The code then converts this result to a NumPy array using the np.array()

function, and then converts that array to a string using the `str()` function. Any brackets are removed from the string.

This is then added to the button to display the total sales for the most recent date as a text.

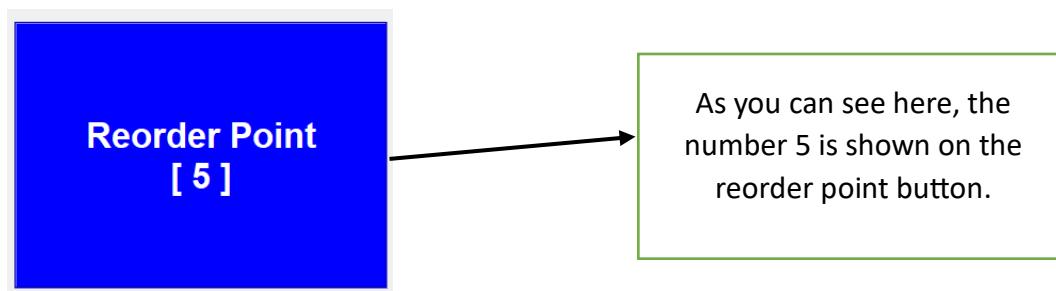
Testing: Shows a number on the Reorder point button to show items less than reorder point

I have taken a screenshot of the product treeview to show the product details (Product name and ID) as well as the reorder point and Stock Quantity.

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	25
2	Junk	Doughnut	100	65
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	0
5	Vitamin_A	Carrots	190	100
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	55
10	Milk	Paneer	30	80
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Product ID	Category	Product	Reorder Point	Stock Quantity
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	0
5	Vitamin_A	Carrots	190	100
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	55
10	Milk	Paneer	30	80
11	protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	300
14	Cake	Carrot	50	60

Here, I have highlighted the Product ID's where the Reorder Point is greater than the Stock Quantity. There are 5 products that are below the reorder point and so the button should show the number 5.



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>

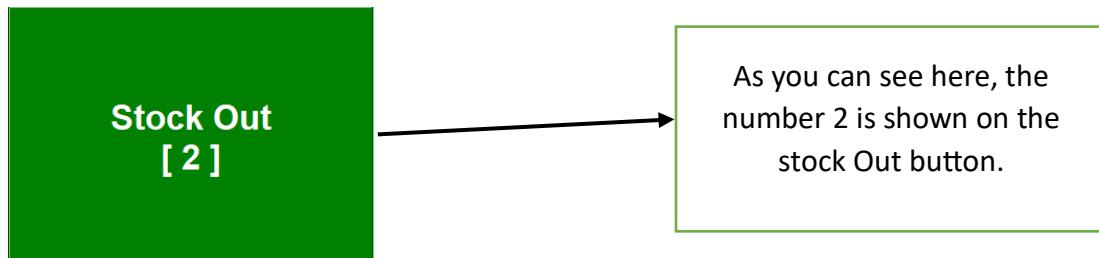
<u>48.</u>	Reorder Point button should show the number of products that are below the reorder point	N/A	N/A	N/A	Number should be shown on top of the reorder point button	Passed – the screenshot above shows the number 5 on the Reorder Point button which is correct since there are 5 products below the reorder point as highlighted in my treeview above
------------	------------------------------------------------------------------------------------------	-----	-----	-----	-----------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Testing: Shows a number on the Stock Out button to show items less than reorder point

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	25
2	Junk	Doughnut	100	65
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	0
5	Vitamin_A	Carrots	190	100
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	55
10	Milk	Paneer	30	80
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Product ID	Category	Product	Reorder Point	Stock Quantity
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	0
5	Vitamin_A	Carrots	190	100
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	55
10	Milk	Paneer	30	80
11	protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	300
14	Cake	Carrot	50	60

Here, I have highlighted the products that are out of stock. In this case, Product ID 3 and 4 are out of stock (0 stock quantity). According to the logic of my program, the Stock-out button should show '2' as the text.



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>49.</u>	Stock-out button should show the number of products where the quantity is 0	N/A	N/A	N/A	Number should be shown on top of the stock-out button	Passed – the screenshot above shows the number 2 on the Stock Out button which is correct since there are 2 products with 0 quantity left as highlighted in my treeview above

Testing: Show total sales in value (£) for most recent date

	A	B	C	D
1	2023-03-15	3	100	500
2	2023-03-15	4	10	50
3	2023-03-15	2	20	80
4	2023-03-15	1	90	450

Here, I have created the sales data in my csv file that will be imported into the system. Here, the 4th column shows the sales in value(£). The total sales should be $500 + 50 + 80 + 450 = 1080$. The label for the most recent sales (£) should show 1080.

When I press import file in the Sales window:



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>50.</u>	Recent sales showing the total amount of sales in value (£) for all the products for the most recent date	N/A	N/A	N/A	Number showing the total amount of value made (£) for all products sold on the most recent date	Passed – the screenshot above shows the total sales made for 15 th March which was £1080.

Reorder Point and Stock Out message alert

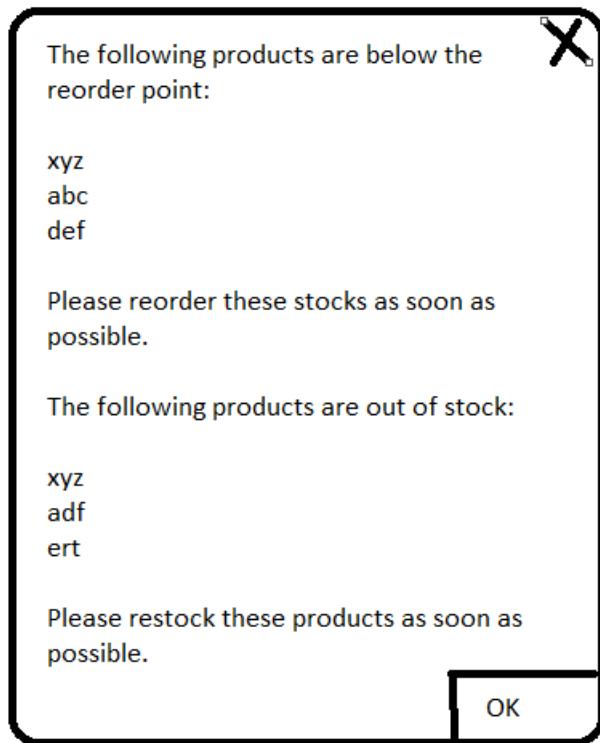
In my design section, I had 2 different designs for my reorder point and stock-out alert. After a quick discussion with my client, we have decided to change the design.

Parth: I would like to change the design for these alerts so that there is one big message box showing the reorder point and stock out alerts.

Me: Ok, this will be done, but the alert message box will be quite big in size.

Parth: No problem, as long as I can see the products clearly that are out of stock or below the reorder point.

I have shown this design below to Parth.



Parth: I really like this design and the text that tells the user to order the stocks as soon as possible.

Me: Ok, should I use this design then?

Parth: Yes, I would like this alert to appear every time the user goes on to the dashboard window.

Me: Yes, I can do that. The alert will only be shown once when the user goes on to the dashboard. To show the alert again, the user would have to close the entire program and run it again.

Parth: How will I know if the user has read the alert?

Me: The message box will not allow the user to go to other windows from the dashboard unless the user has pressed the cross sign or the ok button on the message box.

Parth: Ok, that is great.

Code to show reorder point and stock-out alert

```
def show_status(self): # function to create reorder point and stock out alerts
    connection = sqlite3.connect(r'sms.db') # connecting to database
    cur = connection.cursor()
    try:
        # Get products that are below reorder point
        cur.execute("SELECT * FROM Product WHERE Quantity < Reorder_Point")
        reorder_point_products = cur.fetchall()

        # Get products that are out of stock
        cur.execute("SELECT * FROM Product WHERE Quantity = 0")
        stock_out_products = cur.fetchall()
```

This code here is used to get the products that are below the reorder point and products that are stocked out.

```

# Create message for reorder point products
if len(reorder_point_products) > 0:# if the number of products below the reorder point is greater than 0
    reorder_point_message = "The following products are below the reorder point:\n\n"
    for product in reorder_point_products: # for loop
        reorder_point_message += f"{product[2]}\n"# adding products to message
    reorder_point_message += "\nPlease reorder these products as soon as possible." # add this text to the message
else:
    reorder_point_message = ""

# Create message for stock out products
if len(stock_out_products) > 0:
    stock_out_message = "The following products are out of stock:\n\n"
    for product in stock_out_products:# for loop
        stock_out_message += f"{product[2]}\n"# adding products to message
    stock_out_message += "\nPlease restock these products as soon as possible." # add this text to the message
else:
    stock_out_message = ""

```

This code is split into 2 parts. The top half creates a message for the reorder point products where it iterates through the variable “reorder_point_products” and adds the products from those records to the message. It then adds a final message at the end to conclude the message. If the length of the variable is not greater than 0 then the message is blank.

The second half of the program does the same thing as the top half but when the products are stocked out (0 quantity in the store). It uses SQL query like the first to get the products which are then added to the message.

```

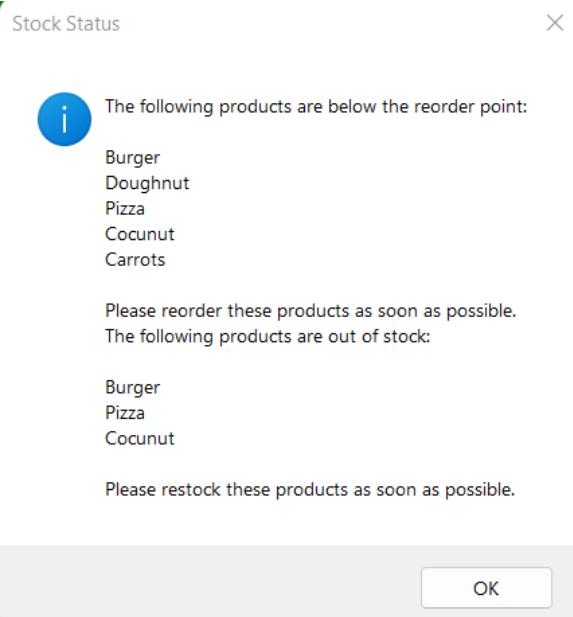
# Show message box if there are any products below reorder point or out of stock
if reorder_point_message != "" or stock_out_message != "":
    messagebox.showinfo("Stock Status", f"{reorder_point_message}\n{stock_out_message}", parent=self.wind) # this shows message box

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

This code above adds the reorder point message and the stockout message to the message box if the messages are not blank.

Testing: Messagebox showing the items out of stock or need reordering



When I go to the dashboard window, I get a messagebox as shown on the left. As you can see the name of the Products are mentioned for the products that need reordering or are out of stock.

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	0
2	Junk	Doughnut	100	45
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	0
5	Vitamin_A	Carrots	190	100
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	55
10	Milk	Paneer	30	80
11	protein	Tofu	30	50
12	Protein	Almonds	50	60

Product ID	Category	Product	Reorder Point	Stock Quantity
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	0
5	Vitamin_A	Carrots	190	100
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	55
10	Milk	Paneer	30	80
11	protein	Tofu	30	50
12	Protein	Almonds	50	60
13	Protein	Cashew	30	300
14	Cake	Carrot	50	60

As you can see in the treeview above, the records with a pink mark show that the product has 0 quantity. In this case it is Burger, Pizza and Cocunut. This matches the products shown on the bottom half of the messagebox for the stocked out items.

The records with a red dot show that the product current quantity is below the reorder point. In this treeview, it is Burger, Doughnut, Pizza, Cocunut and Carrots. This matches the products shown on the top half of the messagebox for the products that need reordering.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>51.</u>	Message box being displayed to show the items that are out of stock or need reordering	N/A	N/A	N/A	Message box should show correctly the names of products that are below the reorder point or have no quantity in the shop. This should be shown once.	Passed – the screenshot above shows correctly the products that are out of stock and are below the reorder point in the Tkinter messagebox.

Client Review

Parth: The tests that you have done clearly show that the dashboard label and buttons are working properly. With regards to the

I have shown this stage to my client, Parth who is happy with the dashboards entire functionality and so I can move on to the next stage.

Stage 7: Reorder Point window

I have discussed the functionality of this window with Parth to ensure the program works as he wants.

Me: In this window, I will create a treeview that will show the names of the Products and the connected suppliers so that the user can click on one of the records which will then populate the supplier's Email address on to the entry forms. To send an email to the suppliers, I will use SMTPlib which will require the user putting their email address and their password. However, I will ensure that the program automatically uses their email address and their password without any user input.

From the login design feedback, Parth wanted the program to work so that the email address could be manually hard-coded into the program but the password should not be seen in the code. I will confirm with Parth about where he would like the password and email address to be stored.

Me: Where would you like the password and email address to be stored?

Parth: Can the password and email address be stored in an external CSV file?

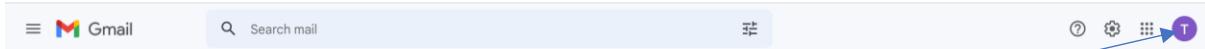
Me: Yes, I can write the password and email address on a csv file which will be separated by commas and can be accessed using python.

Parth: Yes that is great. Will I have to use an actual password or can something else be used?

Me: We can go on the gmail account and turn on 2-factor authentication. This will create a random password that can be accessed using the Python application. That way we don't have to put in the actual password onto the CSV file.

Getting an App password for Python

1. To get the App password, the user will need to sign into their Gmail account.



Here as you can see in the top right corner, there is a google account icon.

2. Once the user clicks on this icon, they should click on "Manage your Google Account".

The screenshot shows the Google Account dashboard. On the left, there's a sidebar with links: Home, Personal info, Data and privacy, Security (which has a blue arrow pointing to it), People and sharing, Payments and subscriptions, and About. The main area is titled "Welcome, Test Purposes" and says "Manage your info, privacy and security to make Google work better for you. [Find out more](#)". It features two cards: "Privacy & personalisation" (with a gear icon) and "You have security recommendations" (with a shield icon). Below these is a section titled "Privacy suggestions available" with a "Review suggestion (1)" link. At the bottom of the main area, there are links for Privacy, Terms, Help, and About.

This screen will be shown.

3. Click on security (left hand side).
4. Enable 2-step Verification

How you sign in to Google

Make sure that you can always access your Google Account by keeping this information up to date

2-Step Verification

On since 11 Feb



Here, I have created one from before so I will be using that as my App password.

5. Go on App passwords.

App passwords

App passwords aren't recommended and are unnecessary in most cases. To help keep your account secure, use 'Sign in with Google' to connect apps to your Google Account.

The screenshot shows the "App passwords" section. It has a header "App passwords" and below it, it says "1 password". There is a small orange arrow pointing to the right at the end of the row.

6. When you click on the next arrow you should see something like this:

[← App passwords](#)

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

Your app passwords

Name	Created	Last used
Python	11 Feb	14 Mar

Select the app and device for which you want to generate the app password.

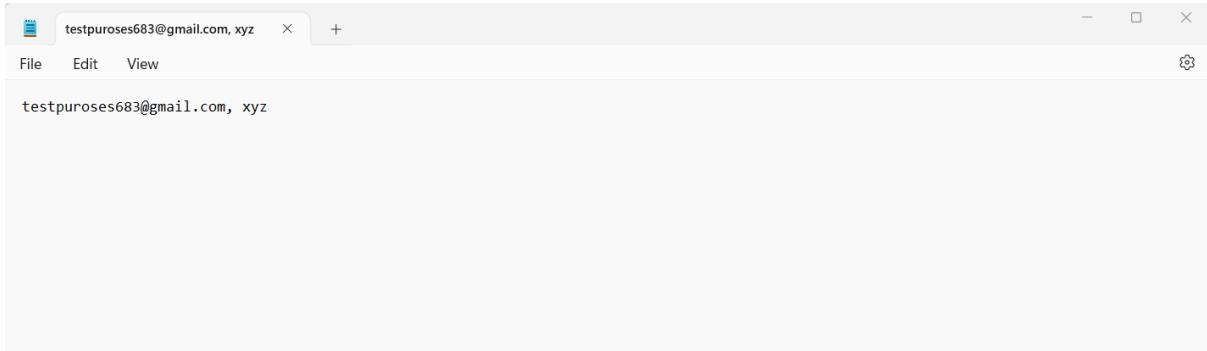
Select app: Select device: GENERATE

When you click on select app, put Other and type 'Python'.

7. When you click on device, put the OS of the device you're using such as Mac, Windows etc.
8. Then click generate.
9. A password will be generated.

The generated password will be stored in the CSV file along with the Email address of the sender (user).

The CSV file will look something like this:



Here, the email address and the password have been separated by commas.

I have shown this to Parth who now understands how the program will function.

Developing the code

```
from tkinter import ttk, messagebox # used to give message
import sqlite3 # used for database
import smtplib # used to send emails
from tkinter import *
import csv # used for csv file
```

Here, I have imported all the libraries I will need. Here, I am importing smtplib as I will be using this so that my client can send emails to their suppliers. I am also importing csv as this

will be used to read the username and password that is stored in it. The username and password is essential to send emails.

```
#####
# Reorder Point treeview #####
scrollx = Scrollbar(Reorder_Point_Table_Frame, orient=VERTICAL)
scrolly = Scrollbar(Reorder_Point_Table_Frame, orient=HORIZONTAL)
self.Reorder_point_table = ttk.Treeview(Reorder_Point_Table_Frame,
                                         columns=("Product_ID", "Product", "Quantity", "Reorder_Point", "Supp ID", "Supp Name", "Supp Email"),
                                         yscrollcommand=scrollx.set, xscrollcommand=scrollx.set)

scrollx.pack(side=BOTTOM, fill=X)
scrolly.pack(side=RIGHT, fill=Y)
scrollx.config(command=self.Reorder_point_table.xview)
scrolly.config(command=self.Reorder_point_table.yview)

self.Reorder_point_table.heading("Product_ID", text="Product ID")
self.Reorder_point_table.heading("Product", text="Product")
self.Reorder_point_table.heading("Quantity", text="Stock Quantity")
self.Reorder_point_table.heading("Reorder_Point", text="Reorder Point")
self.Reorder_point_table.heading("Supp ID", text="Supplier ID")
self.Reorder_point_table.heading("Supp Name", text="Supplier Name")
self.Reorder_point_table.heading("Supp Email", text="Supplier Email")
self.Reorder_point_table["show"] = "headings"

self.Reorder_point_table.column("Product_ID", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Product", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Quantity", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Reorder_Point", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Supp ID", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Supp Name", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Supp Email", width=330, stretch=NO) # fixed width
```

Here, I have defined the treeview that I am going to use so that my client can see the products that are below the reorder point. Here, I have added fields from the Product table and the Supplier table.

```
#####
# Email Panel #####
# label to show where to enter the recipient's email address
self.address_field = Label(self.wind, text="Recipient Address :", font=("", "15"), bg="#4CAF50", fg="black")
self.address_field.place(x=15, y=370, height=50)

# label to show where to enter the message
self.email_body_field = Label(self.wind, text="Message :", font=("", "15"), bg="#4CAF50", fg="black")
self.email_body_field.place(x=15, y=560, height=50)

self.address = StringVar() # variable to store recipient's address
self.email_body = StringVar() # variable to store what user types as message
```

Here, I have created 2 labels to show what values go into the corresponding entry boxes and have defined two variables – one that stores the recipient's address and another that stores the user's message.

```
# this is where the recipient's (suppliers) email address is put
self.address_form = Entry(self.wind, textvariable=self.address, width="80", font=("", "15"), bg="lightyellow",
                           fg="black").place(x=15, y=450, height=60)

# this is where the user enters their message
self.email_body_form = Entry(self.wind, textvariable=self.email_body, width="80", font=("", "15"),
                           bg="lightyellow", fg="black").place(x=15, y=650, height=60)
```

These are the entry boxes where the user will give the Supplier's address and the message that they want to give. The variables defined will store whatever the user inputs into the entry boxes.

Function to show products below reorder point and linked suppliers in treeview

```
#####
# Reorder point on treeview #####
def update_reorder_point(self):
    connection = sqlite3.connect(r'sms.db') # connection to the database
    cur = connection.cursor()
    # this sql join the sourcing table with the product table and the supplier table so that the supplier and product details can be fetched
    # gets and joins records from tables if the reorder point is greater than the current quantity of a certain product
    reorder_point = "select Product.Product_ID, Product.Product, Product.Quantity, Product.Reorder_Point, " \
                    "Supplier.ID, Supplier.Name, Supplier.Email FROM supplier, Product, Sourcing WHERE Sourcing.Product_ID = Product.Product_ID" \
                    " AND Sourcing.Supplier_ID = supplier.ID AND Product.Quantity < Product.Reorder_Point ORDER BY Product.Quantity"
    rows = cur.execute(reorder_point).fetchall() # fetches all the records that fulfils the condition
    connection.commit() # commits the changes
    for r in rows: # iterates through the values stored in rows
        self.Reorder_point_table.insert('', END, values=r) # adds it to the treeview
```

This function above uses SQL query to fetch data from multiple tables which are Product, Supplier and Sourcing. Here, the sourcing table is used to connect the Product and Supplier tables together using the foreign keys of Product ID and Supplier ID respectively. Once the SQL query joins the tables together, it filters the results to show only products that have a quantity below their reorder point. This is then ordered by the quantity of the product in ascending order. All the records that fit this criteria are stored in the variable 'rows' which is then iterated through and added to the reorder point treeview.

```
# when the user clicks on one of the records in the treeview, the email address is populated onto the email address entry field
def get_Reorder_Point(self, sourcing): # this gets the row of data
    self.focus = self.Reorder_point_table.focus()
    self.data = self.Reorder_point_table.item((self.focus))
    row = self.data['values']
    self.address.set(row[6])
```

This function here allows the user to click on a record from the reorder point treeview which will then display the supplier's email address of the selected record into the address entry form. This ensures that the user does not make any spelling mistakes and so an email can be sent to an existing email address. Here, I have not added any validation as I have added validation in my supplier window when the user inputs or updated an existing supplier's email address.

```
# this function gets the username and password of the user from the csv file
def fetch_details(self):
    with open('login.csv', 'r') as file: # opens the csv file so that it can be read
        reader = csv.reader(file) # csv reader object
        for row in reader: # iterates through the rows in the csv file
            username = row[0] # first data is username which is the sender's email address
            password = row[1] # second data is an app password
    return username, password # returns the username and password as a tuple
```

This function above fetches the details of the user's username (their email address is their username) and their password. This password is the App password that I have previously mentioned before. The code here opens the csv file in read mode (so that it can't be edited) and creates a csv reader object. The rows of the csv file are then iterated and the username and password are set accordingly. Here, there is only one record of the username and password. The username and password are returned as a tuple so that they can be used in other functions of the program.

Function to send the email

```
##### function to send email to supplier #####
def send_email(self):
    try:
        username, password = self.fetch_details() # calls the fetch_details() function to get username and password from the csv file
        to = self.address.get() # variable defined "to" which stores the recipient's address
        email_body = self.email_body.get() # content of the message
        if to == "" or email_body == "": # if any of the fields are empty
            messagebox.showerror("Incomplete input", "Please fill in all the entry forms", parent=_self.wind)
```

This first half of the function calls the `fetch_details()` function and gets the username and password from the csv file. I have created a variable called “`to`” which stores the supplier’s address (recipient).

I have also created a variable called “`email_body`” that stores the message that the user enters onto the entry field. The reason I have created these variables instead of using the previous variables is because these variables are much easier to comprehend, and the user can instantly know what the variables (“`to`” and “`email_body`”) store. Here, I have added a validation to ensure that the user enters input for all the given entry boxes in the window. If they do not fill in all the details, a message occurs asking the user to fill in all the entry forms.

```
else:
    # create an instance of smtplib.SMTP
    # port number = 587
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls() # tells server we want to use TLS encryption - this ensures no third party can read or modify sent data
    server.login(username, password) # username and password is retrieved from csv file
    server.sendmail(username, to, email_body) # takes in sender's email address, recipients email address and content of the message as parameters
    messagebox.showinfo("Sent", "Email has been sent", parent=_self.wind) # this shows a message if the email has been sent successfully

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=_self.wind) # this gives us any errors that may occur when running the program
```

If the user fills in all the forms, then an email should be sent. Here, I have created an instance of the `SMTP` class which connects to the `SMTP` server on port 587. Port 587 is quite a common port for sending emails.

`Server.starttls()` allows a TLS (Transport Layer security) encrypted connection with the server so that the email message is transmitted safely without any third party users from gaining access to it.

Here, the `server.login()` method takes in 2 parameters – the username and the password which are retrieved from the csv file as shown in the top half of this function (look previous code).

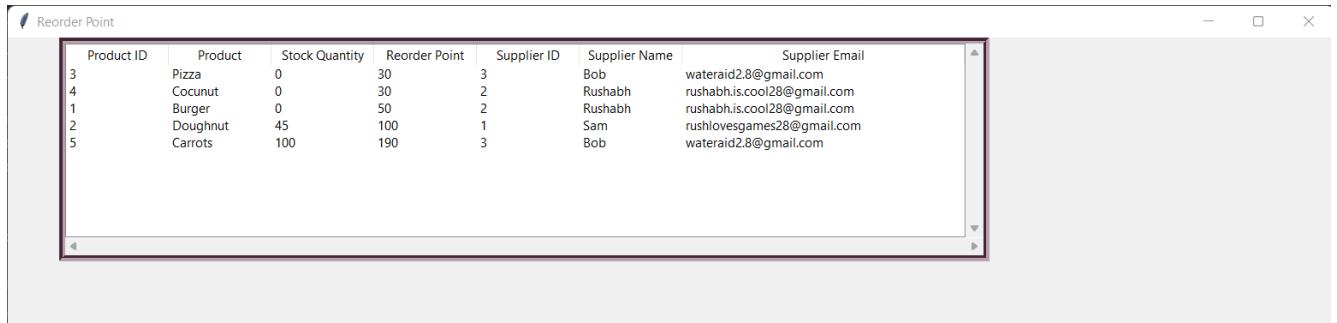
The `server.sendmail()` method takes in 3 argument – (sender’s email address, the recipient’s email address and the content of the message). This allows the message to be sent to the intended supplier.

To ensure that the email has been sent correctly, I have added a messagebox that shows “Email has been sent” when the program runs successfully.

This code below creates a button to send the message. It is linked with the function shown previously that sends emails to the suppliers.

```
# send button
self.email_btn = Button(self.wind, text = "Send message", command = self.send_email, bg = "light blue", width = "20", height="6").place(x = 1330, y = 478)
```

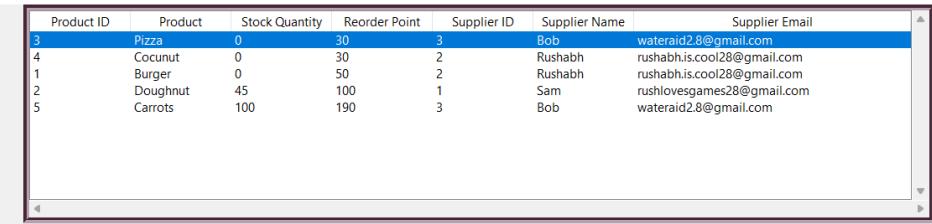
Testing: Treeview showing products and suppliers when product quantity < reorder point



As you can see here, the treeview successfully shows the products where the stock quantity is less than the reorder point. Information about the supplier such as their name and email address are also included as part of the record so the user can send an email to the supplier.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>52.</u>	Treeview showing products and linked suppliers when product quantity is below reorder point	N/A	N/A	N/A	Treeview containing products and linked suppliers that are below reorder point	Passed – screenshot above shows the products that need reordering and their linked suppliers along with the supplier's email address

Testing: Email being sent to suppliers

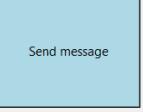


Recipient Address :

wateraid2.8@gmail.com

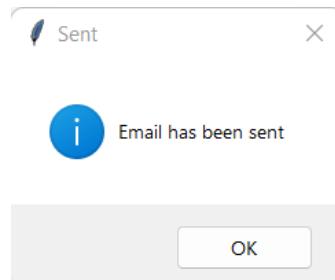
Message :

Hi Bob. Could you please bring in more pizzas to the shop. We need at least 30 pizzas. Thanks



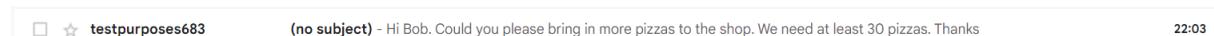
When I select the first record, the email address gets populated onto the entry field so that the user doesn't have to manually type in the supplier's email address.

When I click send message:

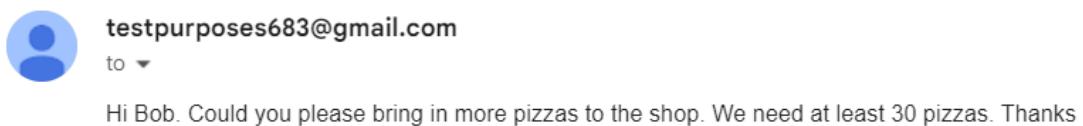


This message box appears roughly 1-2 seconds later.

I will check the email to see if the message appears.



(no subject) 



testpurposes683@gmail.com
to ▾
Hi Bob. Could you please bring in more pizzas to the shop. We need at least 30 pizzas. Thanks

Here, the email has been sent successfully.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>53.</u>	User should be able to send email to the suppliers (same test for stock_out window as well)	N/A	N/A	N/A	Suppliers' should get an email sent by the user (client who owns the store)	Passed – Email gets successfully sent to supplier within a short frame time

Though the program passes, the screenshot shows that there is no subject. I have discussed this with Parth who thinks that it is better if the subject is not empty. That way, the supplier will not get confused about what the email is about and who it is by. Parth doesn't mind the subject being hard coded into the program.

Me: What do you want the subject of the email to be?

Parth: The subject can be called “One-Stop needs more products to be ordered”.

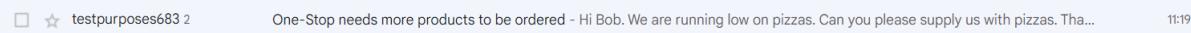
```
##### function to send email to supplier #####
def send_email(self):
    try:
        username, password = self.fetch_details() # calls the fetch_details() function to get username and password from the csv file
        to = self.address.get() # variable defined "to" which stores the recipient's address
        email_body = self.email_body.get() # content of the message
        if to == "" or email_body == "": # if any of the fields are empty
            messagebox.showerror("Incomplete input", "Please fill in all the entry forms", parent=_self.wind)
        else:
            subject = "One-Stop needs more products to be ordered" # subject message
            email_message = f'Subject: {subject}\n\n{email_body}' # message that stores subject and email body content
            # create an instance of smtplib.SMTP
            # port number = 587
            server = smtplib.SMTP('smtp.gmail.com', 587)
            server.starttls() # tells server we want to use TLS encryption - this ensures no third party can read or modify sent data
            server.login(username, password) # username and password is retrieved from csv file
            server.sendmail(username, to, email_message) # takes in sender's email address, recipients email address and content of the message as parameters
            messagebox.showinfo("Sent", "Email has been sent", parent=_self.wind) # this shows a message if the email has been sent successfully

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # this gives us any errors that may occur when running the program
```

Here, I have amended the code by adding a variable called ‘subject’ that stores the subject of the email. I have also created a variable called “email_message” that takes in the data stored in the ‘subject variable’ and in ‘email_body’ variable by using string formatting.

In the sendmail() method, I have replaced the variable ‘email_body’ with ‘email_message’ so that it includes the subject of the email as well.

When I click on send button:



Here, you can see the subject “One-Stop needs more products to be ordered.”

One-Stop needs more products to be ordered Inbox ×

 **testpurposes683@gmail.com**
"Hi Bob, we need at least 30 pizzas to be reordered." Thanks

 **testpurposes683@gmail.com**
to ▾
Hi Bob. We are running low on pizzas. Can you please supply us with pizzas. Thanks

Subject

Client review

I have shown the reorder point window to my client who approves the development of this program and how the email will look like to the suppliers.

Stage 8: Stock-out window

The code for the Stock-out window is very similar to the reorder point design. There are only 2 major differences:

1. The SQL query will give the products and its linked suppliers when product quantity in the store is 0.
2. The subject of the email will be different.

Other than that, the functionality is identical, so I won't be showing most of the code.

Function to show stocked-out products and linked suppliers on Treeview

```
def update_Stock_Out(self):
    connection = sqlite3.connect(r'sms.db') # connecting to database
    cur = connection.cursor()
    # sql query to get the products and supplier details using the source table where there is 0 quantity left in the store
    stocked_out = "select Product.Product_ID, Product.Product, Product.Quantity, Product.Reorder_Point, "
    "Supplier.ID, Supplier.Name, Supplier.Email FROM supplier, Product, Sourcing "
    "WHERE Sourcing.Product_ID = Product.Product_ID AND Sourcing.Supplier_ID = supplier.ID "
    "AND Product.Quantity = 0"
    rows = cur.execute(stocked_out).fetchall() # gets all the records that fulfil the condition
    connection.commit()
    for r in rows: # iterates through the values stored in rows
        self.Stock_Out_table.insert('', END, values=r) # adds to stock out treeview
```

This code here uses SQL query to get the Product and Supplier tables using the Sourcing table along with the Product and Supplier table. Here, the records are retrieved if the product quantity in the store is 0. Once again, all these records that match this criteria are retrieved, and iterated so that they can be placed one by one into the Stock-out treeview.

Function to send email

Based on previous testing, I have ensured that there is no missing subjects.

Here the subject is “One-Stop products are out of stock”. I have shown this to my client who is fine with the subject.

```
##### function to send email to supplier #####
def send(self):
    try:
        username, password = self.fetch_details() # calls the fetch_details() function to get username and password from the csv file
        to = self.address.get() # variable defined "to" which stores the recipient's address
        email_body = self.email_body.get() # content of the message
        if to == "" or email_body == "": # if the entry fields are blank
            messagebox.showerror("Incomplete input", "Please fill in all the entry forms", parent=self.wind)
        else:
            subject = "One-Stop products are out of stock" # subject message
            email_message = f'Subject: {subject}\n\n{email_body}' # message that stores subject and email body content
            # create an instance of smtplib.SMTP
            # port number = 587
            server = smtplib.SMTP('smtp.gmail.com', 587)
            server.starttls() # tells server we want to use TLS encryption - this ensures no third party can read or modify sent data
            server.login(username, password) # username and password is retrieved from csv file
            server.sendmail(username, to, email_message) # takes in sender's email address, recipients email address and content of the message as parameters
            messagebox.showinfo("Sent", "Email has been sent", parent=self.wind) # this shows a message if the email has been sent successfully

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}",
                            parent=self.wind) # this gives us any errors that may occur when coding
```

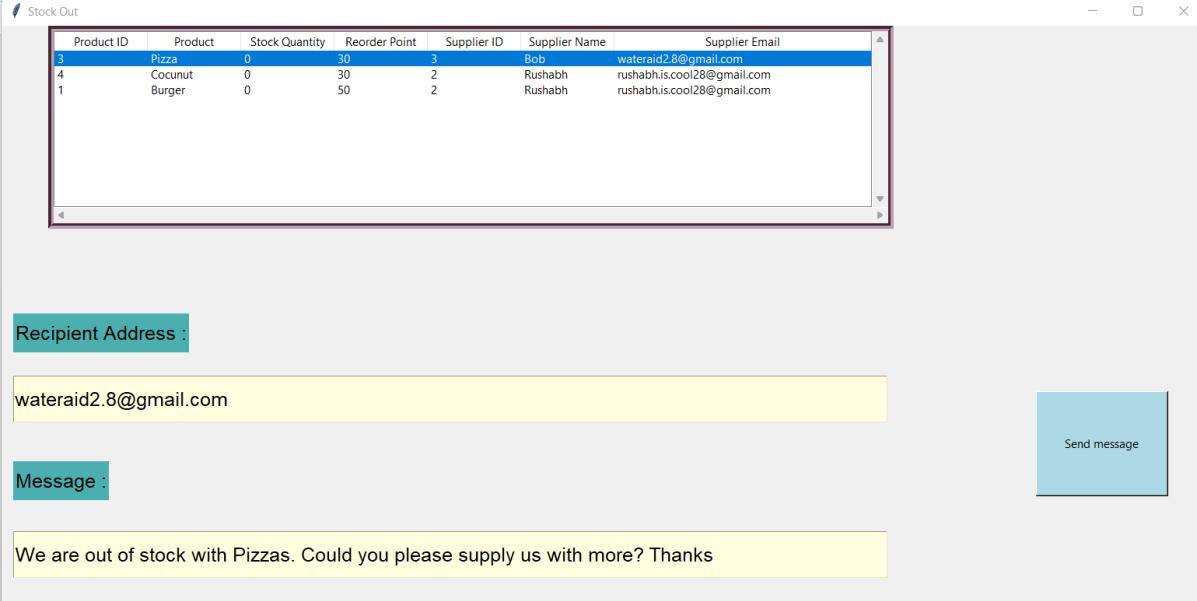
Testing: Treeview showing products and linked suppliers with 0 stock Quantity

Product ID	Product	Stock Quantity	Reorder Point	Supplier ID	Supplier Name	Supplier Email
3	Pizza	0	30	3	Bob	wateraid2.8@gmail.com
4	Cocunut	0	30	2	Rushabh	rushabh.is.cool28@gmail.com
1	Burger	0	50	2	Rushabh	rushabh.is.cool28@gmail.com

In this treeview, all the products have a stock quantity of 0.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>54.</u>	Treeview should show products and linked suppliers when product quantity is 0	N/A	N/A	N/A	Treeview containing products and linked suppliers where quantity is 0	Passed – Treeview shows products and its suppliers when there is 0 quantity.

Testing: Email should be sent to supplier



Recipient Address : wateraid2.8@gmail.com

Message :

We are out of stock with Pizzas. Could you please supply us with more? Thanks

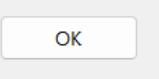
Send message

Just like in my reorder point testing, I have completed all the inputs and have pressed the send message button.

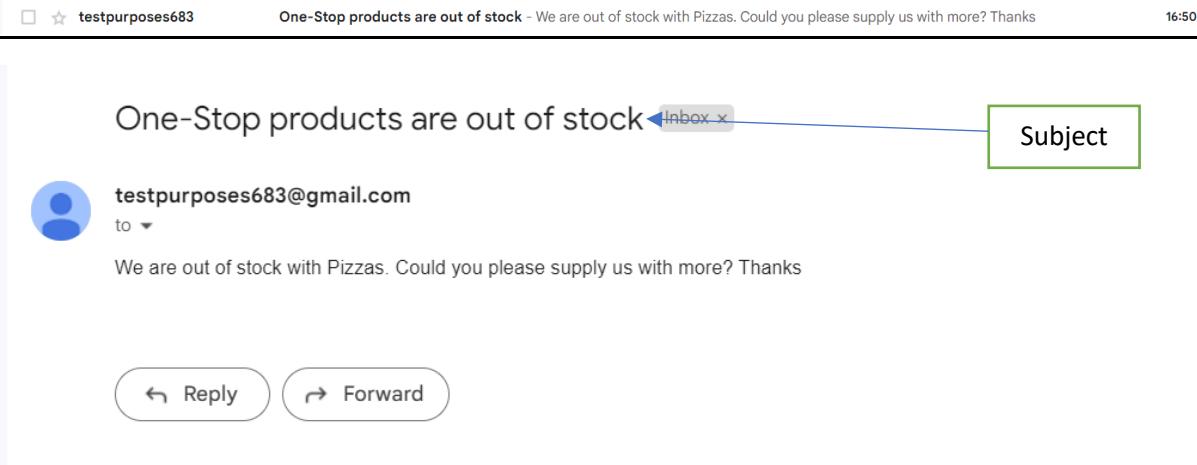
 Sent 



A message appears telling us that the email has been sent.

 OK

This is how the supplier will see the message.



One-Stop products are out of stock 

 testpurposes683@gmail.com
to ▾

We are out of stock with Pizzas. Could you please supply us with more? Thanks

Here, the message is sent successfully to the supplier's email address and the subject is displayed clearly.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>53.</u>	User should be able to send email to the suppliers (same test for stock_out window as well)	N/A	N/A	N/A	Suppliers' should get an email sent by the user (client who owns the store)	Passed – Email gets sent successfully to the supplier. I have also added the subject of the email to make it clear to the suppliers about what the email is about.

I have shown this to my client who is happy for me to proceed to the next stage.

Stage 9: Login system – 2 factor authentication

Login table

```
cur.execute("CREATE TABLE IF NOT EXISTS login(ID integer PRIMARY KEY AUTOINCREMENT, Username text, Password text, Email text)")
con.commit()
```

Here, I have created the login table where the ID, username, Password and Email fields have been created. This will be used for storing the login details of the user.

ID	Username	Password	Email
Filter	Filter	Filter	Filter
1	Trial_test	Testing_login	testpurposes683@gmail.com

This is how it looks like in the login table.

Importing modules

```
from tkinter import *
from tkinter import messagebox # used for showing messages
import sqlite3 # used for databases
import os # this controls the window the program goes into next
import smtplib # used for sending emails
import csv # used to send details to send emails
import random # will use random function to generate opt code
```

Here, I am importing the random function which will be used to generate a random number for the OTP code.

```
class login:
    def __init__(self, wind):
        self.wind = wind
        self.wind.title("Login system")
        self.wind.geometry("1900x990+0+0")

        self.otp = '' # this will be the generated OTP code once the user enters the username and password correctly

        login_section = Frame(self.wind, bd=2, relief=RIDGE)
        login_section.place(x=600, y=90, width=750, height=860)

        login_label = Label(login_section, text="Login to access dashboard", font=("Elephant", 30, "bold")).place(x=0, y=30, relwidth=1)

        username_lbl = Label(login_section, text="Username:", font=("DaunPenh", 15, "italic")).place(x=50, y=130)

        self.username = StringVar() # stores the username
        self.password = StringVar() # stores the password
```

This code here defines 2 variables – the username and the password. This is what the user will enter into the login window as the first verification to access the Stock Management software windows.

```
##### username entry box #####
username_entry = Entry(login_section, textvariable= self.username, font=("times new roman", 15), bg="yellow").place(x=50, y=170)

password_lbl = Label(login_section, text="Password:", font=("DaunPenh", 15, "italic")).place(x=50, y=380)

#####
password_entry = Entry(login_section, textvariable= self.password, font=("times new roman", 15), bg="yellow").place(x=50, y=420)
```

Above, I have created 2 entry boxes to enter the username and password.

Function to fetch details

```
def fetch_details(self):
    with open('login.csv', 'r') as file: # opens csv file
        reader = csv.reader(file) # creates a reader object
        for row in reader: # iterates through the csv file
            email = row[0] # sender's email
            password = row[1] # sender's password
            to_email = row[0] # recipient's email
    return email, password, to_email # returns them
```

This code gets the email, password and the email to send it to by opening the csv file and reading through it. Here, the recipient and the sender email are the same.

```
gmail_user, gmail_password, to_email = self.fetch_details() # calls the function to get sender email, password and who to send it to
```

This code above creates variables to store the sender' email, password and the recipient's email.

Function to send email

```
def send_email(self,to_email, otp, gmail_user, gmail_password):
    sent_from = gmail_user
    to = [to_email]
    self.otp = random.randint(100000, 999999) # generates a random six digit number
```

In this part of the code, I have created variables (“sent_from” and “to”) to define the sender and the recipient’s email address. I have also used the “self.otp” variable to store a random number for my OTP code between 100000 and 999999.

```
subject = "Your OTP code"
body = "Your otp code is:" + str(self.otp)
email_message = f'Subject: {subject}\n\n{body}'
```

This code gets the subject and the body which is the main content of the email. It is then combined into a new variable called ‘email_message’. The body variable shows the OTP code in a string format.

```
try:
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(gmail_user, gmail_password) # sender's details
    server.sendmail(sent_from, to, email_message) # sends mail - to the sender
    server.close()
    print('Email sent!')
except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)
```

This code here sends the email to the recipient. This program for sending emails is similar to the functions used in the reorder point and stock out windows with the exception that the OTP code is added in the content of the email and that there is no user input.

```
C:\Users\rusha\PycharmProjects\pythonProject5\venv\Scripts\python.exe C:/Users/rusha/PycharmProjects/pythonProject5/login.py
File "C:/Users/rusha/PycharmProjects/pythonProject5/login.py", line 181
    email_message = f'Subject: {subject}\n\n{body}'
                                         ^
SyntaxError: EOL while scanning string literal
```

Here, I get a syntax error meaning that I am missing some punctuation.

```
email_message = f'Subject: {subject}\n\n{body}'
```

I have fixed the error by adding an apostrophe at the end of the line to define the string.

When I run the program:

I no longer get the error and it shows the login window.

The diagram illustrates a login interface. It features two input fields: 'Username' and 'Password', each preceded by a yellow placeholder bar. Below these is a blue 'Log in' button. At the bottom left is a green link 'Forgot Password?' and at the bottom right is an orange link 'Click Here!'. Blue arrows point from the placeholder bars to their respective field labels.

Here, I cannot test the email yet, as I need to add validation for the user login.

Function to validate login

This function is used if the user knows their username and password. The function will check to see if it is correct.

```
def login_validation(self):
    connection = sqlite3.connect(r'sms.db') # connecting to database
    cur = connection.cursor()
    try:
        if self.username.get() == "" or self.password.get() == "": # if the fields are left blank
            messagebox.showerror("Incomplete", "Please fill in all the details", parent=self.wind) # shows a message error
            return
    
```

This code checks if the username or the password is empty. If so an error message appears and the program is returned so that other parts of the function cannot be run.

```
else:
    # sql to check if the user input exists in the login table
    cur.execute("select * from login where Username=? AND Password=?",(self.username.get(), self.password.get()))
    login_details = cur.fetchone()
    if login_details == None: # if it doesn't exist in login table
        messagebox.showerror("Error", "Password or Username is wrong") # error message occurs
        return
    
```

This part of the code is run if the user has put input into both the username and the password entry boxes. It is then checked if it matches the one stored in the login table using a SQL query where the record is searched using the username and the password entered by the user. If there are no records found, this means that the user's input is invalid as the SQL query cannot retrieve the data. Here, I have used an AND condition as part of my WHERE condition to ensure that both the inputs entered by the user match the login table.

```
##### if the username and password is correct #####
messagebox.showinfo("Hi", "Sending OTP on email") # sends otp
self.two_factor = Toplevel(self.wind) # creates a new small window on top
self.two_factor.title("2 factor authentication")
self.two_factor.geometry("500x500+500+110")
self.two_factor.focus_force() # tells the program that this window is active and input should be taken from this window
self.two_factor_code = IntVar() # defining the variable. This will store the OTP code that the user enters once their username and password is correct
    
```

However, if the password and username are correct, then a message is shown saying that it is sending an OTP code.

A new window called “self.two_factor” is opened and overlaps with the current window. The features of the new window are set and ensured that the program knows that this window is active by using the `focus_force()` method.

I have also created a variable to store the user’s input of the OTP code.

```
#####
# Label to show what to enter in the entry form #####
OTP_lbl = Label(self.two_factor, text=" Enter OTP Code:", font=('goudy old style', 15, 'bold'), bg = "#3f51b5", fg = "white").pack(side=TOP, fill=X)

#####
# Entry form to enter otp code #####
enter_otp = Entry(self.two_factor, textvariable=self.two_factor_code, font=('goudy old style', 15, 'bold')).place(x=120, y=150)

#####
# This is a submit otp button #####
self.otp_submit = Button(self.two_factor, text="Submit OTP", command=self.two_factor_authentification, font=('goudy old style' , 15, 'bold')).place(x=160 , y=230, width=150, height=50)

gmail_user, gmail_password, to_email = self.fetch_details() # calls the function to get sender email, password and who to send it to
self.send_email(to_email, self.otp, gmail_user, gmail_password) # calls the send_email function
```

This code above creates a label, entry and a button to submit the OTP code. Here, these widgets are created on the new window (“self.two_factor”).

The submit button is connected a function which I will explain later on.

Here, “`gmail_user`”, “`gmail_password`”, “`to_email`” are assigned as the email, password, `to_email` from the `fetch_details()` functions.

I have also called the `send_email()` function that takes in 4 arguments which are the recipient’s email address, the OTP code, the sender’s email address and their password.

The variable “`self.two_factor_code` stores the input that the user puts for the OTP code in the entry box.

Function for 2 factor authentication

This function will be called once the user has entered their username and password correctly.

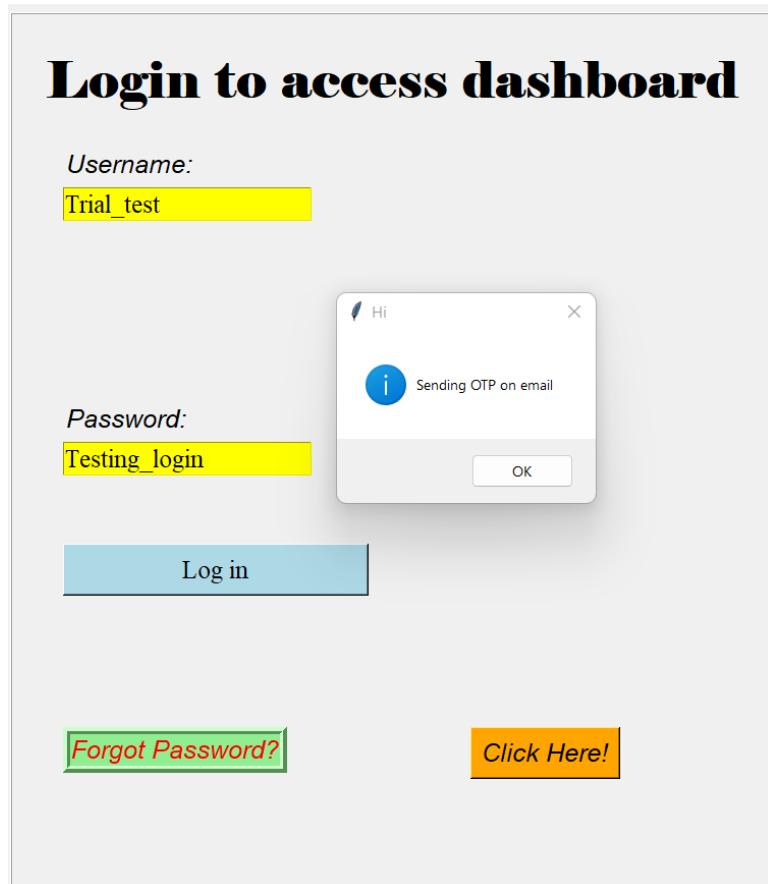
```
#####
## calls function once the username and password is correct #####
def two_factor_authentification(self):
    if int(self.otp) == int(self.two_factor_code.get()): # if the generated otp matches the user's input of otp code
        messagebox.showinfo("Information", "OTP code is correct", parent=...self.two_factor)
        os.system(r"C:\Users\rusha\PycharmProjects\pythonProject5\sms_dashboard.py") # opens the dashboard window
    else: # else an error occurs
        messagebox.showerror("Error", "Please type correct otp code", parent=...self.two_factor)
    return
```

Here, the program checks if the generated OTP code matches the one entered by the user. If so, a message shows saying that the “OTP code is correct”. Then, the program allows the user to access the dashboard file where I have hard coded the file path.

However, if the OTP code entered by the user is wrong, then an error message is shown.

Testing: When the user enters then username and password correctly, OTP code should be sent

<u>55.</u>	Random OTP number is generated and sent via email	N/A	N/A	N/A	6-digit random number generated each time
------------	---------------------------------------------------	-----	-----	-----	-------------------------------------------



Here, I have entered the username and password correctly as it shows the message showing OTP code is being sent via email.

Your OTP code Inbox ×

 testpurposes683@gmail.com
to bcc: me ▾
Your otp code is:607033

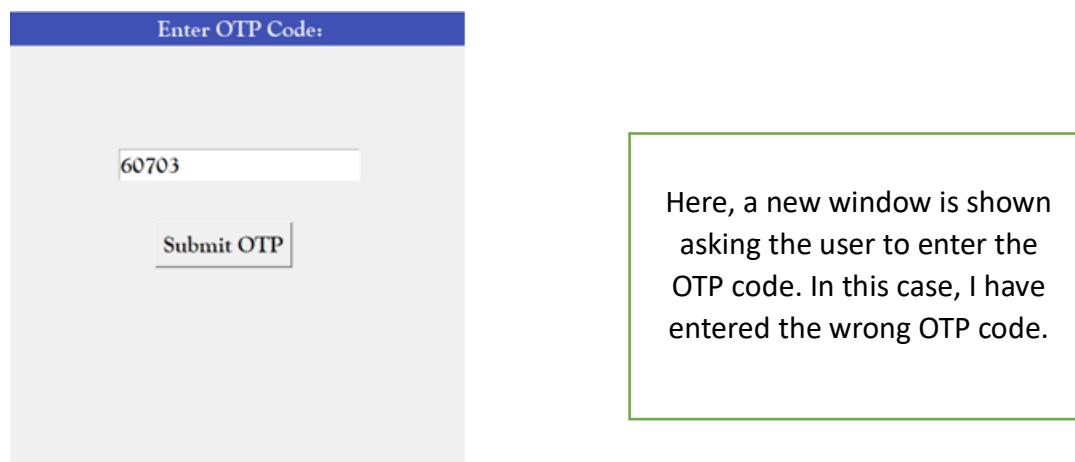
← Reply → Forward

Here, this screenshot shows the 6-digit OTP code.

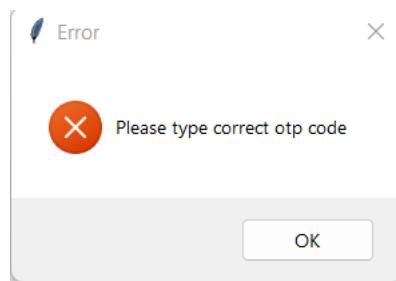
<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>55.</u>	Random OTP number is generated and sent via email	N/A	N/A	N/A	6-digit random number generated each time	Passed – Email sent showing the OTP code

Testing: Checking if OTP code matches the user's input

<u>56.</u>	Check if OTP code is correct	Correct OTP code	N/A	Invalid OTP code	No error message appearing
------------	------------------------------	------------------	-----	------------------	----------------------------



When I press the Submit OTP button:

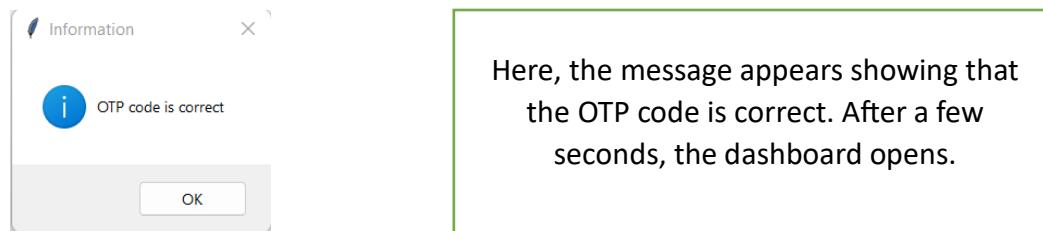


The program gives an error message asking the user to enter the correct OTP code. This ensures that in case a third-party user gets access to the intended user's login details (such as their username and password), they won't be able to access the dashboard window as they will not receive the generated OTP code.

If I enter the correct OTP code:



Here, I have entered my login details again, and when I submit the correctly entered OTP code:



<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
--------------------	-------------------------	---------------	-----------------	------------------	-------------------------	-----------------------

56.	Check if OTP code is correct	Correct OTP code	N/A	Invalid OTP code	No error message appearing	Passed – program doesn't allow incorrect OTP code. Program allows the user to enter the dashboard if the OTP code is correct (once username and password have been entered correctly)
-----	------------------------------	------------------	-----	------------------	----------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

If the user forgets their password, then they should at least know their username.

Function if user forgets password.

Precondition: The user must know their username

```
##### function if user forgets their password #####
def forget_password(self):
    connection = sqlite3.connect(r'sms.db') # connecting your database
    cur = connection.cursor()
    try:
        if self.username.get() == "": # if the username is empty
            messagebox.showerror("Incomplete", "Please fill in the username to get the OTP code", parent=_self.wind)
            return
        else:
            # checks if username is correct
            cur.execute("Select Email from login where Username=?", (self.username.get(),)) # get the email address from the user's username
            email = cur.fetchone()
            if email == None: # no record found
                messagebox.showerror("Error", "Username doesn't exist", parent=_self.wind)
                return
```

This program checks if the user has put an input into the username entry box. If the user has left this entry box empty, then an error message appears.

If the user has entered an input, then an SQL query is used to fetch the email address of the user from the login table using the user's input as part of the WHERE condition. If no records are found, this means that the user's input of their username is invalid and an error message occurs.

```

else:
    self.var_otp_code = IntVar() # this otp variable stores the otp code that the user will enter to change their password
    self.var_updated_password = StringVar() # stores updated password from user's input
    self.confirm_password = StringVar() # stores the user's confirmed password input

    self.forget_password = Toplevel(self.wind) # creates a new window
    self.forget_password.title("New Password")
    self.forget_password.geometry("500x500+500+110")
    self.forget_password.focus_force()

```

If the username is correct, then 3 variables are defined:

- “self.var_otp_code” – this variable stores the user’s input of the OTP code that is needed to change the password. This variable is different than the one used for the 2-factor authentication.
- “self.var_updated_password” – This stores the user’s new password that they want
- “self.confirm_password” – This stores the user’s confirmed password

Here, a forget password window is created.

```

title = Label(self.forget_password, text="Reset Password", font=('goudy old style', 15, 'bold'))
    bg = "#3f51b5", fg = "white").pack(side=TOP, fill=X)

lbl_forget_password = Label(self.forget_password, text="Enter OTP sent on Email",
    font=('goudy old style', 15, 'bold')).place(x=20, y=60)

```

This code here creates a label showing the title “Reset Password” and a label to tell the user to enter the OTP code sent on email.

```

#####
entry_forget_password = Entry(self.forget_password, textvariable=self.var_otp_code, font=('goudy old style', 15, 'bold')).place(x=20, y=100)

self.btn_send = Button(self.forget_password, text="Check OTP", command=self.check_otp,
    font=('goudy old style', 15, 'bold')).place(x=20, y=150, width=150, height=50)

updated_pass_lbl = Label(self.forget_password, text="Update password", font=('goudy old style', 15, 'bold')).place(x = 20, y=240)

```

Here, I have created an entry box for the user to enter the OTP code that they will need to change their password.

A button has also been created which is linked to a function that checks if the OTP code is correct or not.

```

updated_pass_lbl = Label(self.forget_password, text="Update password", font=('goudy old style', 15, 'bold')).place(x = 20, y=240)

#####
input_new_pass = Entry(self.forget_password, textvariable=self.var_updated_password, font=('goudy old style', 15, 'bold')).place(x=20, y=290)

lbl_confirm_new_password = Label(self.forget_password, text="Confirm new password", font=('goudy old style', 15, 'bold')).place(x=20, y=340)

#####
entry_confirm_new_password = Entry(self.forget_password, textvariable=self.confirm_password, font=('goudy old style', 15, 'bold')).place(x=20, y=390)

self.update_btn = Button(self.forget_password, text="Update password", command = self.update_password,
    font=('goudy old style', 15, 'bold')).place(x=20, y=440, width=200, height=50)

gmail_user, gmail_password, to_email = self.fetch_details()
self.send_email(to_email, self.otp, gmail_user, gmail_password)

```

Here, I have created entry boxes for the user to enter their new password and to confirm the new password. A button has also been created that is used to update the password once everything has been validated. Here, an email gets sent to the user (just like when the user enters their username and password correctly) showing the OTP code that is generated in the send_email() function.

Function that validates the OTP code if the user has forgotten their password

```
def check_otp(self):
    if int(self.otp) == int(self.var_otp_code.get()): # if the otp code entered by user matches the generated one
        messagebox.showinfo("Information", "OTP code is correct", parent=self.forget_password)
    else:
        # shows an error
        messagebox.showerror("Error", "Please type correct otp code", parent=self.forget_password)
    return
```

This code here checks if the OTP code matches the one generated, else an error occurs.

Function to update password

```
def update_password(self):
    if self.var_updated_password.get() == "" or self.confirm_password.get() == "": # if any of the fields are empty
        messagebox.showerror("Error", "Password is required", parent=self.forget_password)
        return
    elif self.var_updated_password.get() != self.confirm_password.get(): # if the confirmed password and the changed password do not match
        messagebox.showerror("Error", "Please check that the password entered is the same", parent=self.forget_password)
        return
```

This part of the function checks if the user has entered in both the new password field and confirm password field. If not, then an error message is shown.

If the new password and the confirmed password do not match, then again an error message is shown.

```
else:
    con = sqlite3.connect(database=r'sms.db') # connect to database
    cur = con.cursor()
    try:
        # updates the password by using the username as part of the where condition
        cur.execute("Update Login SET Password=? where Username=?".format(self.var_updated_password.get(), self.username.get()))
        # message showing that the password has been updated successfully
        messagebox.showinfo("Password", "Successfully updated password", parent=self.forget_password)
        con.commit()
    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.forget_password)
```

However, if the new password and the confirmed password match, then a database connection is made, and SQL query is used to updated the password in the login table.

Testing: OTP code validation when user forgets password



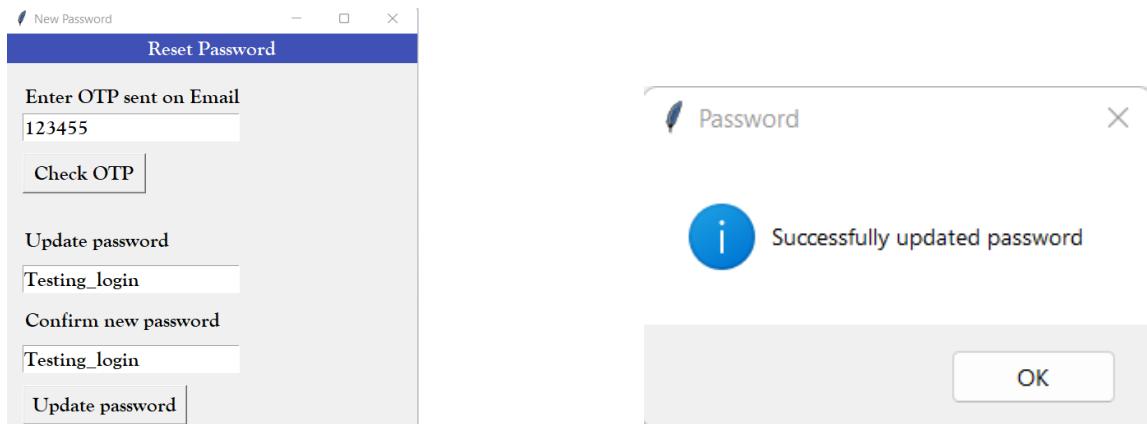
testpurposes683@gmail.com

to bcc: me ▾

Your otp code is:719114

Reply

Forward



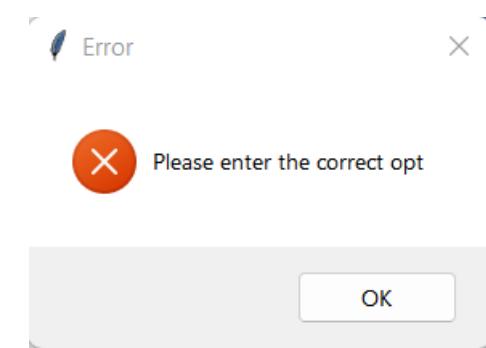
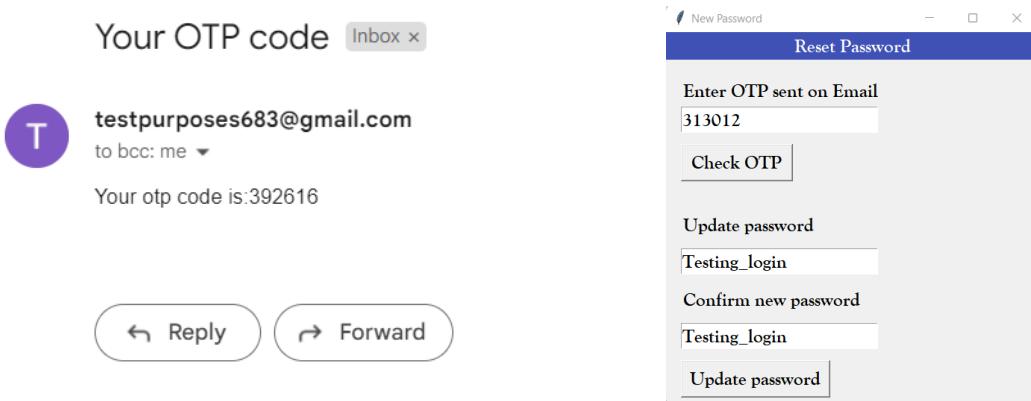
Here, as you can see, I have entered the wrong OTP code. However, the message box shows that the password has been successfully updated.

```
elif int(self.otp) != int(self.var_otp_code.get()): # if the otp code entered by the user doesn't match the one generated
    messagebox.showerror("Error", "Please enter the correct otp", parent=_self.forget_password)
    return
elif self.var_otp_code.get() == "": # if the user doesn't enter otp code
    messagebox.showerror("Error", "Please fill in the OTP code", parent=_self.forget_password)
    return
```

I have added validation to the OTP code on the update_password() function.

This same code is written in the check_otp() code.

I have run the program again and have entered the wrong OTP code:



After adding the validation, an error message occurs telling the user to enter the correct OTP code. This function now validates the program. Since this function repeats the code written in the check_otp() function, I have decided to remove that function (check_otp) as the OTP code can be checked in the update_password() function.

This helps remove the redundancy in code.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>56.</u>	Check if the OTP code is correct	N/A	N/A	Invalid OTP code	No error message appearing	Initially failed as program allows password to be updated, even if OTP code entered was wrong. It has now been fixed by adding validation to the update_password() function.

Testing: Password being updated

Your OTP code Inbox ×



testpurposes683@gmail.com
to bcc: me ▾

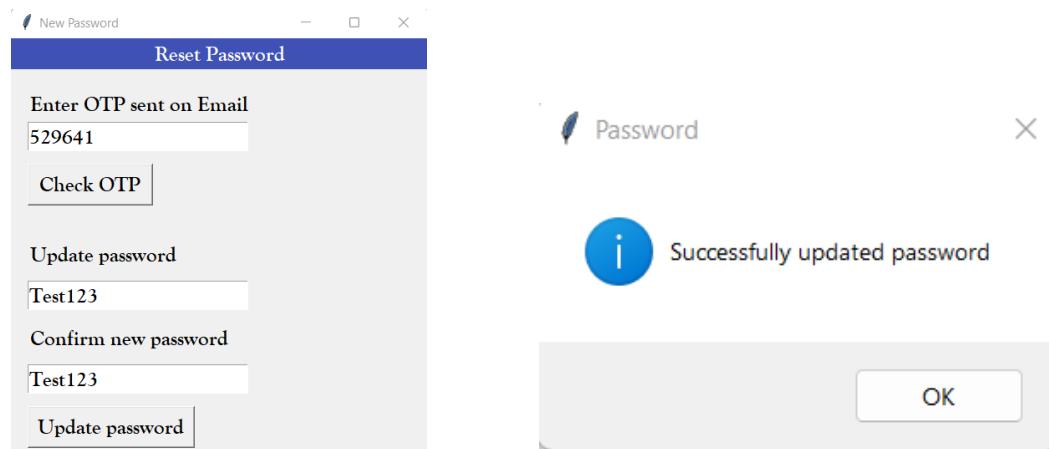
Your otp code is:529641

Reply

Forward

	ID	Username	Password	Email
	Filter	Filter	Filter	Filter
1	1	Trial_test	Testing_login	testpurposes683@gmail.com

This is the current password stored in the login table.



Here, I have entered the correct OTP code which results in a message showing that the password has been successfully updated.

	ID	Username	Password	Email
	Filter	Filter	Filter	Filter
1	1	Trial_test	Test123	testpurposes683@gmail.com

As you can see here, the Password has been successfully changed.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>57.</u>	Password being updated	N/A	N/A	N/A	Password should be updated in the login table. The next time the user logs in, the new password should be used.	Pass – Password gets changed successfully

Client Review:

I have shown this to my client who is happy with the login system:

Me: Are there any changes you would like in the program?

Parth: Yes, from the tests shown the password is visible when entering it in the entry boxes. Can you ensure that the password cannot be seen in plain English?

Me: Yes, I can change the password so that all the characters show the same letter.

Parth: Yes, that would be great. Can you change it to an asterisk (*)?

Me: Yes, I can do that. Should I do this for all the entry boxes that require a password?

Parth: Yes.

Code to ensure that the password is covered with asterisks

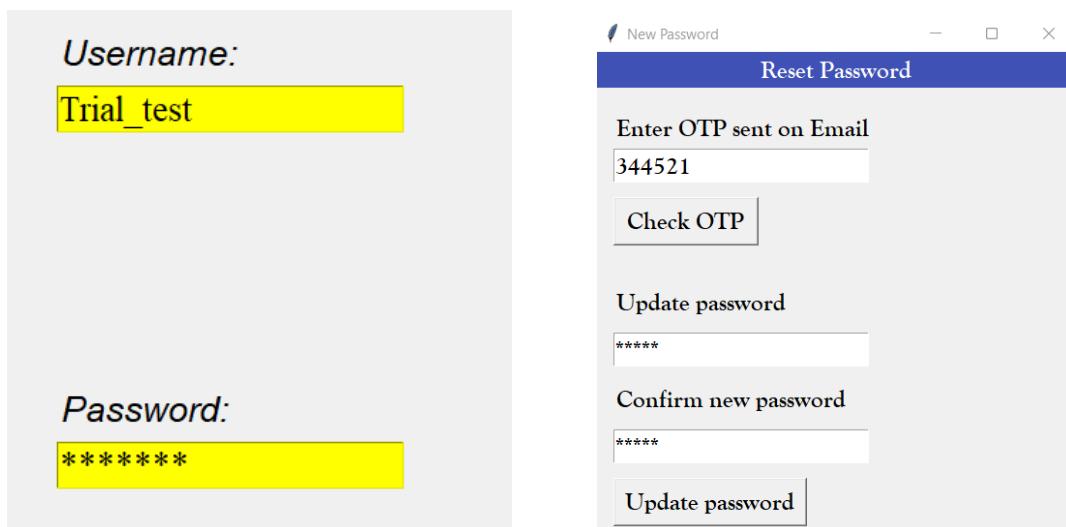
```
#####
# password entry box #####
password_entry = Entry(login_section, textvariable= self.password, show = "*", font= ("times new roman", 15), bg = "yellow").place(x = 50, y = 420)

#####
# entry box to enter user's new password #####
input_new_pass = Entry(self.forget_password, textvariable= self.var_updated_password, show = "+", font= ('goudy old style', 15, 'bold')).place(x=20, y=290)

#####
# entry box to confirm the user's new password #####
entry_confirm_new_password = Entry(self.forget_password, textvariable= self.confirm_password, show = "*",
                                    font= ('goudy old style', 15, 'bold')).place(x=20, y=390)
```

In this code here I have added 'show = "*" which means that all the characters in the password will be visible in the form of an asterisk.

Testing: Password characters converted to *



Here, as you can see the passwords are covered with asterisks.

<u>Test Number</u>	<u>What I'm testing</u>	<u>Normal</u>	<u>Boundary</u>	<u>Erroneous</u>	<u>Expected Results</u>	<u>Actual outcome</u>
<u>58.</u>	Password should not be visible in English	N/A	N/A	N/A	When user types in their password, it should be converted into asterisks.	Pass – Password is no longer visible as it is covered in asterisks.

Client review:

I have shown this to my client who is happy with the login system and so there are no more changes to be made.

Stage 10: Exit button

```
butn_exit = Button(LeftMenu, text="Exit", font=("times new roman", 42, "bold"), bg="white", bd=3,
command = wind.destroy, cursor="plus").pack(side=TOP, fill=X) # exit button
```

Here, I have added the code 'command = wind.destroy' which destroys the window itself.

Evaluation

Stakeholder testing

I have given a copy of my software to Parth along with all the csv files and the database that are stored in a single folder.

I have also asked the following questions: (These questions may or may not be covered in the post-development data).

How user-friendly was the GUI? – This question will overlap with the usability features section so I will cross-reference my client's response.

Was the program able to import the CSV Sales file easily?

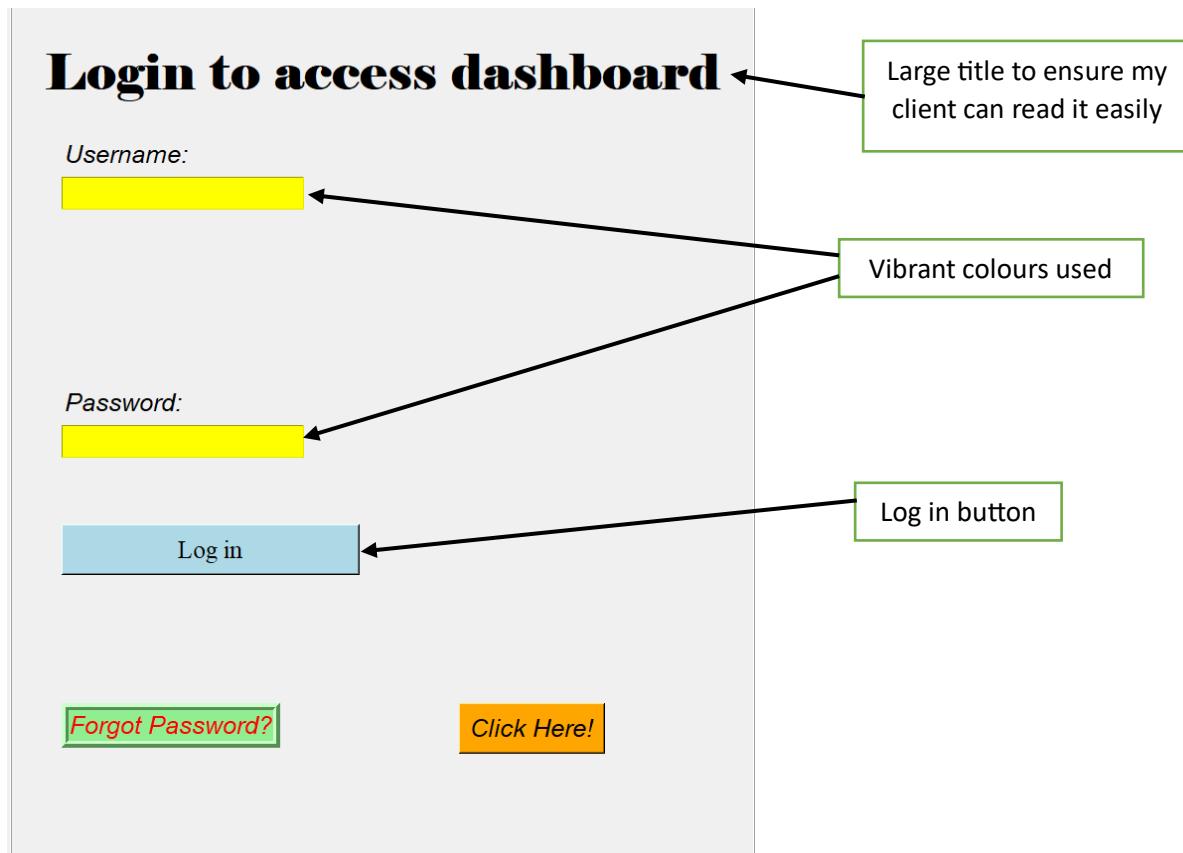
Were all the food stocks updated correctly once the CSV Sales file was imported?

Were all the graphs easy to interpret and read?

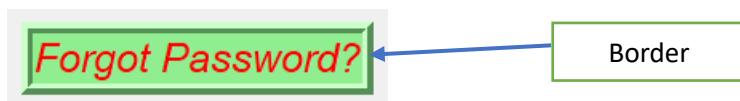
Was the program able to forecast sales for both quantity and value (£)? - I have asked this question as my client asked me to add this feature in my program as I was developing the solution for the sales window.

Client response

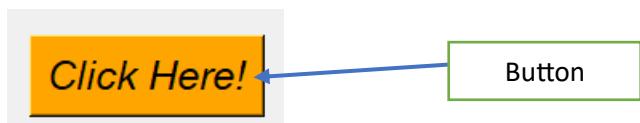
Question: How user-friendly was the GUI?

Login

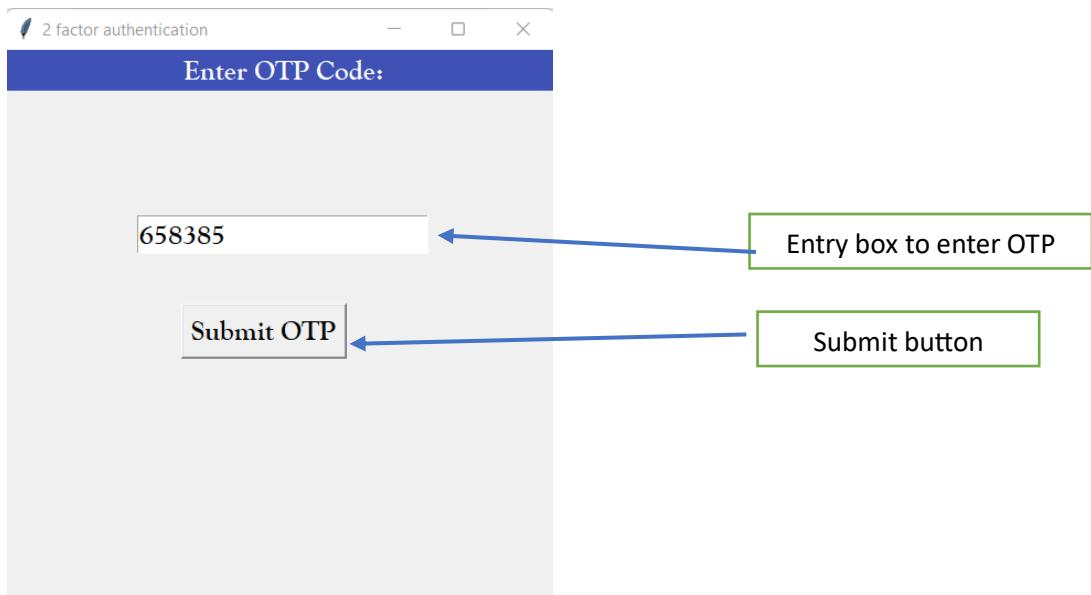
Parth: The main login window looked really nice with lots of vibrant colours used especially for the entry boxes. The title is also large enough to read to tell me what I have to do in this window. I also like the colour of the button to log in.



However, what I didn't like was the Forgot Password label. I didn't like the text being in italics and the border of the button. The border of the label could have been just a regular outline like how the login button is instead of making it too fancy. Also, the text in the label seems to congested.



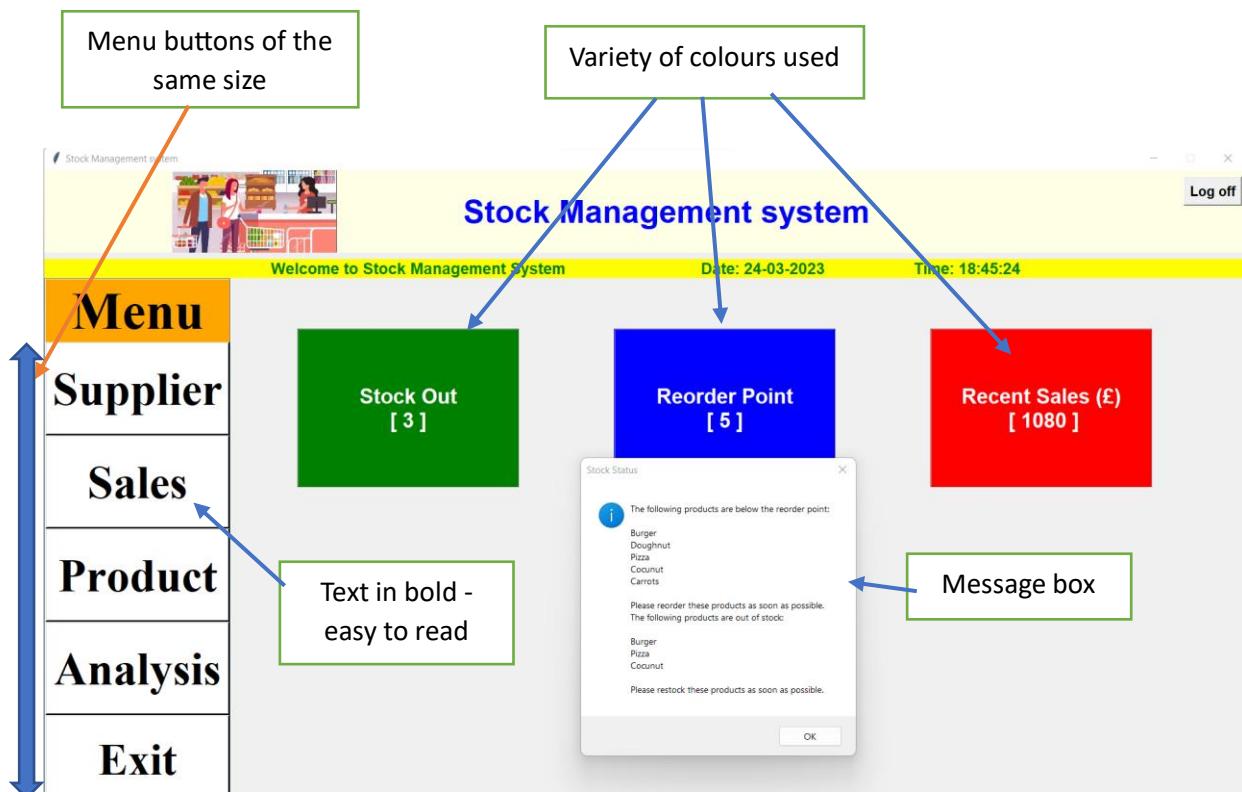
Here, I didn't like the "Click Here!" text being in italics.



The OTP code for the 2-factor authentication is easy to use as I can easily enter the OTP code in the entry box and the 'submit' button works as well.

The heading with the blue background and the contrasting text colour looks nice. However, I don't like the rest of the background being grey. The window can be made more attractive by adding a wide range of colours.

Dashboard



Parth: The dashboard looks elegant as there are lots of colours used for the Stock-out, Reorder point buttons and for the recent sales label.

The label and the reorder point and stock-out buttons are easy to read as the text and the backgrounds have contrasting colours.

The menu button texts are large and are in bold which helps me identify which window coordinates one aspect of the system. The menu buttons are also of the same height and width making it look uniform and tidy.

The message box showing the status of the grocery store is very informative and it is easy to identify the products that are out of stock or below the reorder point as they are named one after the other as a column.

Supplier

Supplier ID	Name	Email
1	Sam	rushabh.1s.cool2@gmail.com
2	Rushabh	rushabh.1s.cool2@gmail.com
3	Bob	wateraid2.0@gmail.com
4	Nijan	nijankannathasan@gmail.com
5	Test	Test@gmail.com
6	Rayyan	elhambre049@gmail.com
15	Trial	Trial@gmail.com

The GUI for the supplier is really easy to interact with. For example, it is easy to enter the supplier details into the entry boxes. The labels tell me what input should go in each of the entry boxes.

The 4 buttons are easy to use and the colours used for them look nice. This makes it easy to identify the functionality of each button.

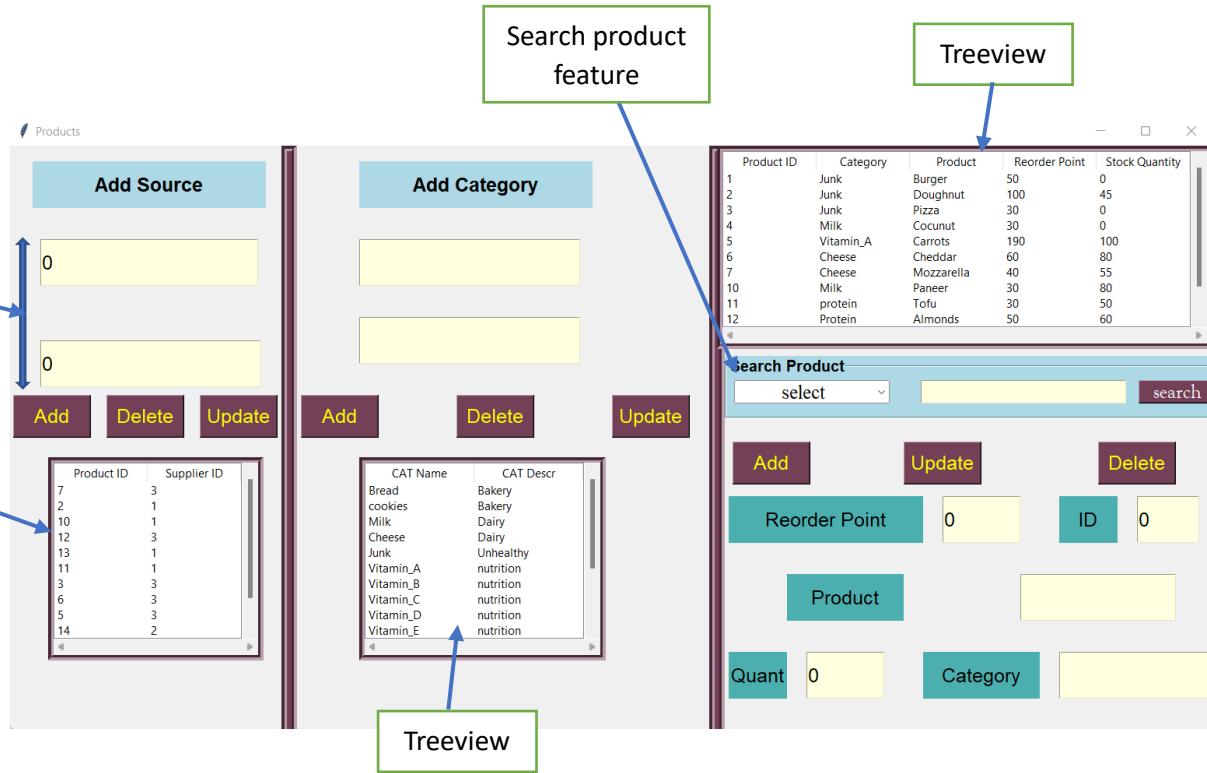
There is a search feature at the top which allows me to search for the suppliers.



I really like having the choice to select one of the options above as I can type into the search bar whichever supplier detail (name, email and ID) I remember the most.

The treeview on the right side of the window helps to look at the data in the Supplier table without going on the application. This saves time as I can look at the supplier details in the program itself and can use it along with the search feature to hide any unmatched records.

Product window

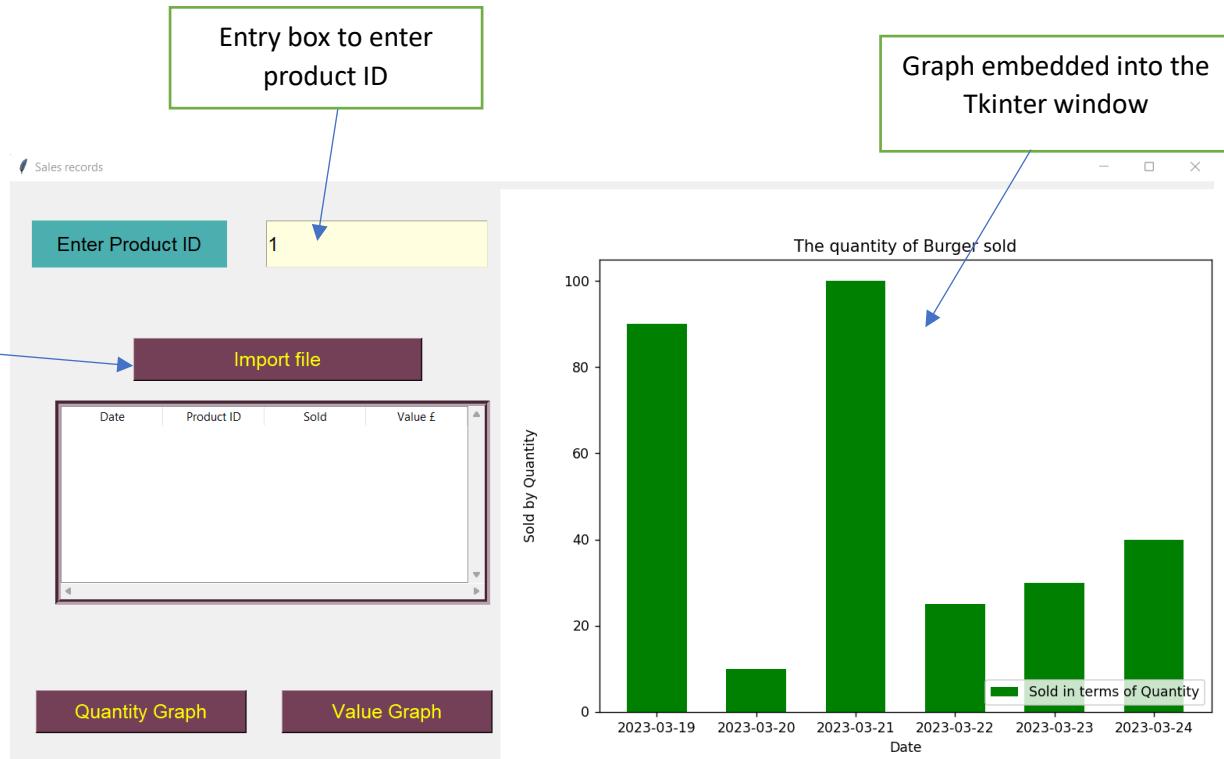


Parth: I really liked the idea of using frames to split the different aspects of the product window. There are lots of treeviews used to help me know the current data for each table.

Once again, the entry boxes are large enough to enter data into them. The buttons to add, update and delete are clear and easy to read as they are spaced out equally, though the buttons in the source section look a bit congested.

I really like the Search Product feature as I can easily look for a certain item without scrolling through hundreds of treeview records. This allows me to manually update the quantity of any products by using this feature.

The labels in the product section are not equally sized but I can identify what entry box corresponds to the fields in the product table. Though there are no labels used for the Source and Category section, I have discussed this with Rushabh who said that the entry boxes are in the order of the fields of their respective table.

Sales window

Parth: I like the use of the entry box to enter the Product ID and then the corresponding graphs will show up. That way, I can view all the existing products in my store and see their individual sales performance for the last 7 days. I like the name of the title being shown in the graph – that way, I can identify the product much more easily than the title showing Product ID 1.

The import file button is large enough to read and is easy to use as I just have to click on the button and then automatically the CSV file gets imported to the Sales table.

I don't see much point of having the treeview as it doesn't interact with the graphs – though it can be useful in showing whether the csv file has been successfully imported or not by looking at the recent sales data.

The graph buttons are helpful in generating graphs / new graphs each time I have entered a product ID.

The graph itself is displayed elegantly and the bars are equally spaced apart which helps to interpret data much more easily and also helps to identify the date of each of the bars. The bars are of good size – which helps me to read the graph much more quickly.

However, with the graph, it doesn't show the exact value of the data when I put my cursor near the graphs. Though I can estimate the rough sales made, it would be better if there was a label on top of the bar to show the exact sales of that bar.

Analysis

The screenshot shows an Analysis window with two main sections: 'Worst 5 Products' on the left and 'Top 5 Products' on the right. Each section contains a treeview control displaying product data. A green box labeled 'Labels – large enough to read' points to the top of the window. Another green box labeled 'Treeview in ascending order' points to the 'Worst 5 Products' treeview, which lists products from ID 4 to 5. A blue arrow points from this treeview to the 'Top 5 Products' treeview, which lists products from ID 1 to 5. A blue arrow also points from the 'Top 5 Products' treeview back to the 'Worst 5 Products' treeview. A green box labeled 'Treeview in descending order' points to the 'Top 5 Products' treeview. Two 'Graph button' labels with arrows point to a central 'Show graph' button.

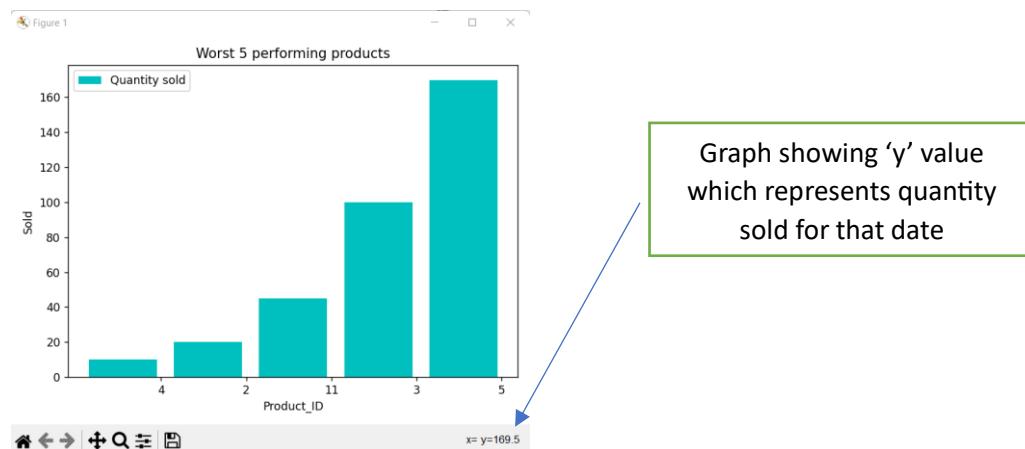
Product_ID	Product	Sold
4	Cocnut	10
2	Doughnut	20
11	Tofu	45
3	Pizza	100
5	Carrots	170

Product_ID	Product	Sold
1	Burger	295
10	Paneer	210
5	Carrots	170
3	Pizza	100
11	Tofu	45

Parth: This Analysis window looks really nice as the labels are nice and big which makes it really easy to identify which treeview contains the worst 5 or top 5 products. I liked the treeview showing the product name as well as the product ID as it makes it easier to identify the name of the product rather than using the search feature in the product window to get the name of the product.

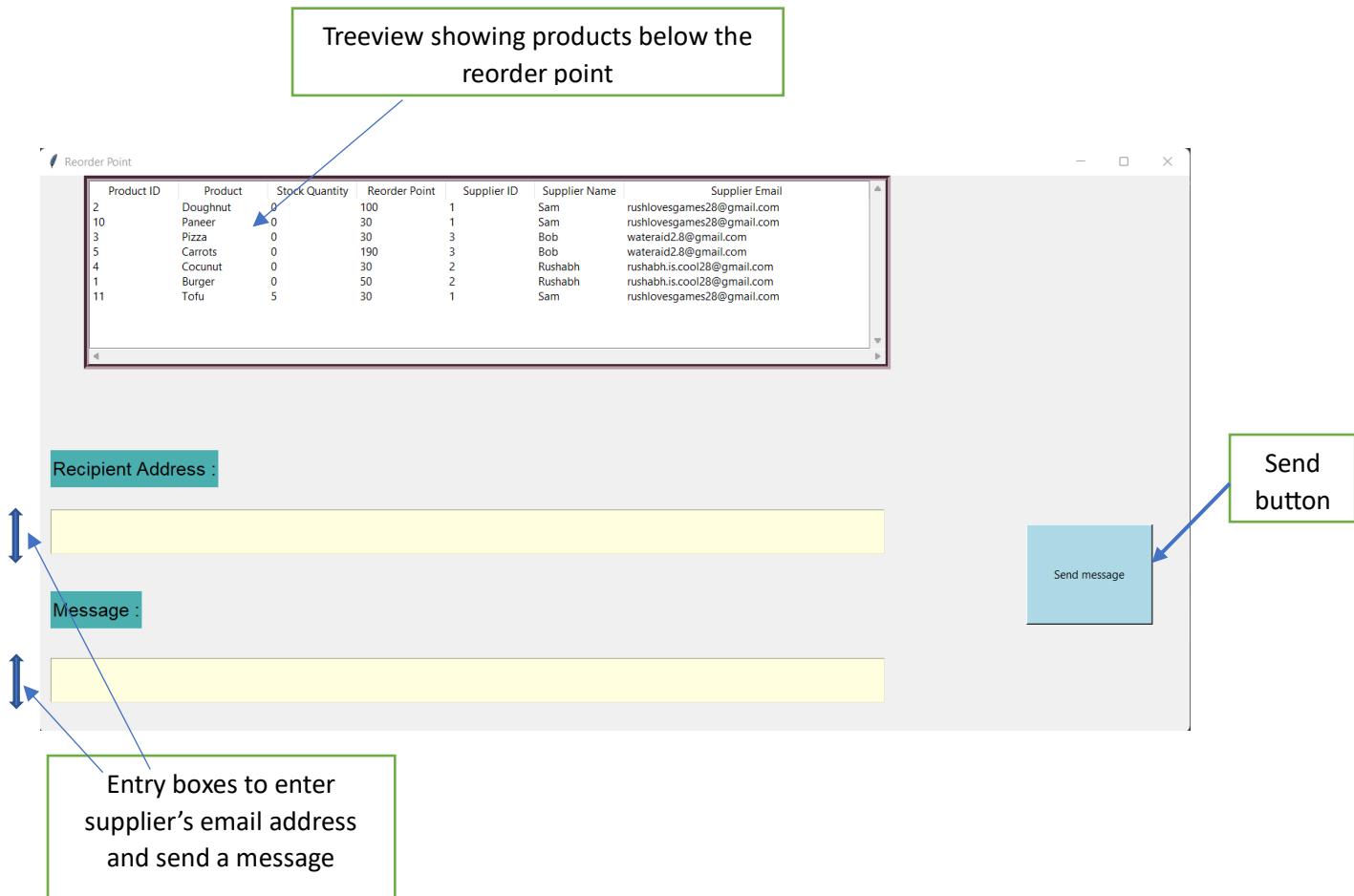
The treeviews are also ordered based on the quantity sold, so it is easy to identify the worst 5 and top 5 products.

The graph buttons are visible and have been made well. It automatically produces the graphs when I click on the buttons.



Here, the graphs also show the quantity sold (bottom-right: $y = 169.5$) when you move your cursor to the top of each bar.

Reorder-point



Parth: This window allows me to see the products that need reordering and it also shows the details of the suppliers that are connected to it. The labels help me to identify what each entry box is used for. The send button designed looks fine and is easy to use. Perhaps, the text "send message" could be a bit bigger but other than that, the button is fine.

I like the idea of clicking on a treeview record, which then populates the email address onto the entry box. This helps saves time and prevents any spelling errors instead of typing the entire email address.

The size of the entry box for the recipient email address is suitable, but the entry box for the message could have designed so that it starts a new line each time a sentence is the same length as the entry box. This would allow me to see all the sentences of the message and check for any grammar errors without having to look at one single very long sentence.

Stock-out

Treeview showing products out of stock

Product ID	Product	Stock Quantity	Reorder Point	Supplier ID	Supplier Name	Supplier Email
2	Doughnut	0	100	1	Sam	rushlovesgames28@gmail.com
10	Paneer	0	30	1	Sam	rushlovesgames28@gmail.com
3	Pizza	0	30	3	Bob	wateraid28@gmail.com
5	Carrots	0	190	3	Bob	wateraid28@gmail.com
4	Cocunut	0	30	2	Rushabh	rushabh.is.cool28@gmail.com
1	Burger	0	50	2	Rushabh	rushabh.is.cool28@gmail.com

Recipient Address :

Message :

Send message

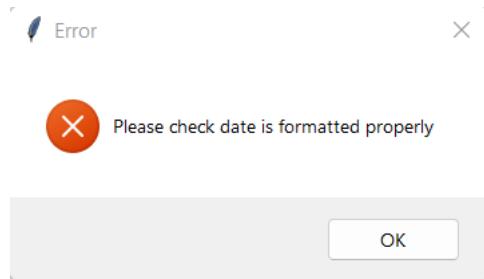
Send button

Entry boxes to enter supplier's email address and send a message

Parth: Same comment as mentioned in reorder point.

Question: Was the program able to import the CSV Sales file easily?

Parth: In most of the cases, there was no problem in importing the CSV file into the Sales table. However, sometimes the program would show an error message telling me to check whether the date is formatted correctly or not, even though the date was in the right format.



Follow on question:

Question: What did you do to ensure that the csv file gets imported successfully every time?

Parth: I just checked the date in the CSV file and ensured that it was in the 'YYYY-MM-DD' format. Whenever I open the file, the date is sometimes in the wrong format, so I just

ensure that I change the format of the date. Most of the time, the date is in the right format, but I still converted it again to the 'YYYY-MM-DD' format to ensure the error message doesn't show up again.

Question: When you corrected the format of the date, did the import button work?

Parth: Yes, it worked after I changed the format of the date.

Question: Were all the food stocks updated correctly once the CSV Sales file was imported?

Parth: Yes, all the food stocks were updated correctly. However, there were a few cases where I had pressed the import file button more than once accidentally, which caused the product levels to be updated again.

Follow on:

Me: Yes, pressing the import file button will cause the stock levels to be updated again. I have designed it this way, in case the CSV sales file is missing some data.

Parth: Ok, yes that's fine. I will try and remember that.

Me: Do you think it would have been better if I added a message telling that clicking the import file will update the product quantities again.

Parth: Yes, that would really help me from pressing the import file more than once unnecessarily.

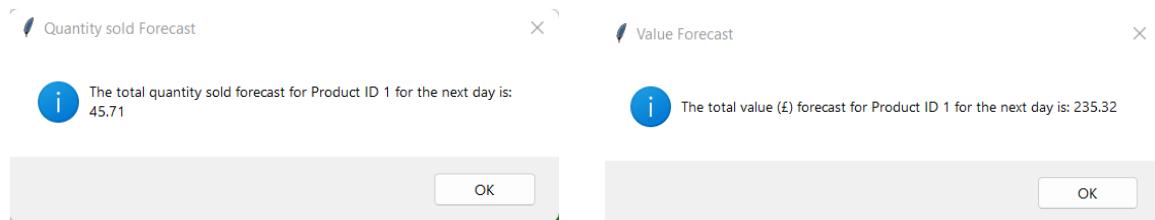
Were all the graphs easy to interpret and read?

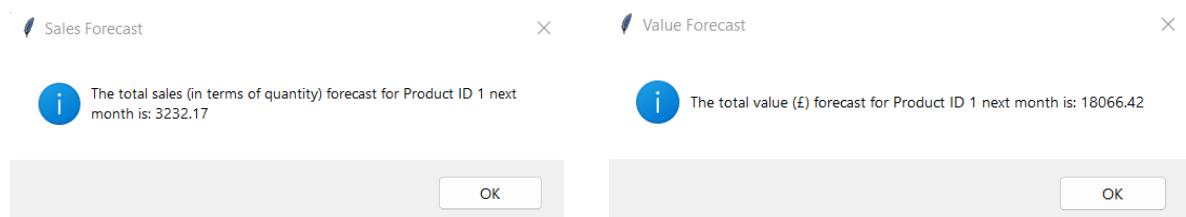
Parth: Yes, in the sales window, I can easily compare the bars as the width is of good size and the bars are spread out evenly. This has helped me tell on which days I created better sales for that product. However, as mentioned before, the graph doesn't show me the exactly (y value) of the graph, unlike in the Analysis section where I can get the exact total quantity sold for one week by moving my cursor to the top of the bar.

Nevertheless, the graphs are really helpful in assessing the performance of individual products (sales window graphs) as well as comparing it to others.

Was the program able to forecast sales for both quantity and value (£)?

Parth: Yes, each time I clicked on either the quantity or the value graph, 2 messages appeared showing the forecast for tomorrow and the total forecast for next month.





Client testing using post development data

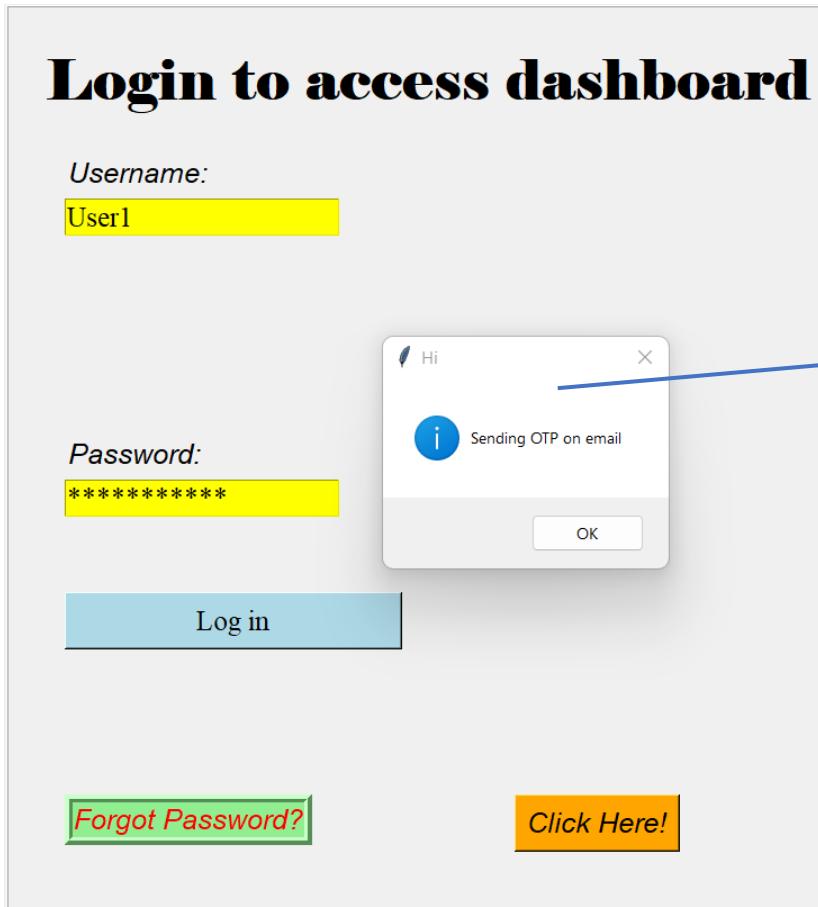
Here my client Parth will use the post development data that is linked to my success criteria to test the software and then give a rating out of 10 to assess how effectively the software match the success criteria.

Test ID: 1

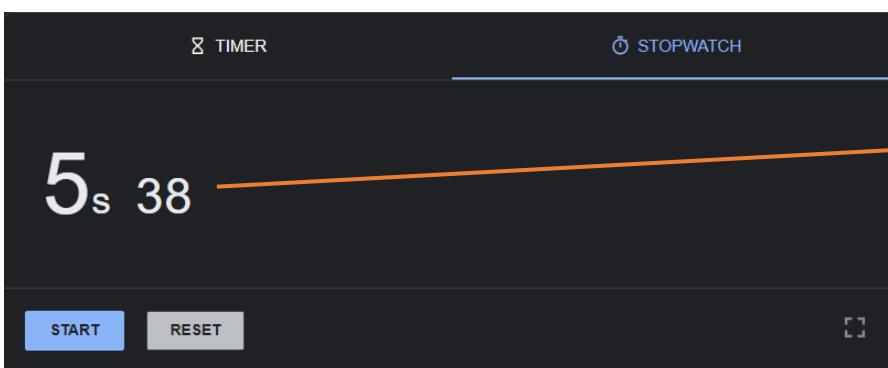
Before my client tests the software, they have changed their password and username in the login table. I have told my client that the OTP code will appear in his email when he presses the “ok” button on the message box.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
1.	Login to the system with a username and password and check if an OTP number is received on mobile phone.	Enter username: User1 Enter password: Inventory#1 Click on 'Log In'	OTP received on the registered phone number of User1. OTP should be received within 3 minutes.	S1, S4

Here, my client has their email application on their phone.



□ ★ me Your OTP code - Your otp code is:994642 12:07



Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
1.	Login to the system with a username and password and	Enter username: User1 Enter password: Inventory#1	OTP received on the registered phone number of User1.	S1, S4	Pass – OTP code received

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
	check if an OTP number is received on mobile phone.	Click on 'Log In'	OTP should be received within 3 minutes.		within 6 seconds

Test ID 2:

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
2.	Login to the system with a username and password to receive an OTP number on registered mobile phone. Once OTP is entered, access is allowed to system.	Enter username: User1 Enter password: Inventory#1 Click on 'Log In' Enter OTP and then click on 'Submit'	Once OTP is entered, access is allowed to system.	S1

Testing: Checking if OTP code matches the user's input – My client has cross-referenced this testing with the iterative testing I did in the development section.

In the link shown, I had put an incorrect OTP code which showed an error message.

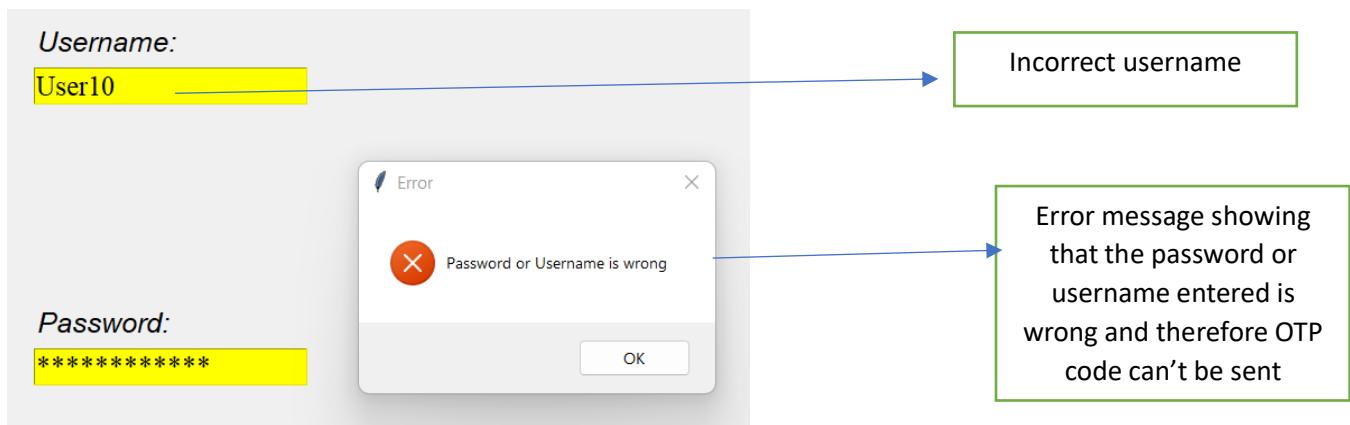
However, when I put the correct OTP code, a message shows up telling me that the OTP code is correct, and after a few seconds the dashboard opens up.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
2.	Login to the system with a username and password to receive an OTP number on registered mobile phone. Once OTP is entered, access	Enter username: User1 Enter password: Inventory#1 Click on 'Log In' Enter OTP and then click on 'Submit'	Once OTP is entered, access is allowed to system.	S1	Pass – message showing that OTP code is correct and then a few seconds the later,

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
	is allowed to system.				the dashboard opens up.

Test ID: 3

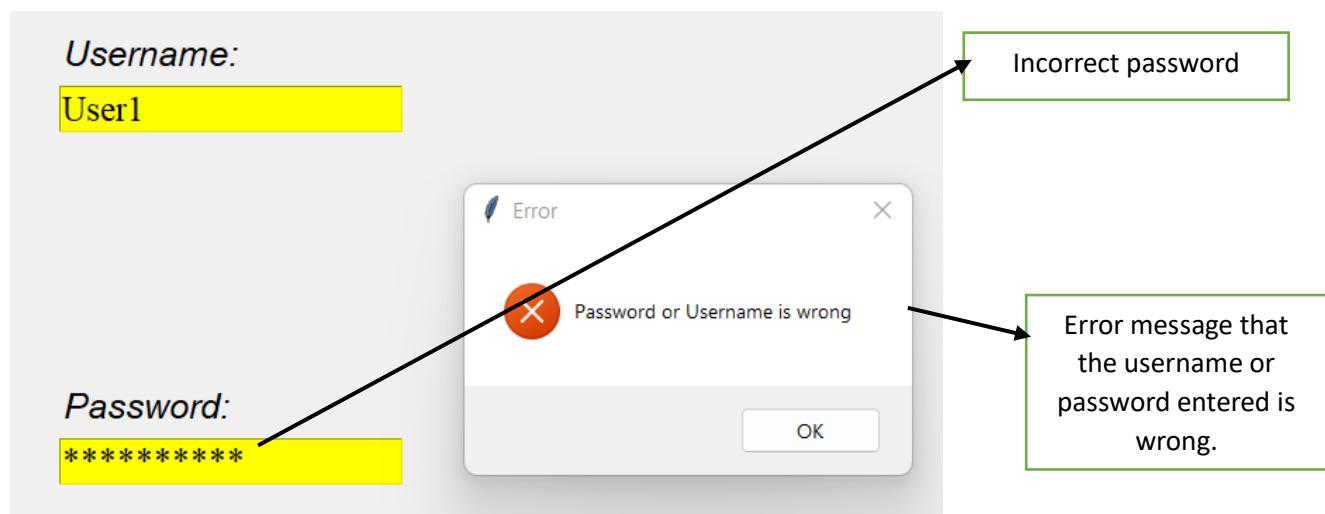
Test ID	Description	Test Data	Expected Results	Reference Success Criteria
3.	Login to the system with an incorrect username	Enter username: User10 Enter password: Inventory#1 Click on 'Log In'	Message displayed on screen saying 'Invalid username or password'. There should be no OTP received on phone.	S2



Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
3.	Login to the system with an incorrect username	Enter username: User10 Enter password: Inventory#1 Click on 'Log In'	Message displayed on screen saying 'Invalid username or password'. There should be no OTP received on phone.	S2	Pass – an error message occurs showing that the username or password is wrong

Test ID: 4

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
4.	Login to the system with an incorrect password	Enter username: User1 Enter password: Inventory1 Click on 'Log In'	Message displayed on screen saying 'Invalid username or password.' There should be no OTP received on phone.	S2



Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
4.	Login to the system with an incorrect password	Enter username: User1 Enter password: Inventory1 Click on 'Log In'	Message displayed on screen saying 'Invalid username or password.' There should be no OTP received on phone.	S2	Pass – an error message occurs showing that the username or password is wrong

Test ID: 5

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
5.	Login to the system with valid username and password but an incorrect OTP	Enter username: User1 Enter password: Inventory#1 Click on 'Log In' Do not enter the OTP received on phone but put a different OTP number.	Message displayed on screen saying 'Invalid OTP.' Access to system is not allowed.	S2

Testing: Checking if OTP code matches the user's input

My client has cross-referenced this testing with the iterative testing I did. Here, if the OTP code entered is wrong, then an error message shows up asking to type the correct OTP code.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
5.	Login to the system with valid username and password but an incorrect OTP	Enter username: User1 Enter password: Inventory#1 Click on 'Log In' Do not enter the OTP received on phone but put a different OTP number.	Message displayed on screen saying 'Invalid OTP.' Access to system is not allowed.	S2	Pass – an error message occurs telling the user to enter the correct OTP code

Test ID: 6

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
6.	User forgets password and is allowed to reset it.	Enter username: User1 Click on 'Click Here' User enters the OTP received to allow the new password to be updated. User must enter the password	OTP is received on Email And user can enter a new password.	S3

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
		again to confirm the new password		

Testing: OTP code validation when user forgets password

This cross-referenced test ensures that the user has entered the right OTP code. Before, the test didn't work, but after adding validation, the program ensures that the OTP code entered should be right if the user wants to change their password.

The top screenshot shows an email inbox with an incoming message from 'testpurposes683@gmail.com'. The message subject is 'Your OTP code'. The body of the email contains the text 'Your otp code is:570487'. A blue arrow points from this text to a callout box labeled 'OTP code sent on email'. Below the email are two buttons: 'Reply' and 'Forward'.

The bottom screenshot shows a 'Reset Password' window titled 'Reset Password'. It has three input fields: 'Enter OTP sent on Email', 'Update password', and 'Confirm new password'. A large double-headed vertical arrow connects the 'Enter OTP sent on Email' field to the 'Update password' field. Another double-headed vertical arrow connects the 'Update password' field to the 'Confirm new password' field. A blue arrow points from the 'Enter OTP sent on Email' field to a callout box labeled 'New window shown if the user forgets their password'. Another blue arrow points from the 'Update password' field to a callout box labeled 'Entry boxes are not equally spaced and there is a big gap in the window'.

Testing: Password being updated

This cross-referenced test shows the password being changed successfully in the login table.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
6.	User forgets password and	Enter username: User1	OTP is received on Email And	S3	Pass – New

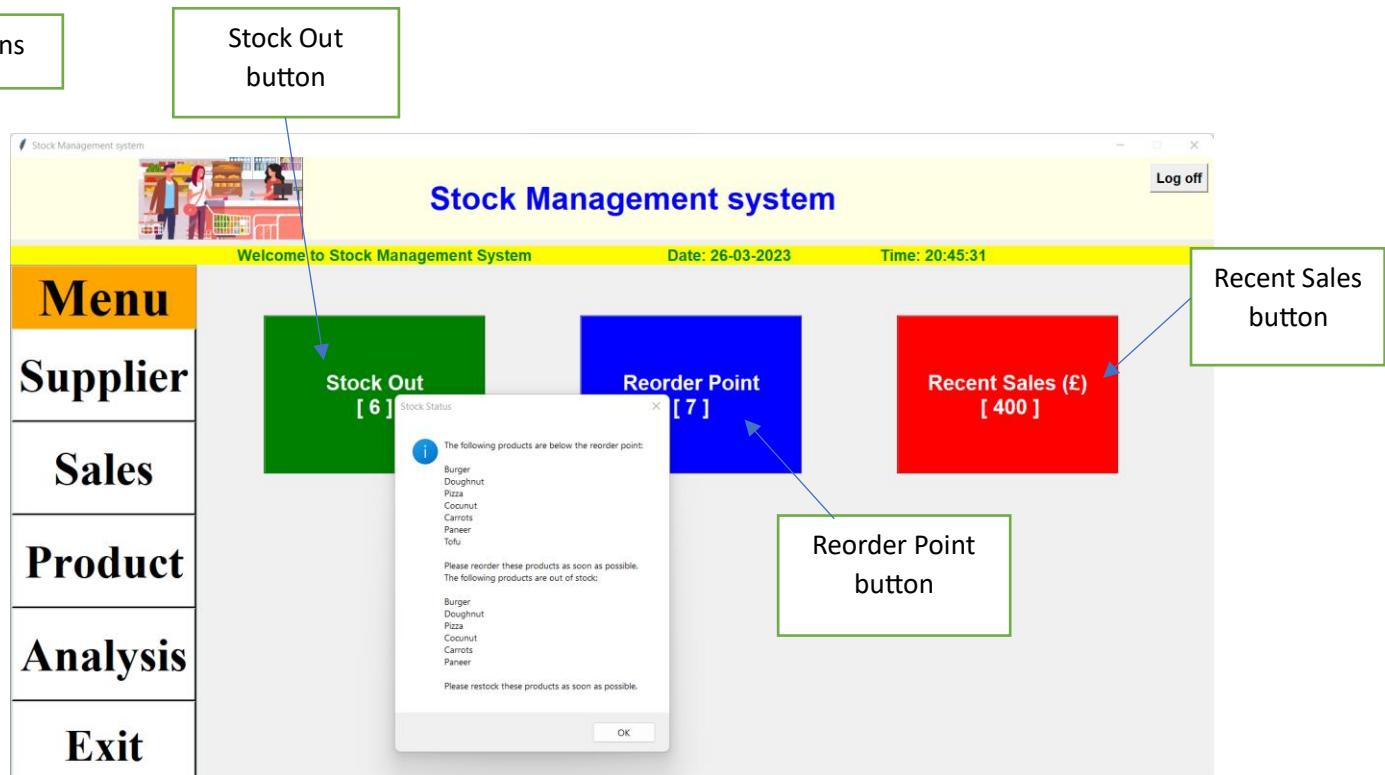
Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
	is allowed to reset it.	Click on 'Click Here' User enters the OTP received to allow the new password to be updated. User must enter the password again to confirm the new password	user can enter a new password.		window is shown that allows OTP code, password and the confirmed password to be entered

Test ID: 7

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
7.	GUI testing – When user logs in, they see the dashboard with menu icons on left and Stock Out, Reorder Point, and Yesterdays sales.	Enter username: User1 With a valid password to login.	Menu icons on the left for Supplier, Sales, Product, Inventory Adjustment, Analysis, and Exit are shown. Stock Out label shows the number of products currently stocked out. Reorder Point label shows the number of products which have reached Reorder Point and need new ordering. Sales label shows the total value of yesterday's sales in £ for all products.	S5, S6

Testing: Checking if OTP code matches the user's input

This cross-reference shows that when the user enters the correct OTP code, the dashboard window opens.



Here, the dashboard shows all the menu buttons and the Stock out and reorder point buttons show the number of products fulfilling the criteria.

The recent sales label shows the most recent total sales in terms of value (£).

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
7.	GUI testing – When user logs in, they see the dashboard with menu icons on left and Stock Out, Reorder Point, and Yesterdays sales.	Enter username: User1 With a valid password to login.	Menu icons on the left for Supplier, Sales, Product, Inventory Adjustment, Analysis, and Exit are shown. Stock Out label shows the number of products	S5, S6	Pass – Dashboard shows all the features such as the menu icons, reorder point and stock out buttons and the most

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
			<p>currently stocked out.</p> <p>Reorder Point label shows the number of products which have reached Reorder Point and need new ordering.</p> <p>Sales label shows the total value of yesterday's sales in £ for all products.</p>		recent sales label.

Test ID: 8

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
8.	Menu icons and labels are easy to read and have contrasting colours	Enter username: User1 With a valid password to login.	Menu icons and labels are easy to read and have contrasting colours	S6, S7

Dashboard

Here, I have cross-referenced my client's response to the questions I asked in the evaluation section. My client was able to identify the text in the menu buttons as the text was bold. My client could also identify the numbers shown on the reorder point and stock out buttons as the text shown in white was in contrast with the dark colours used as the background for the buttons.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
8.	Menu icons and labels are easy to read and have contrasting colours	Enter username: User1 With a valid password to login.	Menu icons and labels are easy to read and have contrasting colours	S6, S7	Pass – The menu icons are easy to read as text is in bold and the label and the buttons have contrasting colours to emphasize the text.

[Test ID: 9](#)

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
9.	Form fields allow searching of records easily	Enter username: User1 With a valid password to login. Click on the menu icon 'Supplier' and search for a supplier.	It should be easy to enter text in the fields.	S8

Testing: Search function

This cross-referenced test shows that it is easy to search for a supplier by selecting an option to choose the supplier from and then entering the text in the search bar.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
9.	Form fields allow searching of records easily	Enter username: User1 With a valid password to login.	It should be easy to enter text in the fields.	S8	Pass – The cross-referenced test shows that suppliers

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
		Click on the menu icon 'Supplier' and search for a supplier.			can be searched using the search feature which shows all the matching records that fit the user's criteria that they input.

Test ID: 10

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
10.	Menu buttons should navigate to different areas of the software	Enter username: User1 With a valid password to login. 1] Click on the menu icon 'Supplier'. 2] Click on the menu icon 'Sales' 3] Click on the menu icon 'Product' 4] Click on the menu icon 'Analysis' 5] Click on the menu icon 'Exit'	After logging in successfully, 1] Supplier screen is displayed which allows searching for a supplier 2] Sales screen is displayed where import function is available 3] Product screen is displayed which allows searching for a product. 4] Analysis screen is displayed. 5] User exits from the software.	S9



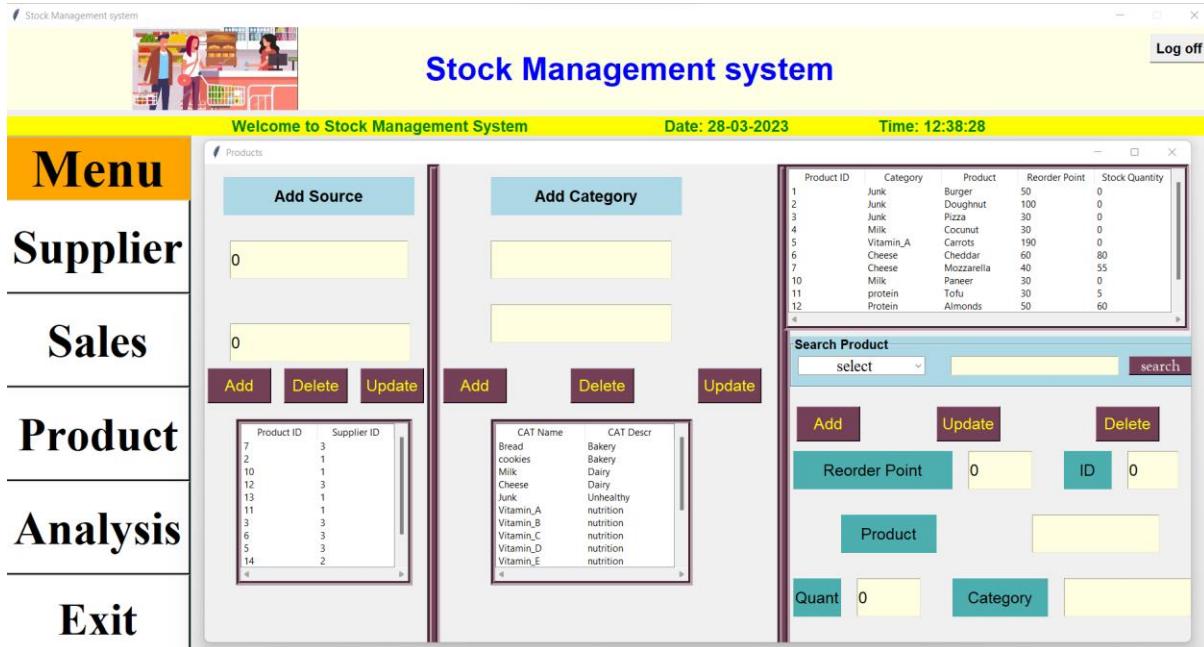
When supplier button is clicked:



When Sales button is clicked:



When Product button is clicked:



When analysis button is clicked:



When I click on the exit button: (it shows me the Pycharm application)



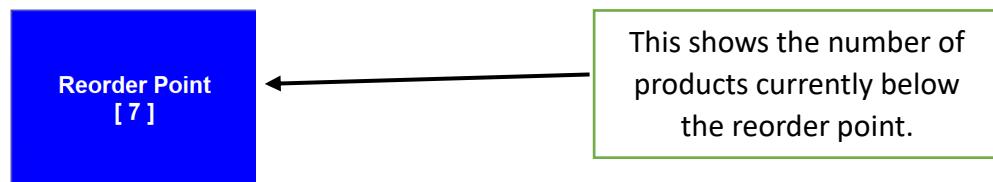
Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
10.	Menu buttons should navigate to different areas of the software	Enter username: User1 With a valid password to login. 1]Click on the menu icon 'Supplier'. 2]Click on the menu icon 'Sales' 3]Click on the menu icon 'Product' 4]Click on the menu icon 'Analysis' 5]Click on the menu icon 'Exit'	After logging in successfully, 1] Supplier screen is displayed which allows searching for a supplier 2] Sales screen is displayed where import function is available 3] Product screen is displayed which allows searching for a product. 4] Analysis screen is displayed. 5] User exits from the software.	S9	Pass – clicking on the menus in the dashboard show up a new window

[Test ID: 11](#)

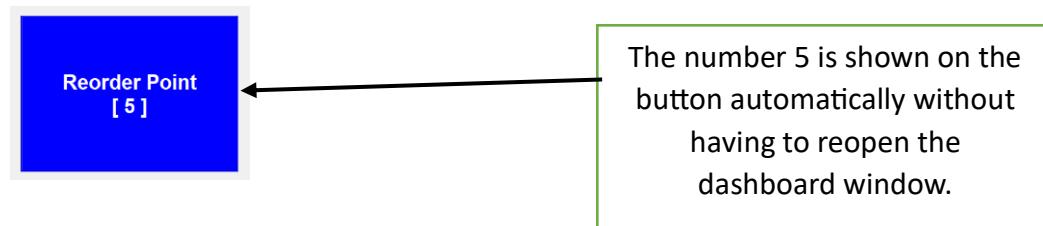
Test ID	Description	Test Data	Expected Results	Reference Success Criteria
11.	Reorder point on Dashboard providing count and further details. Ref: <u>Reorder point design</u>	1] Login to the application (Ensure there are some products which have inventory less than Reorder point). 2] On the Dashboard, the Reorder point label should show the count of products currently having	1] User logged into application. 2] The count of products displayed on dashboard is correct. 3] On clicking the count, it opens a new window showing the list of products. This should contain the Product, current	S10

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
		<p>inventory below Reorder point.</p> <p>3]Click on the count and it should open a new window showing the details of products</p> <p>4]Click on one of the products from the details view to populate the supplier details in the email panel below.</p> <p>5]For the product chosen in #4 above, User can enter any free text in the email panel and send the email to supplier.</p>	<p>inventory Qty, Supplier details.</p> <p>4]After clicking on the product in the details panel, automatically the email panel below should get populated.</p> <p>5]Enter additional text – e.g. Supply 2 cases and click on the 'Send' button to send email to supplier.</p>	

Testing: Checking if OTP code matches the user's input -This cross-referenced test shows that the user will successfully see the dashboard page once the login details and the OTP code are entered correctly.



If I make 2 of the products have a quantity greater than the reorder point then:



Testing: Treeview showing products and suppliers when product quantity < reorder point –

This test shows a treeview containing all the products where the current quantity in the store is less than the reorder point (from the product table).

Testing: Email being sent to suppliers – This test shows an email being sent to the suppliers by entering text into the message entry box along with the supplier's email address.

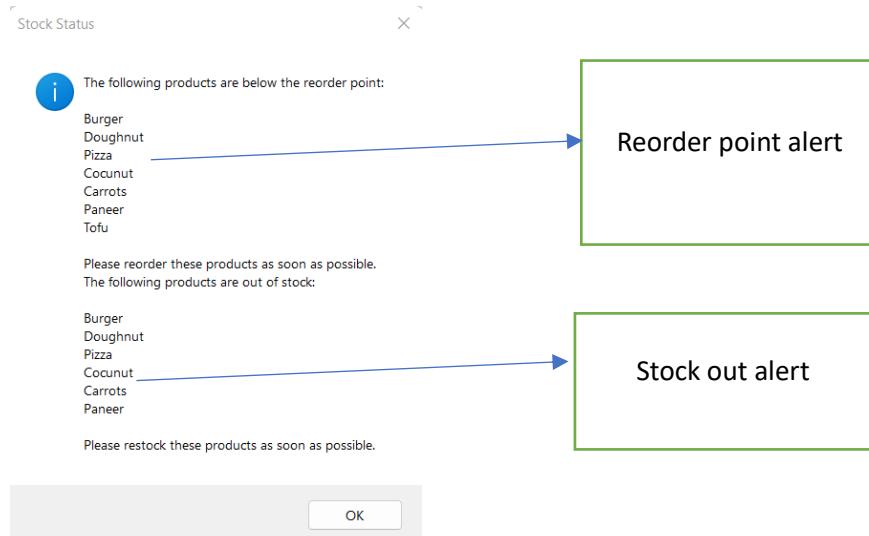
Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
11.	Reorder point on Dashboard providing count and further details. Ref: <u>Reorder point design</u>	<p>1]Login to the application (Ensure there are some products which have inventory less than Reorder point).</p> <p>2]On the Dashboard, the Reorder point label should show the count of products currently having inventory below Reorder point.</p> <p>3]Click on the count and it should open a new window showing the details of products</p> <p>4]Click on one of the products from the details view to populate the supplier details in the email panel below.</p> <p>5]For the product chosen in #4 above, User can enter any free text in the email panel</p>	<p>1] User logged into application.</p> <p>2]The count of products displayed on dashboard is correct.</p> <p>3]On clicking the count, it opens a new window showing the list of products. This should contain the Product, current inventory Qty, Supplier details.</p> <p>4]After clicking on the product in the details panel, automatically the email panel below should get populated.</p> <p>5]Enter additional text – e.g. Supply 2 cases and click on the 'Send' button to send email to supplier.</p>	S10	Pass – When successfully logged in, the dashboard shows the reorder point button which displays the number '7' meaning that 7 products are below the reorder point.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
		and send the email to supplier.			

Test ID: 12

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
12.	Reorder point alert Ref: <u>Reorder point design</u>	Test data to have inventory below Reorder point. When user logs in, an alert is shown for every product that is under Reorder point. This alert can be on any page (not restricted to dashboard).	Alert correctly shows the products that are having inventory below Reorder point. User needs to click on the 'X' to close the alert window. Alert should remain until it is closed by user.	S11





The message is shown as soon as the dashboard is open. The user cannot go onto any other window without clicking the close sign on the message.

The alert shows all the products that are below the reorder point.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
12.	Reorder point alert Ref: <u>Reorder point design</u>	Test data to have inventory below Reorder point. When user logs in, an alert is shown for every product that is under Reorder point. This alert can be on any page (not restricted to dashboard).	Alert correctly shows the products that are having inventory below Reorder point. User needs to click on the 'X' to close the alert window. Alert should remain until it is closed by user.	S11	Mostly passed – alert doesn't show up on every window as the user must close the alert to access the other windows. Name of products below reorder point are shown.

Test ID: 13

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
13.	Stock-out on Dashboard providing count and further details. Ref: <u>Stock-out</u> <u>Dashboard Button design</u>	<p>1]Login to the application (Ensure there are some products which have inventory as zero).</p> <p>2]On the Dashboard, the Stock-out label should show the count of products currently having zero inventory.</p> <p>3]Click on the count and it should open a new window showing the details of products</p> <p>4]Click on one of the products from the details view to populate the supplier details in the email panel below.</p> <p>5]For the product chosen in #4 above, User can enter any free text in the email panel and send the email to supplier.</p>	<p>1] User logged into application.</p> <p>2]The count of products displayed on dashboard is correct.</p> <p>3]On clicking the count, it opens a new window showing the list of products. This should contain the Product, Supplier details.</p> <p>4]After clicking on the product in the details panel, automatically the email panel below should get populated.</p> <p>5]Enter additional text – e.g. Supply 2 cases and click on the 'Send' button to send email to supplier.</p>	S12

Testing: Checking if OTP code matches the user's input – This test shows the dashboard window being opened once the OTP code is entered correctly as part of the 2-factor authentication.



This Stock Out button shows the number 6 meaning that 6 products have a current quantity of 0 in the store.

If I add quantity to 3 of the stocked-out products, then the button shows this:



Testing: Treeview showing products and linked suppliers with 0 stock Quantity – This cross-referenced test shows the products with 0 quantity in the store.

Testing: Email should be sent to supplier – This cross-referenced test shows email being sent to suppliers.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
13.	Stock-out on Dashboard providing count and further details. Ref: <u>Stock-out Dashboard Button design</u>	1]Login to the application (Ensure there are some products which have inventory as zero). 2]On the Dashboard, the Stock-out label should show the count of products currently having zero inventory. 3]Click on the count and it should open a new window showing the details of products 4]Click on one of the products from the details	1] User logged into application. 2]The count of products displayed on dashboard is correct. 3]On clicking the count, it opens a new window showing the list of products. This should contain the Product, Supplier details. 4]After clicking on the product in the details panel, automatically the email panel below should get populated.	S12	Pass - user logs in successfully which then automatically shows the dashboard window. The stock out button shows the number of products out of stock. When you click on the button, the treeview is shown showing the product details as well as the linked suppliers as one whole record.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
		view to populate the supplier details in the email panel below. 5]For the product chosen in #4 above, User can enter any free text in the email panel and send the email to supplier.	5]Enter additional text – e.g. Supply 2 cases and click on the 'Send' button to send email to supplier.		

Test ID: 14

Test Id	Description	Test Data	Expected Results	Reference Success Criteria
14.	Stock-out alert	Test data to have inventory as zero for some products. When user logs in, an alert is shown for every product that has zero inventory. This alert can be on any page (not restricted to dashboard).	Alert correctly shows the products that are stocked-out. User needs to click on the 'X' to close the alert window. Alert should remain until it is closed by user.	S13

Test ID: 12 – This post-development test explains that the alert message must be closed first before other parts of the program can be accessed.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
14.	Stock-out alert	Test data to have inventory	Alert correctly shows the products that	S13	Mostly Pass – The alert shows the

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
		as zero for some products. When user logs in, an alert is shown for every product that has zero inventory. This alert can be on any page (not restricted to dashboard).	are stocked-out. User needs to click on the 'X' to close the alert window. Alert should remain until it is closed by user.		products that are out of stock. The alert will remain on the dashboard window until the user clicks on the close button. Otherwise, other buttons cannot be clicked.

Test ID: 15

Test Id	Description	Test Data	Expected Results	Reference Success Criteria
15.	Import Sales data Ref: <u>Sales design</u>	Create a file with sales data from previous day. File name is 'sales' and contains Date, Item, Qty. Login to system and click on Import.	Records from file are imported and visible in the Tree view after import.	S14

Testing: Importing CSV file into the Sales table – This test shows that the treeview will show sales records for the last 7 days.

A	B	C	D	E
1	2023-03-28	5	100	500
2	2023-03-28	11	10	50
3				

Sales data from yesterday.

Date	Product ID	Sold	Value £
2023-03-23	1	30	150
2023-03-23	5	30	150
2023-03-24	1	40	200
2023-03-24	5	40	200
2023-03-28	5	100	500
2023-03-28	11	10	50

Here, the treeview shows the sales record for the last 7 days. As you can see, the records from the sales CSV file have been added into the treeview.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
15.	Import Sales data Ref: <u>Sales design</u>	Create a file with sales data from previous day. File name is 'sales' and contains Date, Item, Qty. Login to system and click on Import.	Records from file are imported and visible in the Tree view after import.	S14	Pass – the sales file is successfully imported with yesterday's sales records and is visible in the treeview

Test ID: 16

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
16.	Import Sales data Ref: <u>Sales design</u>	Create a file with sales data from previous day. This is an Erroneous test – Name the File as 'sales_'. Login to system and click on Import.	Due to the file name having an underscore in the name, it should not be imported and error message provided to user.	S15

Sales Data				
Date	Product ID	Sold	Value £	
2023-03-23	1	30	150	
2023-03-23	5	30	150	
2023-03-24	1	40	200	
2023-03-24	5	40	200	
2023-03-28	5	200	1000	
2023-03-28	11	20	100	

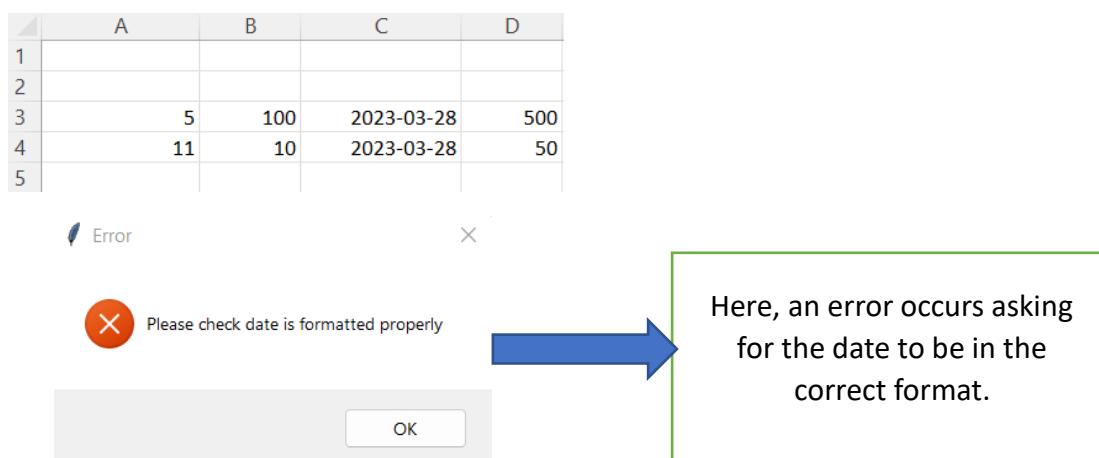
Here, the treeview doesn't show the sales records for today's date. Instead, it adds the sales again for yesterday's records from the correctly spelt file name.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
16.	Import Sales data Ref: <u>Sales design</u>	Create a file with sales data from previous day. This is an Erroneous test – Name the File as 'sales_'. Login to system and click on Import.	Due to the file name having an underscore in the name, it should not be imported and error message provided to user.	S15	Pass – program only imports the file with the same name as mentioned in the code itself.

Test ID: 17

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
17.	Import Sales data Ref: <u>Sales Design</u>	Create a file with sales data from previous day. This is	File should not be imported and error	S15

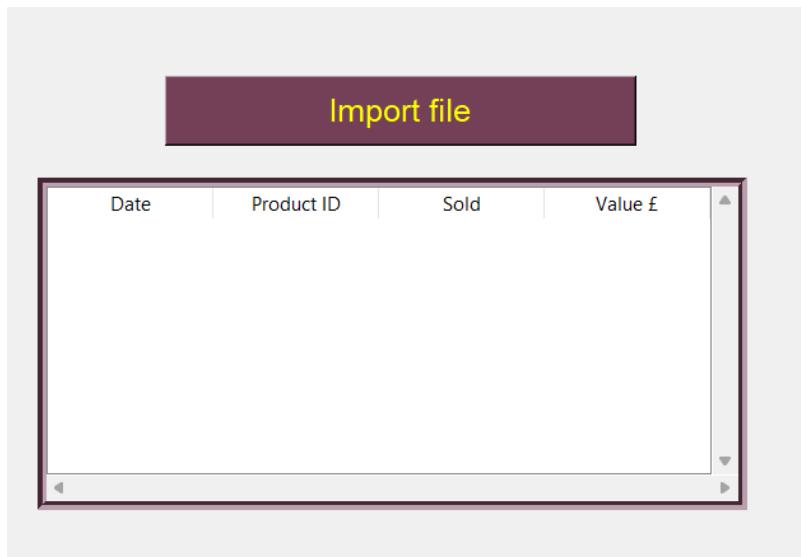
Test ID	Description	Test Data	Expected Results	Reference Success Criteria
		<p>an Erroneous test. File name is 'sales' and contains Item, Qty, Date, Value (expected format was Date, Item, Qty, Value).</p> <p>Login to system and click on Import.</p>	message provided to user.	



Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
17.	Import Sales data Ref: <u>Sales Design</u>	<p>Create a file with sales data from previous day. This is an Erroneous test.</p> <p>File name is 'sales' and contains Item, Qty, Date, Value (expected format was Date, Item, Qty, Value).</p> <p>Login to system and click on Import.</p>	File should not be imported and error message provided to user.	S15	Pass – program shows error as the date format is not correct

Test ID: 18

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
18.	Import Sales data Ref: <u>Sales design</u>	<p>Create a file with sales data from previous day. File name is 'Sales' and contains Date, Item, Qty.</p> <p>Login to system and click on Import.</p>	<p>Check the time required to complete the import. It should not be more than 1 minute.</p>	S16



Date	Product ID	Sold	Value £
2023-03-24	1	40	200
2023-03-24	5	40	200
2023-03-28	5	200	1000
2023-03-28	11	20	100
2023-03-29	5	100	500
2023-03-29	11	10	50

When the import button is pressed, the treeview is automatically updated.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
18.	Import Sales data Ref: <u>Sales design</u>	Create a file with sales data from previous day. File name is 'Sales' and contains Date, Item, Qty. Login to system and click on Import.	Check the time required to complete the import. It should not be more than 1 minute.	S16	Pass - Importing the CSV file is done automatically and so it takes less than 1 minute.

Test ID: 19

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
19.	Import Sales data Ref: <u>Sales design</u>	Login to system and click on the 'Graph by number if items sold'	Graph is displayed with last 7 days of sales data in Qty	S17

Testing: Sales Quantity Graph for last 7 days

This test shows the graph for the items sold for the last seven days. However, this test only shows a few of the bars.

Sales window

This cross-referenced test shows more bars for the Sales quantity graph.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
19.	Import Sales data Ref: <u>Sales design</u>	Login to system and click on the 'Graph by number if items sold'	Graph is displayed with last 7 days of sales data in Qty	S17	Pass - Graph produced in terms of quantity sold for the last 7 days

Test ID: 20

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
20.	Import Sales data Ref: <u>Sales design</u>	Login to system and click on the 'Graph by value'	Graph is displayed with last 7 days of sales value	S17

Testing: Value graph (£) for the last seven days

This test shows the graph in terms of value made (£) for the last 7 days.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
20.	Import Sales data Ref: <u>Sales design</u>	Login to system and click on the 'Graph by value'	Graph is displayed with last 7 days of sales value	S17	Graph by value shown for the last 7 days

Test ID: 21

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
21.	Analysis	Login to system and 1] click on 'Analysis' tab from the menu. 2] Close the top five graph 3] Close the least five graph	1] The top five items are displayed in a graph 2] Next graph of least five items is displayed 3] Tree view showing both top and least items displayed.	S18

Testing: Graph showing top 5 products – This test shows the graph for the top 5 performing products

Testing: Shows top 5 products in descending order in treeview – Treeview showing top 5 products

Testing: Graph to show worst 5 products – Graph showing worst 5 products

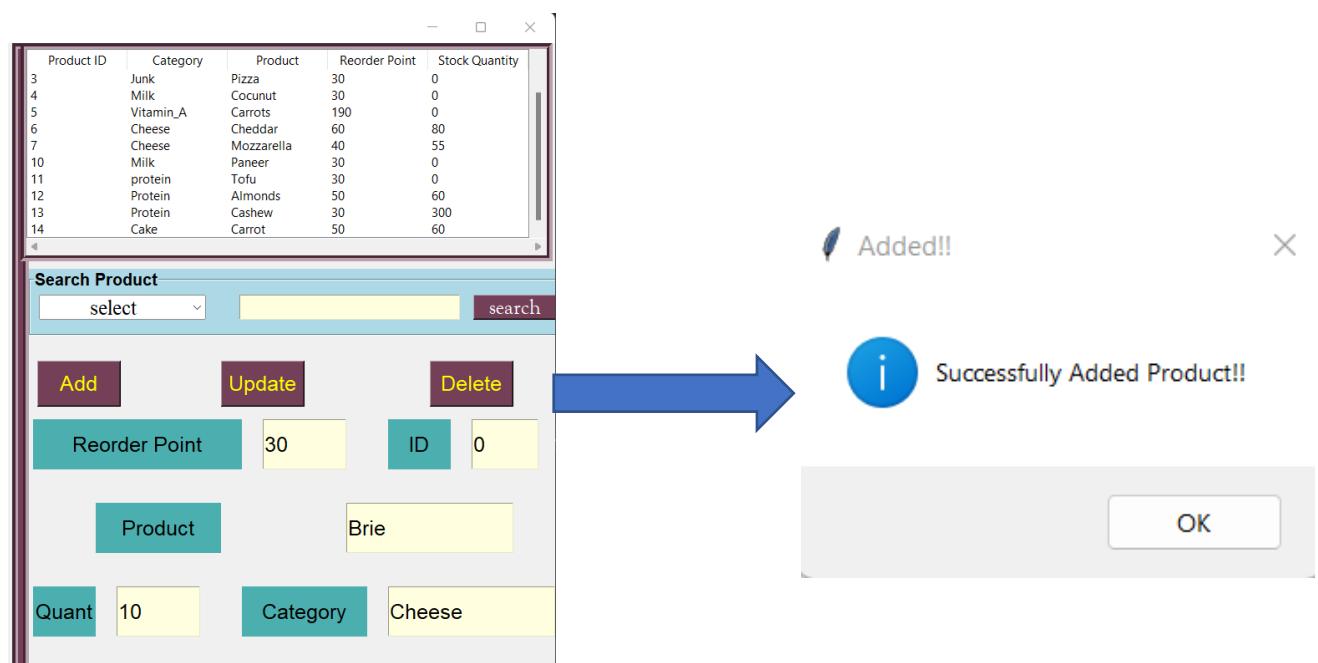
Testing: Treeview showing worst 5 products in ascending order – Treeview showing worst 5 products

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
21.	Analysis	Login to system and 1] click on 'Analysis' tab from the menu. 2] Close the top five graph 3] Close the least five graph	1] The top five items are displayed in a graph 2] Next graph of least five items is displayed 3] Tree view showing both top and least items displayed.	S18	Pass - The graphs and treeviews are displayed. Treeview showing the worst and top 5 products are shown as soon as the user goes to the analysis window. Graphs are shown by clicking on their respective buttons.

[Test ID: 22](#)

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
22.	Add new Product	Login to system and 1] click on 'Product' tab from the menu. 2] Enter category name 3] Enter product name	Product Id is created automatically, and new Product is stored with its category, inventory and reorder qty.	S19

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
		4] Enter any existing Inventory Qty (if any) 5] Enter Reorder point		



Product ID	Category	Product	Reorder Point	Stock Quantity
4	Milk	Cocunut	30	0
5	Vitamin_A	Carrots	190	0
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	55
10	Milk	Paneer	30	0
11	protein	Tofu	30	0
12	Protein	Almonds	50	60
13	Protein	Cashew	30	300
14	Cake	Carrot	50	60
29	Cheese	Brie	30	10

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
22.	Add new Product	Login to system and	Product Id is created	S19	Pass - New product

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
		<p>1] click on 'Product' tab from the menu.</p> <p>2] Enter category name</p> <p>3] Enter product name</p> <p>4] Enter any existing Inventory Qty (if any)</p> <p>5] Enter Reorder point</p>	automatically, and new Product is stored with its category, inventory and reorder qty.		record is created with the Product ID being automatically created. The user inputs everything else

[Test ID: 23](#)

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
23.	Add new Supplier	<p>Login to system and</p> <p>1] click on 'Supplier' tab from the menu.</p> <p>2] Enter Supplier Id</p> <p>3] Enter Supplier name</p> <p>4] Enter Supplier Email</p>	Supplier is created	S20

Supplier details		Supplier ID	Name	Email
Supplier Name	Varun	1	Sam	rushilovesgames28@gmail.com
		2	Rushabh	rushabh.is.cool28@gmail.com
		3	Bob	wateraid28@gmail.com
		4	Nijan	nijankannathasan@gmail.com
		5	Test	Test@gmail.com
		6	Rayyan	elhambre049@gmail.com
		15	Trial	Trial@gmail.com
Supplier ID	16			
Supplier Email	Trial_test@gmail.com			

When the add button is pressed:

The screenshot shows a modal dialog box with a blue header bar containing the text "Added!!" and a close button "X". Below the header, there is a blue circular icon with a white "i" and the text "Successfully Added: Varun". At the bottom of the dialog is an "OK" button. To the right of the dialog is a table titled "Supplier ID" with columns for "Supplier ID", "Name", and "Email". The table contains 16 rows of data, with the last row being "Varun" (Supplier ID 16). An arrow points from the "Varun" entry in the table to the "Successfully Added: Varun" message in the dialog.

Supplier ID	Name	Email
1	Sam	rushlovesgames28@gmail.com
2	Rushabh	rushabh.is.cool28@gmail.com
3	Bob	wateraid2.8@gmail.com
4	Nijan	nijankannathasan@gmail.com
5	Test	Test@gmail.com
6	Rayyan	elhambre049@gmail.com
15	Trial	Trial@gmail.com
16	Varun	Trial_test@gmail.com

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
23.	Add new Supplier	Login to system and 1] click on 'Supplier' tab from the menu. 2] Enter Supplier Id 3] Enter Supplier name 4] Enter Supplier Email	Supplier is created	S20	Pass - New supplier is created as shown on the treeview above

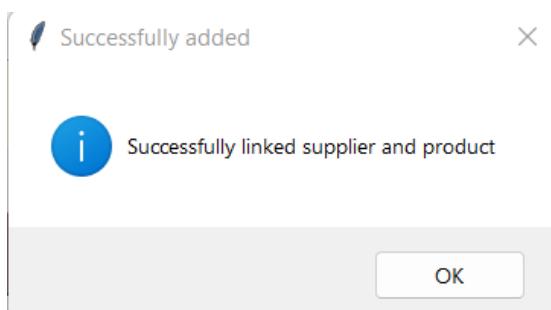
Test ID: 24

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
24.	Attaching Product to Supplier	Login to system and 1] click on 'Product' tab from the menu. 2] In the 'Source' section, enter the Product Id and Supplier Id. Click on 'Add'.	Product ID and Supplier ID are linked and visible in the Tree view	S21

Add Source

29	←	Product ID
16	←	Supplier ID

Add Delete Update



Product ID	Supplier ID
12	3
13	1
11	1
3	3
6	3
5	3
14	2
4	2
1	2
29	16

Treeview showing the supplier and product ID being added

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
24.	Attaching Product to Supplier	Login to system and 1] click on 'Product' tab from the menu.	Product ID and Supplier ID are linked and visible in the Tree view	S21	Pass – Product ID and Supplier ID are successfully

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
		2] In the 'Source' section, enter the Product Id and Supplier Id. Click on 'Add'.			added together in the Sourcing table. This allows the supplier and the product to be linked together

Test ID: 25

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
25.	Add Category	1] click on 'Product' tab from the menu. 2] In the 'Category' section, enter the Category Name and Category Description. Click on 'Add'.	Category is added and visible in the Tree view	S22

Testing: Add function in Category table – This cross referenced test shows the Category name and description being added successfully to the category table for the normal and boundary data.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
25.	Add Category	1] click on 'Product' tab from the menu. 2] In the 'Category' section, enter the Category Name and	Category is added and visible in the Tree view	S22	Pass – Category name and description added successfully

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
		Category Description. Click on 'Add'.			

Test ID: 26

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
26.	Update Category Description	1] click on 'Product' tab from the menu. 2]In the 'Category' section, from the TreeView, select the specific category that needs update. 3] The selected values will be visible in the value box above. Enter the new description and click on 'Update'	Category description is updated and visible in the Tree view.	S23

Testing: Update function in Category table – This cross-referenced test allows the category description to be updated.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
26.	Update Category Description	1] click on 'Product' tab from the menu. 2]In the 'Category' section, from the TreeView, select the specific category that needs update.	Category description is updated and visible in the Tree view.	S23	Pass – Category description updated successfully

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
		3] The selected values will be visible in the value box above. Enter the new description and click on 'Update'			

Test ID: 27

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
27.	User should be able to update following attributes of item: Item description, Reorder point, Category	1] click on 'Product' tab from the menu. 2] From the Tree view showing list of products, select the specific product that needs update. 3] In the section below, the values will be populated. Enter a new reorder point, Product description, Category click on 'Update'	For a specific product, the following attributes are updated: <ul style="list-style-type: none">• Reorder Point• Product description• Category	S24

Product ID	Category	Product	Reorder Point	Stock Quantity
1	Junk	Burger	50	0
2	Junk	Doughnut	100	0
3	Junk	Pizza	30	0
4	Milk	Cocunut	30	0
5	Vitamin_A	Carrots	190	0
6	Cheese	Cheddar	60	80
7	Cheese	Mozzarella	40	55
10	Milk	Paneer	30	0
11	protein	Tofu	30	0
12	Protein	Almonds	50	60

Search Product

select search

Add Update Delete

Reorder Point ID

Product

Quant Category

Selected record in treeview

Reorder point changed to 30

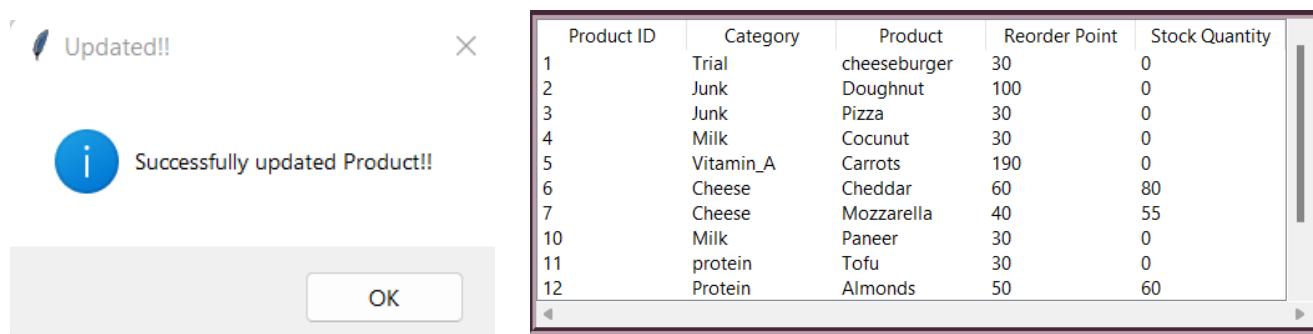
Reorder Point ID

Product

Quant Category

Changed product name to cheeseburger

Category changed to 'Trial'



Treeview shows the product record being changed successfully. The reorder point is changed to 30, the category name is changed to 'Trial' and the product name /description is changed to 'cheeseburger'.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
27.	User should be able to update following attributes of item: Item description, Reorder point, Category	1] click on 'Product' tab from the menu. 2]From the Tree view showing list of products, select the specific product that needs update. 3] In the section below, the values will be populated. Enter a new reorder point, Product description, Category click on 'Update'	For a specific product, the following attributes are updated: <ul style="list-style-type: none"> • Reorder Point • Product description • Category 	S24	Pass – Product name (description), reorder point and category are successfully updated

Test ID: 28

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
28.	Delete Product	1] click on 'Product' tab from the menu. 2] From the Tree view showing list of products, select the specific product that needs delete. 3] click on 'Delete'	Product is deleted and no longer visible in Tree view. <ul style="list-style-type: none"> • Before Deletion a confirmation message is displayed to user. 	S25

Testing: Deleting a selected record – This test shows a record being selected from the treeview and then deleted. A confirmation message is shown to ensure the user doesn't accidentally delete a record just by pressing the delete button.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
28.	Delete Product	1] click on 'Product' tab from the menu. 2] From the Tree view showing list of products, select the specific product that needs delete. 3] click on 'Delete'	Product is deleted and no longer visible in Tree view. <ul style="list-style-type: none"> • Before Deletion a confirmation message is displayed to user. 	S25	Pass – Product record is deleted successfully when selected from treeview. A confirmation message is shown before record is deleted.

Test ID: 29

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
29.	Update Supplier Name or Email	Login to system and 1] click on 'Supplier' tab from the menu. 2] From the Tree view showing list of Suppliers, select the specific Supplier that needs update. 3] Enter Supplier name 4] Enter Supplier Email 5] Click on 'Update'	• Supplier Name and Email are updated.	S26

Testing: modify function – This iterative test is used to update the supplier details – name and email address.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
29.	Update Supplier Name or Email	Login to system and 1] click on 'Supplier' tab from the menu. 2] From the Tree view showing list of Suppliers, select the specific Supplier that needs update. 3] Enter Supplier name 4] Enter Supplier Email 5] Click on 'Update'	• Supplier Name and Email are updated.	S26	Pass – Supplier details are updated. Supplier ID is not updated as it is a primary key.

Test ID: 30

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
30.	Delete Supplier	Login to system and 1] click on 'Supplier' tab from the menu. 2] From the Tree view showing list of Suppliers, select the specific Supplier that needs delete. 3] Click on 'Delete'	Supplier is deleted and no longer visible in Tree view. • Before Deletion a confirmation message is displayed to user.	S27

Testing: Delete function – This test shows the selected supplier record being deleted.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
30.	Delete Supplier	Login to system and 1] click on 'Supplier' tab from the menu. 2] From the Tree view showing list of Suppliers, select the specific Supplier that needs delete. 3] Click on 'Delete'	Supplier is deleted and no longer visible in Tree view. • Before Deletion a confirmation message is displayed to user.	S27	Pass – Supplier record selected is deleted from the supplier table. A confirmation message is shown to ensure that the selected record is deleted and not anything else.

Test ID: 31

Test ID	Description	Test Data	Expected Results	Reference Success Criteria
31.	Update Inventory of a product	<p>1] click on 'Product' tab from the menu.</p> <p>2] From the Tree view showing list of products, select the specific product that needs update or viewing.</p> <p>3] In the section below, the values will be populated. Enter new Inventory Qty and click on 'Update'</p>	<ul style="list-style-type: none"> For a specific product, the Inventory is visible and can be updated. 	S28, S31

Testing: Updating the quantity and reorder point of Product table – In this test, the quantity of the product can be updated manually.

Test ID	Description	Test Data	Expected Results	Reference Success Criteria	Actual results
31.	Update Inventory of a product	<p>1] click on 'Product' tab from the menu.</p> <p>2] From the Tree view showing list of products, select the specific product that needs update or viewing.</p> <p>3] In the section below, the values will be populated. Enter new Inventory Qty and click on 'Update'</p>	<ul style="list-style-type: none"> For a specific product, the Inventory is visible and can be updated. 	S28, S31	Pass – quantity of product can be updated

Evaluation of success criteria using Post development testing

<u>Success criteria no.</u>	<u>Success criteria</u>	<u>Rating out of 10 by client</u>	<u>Feedback with evidence</u>
1.	<p>Client should be able to type in their username and password to access the system.</p> <p>Once the user enters their username and password correctly, they should get a code sent to their email that they would have to enter in order to finally gain access to the system.</p>	10	<p>Test ID 1 shows the OTP code being received within a few seconds when the user enters the username and password correctly.</p> <p><u>Test ID: 1</u></p> <p>Test ID 2 shows cross-referenced iterative test that shows the dashboard window being opened when correct OTP code has been entered.</p> <p><u>Test ID: 2:</u></p>
2.	The user should not be logged in to the system if he types an incorrect password or username.	10	<p>Test ID 3 shows client entering the wrong username, causing the program to show an error message.</p> <p><u>Test ID: 3</u></p> <p>Test ID 4 shows the same error message when user enters wrong password.</p> <p><u>Test ID: 4</u></p> <p>Test ID 5 shows cross-referenced iterative test that first shows an error message occurring when OTP code is entered wrong.</p> <p><u>Test ID: 5</u></p>
3.	There should be a forget password section so that the user can still login	8	Test ID 6 shows a new window when the user has forgotten their password. My

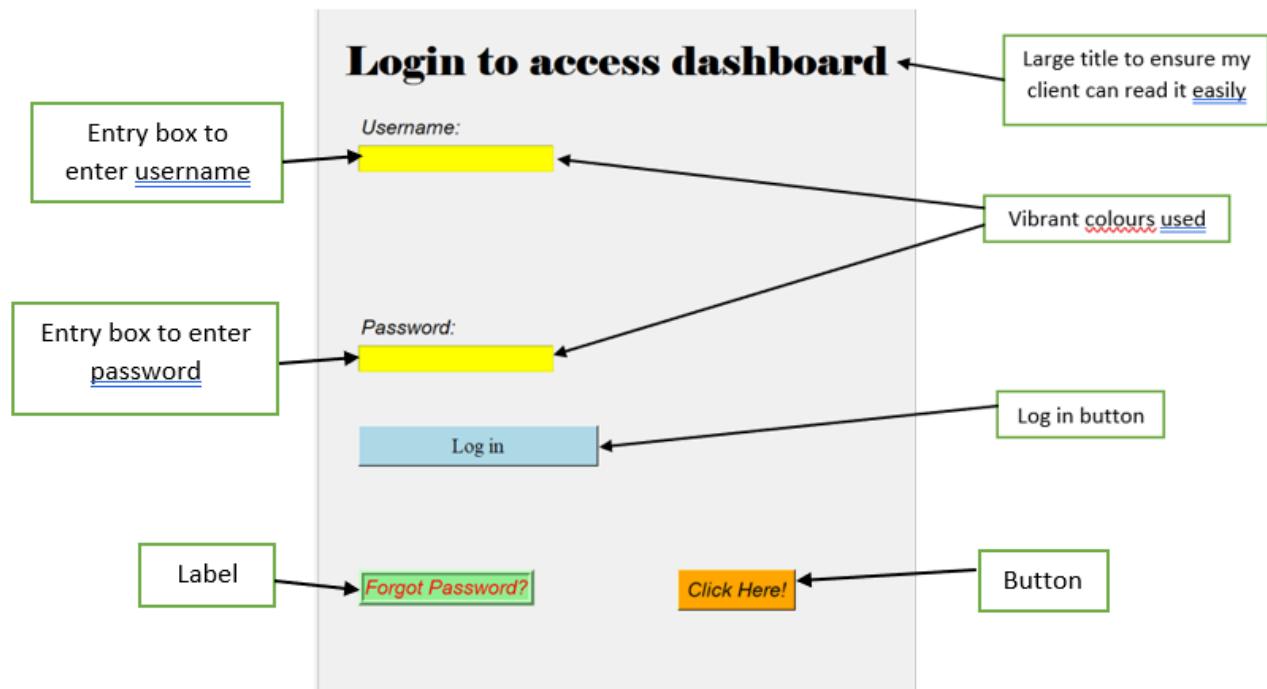
			in decimals instead of a whole number.
33.	Forecast sales in terms of value made (£) for next day and for next month for a given product.	10	Last page of Client response shows screenshots of 2 message boxes showing the predicted sales for tomorrow and for next month in terms of value made (£). Decimal places are acceptable as it is the value made which doesn't have to be a whole number.

Usability features

Login – first window

Here, I have taken a screenshot of the client's response about how user-friendly the GUI is.

Final outcome:



		<p>border was a bit too fancy.</p> <p><u>Login</u></p> <p>In the Product window, a lot of Tkinter widgets have been used. My client liked the idea of splitting the window using frames and showing treeviews instead of going back to the database application.</p> <p>However, my client said that the labels in the product section (right side of the product window) could have been of equal size. Also, there were no labels in the category and sourcing section which would make it confusing to my client about what to enter in the entry boxes.</p> <p><u>Product window</u></p> <p>The other windows were fine.</p> <p>My client said that the entry boxes for sending messages (Reorder point and Stock-out windows) could have been designed so that they start a new line instead of the message being one big line.</p> <p><u>Reorder-point</u></p>
--	--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

			<p>The sales window looked really nice as the graphs were equally spaced making it easier for my client to read. However, the exact sales data wasn't shown.</p> <p><u>Sales window</u></p> <p>With regards to the analysis window, my client had no problems with the GUI.</p> <p>Overall, there was GUI in every window, however some of the windows need to be a bit more organised.</p>
6.	The menu icons should be big enough for my client to see clearly.	10	<p>Test ID 7 dashboard screenshot shows menu icons in big and bold text with contrasting colours of black text and white background.</p> <p><u>Test ID: 7</u></p> <p>Test ID 8 cross-references the client's response to the questions I had set before the post development testing. Client said that the menu buttons are large and are of the same dimensions which makes the entire menu section look uniform and organised.</p> <p><u>Test ID: 8</u></p>
7.	There should be contrasting colours in	10	

	the text which will be used to make the dashboard look lively.		Test ID 8 cross-referenced dashboard screenshot shows the reorder point and stock-out buttons and the recent sales label in white text with contrasting dark colours. <u>Test ID: 8</u> Same comment about the menus as explained above. <u>Test ID: 7</u>
8.	The form fields should be big enough to type or search for a food product.	10	Test ID 9 cross-referenced search function shows that the user can with relative ease enter the data into the form fields. <u>Test ID: 9</u>
9.	GUI should have buttons that go on to different sections such as the analysis menu displaying the analysis of the food products etc.	10	Test ID 10 shows the screenshots of all the windows when the menu buttons are clicked. Each new window is on top of the dashboard window such that the menu and the title of the dashboard can be seen, but the recent sales label and the reorder point and stock out buttons cannot be seen. <u>Test ID: 10</u>
10.	If any of the item's inventory falls below a reorder point, then the reorder point button should be	10	Test ID 11 shows a screenshot of the reorder point button that displays the number of items

	raised in the Dashboard. This button when clicked should show the items that are low in stock. The button should be big enough so that client can read it.		below the reorder point. The 2 nd cross-referenced test ("Email being sent to suppliers") shows the reorder point window containing the products below the reorder point in a treeview. <u>Test ID: 11</u>
11.	The alert should remain there until client clicks on the close button. This would imply that it has been read.	10	Test ID 12 shows that the alert appears as soon as the dashboard is open. Other parts of the program are inaccessible if the user doesn't click the close button. <u>Test ID: 12</u>
12.	If any of the items inventory becomes zero, then the stock out button should be raised in the Dashboard. This button when clicked should show the items that are out of stock. The button should be big enough so that client can read it.	10	Test ID 13 shows a screenshot of the stock-out button. The 2 nd cross-referenced test shows the treeview containing the products with 0 stock quantity in the store. <u>Test ID: 13</u>
13.	The alert should remain there until client clicks on the close button. This would imply that it has been read.	10	Same as Success Criteria 11. <u>Test ID: 14</u> – This test cross-references Test ID: 12 as the reorder point and stock-out together form one big alert.
14.	User to have the ability to import sales files which gets	8	When asking a set of questions to my client before the post development testing,

	imported into the database.		<p>my client said that sometimes the program shows an error stating that the date is in the wrong format, though it is in the right format. My client, however, has fixed this issue by converting the date to the 'YYYY-MM-DD' format even if it already exists like that.</p> <p>Test ID 15 shows the sales file being imported into the system. The 2 sales records are then visible in the treeview meaning that the file has been successfully imported.</p> <p><u>Test ID: 15</u></p>
15.	The sales file should have the same structure as database.	10	<p>The sales file has the same name and the same order of columns as the sales table.</p> <p>Test ID 16 shows the sales file not being imported for a differently named CSV file.</p> <p><u>Test ID: 16</u></p> <p>Test ID 17 shows what happens if the fields in the sales csv file are switched around. An error message occurs saying that the date is not in the right format and hence the file cannot be imported.</p>

			<u>Test ID: 17</u> Overall, this objective has been met.
16.	It shouldn't take too long to import the file.	10	Provided the date is in the correct format, the file should be imported immediately as shown in Test ID 18. <u>Test ID: 18</u>
17.	User should be able to view the sales data in Qty and Value in a bar chart for last 7 days	9	The graphs for the last 7 days are produced successfully as shown by my client before the post development testing. <u>Test ID: 19</u> <u>Sales window</u> However, my client couldn't figure out the exact value of the sales (for both quantity and value made) as it doesn't show the y value when the cursor is placed on top of the bar.
18.	User should be able to find the top 5 least and most popular items in terms of Qty. This should be displayed in the form of a leaderboard that gets updated over time. The information should be presented in form of bar chart.	10	Test ID 21 cross-references some of the tests that show a graph showing the top 5 and worst 5 products. There is also a treeview (leaderboard) that shows the top 5 items (descending order) and worst 5 items (ascending order). The treeview shows the name of the Products so that the client

			doesn't have to find the name of the Product from the data stored in the product table. <u>Test ID: 21</u>
19.	User should be able to enter new items easily using the User Interface.	10	Test ID 22 shows screenshots of client entering a new product record by filling out the entry boxes of the product section in the product window. Once my client has clicked on the add button, the new record is shown on the product treeview. <u>Test ID: 22</u>
20.	User should be able to add new supplier easily using the User interface	10	Test ID 23 show screenshots of client entering a new supplier record by filling out the entry boxes of the supplier window. Once my client has clicked the add button, the new record is shown on the supplier treeview. <u>Test ID: 23</u>
21.	User should be able to attach the supplier to item	10	Test ID 24 shows an existing Product and Supplier ID being added to the Sourcing table and treeview. When the add button is clicked, the record appears on the treeview. <u>Test ID: 24</u>
22.	User should be able to add a category for each item	10	Test ID 25 cross-references a test that shows category name and description added successfully to the

			category table and treeview when the data type is not an integer. <u>Test ID: 25</u>
23.	User should be able to update the category description	10	Test ID 26 cross references another test that shows category description being updated successfully for the normal and boundary data. <u>Test ID: 26</u>
24.	User should be able to update following attributes of item: Item name/ description, Reorder point, Category	10	Test ID 27 shows screenshots of the client changing the name of the product, the reorder point and the category name. When the client pressed the update button, the changes were reflected successfully in the current record. <u>Test ID: 27</u>
25.	User should be able to delete an item. There should be a message to confirm deletion. This action should delete data from all relevant tables.	10	Test ID 28 cross-references an iterative test that shows a record being deleted from the product table. The user selects the record and then when the user presses the delete button, a confirmation message is shown to ensure that the user wants to delete that certain record. If yes is clicked, then the record selected is deleted. <u>Test ID: 28</u>

26.	User should be able to update following attributes of Supplier: Name, email	10	Test ID 29 cross-references an iterative test that shows the supplier details being updated. The program doesn't allow the name to be of an integer type. <u>Test ID: 29</u>
27.	User should be able to delete a supplier. There should be a message to confirm deletion. This action should delete data from all relevant tables.	10	Test ID 30 cross-references an iterative test to delete a supplier record. A confirmation message is shown before the record is deleted. <u>Test ID: 30</u>
28.	User should have the ability to update Inventory qty for any item.	10	Test ID 31 cross-references an iterative test that shows the quantity of the product being updated and the changes to the record are shown in the treeview. <u>Test ID: 31</u>
29.	System should allow user to email the supplier with the Item and Qty details needed.	10	The 2 cross-referenced tests shown below show email being sent to suppliers when the product quantity is below reorder point or out of stock. <u>Testing: Email being sent to suppliers – Reorder point</u> <u>Testing: Email should be sent to supplier – Stock-Out</u>
30.	System to generate a report that gives a summary of yesterday sales value (not Qty) grouped by categories.	0	The iterative test shown below doesn't show the sales in value(£) by categories but instead shows the

			total sales for the most recent date. <u>Testing: Show total sales in value (£) for most recent date</u>
31.	System menu should allow searching for an item and get the inventory for that item.	10	<p><u>Testing: When user puts all input (option and text)</u> – This test shows the search function being used to search for a record from the product table. When user enters something in the search bar and selects one of the fields to choose from, the records that don't match the user's criteria are removed.</p> <p><u>Test ID: 31</u> – This test cross references an iterative test that shows that if a record is selected from the treeview, its details are populated (shown) on the entry boxes.</p>

Since my client had asked me to add a feature that can forecast sales in terms of quantity sold and value made for the next day and for the next month, I have added 2 more success criteria that isn't shown in the Analysis section.

32.	Forecast sales in terms of quantity sold for next day and for next month for a given product.	9	Last page of Client response shows screenshots of two message boxes showing the predicted sales for tomorrow and for next month in terms of quantity sold. Forecast was shown
-----	-----------------------------------------------------------------------------------------------	---	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

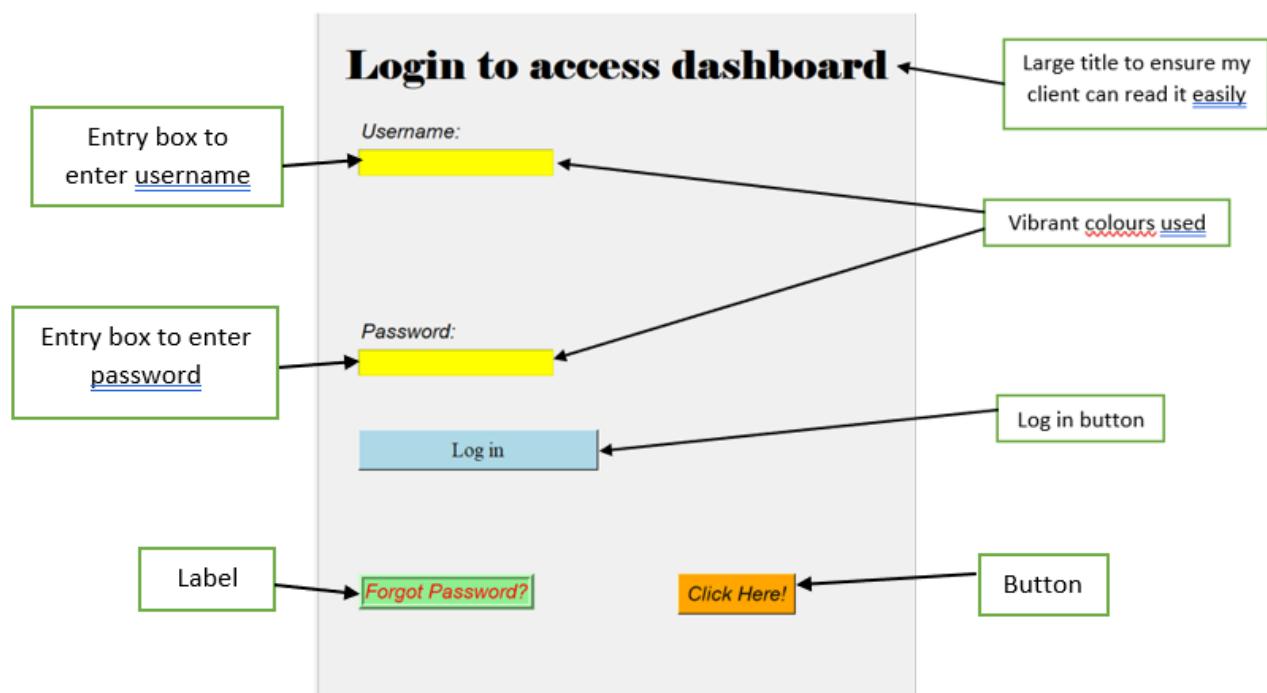
			in decimals instead of a whole number.
33.	Forecast sales in terms of value made (£) for next day and for next month for a given product.	10	Last page of Client response shows screenshots of 2 message boxes showing the predicted sales for tomorrow and for next month in terms of value made (£). Decimal places are acceptable as it is the value made which doesn't have to be a whole number.

Usability features

Login – first window

Here, I have taken a screenshot of the client's response about how user-friendly the GUI is.

Final outcome:



Username:

Trial_test

Candidate number: 6117

Text is visible and easy to read as it has a contrasting colour to the yellow entry box background

Password not shown in English to ensure it is not visible to the public (grocery store customers).

Password:

Login Design Feedback part 1 – (first design)

Here, I have cross-referenced the intended design of the login window as planned out in the design section.

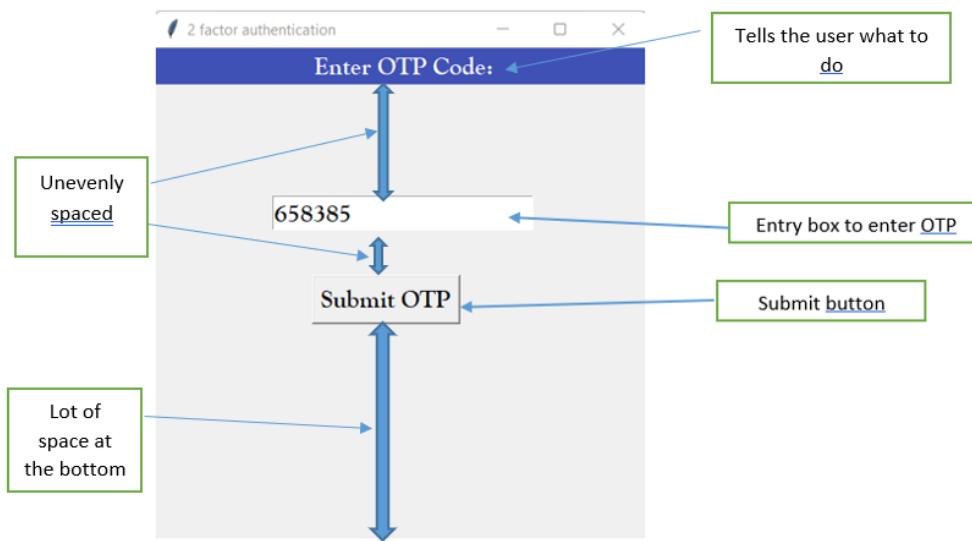
In the intended design, I was going to split the window into 2 parts – the left half that shows an image and the second half that shows the Tkinter widgets – the username and password labels and entry boxes, the forgot password label and the button to click if the user has forgotten their password.

Comparing the final output with the intended design, I have not added an image logo as the Tkinter widgets take up all the space in the window. The intended design could have been made by adding a frame to split the window into half and then adding an image using the same code that was used to display an image in the dashboard. Also, there was no arrow to show the user where to click the button also the text ‘Click Here’ is self-explanatory.

From the client’s feedback, the forgot password label outline could have been normal just like the one in the intended design, instead of making the label have a 3D effect.

Additionally, the text ‘Click Here’ could have been in normal font without it being in italics.

Login – 2 Factor authentication window



Login Design Feedback part 1 – (second design)

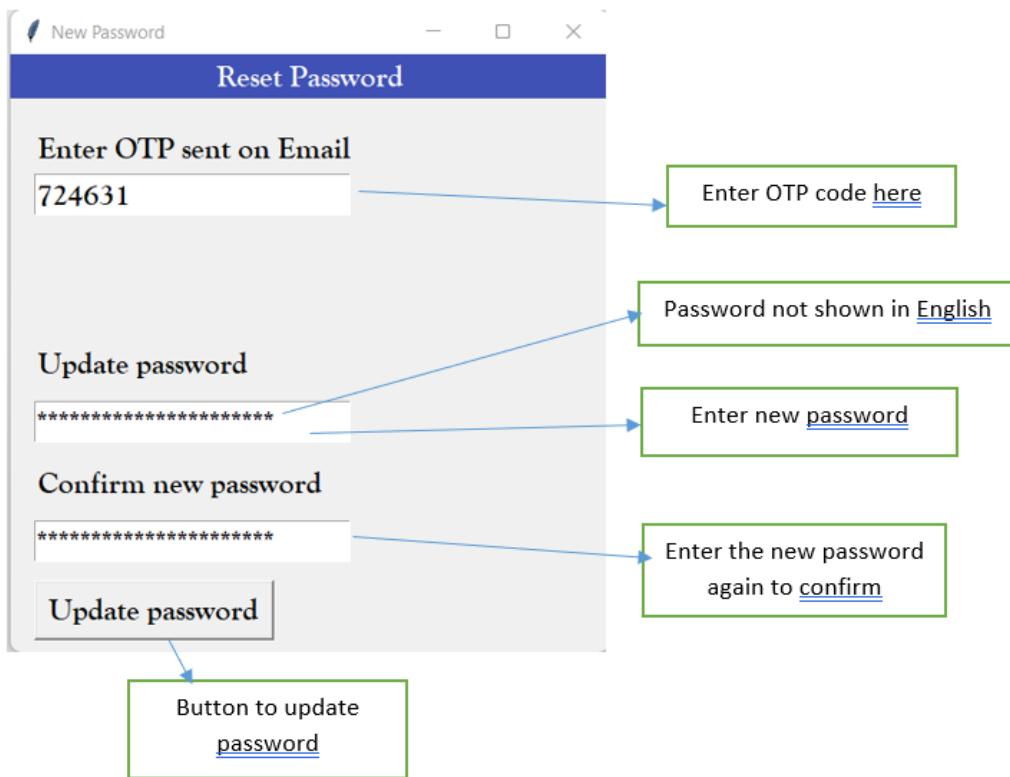
The cross-referenced intended design shows the window to be small with all the Tkinter widgets equally spaced apart. The output as shown above shows the widgets to be unevenly spaced apart as the distances between each widget is different.

In the screenshot above, there is a lot of gap at the bottom in contrast to the design referenced. This could be fixed by reducing the height of the window from the bottom. Other than that, all the widgets have been added and the user can easily add the OTP code in the entry box and press the submit button.

Login – forgot password window

Original design: [Forget Password design](#)

This cross-referenced link above shows what the design was going to look for the forgot password window.



The screenshot above shows the window having all the Tkinter widgets when comparing it to the original design. The user can easily enter the OTP code, and password in the entry boxes. The labels showing what to enter in the entry boxes are easy to read as they have a contrasting colour to the background of the window. The passwords are also not shown in English to avoid it being shown to the public.

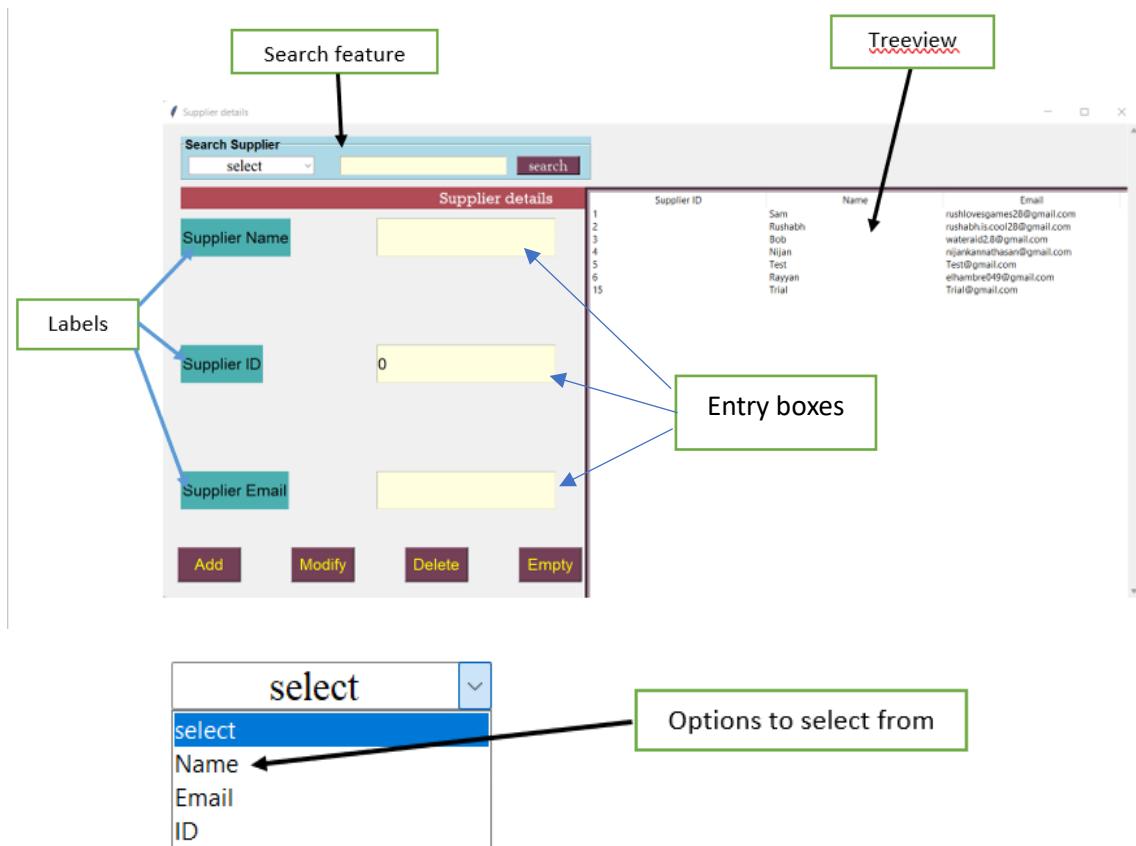
However, the screenshot doesn't match the layout of the cross-referenced design. The screenshot shows all the Tkinter widgets (entry boxes and labels) to be shifted to the left in contrast to the original design where everything was in the middle of the window. The entry boxes shown in the original design are shown to have different background colours (green, light blue and red), whereas the screenshot shows the entry boxes to have a white background colour. The text of the button is different from the original design – however this isn't much of a concern as both the texts ("change" and "Update Password") refer to the same thing.

To improve the screenshot design, I could have added colours to the entry boxes to make the window look more vibrant and attractive and could have changed the x position of the Tkinter widgets so that it is in the centre of the window.

Supplier Window

Original design of supplier window: [Supplier Design](#)

Final output of supplier window:



Client's comments on Supplier Window: Supplier

Here, I have cross-referenced the client's comments about the features of the Supplier window.

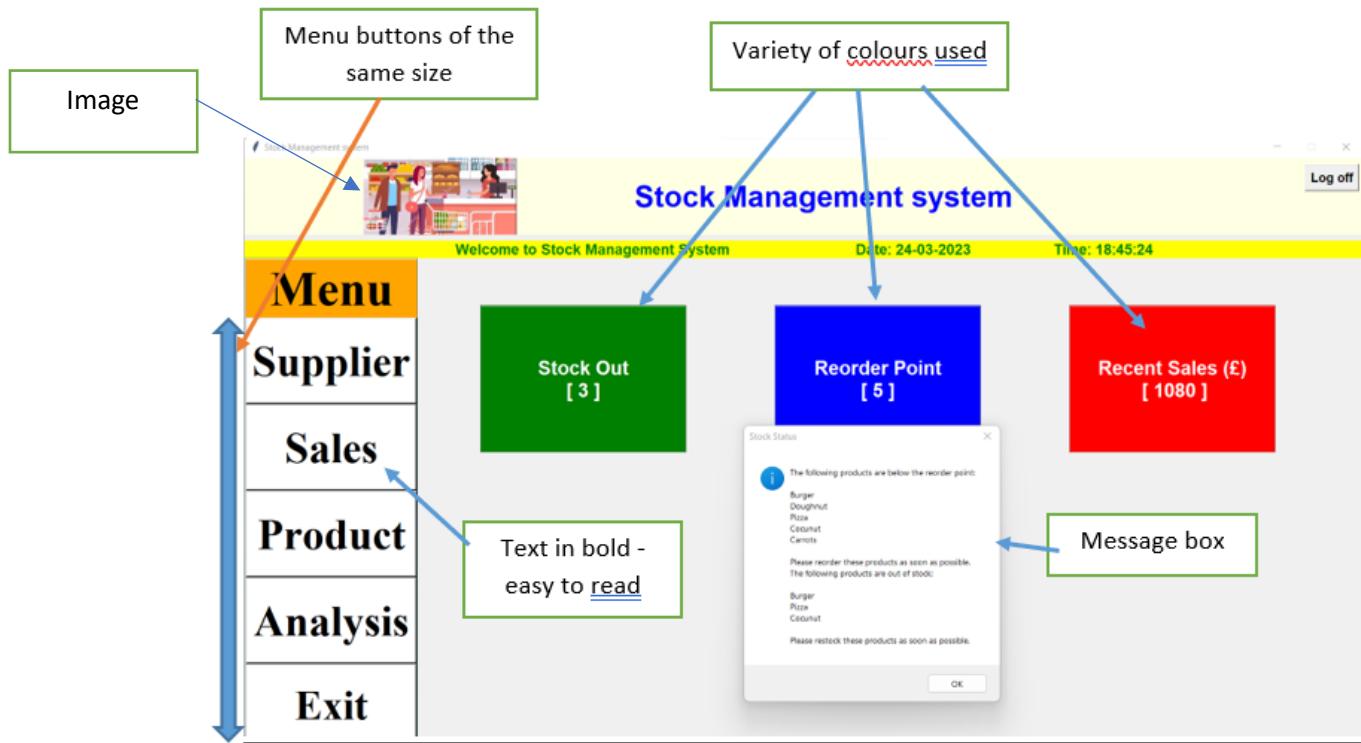
The original design and the actual design of the windows are very similar. The only difference is that the search feature has been extended horizontally to the end of the window in the original design. This could have easily been done in my code which would have made the entry box to search for a supplier much more longer giving more space to enter an input.

All the buttons are easy to use and the text inside the buttons are self-explanatory. The treeview helps the user show if a record has been added, updated and deleted as it instantly shows any changes to the treeview as soon as any of the add, update and delete buttons are clicked.

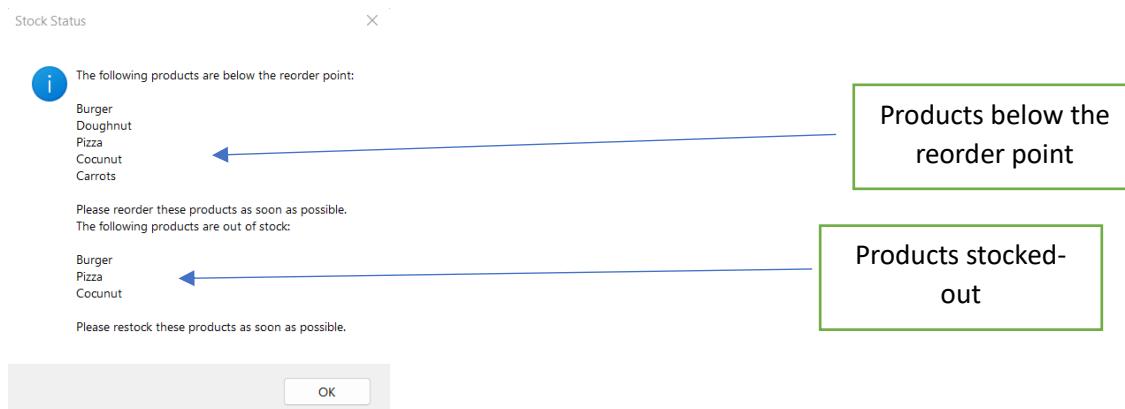
Dashboard

Original design: Dashboard design

Final output:



Client's comments on Dashboard window: Dashboard



The message box that appears as soon as the dashboard page is open is very useful in telling the user what products are out of stock or below the reorder point.

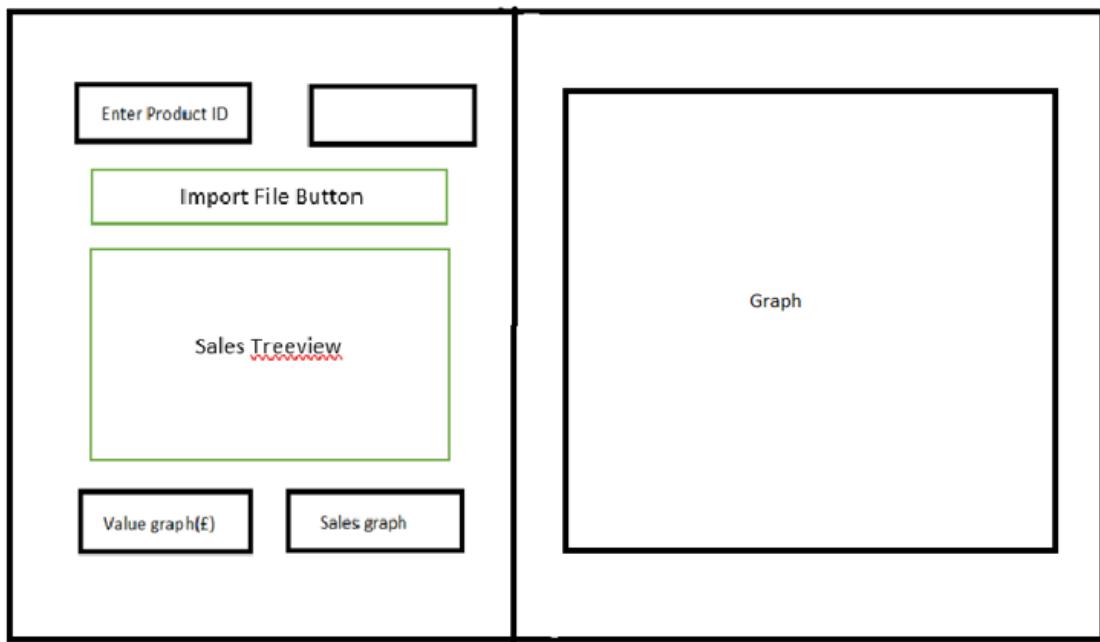
The message doesn't close or allow other parts of the program to run until the user presses the ok button. This means that the user must have seen this message before they can use the program.

The dashboard shows all the functionality as designed. All the menu buttons have been added, and the stock-out and reorder point buttons show the current products out of stock or below the reorder point. The layout is the same, although there are a few changes. I have added an image to make the dashboard look a bit more attractive and have added a time label to show the current time.

I have also used a white background for the menu buttons and black text. The original design shows the menu buttons to have a brown background and a white text. I decided to change the colour so that the menu buttons do not look too extravagant.

Sales window

Original design:



Final design shown below:



Client's comment on sales window: **Sales window**

My client liked the features of the sales window such as the graphs and said that all the buttons were easy to read and use. However, the exact value of the sales is not shown.

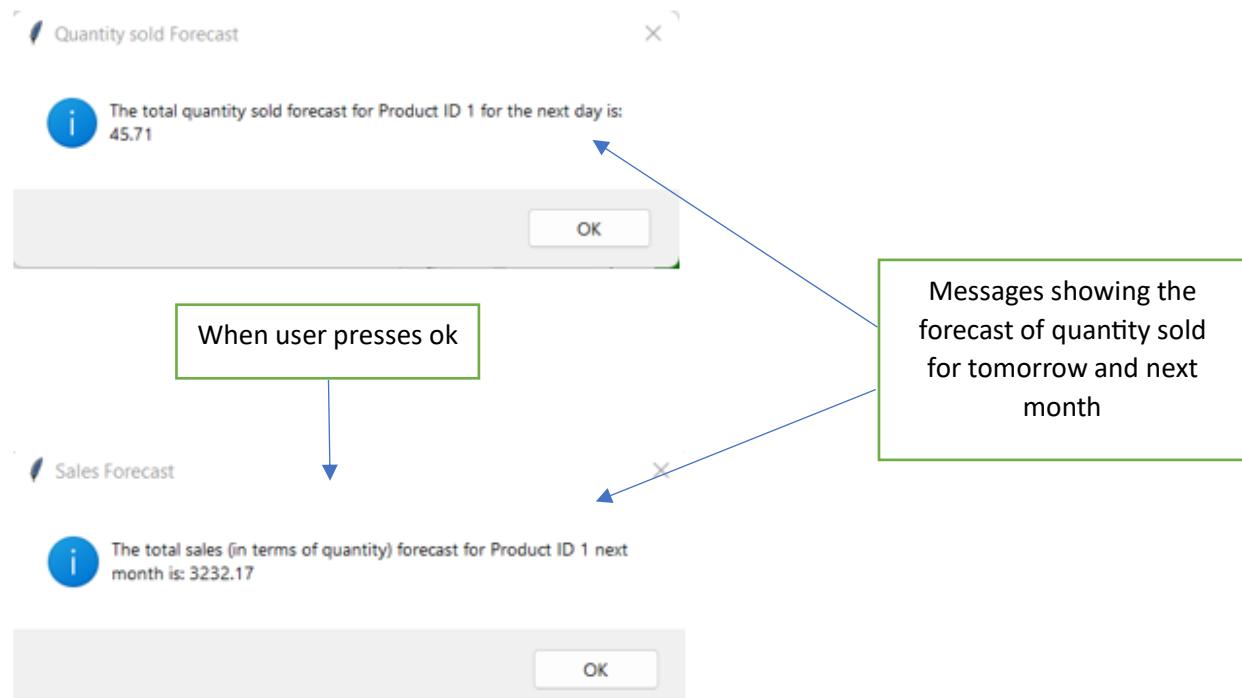
The bars in the graphs are of equal width which makes the graph easy to interpret and looks visually attractive when displaying the data. The bars are also not congested which helps the user to know clearly on what date were the most sales by quantity sold made for a given product.

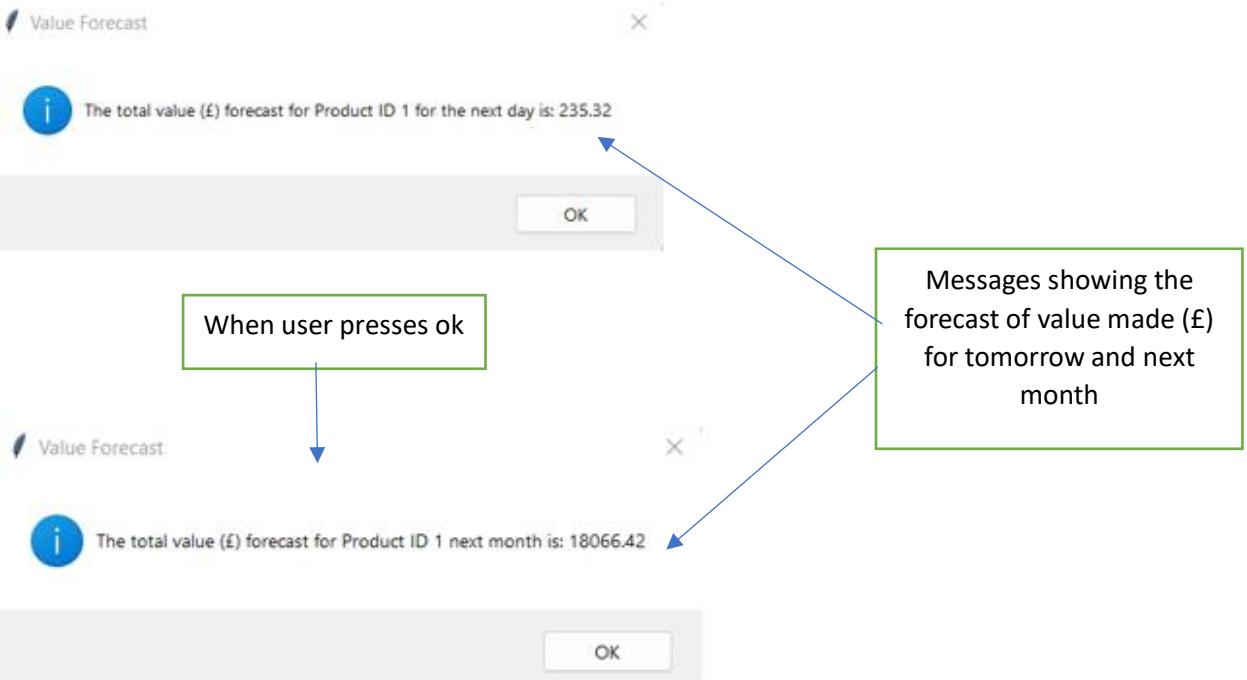
The original design and the final design have the same layout with all the Tkinter widgets being in the same position as planned.

The Graph title shows the name of the product which helps the user to know what product the graphs are being generated for.

A label is shown to tell the user what to enter in the entry box. If the label was not there, then the user would be confused about what data should be entered in the entry box.

To make it easier for the user, instead of telling the user to enter the Product ID, I could have designed the program to allow them to enter the product name instead – as it is easier for the user to remember instead of going on the Product window and using the search utility to find the Product ID for the given product.

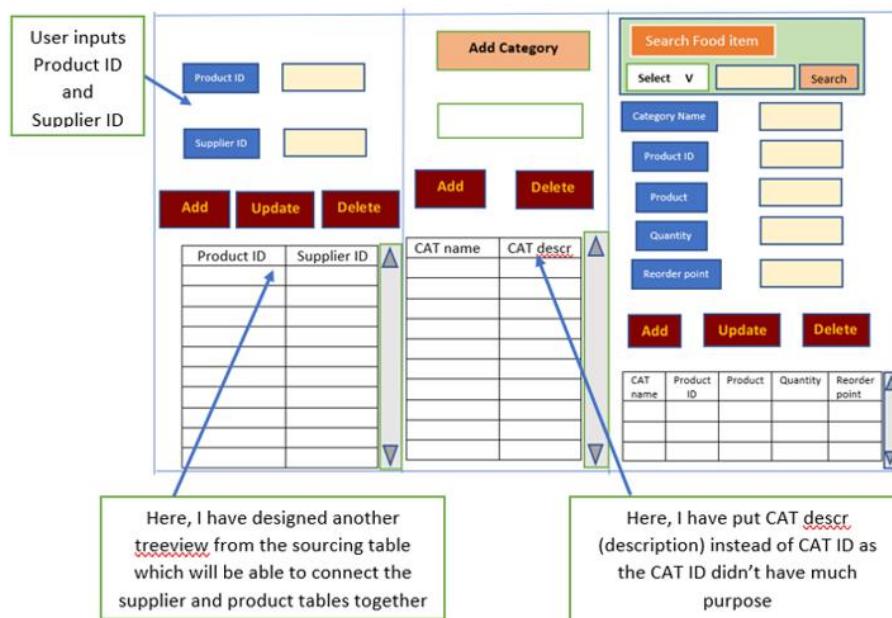




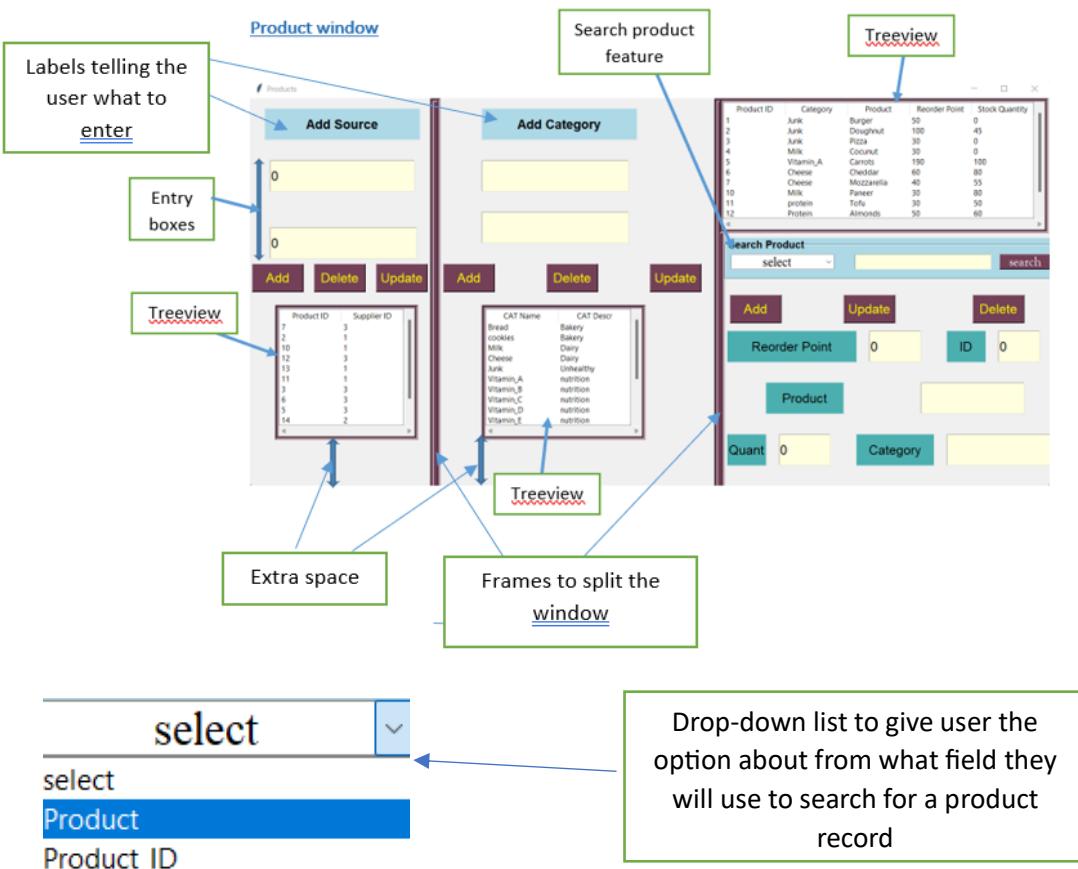
These messages are shown when the user clicks on either the quantity or value graph button. The text is displayed clearly although it could be larger in size. The forecasting messages help the user to make decisions about how much sales they are predicted to make based on the ARIMA model. This can help them compare the predicted forecast values and the actual sales made on that day. This can help them know if their business is running well or not.

Product window

Original design:



Final design:



Client feedback on Product window: Product window

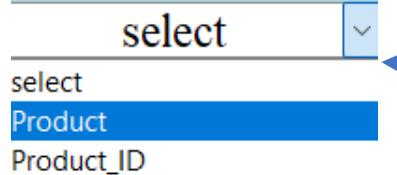
My client really liked the use of frames to split the window into 3 parts (sourcing, category and product) – with each part being allocated its own functionality.

My client was able to read the text properly and found all the Tkinter buttons to be of appropriate size and easy to use.

One point my client mentioned was that there were no label for the sourcing and category section of the product window. This could be added by making the sourcing and category treeview shift down (since there's extra space at the bottom -see annotation) and then add each label above the corresponding entry box.

My client also mentioned that the labels in the product section were uneven and varied in size by a lot. This design is in contrast to the intended design where most of the labels are lined up together and are of similar size. The labels could be improved by making all the labels of the same width and making two columns of labels (ensuring that in each column, the labels are lined up) so that all the space can be utilized.

In the product section (third part of the product window), the original design and the final design have different layouts but the same functionality. The treeview in the final design is at the top and is placed perfectly with no gaps around the sides of the treeview. The search



Drop-down list to give user the option about from what field they will use to search for a product record

feature also takes the entire space unlike in my original design, where there was space in the sides.

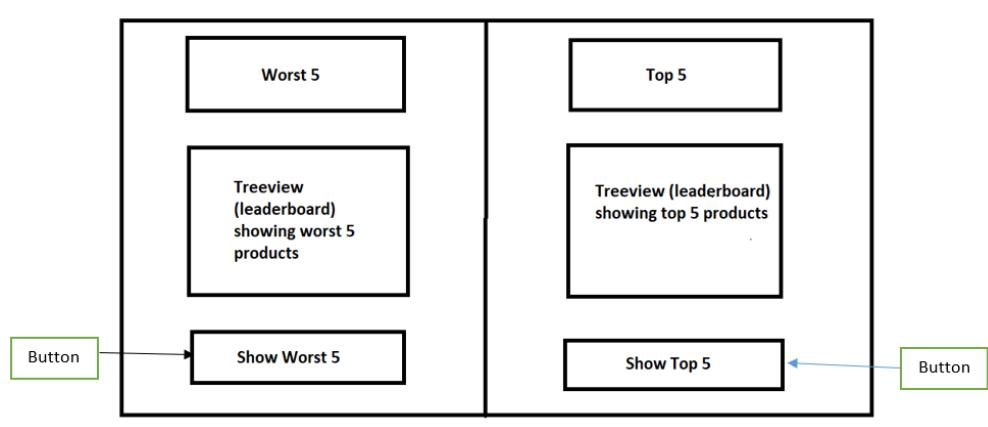
The search feature is really helpful as it allows my client to search for a certain record, without wasting time by scrolling through the treeview. The use of the drop-down list gives my client an option of what field to choose from – the Product ID or the Product. This makes the search feature more efficient if the user can remember any of the two.

I have also added an update button on the category section so that the user can update the category description. This can help the user to know what the category is by providing more information about it.

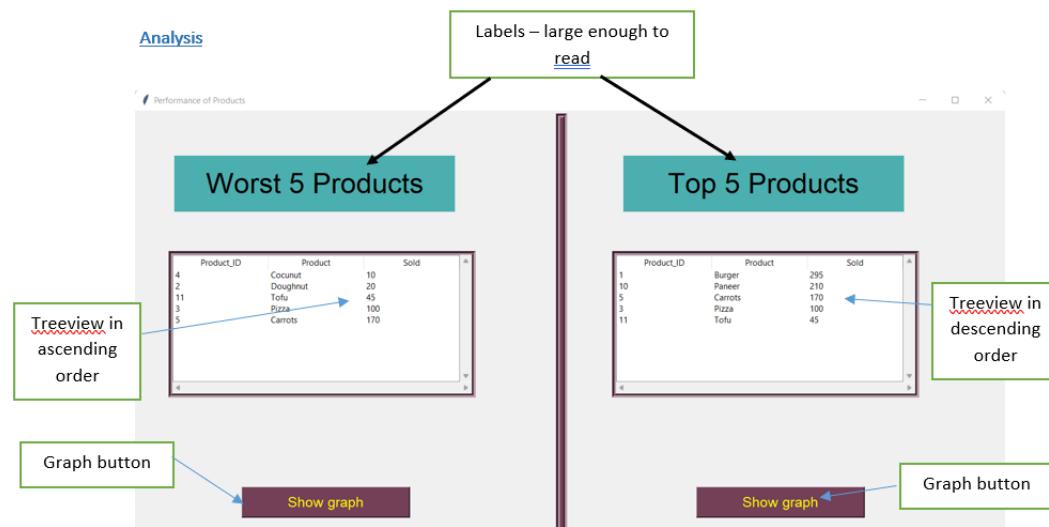
I probably could have used ‘caching’ to save any Product or Product ID’s that my client frequently searches. This would have helped to save time for my client as it would appear as part of the drop-dpwn list.

Analysis window

Original design:



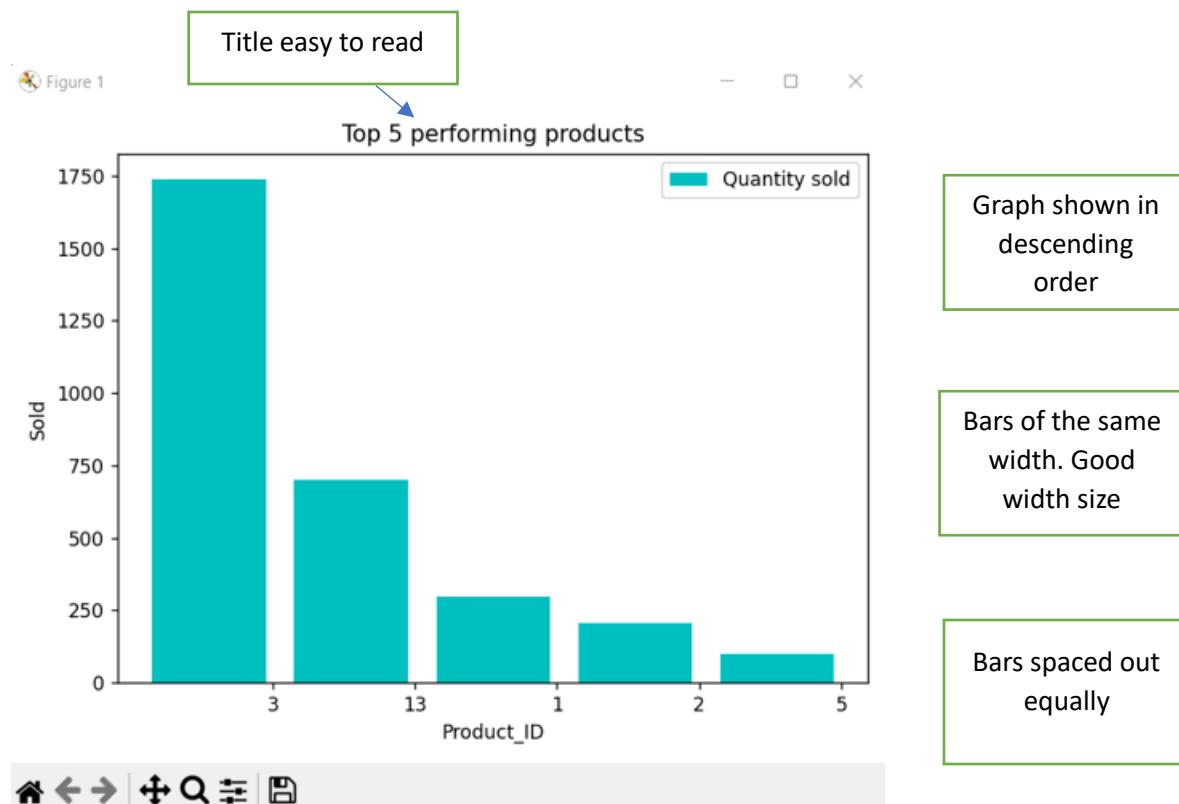
Final design:



Client's comments on Analysis window: [Analysis](#)

My client has no problems with this window. The buttons are easy and clear to read and the treeview shows the name of the products which is easy to refer to when viewing the graphs. Unlike the sales graphs, the exact value can be determined by moving the cursor towards the top of the bar.

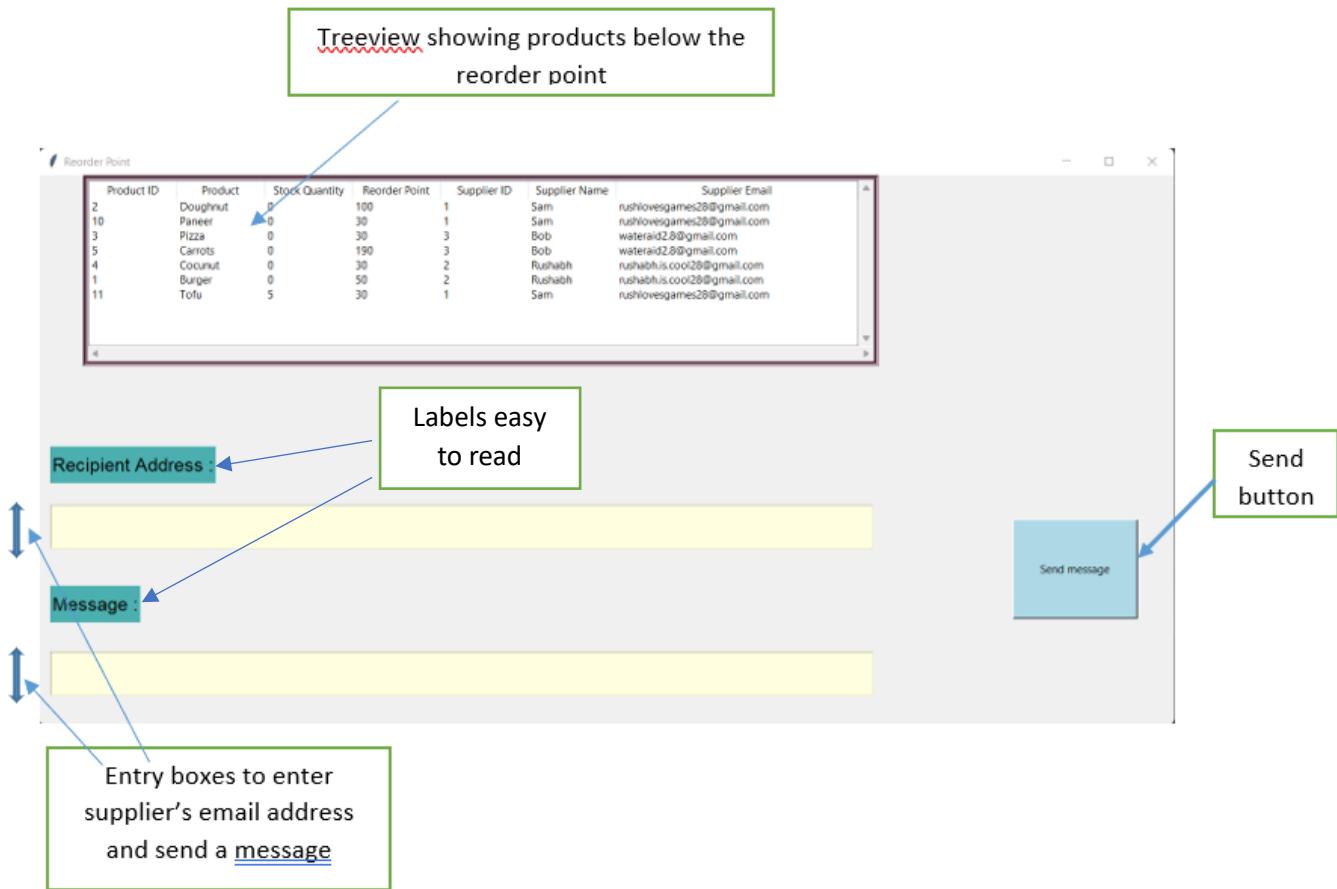
All the features are well made, and the labels are large enough to tell the user which half of the analysis window shows the worst 5 and top 5 products. The final design has the same layout and functionality as the original design.



(This graph is of a different screenshot from a different week so the treeview shown above in the screenshot doesn't relate to this graph). If you want to see the treeview and the graphs showing the same top 5 and worst 5 products, then go back to the Analysis development stage. [Stage 5: Analysis Window](#)

Here, the graphs are shown nicely with the bars created to a good width size so that it is easy to interpret the graph. The bars are also spaced out equally and are not too congested, ensuring that the user can tell the bar for each Product ID.

Reorder point window



Client comments on Reorder point window: Reorder-point

The treeview has been nicely made as it shows all the products below the reorder point and the suppliers that are supplying the products. The user can easily click on the treeview record which then shows the supplier's email address on the entry box.

My client liked the idea of not having to type the email address of the supplier or else it would have been too tedious and there could be chances of spelling mistakes occurring.

My client mentioned that the entry box to type the message could have been designed so that each line is of a fixed width and a new line will start if the sentence exceeds the fixed width.

The send button has a contrasting background colour compared to the entire window which helps the user to clearly see the button to send the message. The text inside the button could have probably been a bit bigger to emphasise the functionality of the button.

The labels are clear enough to read and allow my user to know where to enter the message.

Here, the product records are in ordered by how much stock quantity they have left in the store. Here, tofu is the last record as it has the most quantity out of all the products below the reorder point.

This makes sure that my client contacts the supplier of the product first when there is currently the smallest quantity available in the store.

Stock-Out window



(Same usability features as the reorder point window).

Client's comments on the Stock-Out window: **Stock-out**

Limitations

The most common limitation in my software was some of the Tkinter windows not being designed as planned out in my design section. For example, based on success criteria 3, the forgot password window had Tkinter widgets that were not equally spaced out in contrast to the original design. My client wasn't very happy with the structure of the window as the widgets were not in proportion to each other. This could have been avoided by scaling the window size down and repositioning the widgets so that they are roughly spaced apart equally.

I wasn't able to achieve one of the success criteria (30) where I had to show yesterday's sales value grouped by categories. My program instead shows yesterday's total sales, but not by category. This could be done by using an SQL query to add up the most recent sales value by using the sum() function and then using the 'group by' statement to show the sum by categories.

There were also some missing elements in my GUI. For example, in the Product window, there were no labels in the Sourcing and category section to tell the user in which entry box the data should go in. This could have been avoided by moving the Sourcing and Category treeviews down as there was a lot of unused space at the bottom, and then readjusting all the other entry boxes and buttons, so that the labels can be added on top of their corresponding entry boxes.

Another issue was sometimes an error message was showing up when my client was clicking the 'import file' button telling the user to make sure the date is in the correct format, even though it was. The solution to this problem was for my client to convert the date to the 'YYYY-MM-DD' format even though it was already in that format. This, however, could cause a lot of problems if there are a lot sales records in the CSV file. This would take the user some time to highlight all the cells where the date has been written and then convert it to the 'YYYY-MM-DD' format.

Another limitation is the accuracy of the ARIMA model to forecast future sales. Forecasting doesn't always produce accurate values and so the predicted sales may be unreliable for my client to use and make decisions with. When using the ARIMA model, there were 3 parameters I had to use – the p,d and q value. These values set can be subjective based on past data but due to limited time, I set these parameters equal to 1 without going too much in depth. The forecasting feature may be unsuitable currently due to not enough data for each product, but if my client's grocery store grows more in the future, then this feature can be very useful as it is more likely to make better forecasts when there is more past data to use.

In my product table, for each product I had entered a reorder point. When adding product records in the product table, I didn't calculate the reorder point using any formulas but instead I placed random reorder point numbers for each of my product. This could be improved by understanding more about how reorder points should be calculated and then

confirming with my client about whether the reorder point is suitable for that certain product.

Lastly, another limitation are the stock-out and reorder point message alerts. These alerts are only shown once – just as soon as the user gets access to the dashboard window. The problem is if the user manually updates a product or imports the sales file from yesterday, then there won't be any new reorder point or stock-out alerts for any new products that are below the reorder point or are stocked-out. The alert message for the new products will only be shown if the user logs into the system again. This could be solved by creating a new function that displays a reorder point and stock-out alert every 30 minutes. This will ensure that my client doesn't forget to ask the suppliers to supply more of the products to the grocery store and ensures that all the products below the reorder point or are stocked-out throughout the day are mentioned in the alert.

Maintenance Issues

The biggest maintenance issue is using SQLite as the database engine. SQLite has a limited database size meaning that as my client's grocery store business grows over time, a much larger database will be required. This may cause problems if my client relies on SQLite as the database engine as a long-term use. My client would have to switch to another database engine such as MySQL that can handle larger databases.

Additionally, as the client's grocery store grows, a lot more processes will occur and there will be a larger dependency on the program. There are likely to be more members of staff in the store who would also have access to the software. If more than one person is using the software, then there will be lots of processing taking place such as writing to database tables. Unfortunately, SQLite doesn't allow multiple writes at the same time and so when the members of the store try to add data to the tables, the database is likely to get locked.

Another maintenance issue is logging in to the system. The user is expected to know their username if they have forgotten their password. This can be a serious issue if the user has forgotten their username as well. To tackle this issue, the program could be made so that the manager of the grocery store has full access to the system and just needs to enter their name and surname to access the entire system. If there are more members to the store, then the program could be designed to have user access levels so that junior and senior members have different access rights to the program. For example, a junior member may not be allowed to update or import CSV files as this will affect the graphs and the forecasts that are generated which could result in business decisions made wrongly, if any incorrect sales records have been recorded.

Lastly, another maintenance issue is the user sending emails to the suppliers. From the tests I did, I only used a few products that existed in the grocery store. This means that only a few items would be below the reorder point and so the client would only have to send some emails to the suppliers. However, if there were around 100 products below the reorder point, then my client would have to send 100 emails to the suppliers. This is quite repetitive and inefficient, as it is impractical for my client to send that many emails throughout the day.

The program could be designed so that automated emails are sent to the suppliers as soon as a product is below the reorder point. This removes the redundancy of writing the same code for both the stock-out and reorder point window. Although my client didn't want automated emails (reference: **Stock-out Dashboard Button design**), this could be set as an option in case my client decides to use the automated email feature.

Bibliography

1. https://www.google.com/search?q=how+to+build+tkinter+gui+using+oop&rlz=1C1CBF_en-GBGB975GB976&oq=how+to+build+tkinter+gui+using+oop&aqs=chrome..69i57j33i10i160l2.4512j0j4&sourceid=chrome&ie=UTF-8#fpstate=ive&vld=cid:18f567ff
2. Jason Brownlee (2017). How to Create an ARIMA Model for Time Series Forecasting in Python. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/>.
3. www.youtube.com. (n.d.). Create a custom email GUI app using python (Tkinter and Smtpplib). [online] Available at: <https://www.youtube.com/watch?v=qfOgihp-gEU>.
4. www.youtube.com. (n.d.). OTP Verification System Using Python | OTP Verification in Python | Python Projects | Simplilearn. [online] Available at: <https://www.youtube.com/watch?v=j1Hag4Kbpjs>.
5. www.youtube.com. (n.d.). How to Query Multiple Tables with JOINs. [online] Available at: <https://www.youtube.com/watch?v=7XDGHwclZck>.
6. <https://stackoverflow.com/questions/2887878/importing-a-csv-file-into-a-sqlite3-database-table-using-python>
7. www.youtube.com. (n.d.). Upload A CSV File (Or Any Data File) To SQLite Using Python. [online] Available at: <https://www.youtube.com/watch?v=UZlhVmkrAEs>.
8. Analytics Vidhya. (2020). How to Create an ARIMA Model for Time Series Forecasting in Python? [online] Available at: <https://www.analyticsvidhya.com/blog/2020/10/how-to-create-an-arima-model-for-time-series-forecasting-in-python/>.

9. Available at:

https://www.google.com/search?q=stock+management+system+in+python&rlz=1C1CHBF_en-GBGB975GB976&oq=stock+management+system+in+python&aqs=chrome.69i57j0i13i512j0i22i30l6j0i390i650l2.5991j0j4&sourceid=chrome&ie=UTF-8#fpstate=ive&vld=cid:9ed78fc1.

10. GeeksforGeeks. (2021). How to Import a CSV file into a SQLite database Table using Python? [online] Available at: <https://www.geeksforgeeks.org/how-to-import-a-csv-file-into-a-sqlite-database-table-using-python/>.

11. ProjectPro. (n.d.). How to Build ARIMA Model in Python for time series forecasting? [online] Available at: <https://www.projectpro.io/article/how-to-build-arima-model-in-python/544>.

12. www.javatpoint.com. (n.d.). Python Tkinter Button - Javatpoint. [online] Available at: <https://www.javatpoint.com/python-tkinter-button>.

13. ActiveState. (n.d.). How to Display Data in a Table using Tkinter. [online] Available at: <https://www.activestate.com/resources/quick-reads/how-to-display-data-in-a-table-using-tkinter/>.

14. 867, W. (n.d.). Retrieved from:

www.youtube.com. (n.d.). #1 How to Create Inventory Management System with Database in Python | 0 to Hero Course | Billing. [online] Available at: <https://www.youtube.com/watch?v=uxLuAz7b1tU&list=PL4P8sY6zvjk6ef4lpm6XiwJVRahLCp6DI&index=2>

15. www.tutorialspoint.com. (n.d.). How to get the value of an Entry widget in Tkinter. [online] Available at: <https://www.tutorialspoint.com/how-to-get-the-value-of-an-entry-widget-in-tkinter>.

16. datatofish.com. (n.d.). How to Create a Bar Chart in Python using Matplotlib - Data to Fish. [online] Available at: <https://datatofish.com/bar-chart-python-matplotlib/>.

Appendix

Code for creating tables and setting up database

```

import sqlite3 # to use sqlite3
def create_dtbse():
    con = sqlite3.connect(database=r"SMS.db") # connecting to database
    cur = con.cursor()

    # supplier table
    cur.execute("CREATE TABLE IF NOT EXISTS supplier(ID integer PRIMARY KEY, Name text, Email text)") # supplier table
    con.commit()

    # login table
    cur.execute("CREATE TABLE IF NOT EXISTS login(ID integer PRIMARY KEY AUTOINCREMENT, Username text, Password text, Email text)")
    con.commit()

    # category table
    cur.execute("CREATE TABLE IF NOT EXISTS category(CAT_NAME PRIMARY KEY NOT NULL, CAT_DESCR text NOT NULL)")
    con.commit()

    # Product table
    cur.execute("CREATE TABLE IF NOT EXISTS Product(Product_ID integer PRIMARY KEY AUTOINCREMENT, CAT_NAME text NOT NULL, Product_text NOT NULL, "
               "Reorder_Point integer NOT NULL, Quantity integer NOT NULL, FOREIGN KEY (CAT_NAME) REFERENCES category(CAT_NAME) ON DELETE CASCADE);")
    con.commit()

    # Sourcing table
    cur.execute("CREATE TABLE IF NOT EXISTS Sourcing(Product_ID integer, Supplier_ID integer, foreign key (Product_ID) references Product(Product_ID), "
               "foreign key (Supplier_ID) references supplier(ID), primary key (Product_ID, Supplier_ID))")
    con.commit()

```

```

# Sales table
cur.execute("CREATE TABLE IF NOT EXISTS Sales(Date text, Product_ID integer, Sold integer, Value integer, foreign key (Product_ID) references "
           "Product(Product_ID), primary key(Date, Product_ID))")
con.commit()

create_dtbse() # calling the function

```

Code for Dashboard window

```

from tkinter import * # this is used to create the GUI of my stock management system
from PIL import Image, ImageTk
from supplier import SupplierClass # to use supplier window
from Sales_records import SalesClass # to use sales window
from product import ProductClass # to use product window
from Analysis import AnalysisClass # to use analysis window
from Reorder_Point import Reorder_Point # to use reorder point window
from Stock_Out import Stock_Out # to use stock-out window
import numpy as np
import sqlite3 # to use sqlite as the database
from tkinter import ttk, messagebox # this will show any errors or any info when giving input
import time # to use time function
from datetime import datetime
class SMS: # creating a class called SMS - short for Stock Management System
    def __init__(self, wind): # short for window
        self.wind = wind
        self.wind.geometry("1900x990+0+0") # this sets the dimensions of the window
        self.wind.title("Stock Management system") # this is the title of the window

```

```

# path showing where my image is
self.image_title = PhotoImage(file=r"C:\Users\rusha\PycharmProjects\pythonProject5\grocery_store_image.png")

title = Label(self.wind, text = "Stock Management system", image = self.image_title, compound = LEFT,
font = ("arial", 30, "bold"), bg = "#FFFFE4", fg = "blue", anchor = "w", padx= 200).place(x=0, y=0, relwidth= 1, height = 130)

self.wind.resizable(False, False) # you can't resize the screen

butn_log = Button(self.wind, text="Log off", font =("arial", 13, "bold"), command = wind.destroy) # creating a log off button
butn_log.place(x = 1800, y = 10) # this places the log off button depending on its position

self.time = Label(self.wind, text=" Welcome To The Stock Management system!!\t\t Date: DD-MM-YYYY\t\t Time: HH:MM:SS",
compound=LEFT, font=("arial", 15, "bold"), bg="yellow", fg="green")
self.time.place(x = 0, y = 140, relwidth=1, height = 30)

LeftMenu = Frame(self.wind, bd=2, bg = "lightblue", relief=GROOVE, cursor = "circle") # this changes the cursor to circle when its on the frame
LeftMenu.place(x=0, y=170, width=295, height=1000) # this is used to adjust the placement of the frame as well as its dimensions

```

```

#####
-----These sections will go under the left menu -----
#####

menu_sect =Label(LeftMenu, text = "Menu", font = ("times new roman", 50, "bold"), bg = "orange").pack(side = TOP, fill = X) # this is a label

butn_supplier = Button(LeftMenu, text = "Supplier", command = self.supplier,
font = ("times new roman", 44, "bold"), bg = "white", bd = 3, cursor = "plus").pack(side = TOP, fill = X) # supplier button

butn_sales = Button(LeftMenu, text="Sales", command = self.Sales_records,
font = ("times new roman", 44, "bold"), bg="white", bd=3, cursor="plus").pack(side=TOP, fill=X) # sales button

butn_product = Button(LeftMenu, text="Product", command = self.product,
font = ("times new roman", 44, "bold"), bg="white", bd=3, cursor="plus").pack(side=TOP, fill=X) # product button

butn_analysis = Button(LeftMenu, text = "Analysis", command= self.Analysis,
font = ("times new roman", 44, "bold"), bg = "white", bd = 3, cursor = "plus").pack(side = TOP, fill = X) # analysis button

butn_exit = Button(LeftMenu, text="Exit", font=( "times new roman", 42, "bold"), bg="white", bd=3,
command = wind.destroy, cursor="plus").pack(side=TOP, fill=X) # exit button

self.lbl_stockout = Button(self.wind, text = "Stock-Out \n[0]", bg = "green", command= self.Stock_Out, relief = "groove",
fg = "white", font = ("arial", 20, "bold"))

```

```

self.lbl_stockout.place(x = 400, y = 250, height = 250, width = 350)

self.lbl_reorder = Button(self.wind, text="Reorder-Point \n[0]", bg="blue", command= self.Reorder_Point, relief="groove", fg="white", font =("arial", 20, "bold"))
self.lbl_reorder.place(x=900, y=250, height=250, width=350)

self.lbl_sales_recent = Label(self.wind, text="Recent Sales \n[0]", bg="red", relief="groove", fg="white", font =("arial", 20, "bold"))
self.lbl_sales_recent.place(x=1400, y=250, height=250, width=350)

self.update_label() # calls the update function
self.show_status()

def update_label(self):
    connection = sqlite3.connect(r'sms.db')
    cur = connection.cursor()
    try:
        ###### Reorder point #####
        cur.execute("Select * from Product WHERE Quantity < Reorder_Point")
        reorder_point = cur.fetchall() # fetches all the records where the quantity is less than the reorder point
        self.lbl_reorder.config(text=f'Reorder Point\n[ {str(len(reorder_point))} ]') # displays the number of items that fulfil the condition

        ##### Stock Out #####
        cur.execute("Select * from Product WHERE Quantity = 0") # select all record where quantity is 0
        stock_out = cur.fetchall()
        self.lbl_stockout.config(text=f'Stock Out\n[ {str(len(stock_out))} ]') # gets the number of products that have 0 quantity and adds it to label
    except:
        pass

```

```

#####
##### Recent Sales #####
#####

cur.execute("SELECT Sum(Value) FROM Sales where Date = (SELECT MAX(Date) FROM Sales)") # gets the total sum of sales from the most recent date
recent_sales = cur.fetchall()

array_recent_sales = np.array(recent_sales) # creates a numpy array from the database query
array_recent_sales_new = str(array_recent_sales) # converting to string

most_recent_sales = array_recent_sales_new[2:-2] # this removes any brackets

self.lbl_sales_recent.config(text=f' Recent Sales (£) \n [ {str(most_recent_sales)} ]') # add it to the label

time_ = time.strftime("%H:%M:%S") # time
date_ = time.strftime("%d-%m-%Y") # date

self.time.config(text=f"Welcome to Stock Management System\t Date: {str(date_)}\t Time: {str(time_)}") # this ensures that the labels change in real time
self.time.after(200, self.update_label)

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

```

def show_status(self): # function to create reorder point and stock out alerts
    connection = sqlite3.connect(r'sms.db') # connecting to database
    cur = connection.cursor()
    try:
        # Get products that are below reorder point
        cur.execute("SELECT * FROM Product WHERE Quantity < Reorder_Point")
        reorder_point_products = cur.fetchall()

        # Get products that are out of stock
        cur.execute("SELECT * FROM Product WHERE Quantity = 0")
        stock_out_products = cur.fetchall()

        # Create message for reorder point products
        if len(reorder_point_products) > 0: # if the number of products below the reorder point is greater than 0
            reorder_point_message = "The following products are below the reorder point:\n\n"
            for product in reorder_point_products: # for loop
                reorder_point_message += f"{product[2]}\n" # adding products to message
            reorder_point_message += "\nPlease reorder these products as soon as possible." # add this text to the message
        else:
            reorder_point_message = ""

        # Create message for stock out products
        if len(stock_out_products) > 0:
            stock_out_message = "The following products are out of stock:\n\n"
            for product in stock_out_products: # for loop
                stock_out_message += f"{product[2]}\n" # adding products to message
            stock_out_message += "\nPlease restock these products as soon as possible." # add this text to the message
        else:
            stock_out_message = ""

        # Show message box if there are any products below reorder point or out of stock
        if reorder_point_message != "" or stock_out_message != "":
            messagebox.showinfo("Stock Status", f"{reorder_point_message}\n{stock_out_message}", parent=self.wind) # this shows message box

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

```

# Create message for reorder point products
if len(reorder_point_products) > 0: # if the number of products below the reorder point is greater than 0
    reorder_point_message = "The following products are below the reorder point:\n\n"
    for product in reorder_point_products: # for loop
        reorder_point_message += f"{product[2]}\n" # adding products to message
    reorder_point_message += "\nPlease reorder these products as soon as possible." # add this text to the message
else:
    reorder_point_message = ""

# Create message for stock out products
if len(stock_out_products) > 0:
    stock_out_message = "The following products are out of stock:\n\n"
    for product in stock_out_products: # for loop
        stock_out_message += f"{product[2]}\n" # adding products to message
    stock_out_message += "\nPlease restock these products as soon as possible." # add this text to the message
else:
    stock_out_message = ""

# Show message box if there are any products below reorder point or out of stock
if reorder_point_message != "" or stock_out_message != "":
    messagebox.showinfo("Stock Status", f"{reorder_point_message}\n{stock_out_message}", parent=self.wind) # this shows message box

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

```
def supplier(self):
    self.supplier_1 = Toplevel(self.wind) # creates a window on top of dashboard
    self.new_supp = SupplierClass(self.supplier_1)

def Sales_records(self):
    self.Sales = Toplevel(self.wind) # creates a window on top of dashboard
    self.new_sales = SalesClass(self.Sales)

def product(self):
    self.Products = Toplevel(self.wind) # creates a window on top of dashboard
    self.product_item = ProductClass(self.Products)

def Analysis(self):
    self.analysis = Toplevel(self.wind) # creates a window on top of dashboard
    self.new_analysis = AnalysisClass(self.analysis)

def Reorder_Point(self):
    self.Reorder_Point = Toplevel(self.wind) # creates a window on top of dashboard
    self.new_Reorder_Point = Reorder_Point(self.Reorder_Point)

def Stock_Out(self):
    self.Stock_Out = Toplevel(self.wind) # creates a window on top of dashboard
    self.new_Stock_Out = Stock_Out(self.Stock_Out)
```

```
if __name__ == "__main__":
    wind = Tk()
    obj = SMS(wind)
    wind.mainloop() # this executes Tkinter and runs the application
```

Code for Supplier window

```

from tkinter import *    # this is used to create the GUI of my stock management system
from tkinter import ttk, messagebox # this will show any errors or any info when giving input
import sqlite3 # using SQLite as database engine
import re # regular expressions used for validation
class SupplierClass: # creating a class called SupplierClass
    def __init__(self, wind):    # short for window
        self.wind = wind
        self.wind.title("Supplier details") # this is the title of the window

        # supplier variables
        self.var_searchuse = StringVar() # this stores the option that the user selects from drop-down list
        self.var_searchtxt = StringVar() # this stores the text that the user will enter in the search bar
        self.var_supplier_id = IntVar()
        self.var_supplier_name = StringVar()
        self.var_supplier_email = StringVar()

        # adjusting the window
        window_width = 1550
        window_height = 750
        screen_width = wind.winfo_screenwidth()
        screen_height = wind.winfo_screenheight()
        center_x = int(130+(screen_width / 2 - window_width / 2)) # centre x
        center_y = int(50+(screen_height / 2 - window_height / 2)) # centre y

        # set the position of the window to the center of the screen
        self.wind.geometry(f'{window_width}x{window_height}+{center_x}+{center_y}')

```

```

# creating a labelframe
Search_box = LabelFrame(self.wind, text = "Search Supplier", bg = "lightblue", font = ("arial", 12, "bold"))
Search_box.place(x = 30, y = 20, width = 650, height = 70)

# options to search from - Name, Email or ID

option_block = ttk.Combobox(Search_box, textvariable= self.var_searchuse, values = ("select", "Name", "Email", "ID"),
                           state= "readonly", justify = CENTER,
                           font = ("times new roman", 15))
option_block.place(x = 10, y= 5, width = 200, height = 30)
option_block.current(0)

txt_block = Entry(Search_box, textvariable= self.var_searchtxt ,font = ("goudy old style", 15), bg = "lightyellow").place(x= 250, y = 5, height = 30)

# search button
btn_block = Button(Search_box, text = "search", command = self.search, font = ("goudy old style", 15), bg = "#734058",
                  fg = "white").place(x= 530, y = 5, width = 100, height = 30)

title = Label(self.wind, text = "Supplier details", anchor = CENTER,
             font = ("Rockwell",15), bg = "#af4c56", fg = "white").place(x=30, y = 100, width = 1000)

lbl_suppl_id = Label(self.wind, text = "Supplier ID",
                     font = ("Rosewood Std Regular",15), bg = "#4cafaf", fg = "black").place(x=30, y = 350, height = 60)

lbl_suppl_name = Label(self.wind, text = "Supplier Name",
                      font = ("Rosewood Std Regular",15), bg = "#4cafaf", fg = "black").place(x=30, y = 150, height = 60)

```

```

lbl_suppl_email = Label(self.wind, text = "Supplier Email", font = ("Rosewood Std Regular",15),
                        bg = "#4cafaf", fg = "black").place(x = 30, y = 550, height = 60)

txt_suppl_id = Entry(self.wind, textvariable= self.var_supplier_id, font = ("Rosewood Std Regular",15),
                      bg = "lightyellow", fg = "black").place(x=340, y = 350, height = 60)

txt_suppl_name = Entry(self.wind, textvariable= self.var_supplier_name, font = ("Rosewood Std Regular",15),
                       bg = "lightyellow", fg = "black").place(x=340, y = 150, height = 60)

txt_suppl_email = Entry(self.wind, textvariable= self.var_supplier_email, font = ("Rosewood Std Regular",15),
                        bg = "lightyellow", fg = "black").place(x = 340, y = 550, height = 60)

#### Adding button
Add_btn = Button(self.wind, text = "Add", command = self.add, bg = "#734058",
                 font = ("OCR A Std", 15), fg = "yellow").place(x = 25, y = 670, height = 55, width = 100)

#### Modifying Button (updates)
Mdfy_btn = Button(self.wind, text="Modify", command = self.modify, bg="#734058",
                   font=("OCR A Std", 15), fg="yellow").place(x=205, y=670, height= 55, width = 100)

#### Deleting Button
Dlt_btn = Button(self.wind, text="Delete", command = self.delete_record, bg="#734058",
                  font=("OCR A Std", 15), fg="yellow").place(x=385, y=670, height=55, width = 100)

```

```

#### Empty Button

Empty_btn = Button(self.wind, text = "Empty", command = self.empty, bg = "#734058",
                   font = ("OCR A Std", 15), fg = "yellow").place(x = 565, y = 670, height = 55, width = 100)

#### Having a vertical frame section

Sply_frame = Frame(self.wind, bd = 7, relief = GROOVE, highlightcolor="red", bg = "#734058", cursor = "circle" )
Sply_frame.place(x = 670, y = 100, height = 900, relwidth = 1 )

#### vertical and horizontal scrollbar
myscrolly = Scrollbar(self.wind, orient= VERTICAL)
myscrollx = Scrollbar(Sply_frame, orient = HORIZONTAL)

```

```
#####
self.SupplyTable = ttk.Treeview(Sply_frame, columns=("ID", "Name", "Email"), yscrollcommand= myscrolly.set, xscrollcommand= myscrollx.set)

myscrollx.config(command = self.SupplyTable.xview)
myscrolly.config(command = self.SupplyTable.yview)

myscrollx.pack(side = BOTTOM, fill = X)
myscrolly.pack(side = RIGHT, fill = Y)

self.SupplyTable.heading("ID", text="Supplier ID")
self.SupplyTable.heading("Name", text = "Name")
self.SupplyTable.heading("Email", text="Email")
self.SupplyTable["show"] = "headings"

self.SupplyTable.column("ID", width =280, stretch=NO) # fixed width
self.SupplyTable.column("Name", width = 280, stretch=NO) # fixed width
self.SupplyTable.column("Email", width =280, stretch= NO) # fixed width

self.SupplyTable.pack(fill=BOTH, expand=1)
self.SupplyTable.bind("<ButtonRelease-1>", self.get_suppliers)

self.display_data()
```

```
#####
def add(self): # creating a function to add supplier details to the database
    con = sqlite3.connect(database= r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        # makes sure that none of the entry fields are empty
        if self.var_supplier_id.get() == "" or self.var_supplier_email.get() == "" or self.var_supplier_name.get() == "":
            messagebox.showerror("Incorrect Input", "Please fill in all the details", parent = self.wind ) # validation. Cannot add supplier details without supplier ID
        else:
            cur.execute("Select * from supplier where ID=?",(self.var_supplier_id.get(),))
            row = cur.fetchone() # fetches the next row of data
            if row != None:
                messagebox.showerror("Error", "This supplier ID already exists", parent = self.wind)
            else:
                # checks that the supplier name only has alphabets
                if not self.var_supplier_name.get().isalpha():
                    messagebox.showerror("Incorrect Input", "Please enter a valid name", parent=self.wind)
                    return

                # ensures that the email format follows a specific pattern
                email_pattern = r'\b[A-Za-z0-9._%+-]*@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
                if not re.match(email_pattern, self.var_supplier_email.get()):
                    messagebox.showerror("Incorrect Input", "Please enter a valid email address", parent=self.wind)
                    return
```

```
# adds supplier details to the supplier table
cur.execute("Insert into supplier(ID, Name, Email) values(?, ?, ?)", (
    self.var_supplier_id.get(),
    self.var_supplier_name.get(),
    self.var_supplier_email.get(),
))

con.commit() # commit any changes
messagebox.showinfo("Added!!", "Successfully Added: " + ' ' + str(self.var_supplier_name.get()), parent = self.wind)
self.empty()

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent = self.wind)
```

```

def display_data(self): # display the data stored in the database
    con = sqlite3.connect(database = r'sms.db') # connecting it to the database
    cur = con.cursor() # database cursor
    try:
        cur.execute("select * from supplier") # sql query
        rows = cur.fetchall() # get all the records
        self.SupplyTable.delete(*self.SupplyTable.get_children())
        for row in rows: # for loop
            self.SupplyTable.insert('', END, values = row)

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

##### select function #####
def get_suppliers(self, suppliers): # this gets the row of data
    self.focus = self.SupplyTable.focus()
    self.data = self.SupplyTable.item((self.focus))
    row = self.data['values']
    self.var_supplier_id.set(row[0]), # indexing starts from 0
    self.var_supplier_name.set(row[1]),
    self.var_supplier_email.set(row[2])

```

```

##### modifying function #####
def modify(self): # creating a function to change the supplier details to the database
    con = sqlite3.connect(database= r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        if self.var_supplier_id.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter an existing supplier ID", parent=self.wind)
            # validation. Cannot modify supplier details without supplier ID
        else:
            cur.execute("Select * from supplier where ID=?", (self.var_supplier_id.get(),))
            row = cur.fetchone() # fetches the next row of data
            if row == None: # will only update if I have an existing supplier ID
                messagebox.showerror("Error", "This supplier ID is invalid") # validation
                return

            # checks that the supplier name only has alphabets
            if not self.var_supplier_name.get().isalpha():
                messagebox.showerror("Incorrect Input", "Please enter a valid name", parent=self.wind)
                return

            # ensures that the email format follows a specific pattern
            email_pattern = r'\b[A-Za-z0-9_.%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
            if not re.match(email_pattern, self.var_supplier_email.get()):
                messagebox.showerror("Incorrect Input", "Please enter a valid email address", parent=self.wind)
                return
    
```

```

else:
    # this updates the supplier details
    cur.execute("Update supplier SET Name=?, Email=? WHERE ID=?", # we need a where statement for our query
               self.var_supplier_name.get(),
               self.var_supplier_email.get(),
               self.var_supplier_id.get(),
               )
    con.commit() # commit any changes
    messagebox.showinfo("Updated!!", "Successfully updated Supplier!!", parent=self.wind)
    self.empty() # calling the function we made

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent = self.wind)

```

```
#####
Delete function #####
def delete_record(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_supplier_id.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter an existing supplier ID", parent=self.wind)
        else:
            cur.execute("Select * from supplier where ID = ?", (self.var_supplier_id.get(),))
            row = cur.fetchone()
            if row == None: # if there is no existing supplier IF
                messagebox.showerror("Error", "This supplier ID is invalid.")
            else:
                # confirmation message to delete
                ask = messagebox.askyesno("Confirmation", "Do you want to delete this record?", parent = self.wind)
                if ask == True:
                    cur.execute("Delete from supplier where ID=?", (self.var_supplier_id.get(),)) # sql query to delete supplier record
                    con.commit()
                    messagebox.showinfo("Delete", "Successfully deleted", parent = self.wind)
                    self.empty() # calls the empty function
    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)
```

```
#####
Empty function #####
def empty(self):
    self.var_supplier_id.set("") # this makes the form text disappear
    self.var_supplier_name.set("") # this makes the form text disappear
    self.var_supplier_email.set("") # this makes the form text disappear
    self.display_data()
```

```
#####
Search function #####
def search(self):
    con = sqlite3.connect(database=r'sms.db') # connecting it to the database
    cur = con.cursor() # database cursor
    try:
        if self.var_searchuse.get() == "select": # the user should select either name, email or ID
            messagebox.showerror("Error", "Select one of the options", parent = self.wind) # error due to no option selected
        elif self.var_searchtxt.get() == "": # if nothing is entered in the search bar
            messagebox.showerror("Error", " Please type what you want to search", parent=self.wind)
        else:
            cur.execute("select * from supplier where "+self.var_searchuse.get()+" LIKE '%"+self.var_searchtxt.get()+"%'") # whatever we search can be in any position
            rows = cur.fetchall() # get all the records
            if len(rows)!=0:
                self.SupplyTable.delete(*self.SupplyTable.get_children())
                for row in rows:
                    self.SupplyTable.insert('', END, values=row) # show the records that match the user's criteria
            else:
                messagebox.showerror("Error", "Nothing found", parent = self.wind)

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # this gives us any errors that may occur when coding
```

```
if __name__ == "__main__":
    wind = Tk()
    obj = SupplierClass(wind)
    wind.mainloop() # this executes Tkinter
```

Code for Product window

```

from tkinter import *    # this is used to create the GUI of my stock management system
import numpy as np
from tkinter import ttk, messagebox
import sqlite3
from tkinter import filedialog
from array import *
import numpy
import csv
class ProductClass: # creating a class called Product Class
    def __init__(self, wind):
        self.wind = wind
        self.wind.title("Products")
        window_width = 1550
        window_height = 750
        screen_width = wind.winfo_screenwidth()
        screen_height = wind.winfo_screenheight()
        center_x = int(130+(screen_width / 2 - window_width / 2)) # centre x
        center_y = int(50+(screen_height / 2 - window_height / 2)) # centre y

        # set the position of the window to the center of the screen
        self.wind.geometry(f'{window_width}x{window_height}+{center_x}+{center_y}')

```

```

# defining the variables
self.var_searchuse = StringVar() # stores the user's choice that they select from drop-down list
self.var_searchtxt = StringVar() # stores the user's input on the product search bar
self.var_product = StringVar() # stores the name of product
self.var_quantity = IntVar()
self.var_product_id = IntVar()

self.var_cat_name = StringVar()
self.var_Reorder_Point = IntVar()
self.var_cat_descr = StringVar()

self.var_supplier_ID = IntVar()

Search_box = LabelFrame(self.wind, text = "Search Product", bg = "lightblue", font = ("arial", 12, "bold"))
Search_box.place(x = 920, y = 270, width = 650, height = 80)

block = ttk.Combobox(Search_box, textvariable=self.var_searchuse, values=("select", "Product", "Product_ID"),
                     state="readonly", justify= CENTER, font=( "times new roman", 15))
block.place(x=10, y=5, width=200, height=30)
block.current(0)

txt_block = Entry(Search_box, textvariable=self.var_searchtxt, font=( "goudy old style", 15),
                  bg="lightyellow").place(x=250, y=5, height=30)
btn_block = Button(Search_box, text="search", command=self.search, font=( "goudy old style", 15),
                   bg="#734058", fg="white").place(x=530, y=5, width=100, height=30)

```

```
#####
# Sourcing widgets #####
#####

Supplier_Entry = Entry(self.wind, textvariable=self.var_supplier_ID, font = ("Rosewood Std Regular",15),
                      bg = "lightyellow", fg = "black").place(x = 40, y = 250, height = 60)

product_ID_1_entry = Entry(self.wind, textvariable= self.var_product_id,
                           font = ("Rosewood Std Regular",15), bg = "lightyellow", fg = "black").place(x = 40, y = 120, height = 60, width= 280)

Add_Sourcing_btn = Button(self.wind, text = "Add",
                          bg = "#734058", command = self.add_source,
                          font = ("OCR A Std", 15), fg = "yellow").place(x = 5, y = 320, height = 55, width = 100)

Delete_Sourcing_btn = Button(self.wind, text = "Delete",
                             bg = "#734058", command = self.delete__sourcing_record, font = ("OCR A Std", 15),
                             fg = "yellow").place(x = 125, y = 320, height = 55, width = 100)

update_Sourcing_btn = Button(self.wind, text = "Update", bg = "#734058",
                             command = self.update_sourcing_record,
                             font = ("OCR A Std", 15), fg = "yellow").place(x = 245, y = 320, height = 55, width = 100)
```

```
#####
# category widgets #####
#####

txt_category= Entry(self.wind, textvariable= self.var_cat_name,
                     font = ("Rosewood Std Regular",15), bg = "lightyellow", fg = "black").place(x = 450, y = 120, height = 60)

Add_btn = Button(self.wind, text = "Add", command = self.add, bg = "#734058",
                 font = ("OCR A Std", 15), fg = "yellow").place(x = 375, y = 320, height = 55, width = 100)

descr_category = Entry(self.wind, textvariable= self.var_cat_descr,
                      font = ("Rosewood Std Regular",15), bg = "lightyellow",
                      fg = "black").place(x = 450, y = 220, height = 60)

Dlt_btn = Button(self.wind, text = "Delete", command = self.delete_record, bg = "#734058",
                  font = ("OCR A Std", 15), fg = "yellow").place(x = 575, y = 320, height = 55, width = 100)

Update_btn = Button(self.wind, text = "Update", command = self.update_category, bg = "#734058",
                     font = ("OCR A Std", 15), fg = "yellow").place(x = 775, y = 320, height = 55, width = 100)
```

```
#####
# product widgets #####
#####

txt_product = Entry(self.wind, textvariable= self.var_product,
                    font = ("Rosewood Std Regular",15), bg = "lightyellow",
                    fg = "black").place(x = 1300, y = 550, height = 60, width = 200)

Add_product_btn = Button(self.wind, text = "Add", command = self.add_product,
                        bg = "#734058", font = ("OCR A Std", 15),
                        fg = "yellow").place(x = 930, y = 380, height = 55, width = 100)

Update_product_btn = Button(self.wind, text = "Update",
                            command = self.update_product, bg = "#734058", font = ("OCR A Std", 15),
                            fg = "yellow").place(x = 1150, y = 380, height = 55, width = 100 )

Delete_product_btn = Button(self.wind, text = "Delete", command = self.delete_product, bg = "#734058",
                            font = ("OCR A Std", 15), fg = "yellow").place(x = 1400, y = 380, height = 55, width = 100 )

quantity_product = Entry(self.wind, textvariable= self.var_quantity,
                        font = ("Rosewood Std Regular",15), bg = "lightyellow",
                        fg = "black").place(x = 1025, y = 650, height = 60, width=100)

reorder_point_product = Entry(self.wind, textvariable= self.var_Reorder_Point,
                             font = ("Rosewood Std Regular",15),
                             bg = "lightyellow", fg = "black").place(x = 1200, y = 450, height = 60, width= 100)
```

```

product_ID_entry = Entry(self.wind, textvariable= self.var_product_id,
                        font = ("Rosewood Std Regular",15),
                        bg = "lightyellow", fg = "black").place(x = 1450, y = 450, height = 60, width= 80)

category_entry = Entry(self.wind, textvariable= self.var_cat_name, font = ("Rosewood Std Regular",15),
                        bg = "lightyellow", fg = "black").place(x = 1350, y = 650, height = 60, width= 200)

lbl_product_id = Label(self.wind, text = "ID", font = ("Rosewood Std Regular",15), bg = "#4cafaf",
                        fg = "black").place(x=1350, y = 450, height = 60, width = 75)

lbl_reorder_point = Label(self.wind, text = "Reorder Point", font = ("Rosewood Std Regular",15),
                           bg = "#4cafaf", fg = "black").place(x=925, y = 450, height = 60, width = 250)

lbl_quantity = Label(self.wind, text = "Quant", font = ("Rosewood Std Regular",15),
                      bg = "#4cafaf", fg = "black").place(x=925, y = 650, height = 60, width = 75)

lbl_category = Label(self.wind, text = "Category", font = ("Rosewood Std Regular",15),
                      bg = "#4cafaf", fg = "black").place(x=1175, y = 650, height = 60, width = 150)

lbl_product = Label(self.wind, text = "Product", font = ("Rosewood Std Regular",15),
                     bg = "#4cafaf", fg = "black").place(x=1000, y = 550, height = 60, width = 150)

```

```

#####
# Frames #####
#####

sourcing_frame = Frame(self.wind, bd = 7, relief = GROOVE, highlightcolor="red", bg = "#734058", cursor = "circle", height=2000, width=20 )
sourcing_frame.place(x = 50, y = 400)

cat_frame = Frame(self.wind, bd = 7, relief = GROOVE, highlightcolor="red", bg = "#734058", cursor = "circle", height=2000, width=20 )
cat_frame.place(x = 450, y = 400)

Entry_Frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle",height=2000, width=20)
Entry_Frame.place(x=350, y=0)

Product_Frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle",height=2000, width=20)
Product_Frame.place(x = 900, y = 0)

Product_Table_Frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle",height=2000, width=20)
Product_Table_Frame.place(x = 910, y = 0)

Add_CAT_lbl = Label(self.wind, text="Add Category", bg="lightblue", font=("arial", 15, "bold"))
Add_CAT_lbl.place(x=450, y=20, width=300, height=60)

Add_Source_lbl = Label(self.wind, text = "Add Source", bg="lightblue", font=("arial", 15, "bold"))
Add_Source_lbl.place(x = 30, y = 20, width = 300, height = 60)

```

```
#####
# Product Treeview #####
#####

scrollly = Scrollbar(Product_Table_Frame, orient=VERTICAL)
scrollx = Scrollbar(Product_Table_Frame, orient=HORIZONTAL)
self.Product_Table = ttk.Treeview(Product_Table_Frame,
                                  columns=("Product_ID", "CAT_NAME", "Product", "Reorder_Point", "Quantity"),
                                  yscrollcommand=scrollly.set, xscrollcommand=scrollx.set)

scrollx.pack(side=BOTTOM, fill=X)
scrolly.pack(side=RIGHT, fill=Y)
scrollx.config(command=self.Product_Table.xview)
scrolly.config(command=self.Product_Table.yview)

self.Product_Table.heading("Product_ID", text="Product ID")
self.Product_Table.heading("CAT_NAME", text="Category")
self.Product_Table.heading("Product", text="Product")
self.Product_Table.heading("Reorder_Point", text="Reorder Point")
self.Product_Table.heading("Quantity", text="Stock Quantity")
self.Product_Table["show"] = "headings"

self.Product_Table.column("Product_ID", width=120, stretch=NO) # fixed width
self.Product_Table.column("CAT_NAME", width=120, stretch=NO) # fixed width
self.Product_Table.column("Product", width=120, stretch=NO) # fixed width
self.Product_Table.column("Reorder_Point", width=120, stretch=NO) # fixed width
self.Product_Table.column("Quantity", width=120, stretch=NO) # fixed width
```

```
self.Product_Table.pack(fill=BOTH, expand=1)
self.Product_Table.bind("<ButtonRelease-1>", self.get_Product)
self.display_Product_data()

#####
# Category Treeview #####
#####

myscrolly = Scrollbar(cat_frame, orient=VERTICAL)
myscrollx = Scrollbar(cat_frame, orient=HORIZONTAL)

self.Cat_Table = ttk.Treeview(cat_frame, columns=("CAT_NAME", "CAT_DESCR"), yscrollcommand=myscrolly.set, xscrollcommand=myscrollx.set)

myscrollx.pack(side=BOTTOM, fill=X)
myscrolly.pack(side=RIGHT, fill=Y)
myscrollx.config(command=self.Cat_Table.xview)
myscrolly.config(command=self.Cat_Table.yview)

self.Cat_Table.heading("CAT_NAME", text="CAT Name")
self.Cat_Table.heading("CAT_DESCR", text = "CAT Descr")
self.Cat_Table["show"] = "headings"

self.Cat_Table.column("CAT_NAME", width =140, stretch=NO) # fixed width
self.Cat_Table.column("CAT_DESCR", width = 140, stretch=NO) # fixed width

self.Cat_Table.pack(fill=BOTH, expand=1)
self.Cat_Table.bind("<ButtonRelease-1>", self.get_category)
```

```
#####
# Sales Treeview #####
#####

source_scrolly = Scrollbar(sourcing_frame, orient=VERTICAL) # vertical scrollbar
source_scrollx = Scrollbar(sourcing_frame, orient=HORIZONTAL) # horizontal scrollbar

self.sourcing_table = ttk.Treeview(sourcing_frame, columns=("Product_ID", "Supplier_ID"), yscrollcommand=source_scrolly.set, xscrollcommand=source_scrollx.set)
source_scrollx.pack(side=BOTTOM, fill=X)
source_scrolly.pack(side = RIGHT, fill=Y)

source_scrollx.config(command=self.sourcing_table.xview)
source_scrolly.config(command=self.sourcing_table.yview)

self.sourcing_table.heading("Product_ID", text="Product ID")
self.sourcing_table.heading("Supplier_ID", text="Supplier ID")
self.sourcing_table["show"] = "headings"

self.sourcing_table.column("Product_ID", width=120, stretch=NO)
self.sourcing_table.column("Supplier_ID", width=120, stretch=NO)
self.sourcing_table.pack(fill=BOTH, expand=1)
self.sourcing_table.bind("<ButtonRelease-1>", self.get_sourcing)
self.display_source_data()
```

```

        self.display_data()

##### Sourcing Treeview functions #####
def add_source(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_product_id.get() == "" or self.var_supplier_ID.get() == "": # if there is no input
            messagebox.showerror("Error", "Please fill in all the forms", parent = self.wind)
            return
        else:
            # since supplier to product is a one to many relationship
            # no product should have more than 1 supplier
            cur.execute("Select * from Sourcing where Product_ID=?", (self.var_product_id.get(),))
            row = cur.fetchone()
            if row != None:
                messagebox.showerror("Error", "This Product ID is invalid", parent = self.wind)
                return
            ##### This checks to see if the Product ID is existing or not #####
            cur.execute("Select Product_ID from Product where Product_ID=?", (self.var_product_id.get(),))
            fetch_product_ID = cur.fetchone()
            if fetch_product_ID == None:
                messagebox.showerror("Error", "This is not an existing Product ID", parent = self.wind)
                return
    
```

```

cur.execute("SELECT ID from supplier where ID=?", (self.var_supplier_ID.get(),))
fetch_Supplier_ID = cur.fetchone()
if fetch_Supplier_ID == None:
    messagebox.showerror("Error", "This is not an existing Supplier ID", parent=self.wind)
    return
else:
    ## inserting product ID and supplier ID
    cur.execute("Insert into Sourcing(Product_ID,Supplier_ID) values(?,?)",
               (self.var_product_id.get(),
                self.var_supplier_ID.get()))
    con.commit()
    messagebox.showinfo("Successfully added", "Successfully linked supplier and product", parent = self.wind)
    self.display_source_data()

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

```

def display_source_data(self): # display the data stored in the database
    con = sqlite3.connect(database = r'sms.db') # connecting it to the database
    cur = con.cursor() # database cursor
    try:
        cur.execute("select * from Sourcing") # sql query
        rows = cur.fetchall() # get all the records
        self.sourcing_table.delete(*self.sourcing_table.get_children())
        for row in rows: # for loop
            self.sourcing_table.insert('', END, values = row)

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

```

def update_sourcing_record(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_product_id.get() == "" or self.var_supplier_ID.get() == "": # if fields are empty
            messagebox.showerror("Incomplete", "Please select a record from the treeview", parent = self.wind)
            return
    else:
        # this checks if there is an existing product ID in the product table
        cur.execute("Select * from Sourcing where Product_ID = ? ",(self.var_product_id.get(),))
        row = cur.fetchone()
        if row == None:
            messagebox.showerror("Error", "Please choose an existing Product ID", parent = self.wind)
            return
        # this checks if there is an existing Supplier ID in the supplier table
        cur.execute("Select * from Sourcing where Supplier_ID =? ",(self.var_supplier_ID.get(),))
        row = cur.fetchone()
        if row == None:
            messagebox.showerror("Error", "Please choose an existing Supplier ID", parent = self.wind)
            return

```

```

    else:
        cur.execute("Update Sourcing SET Supplier_ID=? WHERE Product_ID=? ", ( # we need a where statement for our query
            self.var_supplier_ID.get(),
            self.var_product_id.get()
        ))
        con.commit()
        messagebox.showinfo("Success", "Successfully updated", parent = self.wind)
        self.display_source_data()
except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # shows the error

```

```

def delete_sourcing_record(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_product_id.get() == "": # if product entry box is empty
            messagebox.showerror("Incorrect Input", "Please enter an existing Product ID", parent=self.wind)
            return
    else:
        cur.execute("Select * from Sourcing where Product_ID = ? ",(self.var_product_id.get(),)) # using product id as part of the where clause
        row = cur.fetchone()
        if row == None: # if product ID doesn't exist
            messagebox.showerror("Error", "This Product ID is invalid")
        else:
            ask = messagebox.askyesno("Confirmation","Do you want to delete this record?", parent = self.wind) # confirmation
            if ask == True:
                cur.execute("Delete from Sourcing where Product_ID=? ",(self.var_product_id.get(),)) # the ID is the primary key
                con.commit()
                messagebox.showinfo("Delete", "Successfully deleted", parent = self.wind)
                self.display_source_data() # automatically deletes it without refreshing the tkinter window
    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # shows the error

```

```

def get_sourcing(self, sourcing): # this gets the row of data
    self.focus = self.sourcing_table.focus()
    self.data = self.sourcing_table.item((self.focus))
    row = self.data['values']
    self.var_supplier_ID.set(row[1])
    self.var_product_id.set(row[0])

```

```
#####
# Category Treeview functions #####
#####

def add(self): # creating a function to add category details to the database
    con = sqlite3.connect(database=r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        if self.var_cat_name.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter a category",
                                  parent=self.wind) # validation.

        if not self.var_cat_name.get().isalpha(): # checks if the value is in alphabets
            messagebox.showerror("Incorrect Input", "Please enter a valid category name", parent=self.wind)
            return

        if not self.var_cat_descr.get().isalpha(): # checks if the value is in alphabets
            messagebox.showerror("Incorrect Input", "Please enter a valid category description", parent=self.wind)
            return

    else:
```

```
        cur.execute("Select * from category where UPPER(CAT_NAME)=?", (self.var_cat_name.get().upper(),))
        row = cur.fetchone() # fetches the next row of data
        if row != None:
            messagebox.showerror("Error", "The Category already exists", parent = self.wind)
            # parent = self.wind ensures that even if an error occurs it will not return to the dashboard window
            return
        else:
            cur.execute("Insert into category(CAT_NAME, CAT_DESCR) values(?,?)", ( # inserting into category table
                self.var_cat_name.get(),
                self.var_cat_descr.get()
            ))
            con.commit() # commit any changes
            messagebox.showinfo("Added!!", "Successfully Added Category!!", parent=self.wind)
            self.display_data()
    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)
```

```
def delete_record(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_cat_descr.get() == "" and self.var_cat_name == "": # if the entry boxes are empty
            messagebox.showerror("Incorrect Input", "Please choose a category and its description", parent=self.wind)
        else:
            cur.execute("Select * from category where CAT_NAME = ?", (self.var_cat_name.get(),))
            row = cur.fetchone()
            if row == None:
                messagebox.showerror("Error", "This Category name is invalid")
            else:
                # confirmation message
                ask = messagebox.askyesno("Confirmation", "Do you want to delete this record?", parent = self.wind)
                if ask == True:
                    cur.execute("Delete from category where CAT_NAME=?",(self.var_cat_name.get(),)) # the ID is the primary key
                    con.commit()
                    messagebox.showinfo("Delete", "Successfully deleted", parent = self.wind)
                    self.CAT_empty()

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)
```

```
def update_category(self):
    con = sqlite3.connect(database=r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        if self.var_cat_name.get() == "" and self.var_cat_descr.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter an existing Category", parent=self.wind)
        else:
            cur.execute("Select * from category where Upper(CAT_NAME)=?", (self.var_cat_name.get().upper(),))
            row = cur.fetchone() # fetches the next row of data
            if row == None: # if there is no existing category
                messagebox.showerror("Error", "This CAT name is invalid. Only existing category names can be updated", parent = self.wind) # validation
```

```

        else:
            cur.execute("Update category SET CAT_DESCR =? WHERE CAT_NAME=?",
                        (self.var_cat_descr.get(),
                         self.var_cat_name.get()))
        con.commit()
        messagebox.showinfo("Updated!!", "Successfully updated Category Description!!", parent=self.wind)
        self.display_data() # this helps to display the data. Once the update has been made, the changed data will be displayed automatically

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent = self.wind)

```

```

    def display_data(self): # display the data stored in the database
        con = sqlite3.connect(database = r'sms.db') # connecting it to the database
        cur = con.cursor() # database cursor
        try:
            cur.execute("select * from category") # sql query
            rows = cur.fetchall() # get all the records
            self.Cat_Table.delete(*self.Cat_Table.get_children())
            for row in rows: # for loop
                self.Cat_Table.insert('', END, values = row)

        except Exception as ex:
            messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

    def get_category(self, category): # this gets the row of data
        self.focus = self.Cat_Table.focus()
        self.data = self.Cat_Table.item((self.focus))
        row = self.data['values']
        self.var_cat_name.set(row[0])
        self.var_cat_descr.set(row[1])

    def CAT_empty(self):
        self.var_cat_name.set("")
        self.var_cat_descr.set("")
        self.display_data()

```

```

#####
# Product treeview functions #####
#####

def add_product(self): # creating a function to add product details to the table
    con = sqlite3.connect(database=r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        # if no input is found in the product entry box
        if self.var_product.get() == "":
            messagebox.showerror("Incorrect Input", "Please enter a Product",
                                parent=self.wind)
            return

        # checks whether the product input is in alphabets or not
        if not self.var_product.get().isalpha():
            messagebox.showerror("Incorrect Input", "Please enter the correct data type for product", parent=self.wind)
            return
    
```

```

        else:
            cur.execute("Select * from Product where Upper(Product) =?", (self.var_product.get().upper(),))
            row = cur.fetchone() # fetches the next row of data
            if row != None:
                messagebox.showerror("Error", "This Product already exists", parent = self.wind)
                return
            cur.execute("Select CAT_NAME from category where Upper(CAT_NAME) =?",(self.var_cat_name.get().upper(),))
            fetch = cur.fetchone()
            if fetch == None: # if there is no existing category
                messagebox.showerror("Error", "Please add an existing category", parent = self.wind)
            else:
                cur.execute(
                    # adding to the product table
                    "Insert into Product( CAT_NAME, Product, Reorder_Point, Quantity) values(?, ?, ?, ?)",
                    (
                        self.var_cat_name.get(),
                        self.var_product.get(),
                        self.var_Reorder_Point.get(),
                        self.var_quantity.get(),
                    )
                )
                con.commit() # commit any changes
                messagebox.showinfo("Added!!", "Successfully Added Product!!", parent=self.wind)
                self.display_Product_data()

```

```

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) ##### shows an error

#####
# Updating Product #####
#####

def update_product(self):
    con = sqlite3.connect(database=r'sms.db') # connecting to sqlite
    cur = con.cursor() # This creates a cursor
    try:
        if self.var_product_id.get() == "": # if the product ID selected is not reflected in the entry field
            messagebox.showerror("Incorrect Input", "Please select an existing Product ID", parent=self.wind)
            return

        if self.var_product.get() == "": # if there is no input in the product entry
            messagebox.showerror("Incorrect Input", "Please select an existing Product ", parent=self.wind)
            return

        if self.var_cat_name.get() == "": # if there is no category in the category name entry
            messagebox.showerror("Incorrect Input", "Please select a existing Category ", parent=self.wind)
            return

        # checks whether the product input is in alphabets or not
        if not self.var_product.get().isalpha():
            messagebox.showerror("Incorrect Input", "Please enter the correct data type for product",parent=self.wind)
            return
    
```

```

    cur.execute("Select CAT_NAME from category where upper(CAT_NAME) =?", (self.var_cat_name.get().upper(),))
    fetch = cur.fetchone()
    if fetch == None:
        messagebox.showerror("Error", "Please update to an existing category", parent=self.wind)
        return
    else:
        cur.execute("Select * from Product where Product_ID=? ", (self.var_product_id.get(),))
        row = cur.fetchone() # fetches the next row of data
        if row == None: # will only update if I have the Product ID
            messagebox.showerror("Error", "This Product ID is invalid") # validation
            return

        else:
            ##### updating product table #####
            cur.execute("Update Product SET CAT_NAME=?, Product=?, Reorder_Point=?, Quantity=? WHERE Product_ID=?", ( # we need a where statement for our query
                self.var_cat_name.get(),
                self.var_product.get(),
                self.var_Reorder_Point.get(),
                self.var_quantity.get(),
                self.var_product_id.get(),
            )

```

```

        })
        con.commit()
        messagebox.showinfo("Updated!!", "Successfully updated Product!!", parent=self.wind)
        self.display_Product_data() # this helps to display the data. Once the update has been made, the changed data will be displayed automatically

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent = self.wind)

```

```

def delete_product(self):
    con = sqlite3.connect(database=r'sms.db')
    cur = con.cursor()
    try:
        if self.var_product_id.get() == "": # if entry box for product ID is blank
            messagebox.showerror("Incorrect Input", "Please choose a Product ID to be deleted", parent=self.wind)
        else:
            cur.execute("Select * from Product where Product_ID = ? ",(self.var_product_id.get(),)) # uses product id to fetch the existing record
            row = cur.fetchone()
            if row == None:
                messagebox.showerror("Error", "This Product ID is invalid", parent = self.wind)
            else:
                # confirmation of deletion
                ask = messagebox.askyesno("Confirmation","Do you want to delete this record?", parent = self.wind)
                if ask ==True:
                    cur.execute("Delete from Product where Product_ID=? ",(self.var_product_id.get(),)) # the ID is the primary key
                    con.commit()
                    messagebox.showinfo("Delete", "Successfully deleted", parent = self.wind)
                    self.display_Product_data()

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

```

def search(self):
    con = sqlite3.connect(database=r'sms.db') # connecting it to the database
    cur = con.cursor() # database cursor
    try:
        if self.var_searchuse.get() == "select": # the user should select either product name or ID
            messagebox.showerror("Error", "Select one of the options", parent = self.wind) # error due to incorrect search
        elif self.var_searchtxt.get() == "": # if the user doesn't type anything in the product search bar
            messagebox.showerror("Error", " Please type what you want to search", parent=self.wind)
        else:
            cur.execute("select * from Product where "+self.var_searchuse.get()+" LIKE '%"+self.var_searchtxt.get()+"%'") # whatever we search can be in any position
            rows = cur.fetchall() # get all the records
            if len(rows)!=0:
                self.Product_Table.delete(*self.Product_Table.get_children())
                for row in rows:
                    self.Product_Table.insert('', END, values=row) # show records that match user's input

            else:
                messagebox.showerror("Error", "Nothing found", parent = self.wind)

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # this gives us any errors that may occur when coding

```

```

def display_Product_data(self): # display the data stored in the database
    con = sqlite3.connect(database=r'sms.db') # connecting it to the database
    cur = con.cursor() # database cursor
    try:
        cur.execute("select * from Product") # sql query
        rows = cur.fetchall() # get all the records
        self.Product_Table.delete(*self.Product_Table.get_children())
        for row in rows: # for loop
            self.Product_Table.insert('', END, values=row)

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

def get_Product(self, product): # this gets the row of data
    self.focus = self.Product_Table.focus()
    self.value = self.Product_Table.item((self.focus))
    row = self.value['values']
    self.var_product_id.set(row[0])
    self.var_cat_name.set(row[1])
    self.var_product.set(row[2])
    self.var_Reorder_Point.set(row[3])
    self.var_quantity.set(row[4])

```

```

def empty(self):
    self.var_product_id.set("") # this makes the form text disappear
    self.var_cat_name.set("")
    self.var_product.set("")
    self.var_Reorder_Point.set("")
    self.var_quantity.set("")
    self.display_data()

```

```

def update_stock_levels():
    connection = sqlite3.connect(r'sms.db') # connecting to the database
    cursor = connection.cursor() # creating a cursor object

    with open('sales.csv', 'r') as f: # opening the csv file and reading its contents
        reader = csv.reader(f)
        for row in reader: # iterating through each row in the csv file
            product_id = row[1] # getting the Product ID from the csv file and storing it in a variable
            sold_quantity = row[2] # getting the sold quantity from the CSV file and storing it in a variable
            if not sold_quantity: # if there is no sold quantity, then it should skip to the next row
                continue
            sold_quantity = int(sold_quantity) # converting the sold quantity to an integer type

            # gets the current stock quantity from the Product table based on the Product ID in the CSV file
            cursor.execute("SELECT Quantity FROM Product WHERE Product_ID = ?", (product_id,))
            current_stock = cursor.fetchone()[0]
            updated_stock = max(current_stock - sold_quantity, 0) # subtraction to get the updated stock levels for a product
            # updated stock stores the updated quantity value remaining for a product
            # product ID stores the Product ID from the CSV file
            # Below is the query to update the product stock quantity using the Product ID as part of the where condition
            cursor.execute("UPDATE Product SET Quantity = ? WHERE Product_ID = ?", (updated_stock, product_id))

    connection.commit() # commit the changes
    connection.close()

```

```

if __name__ == "__main__":
    wind = Tk()
    obj = ProductClass(wind)
    wind.mainloop()

```

Code for Sales window

```
from tkinter import *      # this is used to create the GUI of my stock management system
import numpy as np
from tkinter import ttk, messagebox
import sqlite3
import csv
import matplotlib.pyplot as plt # allows graphs to be made
from product import *
import pandas as pd
from product import ProductClass
import datetime
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg # provides a canvas that can display graph in tkinter window
import tkinter as tk
from statsmodels.tsa.arima.model import ARIMA # allows ARIMA model to be used
```

```
class SalesClass: # creating a class called SalesClass
    def __init__(self, wind):
        self.wind = wind
        self.wind.title("Sales records")

        # used to set the dimensions of the sales window
        window_width = 1550
        window_height = 750
        screen_width = wind.winfo_screenwidth()
        screen_height = wind.winfo_screenheight()
        center_x = int(130+(screen_width / 2 - window_width / 2)) # centre x
        center_y = int(50+(screen_height / 2 - window_height / 2)) # centre y

        self.var_product_ID = IntVar() # variable that I am going to use

        product_id_label = Label(self.wind, text = "Enter Product ID", # label for telling the user where to enter the product ID
                               font = ("Rosewood Std Regular",15), bg = "#4CAF50", fg = "black").place(x=30, y = 50, height = 60, width = 250) # positioning of labels

        txt_product = Entry(self.wind, textvariable= self.var_product_ID, # entry box to enter the Product ID
                           font = ("Rosewood Std Regular",15), bg = "lightyellow", fg = "black").place(x = 330, y = 50, height = 60)

        # set the position of the window to the center of the screen
        self.wind.geometry(f'{window_width}x{window_height}+{center_x}+{center_y}')
```

```
sales_frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=2000, width=2000)
sales_frame.place(x=60, y=280)

border_middle_frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=2000, width=20)
border_middle_frame.place(x=695, y=10) # places the middle frame into a specific coordinate

scrollx = Scrollbar(sales_frame, orient=VERTICAL) # vertical scrollbars
scrolly = Scrollbar(sales_frame, orient=HORIZONTAL) # horizontal scrollbars

#####
# Sales Treeview #####
self.Sales_Table = ttk.Treeview(sales_frame,
                                columns=("Date", "Product_ID", "Sold", "Value"),
                                yscrollcommand=scrolly.set, xscrollcommand=scrollx.set)

scrollx.pack(side=BOTTOM, fill=X)
scrolly.pack(side=RIGHT, fill=Y)
scrollx.config(command=self.Sales_Table.xview)
scrolly.config(command=self.Sales_Table.yview)
```

```

self.Sales_Table.heading("Date", text="Date")
self.Sales_Table.heading("Product_ID", text="Product ID")
self.Sales_Table.heading("Sold", text="Sold")
self.Sales_Table.heading("Value", text="Value £")
self.Sales_Table["show"] = "headings"

self.Sales_Table.column("Date", width=130, stretch=NO) # fixed width
self.Sales_Table.column("Product_ID", width=130, stretch=NO) # fixed width
self.Sales_Table.column("Sold", width=130, stretch=NO) # fixed width
self.Sales_Table.column("Value", width=130, stretch=NO) # fixed width

self.Sales_Table.pack(fill=BOTH, expand=1)

#####
Quantity_sold_btn = Button(self.wind, text = "Quantity Graph", command = self.display_sold_quantity_graph,
                           bg = "#734058", font = ("OCR A Std", 15), fg = "yellow").place(x = 35, y = 650, height = 55, width = 270)

#####
Import_file_btn = Button(self.wind, text="Import file",bg = "#734058", command = self.import_files,
                         font = ("OCR A Std", 15), fg = "yellow").place(x = 160, y = 200, height = 55, width = 370) # buttons that will import the file

#####
Value_(£)_btn = Button(self.wind, text = "Value Graph", command = self.display_value_graph,
                       bg = "#734058", font = ("OCR A Std", 15), fg = "yellow").place(x = 350, y = 650, height = 55, width = 270)

```

```

def import_files(self):
    connection = sqlite3.connect(r'sms.db') # connecting to the database
    cursor = connection.cursor() # creating a cursor object
    with open("sales.csv") as file:
        contents = csv.reader(file) # csv reader object - reading the contents of the csv file
        insert_records = "INSERT INTO Sales (Date, Product_ID, Sold, Value) VALUES(?, ?, ?, ?)" # this imports the file data into the table
        for row in contents:
            print(row)
            try:
                datetime.datetime.strptime(row[0].strip(), '%Y-%m-%d') # converts the first column in the sales csv file to a datetime object
            except ValueError:
                result = messagebox.showerror("Error", "Please check date is formatted properly", parent=self.wind) # this checks that the date is in the correct format
                if result == "ok":
                    messagebox.destroy() # if user presses ok, then the messagebox should disappear
                    continue
            # Check if a record already exists for the current Date and Product_ID
            select_existing = "SELECT * FROM Sales WHERE Date=? AND Product_ID=?"
            existing_record = cursor.execute(select_existing, (row[0], row[1])).fetchone()

```

```

if existing_record is not None:
    # Record already exists, so update the Sold and Value columns
    update_record = "UPDATE Sales SET Sold=Sold+?, Value=Value+? WHERE Date=? AND Product_ID=?"
    cursor.execute(update_record, (row[2], row[3], row[0], row[1]))
else:
    # Record does not exist, so insert a new record
    cursor.execute(insert_records, row)
select_all = "SELECT * FROM Sales WHERE Date > (SELECT DATE('now', '-7 day'))" # sql statement to select all the fields in the sales table within the last 7 days
rows = cursor.execute(select_all).fetchall()
for r in rows: # using the for loop to iterate through the values stored in the row variable
    self.Sales_Table.insert('', END, values=r) # inserting into sales treeview
connection.commit() # commit changes to the database
ProductClass.update_stock_levels() # calling the function to update stock levels in product class

```

```

def forecast_Quantity_sold_for_next_month(self):
    conn = sqlite3.connect(r'sms.db') # connecting to database
    # Date and sum of all sold quantity within that month for a given product ID
    query = "SELECT strftime('%Y-%m', Date) AS Month, SUM(Sold) AS Total_Sales FROM Sales WHERE Product_ID =? GROUP by Month ORDER BY Month"
    df_sales = pd.read_sql_query(query, conn, params=[self.var_product_ID.get()])
    df_sales.index = pd.to_datetime(df_sales['Month']) # sets the index of the dataframe to the Month column
    df_sales = df_sales.drop(columns=['Month']) # drops month column as it is no longer needed
    df_sales = df_sales.asfreq('MS') # sets frequency of the data to monthly
    df_sales.fillna(0, inplace=True) # if data is blank, fill it with 0

    model = ARIMA(df_sales, order=(1,1,1), freq='MS') # p=1, d=1, q=1
    result = model.fit() # fit the model with the sales data

    # Forecast future sales
    forecast = result.forecast(steps=1, typ='levels') # forecast for the next month

    message = f"The total sales (in terms of quantity) forecast for Product ID {self.var_product_ID.get()} next month is: {forecast[0]:.2f}"
    messagebox.showinfo("Sales Forecast", message, parent=self.wind) # message to forecast total quantity sold for the given Product ID

```

```

def forecast_value_in_pounds_for_next_month(self):
    conn = sqlite3.connect(r'sms.db') # connecting to the database
    # Date and sum of all value made for a given product ID per month
    query = "SELECT strftime('%Y-%m', Date) AS Month, SUM(Value) AS Total_Value FROM Sales WHERE Product_ID =? GROUP by Month ORDER BY Month"
    df_sales = pd.read_sql_query(query, conn, params=[self.var_product_ID.get()])
    df_sales.index = pd.to_datetime(df_sales['Month']) # sets index of the dataframe to the Month column
    df_sales = df_sales.drop(columns=['Month']) # drops month column as it is no longer needed
    df_sales = df_sales.asfreq('MS') # sets frequency of the data to monthly
    df_sales.fillna(0, inplace=True) # if data is blank, fill it with 0

    model = ARIMA(df_sales, order=(1, 1, 1), freq='MS') # p=1, d=1, q=1
    result = model.fit() # fit the model with the sales data

    # Forecast future sales
    forecast = result.forecast(steps=1, typ='levels')

    message = f"The total value (£) forecast for Product ID {self.var_product_ID.get()} next month is: {forecast[0]:.2f}"
    messagebox.showinfo("Value Forecast", message, parent=self.wind) # message to forecast total value for next month for the given product ID

```

```

def forecast_value_in_pounds_tomorrow(self):
    conn = sqlite3.connect(r'sms.db') # connect to the database
    # sql statement to fetch the Date, and value records that are then ordered by their date
    query = "SELECT strftime('%Y-%m-%d', Date) AS Date, SUM(Value) AS Total_Value FROM Sales WHERE Product_ID =? GROUP by Date ORDER BY Date"
    # params argument used to pass the Product ID obtained from its variable as a parameter to the query
    df_sales = pd.read_sql_query(query, conn, params=[self.var_product_ID.get()])

    df_sales.index = pd.to_datetime(df_sales['Date']) # sets the index of the dataframe to the Date column
    df_sales = df_sales.drop(columns=['Date']) # drops date column as it is no longer needed
    df_sales = df_sales.asfreq('D') # sets the frequency of the data to daily
    df_sales.fillna(0, inplace=True) # fills missing values with 0

    # using an arima model for the sales data
    model = ARIMA(df_sales, order=(1, 1, 1), freq='D') # p=1, d=1, q=1
    result = model.fit() # fits the model

    # Forecast future sales
    forecast = result.forecast(steps=1, typ='levels') # forecast for the next day
    message = f"The total value (£) forecast for Product ID {self.var_product_ID.get()} for the next day is: {forecast[0]:.2f}"
    messagebox.showinfo("Value Forecast", message, parent=self.wind) # messagebox showing forecast in value for tomorrow

```

```

def forecast_quantity_sold_tomorrow(self):
    conn = sqlite3.connect(r'sms.db') # connect to the database
    # sql statement to fetch the Date, and Sold records that are then ordered by their date
    query = "SELECT strftime('%Y-%m-%d', Date) AS Date, SUM(Sold) AS Total_Sold FROM Sales WHERE Product_ID =? GROUP by Date ORDER BY Date"
    # params argument used to pass the Product ID obtained from its variable as a parameter to the query
    df_sales = pd.read_sql_query(query, conn, params=[self.var_product_ID.get()])

    df_sales.index = pd.to_datetime(df_sales['Date']) # sets the index of the dataframe to the Date column
    df_sales = df_sales.drop(columns=['Date']) # drops date column as it is no longer needed
    df_sales = df_sales.asfreq('D') # sets the frequency of the data to daily
    df_sales.fillna(0, inplace=True) # fills missing values with 0

    # using an arima model for the sales data
    model = ARIMA(df_sales, order=(1, 1, 1), freq='D') # p=1, d=1, q=1
    result = model.fit() # fits the model

    # Forecast future sales
    forecast = result.forecast(steps=1, typ='levels') # forecast for the next day
    message = f"The total quantity sold forecast for Product ID {self.var_product_ID.get()} for the next day is: {forecast[0]:.2f}"
    messagebox.showinfo("Quantity sold Forecast", message, parent=self.wind) # messagebox showing forecast in Quantity sold for tomorrow

```

```

def display_value_graph(self):
    # create a Matplotlib figure with a size of 8.1* 5 inches
    fig = plt.Figure(figsize=(8.1, 6), dpi=100)
    ax = fig.add_subplot(111)

    # fetch data from the database and add it to the figure
    connection = sqlite3.connect(r'sms.db') # creating a database connection
    cursor = connection.cursor()
    select_value = cursor.execute("Select Date, Value FROM Sales WHERE Date > (SELECT DATE('now','-7 day')) and "
                                  "Product_ID =?" , (self.var_product_ID.get(),)).fetchall() # getting data within the last 7 days
    dates = [] # dates list
    value = [] # value list
    for row in select_value:
        dates.append(row[0]) # adding to dates list
        value.append(row[1]) # adding to value list
    # create a bar chart of the data
    ax.bar(dates, value, color='g', width=0.60, label="Value £")
    ax.set_xlabel('Date')
    ax.set_ylabel('Value')

```

```

# get the name of the Product from the Product table
name_of_product = cursor.execute("Select Product FROM Product WHERE Product_ID =?", (self.var_product_ID.get(),)).fetchone()[0]
ax.set_title(f'The value of {name_of_product} (£)') # title with name of product
ax.legend(loc="lower right")

# create a Tkinter canvas that can display the Matplotlib figure
canvas = FigureCanvasTkAgg(fig, master=self.wind)
canvas.draw()
canvas.get_tk_widget().place(x=630, y=10) # this sets the position of the graph in the Tkinter window
self.forecast_value_in_pounds_tomorrow() # calls the function to forecast value for tomorrow
self.forecast_value_in_pounds_for_next_month() # calls the function to forecast value for next month

```

```

def display_sold_quantity_graph(self):
    # figsize - width and height of 8.5 and 6 inches
    fig = plt.Figure(figsize=(8.1, 6), dpi=100)
    ax = fig.add_subplot(111)

    # fetch data from the database and add it to the figure
    connection = sqlite3.connect(r'sms.db')
    cursor = connection.cursor()
    select_value = cursor.execute("Select Date, Sold FROM Sales WHERE Date > (SELECT DATE('now','-7 day')) and "
                                  "Product_ID =?" , (self.var_product_ID.get(),)).fetchall() # getting the data for last 7 days

```

```

dates = [] # dates list
sold = [] # sold list
for row in select_value:
    dates.append(row[0]) # adding to dates list
    sold.append(row[1]) # adding to sold list
# create a bar chart of the data
ax.bar(dates, sold, color='g', width=0.60, label="Sold in terms of Quantity")
ax.set_xlabel('Date')
ax.set_ylabel('Sold by Quantity')

```

```

# get the name of product from the product table
name_of_product = cursor.execute("Select Product FROM Product WHERE Product_ID = ?".format(self.var_product_ID.get()),).fetchone()[0]
ax.set_title(f'The quantity of {name_of_product} sold') # title with name of product
ax.legend(loc="lower right")
# move the y-axis to the left
ax.yaxis.set_label_coords(-0.1, 0.5)

# set the y-axis label and move it to the left
ax.set_ylabel('Sold by Quantity', labelpad=15)

```

```

# create a Tkinter canvas that can display the Matplotlib figure
canvas = FigureCanvasTkAgg(fig, master=self.wind)
canvas.draw()
canvas.get_tk_widget().place(x=630, y=10) # this sets the position of the graph in the Tkinter window
self.forecast_quantity_sold_tomorrow() # calls function
self.forecast_Quantity_sold_for_next_month() # calls function

if __name__ == "__main__":
    wind = Tk()
    obj = SalesClass(wind)
    wind.mainloop() # start the GUI

```

Code for Analysis window

```

# Import necessary modules and libraries
from tkinter import ttk, messagebox # for creating GUI and displaying messages
import sqlite3 # for interacting with SQLite database
from tkinter import filedialog # for selecting files using a dialog box
from array import * # for working with arrays
import numpy # for performing numerical operations on arrays
import matplotlib.pyplot as plt # for creating graphs
from tkinter import * # for creating GUI

class AnalysisClass: # creating a class called Analysis Class
    def __init__(self, wind):
        self.wind = wind
        self.wind.title("Performance of Products")

        # Setting the dimensions of the window
        window_width = 1550
        window_height = 750
        screen_width = wind.winfo_screenwidth()
        screen_height = wind.winfo_screenheight()
        center_x = int(130+(screen_width / 2 - window_width / 2)) # centre x
        center_y = int(50+(screen_height / 2 - window_height / 2)) # centre y

        # set the position of the window to the center of the screen
        self.wind.geometry(f'{window_width}x{window_height}+{center_x}+{center_y}')

```

```

# Label showing worst 5 products
worst_5_lbl = Label(self.wind, text = "Worst 5 Products", font = ("Rosewood Std Regular",30), bg = "#4cafaf", fg = "black")\
.place(x=70, y = 80, height = 100, width = 500)

# Label showing top 5 products
top_5_lbl = Label(self.wind, text = "Top 5 Products", font = ("Rosewood Std Regular",30), bg = "#4cafaf", fg = "black")\
.place(x=870, y = 80, height = 100, width = 500)

# This frame will contain the treeview showing the best 5 products
top_5_frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=2000, width=2000)
top_5_frame.place(x=850, y=250) # placing the frame

# This frame will contain the treeview showing the worst 5 products
worst_5_frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=2000, width=2000)
worst_5_frame.place(x=60, y=250) # placing the frame

#####
# this is a frame to split the entire window into half #####
Middle_Frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=3000, width=20)
Middle_Frame.place(x=750, y=5)

```

```

# Button that shows the worst 5 products in a graph format when clicked
worst_5_btn = Button(self.wind, text="Show graph", command=self.worst_5_products, bg="#734058",\
font=("OCR A Std", 15), fg="yellow").place(x=190, y=670, height=55, width=300)

# Button that shows the top 5 products in a graph format when clicked
top_5_btn = Button(self.wind, text="Show graph", command=self.top_5_products, bg="#734058",\
font=("OCR A Std", 15), fg="yellow").place(x=1000, y=670, height=55, width=300)

```

```

#####
# Treeview for top 5 products #####
rightscrolly = Scrollbar(top_5_frame, orient=VERTICAL) # vertical scrollbar
rightscrollx = Scrollbar(top_5_frame, orient=HORIZONTAL) # horizontal scrollbar

# defining the treeview
self.best_5_table = ttk.Treeview(top_5_frame, columns=("Product_ID", "Product", "Sold"),
                                 yscrollcommand=rightscrolly.set, xscrollcommand=rightscrollx.set)

rightscrollx.pack(side=BOTTOM, fill=X)
rightscrolly.pack(side=RIGHT, fill=Y)
rightscrollx.config(command_= self.best_5_table.xview)
rightscrolly.config(command_= self.best_5_table.yview)

self.best_5_table.heading("Product_ID", text="Product_ID")
self.best_5_table.heading("Product", text="Product")
self.best_5_table.heading("Sold", text="Sold")

self.best_5_table["show"] = "headings"

self.best_5_table.column("Product_ID", width=170, stretch=NO) # fixed width
self.best_5_table.column("Product", width=170, stretch=NO) # fixed width
self.best_5_table.column("Sold", width=170, stretch=NO) # fixed width

self.best_5_table.pack(fill=BOTH, expand=1)

```

```
#####
# Treeview for the worst 5 products #####
#####

scrolly = Scrollbar(worst_5_frame, orient=VERTICAL) # vertical scrollbar
scrollx = Scrollbar(worst_5_frame, orient=HORIZONTAL) # horizontal scrollbar

# least 5 treeview
self.least_5_table = ttk.Treeview(worst_5_frame, columns=("Product_ID", "Product", "Sold"),
                                  yscrollcommand=scrolly.set, xscrollcommand=scrollx.set)

scrollx.pack(side=BOTTOM, fill=X)
scrolly.pack(side=RIGHT, fill=Y)
scrollx.config(command=self.least_5_table.xview)
scrolly.config(command=self.least_5_table.yview)

self.least_5_table.heading("Product_ID", text="Product_ID")
self.least_5_table.heading("Product", text="Product")
self.least_5_table.heading("Sold", text="Sold")

self.least_5_table["show"] = "headings"

self.least_5_table.column("Product_ID", width=170, stretch=NO) # fixed width
self.least_5_table.column("Product", width=170, stretch=NO) # fixed width
self.least_5_table.column("Sold", width=170, stretch=NO) # fixed width

self.least_5_table.pack(fill=BOTH, expand=1)
```

```
#####
# Calling all the functions from below #####
#####

self.display_worst_5_products_leaderboard() # (Leaderboard)
self.top_5_products_leaderboard() # (Leaderboard)

#####
# Graph showing top 5 products #####
#####

def top_5_products(self):
    connection = sqlite3.connect(r'sms.db') # connect to the database
    cursor = connection.cursor()
    # this uses the Product ID and sums up all the quantity sold within one week. It is ordered in descending order
    cursor.execute("SELECT Product_ID, SUM(Sold) AS total_revenue FROM Sales WHERE Date > (SELECT DATE('now', '-7 day')) "
                  "GROUP BY Product_ID ORDER BY SUM(Sold) DESC LIMIT 5") # limit 5 means only the first 5 products are fetched
    show_top_5 = cursor.fetchall() # this fetches all the first (top) 5 products
    product_ID = [] # creates a list to store multiple Product ID's
    sold = [] # creates a list to store multiple sold data
    for row in show_top_5: # iterates through the top 5 values
        product_ID.append(row[0]) # adds to the list
        sold.append(row[1]) # adds to the list
```

```
# Set the width and gap between bars in the bar chart
width = 0.8
gap = 0.05

# Compute the positions of each bar
positions = range(1, len(show_top_5) + 1)
positions = [pos - width / 2 - gap for pos in positions]

# create the bar chart
plt.bar(positions, sold, width=width, color='c', label="Quantity sold")
plt.xticks(range(1, len(show_top_5) + 1), product_ID)

plt.xlabel('Product_ID') # x axis title
plt.ylabel('Sold') # y axis title
plt.title('Top 5 performing products') # title
plt.tight_layout()
plt.legend()
plt.show() # display the bar chart
```

```
#####
# Graph showing worst 5 products #####
#####

def worst_5_products(self):
    connection = sqlite3.connect('sms.db') # connects tp database
    cursor = connection.cursor()
    # fetches the product id and sum of sold quantity within one week in ascending order and retrieves first 5 products
    cursor.execute("SELECT Product_ID, SUM(Sold) AS total_revenue from Sales WHERE Date > (SELECT DATE('now', '-7 day')) "
                  "GROUP BY Product_ID ORDER BY SUM(Sold) LIMIT 5")
    show_worst_5 = cursor.fetchall()
    product_id = [] # list
    sold = [] # list
    for row in show_worst_5: # for loop
        product_id.append(row[0]) # add to list
        sold.append(row[1]) # add to list
    width = 0.8
    gap = 0.05
    # Compute the positions of each bar
    positions = range(1, len(show_worst_5) + 1)
    positions = [pos - width / 2 - gap for pos in positions]
```

```
plt.bar(positions, sold, width=width, color='c', label="Quantity sold")
plt.xticks(range(1, len(show_worst_5) + 1), product_id)

# features of the graph
plt.xlabel('Product_ID')
plt.ylabel('Sold')
plt.title('Worst 5 performing products')
plt.tight_layout()
plt.legend()
plt.show()
```

```
#####
# Leaderboard showing worst 5 products #####
#####

def display_worst_5_products_leaderboard(self):
    connection = sqlite3.connect('sms.db') # connecting to database
    cursor = connection.cursor()
    ##### Connecting Sales and Product tables to fetch Product field #####
    ##### shows products in ascending order based on total quantity sold within one week #####
    show_worst_5 = "SELECT Sales.Product_ID, Product.Product, SUM(Sold) AS total_revenue from Sales" \
                  " INNER JOIN Product ON Product.Product_ID = Sales.Product_ID WHERE Date > (SELECT DATE('now', '-7 day'))" \
                  " GROUP BY Sales.Product_ID ORDER BY SUM(Sold) LIMIT 5" # limit 5 means only first 5 products are fetched
    rows = cursor.execute(show_worst_5).fetchall()
    connection.commit()
    for r in rows: # iterating through values stored in rows
        self.least_5_table.insert('', END, values=r) # adding it to least_5 treeview
```

```
#####
# Leaderboard showing top 5 products #####
#####

def top_5_products_leaderboard(self):
    connection = sqlite3.connect('sms.db') # connect to database
    cursor = connection.cursor()
    # here I have joined the Product and Sales table through their Product field as it is common in both tables
    # This helps to fetch the name of the Product from the Product table by joining the 2 tables together
    top_5 = "SELECT Sales.Product_ID, Product.Product, SUM(Sold) AS total_revenue from Sales" \
            " INNER JOIN Product ON Product.Product_ID = Sales.Product_ID WHERE Date > (SELECT DATE('now', '-7 day'))" \
            " GROUP BY Sales.Product_ID ORDER BY SUM(Sold) DESC LIMIT 5"
    # LIMIT 5 means only 5 products will be shown on the treeview
    rows = cursor.execute(top_5).fetchall()
    connection.commit()
    for r in rows: # iterates through the values stored in rows
        self.best_5_table.insert('', END, values=r) # inserts into best 5 treeview

if __name__ == "__main__":
    wind = Tk()
    obj = AnalysisClass(wind)
    wind.mainloop() # runs the program
```

Code for Stock-Out window

```

from tkinter import ttk, messagebox
import sqlite3
import csv
import smtplib
from tkinter import *
from tkinter import filedialog
from array import *
import numpy
class Stock_Out:
    def __init__(self, wind):
        self.wind = wind
        self.wind.title("Stock Out")

        # sets dimensions of the window
        window_width = 1550
        window_height = 750
        screen_width = wind.winfo_screenwidth()
        screen_height = wind.winfo_screenheight()
        center_x = int(130+(screen_width / 2 - window_width / 2)) # centre x
        center_y = int(50+(screen_height / 2 - window_height / 2)) # centre y

        # set the position of the window to the center of the screen
        self.wind.geometry(f'{window_width}x{window_height}+{center_x}+{center_y}')

Stock_Out_Table_Frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=2000, width=20)
Stock_Out_Table_Frame.place(x=60, y=0)

```

```

#####
scrolly = Scrollbar(Stock_Out_Table_Frame, orient=VERTICAL)
scrollx = Scrollbar(Stock_Out_Table_Frame, orient=HORIZONTAL)
self.Stock_Out_table = ttk.Treeview(Stock_Out_Table_Frame,
                                    columns=("Product_ID", "Product", "Quantity", "Reorder_Point", "Supp ID", "Supp Name", "Supp Email"),
                                    yscrollcommand=scrolly.set, xscrollcommand=scrollx.set)

scrollx.pack(side=BOTTOM, fill=X)
scrolly.pack(side=RIGHT, fill=Y)
scrollx.config(command=self.Stock_Out_table.xview)
scrolly.config(command=self.Stock_Out_table.yview)

self.Stock_Out_table.heading("Product_ID", text="Product ID")
self.Stock_Out_table.heading("Product", text="Product")
self.Stock_Out_table.heading("Quantity", text="Stock Quantity")
self.Stock_Out_table.heading("Reorder_Point", text="Reorder Point")
self.Stock_Out_table.heading("Supp ID", text="Supplier ID")
self.Stock_Out_table.heading("Supp Name", text="Supplier Name")
self.Stock_Out_table.heading("Supp Email", text="Supplier Email")
self.Stock_Out_table["show"] = "headings"

```

```

self.Stock_Out_table.column("Product_ID", width=120, stretch=NO) # fixed width
self.Stock_Out_table.column("Product", width=120, stretch=NO) # fixed width
self.Stock_Out_table.column("Quantity", width=120, stretch=NO) # fixed width
self.Stock_Out_table.column("Reorder_Point", width=120, stretch=NO) # fixed width
self.Stock_Out_table.column("Supp ID", width=120, stretch=NO) # fixed width
self.Stock_Out_table.column("Supp Name", width=120, stretch=NO) # fixed width
self.Stock_Out_table.column("Supp Email", width=330, stretch=NO) # fixed width

self.Stock_Out_table.pack(fill=BOTH, expand=1)
self.Stock_Out_table.pack(fill=BOTH, expand=1)
self.Stock_Out_table.bind("<ButtonRelease-1>", self.get_Stock_Out)

self.update_Stock_Out() # calls the function

```

```
#####
# Email Panel #####
#####

# label to show where to enter the email address
self.address_field = Label(self.wind, text="Recipient Address :", font=("Rosewood Std Regular", 15), bg="#4CAF50", fg="black")
self.address_field.place(x=15, y=370, height=50)

# label to show where to enter the email body message
self.email_body_field = Label(self.wind, text="Message :", font=("Rosewood Std Regular", 15), bg="#4CAF50", fg="black")
self.email_body_field.place(x=15, y=560, height=50)

self.address = StringVar() # stores the sender's address
self.email_body = StringVar() # stores the content of the message

# this is where the recipient's (suppliers) email address is put
self.address_form = Entry(self.wind, textvariable=self.address, width="80",
                           font=("Rosewood Std Regular", 15), bg="lightyellow", fg="black").place(x=15, y=450, height=60)

# this is where the user enters their message
self.email_body_form = Entry(self.wind, textvariable=self.email_body, width="80", font=("Rosewood Std Regular", 15),
                             bg="lightyellow", fg="black").place(x=15, y=650, height=60)

# Button to send email
self.email_btn = Button(self.wind, text="Send message", command=self.send, bg="light blue", width="20", height="6").place(x=1330, y=470)
```

```
#####
# function to send email to supplier #####
def send(self):
    try:
        username, password = self.fetch_details() # calls the fetch_details() function to get username and password from the csv file
        to = self.address.get() # variable defined "to" which stores the recipient's address
        email_body = self.email_body.get() # content of the message
        if to == "" or email_body == "": # if the entry fields are blank
            messagebox.showerror("Incomplete input", "Please fill in all the entry forms", parent=self.wind)
        else:
            subject = "One-Stop products are out of stock" # subject message
            email_message = f'Subject: {subject}\n\n{email_body}' # message that stores subject and email body content
            # create an instance of smtplib.SMTP
            # port number = 587
            server = smtplib.SMTP('smtp.gmail.com', 587)
            server.starttls() # tells server we want to use TLS encryption - this ensures no third party can read or modify sent data
            server.login(username, password) # username and password is retrieved from csv file
            server.sendmail(username, to, email_message) # takes in sender's email address, recipients email address and content of the message as parameters
            messagebox.showinfo("Sent", "Email has been sent", parent=self.wind) # this shows a message if the email has been sent successfully

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}",
                             parent=self.wind) # this gives us any errors that may occur when coding
```

```
def update_Stock_Out(self):
    connection = sqlite3.connect(r'sms.db') # connecting to database
    cur = connection.cursor()
    # sql query to get the products and supplier details using the source table where there is 0 quantity left in the store
    stocked_out = "select Product.Product_ID, Product.Product, Product.Quantity, Product.Reorder_Point, " \
                  "Supplier.ID, Supplier.Name, Supplier.Email FROM supplier, Product, Sourcing " \
                  "WHERE Sourcing.Product_ID = Product.Product_ID AND Sourcing.Supplier_ID = supplier.ID " \
                  "AND Product.Quantity = 0"
    rows = cur.execute(stocked_out).fetchall() # gets all the records that fulfil the condition
    connection.commit()
    for r in rows: # iterates through the values stored in rows
        self.Stock_Out_table.insert('', END, values=r) # adds to stock out treeview

    # when the user clicks on one of the records in the treeview, the email address is populated onto the email address entry field
def get_Stock_Out(self, stock_out): # this gets the row of data
    self.focus = self.Stock_Out_table.focus()
    self.data = self.Stock_Out_table.item((self.focus))
    row = self.data['values']
    self.address.set(row[6])
```

```
# this function gets the username and password of the user from the csv file
def fetch_details(self):
    with open('login.csv', 'r') as file:# opens the csv file so that it can be read
        reader = csv.reader(file)# csv reader object
        for row in reader:# iterates through the rows in the csv file
            username = row[0]# first data is username which is the sender's email address
            password = row[1]# second data is an app password
    return username, password# returns the username and password as a tuple

if __name__ == "__main__":
    wind = Tk()
    obj = Stock_Out(wind)
    wind.mainloop() # runs the program
```

Code for Reorder Point window

```
from tkinter import ttk, messagebox# used to give message
import sqlite3# used for database
import smtplib# used to send emails
from tkinter import *
import csv# used for csv file
from array import *
import numpy
from tkinter import filedialog
class Reorder_Point:
    def __init__(self, wind):
        self.wind = wind
        self.wind.title("Reorder Point")

        # dimensions of the window
        window_width = 1550
        window_height = 750
        screen_width = wind.winfo_screenwidth()
        screen_height = wind.winfo_screenheight()
        center_x = int(130+(screen_width / 2 - window_width / 2))# centre x
        center_y = int(50+(screen_height / 2 - window_height / 2))# centre y

        # set the position of the window to the center of the screen
        self.wind.geometry(f" {window_width}x{window_height}+{center_x}+{center_y}")

    Reorder_Point_Table_Frame = Frame(self.wind, bd=7, relief=GROOVE, highlightcolor="red", bg="#734058", cursor="circle", height=2000, width=20)
    Reorder_Point_Table_Frame.place(x=60, y=0)
```

```
#####
Reorder Point treeview #####
scrolly = Scrollbar(Reorder_Point_Table_Frame, orient=VERTICAL)
scrollx = Scrollbar(Reorder_Point_Table_Frame, orient=HORIZONTAL)
self.Reorder_point_table = ttk.Treeview(Reorder_Point_Table_Frame,
                                         columns=("Product_ID", "Product", "Quantity", "Reorder_Point", "Supp ID", "Supp Name", "Supp Email"),
                                         yscrollcommand=scrolly.set, xscrollcommand=scrollx.set)

scrollx.pack(side=BOTTOM, fill=X)
scrolly.pack(side=RIGHT, fill=Y)
scrollx.config(command=self.Reorder_point_table.xview)
scrolly.config(command=self.Reorder_point_table.yview)
```

```

self.Reorder_point_table.heading("Product_ID", text="Product ID")
self.Reorder_point_table.heading("Product", text="Product")
self.Reorder_point_table.heading("Quantity", text="Stock Quantity")
self.Reorder_point_table.heading("Reorder_Point", text="Reorder Point")
self.Reorder_point_table.heading("Supp ID", text="Supplier ID")
self.Reorder_point_table.heading("Supp Name", text="Supplier Name")
self.Reorder_point_table.heading("Supp Email", text="Supplier Email")
self.Reorder_point_table["show"] = "headings"

self.Reorder_point_table.column("Product_ID", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Product", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Quantity", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Reorder_Point", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Supp ID", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Supp Name", width=120, stretch=NO) # fixed width
self.Reorder_point_table.column("Supp Email", width=330, stretch=NO) # fixed width

self.Reorder_point_table.pack(fill=BOTH, expand=1)
self.Reorder_point_table.pack(fill=BOTH, expand=1)
self.Reorder_point_table.bind("<ButtonRelease-1>", self.get_Reorder_Point)

self.update_reorder_point() # calls the function

```

```

#####
# Email Panel #####
#####

# label to show where to enter the recipient's email address
self.address_field = Label(self.wind, text="Recipient Address :", font=("Rosewood Std Regular", 15), bg="#4cafaf", fg="black")
self.address_field.place(x=15, y=370, height=50)

# label to show where to enter the message
self.email_body_field = Label(self.wind, text="Message :", font=("Rosewood Std Regular", 15), bg="#4cafaf", fg="black")
self.email_body_field.place(x=15, y=560, height=50)

self.address = StringVar() # variable to store recipient's address
self.email_body = StringVar() # variable to store what user types as message

# this is where the recipient's (suppliers) email address is put
self.address_form = Entry(self.wind, textvariable=self.address, width="80", font = ("Rosewood Std Regular", 15), bg = "lightyellow",
                           fg = "black").place(x = 15, y = 450, height=60)

# this is where the user enters their message
self.email_body_form = Entry(self.wind, textvariable=self.email_body, width = "80", font = ("Rosewood Std Regular", 15),
                            bg = "lightyellow", fg = "black").place(x = 15, y = 650, height = 60)

```

```

# send button
self.email_btn = Button(self.wind, text = "Send message", command = self.send_email, bg = "light blue", width = "20", height="6").place(x = 1330, y = 470)

#####
# function to send email to supplier #####
def send_email(self):
    try:
        username, password = self.fetch_details() # calls the fetch_details() function to get username and password from the csv file
        to = self.address.get() # variable defined "to" which stores the recipient's address
        email_body = self.email_body.get() # content of the message
        if to == "" or email_body == "": # if any of the fields are empty
            messagebox.showerror("Incomplete input", "Please fill in all the entry forms", parent = self.wind)
        else:
            subject = "One-Stop needs more products to be ordered" # subject message
            email_message = f'Subject: {subject}\n\n{email_body}' # message that stores subject and email body content
            # create an instance of smtplib.SMTP
            # port number = 587
            server = smtplib.SMTP('smtp.gmail.com', 587)
            server.starttls() # tells server we want to use TLS encryption - this ensures no third party can read or modify sent data
            server.login(username, password) # username and password is retrieved from csv file
            server.sendmail(username, to, email_message) # takes in sender's email address, recipients email address and content of the message as parameters
            messagebox.showinfo("Sent", "Email has been sent", parent = self.wind) # this shows a message if the email has been sent successfully

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind) # this gives us any errors that may occur when running the program

```

```
#####
# Reorder point on treeview #####
def update_reorder_point(self):
    connection = sqlite3.connect(r'sms.db') # connection to the database
    cur = connection.cursor()
    # this sql join the sourcing table with the product table and the supplier table so that the supplier and product details can be fetched
    # gets and joins records from tables if the reorder point is greater than the current quantity of a certain product
    reorder_point = "select Product.Product_ID, Product.Product, Product.Quantity, Product.Reorder_Point, " \
                    "Supplier.ID, Supplier.Name, Supplier.Email FROM supplier, Product, Sourcing WHERE Sourcing.Product_ID = Product.Product_ID" \
                    " AND Sourcing.Supplier_ID = supplier.ID AND Product.Quantity < Product.Reorder_Point ORDER BY Product.Quantity"
    rows = cur.execute(reorder_point).fetchall() # fetches all the records that fulfils the condition
    connection.commit() # commits the changes
    for r in rows: # iterates through the values stored in rows
        self.Reorder_point_table.insert('', END, values=r) # adds it to the treeview

# when the user clicks on one of the records in the treeview, the email address is populated onto the email address entry field
def get_Reorder_Point(self, sourcing): # this gets the row of data
    self.focus = self.Reorder_point_table.focus()
    self.data = self.Reorder_point_table.item((self.focus))
    row = self.data['values']
    self.address.set(row[6])
```

```
# this function gets the username and password of the user from the csv file
def fetch_details(self):
    with open('login.csv', 'r') as file: # opens the csv file so that it can be read
        reader = csv.reader(file) # csv reader object
        for row in reader: # iterates through the rows in the csv file
            username = row[0] # first data is username which is the sender's email address
            password = row[1] # second data is an app password
    return username, password # returns the username and password as a tuple

if __name__ == "__main__":
    wind = Tk()
    obj = Reorder_Point(wind)
    wind.mainloop() # runs the program
```

Code for Login

```
from tkinter import *
from tkinter import messagebox # used for showing messages
import sqlite3 # used for databases
import os # this controls the window the program goes into next
import smtplib # used for sending emails
import csv # used to send details to send emails
import random # will use random function to generate opt code
class login:
    def __init__(self, wind):
        self.wind = wind
        self.wind.title("Login system")
        self.wind.geometry("1900x990+0+0")

        self.otp = '' # this will be the generated OTP code once the user enters the username and password correctly

        login_section = Frame(self.wind, bd=2, relief=RIDGE)
        login_section.place(x=600, y=90, width=750, height=860)

        login_label = Label(login_section, text = "Login to access dashboard", font = ("Elephant", 30, "bold")).place(x = 0, y = 30, relwidth=1)

        username_lbl = Label(login_section, text = "Username:", font=("DaunPenh", 15, "italic")).place(x = 50, y = 130)

        self.username = StringVar() # stores the username
        self.password = StringVar() # stores the password
```

```
#####
# username entry box #####
username_entry = Entry(login_section, textvariable= self.username, font=("times new roman", 15), bg = "yellow").place(x = 50, y = 170)

password_lbl = Label(login_section, text = "Password:", font=("DaunPenh", 15, "italic")).place(x = 50, y = 380)

#####
# password entry box #####
password_entry = Entry(login_section, textvariable= self.password,
                      show = "*", font=("times new roman", 15), bg = "yellow").place(x = 50, y = 420)

login_btn = Button(login_section, text="Log in",
                   command = self.login_validation, font=("times new roman", 15), bg = "Light Blue").place(x = 50, y = 520, width=300)

#####
# If user forgets password #####
Forgot_Password_lbl = Label(login_section, text = "Forgot Password?",
                           borderwidth=7, relief = RIDGE, bg = "lightgreen", fg = "red", font=("DaunPenh", 15, "italic")).place(x = 50, y = 700)

#####
# forget button #####
forget_btn = Button(login_section, text = "Click Here!",
                     command=self.forget_password, font=("DaunPenh", 15, "italic"), bg = "orange").place(x = 450, y = 700)
```

```
def login_validation(self):
    connection = sqlite3.connect(r'sms.db') # connecting to database
    cur = connection.cursor()
    try:
        if self.username.get() == "" or self.password.get() == "": # if the fields are left blank
            messagebox.showerror("Incomplete", "Please fill in all the details", parent = self.wind) # shows a message error
            return
        else:
            # sql to check if the user input exists in the login table
            cur.execute("select * from login where Username=? AND Password=? ",(self.username.get(), self.password.get()))
            login_details = cur.fetchone()
            if login_details == None: # if it doesn't exist in login table
                messagebox.showerror("Error", "Password or Username is wrong") # error message occurs
                return
    
```

```
else:
    ###### if the username and password is correct #####
    messagebox.showinfo("Hi", "Sending OTP on email") # sends otp
    self.two_factor = Toplevel(self.wind) # creates a new small window on top
    self.two_factor.title("2 factor authentication")
    self.two_factor.geometry("500x500+500+110")
    self.two_factor.focus_force() # tells the program that this window is active and input should be taken from this window

    self.two_factor_code = IntVar() # defining the variable. This will store the OTP code that the user enters once their username and password is correct

    ###### Label to show what to enter in the entry form #####
    OTP_lbl = Label(self.two_factor, text=" Enter OTP Code:", font=('goudy old style', 15, 'bold'), bg = "#3f51b5", fg = "white").pack(side=TOP, fill=X)

    ###### Entry form to enter otp code #####
    enter_otp = Entry(self.two_factor, textvariable= self.two_factor_code, font=('goudy old style', 15, 'bold')).place(x=120, y=150)

    ###### This is a submit otp button #####
    self.otp_submit = Button(self.two_factor, text="Submit OTP", command=self.two_factor.authentification,
                           font=('goudy old style', 15, 'bold')).place(x=160, y=230, width=150,height=50)
```

```

        gmail_user, gmail_password, to_email = self.fetch_details() # calls the function to get sender email, password and who to send it to
        self.send_email(to_email, self.otp, gmail_user, gmail_password) # calls the send_email function

    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

##### calls function once the username and password is correct #####
def two_factor_authentication(self):
    if int(self.otp) == int(self.two_factor_code.get()): # if the generated otp matches the user's input of otp code
        messagebox.showinfo("Information", "OTP code is correct", parent=_self.two_factor)
        os.system(r"C:\Users\rusha\PycharmProjects\pythonProject5\sms_dashboard.py") # opens the dashboard window
    else: # else an error occurs
        messagebox.showerror("Error", "Please type correct otp code", parent=_self.two_factor)
    return

```

```

##### function if user forgets their password #####
def forget_password(self):
    connection = sqlite3.connect(r'sms.db') # connecting your database
    cur = connection.cursor()
    try:
        if self.username.get() == "": # if the username is empty
            messagebox.showerror("Incomplete", "Please fill in the username to get the OTP code", parent=_self.wind)
            return
        else:
            # checks if username is correct
            cur.execute("Select Email from login where Username=? ", (self.username.get(),)) # get the email address from the user's username
            email = cur.fetchone()
            if email == None: # no record found
                messagebox.showerror("Error", "Username doesn't exist", parent=_self.wind)
                return
            else:
                self.var_otp_code = IntVar() # this otp variable stores the otp code that the user will enter to change their password
                self.var_updated_password = StringVar() # stores updated password from user's input
                self.confirm_password = StringVar() # stores the user's confirmed password input

                self.forget_password = Toplevel(_self.wind) # creates a new window
                self.forget_password.title("New Password")
                self.forget_password.geometry("500x500+500+110")
                self.forget_password.focus_force()

```

```

title = Label(self.forget_password, text="Reset Password", font=('goudy old style', 15, 'bold'),
             bg="#3f51b5", fg="white").pack(side=TOP, fill=X)

lbl_forget_password = Label(self.forget_password, text="Enter OTP sent on Email",
                           font=('goudy old style', 15, 'bold')).place(x=20, y=60)

##### entry box to enter OTP code #####
entry_forget_password = Entry(self.forget_password, textvariable=self.var_otp_code, font=('goudy old style', 15, 'bold')).place(x=20, y=100)

updated_pass_lbl = Label(self.forget_password, text="Update password", font=('goudy old style', 15, 'bold')).place(x=20, y=240)

##### entry box to enter user's new password #####
input_new_pass = Entry(self.forget_password, textvariable=self.var_updated_password, show = "*", font=('goudy old style', 15, 'bold')).place(x=20, y=290)

lbl_confirm_new_password = Label(self.forget_password, text="Confirm new password", font=('goudy old style', 15, 'bold')).place(x=20, y=340)

##### entry box to confirm the user's new password #####
entry_confirm_new_password = Entry(self.forget_password, textvariable=self.confirm_password, show = "*",
                                    font=('goudy old style', 15, 'bold')).place(x=20, y=390)

```

```

self.update_btn = Button(self.forget_password, text="Update password", command = self.update_password,
                        font=('goudy old style', 15, 'bold')).place(x=20, y=440, width=200, height=50)

gmail_user, gmail_password, to_email = self.fetch_details()
self.send_email(to_email, self.otp, gmail_user, gmail_password)

except Exception as ex:
    messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

```

def update_password(self):
    if self.var_updated_password.get() == "" or self.confirm_password.get() == "": # if any of the fields are empty
        messagebox.showerror("Error", "Password is required", parent=_self.forget_password)
        return
    elif self.var_updated_password.get() != self.confirm_password.get(): # if the confirmed password and the changed password do not match
        messagebox.showerror("Error", "Please check that the password entered is the same", parent=_self.forget_password)
        return
    elif int(self.otp) != int(self.var_otp_code.get()): # if the otp code entered by the user doesn't match the one generated
        messagebox.showerror("Error", "Please enter the correct otp", parent=_self.forget_password)
        return
    elif self.var_otp_code.get() == "": # if the user doesn't enter otp code
        messagebox.showerror("Error", "Please fill in the OTP code", parent=_self.forget_password)
        return

```

```

else:
    con = sqlite3.connect(database=r'sms.db') # connect to database
    cur = con.cursor()
    try:
        # updates the password by using the username as part of the where condition
        cur.execute("Update login SET Password=? where Username=?",(self.var_updated_password.get(),self.username.get(),))
        # message showing that the password has been updated successfully
        messagebox.showinfo("Password", "Successfully updated password", parent=_self.forget_password)
        con.commit()
    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.forget_password)

```

```

def send_email(self,to_email, otp, gmail_user, gmail_password):
    sent_from = gmail_user
    to = [to_email]
    self.otp = random.randint(100000, 999999) # generates a random six digit number
    subject = "Your OTP code"
    body = "Your otp code is:" + str(self.otp)
    email_message = f'Subject: {subject}\n\n{body}'
    try:
        server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login(gmail_user, gmail_password) # sender's details
        server.sendmail(sent_from, to, email_message) # sends mail - to the sender
        server.close()
    except Exception as ex:
        messagebox.showerror("Error", f"Error due to: {str(ex)}", parent=self.wind)

```

```

def fetch_details(self):
    with open('login.csv', 'r') as file:# opens csv file
        reader = csv.reader(file)# creates a reader object
        for row in reader:# iterates through the csv file
            email = row[0]# sender's email
            password = row[1]# sender's password
            to_email = row[0]# recipient's email
    return email, password, to_email# returns them

if __name__ == "__main__":
    wind = Tk()
    obj = login(wind)
    wind.mainloop() # this executes Tkinter and runs the application

```