Centre Number: 56120

POSSY

Candidate Number: 4758

Ollie Barnes

Contents

Analysis

Introduction

My project will be a web-based point of sale (POS) and stock management system called 'POSSY'. Its core purpose is to provide a remote point-of-sale and stock management system to businesses, allowing them to open and close customers' transactions whilst automatically updating their stock management database. The stock management database enables stock to be managed between sites whilst presenting an easy-to-read end-of-day report containing the total daily takings for all locations in one central hub alongside their current stock levels.

You will sign up to the 'POSSY' website as a company using your company email, enabling you to log in and create your own 'menu', consisting of any items that your company sells. Each item in the 'menu' will have a name and a cost. The system operator can then select any items a customer has bought, and the system can automatically display the cost total. Once the operator marks the payment as complete, the point-of-sale system interacts with a stock management database, deducting any items bought by the customer from the total stock. If an item has a stock level below a pre-set value, an alert containing the product's name that is low on stock and the contact details to the supplier is output to the system operator.

To prevent the end-user from being too limited, you may select that the item becomes a dish containing multiple item components when creating a new menu item. Implementing such a feature allows the 'POSSY' stock management system to cater to the catering industry. Without this feature, a dish such as a sausage bap may appear as being out of supply. Yet, the components used to create such a dish are still in ample supply. This would happen due to the database having assigned the item components of the dish to a different dish despite all item components being in stock, just in other forms. This means that without the 'POSSY' stock management system explicitly catering for such a situation, those in the catering industry could not efficiently use the stock management system.

I had the idea for a web-based point-of-sale system when my family expressed a need for such an application in the catering industry. They found the current solutions to be a problem as many remote point-of-sale systems did not have a stock management system to control stock efficiently.

For this project to be successful, I will need to identify which features of a point of sale and stock management system are essential and which features potential end-users would find desirable in such a system that may not yet exist. I would also like to ask those with experience in such systems which positions they prefer buttons on the user interface, such as the pay/complete transaction button, and which colour schemes they think would work best for appeal and reduced strain on the eyes. To achieve this, I will send a questionnaire to local businesses and members of the public with varying levels of experience with such systems. The results will be the deciding factor in the system's layout. I will also have to create a questionnaire to discover the percentage of people with experience with point-of-sale systems and their level of expertise. The results will affect whether I will need a tutorial used to train new users on the system.

To develop this web application, I will first need to investigate how to interact with SQL databases using PHP. I have had minimal prior experience with PHP making this project a challenge; however, understanding PHP is essential to complete this application.

Research

1. 'Erply' point of sale system allows companies to sell stock items no matter where they are whilst updating their stock levels in real-time. It allows for offline capabilities meaning that you can still use the system even when an internet connection is unavailable. It provides companies with detailed order and invoice tracking, comprehensive inventory management and built-in loyalty programs such as coupons.

Features I would like to include:

• Using tabs to display the items in categories helps organise items in the point-of-sale system, potentially decreasing the average transaction time. The use of tabs also allows for a more significant number of items in the point-of-sale system, providing companies with the option to have more sale items than not having items in tabbed categories. If time permits it, I will take this forward into my project design.

▪ Similar 'Pay' and 'Sale' buttons of a similar size ensuring that they are easily pressable when using a touch screen device and distinguishable from other controls in the menu. I will take this forward to my project design.

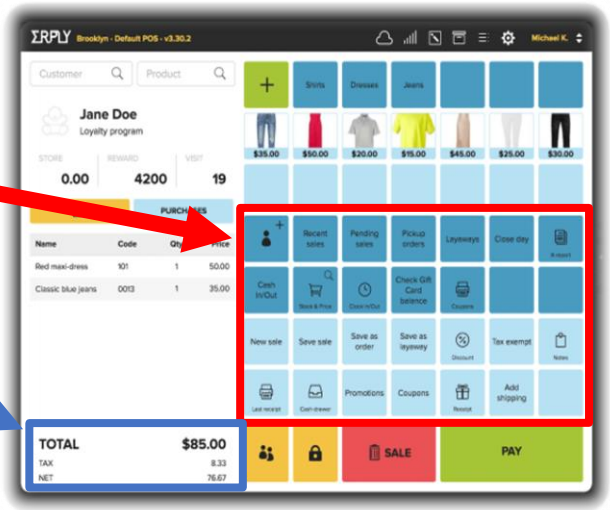
▪ The ability for users to log into the menu using a personal account could allow for more features such as the tracking of individual sales targets. I will add a question to a survey to see whether this is an essential feature in a point-of-sale system, as this will be a time-consuming task to complete.

▪ The ability to use the local point of sale system and stock management system offline, with the global stock management system updated as soon as the system goes online. I will not take this forward into my project as my stakeholders do not require it.

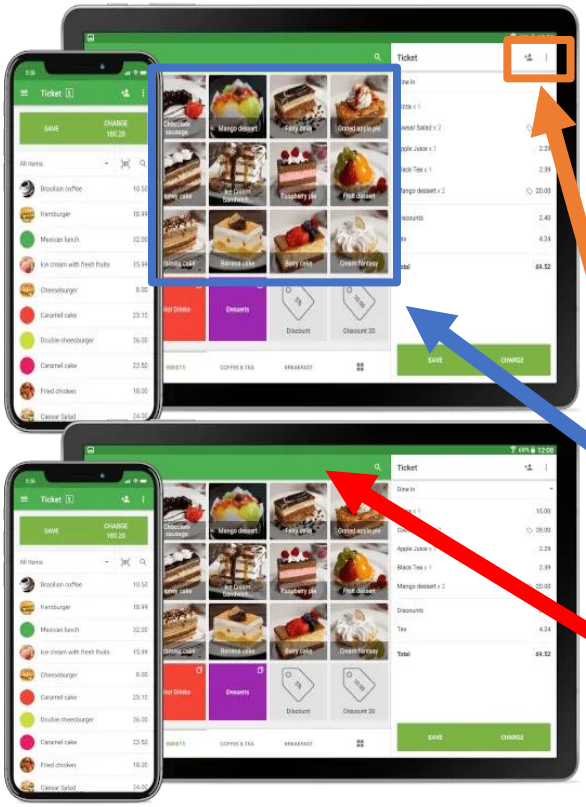
▪ The use of a built-in loyalty program that rewards frequent customers. Although this is a good feature, my stakeholders do not require it and would be challenging to implement; therefore, I will not bring this forward into my project.

Features I do not like:

- Many of the uncommon option buttons are within the main operation space reducing the number of items displayed. The buttons would be better contained within a drop-down menu on the toolbar, saving space.
- The total cost output location has the same colour background as the rest of the page (white), meaning that it does not stand out to the operator



2. ‘loyverse’ point-of-sale and stock management system allows users to transform smartphones into easy-to-use point of sale systems. The selling point of this system is that it has compatibility across numerous devices and enables the user to easily connect to external hardware such as printers providing the customer with the option of both a paper or digital receipt of their transaction.

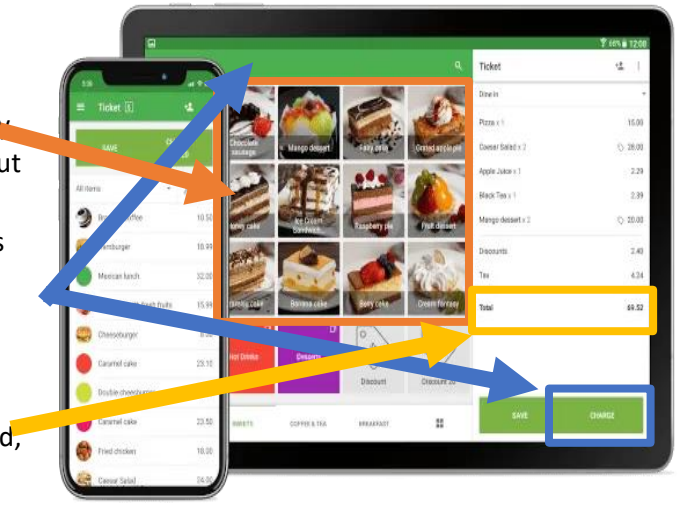


Features I would like to include:

- Compatibility for multiple devices is ideal to widen my target audience. However, this is unlikely to be brought forward into my project as cross-compatibility is likely to take a long time to achieve, and the time scale to complete this project is short. This is a potential limitation of my project.
- A drop-down toolbar takes up minimal space and provides many setting options and features that benefit system operators without compromising menu space. I will take this forward into my project.
- Pictures of items and their item names allow for easy reading and identification, allowing faster transaction times. I will bring this forward into my project.
- A search bar to find items that you may not know the location of on the system could increase the operator's efficiency, reducing transaction times. I am unlikely to include this in my project as the time frame to complete this project is short.

Features I do not like:

- The items do not have a price next to them on the menu; this could be a potential inconvenience to the operator as they may not have the ability to answer customers price concerns without adding the item to an open transaction.
- The green colour scheme. The green colour of the search bar is very similar to the green ‘CHARGE’ button making it difficult to locate for those that are not familiar with this point-of-sale system
- The total cost is output in a small font which makes it blend in with the price of each item on the transaction. It should be bold, as seen in other examples, making it easily identifiable to the operator.



3. ‘CAKE's point-of-sale system allows users to manage their catering businesses in one place with integrated online ordering, curbside management and reservations.

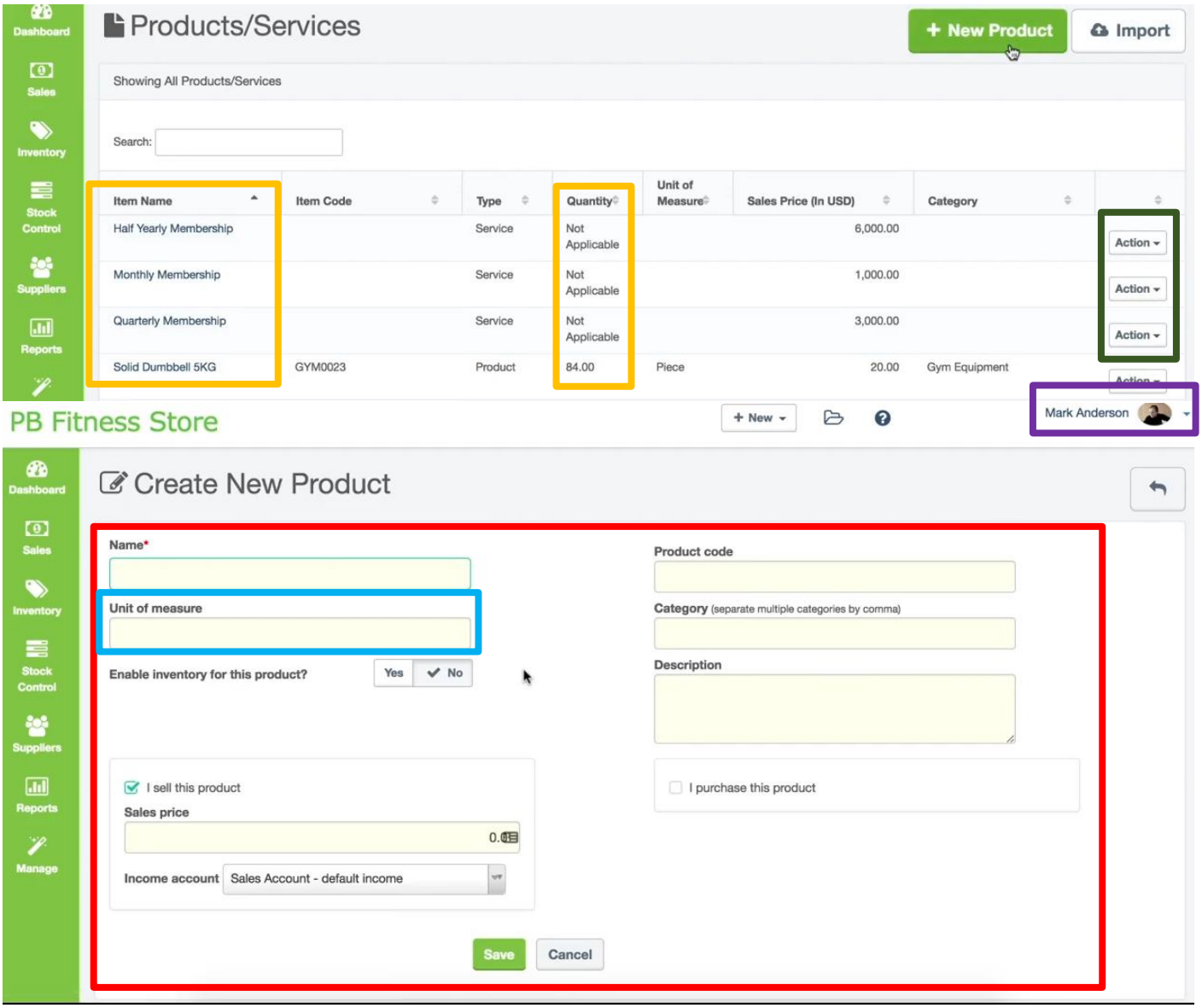
Features I would like to include:

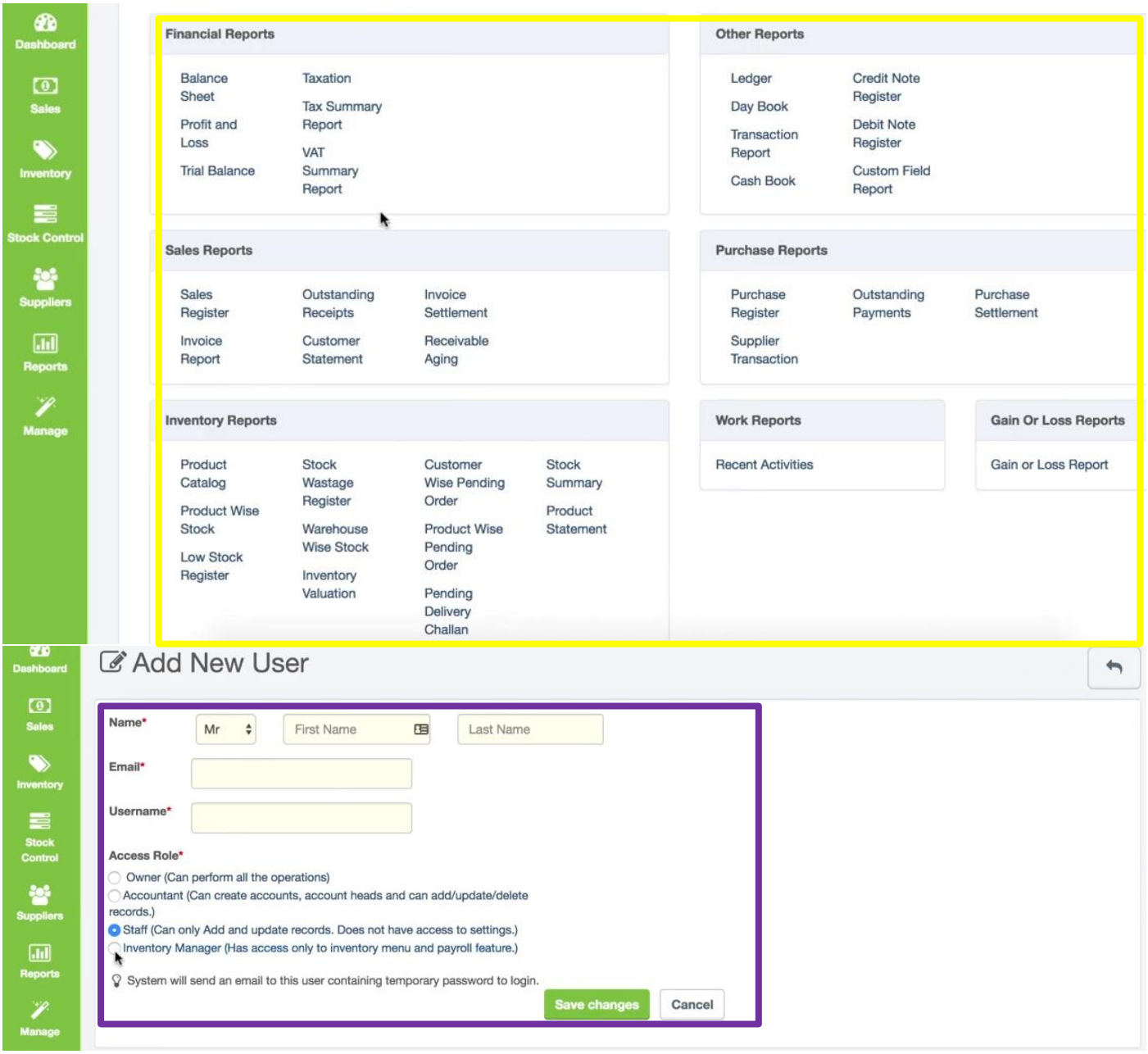
- The name of all items is in smaller text underneath the dish name. I would like to carry forward into my project as a desirable feature.
- The ability to customise the items a dish contains. Although I like this concept, I do not think I will have enough time to implement this. Therefore, I will include this in my project as a desirable feature.
- I like the layout of all buttons and features. All 'tabs' are in a contrasting colour to all dishes, and frequently used actions (void, split, print) are included in the main interface making it more efficient than previous examples by not taking up unnecessary space. I will bring this forward into my project.
- The search bar blends nicely with the colour scheme and is discrete yet extremely useful; this could increase the average transaction time, benefiting the end-user. A search bar is a frequently occurring feature in my research; however, I am unlikely to include this in my project as it will be time-consuming. I will therefore include it as a desirable feature in my project.
- Allows for multiple transactions to be open simultaneously and only closed once the operator completes the transaction. I want to bring this forward into my project as a desirable feature.

Features I do not like:

- Buttons are only colours with text, unlike previous examples that included images. Without images, the transaction time may increase due to operators having to read the name of each dish rather than visually identifying each dish.

4. ‘ProfitBooks Trade – Inventory management software’ is a stock management system that allows you to manage your inventory and systemise your company's operations.



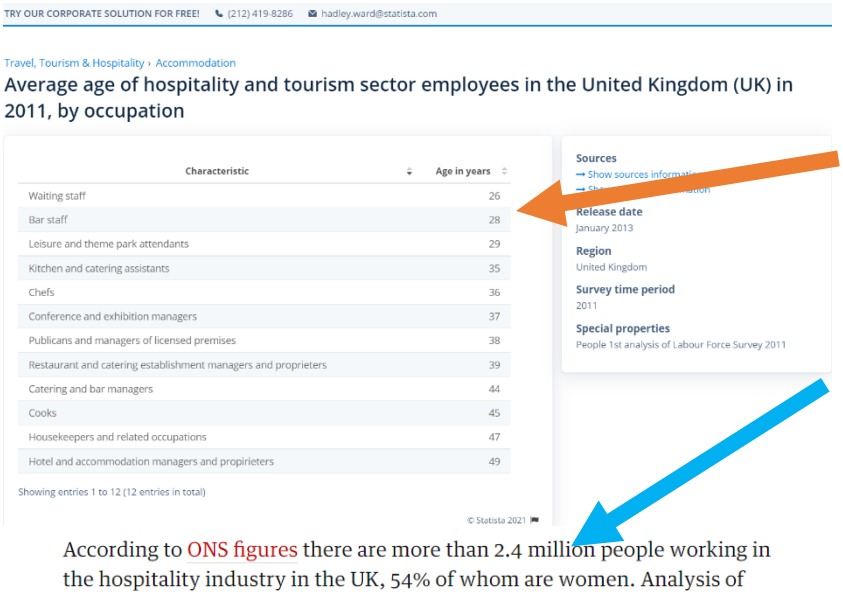


Features I like:

- Allows you to add new products to the stock management system with a user interface that uses multiple input boxes, making the required information about the product clear. I will create a similar menu in my project. **(RED)**
- Relevant data from the stock management database is displayed in a graphical way to the system operator, including the item name, quantity in stock, and the item's category. This will be brought forward into my project. **(ORANGE)**
- Allows the system operator to select a minimum stock value of a product when adding a new product to the stock management database. This will be brought forward into my project. **(LIGHT BLUE)**
- A dedicated section for all kinds of reports, although this is useful, I do not believe I will have enough time to complete this for my project. Therefore, this is a limitation of my project. **(YELLOW)**
- Ability to add a new user to the stock management system with different account types, I will bring this forward into my project if time permits **(PURPLE)**
- Small 'Action' buttons next to each item in the stock management user interface allow the system operator to edit the data regarding that specific item manually. I will bring this forward into my project. **(GREEN)**

User Demographic Research

5. Waiting and bar staff are most likely to use these systems; therefore, I searched the internet for statistics to understand this system's estimated demographic.



According to ONS figures there are more than 2.4 million people working in the hospitality industry in the UK, 54% of whom are women. Analysis of

Results:

This page on the Statista website placed the average age of waiting staff at 26 and the average age of bar staff at 28; therefore, I will assume that the average age of system operators will be 27 years old.

This article from The Guardian newspaper placed the percentage of female waiting staff at 54%; therefore, I will assume that the system is used evenly amongst all genders.

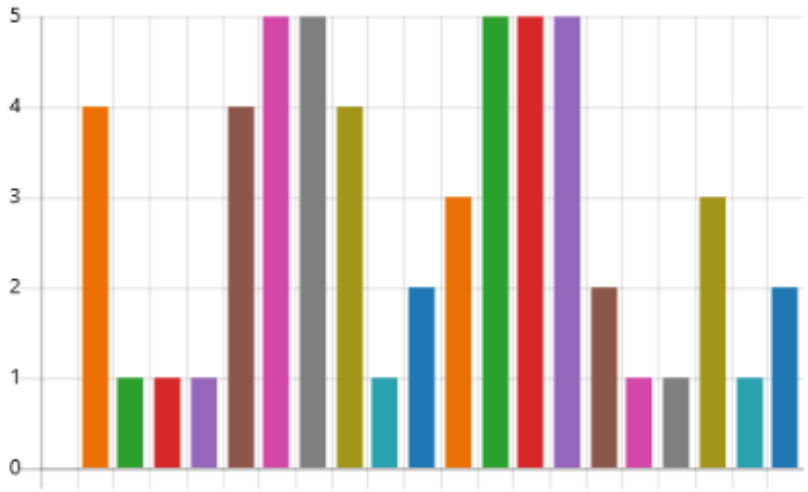
Research Survey

I sent this survey to local businesses and members of the public with varying levels of experience with point-of-sale systems. The results of this survey influence the deciding factor in the point-of-sale system's layout. This survey also discovered the level of expertise of potential end-users, indicating whether I will create a tutorial to train new users on the system or not.

Which of the following features do you think are ESSENTIAL in a point-of-sale and stock management system?

More Details

Items can be categorized in 'tabs' for example: a yellow shirt is categorized in a tab called 'shirts'	0
The 'Complete Transaction' button is large and easily identifiable	4
Each operator can have their own account	1
The point of sale system can work without Wi-Fi	1
Contains a built-in loyalty program	1
Each item in the user interface is clickable	4
The ability to refund a transaction	5
The ability to create a transaction	5
The ability to remove items from a transaction	4
A search bar to search for sale items	1
The colour scheme used does not strain the eyes	2
Different font size depending on button importance	3
Stock level automatically changes when an item is bought	5
When new stock has arrived, there is the ability to update the stock level	5
Alerts telling you when each item is low on stock via the system	5
Alerts outputting the contact information to a supplier via the system if an item is low on stock	2
alerts via text	1
alerts via email	1
Each menu item button has a corresponding name	3
Colour scheme is colour blind friendly	1
The ability to select a type of payment, cash or card	2

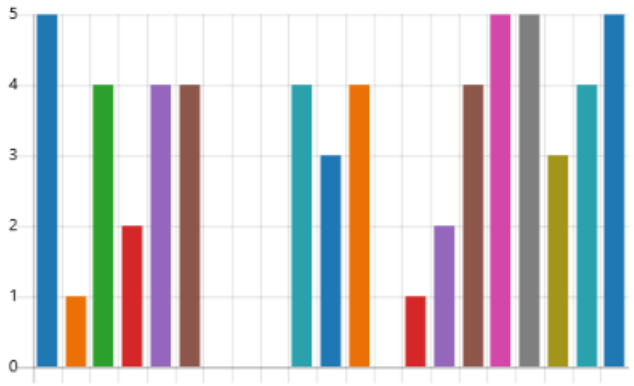


This question aims to inform me of the features that must be prioritised during the development of my project, and will likely become part of my success criteria. Those with much lower values, such as those with only 1 or 2 votes will likely be desireable features as it is unlikely that these features are essential if some of those surveyed did not select them.

Which of the following features do you think are DESIREABLE in a point-of-sale and stock management system?

More Details

Items can be categorized in 'tabs' for example: a yellow shirt is categorized in a tab called 'shirts'	5
The 'Complete Transaction' button is large and easily identifiable	1
Each operator can have their own account	4
The point of sale system can work without Wi-Fi	2
Contains a built-in loyalty program	4
Each item in the user interface is clickable	4
The ability to refund a transaction	0
The ability to create a transaction	0
The ability to remove items from a transaction	0
A search bar to search for sale items	4
The colour scheme used does not strain the eyes	3
Different font size depending on button importance	4
Stock level automatically changes when an item is bought	0
When new stock has arrived, there is the ability to update the stock level	1
Alerts telling you when each item is low on stock via the system	2
Alerts outputting the contact information to a supplier via the system if an item is low on stock	4
alerts via text	5
alerts via email	5
Each menu item button has a corresponding name	3
Colour scheme is colour blind friendly	4
The ability to select a type of payment, cash or card	5

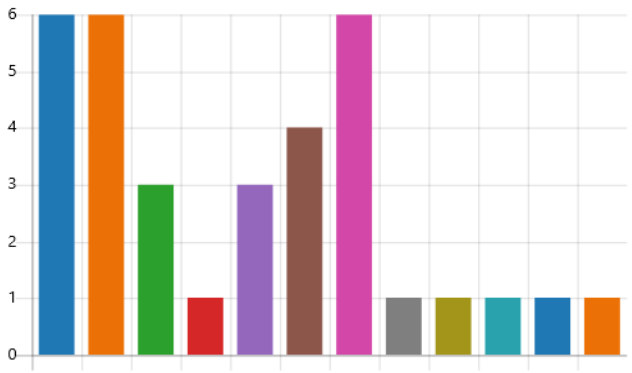


This question aims to inform me of the features that would be good to include during the development of my project, but are not necissary for the final project to function to its intended task. Those with much lower values, such as those with only 1 or 2 votes are less likely to be included in the project as it shows there is a very small market for such features. If such features are included, they will be of the lowest priority.

5. Which of the following colours do you believe reduce strain on the eyes?

More Details

light grey	6
dark grey	6
white	3
yellow	1
dark red	3
light green	4
dark green	6
black	1
dark blue	1
purple	1
orange	1
pink	1



This bar chart shows that light grey, dark grey, dark green, and light green should be key colours within the theme of this project. This question aimed to see which colours cause the least strain on the eyes allowing system operators to use the system for a greater length of time. Following the desired colour scheme will make the system more user friendly therefore widening the target audience.

4. Are there any features (other than those that featured in question 2) that you would find DESIREABLE in a point of sale and stock management system?

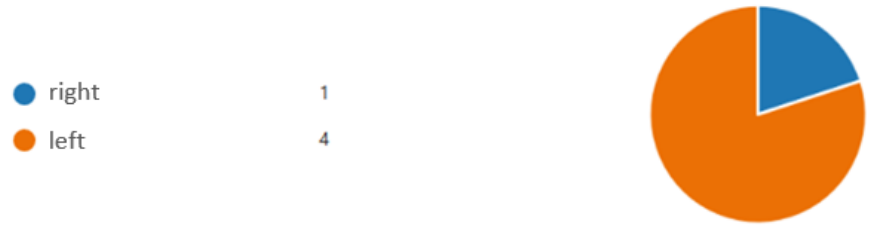
6 Responses

ID ↑	Name	Responses
1	anonymous	An option menu that allows for some features to be disabled as not all features included may be required
2	anonymous	Alerts should be resent to the operator if not acted upon within 10 minutes
3	anonymous	It may be helpful if different users have different permissions to the system depending on their company heirachy
4	anonymous	Ability for users to reset or change forgotten passwords
5	anonymous	No

This question aimed at identifying potential features of the system that had not been previously thought about. The results to this question will be used to add new features to the project.

Would you prefer the 'check out' and 'cost total' column of the user interface on the left or right side?

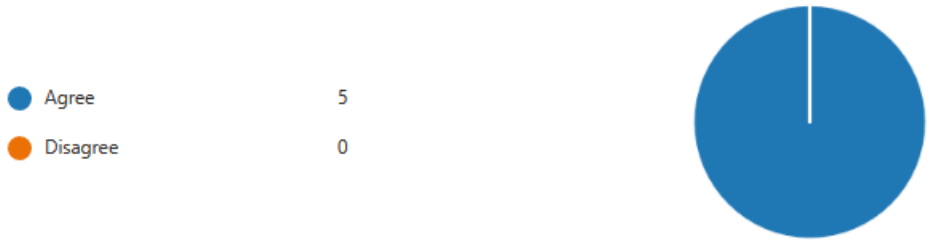
More Details



This question will inform which half of the point of sale page the total cost column will be placed. The results show that left is clearly the preferred position.

This point-of-sale system would be better if the cost of each item could be seen by the operator before it is added to the transaction.

More Details



The result to this question shows that the cost of each sale item should be visable to the system operator before it is added to the transaction. This will allow the system operator to convey the price of an item to a customer without adding it to the transaction.

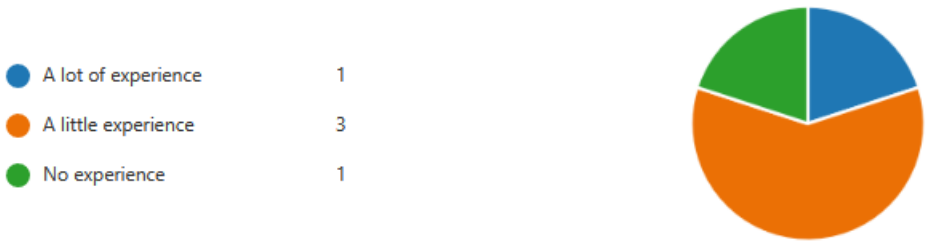
How much of the operation space should the above column take up?

More Details

The result to this question shows quite clearly that the transaction column should take up ¼ of the point of sale page.

How much experience have you had with point of sale and stock management systems

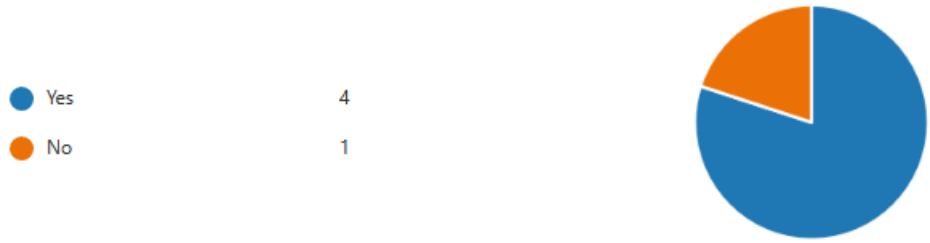
More Details



This question had very split answers, with no clear outcome. Therefore I will create a tutorial video as those with little to no experience will benefit from this.

. Would you prefer each clickable sale item to have a corresponding price displayed next to its name?

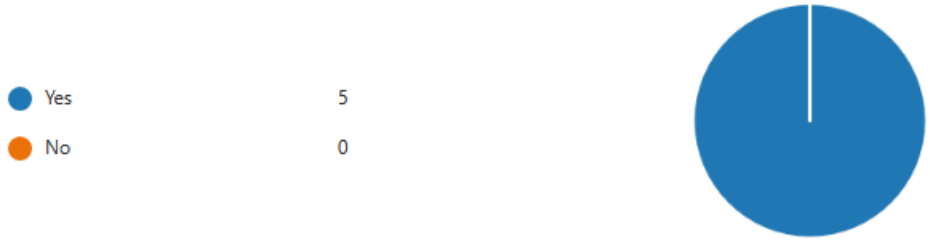
More Details



This question had a clear outcome, with clickable sale items definitely needing their corresponding price displayed next to the button.

. Would you prefer each clickable sale item to have a corresponding image displayed above its name?

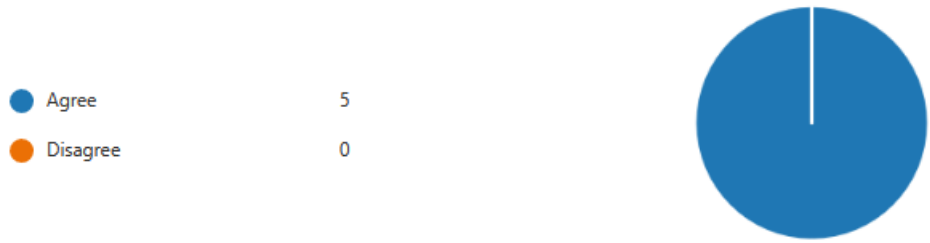
More Details



This question had a clear outcome, with clickable sale items certainly needing an image above them. This will make the system much more efficient, as system operators will be able to quickly view an image of a product, rather than having to read a name.

. It is important that unauthorised users cannot access the point of sale and stock management system

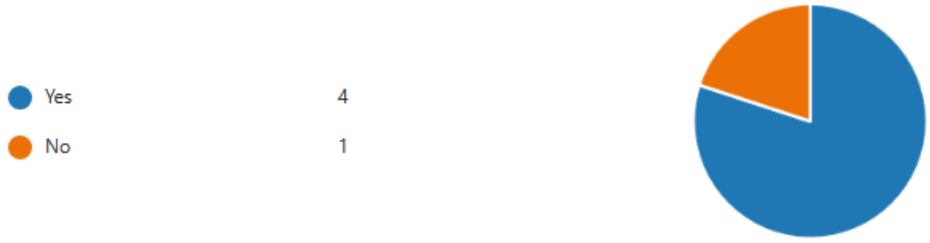
More Details



It is very clear that unauthorised users should not be able to access the pooint of sale and stock management systems. This means that I will certainly include the need to log in to an account in order to access these features.

. Is it important to have a home page where all parts of the point of sale and stock management system can be accessed easily?

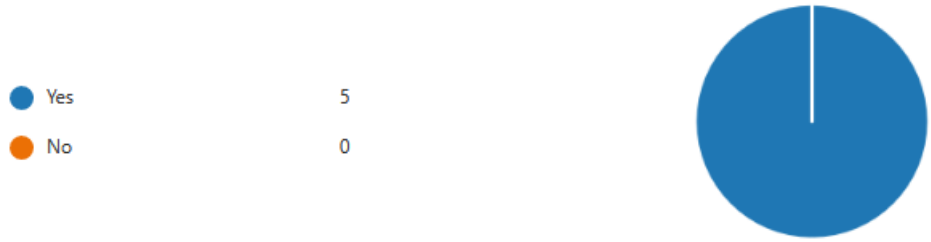
More Details



This question shows that it is clear that the home page should be a centeral hub with access to all other web pages.

. Should a point of sale system allow for as short of a transaction time as possible?

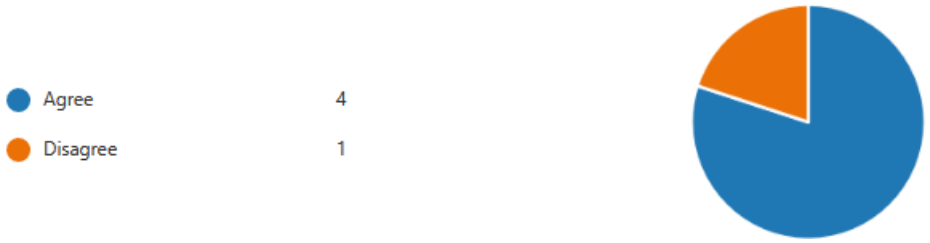
More Details



This shows that efficiency of the system is key, and that any features to increase efficiency of the system should be essential in order to produce a pleasant experience for those using this project.

. It is essential that a stock management system automatically updates the stock level of items.

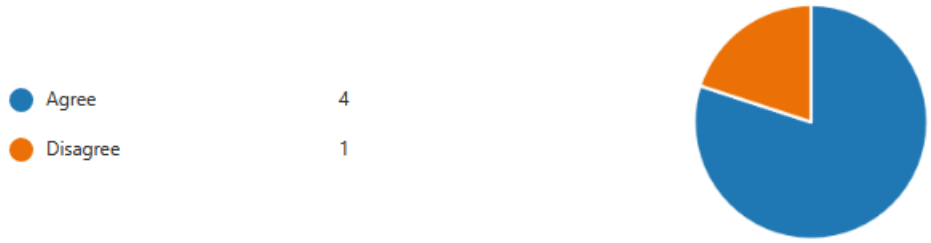
More Details



This question shows that users should not have to manually update the stock management system. Although they should have the ability to manually edit the stock management system, it is very important that the system automatically updates where possible to increase efficiency of this program.

5. It is essential that operators can manually edit the stock management database

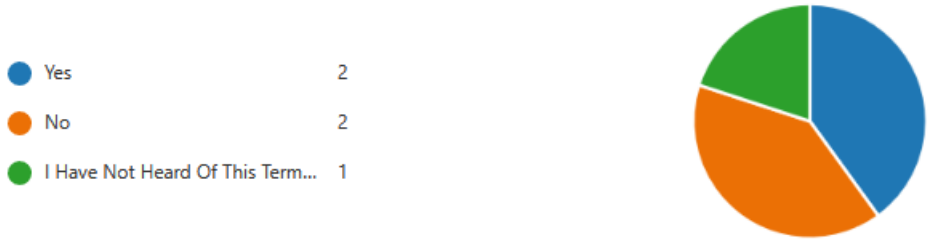
More Details



This shows that despite potential end users wanting to have an automatic stock management system, the manual aspect of this feature is still largely wanted, likely due to it enabling the system to be changed if an out of the box factor occurs such as wastage of stock.

7. Should Two Factor Authorisation Be Required When Creating An Account?

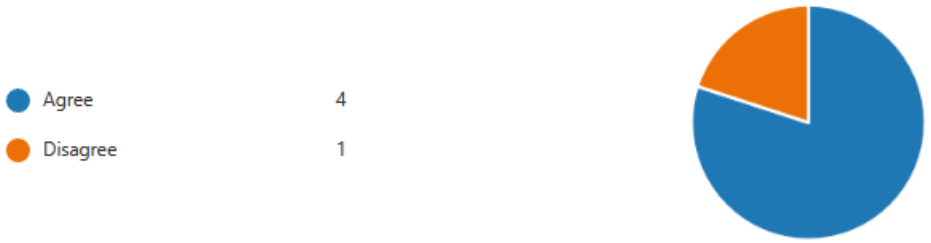
More Details



The results to this question were very split. This feature would be very difficult to implement and is not highly requested, therefore I doubt it will be included in my project, however it should be noted for future developments of the project.

8. The ability to create more than one menu for your company is desireable

More Details



This results to this question show that it is highly requested for a company to be able to create more than one menu within their account, therefore this should be a quite high priority desirable feature. This will also widen the target audience, therefore increasing the quality of my final project if included.

9. Alerts Prompting operators of questions to ask customers is a desireable feature (for example you add a drink to their transaction and the system prompts you to ask the customer whether they would like ice)

More Details

Agree	4
Disagree	1

. System operators should be alerted when an item is low on stock'

More Details

Agree	4
Disagree	1

Stakeholders

I intend my final project to be used by cashiers and waiting staff in local shops, specifically aiming towards shops in the catering industry such as cafes and bars with outside catering vans/food carts. As identified in my research, I expect the demographic of my stakeholders to be approximately 27 years old, whose genders are distributed evenly between male and female, that are interested in reducing transaction times in their workplace. My point-of-sale system will be used throughout every transaction, heavily reducing the workload on cashiers and waiting staff whilst also benefiting the establishment owner by increasing customer satisfaction due to reduced waiting times. My solution is appropriate to the stakeholder's needs as it provides a method of reducing transaction times compared to completing transactions using traditional methods such as pen and paper to calculate the total transaction cost. It also provides an operator-friendly interface with methods of controlling stock levels that would traditionally be completed by a member of staff counting each item whilst simultaneously writing down the stock level on a piece of paper. Not only is the traditional method time-consuming, but it is also prone to human error, of which my system will be much less prone to errors and will be, for the most part, updated automatically.

I have included my stakeholders in the design of my project via the survey conducted in my research. The stakeholders chose which features in a point-of-sale system they found essential and desirable; they also chose their preferred layout.

During development, I will make sure to update my stakeholders on the progress of the project. During testing, I will create a presentation to my stakeholders displaying completed features in use; a survey will then be conducted with my stakeholders, ensuring that each feature works as intended. If any features are not working as intended, I shall return to the user requirements and success criteria to amend that feature until it is working as intended.

After amendments, I will then ask my stakeholders to create a final evaluation. The stakeholders should first evaluate the project against how well it met the success criteria. They should then evaluate each feature against the user requirements. The stakeholders should mention likes and dislikes about the project and overall thoughts about how well the project met the user requirements.

User Requirements

No.	Requirement	Explanation and Justification	Importance
1	OTHER PAGES		
1.1	Home Page	The home page will be the web page that the user first loads and will provide an area to educate new users of 'POSSY' on using the system and inform them of all the features 'POSSY' makes available. The home page will also contain a toolbar at the top of the web page containing button links to all other 'POSSY' web pages, for example, the log-in and sign-up pages. In my research, many of the web pages aimed only at selling their product. I could not find a web-based point of sale system with a home page to educate new users on the system. And, as identified in my survey, 80% of users have little or no experience with a point of sale and stock management system.	**REQUIRED**
1.2.0	Sign-up Page	The sign-up page is a page that allows new users to create an account for the 'POSSY' system. New users must be able to sign up their company to the 'POSSY' system using a username, an email, a phone number, a password, and their company name. New users should not be able to create a new account with a username that already exists. Once a user has completed sign-up, their details will be placed into a 'Users' database and they should be re-directed to the home page. A sign-up page is needed to create new users allowing them to create their menus. Users could not create their accounts without a sign-up page, only permitting hard-coded users access to the system. This was further reinforced by the 100% of users in my survey agreeing with the statement 'It is important that unauthorised users cannot access the point of sale and stock management system'.	**REQUIRED**
1.2.1	The ability for different account types at sign-up with varying permissions to the system	Users should be able to choose an account type (admin/ standard) at sign-up. A standard account will only be able to open and close transactions and must be approved access to a company's menus before accessing them. An admin account will have access to the same permissions as a standard account alongside the ability to edit the menu, change the cost of each item on the menu and be provided with access to the stock management database table. This was something brought up by a user in my survey. The user mentioned that it might be helpful for users to have different permissions depending on their company hierarchy.	**DESIRABLE**
1.3.0	Log-in Page	The log-in page will allow users who have already registered to the 'POSSY' system to log in using their username and password. All entered details will be validated against those they signed up with using the 'Users' database. If the user inputs an incorrect username, they should be re-directed to the sign-up page. If the user inputs a correct username and password, they should be granted permission to the account and its menus. If the user inputs an incorrect password, the system should prompt the user to re-enter their password. A log-in page is needed to allow users access to their accounts. Without user accounts, any person with access to the 'POSSY' website would be able to access and modify the menus of all companies using the system. This was seen in my research of the 'Erply' point-of-sale system, which contained a system for different users to log in to their company accounts.	**REQUIRED**

1.3.1	The ability for users to reset their password if forgotten	If a user forgets their password, they should be able to select a 'forgot password' hyperlink placed on the signup page, which causes an edit to the web page that prompts them to enter their email. If the email entered matches that of their account, they should be sent an email containing a link to a web page. This web page should ask the user to enter a new password and will update the 'Users' database with their new password. This was a desirable feature suggested by a user in my research survey that I believe is useful and will not be too difficult to implement in this project.	**DESIRABLE**
1.4	Daily total	A section of the drop-down menu included in the point-of-sale system that shows system operators the daily total for that day. This total should be reset at the end of each day.	**ESSENTIAL**
1.5	A 'credentials' database table	The 'credentials' database table will be the database table containing the details of all accounts that have signed up to the 'POSSY' system. This database should include a username, a password, an email, a phone number, and a company name. The 'credentials' database table is essential in allowing users to log in to their accounts.	**ESSENTIAL**
2	POS system		
2.1	User Interface	The user interface will provide the system operator with an area to add any item in the point-of-sale system to a transaction within 10 seconds. Each item should have a button that, when clicked, the item is added to the transaction if it is in stock. If the item is not in stock, an alert should be sent to the system operator. The total cost of the transaction should be displayed to the system operator via text in a column on the left-hand side of the system (as chosen in a research survey). The 'total cost' and 'complete transaction' buttons should be displayed on the left-hand side of the page in the same column as the total cost. These buttons should be located at the bottom of this column. Above these 'total cost' and 'complete transaction' buttons, the column should contain the name of each item added to the transaction followed by its sale price in smaller text, with the column taking up approximately ¼ of the user interface. Once a transaction is marked as completed, the point-of-sale system should interact with the stock management system, deducting stock levels as appropriate. In my research, I identified that a user interface of some degree appeared in all existing solutions therefore making it essential that I also included a similar user interface in my project. Specific features of different user interfaces that I have included are highlighted through my research.	**REQUIRED**
2.2.0	Each sale item has its corresponding button labelled with its item name	Each button should be a clickable box that should add the item to the current transaction and update the current total transaction price by a predefined amount when clicked. In my research, all existing solutions of a point-of-sale system used buttons that the operator can press, which, in turn, will update the total cost of the transaction. 60% of my users labelled this as an essential feature in a point-of-sale system.	**REQUIRED**
2.2.1	Each button has a corresponding image	Each pressable button should have a background image of the item alongside its name. In my research, 100% of users agreed that it would be preferable for each clickable sale item to display a corresponding image above its name. This was also seen in many of the examples of point-of-sale systems during my research.	**DESIRABLE**
2.2.2	Each button has the cost of the item labelled next to it.	Each pressable button should have the cost of the item displayed below the item name. In my research, 100% of users agreed with the statement: 'This point-of-sale system would be better if the operator could see the cost of each item before it is added to the transaction' When shown a point-of-sale system without this feature.	**DESIRABLE**

2.2.3	System operators should be able to create new buttons	The system operator should be able to enter an 'edit' mode which allows them to add new buttons/ edit buttons allowing new items to be added to the point-of-sale system. This feature was not seen in my research; however, it is essential that new items can be added to the point-of-sale system without being hard-coded.	**REQUIRED**
2.3	A set of tabs that store items under a category	Each tab should have a named category which, when pressed, expands, displaying all sale item buttons under that category which can then be added to the current transaction. In my research, 100% of users selected the statement 'Items can be categorised in 'tabs'. For example, a yellow shirt is placed in a tab called 'shirts' as a desirable feature in a point-of-sale system.	**DESIRABLE**
2.4	The ability to refund a transaction	Suppose a transaction is accidentally completed when it should not have been, or a customer returns a product. In that case, the amount to refund can be manually entered and deducted from the companies' daily total. This will not update the stock management database table as it is assumed that the items will not go back on sale. In my research, 100% of users selected 'The ability to refund a transaction as an essential feature of a point-of-sale system.	**ESSENTIAL**
2.5	The ability to create a transaction	Once a transaction has been completed, a new transaction should be either opened automatically or opened manually by the system operator. In my research, 100% of users selected that 'The ability to create a transaction is an essential feature of a point-of-sale system.	**ESSENTIAL**
2.6.0	The ability to complete a transaction	Once all items have been added to a transaction, and the customer has paid, the operator presses the 'complete transaction' button. If all items in the transaction are still in stock, the transaction is then completed and closed. The point-of-sale system then interacts with the stock management system, editing the stock levels in respect to the items bought. In my research, all existing solutions of a point-of-sale system included a 'complete transaction'/'checkout' button which closes the current transaction.	**ESSENTIAL**
2.7	The ability to remove items from a transaction	The operator should be able to remove items from a transaction which then deducts the cost of that item from the total sale price. This should be done in the form of a '-' button next to the item added to the transaction. In my research, 80% of users identified 'The ability to remove items from a transaction' as an essential feature in a point-of-sale system.	**ESSENTIAL**
2.8	The ability to add items to a transaction in multiples	Upon selecting an item to add to a transaction, the same item should be added to the transaction by pressing a '+' button next to the item added to the transaction. This feature did not come up in my research; however, upon discussing it with my stakeholders, they agreed that it would be desirable to reduce transaction times.	**DESIRABLE**
2.9	Compatibility with multiple device types	The point-of-sale system should be useable across pc & mobile device types, with each device having its own layout depending on screen size. This feature was presented in my research by the 'loyverse' point-of-sale and stock management system, which had compatibility for phones, tablets & desktop pcs.	**DESIRABLE**
2.10	A search bar	The point-of-sale system should have a search bar that allows the operator to look up a menu sale item if they cannot find it in the user interface. In my research, 80% of users identified 'A search bar to search for items that you cannot immediately find' as a desirable feature in a point-of-sale system.	**DESIRABLE**

2.11	Option Menu	The options menu should be a drop-down menu containing options that the operator could enable or disable, for example, the ability to disable alerts from the stock management system. This was a suggestion by a user in my research survey. They mentioned that it might be entirely possible that some operators may not require specific features; an example could be the stock management system.	**DESIRABLE**
2.12	Ability to create multiple menus for one single company	Companies should be able to create more than one menu allowing for alterations on products/ prices for different store locations. In my research survey, 80% of users surveyed agreed with the statement 'The ability to create more than one menu for your company is desirable.'	**DESIRABLE**
2.13	Alerts to prompt operators	Message prompts are displayed to the operator in the user interface to make sure operators ask customers key questions. Suppose the customer ordered a drink, and the operator inputs the drink into the system. In that case, the system could prompt the user with 'would you like ice in your drink?' with yes/no buttons, which automatically alter the order depending on the operator's response. In my research survey, 80% of users agreed with the statement 'Alerts prompting operators of questions to ask customers is a desirable feature.'	**DESIRABLE**
2.14.0	The ability to customise dishes	An edit icon should be shown next to each sale item added to a transaction. This edit icon should allow the system operator to add and remove products from the sale item. This feature appeared in the 'CAKE' point-of-sale system during my research. Upon discussing this feature with my stakeholders, they agreed that it would be desirable as it would improve the accuracy of the stock management system. This feature is particularly useful for those in the catering industry who may want to add or remove items from a sale item at the customer's request.	**DESIREABLE**
2.14.1	Display added or removed items below the dish name on the sidebar	A '+' or '-' sign should be displayed underneath the dish's name that was added to a transaction and customised. Next to the signs should be the name of items added to or removed from the dish. A similar feature appeared in the 'CAKE' point-of-sale system during my research. In this system, all items on the dish are displayed. However, I do not believe this is necessary; therefore, I will only include changed items to save space on the user interface. This will be useful for system operators to check with the customer what they have ordered.	**DESIREABLE**
2.15	The ability to open multiple transactions at one time	The point-of-sale system should allow multiple transactions to be created at once by pressing a create transaction button. Each transaction should have a unique ID created by the system operator upon creation. There should be the ability to select the transaction number of the desired transaction and submit the value via a button, where all relevant data regarding that specific transaction should appear on the screen. If a transaction number is chosen that does not already exist, a transaction should be opened using the entered transaction ID. This feature caters for restaurants and bars where customers will often not pay until after being provided with the items they have purchased. This system was seen in the 'CAKE' point-of-sale system during my research.	**DESIREABLE**
2.16	Sale Item Database table	A database table containing the name of each sale item, the sale price, the category it is contained within and a list of sub items that the item contains. This sale item database table is essential in storing data regarding the menus contained within the point-of-sale system. Without this database, the system operator would have to manually enter this data every time the program is shut down.	

3	Stock Management System		
3.1	Stock management database table	The stock management database table should be a database table containing the name of all items currently for sale alongside their current stock level, a pre-set minimum stock level, the name of the item supplier, an email for the supplier and a phone number for the supplier. It should also contain a column that indicates whether a specific item has stock currently ordered or not. This was not visibly seen in my research; however, such a feature would have existed in the background as it is required for a stock management system to function.	**ESSENTIAL**
3.2.0	Stock management user interface	The stock management user interface should display the name of each item in the stock management database table, the quantity currently in stock and the minimum stock level. This should be displayed in the form of a table. This was seen in the 'ProfitBooks Trade' Inventory Management System in my research. The feature is key to system operators updating the stock management systems in a time-efficient manner.	**ESSENTIAL**
3.2.1	Manual edits to the stock management database table with a user interface	Manual edits should be done in the form of an 'Edit' button displayed in the last column of each table row in the stock management user interface. The 'edit' button should take the user to a new web page with a user interface that allows the operator to manually edit each item's stock level and change the pre-set minimum stock value, supplier email, & supplier phone number. The operator should also be able to manually edit whether an item is currently on order or not. These should all be edited using text input fields. In my research survey, 80% of users agreed with the statement 'It is essential that operators can manually edit the stock management database'. The use of an edit button was seen in the 'ProfitBooks Trade' Inventory Management System in my research.	**VERY DESIRABLE**
3.3	Ability to add new products to the stock management database table	A '+ New Product' button should be displayed in the stock management system of my project. The button should take the system operator to a new webpage where input fields are displayed for the item name, minimum stock level, order status (ordered, unordered), supplier phone number, supplier email, and item category can be entered. Once entered, a 'save' button should update the stock management database table to include the new item when pressed. This feature was seen in the 'ProfitBooks Trade' Inventory Management System in my research. This feature is key to allowing system operators to add new items to the stock management database table efficiently.	**ESSENTIAL**
3.4	Automatic edits to the stock management database table	Once a transaction has been completed, the stock management system should be updated by the stock levels of each item bought being reduced. In my research survey, 80% of users agreed with the statement, 'It is essential that a stock management system automatically updates the stock level of items.'	**ESSENTIAL**
3.5	Include the date and time stock is due to arrive in the stock management database table	When re-ordering stock, the date and time stock is due to be delivered should be automatically entered into the stock management database table. When this date and time matches with the system date and time, the operator should be displayed a prompt with two buttons, 'delivered', which will automatically set the item to not being on order and 'undelivered', which will leave the stock management database table unchanged. This feature was a desirable request from my stakeholders. It did not appear in my research.	**DESIRABLE**

3.6.0	Alerts to the point-of-sale system user interface	Alerts should be sent to the point-of-sale system user interface whenever an item is below a pre-set minimum stock level. The alert should contain the name of the product low on stock, the supplier's name, and any contact information for the supplier. The alert should have two buttons that, when pressed, will cancel the alert. The 'ordered' button will automatically update the item as ordered in the stock management database table, and the 'unordered' button will make no edits to the stock management database table. In my research survey, 80% of users agreed with the statement 'System operators should be alerted when an item is low on stock.'	**ESSENTIAL**
3.6.1	Re-send system alerts if not acted upon	If the operator does not order stock when sent a low stock alert, the alert is sent again after 20 minutes. A user suggested this feature in my research survey, which I thought would be relatively simple to implement and beneficial to my stakeholders. The user suggested a time frame of 10 minutes; however, I believe this is too soon; therefore, I will use 20 minutes.	**DESIRABLE**
3.6.2	Alerts via email	Operators should have the ability to opt-in to email alerts via the option menu 100% of users in my research survey labelled alerts via email as a desirable feature in a stock management system.	**DESIRABLE**
3.6.3	Alerts via text	Operators should have the ability to opt-in to text message alerts via the options menu. 100% of users in my research survey labelled alerts via text as a desirable feature in a stock management system.	**DESIRABLE**
4	System & Software Requirements		
4.1	Internet Access	As the project is web-based, it will be required that users always have access to the internet to use the system. The system will not be available offline as one of the sole purposes of this project is that it can be accessed from anywhere, requiring internet access. This was outlined in my introduction and is one of the reasons that this project appealed to my stakeholders.	**REQUIRED**
4.2	A device with an installed web browser	A web browser is required to access a website, and without a device with an installed web browser, it would be impossible to access my project as my project is entirely web-based.	**REQUIRED**
4.3	A server host	A server host will be required for the project to be always accessible via the internet. Without a server host, the project cannot run, and, therefore, users cannot access the 'POSSY' website.	**REQUIRED**

Success Criteria

These are the criteria I will use at the end of my project to assess whether my project was successful and has met the end goal, a point-of-sale and stock management system.

- A main menu as described in requirements 1.1
    - Links to log-in, sign-up and 'My Menu' pages
- A 'Users' database containing the account information of users after they have completed sign-up. It should contain:
    - Username (required)
    - Password (required)
    - Company Email (Required)
    - Phone number
- A sign-up page as described in requirements 1.2.0
    - User's entered details are stored in the 'Users' database
- A log-in page as described in requirements 1.3.0
    - User is allowed access if details entered match a user in the 'Users' database
- A point-of-sale system user interface as described in requirements 2.1
- Clickable button to add sale items to transactions as described in requirements 2.2.0
    - The ability for new sale item buttons to be added to the system as described in requirements 2.2.3
- Ability to refund a transaction as outlined in requirements 2.4
- Ability to create a new transaction as outlined in requirements 2.5
- A sale item database table as outlined in requirements 2.16
- A stock management database table as outlined in requirements 3.1
- The ability to add new products to the stock management database table as outlined in requirements 3.3
- Stock management database table automatically edited after a transaction is complete as outlined in requirements 3.4
- Manual edits to the stock management database table to compensate for out of the box factors such as wastages as outlined in requirements 3.2.1
- Stock arrival date contained within the stock management database table used to prompt operators whether the stock has arrived as outlined in requirements 3.5
- Low stock level alerts output to the operator as detailed in requirements 3.6.0
- The project is hosted by a server as outlined in requirements 4.3

Limitations

As I am the only developer of this project, I must place some limitations on my intended solution to ensure that I can create a complete, working solution that has solved the problems outlined in my introduction.

The most significant limitation of this project is the number of features that I can implement into the point-of-sale and stock management system. Whilst I would like to include account security and encryption into my user databases, I cannot include this. These features would drastically increase the development time beyond the deadline provided. As such, it is unlikely that I will include them in this project. Not including security features will make my project prone to malicious attacks. User's details will be visible if an unauthorised user accesses the credentials database table as their details will not be encrypted. Therefore, I advise that users of my system use incredibly secure details. I suggest that the passwords used on this website are unique, therefore, are not in use on other websites.

Another limitation of this project will be its aesthetic capabilities. I have no prior experience in the design of professional web pages; therefore, I will not be able to provide visually appealing web pages for the most part due to my lack of skills. I will try to make pages appear as professional as possible whilst keeping them simple to avoid confusion.

What Makes This a Computable Task

This project is suitable to be solved via computational methods for several aspects, for example:

- It will take much longer for a human to manually write down the order of a customer and calculate the cost of a customer's order. By solving this task via computational methods, the efficiency of the task to solve will be greatly increased, as computers can perform calculations in milliseconds that will take humans at a minimum several seconds. Humans are also more susceptible to miscalculations compared to a computer, hence solving this task via computational method makes the task much more accurate.
- There will be decomposition. Each section of the system will be broken down into smaller, simpler to solve problems. This will make the project more manageable to program by providing a much clearer overview of the task to be solved. The problem can for example be broken down into the stock management system, the point-of-sale system and the tutorial/home pages. This will also help when splitting the code into functions and between files, as it will define clear boundaries between sections of the project.
- My project will be modular – the decomposed problem will be very useful in splitting my project into separate functions and files. This will mean that the project can be easily added to and edited in the future by altering specific functions and files of which it is clear which section of the project each file refers to. Functions within files can be re-used. Hence reducing the amount of time required to code the solution.
- There will be abstraction – I will not use methods that produce dynamic web pages in my project such as ajax as this will take up a lot of time. Instead, I will alter each web page by refreshing the web page with new data, hence saving a lot of coding time. The system operator will also only be able to view the necessary information on the screen at any one time. For example, abstraction will be used when choosing a sale item to select. The system operator will only be able to view the name of the sale item and perhaps an image of the sale item, they will not be able to view the contents of a particular item until it is added to the transaction.

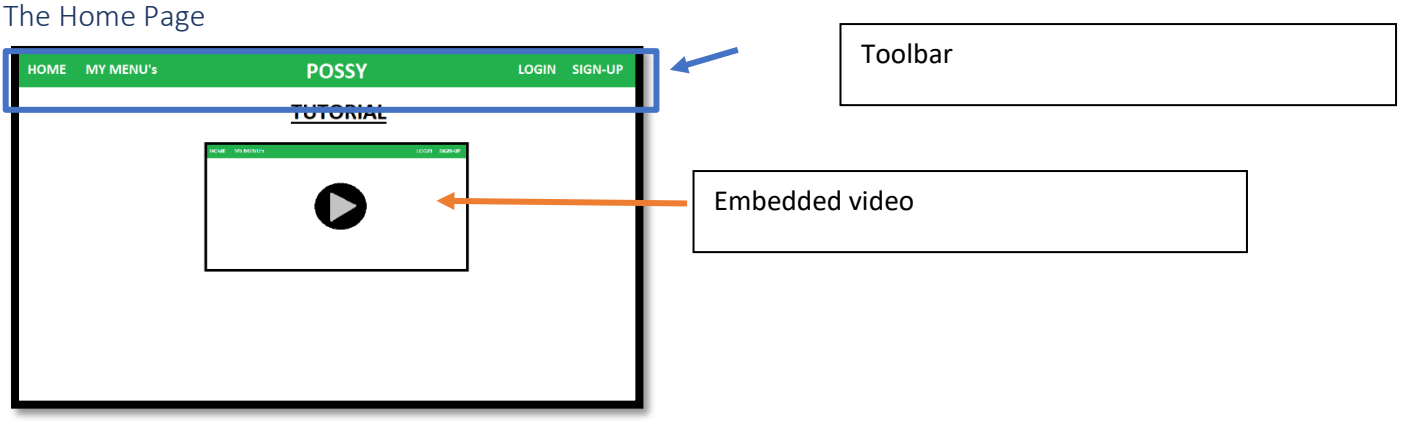
Design Stage

User Interface Design

The toolbar has been placed at the top of every page to be easily visible and identifiable, allowing it to be seen easily by new users. The buttons are large to make it easy for all users to press them quickly without the need for precision clicking. The colour scheme throughout the project has been primarily decided by those in my research survey who chose their most preferred colours for reduced strain on the eyes are the colours green and grey. I have also included the colour black to contrast the white background in many areas, which also helps to highlight the most important areas of each web page. In all user interfaces, bright green buttons are used to display positive actions. These buttons are large so that they are easily clickable and identifiable. In contrast, red buttons display actions that you may need to take caution before pressing. The system operator uses any text input fields to identify the data needed to be entered quickly and in which positions by viewing the text in bold to the side of each input field.

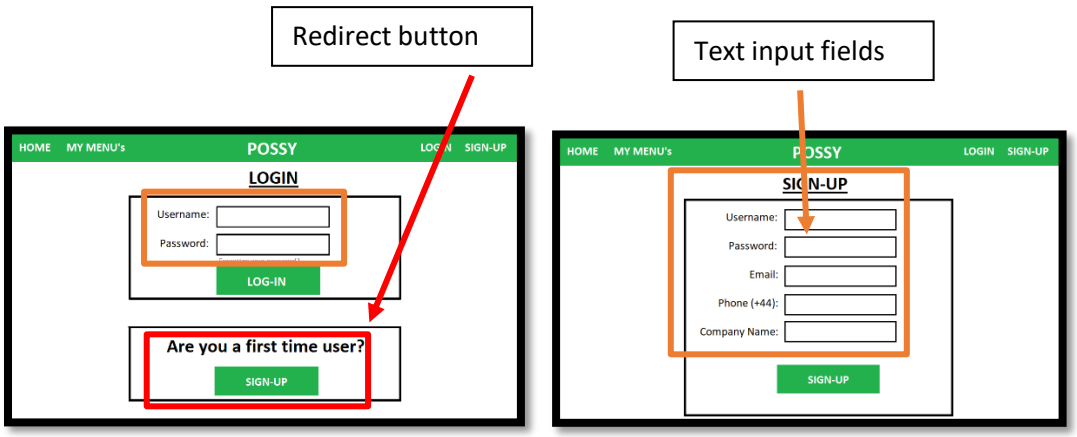
All web icons, red buttons, bright green buttons & text input fields are conformant to the standard convention so that the experience is intuitive.

The Home Page



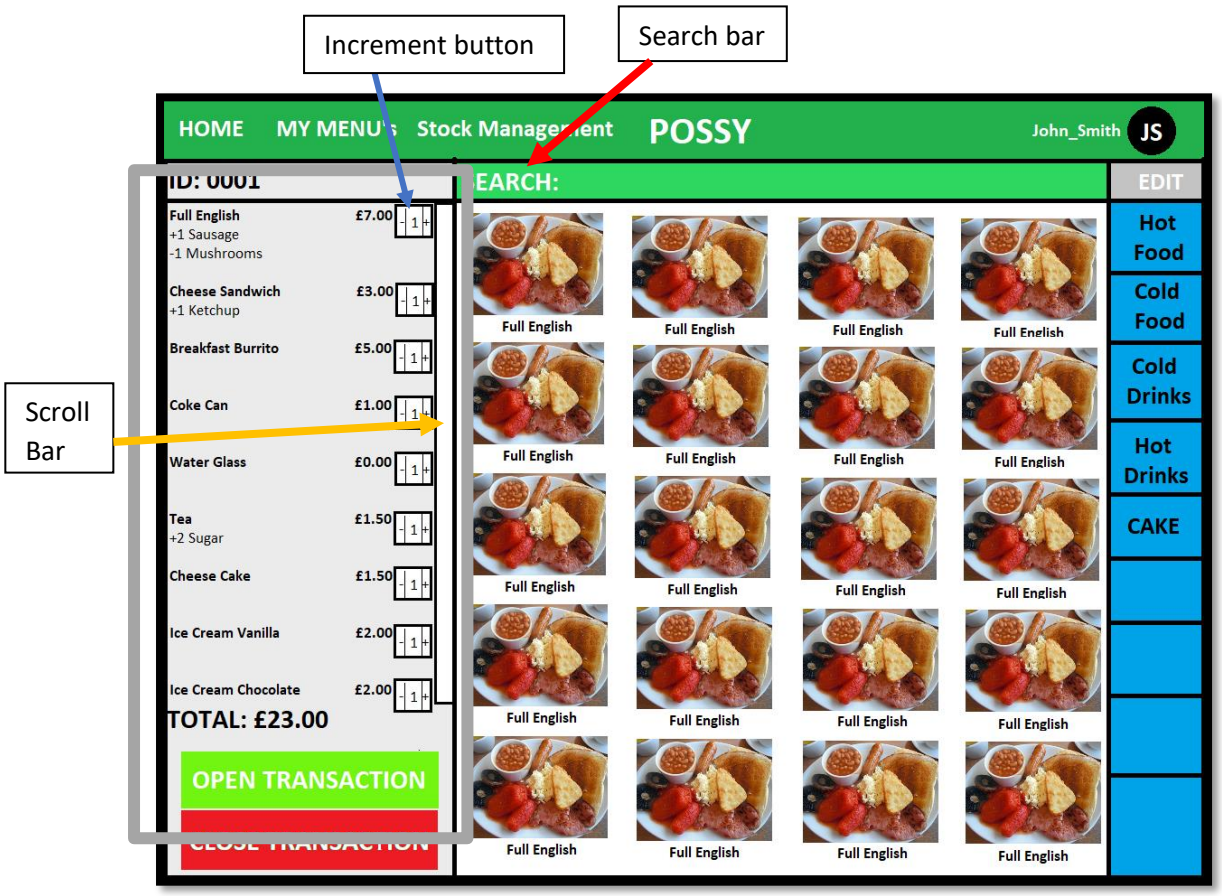
In my research survey, 80% of participants identified as having little or no experience with a point-of-sale system. This caused me to embed the tutorial video into the centre of the landing page so that it is easily visible and accessible by all users without opening new browser tabs.

Signup & log-in Pages



On both pages, large text input fields are used down the centre of the web page in the conventional style so that their purpose is intuitive and they are easily seen. The font type is Calibri as it is read large and clear, ensuring that the type of details required in each text input field is clear. The button at the bottom of the log-in page will redirect the user to the signup page, making it simple to create a new account if users have navigated to the wrong page. This button was placed at the bottom of the user interface as it is less important than the key focus of the page, and it will not be required by many users visiting this page.

Point-of-sale Page



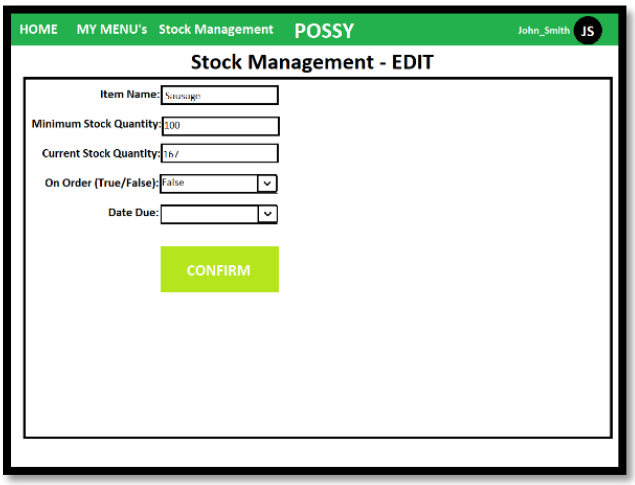
The transaction menu was placed on the left-hand side of the web page as my research survey identified that 80% of survey participants preferred the 'check out' and 'total cost' column on the left-hand side of the user interface. Along the edge of the left-hand column, I chose to use increment buttons to add items in multiples. I chose these as it allows for a much faster addition of items, and they are of the conventional style, making them easily identifiable. These increment buttons also allow items to be removed from a transaction, not just added. To the right of the increment buttons is a scroll bar; this enables the system operator to see an infinite number of items on the current transaction, rather than just a few as it would be without the scroll bar. The scroll bar has been made to look like conventional scrollbars to be easily identifiable by users. The search bar was included at the top of the page; however, it is quite small. This is because I do not expect it to be a frequently used feature, and the program does not need it to function; it just may be a preference by some users. I chose a similar colour green search bar to the green of the toolbar, making it identifiable yet seamlessly blend in with the colour scheme, ensuring that it is not distracting. The search bar was placed at the centre of the web page under the toolbar as it fits well with the rest of the page, making the web page uniform. This is also the same case for the small edit button that was also placed to make the page look uniform. The button is small as it will not be used frequently after primary setup; therefore, it does not take up unnecessary space on the page. The tab list was placed on the right-hand side of the page to allow for more sale items to be included on the page. If placed towards the top or bottom of the page, four fewer sale items would be included in each section. These buttons were made blue so that they are easily distinguishable from all other features on the page. If you add an item to a transaction and the item is out of stock, there will be a visible alert that the system operator can see informing them that they cannot add that item to the transaction.

Low Stock Alert

Drop-down boxes

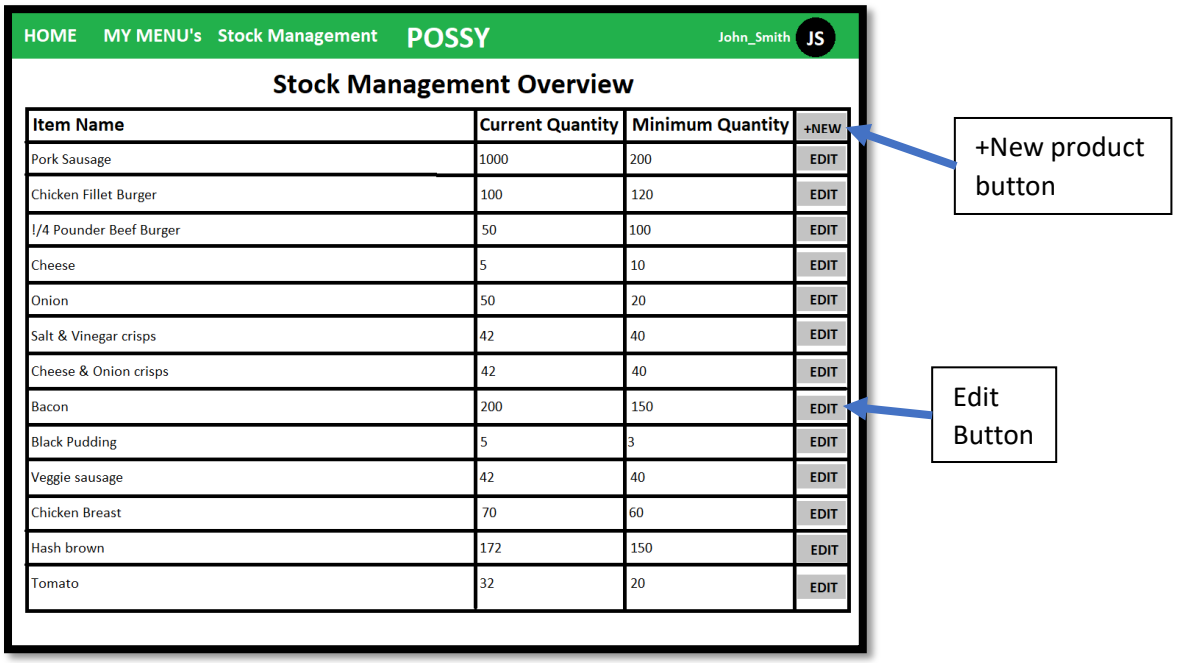
The information displayed is large and bold, allowing any system operator to quickly identify which items are below their minimum stock level and by how much. “LOW STOCK ALERT” is in red as this prompts the system operator that this alert is crucial and that they should read it. I used drop-down boxes to display the date and time as this prevents system operators from inputting invalid data and prevents data from being entered in an incorrect format. The ordered and unordered buttons are large and clear, ensuring that the system operator sees them quickly to reduce the time spent on this alert. Like previous examples, the green and red colours were chosen due to their connotations.

Create Sale Item Page



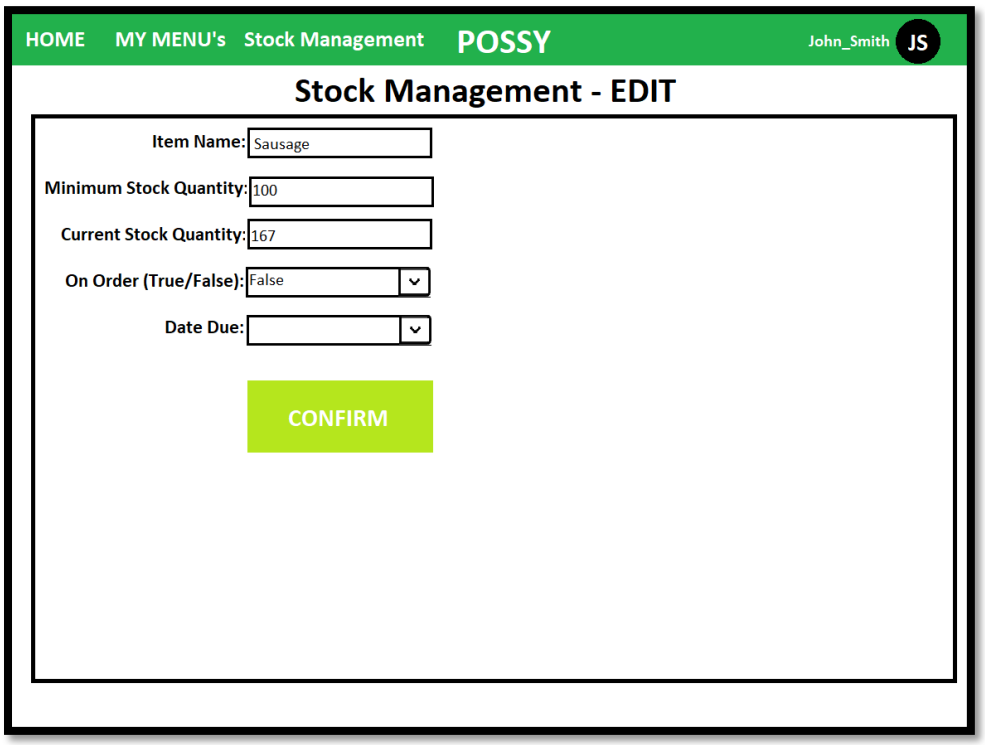
A drop-down box has been used for the “category”, “On Order” and “Date Due” fields to prevent invalid data from being entered. The price, current, and minimum stock level input fields have been created to allow only numerical characters, preventing invalid data such as alphabetical characters from being entered into the fields. The “upload image here” box allows for an image to be uploaded to the sale item. The large upload box has been created for a preview of the image to be displayed after being uploaded and makes it clear that images can be uploaded to the system.

Stock Management Database Overview Page



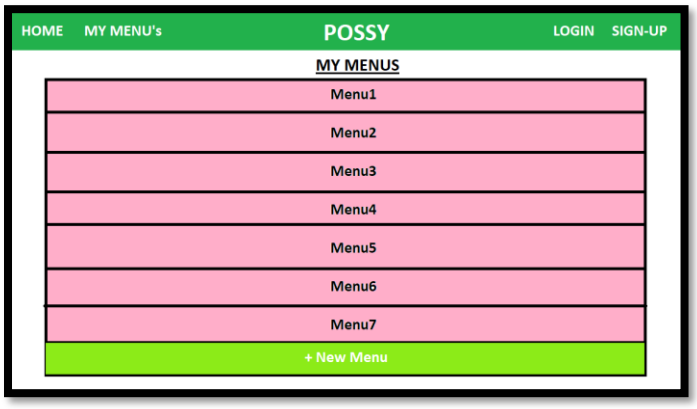
Relevant data from the stock management database table of the database is shown in a table to provide clarity to the system operator. This page is accessed by pressing the “Stock Management” button on the toolbar for easy access. To go back to the point-of-sale system, press the “POSSY” button. The headings are displayed in bold text to clearly label the data where the data is shown in a standard text to fit more pieces of text on the page. The edit and “+NEW” buttons are in grey as the colour was chosen in my research survey for reduced strain on the eyes. It is also in line with convention. I chose to use a button as buttons allow the system operator to be redirected to a new web page easily.

Stock Management Edit



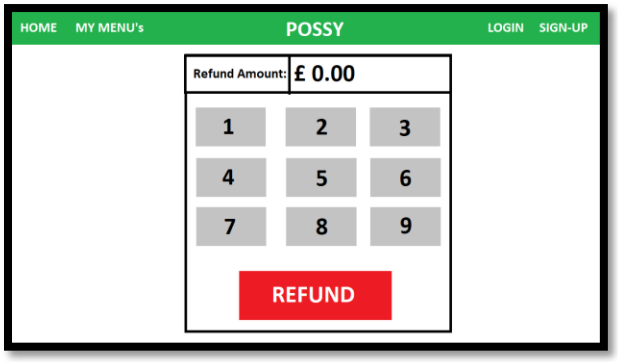
Text input boxes are used as they are easily clicked and visible, and it is clear that the intended purpose of each box is for data to be entered. The minimum and current stock quantity boxes are made only to accept numerical characters, preventing invalid data, e.g. alphabetic characters, from being entered. A drop-down menu was used for the “on order” and “date due” boxes. Again, this is to prevent invalid data from being entered and prevent invalid formatting.

My Menus



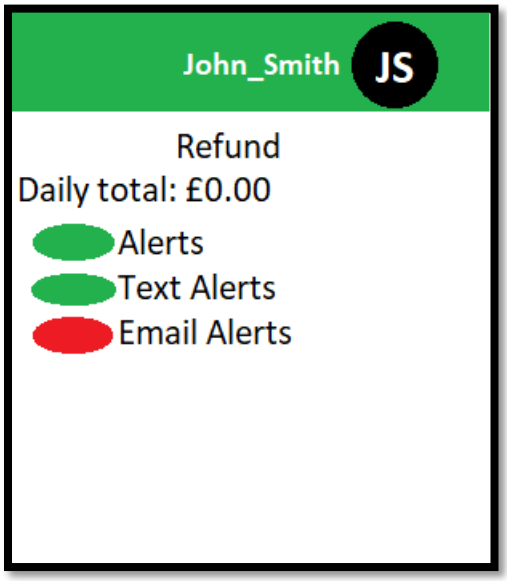
The menu buttons are large, making them easy to press, and they have a unique colour. Each button has a custom name making it easily identifiable. I chose to use buttons as it is easy to redirect users to other web pages using buttons.

Refund Page



The refund page is laid out like a calculator for familiarity and ease. All buttons are large so that they are easily pressed.

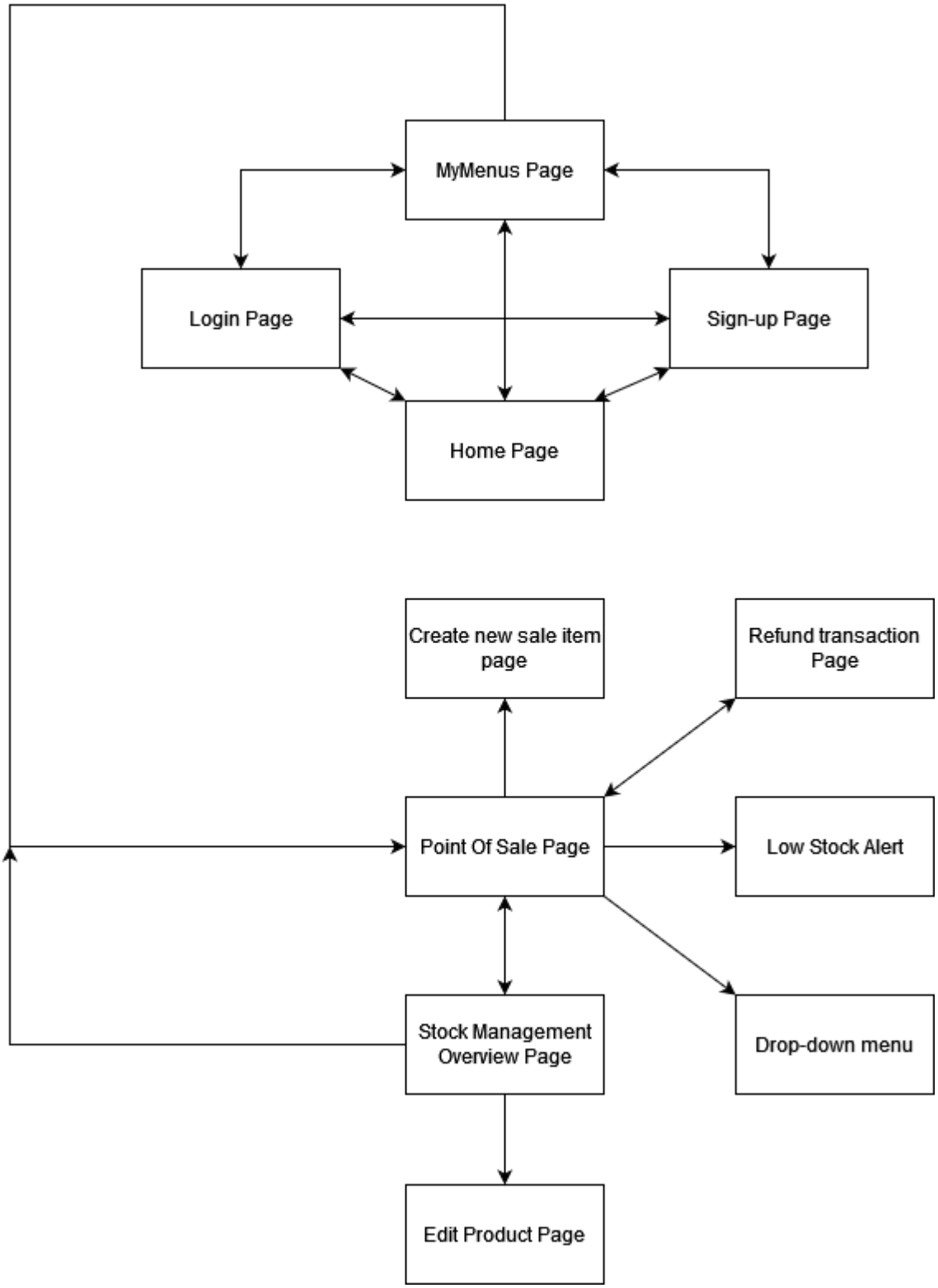
Drop Down Menu



Pressing on the profile picture icon opens this page. Pressing the coloured green/red buttons will toggle whether particular alerts are sent, red meaning off and green meaning on, as with conventions. Pressing the “Refund” button will redirect you to the refund page, where you can refund a transaction. The daily total will be automatically updated in this menu, allowing users to see how much money they have received today. All text in this menu is relatively small as it is designed to be hidden away and not very frequently used. It is likely that this menu will only be used one time daily; therefore, it has very little significance.

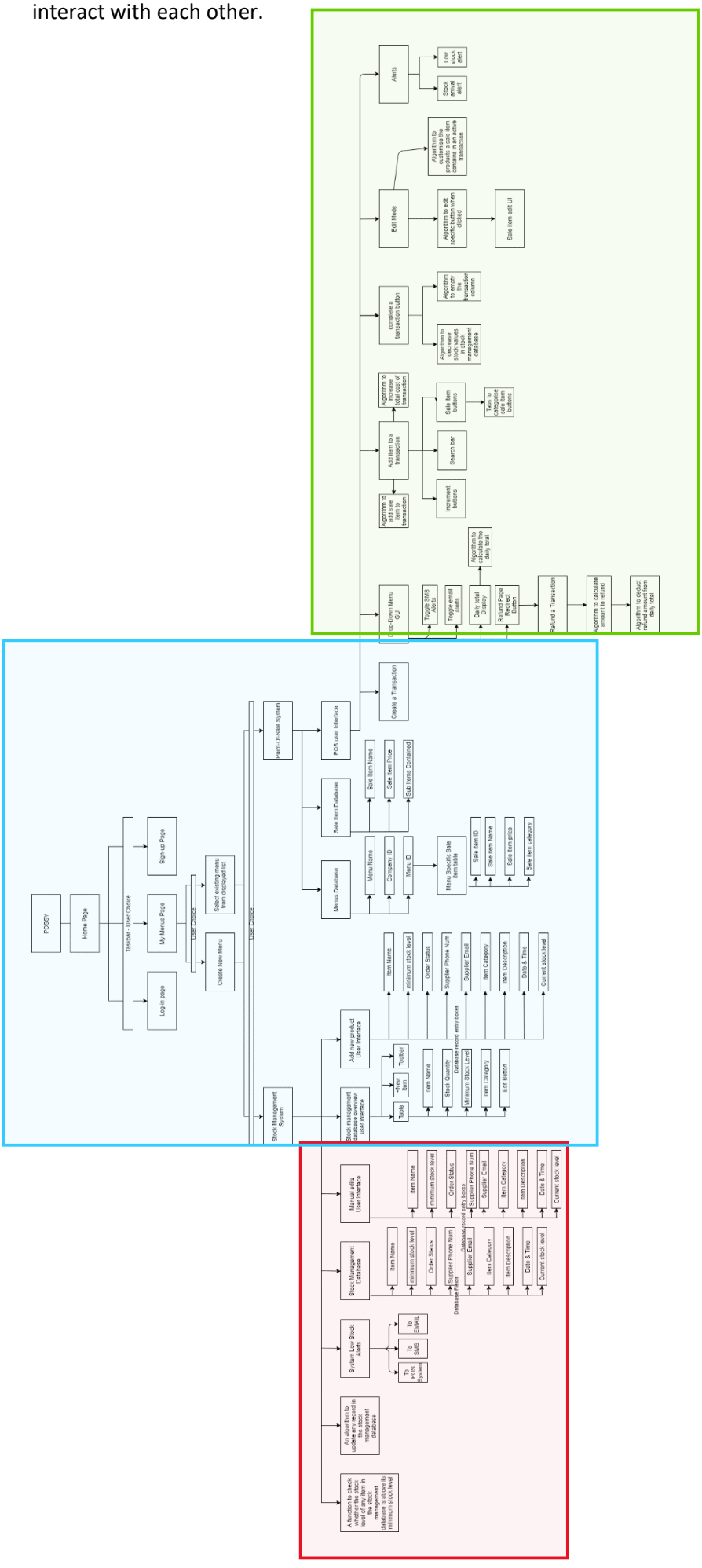
Web-Page Relationship Diagram

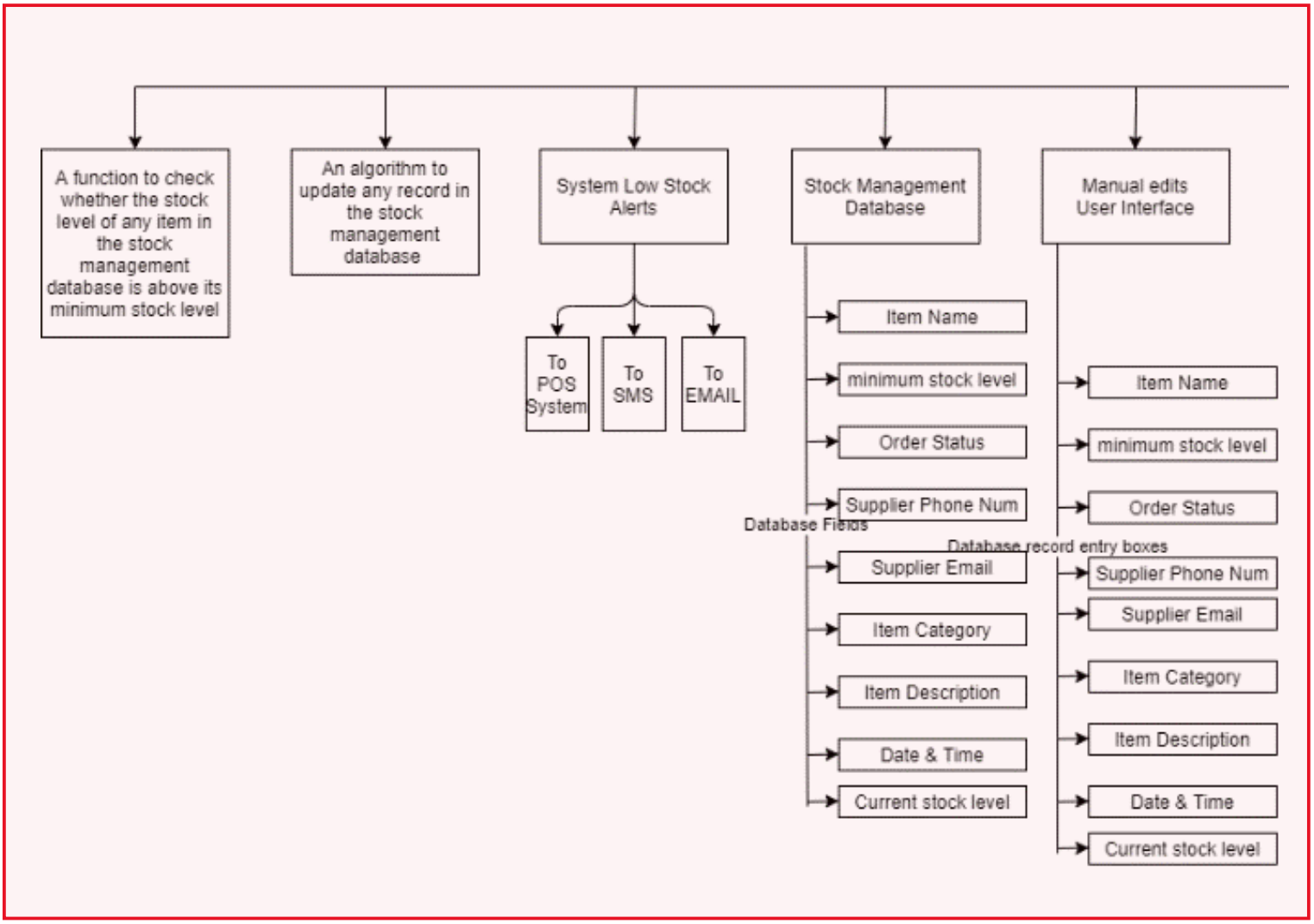
This diagram shows how web pages should be connected to each other, and the direction in which they should be accessed by each other. This will be particularly useful when creating the toolbar on each web page.

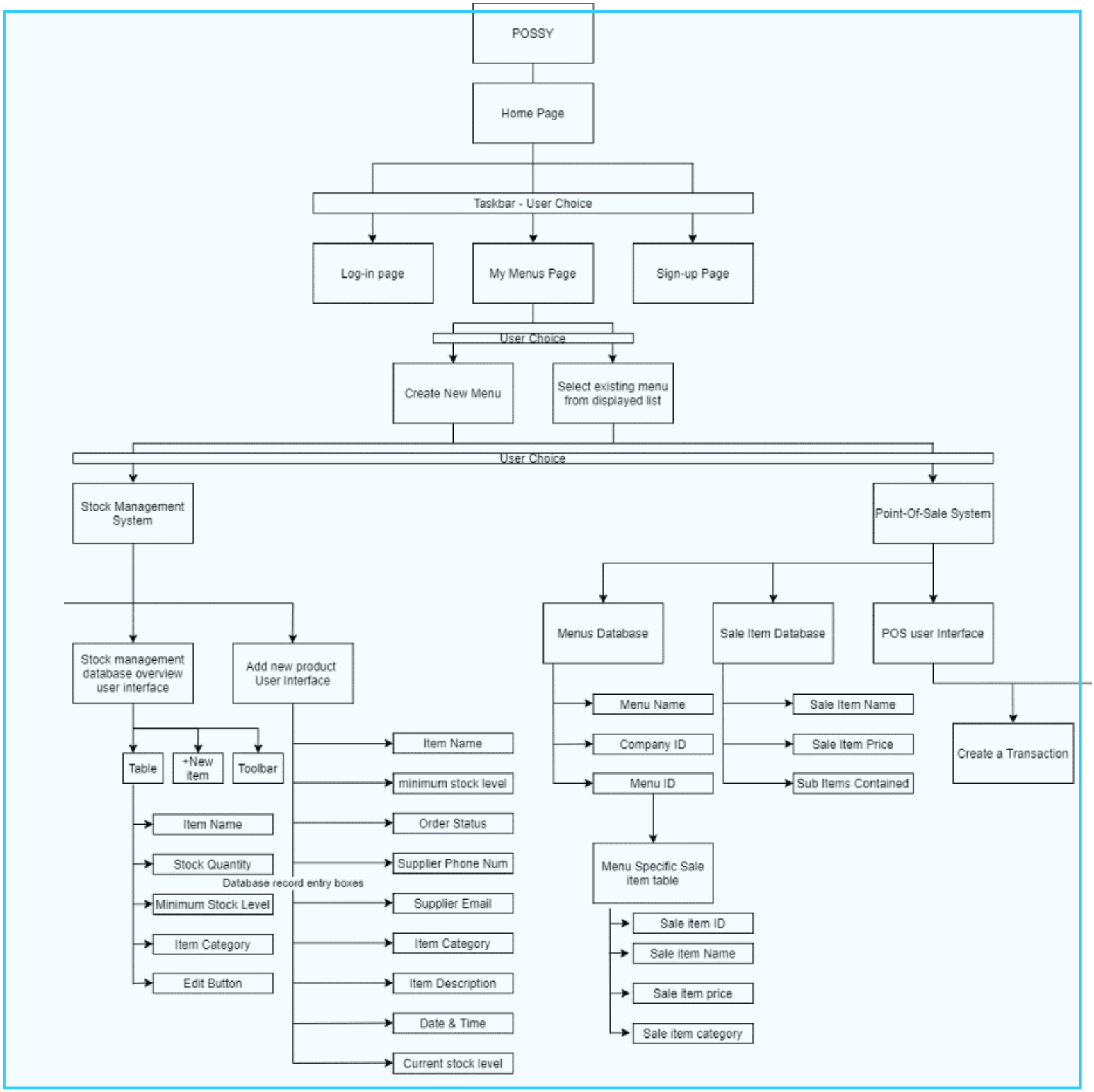


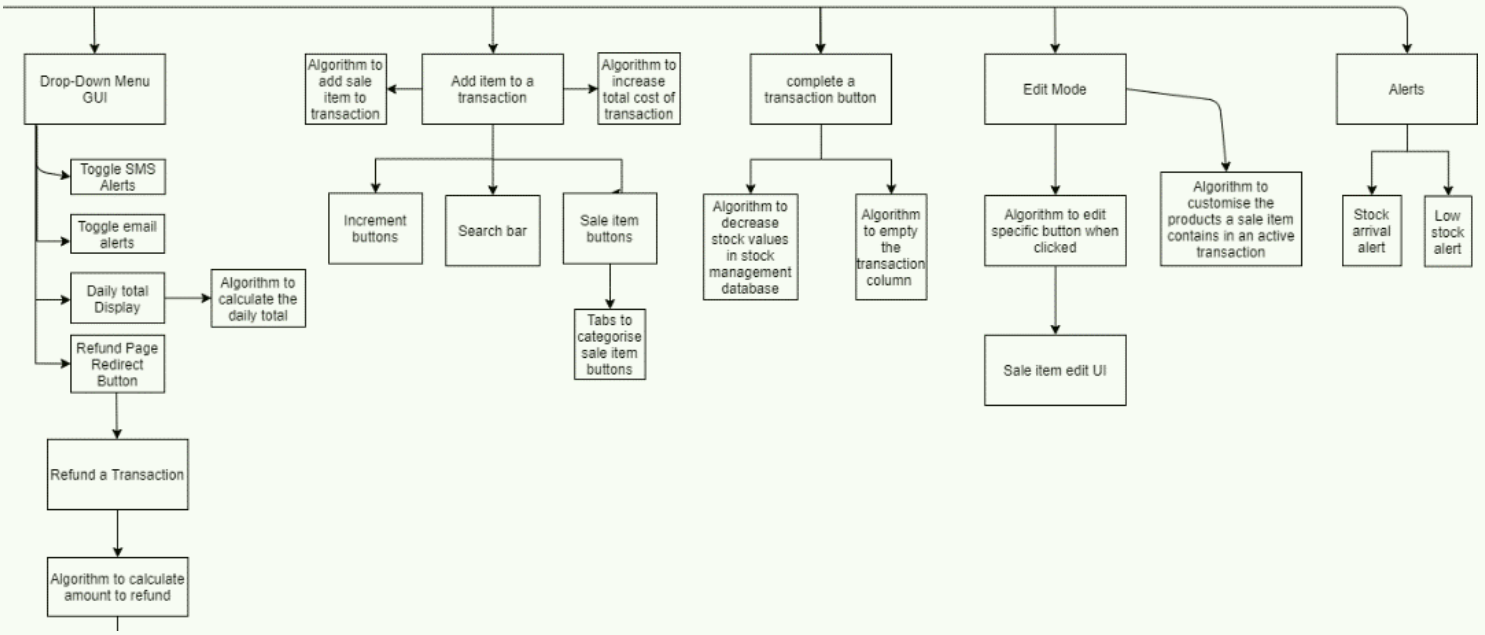
Structure Diagram

This diagram decomposes the task into sub-tasks, showing how the different parts of the solution connect and interact with each other.



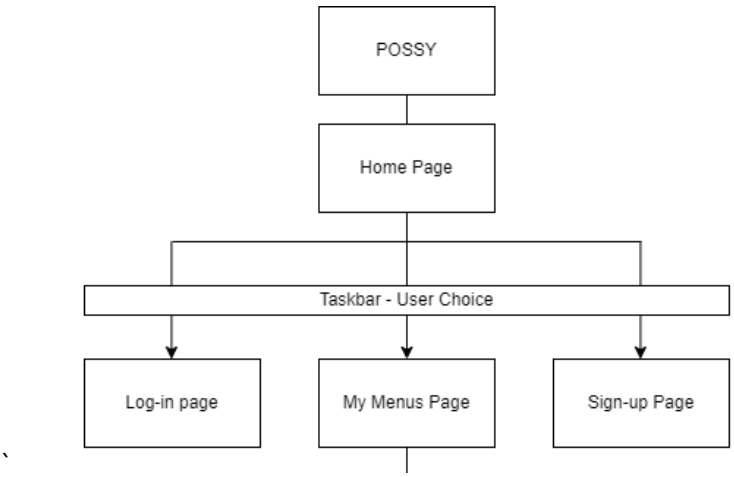

A function to check whether the stock level of any item in the stock management database is above its minimum stock level
An algorithm to update any record in the stock management database
System Low Stock Alerts
To POS System
To SMS
To EMAIL
Stock Management Database
Item Name
minimum stock level
Order Status
Supplier Phone Num
Supplier Email
Item Category
Item Description
Date & Time
Current stock level
Database Fields
Manual edits User Interface
Item Name
minimum stock level
Order Status
Supplier Phone Num
Supplier Email
Item Category
Item Description
Date & Time
Current stock level
Database record entry boxes


POSSY
Home Page
Taskbar - User Choice
Log-in page
My Menus Page
Sign-up Page
User Choice
Create New Menu
Select existing menu from displayed list
User Choice
Stock Management System
Point-Of-Sale System
Stock management database overview user interface
Add new product User Interface
Menus Database
Sale Item Database
POS user Interface
Table
+New item
Toolbar
Item Name
Stock Quantity
Minimum Stock Level
Item Category
Edit Button
Database record entry boxes
Item Name
minimum stock level
Order Status
Supplier Phone Num
Supplier Email
Item Category
Item Description
Date & Time
Current stock level
Menu Name
Company ID
Menu ID
Menu Specific Sale item table
Sale item ID
Sale item Name
Sale item price
Sale item category
Sale Item Name
Sale Item Price
Sub Items Contained
Create a Transaction


Drop-Down Menu GUI
Toggle SMS Alerts
Toggle email alerts
Daily total Display
Algorithm to calculate the daily total
Refund Page Redirect Button
Refund a Transaction
Algorithm to calculate amount to refund
Algorithm to add sale item to transaction
Add item to a transaction
Algorithm to increase total cost of transaction
Increment buttons
Search bar
Sale item buttons
Tabs to categorise sale item buttons
complete a transaction button
Algorithm to decrease stock values in stock management database
Algorithm to empty the transaction column
Edit Mode
Algorithm to edit specific button when clicked
Sale item edit UI
Algorithm to customise the products a sale item contains in an active transaction
Alerts
Stock arrival alert
Low stock alert

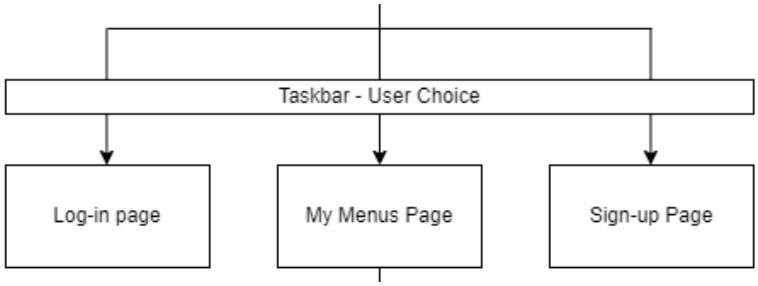
Breaking Structure Diagram into Stages

Stage 1 – The home page. The home page and the task bar linking to the login, my menu and sign-up pages will be created during this stage. These three pages the task bar links to will be created in a later stage.

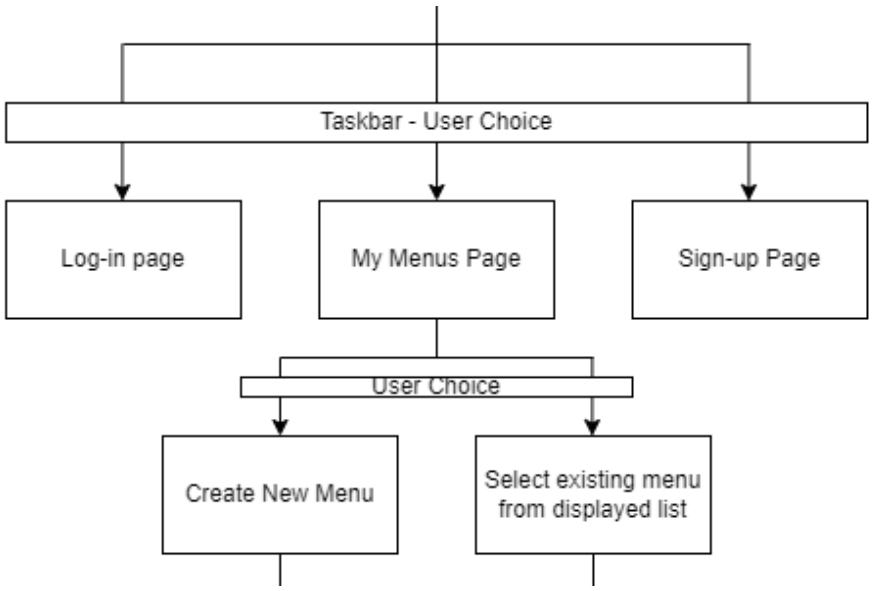


`

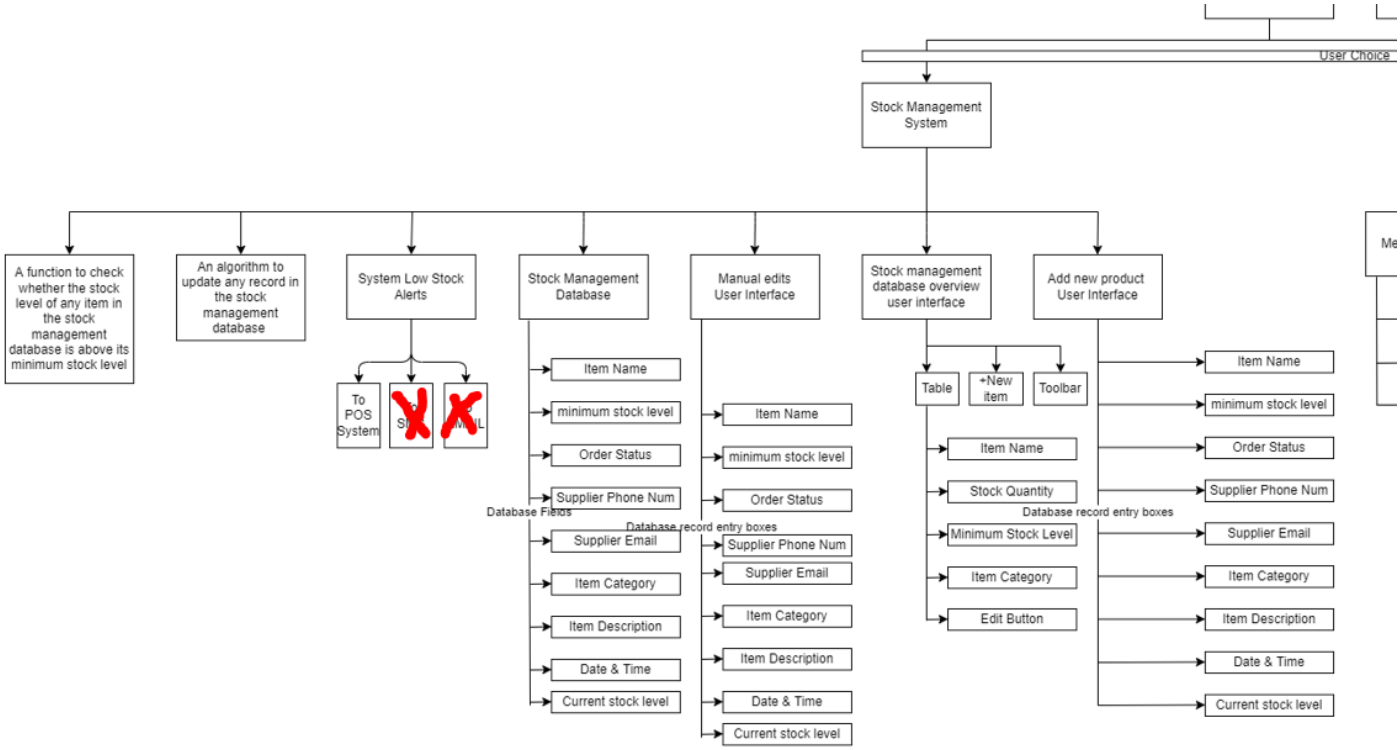
Stage 2 – The login and signup pages. These two stages will allow new users to create a new account, and to access an existing account. Having an account is required before the user can use the main system. The home page will be created at a later stage



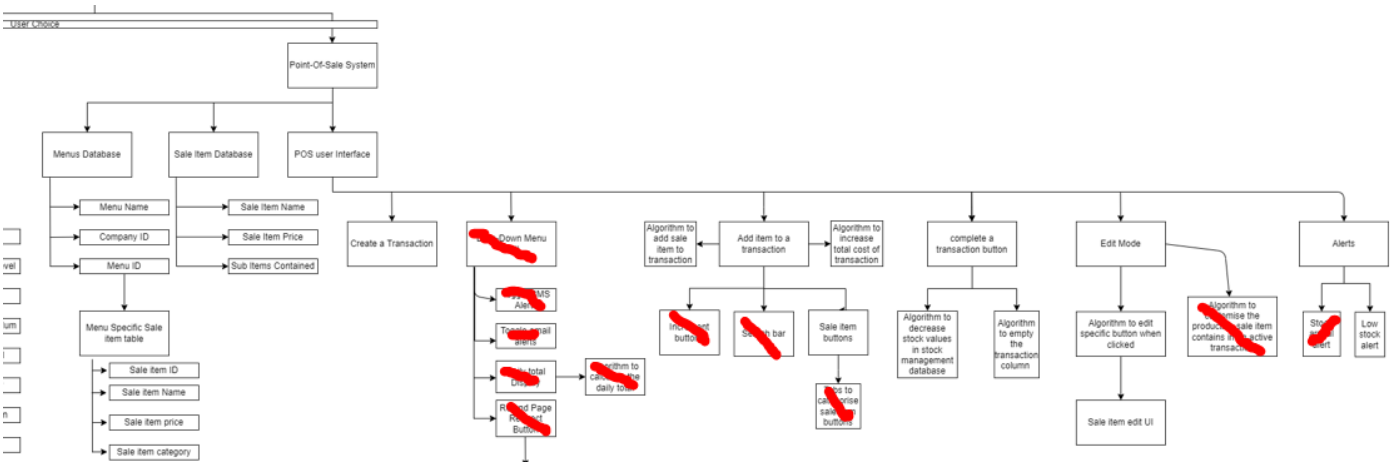
Stage 3 – The my menus page. This stage is the begging of the main system. Here users will be able to create a new menu or open a menu using a name that they have already created. (The login and sign up pages will have been created in the previous stage)



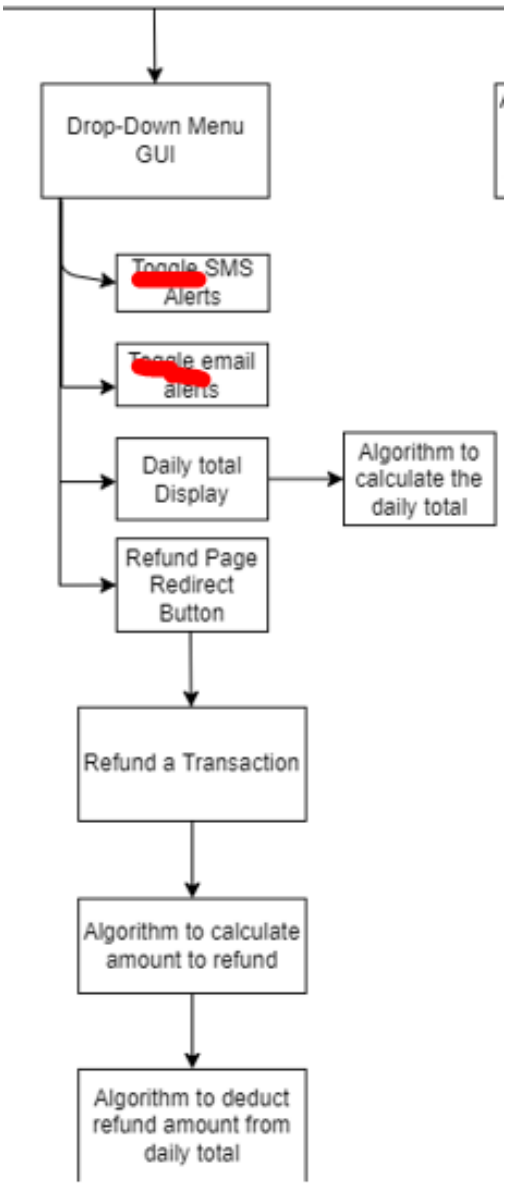
Stage 4 – Stock management system essentials. This stage is the beginning of the stock management system. It aims at making this feature functional, while including the vast majority of the success criteria. The main feature not included here is low stock alerts via SMS and email.



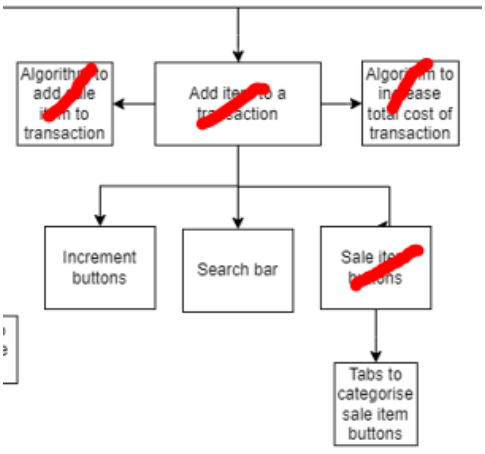
Stage 5 – Point of sale system essentials. This stage is the beginning of the point-of-sale system. It aims at making this feature functional, whilst including most of the success criteria.



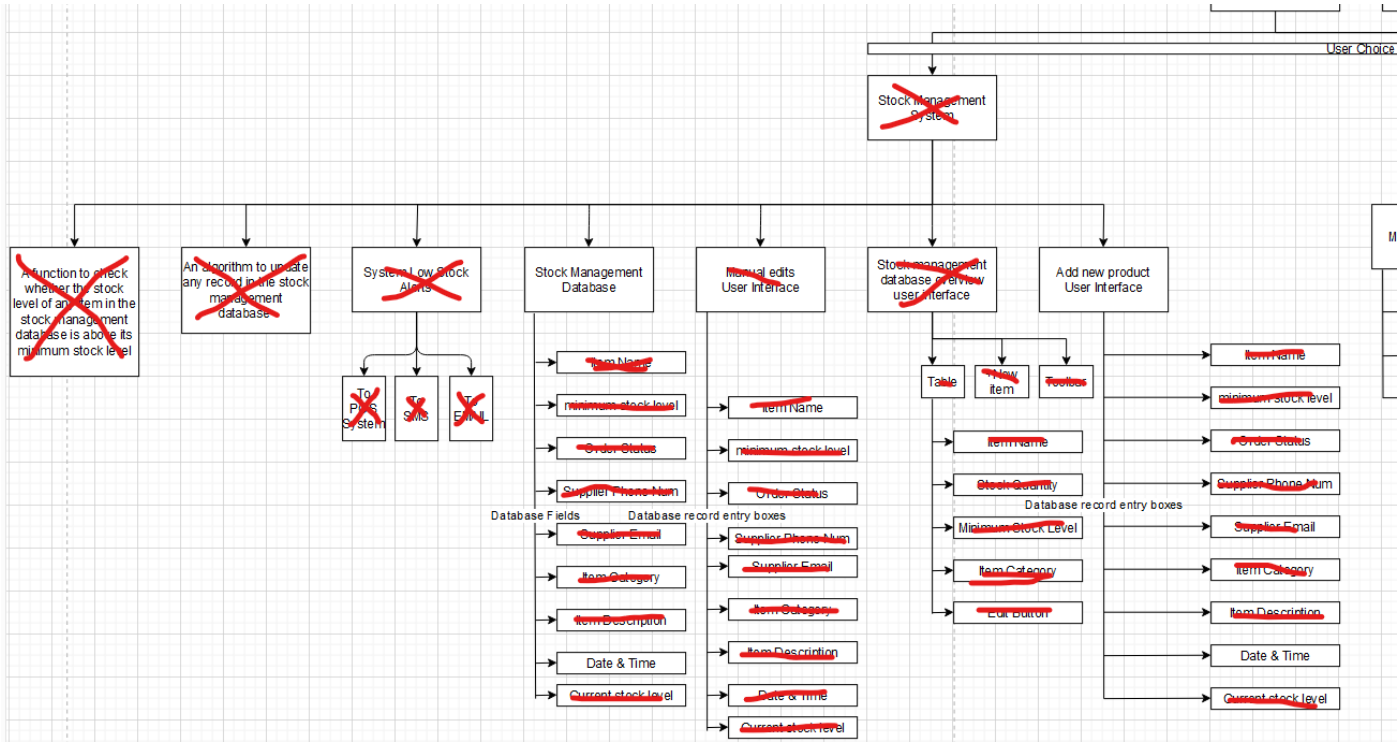
Stage 6 – dropdown menu and refunds. This stage aims to introduce the ability to refund a transaction. This requires a drop-down menu to be created, which will later contain other settings and features.

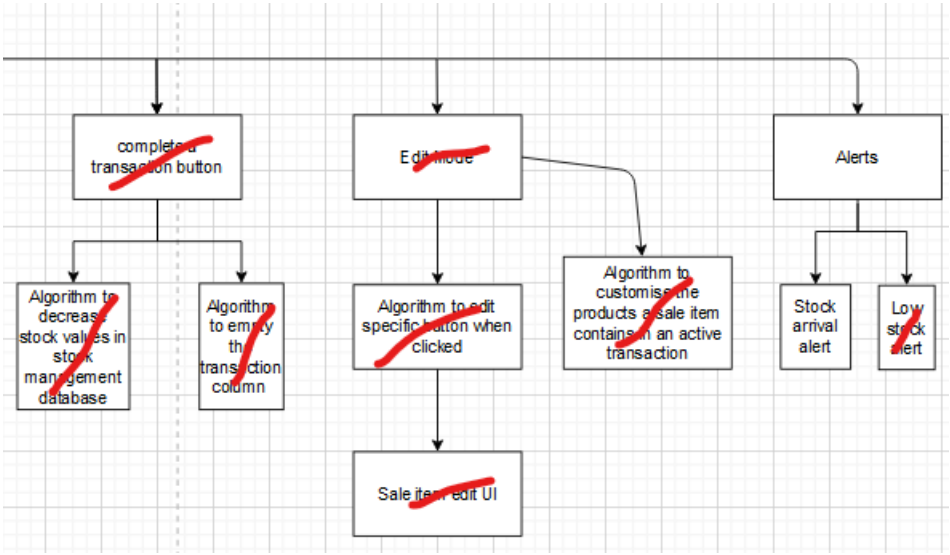


Stage 7 – Point of sale system desirables. This stage will aim to implement the desirable features of the point-of-sale system. Many of these features will not be in the success criteria but are contained within the user requirements.

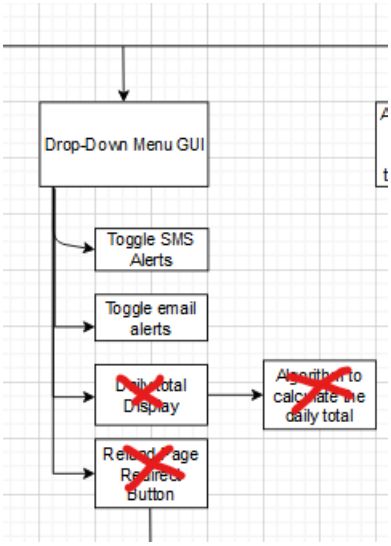


Stage 8 – Stock management system desirables. This stage will aim to implement the desirable features of the stock management system. Many of these features will not be in the success criteria but are contained within the user requirements.





Stage 9 – User quality of life. This stage aims to add any final features to the project. Many of these features do not matter if they are not implemented, but would be very good to widen my target audience and give my project a more professional feel.



Data Dictionary

Variable Name	Variable Type/ Format	Allowed Values	Synonyms	Description
$_SESSION [companyMenuTableName']	Global String			Stores the name of the MySQLi table for the currently open menu
$_SESSION['companyName']	Global string		$companyName	Stores the company name that the currently logged-in user is assigned to
$_SESSION['email']	Global string | false		$email	Stores the email of the currently logged in user
$_SESSION['menuID']	Global mixed			Stores the menuID of the currently open menu
$_SESSION['password']	Global string		$password	Stores the password of the currently logged-in user
$_SESSION['productID']	Global mixed		$productID	Stores the ID of a product to be passed between files
$_SESSION['returnedRows']	Global mysqli_result			Stores the mysqli_result return object for all products that are below their minimum stock values
$_SESSION['shouldSearch']	Global boolean	True/false		Stores whether or not the search function is active within the point-of-sale system
$_SESSION['total']	Global mixed		$total $transactionTotal	Stores the total cost of the current transaction
$_SESSION['TransactionID']	Global mixed	0 - System maximum	$transactionID $transactionNumber	Stores the ID of the currently open transaction
$_SESSION['username']	Global string		$username	Stores the username of the currently logged-in user
$_SESSION['phoneNumber']	Global mixed | false	00000000001 - 99999999999	$phoneNumber	Stores the phone number of the currently logged-in user
$_SESSION['menuQuery']	global string			Stores the query needed to only display sale item buttons with particular names in the point-of-sale system
$arrivalAmount	Int	0 – system maximum		Stores the amount of stock due to arrive during a re-stock
$arrivalDate	String | null	YYYY/MM/DD HH:MM:SS		Stores the date & time retrieved from a MySQLi database table
$category	String			Stores the category name of a sale item
$connection				Stores the connection to the database
$count	Int	System minimum – system maximum	$i	A count variable
$currentDateShort	string	YYYY/MM/DD HH:MM		Stores the current date and time (GMT)
$currentStockValue	Int	0 – system maximum		Stores the value of the current stock amount of a product
$dateTime	string	YYYY/MM/DD HH:MM		Stores the date & time from a HTML form
$minimumStockValue	Int	0 – system maximum	$minimumStock	Stores the value of the minimum stock value of a product
$numRows	Int	0 – system maximum	$numResults	Stores the number of rows returned by a MySQLi query

$price	String	0.00 – system maximum	$itemPrice	Stores the price of a sale item
$productArray	String array	no commas and no whitespace at the beginning and end of an array element		Stores the name of all products contained within a sale item, which can be later used to enter the contents of a sale item into a database table
$productName	string			Stores the name of a product
$query	string			Forms the query that should be passed into the mysqli_query() function to run the given query in the database
$redirectURL	string			Calculates the needed URL to redirect the user to the hasStockArrived.php page when new stock is due to have arrived
$result	mysqli_result		$saleItemContents Result $productQueryResult $currentStockValue QueryResult $transactionHistoty Result $categoryResult	Stores the object returned by a MySQLi query, returns false if no object is returned
$row	Array		$saleItemContents Row $currentStockValue QueryRow $rowID $rowResult $tableRow $categoryRow	Stores a single row of a mysqli query result
$saleItemID	String	0 – System maximum	$id	Stores the ID of a sale item
$saleItemName	String			Stores the name of a sale item
$searchValue	string			Stores the value contained within the search bar inside the point-of-sale system
$tableName	String		$stockManagenent TableName $saleItemContents TableName $numRowsReturned ByQuery $transactionHistory Table	Calculates and stores the name of a MySQLi table to be passed into a MySQLi query
$tabName	String			Stores the name of a category to be displayed as the name of a tab on the right-hand side of the point-of-sale system GUI

Development Plan

Stage 1 – The home page (3hours)

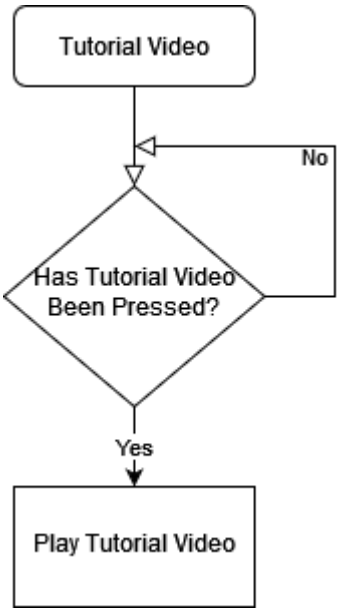
The home page allows access to all other pages on the website. Therefore, before any other web page can be designed, the home page user interface must be designed, and the page must be functional.

Features/functionality:

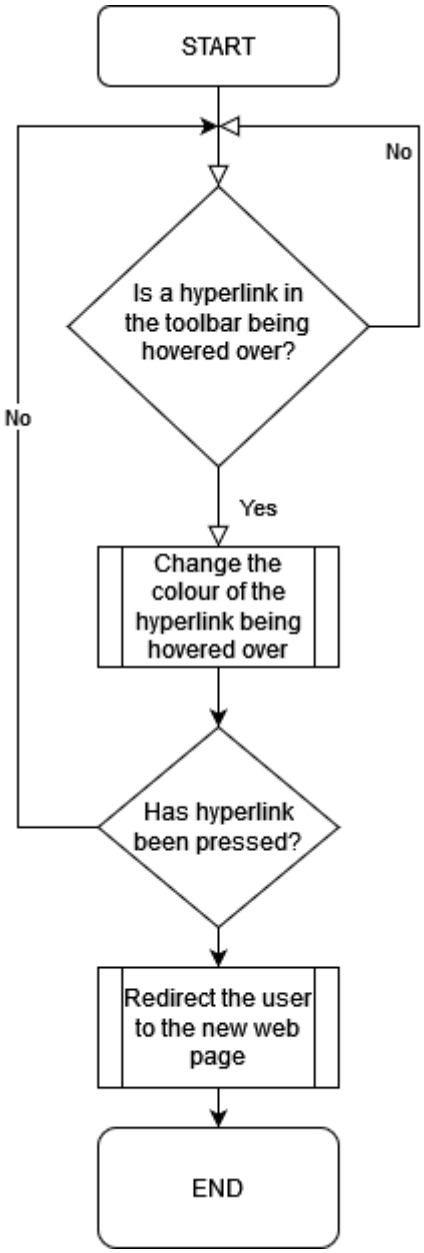
1. A toolbar containing functional redirect buttons to the “log-in”, “signup”, and “My Menus” pages
2. A space in which an embedded tutorial video can be placed upon completion of the project

Stakeholders should be consulted to confirm that the colour of the toolbar meets their needs and that they are content with the overall layout of the page, as similar layouts will be in use throughout the project. If the stakeholders dislike the colour or page layout, they should be changed accordingly.

Test No.	Description	Test Data	Expected Result	Actual Result
1	Does the web page generate correctly sized?	Open website	Web page opens and is sized correctly	
2a	Does the log-in button redirect you to the log-in page? (Placeholder page at this stage)	Click log-in button	Redirected to the log-in page	
2b	Does the signup button redirect you to the signup page? (Placeholder page at this stage)	Click signup button	Redirected to the signup page	
2c	Does the MyMenus button redirect you to the MyMenus page? (Placeholder page at this stage)	Click MyMenus button	Redirected to the MyMenus page	
3a	Does the embedded video play on the web page?	Click video play button	Video plays	



This shows how the web page should respond to the tutorial video being pressed. Once the tutorial video has been pressed, it should play. Otherwise the system should keep checking whether the video has been pressed. This ensusres that the user will only need to press the video once before it begins to play.



This shows how the toolbar should react to a user hovering over one of its hyperlinks, and what should be done if a hyperlink has been pressed. This is constantly looped making sure that the hyperlink being hovered over changes colour when the cursor is over it, but returns to its original colour when not being hovered over. Once a hyperlink has been pressed, the user is redirected to the web page that the hyperlink leads to.

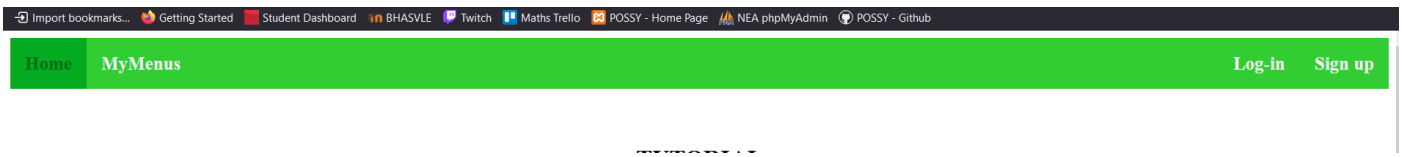
Creating a toolbar

The first stage is to create a toolbar. That contains buttons, making links to other pages in the project. This is a simple task that I have prior experience with; therefore, I expect to create this as intended with very few errors. The first step includes creating a set of hyperlinks in HTML, where the majority align from left to right. This excludes the login and signup pages, which will align from right to left.

```html
<!-- toolbar -->
<div class="toolbar">
    <a class="current" href="HomePage.php">Home</a>
    <a href="MyMenus.php">MyMenus</a>
    <a style="float: right;" href="Signup.php">Sign up</a>
    <a style="float: right;" href="Login.php">Log-in</a>
</div>
```

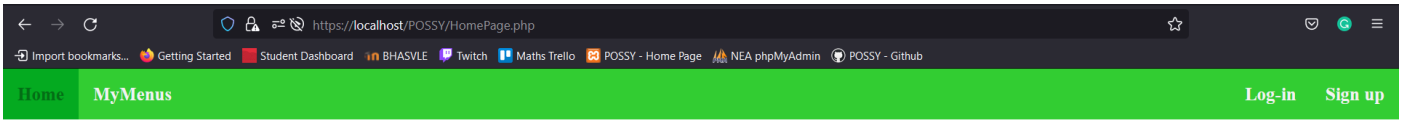
```css
/*TOOLBAR*/
/* Add green background to the toolbar */
.toolbar {
    background-color: limegreen;
    overflow: hidden;
}

/* Style the links in the toolbar */
.toolbar a {
    float: left; /* links move from the left to right. Some links have inline styling to change this*/
    text-decoration: none; /* removes the underline from the hyperlink */
    text-align: center;
    color: #f2f2f2; /* change the colour of text on inactive web pages */
    font-size: 20px;
    font-weight: bold;
    padding: 16px 16px; /* increase the height of the colour above and below the text */
}

/* Add a color to the current web page title */
.toolbar a.current {
    background-color: #04aa20;
    color: white;
}

/* Change the colour of links when hovered over */
.toolbar a:hover {
    background-color: #00ba00;
    color: black;
}
```

This is good, but I am not too fond of the black colour of the text when presenting the active window. I will change this colour to grey instead.



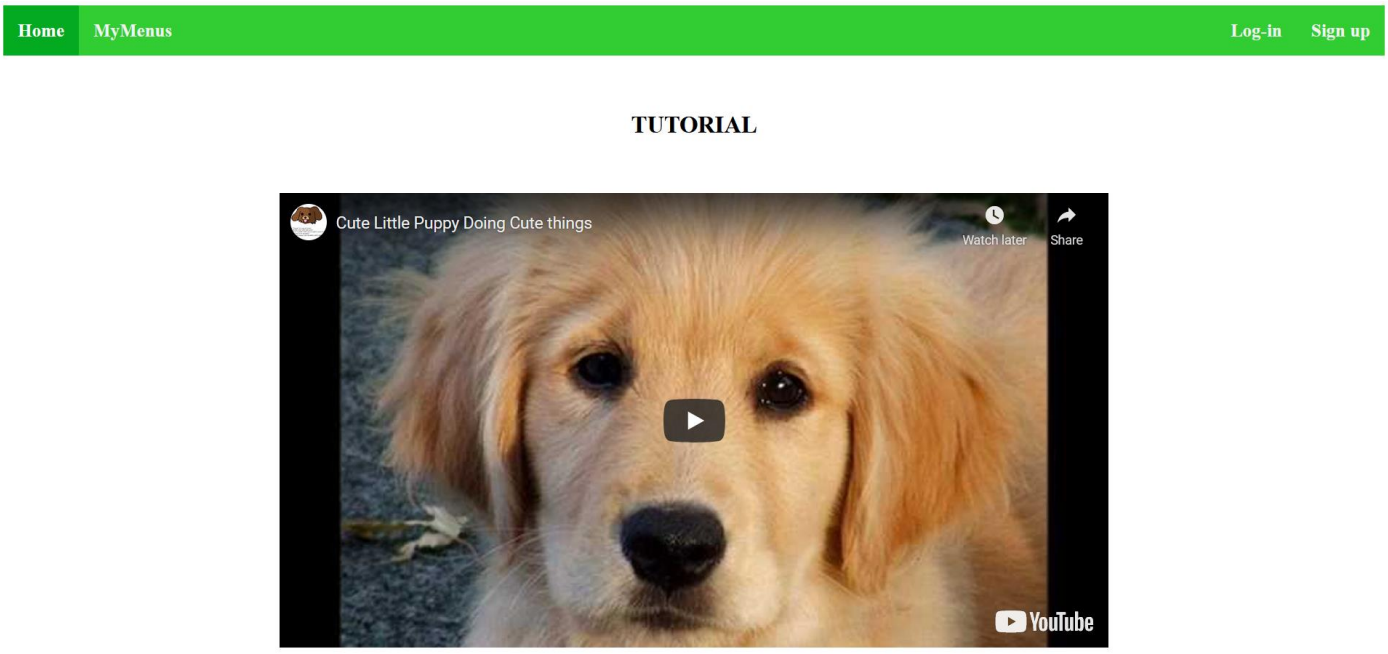
I prefer this grey colour as it is less prominent. However, I do not like that the gap above, to the left, and the right of the toolbar is white, it should be green. I have removed the margins from the HTML <body> tag using CSS to fix this.

```css
/*TOOLBAR*/
body{
    margin: 0px;
}
```



Including an embedded video

This feature is quite simple – I need to create a rectangular box containing a placement video from YouTube until the tutorial video is created at the end of the project.




```html
<!-- embed a youtube video into the centre of the HomePage -->
<div id="embeddedVideo">
    <iframe width="60%" height="500" allowfullscreen
            src="https://www.youtube.com/embed/j5a0jTc9S10?rel=0&amp;controls=0&amp;showinfo=0&amp;autoplay=1">
    </iframe>
</div>
```

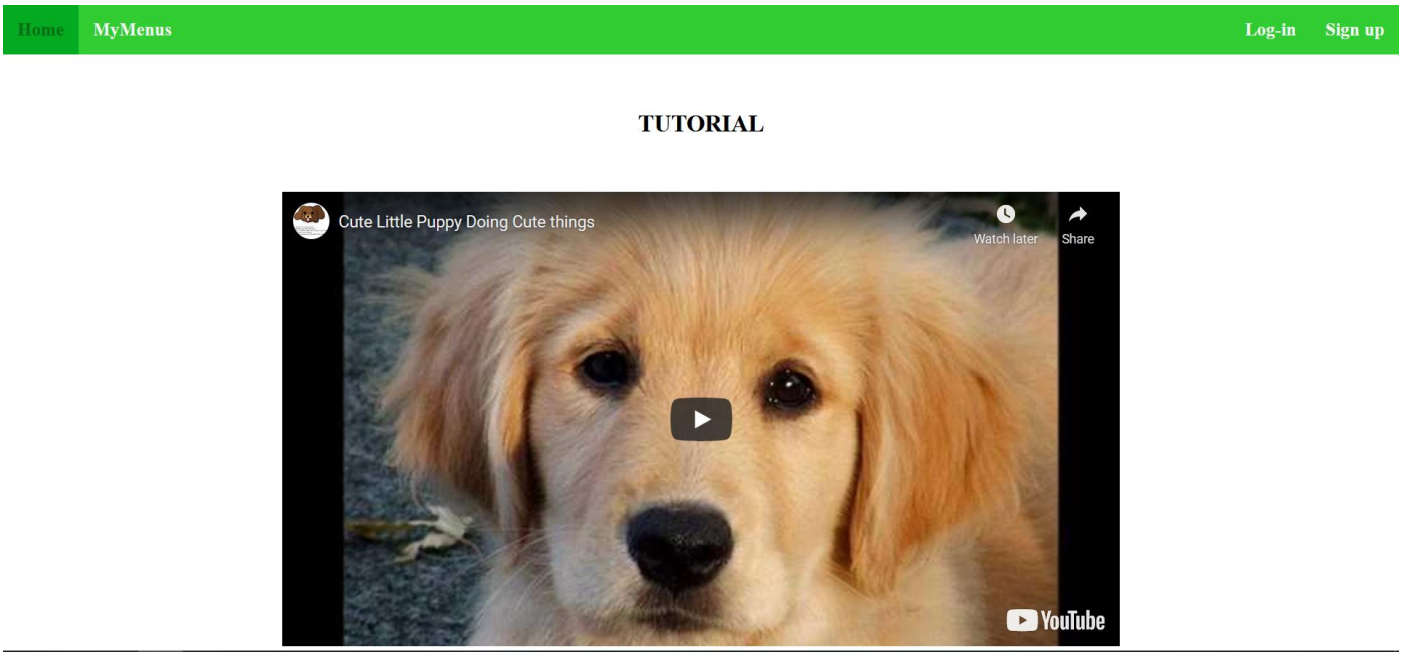
```css
/* EMBEDDED VIDEO */
#embeddedVideo{
    text-align: center;
}
```

I planned for the tutorial video to play automatically once the home page was opened. This is not working as shown in **'Toolbar Video 1.mp4'**. Therefore, this feature will no longer be included in my project.

```
<!-- embed a youtube video into the centre of the HomePage -->
<div id="embeddedVideo">
    <iframe width="60%" height="500" allowfullscreen
            src="https://www.youtube.com/embed/j5a0jTc9S10?rel=0&amp;controls=0&amp;showinfo=0&amp;autoplay=1">
    </iframe>
</div>
```

'**Toolbar Video 1.mp4'** also shows a black border around the video when the page begins to load. This feature was not intentional. To fix this, I will remove the border using CSS.

Here is the final design of the Home Page at this stage. Its features include a working toolbar and embedded video, as shown in **'Home Page Showcase.mp4'**



End-Of-Stage Testing

Test No.	Description	Test Data	Expected Result	Actual Result
1	Does the web page generate correctly sized?	Open website	The web page opens and is correctly sized	The web page opens and is sized correctly. See **Home Page Showcase.mp4**
2a	Does the log-in button redirect you to the log-in page? (Placeholder page at this stage)	Click log-in button	Redirected to the log-in page	Redirected to the log-in page. See **Home Page Showcase.mp4**
2b	Does the signup button redirect you to the signup page? (Placeholder page at this stage)	Click signup button	Redirected to the signup page	Redirected to the signup page. See **Home Page Showcase.mp4'**
2c	Does the MyMenus button redirect you to the MyMenus page? (Placeholder page at this stage)	Click MyMenus button	Redirected to the MyMenus page	Redirected to the MyMenus page. See **Home Page Showcase.mp4'**
3a	Does the embedded video play on the web page?	Click the video play button	Video plays	Video plays. See **Home Page Showcase.mp4'**

Feedback from stakeholders

My stakeholders were consulted to confirm that the colour of the toolbar meets their needs. They are content with the page's overall layout and are happy for similar layouts and colours to be used throughout the project. The green colour of the taskbar was particularly liked, so it will be used as a theme colour throughout the project where possible.

Stage Review

I was pleased with this stage of development, it was easy to complete as I had prior experience in creating toolbars, and the embedded video required minimal programming. From this stage in the project, I have met the following bullet points from my success criteria:

- A main menu as described in requirements 1.1
    - Links to log-in, sign-up and 'My Menu' pages

I completed all tasks at this stage, and I do not believe there are any tasks that I need to return to.

Stage 2 – Log-in and Signup Pages (6hours)

For users to make use of the available features, they must first log in to a registered account. This will involve them signing up and storing their details in a database. They may then login using the email & password that they created. The entered credentials will be compared to those in the Users database and, if they match, the user will be allowed access to that account.

Features/functionality:

1. A database that stores the data for all registered users in a credentials table
2. A signup user interface with text input fields allowing new users to sign up using a username, password, company name, email& phone number
3. A log-in user interface with text input fields allowing registered users to log in using their username and password created during signup
4. An algorithm to add data input into signup fields into the Users database
5. An algorithm to validate email addresses
6. An algorithm to check whether a username is currently in use
7. An algorithm to validate a user's log-in credentials

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Does the log-in page generate correctly sized	Open log-in page	The web page opens and is sized correctly	
1b	Does the signup page generate correctly sized	Open signup page	The web page opens and is sized correctly	
3a	After signing up, does all entered data get saved in the Users database?	Username Password Email@email.com 12345678910 Company	Username Password Email@email.com 12345678910 Company Are visible in a record of the Users database	
2a	Attempt to Input alphabetic characters into signup phone field	abcde123456	123456	
2b	Attempt to signup using an invalid email address	Username2 Password email@email 01234567890 Company	Error – Invalid Email	
2c	Attempt to signup using a username already in use	Username Password email@new.com 01234567890	Error – Username Already In Use	
3a	Attempt to log in using an incorrect password	Username NotPassword	Error – invalid credentials	
3b	Attempt to log in using an incorrect username	NotUsername Password	Error – credentials	

SQL Statements

SELECT Username FROM Credentials WHERE Username = '(variable containing the user's entered username)'

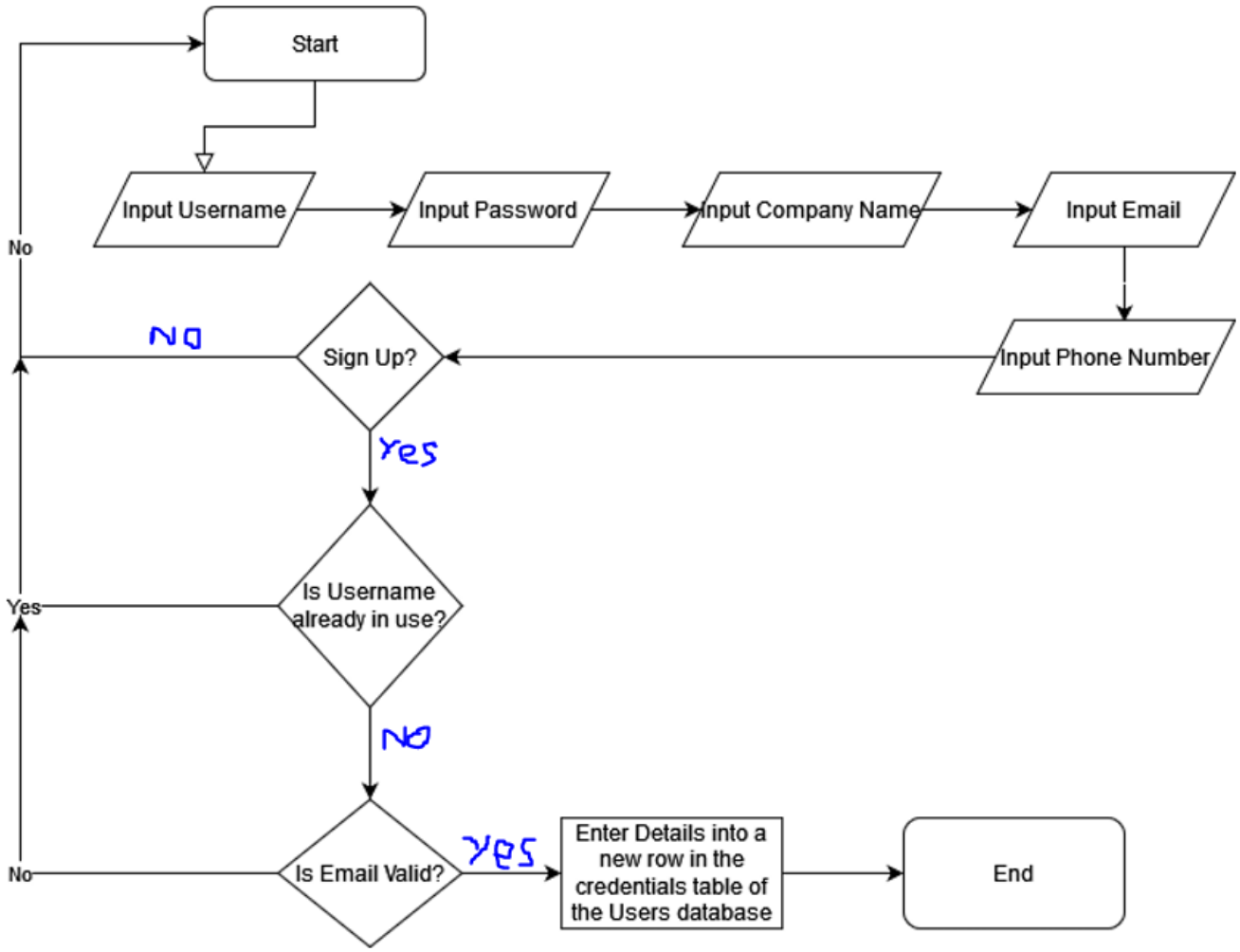
INSERT INTO Credentials(Username,Password,Company,Email,Phone) VALUES([user's entered username], [user's entered password, [user's entered company name], [user's entered email], [user's entered phone number])

SELECT Company, Email, Phone FROM Credentials WHERE Email='[user's entered email name]', Company = '[User's entered company name]', Phone = '[User's entered phone number]'

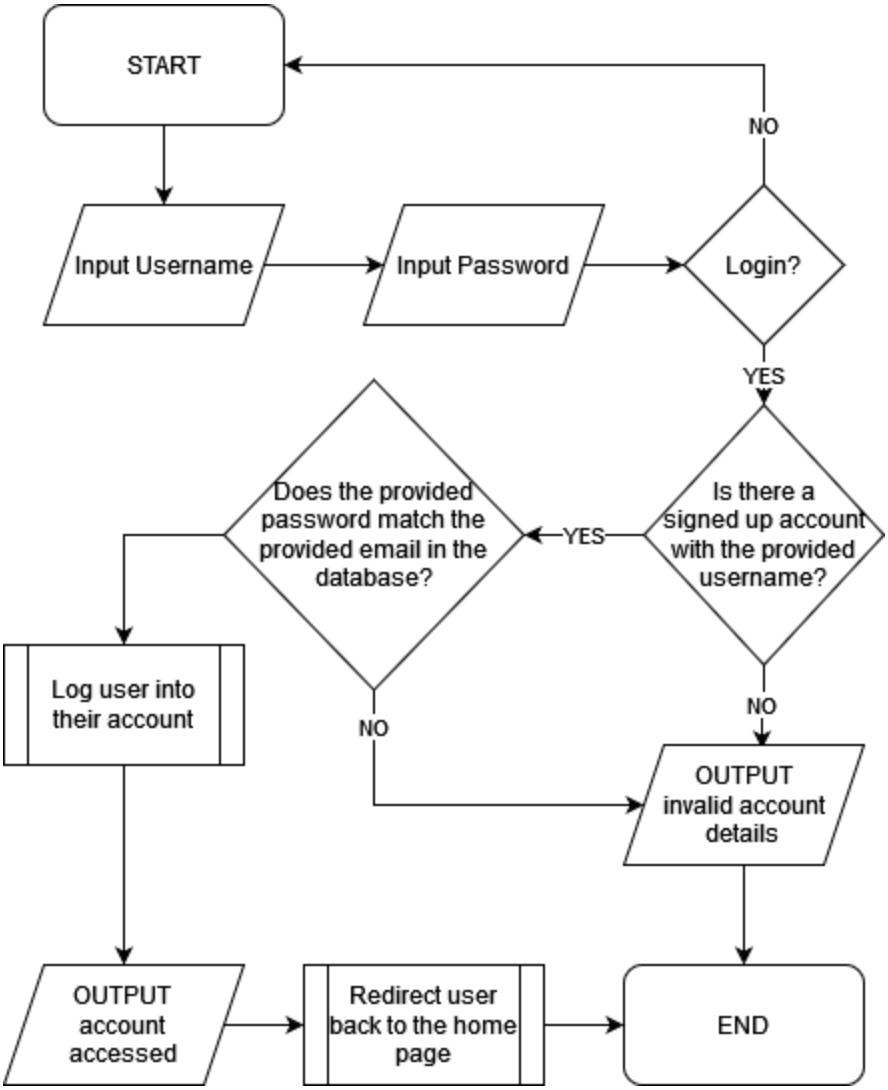
Database table design for this stage:

Credentials		
Username	text	
Password	text	
Company	text	
Email	text	
Phone	integer(11)	
Add field		

Flow Charts



This shows how the sign-up page should work, and the different conditions needed to validate a new account. A user must fifrst enter a username, password, company name, email, and phone number into an input box. They must then choose whether they would like to sign up or not by pressing a button. If yes, the system check if the username is already in use, otherwise the user will need to restart the process. If the user's username is already in use, the signup process will need to be restarted. Otherwise, the email is checked for validity. If the email is valid the new credentials are entered into a row in the database table. Otherwise, the sign up process is restarted.



This shows how the log-in page works. It shows the necissary checks to validate that a user can be granted access to their account, and provides friendly messages to the user dependant on the outcome. The user will first input their username and password into the input box, and the user is asked whether they would like to log in. If they would llike to log in, the system checks whether there is an account with the provided username and password. If there is, the user is logged into their account and the user is informed of this then redirected to the home page. If either of their entered details do not match those in the database, they are told their account details are invalid.

The Toolbar

The toolbar is the same toolbar that is used on the Home Page; therefore, the HTML was copied from there. However, the current page was set as class='current' rather than the Home Page.

```
<!-- toolbar -->
<div class="toolbar">
    <a href="HomePage.php">Home</a>
    <a href="MyMenus.php">MyMenus</a>
    <a class="current" style="float: right;" href="Signup.php">Sign up</a>
    <a style="float: right;" href="Login.php">Log-in</a>
</div>
```

Creating the credentials database table

CREATE TABLE 'Users', 'credentials' ('Username' TEXT NOT NULL, 'Password' TEXT NOT NULL, 'Company'' TEXT NOT NULL, 'Email' TEXT NOT NULL, 'Phone' INT(11) NOT NULL, PRIMARY KEY ('Username'))

I input the above query into the database, creating a credentials table. The resulting table is shown below:

	Username	Password	Company	Email	Phone
Edit Copy Delete	newUser	password	newCompany	firstnamelastname@gmail.com	1234
Edit Copy Delete	user	pass	company	emails@email.com	2147483647
Edit Copy Delete	username	pass	company	emails@email.com	1
Edit Copy Delete	username1	password1	companyName1	emails@email.com	2147483647
Edit Copy Delete	username10	password	myCompany	company@email.com	1234567891
Edit Copy Delete	username100	password	myCompany	company@email.com	1234567891

The signup form

The first step is creating the sign-up form using <form> tags. <input> tags will then be used for the user to input data. When data is input, and the submit button is pressed, the inputs are posted to the process_signup.php file.


```php
<form action="process_signup.php"> <!-- send entered data into process_signup.php for processing -->
    <div class ="signup_container">

        <h1>Sign up</h1>

        <label for="username">Username</label>
        <input type="text" name="username" placeholder="Enter Your Username Here" required>

        <br>

        <label for="password">Password</label>
        <input type="text" name="password" placeholder="Enter Your Password Here" required>

        <br>

        <label for="companyName">Company Name</label>
        <input type="text" name="companyName" placeholder="Enter Your Company Name Here" required>

        <br>

        <label for="email">Email</label>
        <input type="text" name="email" placeholder="Enter Your Email Address Here">

        <br>

        <label for="phone">Phone</label>
        <input type="number" name="phone" placeholder="Enter Your Phone Number Here">

        <br>

        <button type="submit">Sign Up</button>
```

This page has the functions as intended, and the inputs are well placed. There is less styling of inputs than my initial design included. This is due to styles not being an essential part of my project at this stage, and therefore will be implemented in a later stage.

Processing the signup

First, the values input into the HTML form must be retrieved into the process_signup.php file and placed into PHP variables. This is done using $_GET

```php
/* retrieve entered details from form */
$username = $_GET['username'];
$password = $_GET['password'];
$email = $_GET['email'];
$companyName = $_GET['companyName'];
$phoneNumber = $_GET['phone'];
```

I then used a MySQLi query to search the Credentials table in the database for a username of the same value as the input value.

```php
/* Check if username already exists */
$query = "SELECT Username FROM Credentials WHERE Username = '$username'";
$result = mysqli_query($connection, $query);
$numResults = mysqli_num_rows($result);
```

If the MySQLi query returns 0 results, the username entered is unique, and therefore an account can be made using the entered username. This includes inserting the entered values into the credentials table using SQL queries. Initially, only one query was planned. However, I soon noticed that if multiple queries were not used, I would be inserting blank values into some rows of the table.

```php
/* submit details to Credentials table */
if ($numResults == 0) {
    echo "Username is valid\n";// temp

    if ($phoneNumber == null and $email != null) {
        mysqli_query($connection, query: "INSERT INTO Credentials (Username, Password, Company, Email)
            VALUES ('$username', '$password', '$companyName', '$email')");
    } else if ($phoneNumber != null and $email == null) {
        mysqli_query($connection, query: "INSERT INTO Credentials (Username, Password, Company, Phone)
            VALUES ('$username', '$password', '$companyName', '$phoneNumber')");
    } else if ($phoneNumber == null and $email == null) {
        mysqli_query($connection, query: "INSERT INTO Credentials (Username, Password, Company)
            VALUES ('$username', '$password', '$companyName')");
    } else {
        mysqli_query($connection, query: "INSERT INTO Credentials (Username, Password, Company, Email, Phone)
            VALUES ('$username', '$password', '$companyName', '$email', '$phoneNumber')");
    }
```

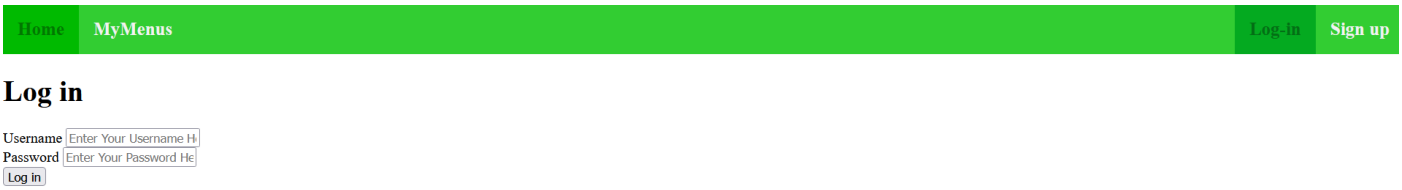
If the user enters a username already in use, the user will be told so via an ouput to the web page.

```php
else{
    echo "Username already in use\n";
}
```

The video **'signup database entry.mp4'** shows that the credentials table is empty. The user can then enter a set of valid inputs, and their data is shown to be stored in the credentials table of the database. The exact details are then attempted to be entered, but the username is already in use; therefore, the user is rejected sign up.

Login page

This form is very similar to the sign-up page, with fewer input boxes. I created input boxes for username and password and then included a submit button to post the inputs to the process_login.php page.



Log in

Username Enter Your Username H
Password Enter Your Password He
Log in

```
<form action="process_login.php"> <!-- send entered data into process_login.php for processing -

    <!-- toolbar -->
    <div class="toolbar">
        <a href="HomePage.php">Home</a>
        <a href="MyMenus.php">MyMenus</a>
        <a style="..." href="Signup.php">Sign up</a>
        <a class="current" style="..." href="Login.php">Log-in</a>
    </div>

    <div class ="loginContainer">

        <h1>Log in</h1>

        <label for="username">Username</label>
        <input type="text" name="username" placeholder="Enter Your Username Here" required>

        <br>

        <label for="password">Password</label>
        <input type="text" name="password" placeholder="Enter Your Password Here" required>

        <br>

        <button type="submit">Log in</button>
    </div>

</form>
```

Processing the login

First, I must retrieve the input data from the login page using $_GET

```php
/* Get username and password from HTML form */
$username = $_GET['username'];
$password = $_GET['password'];
```

I then need a SQL query to search for a matching username and password in the credentials table of the database and obtain the number of results. There is a matching account to the entered username and password if there are results to this query.

```php
$query = "SELECT Username, Password FROM Credentials WHERE Username = '$username' AND Password = '$password'";
$result = mysqli_query($connection, $query);
$numResults = mysqli_num_rows($result);
```

Using another SQL query, I will then select the user's company name, email, and phone number using SQL queries but only if results are returned. I will store these details as session variables. The user can then be given access to the account. If the username and password input do not match, the user cannot access the account and will be informed that this is not their account. I had initially planned to use only one SQL query to gather these three pieces of data from the credentials table. However, I did not know how to do this whilst also validating whether each value returned a result. Therefore I instead used 3 SQL statements, as shown below.

```php
if($numResults != 0){
    /* Get the values of companyName, email, and phoneNumber. If details are note provided, false is placed into the variables
    $query = "SELECT Company FROM Credentials WHERE Username = '$username'";
    $result = mysqli_query($connection,$query);
    $row = $result->fetch_row();
    $companyName = $row[0] ?? false;

    $query = "SELECT Email FROM Credentials WHERE Username = '$username'";
    $result = mysqli_query($connection,$query);
    $email = $result->fetch_row()[0] ?? false;

    $query = "SELECT Phone FROM Credentials WHERE Username = '$username'";
    $result = mysqli_query($connection,$query);
    $phoneNumber = $result->fetch_row()[0] ?? false;

    $_SESSION['username'] = $username;
    $_SESSION['password'] = $password;
    $_SESSION['companyName'] = $companyName;
    $_SESSION['email'] = $email;
    $_SESSION['phoneNumber'] = $phoneNumber;
```

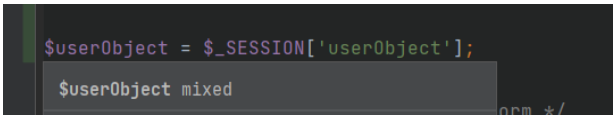
Push failed

```php
else{
    echo "this is not your account!";// temp
}
```

After doing this, I decided that using classes and objects would be a much better approach to storing the details of a user. I, therefore, tried this out, as shown below.

```php
class User{
    private $username;
    private $password;
    private $companyName;
    private $email;
    private $phoneNumber;

    function __construct($setUsername,$setPassword,$setCompanyName,$setEmail,$setPhoneNumber){
        $this->username = $setUsername;
        $this->password = $setPassword;
        $this->companyName = $setCompanyName;
        $this->email = $setEmail;
        $this->phoneNumber = $setPhoneNumber;
    }

    public function getUsername()
    {
        return $this->username;
    }

    public function getPassword()
    {
        return $this->password;
    }
```

However, this did not work out as the objects were not stored as variables of type ‘User’. They were stored as type ‘mixed’ when using session variables to pass the object to other files meaning that the objects would not work correctly in other files. Therefore, this change was reverted. This is shown in the screenshot below.



I realised that I had previously referred to creating multiple databases in my design. This should not be the case. The ‘Users’ database should be the only database, and in sections where another database is mentioned, this is supposed to be a table/ within the ‘Users’ database.

Testing

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Does the log-in page generate correctly sized	Open log-in page	The web page opens and is sized correctly	The web page opens and is sized correctly
1b	Does the signup page generate correctly sized	Open signup page	The web page opens and is sized correctly	The web page opens and is sized correctly
3a	After signing up, does all entered data get saved in the Users database?	Username Password Email@email.com 12345678910 Company	Username Password Email@email.com 12345678910 Company Are visible in a record of the Users database	Credentials are visible in the credentials table of the database. See **signup database entry.mp4**
2a	Attempt to Input alphabetic characters into the signup phone field and sign-up using them	User3 Pass3 Company3 Company3@email.com abcdefg0123	User is not allowed to sign up	The user is unable to sign-up whilst there are alphabetical characters in the phone input box. See **inputting non-numeric chars into sign-up.mp4**
2b	Attempt to signup using an invalid email address	Username2 Password email@email 01234567890 Company	Error – Invalid Email	The user is able to enter an invalid email – there is no email validation See **inputting invalid email.mp4**
2c	Attempt to signup using a username already in use	Username2 Password email@email.com 01234567890	Error – Username Already In Use	The user was told that that particular username was already in use. See **inputting invalid email.mp4**
3a	Attempt to log in using an incorrect password	Username2 NotPassword	Error – invalid credentials	The user is not logged into the account, however, is not informed of this. Therefore, it seems there is no error on their part
3b	Attempt to log in using an incorrect username	NotUsername Password	Error – credentials	The user is not logged into the account, however, is not informed of this. Therefore, it seems that there is no error on their part See **input invalid username.mp4**

During this test, test 2b failed because I had forgotten to implement email validation into my program. To fix this, I will implement email validation into my program by setting the type of the input box to email. Tests 3a and 3b partially failed; although the user was not logged into the account, they were not told that their credentials were invalid. This was because the user was redirected from the log-in page too soon. I will make the program wait before the user is redirected to fix this.

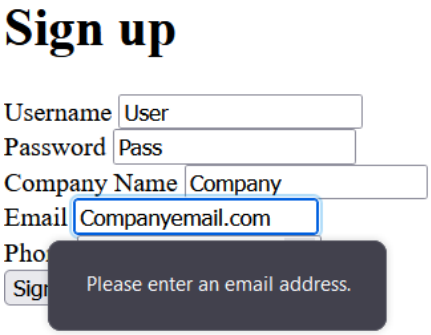
Email Validation

Set the type of the email input box to type='email' rather than type='text'

```
<label for="companyName">Company Name</label>
<input type="text" name="companyName" placeholder="Enter Your Company Name Here" required>
```

```
<label for="email">Email</label>
<input type="email" name="email" placeholder="Enter Your Email Address Here">
```

I then ran the following test to check this was working:

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Sign-up as a new user with an invalid email address	User Pass Company Companyemail.com 12345678910	User is told their email is invalid, and they must enter a valid email	Email is invalid See **invalid email.mp4** 
1b	Attempt to signup using an invalid email address	User1 Pass Company Company@email 12345678910	Error – Invalid Email	Username is valid Username Created. See **invalid email.mp4**
1c	Sign-up as a new user with a valid email address	User2 Pass Company Company@email.com 12345678910	The user is allowed to create an account.	Username is valid Username Created See **invalid email.mp4**

This test showed that test 1b failed. Although the current email validation prevented @ symbols from not being in place, it did not account for the need for at least 1 . at the end of the email. To fix this, I will need to create an algorithm in PHP. First, I will need to split the section of the email after the @ symbol from the rest, then check if this contains a dot. If it does, I will need to check there are alphabetic characters after the dot. Otherwise, the email is invalid.

```php
/* VALIDATE EMAIL */
/* If valid returns true */
/* If invalid returns false */
function validateEmail($email){
    $valid = false;

    /* search through every character in an email until @ is found */
    $count = 0;
    while ($count < strlen($email) AND $email[$count] != "@"){
        $count += 1;
    }
    if($email[$count] == "@") {
        $secondHalfOfEmail = substr($email, offset: $count + 1, length: strlen($email) - $count);

        /* Search for dot (.) in second half of email */
        $count = 0;
        while ($count < strlen($secondHalfOfEmail) AND $secondHalfOfEmail[$count] != "."){
            $count += 1;
        }
        if($secondHalfOfEmail == ".") {
            $emailAfterDot = substr($secondHalfOfEmail, offset: $count + 1, length: strlen($secondHalfOfEmail) - $count);

            /* Search for alphabetic character at the end of the email */
            $alphabetArray = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "
            $i = 0;
            while ($i < strlen($alphabetArray) and $valid == false) {

                $alphabeticCharacter = $alphabetArray[$i];
```

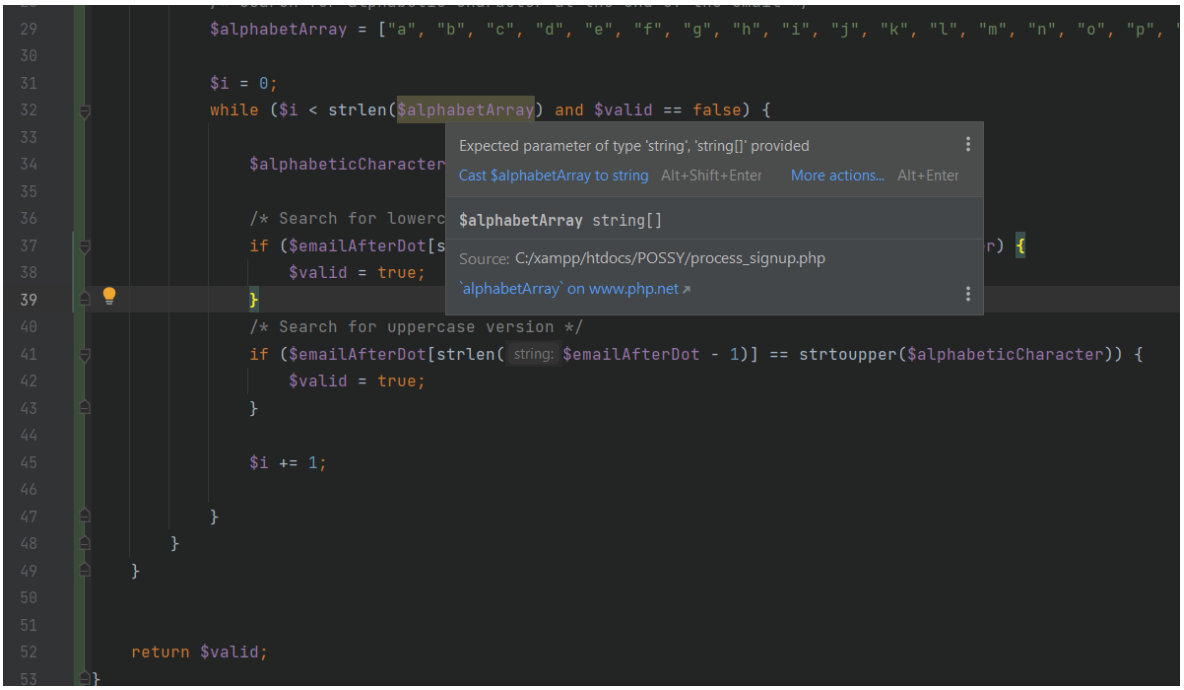
```php
                /* Search for lowercase version */
                if ($emailAfterDot[strlen( string: $emailAfterDot - 1)] == $alphabeticCharacter) {
                    $valid = true;
                }
                /* Search for uppercase version */
                if ($emailAfterDot[strlen( string: $emailAfterDot - 1)] == strtoupper($alphabeticCharacter)) {
                    $valid = true;
                }

                $i += 1;

            }
        }
    }
```

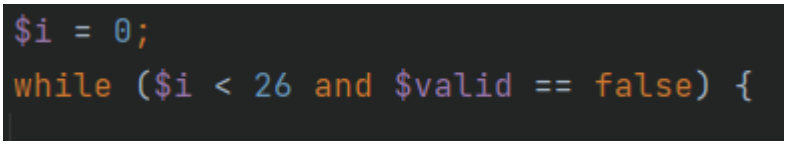
```php
/* submit details to Credentials table if username is not in use and email is valid*/
if ($numResults == 0 && validateEmail($email) == true) {
```

I then re-ran the previous tests.

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Sign-up as a new user with an invalid email address	User3 Pass Company3 Companyemail.com 12345678910	User is told their email is invalid, and they must enter a valid email	Email is invalid See **invalid email issue.mp4**
1b	Attempt to signup using an invalid email address	User4 Pass Company4 Company@email 12345678910	Error – Invalid Email	The username Is already in use See **invalid email issue.mp4**
1c	Sign-up as a new user with a valid email address	User100 Pass Company100 Company@email.com 12345678910	The user is allowed to create an account.	The username is already in use See **invalid email issue.mp4**

The results of this test display a failure for both tests 1b and 1c. In both cases, the username was said to be already in use when it was not. There was a warning that was unnoticed in the code, I tried to use strlen() to find the length of an array. This will be changed to use a constant value instead.





Re=testing the failed tests

Test No.	Description	Test Data	Expected Result	Actual Result
1b	Attempt to signup using an invalid email address	User30 Pass Company30 Company@email 12345678910	Error – Invalid Email	Fatal error = See **invalid email fatal error.mp4**
1c	Sign-up as a new user with a valid email address	User31 Pass Company31 Company@email.com 12345678910	The user is allowed to create an account.	Fatal error = See **invalid email fatal error.mp4**

Both tests 1b and 1c produced the same fatal error:

Fatal error: Uncaught TypeError: Unsupported operand types: string - int in C:\xampp\htdocs\POSSY \process_signup.php:38 Stack trace: #0 C:\xampp\htdocs\POSSY\process_signup.php(69): validateEmail('Company@email.c...') #1 {main} thrown in **C:\xampp\htdocs\POSSY\process_signup.php** on line **38**

```
37              /* Search for lowercase version */
38              if ($emailAfterDot[strlen( string: $emailAfterDot - 1)] == $alphabeticCharacter) {
39                  $valid = true;
40              }
```

There is an integer (-1) inside the brackets of the strlen() function. To fix this, place the -1 outside of the brackets of the strlen() function as was initially intended.

Re=testing the failed tests

Test No.	Description	Test Data	Expected Result	Actual Result
1b	Attempt to signup using an invalid email address	User50 Pass Company50 Company@email 12345678910	Error – Invalid Email	Warning – Uninitialised string Username is already in use See **invalid email warning.mp4**
1c	Sign-up as a new user with a valid email address	User51 Pass Company51 Company@email.com 12345678910	The user is allowed to create an account.	The user is allowed to create an account, and the account is created See **valid email validation.mp4**

Test 1b failed. It produced a warning, and the username was said to already be in use.

Warning: Uninitialized string offset 8 in **C:\xampp\htdocs\POSSY\process_signup.php** on line **26**
Username already in use

```
20          /* Search for dot (.) in second half of email */
21          $count = 0;
22
23          while ($count < strlen($secondHalfOfEmail) AND $secondHalfOfEmail[$count] != "."){
24              $count += 1;
25          }
26          if($secondHalfOfEmail[$count] == ".") {
27              $emailAfterDot = substr($secondHalfOfEmail,  offset: $count + 1,  length: strlen($secondHalfOfEmail) - $count);
28
```

The IF statement on line 26 tries to access a character off the end of the string array. To fix this, I need to first use an if statement making sure that the variable $count is not equal to the length of $secondHalfOfEmail. If it is, this statement should not be executed.

```
27          /* Check that count is not greater than the last index of the array of the string $secondHalfOfEmail */
28          if($count != $secondHalfOfEmail) {
29
30              if ($secondHalfOfEmail[$count] == "." and_) {
31                  $emailAfterDot = substr($secondHalfOfEmail, offset: $count + 1, length: strlen($secondHalfOfEmail) - $count);
32
33                  /* Search for alphabetic character at the end of the email */
34                  $alphabetArray = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t
35
36                  $i = 0;
37                  while ($i < 26 and $valid == false) {
38
39                      $alphabeticCharacter = $alphabetArray[$i];
40
41                      /* Search for lowercase version */
42                      if ($emailAfterDot[strlen($emailAfterDot) - 1] == $alphabeticCharacter) {
43                          $valid = true;
44                      }
45                      /* Search for uppercase version */
46                      if ($emailAfterDot[strlen($emailAfterDot) - 1] == strtoupper($alphabeticCharacter)) {
47                          $valid = true;
48                      }
49
50                      $i += 1;
51
52                  }
53              }
54          }
55      }
```

When resolving this bug, I also noticed that the user would be told the username is already in use, even if it is the email that is incorrect. I need to create a separate output for when the email is incorrect to fix this.

```
131     else{
132         /* Output error messages */
133         if($numResults != 0 ) {
134             echo "Username already in use\n";// temp
135         }
136         else{
137             echo "Email is invalid\n";// temp
138         }
139     }
140
```

I will now re-run the tests that previously failed:

Test No.	Description	Test Data	Expected Result	Actual Result
1b	Attempt to signup using an invalid email address	User70 Pass Company70 Company@email 12345678910	Error – Invalid Email	The email is said to be invalid See **email validation re-test.mp4**
1c	Sign-up as a new user with a valid email address	User71 Pass Company71 Company@email.com 12345678910	The user is allowed to create an account.	The user is signed up. See **email validation re-test.mp4**

All tests have now passed; therefore email validation is now working as was originally intended.

Fixing invalid errors not displaying

Previously, the user was being redirected without the page having a delay before being refreshed

```
header( header: 'Location: HomePage.php');// redirect to home page
```

The page should now have a delay before being refreshed

```
header( header: 'Refresh: 10; URL=HomePage.php');
```

Testing is shown in **testing refresh.mp4.** Although the testing had shown a working solution, the refresh time was too long at 10 seconds. I will therefore change the refresh time to 3 seconds, shortening the time before the redirect.

```
header( header: 'Refresh: 3; URL=HomePage.php');
```

End-of-stage Testing

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Does the log-in page generate correctly sized	Open log-in page	The web page opens and is sized correctly	The web page opens and is sized correctly See **End of stage 2 Test.mp4**
1b	Does the signup page generate correctly sized	Open signup page	The web page opens and is sized correctly	The web page opens and is sized correctly See **End of stage 2 Test.mp4**
2a	After signing up, does all entered data get saved in the Users database?	Username Password Company Email@email.com 12345678910	Username Password Email@email.com 12345678910 Company Are visible in a record of the Users database	Credentials are visible in the credentials table of the database. See **End of stage 2 Test.mp4**
3a	Attempt to Input alphabetic characters into the signup phone field and sign-up using them	User3 Pass3 Company3 Company3@email.com abcdefg0123	User is not allowed to sign up	The user cannot sign up whilst there are alphabetical characters in the phone input box. See **End of stage 2 Test.mp4**
3b	Attempt to signup using an invalid email address	Username2 Password Company email@email 01234567890	Error – Invalid Email	Error = invalid email See **End of stage 2 Test.mp4**
3c	Attempt to signup using a username already in use	Username Password Company Email@email.com 12345678910	Error – Username Already In Use	The user is told that the entered username is already in use. See **End of stage 2 Test.mp4**
4a	Attempt to log in using an incorrect password	Username2 NotPassword	Error – invalid credentials	This account is not yours! See **End of stage 2 Test.mp4**
4b	Attempt to log in using an incorrect username	NotUsername Password	Error – credentials	Invalid Username! See **End of stage 2 Test.mp4**
4c	Log-in correctly	Username Password	The user is logged in and redirected to the home page	The user is logged in and redirected to the home page See **End of stage 2 Test.mp4**

I was very happy with this stage of development; it was easy to complete, although the email validation section took a lot longer than intended, I believe this was because I tried to rush through the programming of this feature. Next time, I will take more time over each section focusing on the quality of the code, reducing bugs. From this stage in the project, I have met the following bullet points from my success criteria:

- A 'Users' database containing the account information of users after they have completed sign-up. It should contain:
    - Username (required)
    - Password (required)
    - Company Email (Required)
    - Phone number
- A sign-up page as described in requirements 1.2.0
    - User's entered details are stored in the 'Users' database
- A log-in page as described in requirements 1.3.0
    - User is allowed access if details entered match a user in the 'Users' database

I completed all tasks at this stage. If I have time, I would like to revisit how error messages are displayed in the login and signup pages, making them modal boxes rather than separate web pages, as I believe this will lead to a more professional project. I would also like to add some styling to those web pages, as was initially intended. I did not add styling at this stage to get a fully working project faster, rather than a half working but good-looking project.

Stage 3 – My Menus Page (3hours)

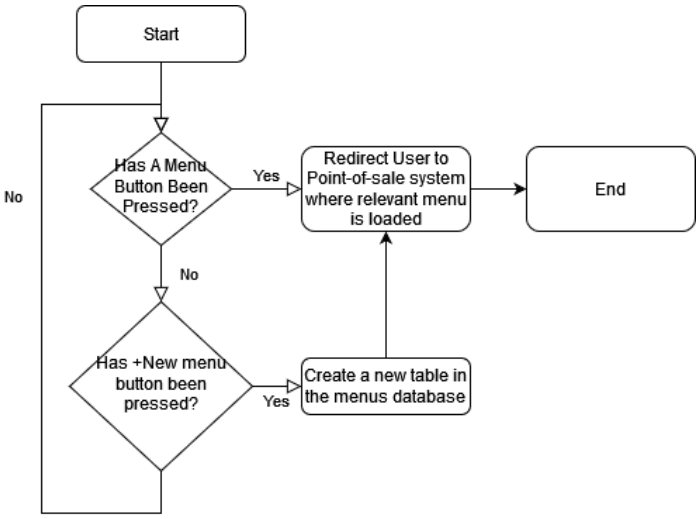
The MyMenus page allows the creation of a new menu by pressing the "+New Menu" button. Otherwise, a user may press on the button for a menu that they have already created. Pressing any such button will take the user straight to the point-of-sale system, loading the desired menu. This page must be created before the point-of-sale system, as it is required that users select or create a menu before gaining access to the point-of-sale page.

Features/functionality:

1. My Menus user interface
2. A Menus database
3. An algorithm to create a new entry in the menus table when a new menu is created
4. Buttons to redirect users to their pre-created menu inside the point-of-sale system
5. A table containing the sale items contained within a menu

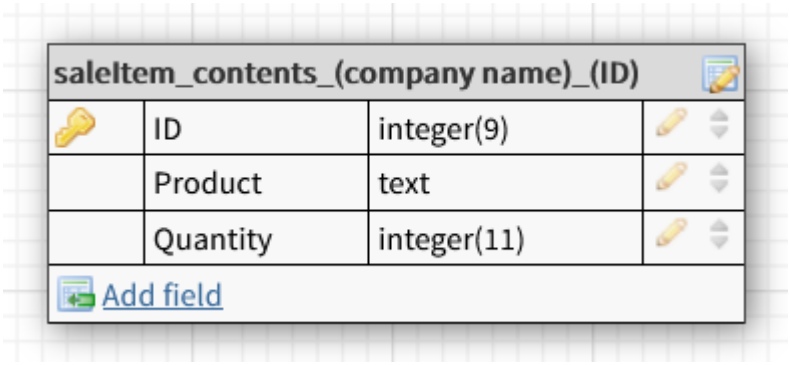
Test No.	Description	Type Of Test	Test Data	Expected Result	Actual Result
1a	Does the My Menus page generate correctly sized		Open My Menus page	The web page opens and is sized correctly	
2a	Can users create a new menu		Press "+NewMenu" Button	The point-of-sale system opens with a new menu	
3a	Can users open an existing menu		Press menu button	The point-of-sale system opens with the existing menu	

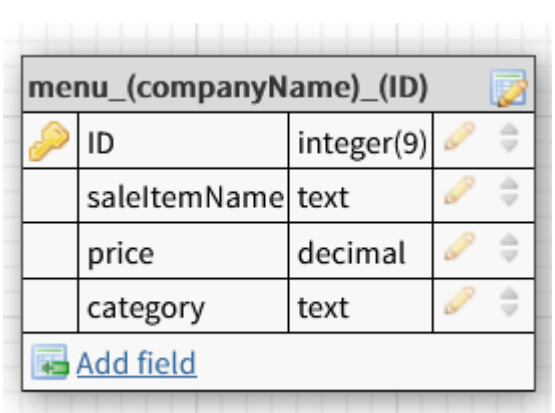
Flow charts

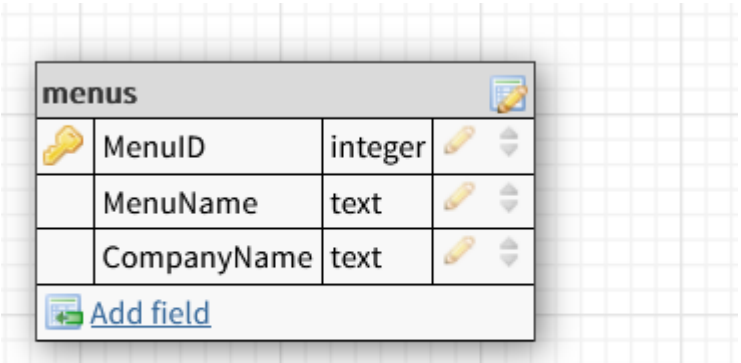


This flow chart shows how a user should access/ create a new menu in the system. The system will constantly check whether a menu button has been pressed, or whether the create new menu button has been pressed until one of them is pressed. If a menu button has been pressed, the relevant menu is loaded and the user is redirected to the point of sale system. If the create new menu button has been pressed, a new table is created within the menus database for this new menu, and the menu is loaded and the user is redirected to the point of sale system.

Database table designs for this stage:







SQL Statements

CREATE TABLE ‘menus’ (‘MenuID’ INT NOT NULL AUTO_INCREMENT, MenuName TEXT NOT NULL, ‘CompanyName’ TEXT NOT NULL, PRIMARY KEY (‘MenuID’))

SELECT MenuID FROM menus WJERE MenuName = ‘[current menu name]’ AND CompanyName = ‘[current company name]’

CREATE TABLE ‘[current menu name]’ (‘saleItemID INT NOT NULL AUTO_INCREMENT, saleItemName TEXT NOT NULL, price DOUBLE NOT NULL, category TEXT NOT NULL, PRIMARY KEY (‘saleItemID’))

Ability to open a pre-existing menu

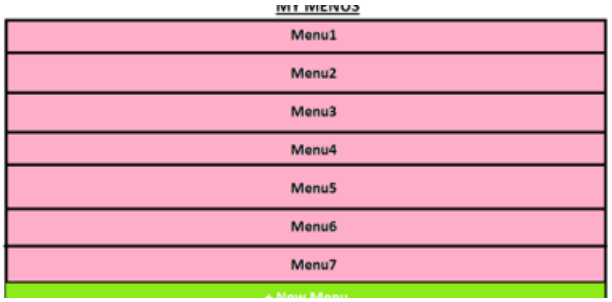
Whilst programming this section, I noticed that the menu would not be given a name unless entered before the "+new menu" button was pressed. Therefore, a text input field was placed next to this button, allowing for each menu to be given a name.

Creating a menus database table

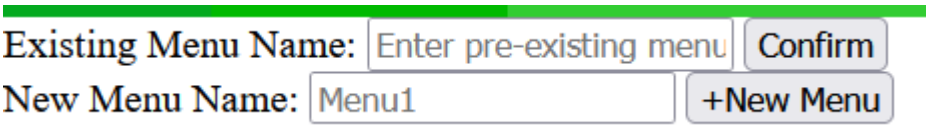
```
CREATE TABLE `users`.`menus` ( `MenuID` INT(9) NOT NULL
AUTO_INCREMENT , `MenuName` TEXT NOT NULL , `CompanyName` TEXT
NOT NULL , PRIMARY KEY (`MenuID`)) ENGINE = InnoDB;
```

Designing the user interface

I had planned to have this user interface as laid out, as shown below.



However, I could not figure out how to stack the menu buttons of pre-existing menus into a table, as was initially intended according to my interface design. Therefore, I will use a search box to search for your menu via name and compare the company name and menu name to ensure that that particular menu belongs to that company. However, this does mean that menu names must be unique within a company. It is possible to create multiple menus of the same name at this stage and it should not be. This could lead to bugs within the program, which could cause fatal errors, such as some menus not being accessible if they have the same name as another menu.



I ran the following test on my program:

Test No	Description	Data	Expected Outcome	Outcome
1.1	Sign-up with an account that is not already signed up	Username1 Password1 Company1 Company1@email.com 01273012730	User is signed up, and their details are added to the Credentials table in the database	User is signed up, and their details are added to the Credentials table in the database
1.2	Log in with the above details	Username1 Password1	User is logged into their account	User is logged into their account
1.3	Create a new menu for their company	Menu1	A new menu is created for their company called 'Menu1'	Warning: Undefined array key "companyName" on line 10 process_newMenu.php
1.4	Open the menu created in the above test and be redirected to the point of sale page	Menu1	The user is redirected to the pointOfSale.php web page	This test was not run as the menu was not created in the above test

Test 1.3 failed. The company name was not being passed correctly into the session variable.

Warning: Undefined array key "companyName" in **C:\xampp\htdocs\POSSY\process_newMenu.php** on line **10**

```
10  $companyName = $_SESSION['companyName'];
```

I checked the process_login.php file to check that the session variable was named correctly when initiated; There were no problems with this.

```
32
33      $_SESSION['username'] = $username;
34      $_SESSION['password'] = $password;
35      $_SESSION['companyName'] = $companyName;
36      $_SESSION['email'] = $email;
37      $_SESSION['phoneNumber'] = $phoneNumber;
38  }
```

It turned out that none of the session variables had been created in this session. This was a one-time error. I logged back into the pre-made account, and it was working well and no longer throwing errors after re-loading the local host. Therefore, I will note this down as a bug at this point and perhaps come back to investigate further at the end of my project.

Creating a table for a new menu to contain its sale items.

I needed first to run a MySQLi query to get the ID of the newly created menu from the menus table in the database. I then need to use this ID to form a table name for the new table. After this, I need a new MySQLi query to create the new table for the menu.

```
/* Get the Menu ID */
$result = mysqli_query($connection, query: "SELECT MenuID FROM menus
          WHERE MenuName = '$menuName' AND CompanyName = '$companyName'");
$row = mysqli_fetch_assoc($result);
$menuID = $row['MenuID'];

/* Create a table to contain the data for the new menu */
$menuName = "menu_".$companyName."_".$menuID;

$query = "CREATE TABLE `users`.`$menuName` ( `saleItemID` INT(9) NOT NULL AUTO_INCREMENT ,
          `saleItemName` TEXT NOT NULL , `price` DOUBLE NOT NULL , `category` TEXT NOT NULL ,
           PRIMARY KEY (`saleItemID`))";
mysqli_query($connection, $query);
```

End-Of-Stage Testing

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Does the My Menus page generate correctly sized	Open My Menus page	The web page opens and is sized correctly	The web page opens and is sized correctly See **MyMenus Testing.mp4**
2a	Can users create a new menu	Input “MyNewMenu” into the new menu input box, then press “+NewMenu” Button	The point-of-sale system opens with a new menu	A new menu is created, but the page is reloaded to the myMenus page. See **MyMenus Testing.mp4**
3a	Can users open an existing menu	Enter “MyNewMenu”, then press the submit button	The point-of-sale system opens with the existing menu	The user is redirected to the point-of-sale page See **MyMenus Testing.mp4**

Although test 2a only partially passed as it did not redirect the user to the point-of-sale page, it instead redirects them back to the myMenus page, I have decided that I prefer the user to be redirected to the myMenus page; therefore, this will not be changed.

I have been happy with the development of this stage; it was pretty simple with a small number of bugs. This stage does not meet any of my success criteria, although it is essential to allow users to access multiple menus.

Although I have included most of the intended features during this stage of development, I have not included buttons for each of a company’s menus to redirect the user to that menu in the point-of-sale page as intended. This is due to my lack of experience using HTML and CSS. Therefore, if time permits, I would like to return to this at the end of my project, implementing the design as initially intended. Otherwise, I would like to style the existing buttons, making the user interface appear more professional.

Stage 4 – Stock Management System – ESSENTIALS (6hours)

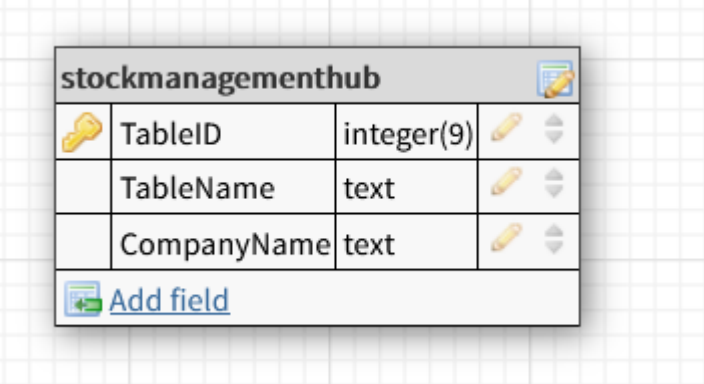
The stock management system will control the stock levels of all products via the stock management database table; the point-of-sale system will automatically update the stock levels in the database in the vast majority of cases. Therefore, the stock management system is needed before the point-of-sale system is created.

Features/functionality:

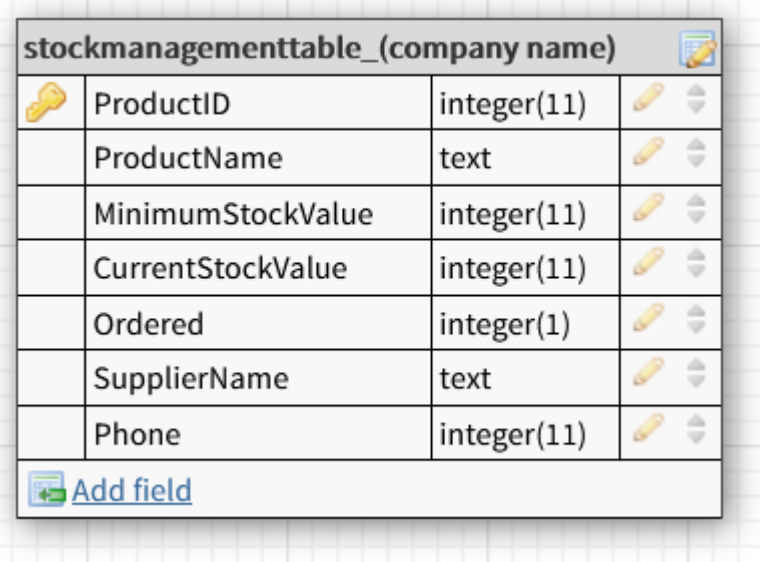
1. A stock management database hub table
2. An algorithm to create a new stock management table for each company when a new company is signed up to POSSY
3. A stock management overview page
    a. A toolbar
    b. A table containing item the headings name, current quantity, and minimum quantity
    c. Edit buttons next to each non-heading row of the table
    d. A +New Item button in the final heading that contains the edit buttons
4. A page used to create & edit products in the stock management database
5. An algorithm used to update/ add a new product to the stock management system table
6. A function to increase/decrease the stock quantity of a specified product
7. A function to check that all items in the stock management database table are above their minimum stock quantities. If a product is below its minimum quantity, a low stock alert should be returned to the point-of-sale system.

Test No.	Description	Test Data	Expected Result	Actual Result
1a	The +New item button redirects you to the Stock Management – Edit page	Click +New item button	The Stock Management – EDIT page opens	
1b	New products can be created and are added to the stock management database table	Sausage 100 167 True 01/23/2030	The test data is held in a record of the stock management database table	
1c	The stock management overview page is updated when a new product is added to the stock management database table	Bacon 100 167 False	Bacon, 100, 167 appears in a record of the table on the stock management overview page	
2a	Enter the name of a product the same as an existing product	Sausage 100 167 False	Error – Product Already Exists	
3a	Enter a negative quantity into the current stock quantity field	-2	Error – Invalid Quantity	
3b	Enter a negative quantity into the minimum stock quantity field	-2	Error – Invalid Quantity	

Database table design for this stage:

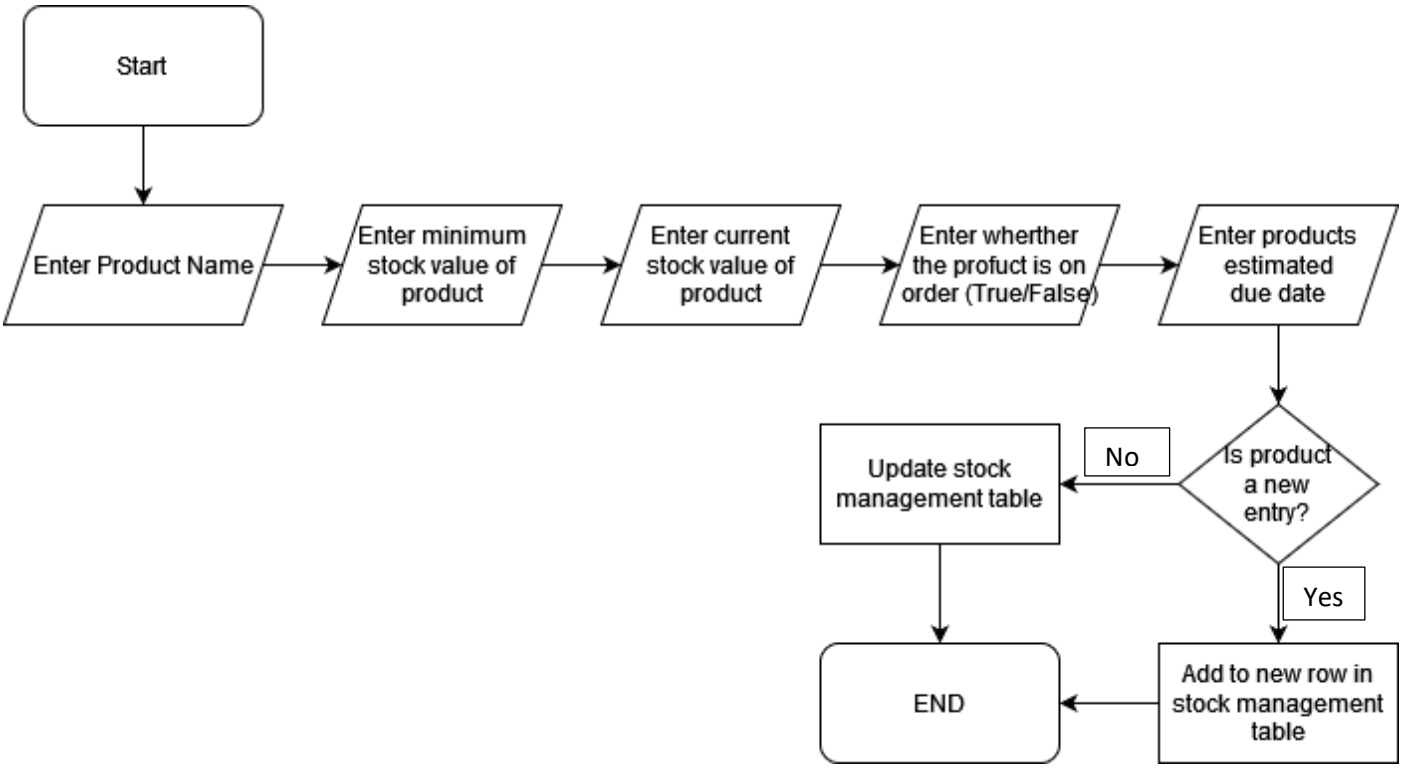


stockmanagementhub		
TableID	integer(9)	
TableName	text	
CompanyName	text	
Add field		

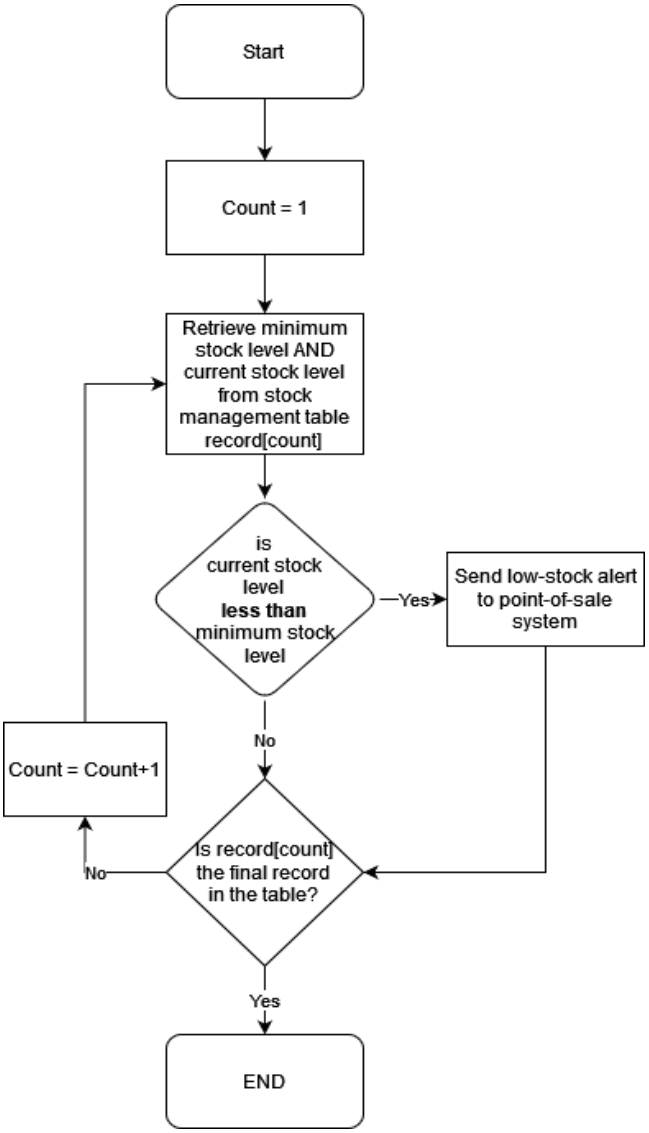


stockmanagementtable_(company name)		
ProductID	integer(11)	
ProductName	text	
MinimumStockValue	integer(11)	
CurrentStockValue	integer(11)	
Ordered	integer(1)	
SupplierName	text	
Phone	integer(11)	
Add field		

Flow charts



This shows how the edit/create product page should work, and what should happen depending on whether the details entered refer to an existing product, or whether a new product should be created in the database. The user must first enter a product name, a minimum stock value, a current stock value, whether the stock is on order, and an estimated due date. If the produce is a new entry, a new row is appended to the stock management table of the database. Otherwise, the stock management table is updated for the current product.



This shows how the stock management system checks for products that are low on stock, and presents the conditions in which a low stock alert should be triggered.First, a counter variable is set. This count variable is then used to retrieve the minimum and current stock value of each product in the stock management table, and it is checked whether the product is below its minimum stock value. If it is, a low stock alert is sent to the user. Otherwise the loop continues and the counter is increased unless the last checked values were contained within the final record of the stock management database table.

SQL Queries

CREATE TABLE ‘stockManagementHub’(‘TableID’ INT NOT NULL AUTO_INCREMENT, ‘TableName’ TEXT NOT NULL, ‘CompanyName’ TEXT NOT NULL, PRIMARY KEY(‘TableID’))

SELECT CompanyName FROM Credentials WHERE CompanyName = ‘[current company name]’

CREATE TABLE ‘stockmanagementTable_[current company name]’ (‘ProductID’ INT NOT NULL AUTO_INCREMENT, ProductName TEXT NOT NULL, ‘MinimumStockValue’ INT NOT NULL, ‘Ordered’ INT NOT NULL, ‘SuppluerName’ TEXT NULL, ‘Phone’ INT(11) NULL, PRIMARY KEY(‘ProductID’))

SELECT ProductName, MinimumStockValue, CurrentStockValue,Ordered,SupplierName,Phone FROM ‘stockmanagementtable_(company’s name)’

Creating a table to store the names of all other stock management tables

I needed to run the below query in the database to create the 'Stock management hub' table. This is a permanent table; therefore, it was not included in the program's code.

```
CREATE TABLE `users`. ( `TableID` INT(9) NOT NULL
AUTO_INCREMENT , `TableName` TEXT NOT NULL , `CompanyName`
TEXT NOT NULL , PRIMARY KEY (`TableID`)) ENGINE = InnoDB;
```

Creating a stock management table during sign-up if one does not already exist for a user's company

I needed to first check if a stock management table for a particular company already existed by running a query searching for the entered company name in the Credentials table of the database. If this returns only one result, the current user is the only user with their company name. This means that a stock management table for their company would not yet have been created, and one needs to be created for their company using a SQL query.

Originally, the column "Ordered" was supposed to be an integer, with 1 as on and 0 as off. I decided that text using 'on' and 'off' is easier, as it is more descriptive. Therefore, I changed this to be type text rather than type number.

```
/* Check if stock management table already exists for that company, if not create one. */
$result = mysqli_query($connection, query: "SELECT Company FROM Credentials WHERE Company = '$companyName'");
$numResults = mysqli_num_rows($result);

if($numResults == 1){
    $tableName = "stockmanagementTable_".$companyName;

    mysqli_query($connection,
        query: "CREATE TABLE `users`.`$tableName` ( `ProductID` INT NOT NULL AUTO_INCREMENT ,
                `ProductName` TEXT NOT NULL , `MinimumStockValue` INT NOT NULL , `Ordered` INT NOT NULL ,
                `SupplierName` TEXT NULL DEFAULT NULL , `Phone` INT(11) NULL ,
                PRIMARY KEY (`ProductID`)); ");
```

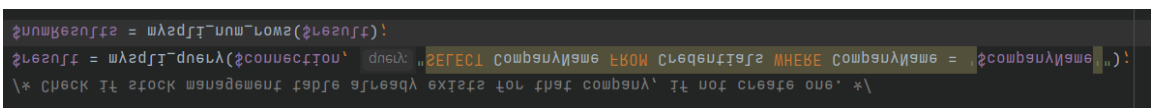
I then used the following test on this section of the program:

Test no	Description	Data	Expected Outcome	Outcome
1.1	Sign-up with an account that is not already signed up	Username2 Password2 Company2 Company2@email.com 01273012731	The user is signed up, and their details are added to the Credentials table in the database. A table 'stockmanagementTable_Company2' is created in the database	The user is signed up, and their details are added to the Credentials table in the database. Fatal Error: Uncaught TypeError: mysqli_num_rows() process_signup.php line 38

The test threw errors as I had written the incorrect field name in the MySQLi query, therefore throwing an error when I tried to put a Boolean value into mysqli_num_rows() as mysqli_query() returned false due to there being no results to the query.

Username is valid
Fatal error: Uncaught TypeError: mysqli_num_rows(): Argument #1 ($result) must be of type mysqli_result, bool given in C:\xampp\htdocs\POSSY\process_signup.php:38 Stack trace: #0 C:\xampp\htdocs\POSSY\process_signup.php(38): mysqli_num_rows(false) #1 {main} thrown in **C:\xampp\htdocs\POSSY\process_signup.php** on line **38**

This was fixed by changing the field name to the correct field name as in the Credentials table, as shown below.



Designing a stock management overview table

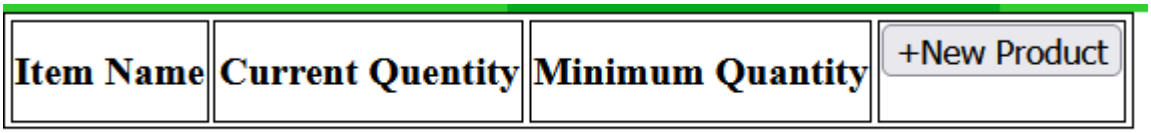
First, I needed to create a table with the headings Item Name, Current Quantity, Minimum Quantity, and Edit.

```
echo "<div class='Container_SM_Overview'>
    <table id='stockManagementOverview'>
        <!-- Set Table Headings -->
        <tr>
            <th>Item Name</th>
            <th>Current Quentity</th>
            <th>Minimum Quantity</th>
            <th><form action='newProduct.php'><button type='submit'>+New Product</button></form></th>
        </tr>
        ".

    "</table>
</div>";
```

I tested that this displayed correctly by running the following tests:

Test no	Description	Data	Expected Outcome	Outcome
2.1	Log in to the pre-made account from the previous test	Username2 Password2	The user is logged into the account.	The user is logged into the account.
2.2	Create a new menu under the company name 'Company2'	ourFirstMenu	A new menu, ourFirstMenu is created	A new menu, ourFirstMenu is created
2.3	Open the newly created, ourFirstMenu	ourFirstMenu	The pointOfSale.php page is opened	The pointOfSale.php page is opened
2.4	Access the stockManagement.php web page	press on 'Stock Management' on the toolbar	The user is redirected to the stock management page	The user is redirected to the stock management page

Here is the resulting page:



Although it would be nice for the '+New Product' to fill the entire heading box, I am happy with this design, and no further improvements need to be made.

Filling the stock management overview table with values

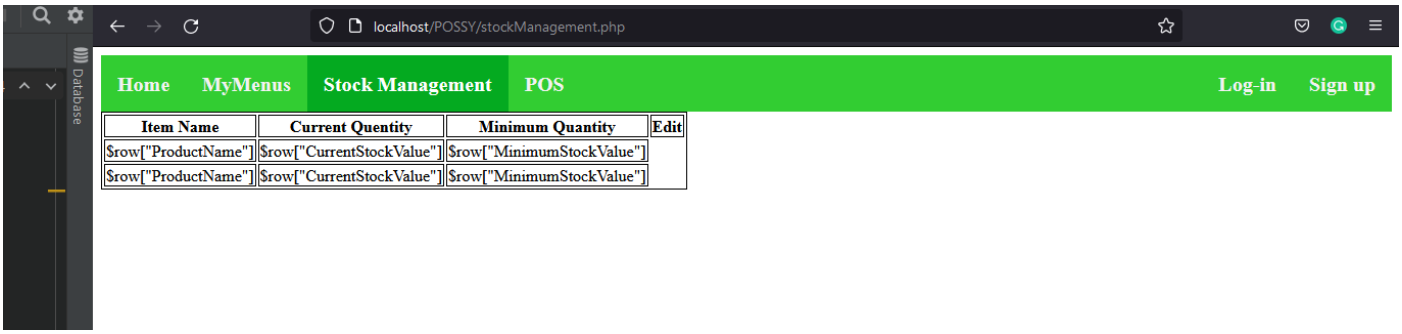
I will need to create a function that runs an SQL query retrieving all rows of the stock management table for a particular company. I then need to loop through the results of the query row by row, inserting the item name, current quantity, and minimum quantity for each product in the companies stock management table.

```php
function addEntriesToTable($companyName, $connection){
    $tableName = "stockmanagementtable_".$companyName;
    $query = "SELECT ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone
              FROM $tableName";
    $result = mysqli_query($connection, $query);

    $row = mysqli_fetch_assoc($result);

    /* Ouput data of each row */
    $ret = "";
    for($i=0;$i<getNumRows($companyName,$connection);$i++){
        $ret = $ret.'<tr><td> $row["ProductName"] </td>
                    <td> $row["CurrentStockValue"]</td>
                    <td> $row["MinimumStockValue"]</td></tr>';
    }
    return $ret;
}
?>
```

To be able to test that this feature is working, I will need to fill the stock management overview table with values from the stock management of a particular company to check. I am already logged into company2 from the last test; therefore, I will manually insert values into their stock management table for testing purposes. The query I ran is shown below

```sql
INSERT INTO stockmanagementtable_company2(ProductName,MinimumStockValue,CurrentStockValue) VALUES("Carrot","100","200");
INSERT INTO stockmanagementtable_company2(ProductName,MinimumStockValue,CurrentStockValue) VALUES("Banana","200","400");
```

When trying to fill the stock management overview table, The table was filled but filled with variables, rather than the contents of the variable.



```php
function addEntriesToTable($companyName, $connection){
    $tableName = "stockmanagementtable_".$companyName;
    $query = "SELECT ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone
              FROM $tableName";
    $result = mysqli_query($connection, $query);

    $row = mysqli_fetch_assoc($result);

    /* Ouput data of each row */
    $ret = "";
    for($i=0;$i<getNumRows($companyName,$connection);$i++){
        $ret = $ret.'<tr><td> $row["ProductName"] </td>
                    <td> $row["CurrentStockValue"]</td>
                    <td> $row["MinimumStockValue"]</td></tr>';
    }
    return $ret;
}
?>
```

The fix to this was relatively simple as I knew that the variables were being outputted as strings. Therefore I just had to use a lot of concatenation to get the variable's value into the table, rather than the variable itself.

```php
    for($i=0;$i<getNumRows($companyName,$connection);$i++){
        $ret = $ret.'<tr><td>'.$row["ProductName"].'</td>
                    <td> '.$row["CurrentStockValue"].'</td>
                    <td>'.$row["MinimumStockValue"].'</td></tr>';
    }
    return $ret;
```

A new problem arose. The same data was being placed into every row of the table.

Item Name	Current Quentity	Minimum Quantity	Edit
Carrot	200	100	
Carrot	200	100	

I placed the mysqli_fetch_assoc() function outside of the for loop, therefore outputting the same values each time, as the row was not updated to the next row of the results. This was easily fixed by placing the function inside the for loop.

```php
  for($i=0;$i<getNumRows($companyName,$connection);$i++){
      $row = mysqli_fetch_assoc($result);
      $ret = $ret.'<tr><td>'.$row["ProductName"].'</td>
                  <td> '.$row["CurrentStockValue"].'</td>
                  <td>'.$row["MinimumStockValue"].'</td></tr>';
  }
  return $ret;
```

Item Name	Current Quentity	Minimum Quantity	+New Product
Carrot	200	100	

Overseen content

Whilst programming this stage, I noticed that there was no point in which I added each company's stock management table to the stock management hub table in my plan. Therefore, I have included in this stage a SQL query to insert the table name and company name of a stock management table into the stock management hub table inside the database.

```
$query = "INSERT INTO stockmanagementhub(TableName,CompanyName)
          VALUES('$tableName', '$companyName')";
mysqli_query($connection,$query);
```

Creating a "button" sending the user to the edit product page

I need to create a hyperlink containing the ID of a product and the table name it is contained within. This is so that values such as the current and minimum stock quantity can be accessed when editing a product.

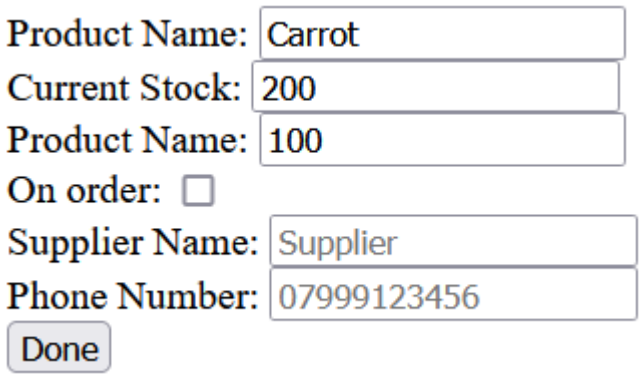
```
<td><a href="EditProduct.php?ProductID='.$row['ProductID'].'&TableName='.$tableName.'">Edit Product</a></td></tr>';
```

Item Name	Current Quentity	Minimum Quantity	+New Product
Carrot	200	100	Edit Product
Banana	400	200	Edit Product

Creating an edit/create new product user interface

I need to create a HTML form containing multiple input boxes, each with the ability to update a certain value of a product. This form also needs a submit button enabling the form to be submitted to a processing page.

```
echo "<form action='process_editProduct.php'>
    <label for='ProductName'>Product Name:</label>
    <input type='text' name='ProductName' value='$_GET[ProductName]' required>

    <br>

    <label for='CurrentStockValue'>Current Stock:</label>
    <input type='text' name='CurrentStockValue' value='$_GET[CurrentStockValue]' required>

    <br>

    <label for='MinimumStockValue'>Product Name:</label>
    <input type='text' name='MinimumStockValue' value='$_GET[MinimumStockValue]' required>

    <br>

    <label for='Ordered'>On order:</label>
    <input type='hidden' value='off' name='Ordered'> <!-- to solve not being posted if off -->
    <input type='checkbox' name='Ordered' value='$_GET[Ordered]'>

    <br>

    <label for='SupplierName'>Supplier Name:</label>
    <input type='text' name='SupplierName' placeholder='Supplier' value='$_GET[SupplierName]'>

    <br>

    <label for='PhoneNumber'>Phone Number:</label>
    <input type='tel' name='PhoneNumber'  pattern='[0-9]{11}' placeholder='07999123456' value='$_GET[PhoneNumber]'>

    <br>
    <button type='submit'>Done</button>
```



A new/existing product processing page

I need to create a processing page with an SQL query to update product values in the company's stock management system after manually editing details of a product, e.g. the product name, minimum stock level, current stock level etc.

```
$query = "'UPDATE $tableName
          SET(ProductName=$productName,MinimumStockValue=$minimumStockValue,CurrentStockValue=$currentStockValue,
          Ordered=$ordered,SupplierName=$supplierName,Phone=$phoneNumber)
          WHERE productID=$productID";
mysqli_query($connection,$query);
```

Whilst programming the ability to add a new product to the stock management system, I discovered that check boxes do not post a value if the value is off; therefore, I kept getting errors in my code. This was fixed by a hidden input box being overwritten if the value is on, and otherwise the hidden input box had a default value of off therefore off was posted.

> **Warning**: Undefined array key "Ordered" in **C:\xampp\htdocs\POSSY\process_newProduct.php** on line **9**
> stockmanagementtable_Cafe

```
<label for='Ordered'>On order:</label>
<input type='checkbox' name='Ordered'>

<br>
```

```
    <label for='Ordered'>On order:</label>
    <input type='hidden' value='off' name='Ordered'> <!-- to solve not being posted if off -->
    <input type='checkbox' name='Ordered' value='on'>
```

Low Stock Alerts

I cannot create a modal box as intended for low stock alerts, as I do not know how to program this. Therefore, I will instead have the same user interface design; however, the previous modal box for low stock alerts will now be a web page for simplicity. I will first need to create a trigger in the stockManagement.php file. This will be done by first using an SQL query, searching through every product in the company's stock management database table and checking whether it is below its minimum stock value. IF the product is below this value, it is returned from the MySQLi query. If the query has results, the company name and table name will be passed into $_SESSION variables, and the user will be redirected to the lowStock.php web page.

```
79  function checkForLowStock($connection,$companyName){
80      $tableName = "stockmanagementtable_".$companyName;
81      $query = "SELECT ProductID,ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone
82                FROM $tableName WHERE CurrentStockValue < MinimumStockValue AND Ordered='off'";
83      $result = mysqli_query($connection, $query);
84
85      if(mysqli_num_rows($result) > 0) {
86          $_SESSION['returnedRows'] = mysqli_fetch_assoc($result);
87          $_SESSION['companyName'] = $companyName;
88          $_SESSION['tableName'] = $tableName;
89
90          header( header: 'Location: lowStock.php');// redirect user
91          exit;
92
93      }
94  }
```

I then need to use the lowStock.php web page to output the product name, its current stock value, its minimum stock value, the supplier's name, and the supplier's phone number. The user then needs to press a button, either 'Ordered' or 'Unordered'. IF stock is ordered, the user will be sent to the processing page. Otherwise, they will be redirected back to their point-of-sale page.

```
echo "Product Name: ".$row['ProductName']."<br>Minimum Stock Value: ".$row['MinimumStockValue']."<br>".
     "Current Stock Value: ".$row['CurrentStockValue']."<br>Supplier Name: ".$row['SupplierName']."<br>".
     "Phone Number: ".$row['Phone'];

echo " <form action='process_lowStock.php'>
       <button type='submit'>Ordered</button>
       </form> ";

echo "<form class='lowStock' action='pointOfSale.php'>
     <button type='submit' >Ignore</button>
     </form>";
?>
```

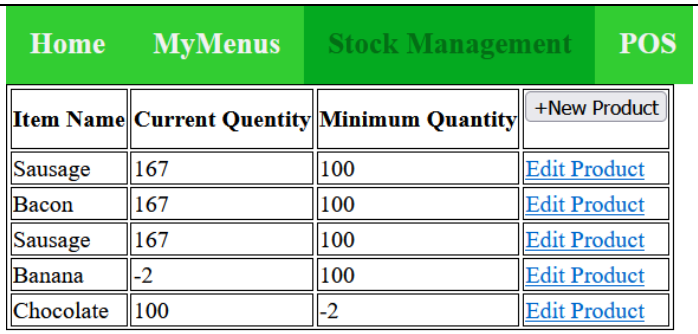
Processing a low stock alert

This was supposed to be created in the next stage of development. However, it is needed for low stock alerts to work effectively. Therefore, I decided instead to include this feature in this stage of development as I would otherwise need to return to this development stage at a later date.

A MySQLi query needs to be run, updating the value of the 'ordered' column in the company's stock management table. The 'ordered' value must be changed from 'off' to 'on' for a product of which new stock has been ordered. This is to prevent this alert from being triggered again when the stock has already been ordered.

```
$query = "UPDATE ".$_SESSION['TableName']." SET Ordered = 'on' WHERE ProductID = ".$_SESSION['ProductID']."'
mysqli_query($connection,$query);
?>
```

Testing

Test No.	Description	Test Data	Expected Result	Actual Result
1a	The +New item button redirects you to the Stock Management – Edit page	Click +New item button	The Stock Management – EDIT page opens	The Stock Management – EDIT page opens
1b	New products can be created and are added to the stock management database table	Sausage 167 100 False 01/23/2030	The test data is held in a record of the stock management database	 See **stock management testing.mp4**
1c	The stock management overview page is updated when a new product is added to the stock management database table	Bacon 167 100 False	Bacon, 100, 167 appears in a record of the table on the stock management overview page	 See **stock management testing.mp4**
2a	Enter the name of a product the same as an existing product	Sausage 167 100 False	Error – Product Already Exists	The product was added to a record in the stock management table See **stock management testing.mp4**
3a	Enter a negative quantity into the current stock quantity field	-2	Error – Invalid Quantity	The product was added to a record in the stock management table See **stock management testing.mp4**
3b	Enter a negative quantity into the minimum stock quantity field	-2	Error – Invalid Quantity	The product was added to a record in the stock management table See **stock management testing.mp4**

Tests 2a, 3a, and 3b failed. Test 2a failed as there is no validation to check whether or not a product of the same name already exists. To resolve this, I will need to implement an SQL query searching for the name of all products. I will then need to compare this to the name provided and output a message if a match is found. Otherwise, a new product will be created. Test 3a and 3b failed as there is no validation to check whether a negative number has been input. I will need to implement an if statement to resolve this, checking whether the number is greater than or equal to zero. Otherwise, there will be a message output, and the product will not be created

Resolving Issue 2a

First, I will create the SQL query mentioned above.

```
$query = "SELECT ProductName FROM $tableName";
$result = mysqli_query($connection, $query);
```

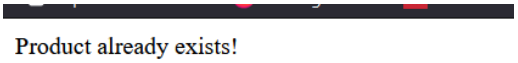
I will then need to loop through these results, comparing the product names to the name of the product entered by the user.

```
/* See if product name already exists */
/* Run SQL query searching for all products and compare product names to entered product name */
function isProductInTable($connection,$tableName,$productName){
    $query = "SELECT ProductName FROM $tableName";
    $result = mysqli_query($connection, $query);

    for ($i = 0; $i < mysqli_num_rows($result); $i++) {
        $row = mysqli_fetch_assoc($result);
        if ($row['ProductName'] == $productName) {
            return true;
        }
    }
    return false;
}
```

The user then needs to be told that the product already exists if this function returns true.

```
    if(isGreaterThanOrEqualToZero($minimumStockValue) AND isGreaterThanOrEqualToZero($currentStockValue)) {
        $query = "INSERT INTO $tableName(ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone)
          VALUES('$productName', '$minimumStockValue', '$currentStockValue', '$ordered',
          '$supplierName', '$phoneNumber')";
        mysqli_query($connection, $query);
        echo "Product Successfully Added";
    }
    else{
        echo "Value Smaller Than Zero Entered, INVALID!";
    }
}
else{
    echo "Product already exists!";
}
```

Checking the solution is working:

Test No.	Description	Test Data	Expected Result	Actual Result
2a	Enter the name of a product the same as an existing product	Sausage 167 100 False	Error – Product Already Exists	Error = Product Already Exists  Product already exists!

Resolving Issues 3a and 3b

This is a simple fix, just one if statement is needed to check the values input are greater than zero. Otherwise, a message needs to be output to the user stating that the value entered is too small.

```
/* Checks if a value is greater than  or equal to zero. */
/* Returns true if value is greater than or equal to zero */
/* Returns false if value is less then 0 */
function isGreaterThanOrEqualToZero($value){
    if($value >= 0){
        return true;
    }
    else {
        return false;
    }
}
```

```
if(isGreaterThanOrEqualToZero($minimumStockValue) AND isGreaterThanOrEqualToZero($currentStockValue)) {
    $query = "INSERT INTO $tableName(ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone)
      VALUES('$productName', '$minimumStockValue', '$currentStockValue', '$ordered',
      '$supplierName', '$phoneNumber')";
    mysqli_query($connection, $query);
}
```

End-Of-Stage Testing

Test No.	Description	Test Data	Expected Result	Actual Result
1a	The +New item button redirects you to the Stock Management – Edit page	Click +New item button	The Stock Management – EDIT page opens	The stock management edit page opens See **end-of-stage testing stock management.mp4**
1b	New products can be created and are added to the stock management database table	Sausage 167 100 False 01/23/2030	The test data is held in a record of the stock management database table	The test data is held in a record of the stock management database table 
1c	The stock management overview page is updated when a new product is added to the stock management database table	Bacon 167 100 False	Bacon, 100, 167 appears in a record of the table on the stock management overview page	Bacon, 100, 167 appears in a record of the stock management overview table  See **end-of-stage testing stock management.mp4**
2a	Enter the name of a product the same as an existing product	Sausage 167 100 False	Error – Product Already Exists	Product already exists! See **end-of-stage testing stock management.mp4**
3a	Enter a negative quantity into the current stock quantity field	Banana -2 100	Error – Invalid Quantity	Invalid input value! See **end-of-stage testing stock management.mp4**
3b	Enter a negative quantity into the minimum stock quantity field	Banana 100 -2	Error – Invalid Quantity	Invalid input value! See **end-of-stage testing stock management.mp4**

I have been happy with the development of this stage; it took as long as was expected. The problems in this stage again came from my oversight in the validation of input fields, something I need to continue to work on into the next stage. This stage has met the following success criteria:

- A stock management database table as outlined in requirements 3.1
- The ability to add new products to the stock management database table as outlined in requirements 3.3
- Manual edits to the stock management database table to compensate for out of the box factors such as wastages as outlined in requirements 3.2.1

It has also begun to meet the following success criteria:

- Low stock level alerts output to the operator as detailed in requirements 3.6.0

I have included all of the intended features during this stage of development. As with many of the previous stages, I would like to return to this stage if time permits at the end of the project to style the pages, particularly input boxes and buttons, making the user interface appear closer to my intended user interface designs.

Stage 5 – Point-Of-Sale system – ESSENTIALS (8hours)

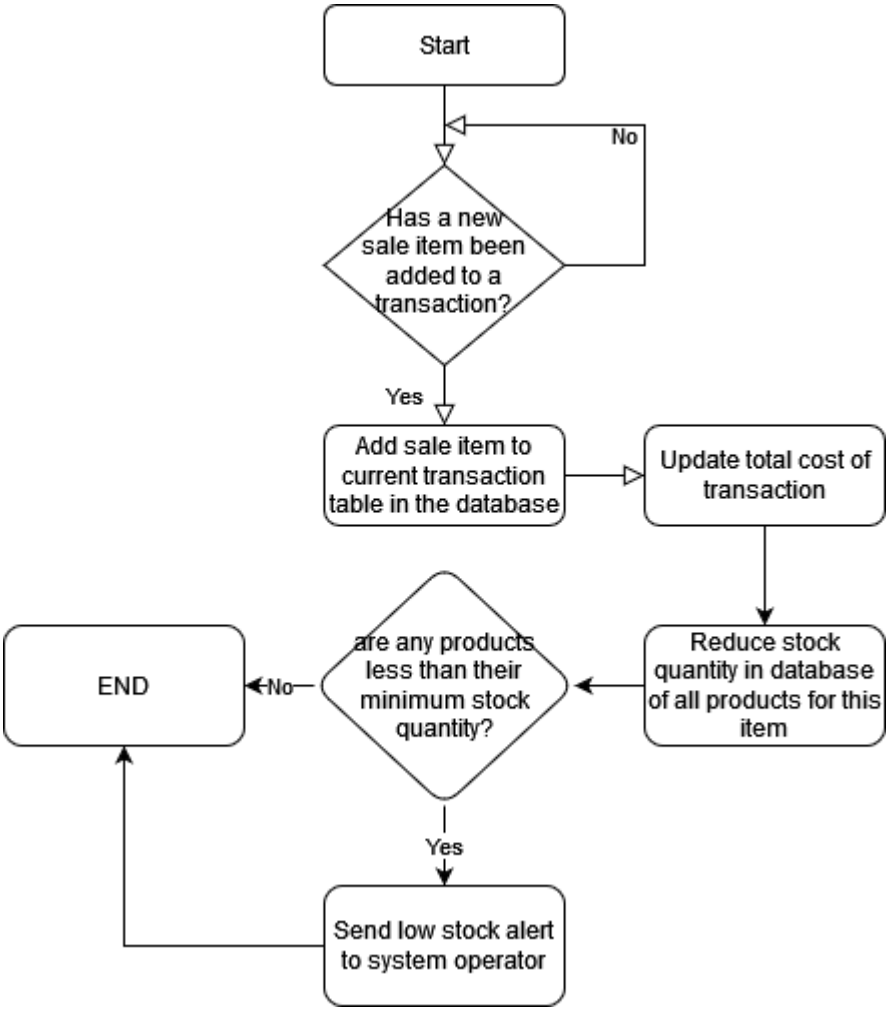
This stage of the project includes a menus database to permanently store the data for each of your menus, the basis of the point-of-sale system, and interactions with the stock management system. The point-of-sale system is a substantial section as it is the core of the entire project. This is the first stage that will directly interact with features implemented in the previous stages.

Features/functionality:

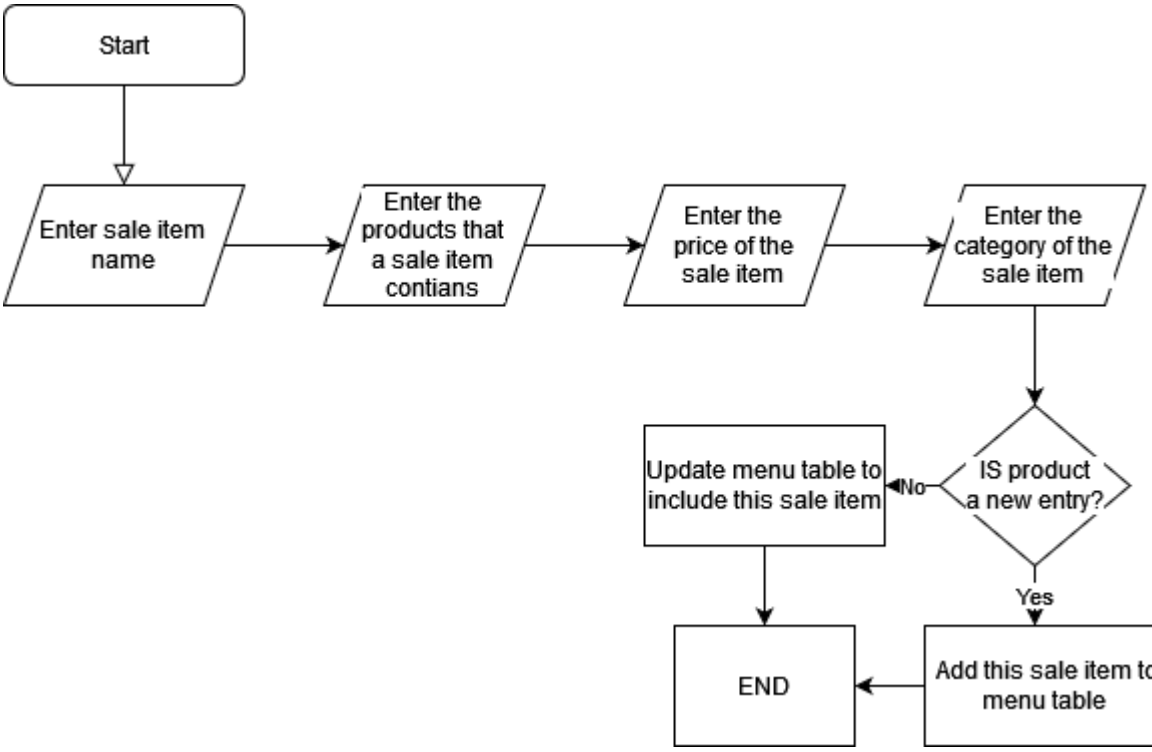
1. Clickable sale item buttons
2. A transaction column (left-hand side of GUI)
3. An algorithm to add sale items to the left-hand transaction column when a sale item button is pressed
4. An algorithm to increase the cost total when sale items are added to the transaction column
5. An algorithm to deduct the stock quantity of products from the stock management database table once the transaction has been marked as complete
6. The ability to mark a transaction as complete – this was not initially included at this stage as a feature, although was referred to in the feature above. However, I do not believe it was made clear enough that this is a feature that needs to be implemented at this stage.
7. A sale item button edit menu
8. An algorithm to open the edit menu for a specific sale item button after the edit button has been pressed, followed by the sale item button that you would like to edit
9. An algorithm to empty the transaction column once the close transaction button has been pressed
10. A low stock alert box *Implemented in a previous stage of the project. However, this still needs to be triggered when a product becomes low on stock

At the end of this stage, stakeholders should be presented with what has been achieved so far. They should confirm whether the base of the program has been developed as intended. If any features are found not to work as intended, they will be changed, and the stakeholders will be reconsulted after the change.

Test No.	Description	Type Of Test	Test Data	Expected Result	Actual Result
1a	Does the "Menus" database store the data for each sale item button in a menu?		(Press Create sale item button) Cheese sandwich Food Cheese,bread,bread,butter 3.00	Test data displays correctly in the menus database	
3b	Does pressing a sale item button add that item to the current transaction?		Press any sale item button	Sale item pressed is added to the current transaction	
3c	Is the total cost of a transaction is increased after a sale item is added to the transaction?		Press any sale item button	Total cost increases by the correct amount	
3d	Does pressing the "close transaction" button close the current transaction?		Press the "close transaction" button	Current transaction closes	
4a	Does pressing a sale item button decrease the product stock level from the stock management system?		Press any sale item button	Correct product quantities are deducted from the stock management database table	

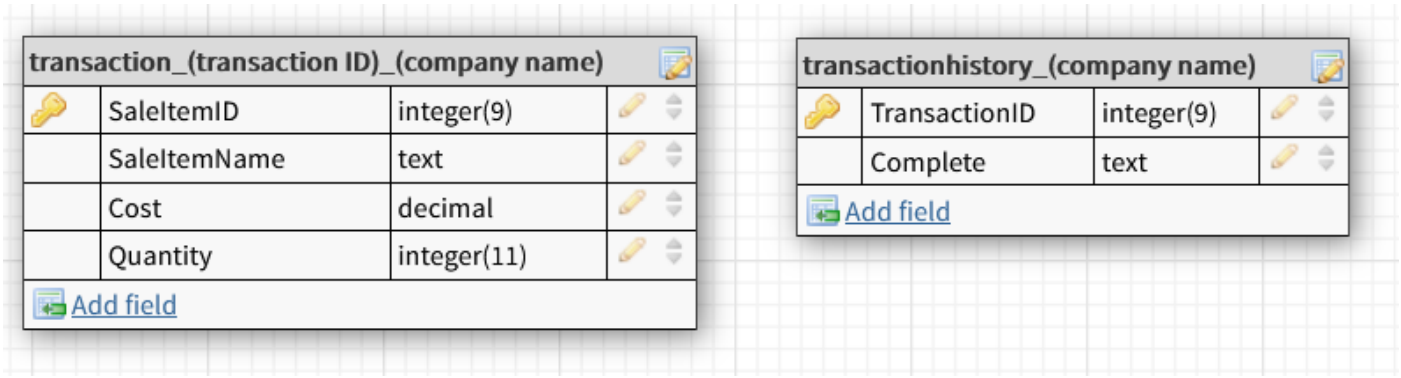


This shows the process that should be taken once a sale item has been added to a transaction. The program should continuously check whether a new sale item has been added to a transaction. If not, it should continue checking. If a sale item has been added, the sale item has to be added to the current transaction in the database, and the total cost of the current transaction must be increased. The products contained within the sale item must then be deduced from the total product quantity in the stock management system. It is then checked if a product has gone below its minimum stock quantity. If it has, a low stock alert is sent to the system operator then the program ends. Otherwise, the program just ends.

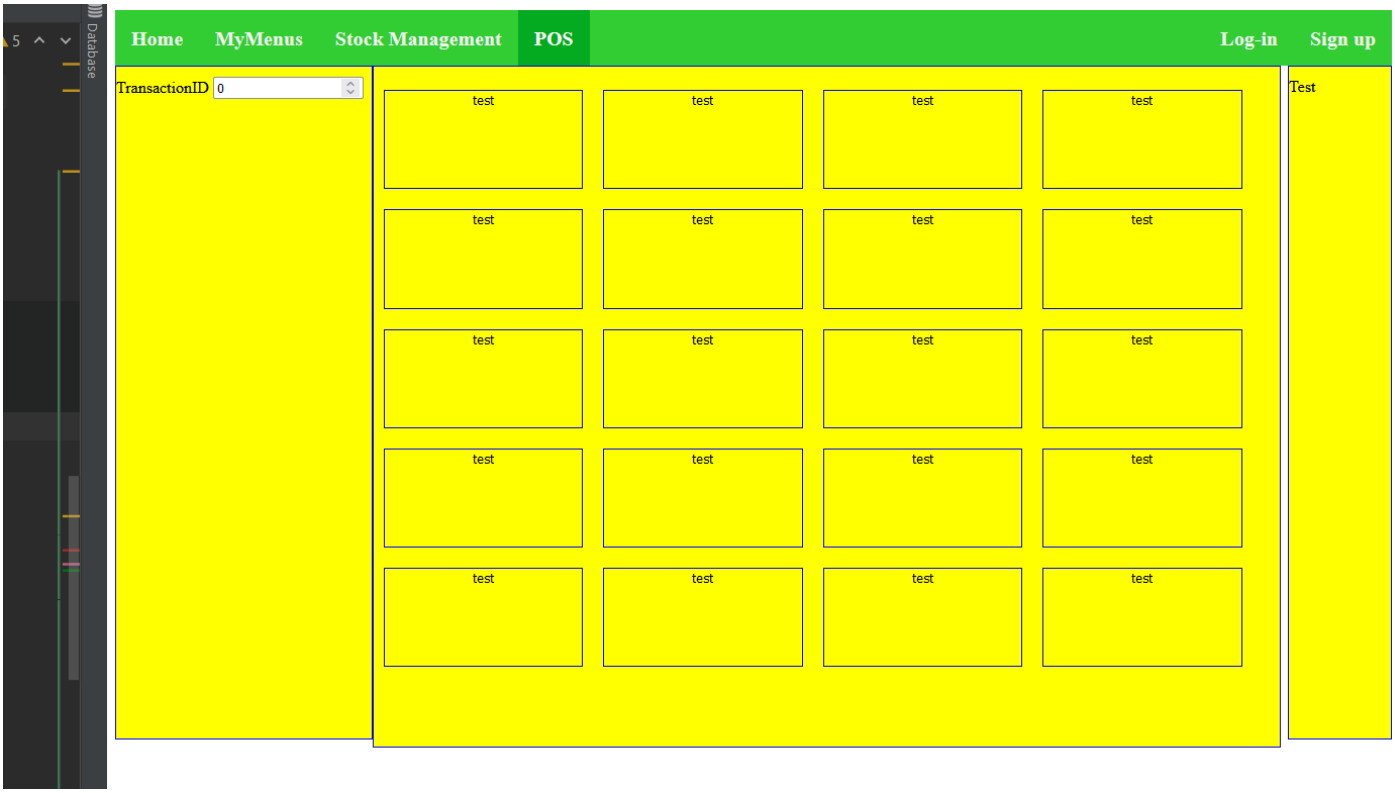


This shows the process of creating/editing a new sale item button, and the different processes required depending on whether the sale item is a new sale item, and hence a row needs to be created for it in the database table. Or if it is an existing entry, and the row it is contained within needs updating. First the user must input the sale item name, its containing products seperated by a comma, the price of the sale item, and the category of the sale item. If the product is a new entry, a record is added to the sale item menu table. Otherwise, the record of the existing sale item is updated.

Database table design for this stage:



User Interface of the point-of-sale page design process




```
<style>
    span.leftGUI{
        display: inline-block;
        float: left;
        width: 20%;
        height: 655px;
        padding-top: 10px;
        border: 1px solid blue;
        background-color: yellow;
    }

    span.saleItem{
        display: inline-grid;
        width: 22%;
        height: 15%;
        border: 1px solid blue;
        background-color: yellow;
        margin: 10px;
    }

    button.newSaleItem{
        display: inline-grid;
        width: 22%;
        height: 15%;
        border: 1px solid blue;
        background-color: lime;
        margin: 10px;
    }
```

```
span.midGUI{
    margin: auto;
    display: inline-block;
    width: 71%;
    height: 655px;
    padding-bottom: 5px;
    padding-top: 13px;
    padding-left 10px;
    border: 1px solid blue;
    background-color: yellow;
    font-size: 18px; /* Change to zero, testing with above 0 */
}

span.rightGUI{
    display: inline-block;
    width: 8%;
    height: 655px;
    padding-top: 10px;
    float:right;
    border: 1px solid blue;
    background-color: yellow;
}

button.saleItem{
    display: inline-grid;
    width: 22%;
    height: 15%;
    border: 1px solid blue;
    background-color: yellow;
    margin: 10px;
}
```

Creating database tables

To be able to process each transaction and store historical transaction data, a table will be needed in the database for each company containing the transaction ID of all transactions and whether the transaction is complete. This table will be created during sign-up if the company is not already signed-up to POSSY. This will also help with the

development of the project in the future, where the aim is to have the ability to open multiple transactions at one time.

```
/* Create A Transaction History Table For This Company */
$query = "CREATE TABLE `users`.`transactionhistory_$companyName` (
          `TransactionID` INT(9) NOT NULL AUTO_INCREMENT , `Complete` TEXT NOT NULL DEFAULT 'no' ,
          PRIMARY KEY (`TransactionID`))";
mysqli_query($connection,$query);
```

Each transaction will also need its own table to store the contents of the transaction. A new table will be created after each successful transaction. The initial table will be created when a new company signs up to POSSY. The entry will also be placed into the transaction history table for this company.

```
/* Create Initial Transaction Table */
$query = "CREATE TABLE `users`.`transation_0_$companyName` ( `SaleItemID` INT(9) NOT NULL ,
         `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT NOT NULL ,
          PRIMARY KEY (`SaleItemID`))";
mysqli_query($connection,$query);
```

Displaying the current transaction ID to the system operator

I will need an SQL query to search for any transactions within the transaction history of a company that are not yet complete. Once an incomplete transaction is found, this is set to the current transaction (later referred to as the default transaction) and is displayed in the left-hand column of the user interface.

```
<div>
    <span class='leftGUI'>
        <?php
            session_start();
            $companyName = $_SESSION['companyName'];

            /* Open Database Connection */
            include "database_connect.php";
            $connection = openConnection();

            $query = "SELECT TransactionID FROM transactionhistory_$companyName WHERE Complete = 'no'";

            $result = mysqli_query($connection,$query);
            $row = mysqli_fetch_assoc($result);

            $transactionID = $row['TransactionID'];
            echo 'transaction ID:'.$transactionID;
        ?>
    </span>
```

Transaction ID: 1

Testing

During this stage, I had deleted all unnecessary tables created in the database from previous tests and decided to create a new account, log in, create a new menu and manually insert some items into it.

Test no	Description	Data	Expected Outcome	Outcome
3.1	Sign-up using the following account details:	User Pass Company1 Company1@email.com 12345678910	The user is signed-up to the system	The user is signed-up to the system
3.2	Log in to the pre-made account from the previous test	User Pass	The user is logged into the account.	The user is logged into the account.
3.3	Create a new menu under the company name 'Company2'	Company1Menu1	A new menu, ourFirstMenu is created	A new menu, ourFirstMenu is created
3.4	Open the newly created, ourFirstMenu	Company1Menu1	The pointOfSale.php page is opened	The pointOfSale.php page is opened

I manually inserted these five items into the menu:

ID	saleItemName	price	category
1	Banana	0	
2	Apple	1.2	
3	Chocolate	1.8	
4	Cheese	2.8	
5	Cookie	0.8	

I opened the point-of-sale page and found a couple of problems. The first problem is that the price inside the database is not automatically stored to 2 decimal places. This error is of low priority. Therefore, it will be fixed at the end of the project unless the priority of this issue changes. However, a much more significant error did appear after loading my new menu:

Fatal error: Uncaught TypeError: mysqli_num_rows(): Argument #1 ($result) must be of type mysqli_result, bool given in C:\xampp\htdocs\POSSY\pointOfSale.php:110 Stack trace: #0 C:\xampp\htdocs\POSSY\pointOfSale.php(110): mysqli_num_rows(false) #1 {main} thrown in **C:\xampp\htdocs\POSSY\pointOfSale.php** on line **110**

It appears that a MySQLi statement is not executing as intended, and therefore no sale item buttons are being displayed.

```
106        $tableName = "menu_".$companyName."_".$menuID;
107
108        $query = "SELECT saleItemName,saleItemID,price FROM $tableName";
109        $result = mysqli_query($connection, $query);
110        $numRows = mysqli_num_rows($result);
```

I had incorrectly used 'saleItemID' rather than 'ID'; hence there were no results in the database. I updated this as shown below.

```
$query = "SELECT saleItemName,ID,price FROM $tableName";
```

This same error occurred throughout the file, with variables using saleItemID rather than ID, throwing errors or displaying incorrect data. This was fixed by updating all cases of 'saleItemID' to 'ID'.

```
115                $saleItemID = $row['saleItemID'];
```

Processing a Clicked Sale Item Button

I will need to post data about the pressed button to a processing page. I had problems understanding how to process a clicked saleItemButton by updating the database in real-time. I figured out that if you make an input of type hidden and give it the value you would like, the values will be posted to the processing PHP document as if the user had input them. My implementation of this is shown below

```
"<form action = 'process_buttonClick_saleItemAdd.php'>
   <input type='hidden' name='transactionID' value='$transactionID'>
   <input type='hidden' name='saleItemName' value='$saleItemName'> <!-- type= hidden
   <input type='hidden' name='saleItemID' value='$saleItemID'>
   <input type='hidden' name='saleItemCost' value='$price'>
   <button class= 'saleItem' type='submit'>$saleItemName <br> $displayPrice</button>

 </form>";
```

localhost/POSSY/process_buttonClick_saleItemAdd.php?transactionID=1&saleItemName=Banana&saleItemID=1&saleItemCost=0

This caused a problem with the layout of my point-of-sale system. Sale item buttons are now being displayed only vertically rather than in a grid form as they were previously.

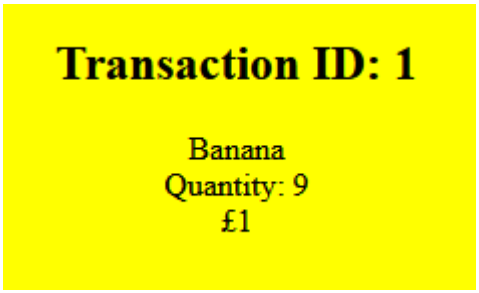
Although this affects the usability of my project, it is not essential to a working project, and therefore, this bug is a low priority and will not be fixed until later in my project unless it becomes of greater priority.

Now I need to process the click of the button to update the transaction interface. To do this, I need first to run a MySQLi query searching for items of the same name as the one pressed in the current transaction. If an item of the same name already exists in the transaction, the quantity of that item is increased using an SQL update query. Otherwise, the sale item's name, cost, and quantity are inserted into the table for this transaction. The user is then re-directed back to the point-of-sale page.

```
$query = "SELECT Quantity FROM $tableName WHERE SaleItemName = '$saleItemName'";
$result = mysqli_query($connection,$query);


If (mysqli_num_rows($result) == 0){
    $query = "INSERT INTO "."$tableName"."(SaleItemName,Cost,Quantity) VALUES('$saleItemName','$cost','1')";
    mysqli_query($connection,$query);

}
else{
    $row = mysqli_fetch_assoc($result);
    $quantity = $row['Quantity'];

    $newQuantity = $quantity + 1;
    $query = "UPDATE $tableName
              SET Quantity = '$newQuantity'
              WHERE SaleItemName = '$saleItemName' ";
    mysqli_query($connection,$query);

}
```

Adding to the Transaction Interface

I now need to add each sale item, its price, and quantity to the transaction column. I can do this by running a MySQLi query searching the current transaction for the sale item name, price, and quantity of all items in the transaction. I can then loop through this, inserting the values into the transaction interface on the screen.




```
//Display the current items on the transaction
$tableName = "transaction_".$transactionID."_".$companyName;
$query = "SELECT SaleItemName,Cost,Quantity FROM $tableName";
$result = mysqli_query($connection,$query);

for($i=0;$i<mysqli_num_rows($result);$i++){
    $row = mysqli_fetch_assoc($result);
    $saleItemName = $row['SaleItemName'];
    $saleItemCost = "£".$row['Cost'];
    $quantity = $row['Quantity'];

    echo "<p align='center'>$saleItemName <br> Quantity: $quantity <br> $saleItemCost</p>";
}
```

To be able to test that the transaction interface was working as intended, I needed to include the ability to add new and edit sale item buttons.

Initially, I had planned for system operators to enter an edit mode to edit sale item buttons. However, I quickly realised that I did not know how to do this, and it would be easier to use a separate page to do so as this is something I had experience with.

I first needed to create a web page with input boxes, allowing for the user to enter the details for a sale item button. The first input box is a drop-down box that allows users to select the sale item button they would like to edit. This drop-down should also allow users to select that they would like to create a new sale item button. Then, there should be input boxes allowing the user to input the name, price and category. Two MySQLi queries will be needed to either update or insert into the current menu table, depending on whether the user chose to create a new sale item or chose to edit an existing sale item.

```php
if($saleItemToEdit == '+new'){
    $menuTableName = $_SESSION['companyMenuTableName'];
    //SQL query creating new entry into the menu
    $query = "INSERT INTO $menuTableName(saleItemName,itemPrice,category)
              VALUES('$saleItemName','$itemPrice','$category')";
    mysqli_query($query);
}
else{
    //SQL query updating current entry in the menu
    $query = "UPDATE $menuTableName SET
              saleItemName = $saleItemName,
              itemPrice = $itemPrice,
              category = $category
              WHERE saleItemName = $saleItemToEdit";
}
```

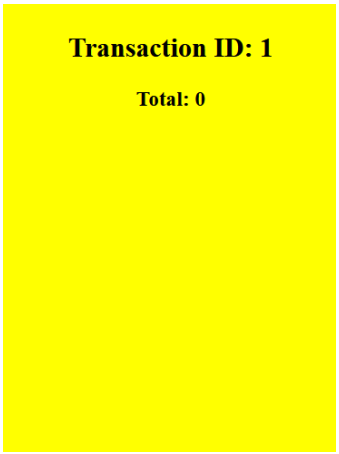
I kept coming across an error when adding a sale item to a transaction. The item would not be added to the transaction, despite the query working well. This was due to the transaction table for each company being misspelt upon creation:

```php
/* Create Initial Transaction Table */
$query = "CREATE TABLE `users`.`transation_1_$companyName` ( `SaleItemID` INT(9) NOT NULL ,
        `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT NOT NULL ,
         PRIMARY KEY (`SaleItemID`))";
mysqli_query($connection,$query);
```

This was fixed by changing ‘transaction’ to ‘transaction’ as was previously intended.

Calculating the total cost of a transaction

I will need to run an SQL query when opening the point-of-sale page that selects the cost and quantity from the current transaction table. I will then need to loop through these values, multiplying together the cost and quantity of each item and adding the value to a variable, $total. This value will then be output at the bottom of the transaction user interface.




```php
/* Output total cost of transaction */
$query = "SELECT Quantity, Cost FROM $tableName";
$result = mysqli_query($connection, $query);

$total = 0;
for($i=0;$i<mysqli_num_rows($result);$i++){
    $row = mysqli_fetch_assoc($result);
    $total += $row['Cost'] * $row['Quantity'];

}
echo "<h3 align='center'>Total: $total</h3>";
```

The layout for this is not quite as intended, as the Total is not displayed at the bottom of the page. However, I do not know how to fix this, and it is a low priority bug. Therefore, I will leave it until a later stage in my project to fix it.

I also realised that there is currently no way to complete a transaction, as mentioned in my list of features to be implemented in this stage.

Completing A Transaction

I will need to place a button that enables the user to complete a transaction at the bottom of the transaction interface. When this button is clicked, it will need to send the user to a processing page.

```
/*Complete Transaction Button */
echo "<form action='process_completeTransaction.php'>
      <button type='submit'>Complete</button>
      </form>";
```

The processing page will run an SQL query ensuring that there are a sufficient number of products to create the sale item, meaning that the sale item is still in stock. If the sale item is not in stock, the user will be returned to the point-of-sale system, and the current transaction will still be open. If the sale item is in stock, all of its containing products will be deducted from the company's stock management table, a new transaction will be created and the old transaction will be closed.

I began to write the initial SQL query, ensuring that the sale item is in stock when I realised I had not included the products of which a sale item contains. I, therefore, must complete this first before coming back to the ability to complete a transaction

Implementing products within a sale item

First, I need to create a table for each sale item with its containing products when each new sale item button is created. If a sale item button is edited, the table name should be renamed to reflect this.

I had not initially planned this, but I will need a further SQL query to add the id of the sale item to the end of the table name, then fetch the saleItemID from the returned results of the query:

```
$query = "SELECT ID FROM $menuTableName
          WHERE saleItemName = '$saleItemName' ";
$result = mysqli_query($connection,$query);

$row = mysqli_fetch_assoc($result);
$id = $row['ID'];

$tablename = "SaleItem_Contents_".$id;
```

Whilst coding this, I noticed that if the current company name was not included in the name of the table, I could end up with two tables of the same name with the same company using it, therefore overriding each other if the sale item had the same ID. To fix this, I have now included the company name in the name of the SaleItem_Contents table in the database.

```
$companyName = $_SESSION['companyName'];
```

```
    $tablename = "SaleItem_Contents_".$companyName."_".$id;
```

Now I will create a table for each sale item which stores its containing products when each new sale item button is created.

```
26      $tablename = "SaleItem_Contents_".$companyName."_".$id;
27
28      $query = "CREATE TABLE `users`.`$tablename` ( `ID` INT(9) NOT NULL AUTO_INCREMENT , `Product` TEXT NOT NULL ,
29              `Quantity` INT NOT NULL , PRIMARY KEY (`ID`))";
30      mysqli_query($connection,$query);
31
```

Ability to add products to a sale item when creating a new sale item

First, I will need to create a large input text box to enter the products the sale item contains.

Although this would be functional, I do not like how the text-area box is aligned with the bottom of the other input boxes rather than being aligned with the top of the other input boxes. To fix this, I will align the input boxes from the top using the CSS attribute vertical-align inside a div tag that wraps around all of the input boxes.

```
11
12  echo "<div vertical-align='top'><form action='process_editSaleItem.php'>";
13  echo "<label for='                </label>";
14  echo "<select name                t'>";
15
16  for ($i=0;$i<mysqli_num_rows($result);$i++){
17      $row = mysqli_fetch_assoc($result);
18      $saleItem = $row['SaleItemName'];
19          echo "<option value='$saleItem'>$saleItem</option>";
20  }
21  echo "<option value='+new'>New Sale Item</option>";
22  echo "</select>";
23
24  echo "<input type='text' name='saleItemName' placeholder='Banana'>";
25  echo "<input type='number' name='saleItemPrice' placeholder='0.00'>";
26  echo "<input type='text' name='saleItemCategory' placeholder='Food'>";
27  echo "<textarea name='products' placeholder='bread,bread,cheese'></textarea>";
28  echo "<button type='submit'>Confirm</button></div>";
```

Attribute vertical-align is not allowed here

I tried to do this using in-line CSS, but it cannot be used inline. To fix this, I will link this page to the style sheet and create a class .verticalAlign and assign the class to the div element.

```
/* Include Style Sheet */
echo "<link rel='stylesheet' href='Styles.css'>";
```

```
59  /* Vertically align text input boxes for edit sale item page */
60  .verticalAlign{
61      vertical-align: top;
62  }
```

```
15  /* Output all input boxes and their values */
16  echo "<div class='verticalAlign'>
17        <form action='process_editSaleItem.php'>";
18  echo "<label for='saleItemToEdit'>Choose a Sale Item: </label>";
19  echo "<select name='saleItemToEdit' id='saleItemToEdit'>";
20
21  for ($i=0;$i<mysqli_num_rows($result);$i++){
22      $row = mysqli_fetch_assoc($result);
23      $saleItem = $row['SaleItemName'];
24          echo "<option value='$saleItem'>$saleItem</option>";
25  }
26  echo "<option value='+new'>New Sale Item</option>";
27  echo "</select>";
28
29  echo "<input type='text' name='saleItemName' placeholder='Banana'>";
30  echo "<input type='number' name='saleItemPrice' placeholder='0.00'>";
31  echo "<input type='text' name='saleItemCategory' placeholder='Food'>";
32  echo "<textarea name='products' placeholder='bread,bread,cheese'></textarea>";
33  echo "<button type='submit'>Confirm</button>
34        </div>";
```

This did not work.

I do not know how to fix this, and this issue is cosmetic therefore is not important to the functions of my project. Therefore, it will be fixed at a later stage in my project only if time permits.

Processing the sale items into product tables

I will now need to insert each product that a sale item contains into the sale item table created previously. First, I will need to get the list of product names from the editSaleItem.php file.

```php
$productString = $_GET['products'];
```

I then need to separate each item name into an array, with each index containing the name of one item. This will be done by separating out any commas.

```php
function productStringToArray($productString){

    $productArray = []; //An array that stores the name of each product
    $tempString = ""; //A temporary string to store a product name when being initially formed,
                //to later be inserted into the $productArray
    $commaFound = true;
    for ($i=0; $i<strlen($productString); $i++){

        /* Search for a comma */
        if($productString[$i] == ","){
            $productArray[] = $tempString; //Append product into array
            $tempString = ""; //Empty string
        }
        else{
            $tempString = $tempString.$productString[$i]; //Append character to $productString
        }
    }
    $productArray[] = $tempString; //Append final product into array

    return $productArray;
```

```php
$productArray = productStringToArray($productString);
```

I now need to make sure that the first and last characters are not whitespace when a product is created. This will prevent bugs in the program.

```php
/* Removes whitespace before first character,
   and after the last character of a string. */
/* Returns the resulting string */
function removeWhitespaceBeforeAndAfterString($string){

    while($string[0] == " "){
        $string = substr($string, offset: 1);
    }
    while ($string[strlen($string)-1] == " "){
        $string = substr($string, offset: 1 - strlen($string));
    }
```

```php
/* If the product is not in the table, insert the product into the stock management database */
if (isProductInTable($connection, $tableName, $productName) == false) {

    removeWhiteSpaceBeforeAndAfterString($productName);

    if(isGreaterThanOrEqualToZero($minimumStockValue) AND isGreaterThanOrEqualToZero($currentStockValue)) {
        $query = "INSERT INTO $tableName(ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone)
          VALUES('$productName', '$minimumStockValue', '$currentStockValue', '$ordered',
          '$supplierName', '$phoneNumber')";
        mysqli_query($connection, $query);
        echo "Product Successfully Added";
    }
    else{
        echo "Value Smaller Than Zero Entered, INVALID!";
    }
}
else{
    echo "Product already exists!";
```

I will then need to enter some validation when product names are created, preventing commas from being entered into product names. This was not initially planned. However, this scenario came to mind when thinking about how this section of the program could break.

```php
function doesContainComma($string){

    $commaFound = false;
    for($i=0; $i<strlen($string); $i++){
        if($string[$i] == ","){
            $commaFound = true;
        }
    }

    return $commaFound;
}
```

```php
if(doesContainComma($productName) == false) {
    removeWhiteSpaceBeforeAndAfterString($productName);

    if (isGreaterThanOrEqualToZero($minimumStockValue) and isGreaterThanOrEqualToZero($currentStockValue)) {
        $query = "INSERT INTO $tableName(ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone
      VALUES('$productName', '$minimumStockValue', '$currentStockValue', '$ordered',
      '$supplierName', '$phoneNumber')";
        mysqli_query($connection, $query);
        echo "Product Successfully Added";
    } else {
        echo "Value Smaller Than Zero Entered, INVALID!";
    }
}
else{
    echo "Invalid character: Comma ',' in product name";
}
```

Now I will need an SQL query to insert each product into the sale item table of the database for that particular product. I will do this using a for loop, meaning that the query will be run multiple times.

When writing the SQL query, I noticed that the program would enable duplicate entries into the array in its current state.

```
/* Inserting products into SaleItem_Contents table for the particular sale item */
$sizeOfProductArray = count($productArray);

for($i=0; $i<$sizeOfProductArray; $i++){
    $query = "INSERT INTO $tablename(Product,Quantity) VALUES($productArray[$i], 1)";
}
```

To fix this, with each iteration of the for loop, I will need a second SQL query checking that the product is not already contained within the array, and if there are results to the query, then the quantity of that product should instead be increased, rather than adding the product again to the array.

```
/* Inserting products into SaleItem_Contents table for the particular sale item */
$sizeOfProductArray = count($productArray);

for($i=0; $i<$sizeOfProductArray; $i++){

    $query = "SELECT Product FROM $tablename WHERE Product = '$productArray[$i]'";
    $result = mysqli_query($connection,$query);
    $numResults = mysqli_num_rows($result); //Fetch the number of results the query produced

    /* Insert product into SaleItem_Contents table if product does not already exist in the table */
    if($numResults == 0) {
        $query = "INSERT INTO $tablename(Product,Quantity) VALUES($productArray[$i], 1)";
        mysqli_query($connection,$query);
    }
    /* Otherwise, update the quantity of such item */
    else{
        $query = "UPDATE $tablename SET(Quantity = 'Quantity + 1') WHERE Product = $productArray[$i]";
        mysqli_query($connection,$result);
    }
}
```

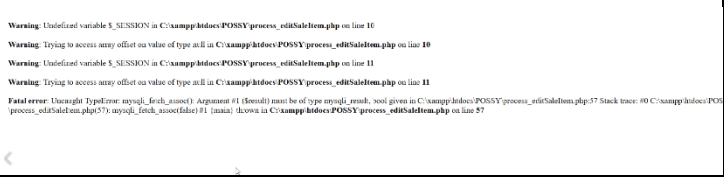
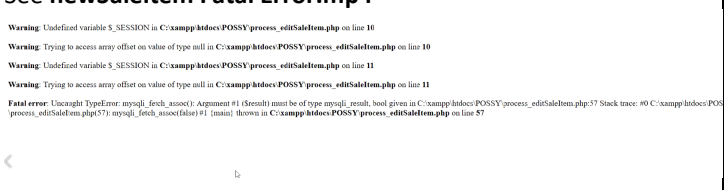
I then wrote the following test cases:

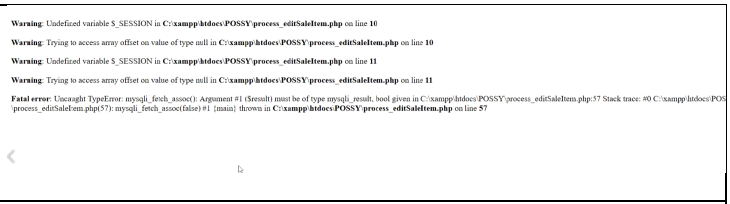
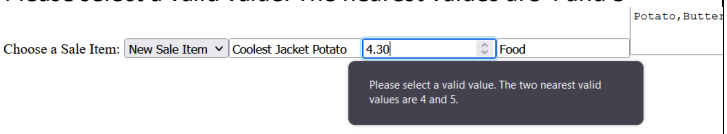
Test No.	Description	Test Data	Expected Result	Actual Result
1a	Create a new product	Potato 50 20 No Press done	A new product Is created and is visible in the stock management table for that particular company	
1b	Create another new product	Butter 100 10 No Press done	A new product Is created.	
2a	Create a new sale item	Select New Sale Item Jacket Potato 4.00 Food Potato,Butter	“Potato” and “Butter” appear as products of the sale item “Jacket Potato”	
2b	Create a new sale item, but put spaces between the contents	Select New Sale Item Cool Jacket Potato 4.00 Food Potato, Butter	“Potato” and “Butter” appear as products of the sale item “Jacket Potato”	
2c	Create a new sale item, but put spaces on either side of the contents	Select New Sale Item Cool Jacket Potato 4.00 Food Potato , Butter	“Potato” and “Butter” appear as products of the sale item “Cool Jacket Potato”	
2d	Create a new sale item, but contain multiple of the same product	Select New Sale Item Coolest Jacket Potato 4.30 Food Potato,Butter,Butter	“Potato” and “Butter” appear as products of the sale item “Coolest Jacket Potato” and butter has a quantity of 2	

When writing these tests, I was viewing my code, trying to find more tests that could potentially break my code when I noticed that I had not removed the whitespace around commas when inputting products into the contents of a sale item; this had only been implemented for new products. I, therefore, edited this code, introducing the following to remove the whitespaces around commas to fix the issue:

```
function productStringToArray($productString){

    $productArray = []; //An array that stores the name of each product
    $tempString = ""; //A temporary string to store a product name when being initially formed,
                      //to later be inserted into the $productArray

    for ($i=0; $i<strlen($productString); $i++){

        /* Search for a comma */
        if($productString[$i] == ","){

            /* If there is a space before the next comma, remove it from the tempString */
            while($tempString[strlen($tempString)-1] == " "){
                $tempString = substr($tempString, offset: 0, length: strlen($tempString)-1);
            }

            $productArray[] = $tempString; //Append product into array
            $tempString = ""; //Empty string
        }
        /* Do Nothing if a space is found after a comma */
        else if($productString[$i] == " " and $productString[$i - 1] == ","){
            //do nothing - BAD PRACTICE
        } //NEW
        else{

            $tempString = $tempString.$productString[$i]; //Append character to $productString
        }
```

I then ran the intended tests, plus a few extras:

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Create a new product	Potato 50 20 No Press done	A new product Is created and is visible in the stock management table for that particular company	A new product Is created and is visible in the stock management table for that particular company See **newProductPageNotResponding.mp4**
1b	Create another new product	Butter 100 10 No Press done	A new product Is created.	A new product Is created See **newProductPageNotResponding.mp4**
1c	Create another new product, but include a comma within the product name	Che,ese 100 10 No Press done	Invalid product name – includes character comma (,) Product is not created	Invalid character: Comma ‘,’ in product name. Product was not created See **newProductPageNotResponding.mp4**
1d	Create another new product, but put spaces on either side of the product name	“ Chicken “ 100 10 No Press done (do not include quotations , they are used to clearly show the spaces either side of the product name)	A new product “Chicken” Is created.	The web page stopped responding See **newProductPageNotResponding.mp4** Error produced: **Fatal error**: Maximum execution time of 120 seconds exceeded in **C:\xampp\htdocs\POSSY\process_newProduct.php** on line **53**
2a	Create a new sale item	Select New Sale Item Jacket Potato 4.00 Food Potato,Butter	“Potato” and “Butter” appear as products of the sale item “Jacket Potato”	Produces a fatal error. See **newSaleItem Fatal Error.mp4** 
2b	Create a new sale item, but put spaces between the content	Select New Sale Item Cool Jacket Potato 4.00 Food Potato, Butter	“Potato” and “Butter” appear as products of the sale item “Jacket Potato”	Produces a fatal error. See **newSaleItem Fatal Error.mp4** 
2c	Create a new sale	Select New Sale Item	“Potato” and “Butter” appear as	Produces a fatal error. See **newSaleItem Fatal Error.mp4**

	item, but put spaces on either side of the content	Cool Jacket Potato 4.00 Food Potato , Butter	products of the sale item “Cool Jacket Potato”	
2d	Create a new sale item, but contain multiple of the same product	Select New Sale Item Coolest Jacket Potato 4.30 Food Potato,Butter,Butter	“Potato” and “Butter” appear as products of the sale item “Coolest Jacket Potato”, and butter has a quantity of 2	Please select a valid value. The nearest values are 4 and 5  See **newSaleItem Fatal Error.mp4**

These tests collectively produced many errors, leading to the failure of tests 1d, 2a, 2b, 2c, and 2d.

Addressing Test 1d Failure

Test 1d failed as the web page stopped responding. The fatal error pointed to line 53 of the process_newProduct.php page, which is inside the body of a while loop.

```
47  function removeWhitespaceBeforeAndAfterString($string){
48
49      while($string[0] == " "){
50          $string = substr($string, offset: 1);
51      }
52      while ($string[strlen($string)-1] == " "){
53          $string = substr($string, offset: 1 - strlen($string));
54      }
55
56      return $string;
57  }
```

I suspect that the while loop never met its condition, hence the web page not responding. Line 53 does not do what was intended. It is intended that the value of $string is updated to remove the last character of the string. To fix this, I will get the substring if string, with offset 0 but with length strlen($string) -1. This will mean that the while statement will no longer be infinite.

```
52      while ($string[strlen($string)-1] == " "){
53          $string = substr($string, offset: 0, length: strlen($string) - 1);
54      }
```

Addressing Test 2a, 2b, 2c Failure

All 3 of these failed tests produced the same five errors:

Warning: Undefined variable $_SESSION in **C:\xampp\htdocs\POSSY\process_editSaleItem.php** on line **10**

Warning: Trying to access array offset on value of type null in **C:\xampp\htdocs\POSSY\process_editSaleItem.php** on line **10**

Warning: Undefined variable $_SESSION in **C:\xampp\htdocs\POSSY\process_editSaleItem.php** on line **11**

Warning: Trying to access array offset on value of type null in **C:\xampp\htdocs\POSSY\process_editSaleItem.php** on line **11**

Fatal error: Uncaught TypeError: mysqli_fetch_assoc(): Argument #1 ($result) must be of type mysqli_result, bool given in C:\xampp\htdocs\POSSY\process_editSaleItem.php:57 Stack trace: #0 C:\xampp\htdocs\POSSY\process_editSaleItem.php(57): mysqli_fetch_assoc(false) #1 {main} thrown in **C:\xampp\htdocs\POSSY\process_editSaleItem.php** on line **57**

The session was not started in this file before the $_SESSION variables were attempted to be used, hence causing all five errors. The first four warnings were caused when initializing variables to values stored inside $_SESSION variables.

```php
<?php
/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

/* Init Variables */
$saleItemToEdit = $_GET['saleItemToEdit'];
$saleItemName = $_GET['saleItemName'];
$itemPrice = $_GET['saleItemPrice'];
$category = $_GET['saleItemCategory'];
$menuTableName = $_SESSION['companyMenuTableName'];
$companyName = $_SESSION['companyName'];
$productString = $_GET['products'];
```

The fatal warning occurred when running a MySQLi query using these variables. Hence, the variables $menuTableName and $saleItemName were null, throwing a fatal error.

```php
    $query = "SELECT ID FROM $menuTableName
              WHERE saleItemName = '$saleItemName' ";
    $result = mysqli_query($connection,$query);
```

To fix this, I will need to start the session at the beginning of the file.

```php
/* Start the session */
session_start();

/* Init Variables */
```

Addressing Test 2d Failure

The result of this test was very unexpected – I was unable to submit the form when using a non-integer number. I suspect that the input field for cost only allows integers.

```
25    echo "<input type='number' name='saleItemPrice' placeholder='0.00'>";
```

To fix this, I will need to update the allowed type in this field to a type that accepts floating-point numbers. There is no float input type in HTML – instead, you need to add the increment value by including an attribute "step". Therefore, to fix this issue, I will set this attribute "step" = 0.01.

```
25    echo "<input type='number' name='saleItemPrice' placeholder='0.00' step='0.01'>";
```

I also noticed that it is currently possible to include negative numbers in the input field. To fix this, I will include a minimum value attribute of 0.00

```
25    echo "<input type='number' name='saleItemPrice' placeholder='0.00' step='0.01' min='0.00'>";
```

Test re-run

I then cleared the stock management table and re-ran the tests:

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Create a new product	+New Product Potato 50 20 No Press done	A new product Is created and is visible in the stock management table for that particular company	A new product Is created and is visible in the stock management table for that particular company See **stage 5 test re-run.mp4**
1b	Create another new product	+New Product Butter 100 10 No Press done	A new product Is created.	A new product Is created. See **stage 5 test re-run.mp4**
1c	Create another new product, but include a comma within the product name	+New Product Che,ese 100 10 No Press done	Invalid product name – includes character comma (,) Product is not created	Invalid product name – includes character comma (,) Product is not created See **stage 5 test re-run.mp4**
1d	Create another new product, but put spaces on either side of the product name	+New Product " Chicken " 100 10 No Press done (do not include quotations, they are used to clearly show the spaces either side of the product name)	A new product "Chicken" Is created.	A new product " Chicken " Is created. The spaces were not removed. See **stage 5 test re-run.mp4**

1e	Re-create an existing product	+New Product Butter 100 10 No Press done	The product cannot be created – it is a duplicate	Product Already exists! See **stage 5 test re-run.mp4**
2a	Create a new sale item	Select New Sale Item Jacket Potato 4.00 Food Potato,Butter	"Potato" and "Butter" appear as products of the sale item "Jacket Potato"	A new sale item was not added to the menu table of the database See **stage 5 test re-run.mp4**
2b	Create a new sale item, but put spaces between the content	Select New Sale Item Cool Jacket Potato 4.00 Food Potato, Butter	"Potato" and "Butter" appear as products of the sale item "Jacket Potato"	A new sale item was not added to the menu table of the database See **stage 5 test re-run.mp4**
2c	Create a new sale item, but put spaces on either side of the content	Select New Sale Item Cooler Jacket Potato 4.00 Food Potato , Butter	"Potato" and "Butter" appear as products of the sale item "Cool Jacket Potato"	A new sale item was not added to the menu table of the database See **stage 5 test re-run.mp4**
2d	Create a new sale item, but contain multiple of the same product	Select New Sale Item Coolest Jacket Potato 4.30 Food Potato,Butter,Butter	"Potato" and "Butter" appear as products of the sale item "Coolest Jacket Potato" and butter has a quantity of 2	A new sale item was not added to the menu table of the database See **stage 5 test re-run.mp4**
2e	Re-create an existing sale item	Select New Sale Item Jacket Potato 4.00 Food Potato,Butter	This sale item already exists!	A new sale item was not added to the menu table of the database
2f	Edit an existing sale item	Select Jacket Potato Jacket Potato 4.10 Food Potato,Butter	The price of Jacket Potato is updated to 4.10	**I was unable to run this test due to a prior test failure**

This test was not as successful as I had hoped, all failed tests are addressed below. I was unable to run test 2f as the test used to create the existing sale item failed in a previous part of the testing phase, and therefore there was no product to edit.

Test 1d failure

Test 1d failed as the spaces on either side of the product name " chicken " were not removed. The function to remove the whitespace ran, but the new string returned from the function was not stored inside a variable.

```
76          removeWhiteSpaceBeforeAndAfterString($productName);
```

To fix this, I will make the variable $productName equal to the function on this line.

```
76          $productName = removeWhiteSpaceBeforeAndAfterString($productName);
```

Test 2a, 2b, 2c, 2e, 2d failure

The failures to these tests were all linked. The sale item was not added to the menu table of the database in all occasions. All failures produced the following error:

Warning: Trying to access array offset on value of type null in **C:\xampp\htdocs\POSSY\process_editSaleItem.php** on line **64**

```
51  if($saleItemToEdit == '+new'){
52      //SQL query creating new entry into the menu
53      $query = "INSERT INTO $menuTableName(saleItemName,itemPrice,category)
54                VALUES('$saleItemName','$itemPrice','$category')";
55      mysqli_query($connection,$query);
56
57      //SQL query selecting the ID of the newly created sale item
58      //from the company's menu table
59      $query = "SELECT ID FROM $menuTableName
60                WHERE saleItemName = '$saleItemName' ";
61      $result = mysqli_query($connection,$query);
62
63      $row = mysqli_fetch_assoc($result);
64      $id = $row['ID'];
65
66      $tablename = "SaleItem_Contents_".$companyName."_".$id;
```

The error suggests that the SQL query ran on line 61 produces no results. Hence, line 64 is trying to access an array index that does not exist. This means that there must be an issue with the query on line 53, in which the sale item is not being inserted into the menu table in the database. Here is the database in question:

ID	saleItemName	price	category

As you can see, the column name inside the query is different to that in the database. Therefore, to fix this issue, I will need to change the column name in the query from "itemPrice" to "price".

Test re-run 2

I will now clear the database tables and re-run the tests:

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Create a new product	Potato 50 20 No Press done	A new product Is created and is visible in the stock management table for that particular company	A new product Is created and is visible in the stock management table for that particular company See **stage 5 test re-run (2).mp4**
1b	Create another new product	Butter 100 10 No Press done	A new product Is created.	A new product Is created. See **stage 5 test re-run (2).mp4**
1c	Create another new product, but include a comma within the product name	Che,ese 100 10 No Press done	Invalid product name – includes character comma (,) Product is not created	Invalid product name – includes character comma (,) Product is not created See **stage 5 test re-run (2).mp4**
1d	Create another new product, but put spaces on either side of the product name	+New Product “ Chicken “ 100 10 No Press done (do not include quotations, they are used to clearly show the spaces on either side of the product name)	A new product “Chicken” Is created.	A new product “Chicken” Is created. See **stage 5 test re-run (2).mp4**
1e	Re-create an existing product	Butter 100 10 No Press done	The product cannot be created – it is a duplicate	Product already exists! See **stage 5 test re-run (2).mp4**
2a	Create a new sale item	Select New Sale Item Jacket Potato 4.00 Food Potato,Butter	“Potato” and “Butter” appear as products of the sale item “Jacket Potato”	No table exists for the contents of this sale item See **stage 5 test re-run (2).mp4**
2b	Create a new sale item, but put spaces between the content	Select New Sale Item Cool Jacket Potato 4.00 Food Potato, Butter	“Potato” and “Butter” appear as products of the sale item “Jacket Potato”	No table exists for the contents of this sale item See **stage 5 test re-run (2).mp4**
2c	Create a new sale	Select New Sale Item	“Potato” and “Butter” appear as products of	No table exists for the contents of this sale item

	item, but put spaces on either side of the content	Cooler Jacket Potato 4.00 Food Potato , Butter	the sale item “Cool Jacket Potato”	See **stage 5 test re-run (2).mp4**
2d	Create a new sale item, but contain multiple of the same product	Select New Sale Item Coolest Jacket Potato 4.30 Food Potato,Butter,Butter	“Potato” and “Butter” appear as products of the sale item “Coolest Jacket Potato” and butter has a quantity of 2	No table exists for the contents of this sale item See **stage 5 test re-run (2).mp4**
2e	Re-create an existing sale item	Select New Sale Item Jacket Potato 4.00 Food Potato,Butter	This sale item already exists!	No table exists for the contents of this sale item See **stage 5 test re-run (2).mp4**
2f	Edit an existing sale item	Select Jacket Potato Jacket Potato 4.10 Food Potato,Butter	The price of the Jacket Potato is updated to 4.10	No table exists for the contents of this sale item See **stage 5 test re-run (2).mp4**

Although test 1d passed this round of testing, tests 2a, 2b, 2c, 2d, and 2f all failed. I also noticed that the number of decimal places of price is not stored correctly in the database, as seen in **stage 5 test re-run (2).mp4**

Test 2a, 2b, 2c, 2d, 2e failure

There was no correctly named table created in the database storing the products of the sale item. The following is the only existing table:

This table should have the ID of the sale item at the end of the table name; however, no errors were produced by the program.

Cost not being recorded to 2 decimal places in the database table

Within each menu, the price of each item is not stored to 2 decimal places as it should be

ID	saleItemName	price	category
8	Jacket Potato	4	Food
9	Cool Jacket Potato	4	Food
10	Cooler Jacket Potato	4	Food
11	Coolest Jacket Potato	4.3	Food

I can fix this by only validating the input if there are two digits after the decimal place.

```php
/* Checks if a floating point number is to 2 decimal places */
/* Returns true if number is to 2 decimal places */
/* Returns false if number is not to 2 decimal places */
function isTwoDecimalPlaces($value){

    /* Calculate the length of $value */
    $value = (string) $value; //cast $value to string
    $lengthValue = strlen($value);

    /* loop through $stringValue until the decimal point is found */
    /* Store the number of decimal places into variable $numberOfPlacesAfterDecimal */
    $decimalPointFound = false;
    $numberOfPlacesAfterDecimal = 0;
    for($i=0; $i<$lengthValue; $i++){
        if ($value[$i] == "."){
            $decimalPointFound = true;
        }
        else if($decimalPointFound == true){
            $numberOfPlacesAfterDecimal += 1;
        }
    }

    /* Return false if number of decimals is not 2 */
    if($numberOfPlacesAfterDecimal != 2){
        return false;
    }
    else{
        return true;
    }
}
```

```php
if($saleItemToEdit == '+new' AND isTwoDecimalPlaces($itemPrice) == true){

    /* SQL query creating new entry into the menu */
    $query = "INSERT INTO $menuTableName(saleItemName,price,category)
              VALUES('$saleItemName','$itemPrice','$category')";
    mysqli_query($connection,$query);

else if(isTwoDecimalPlaces($itemPrice) == false){
    echo "Incorrect number of decimal places";
}
```

I then ran the following test:

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Create a new sale item and check if it is stored to two decimal places	Select New Sale Item Mash Potato 4.00 Food Potato,Butter	“Potato” and “Butter” appear as products of the sale item “Mash Potato”	A new Item is created but the price is not stored to 2 decimal places.

The proposed solution did not work. This solution will stay implemented, as I believe it is still a necessary form of validation. However, it did not solve the intended problem. To fix the problem, I need to change the type of price in the SQL table from double to decimal(10,2). To do this, I will change the query when each menu table is created.

Before:

```
23  $query = "CREATE TABLE `users`.`$menuName` ( `ID` INT(9) NOT NULL AUTO_INCREMENT ,
24            `saleItemName` TEXT NOT NULL , `price` DOUBLE NOT NULL , `category` TEXT NOT NULL ,
25            PRIMARY KEY (`ID`))";
26  mysqli_query($connection, $query);
```

After:

```
23  $query = "CREATE TABLE `users`.`$menuName` ( `ID` INT(9) NOT NULL AUTO_INCREMENT ,
24            `saleItemName` TEXT NOT NULL , `price` DECIMAL(10,2) NOT NULL , `category` TEXT NOT NULL ,
25            PRIMARY KEY (`ID`))";
26  mysqli_query($connection, $query);
```

I then ran the following test:

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Create a new sale item and check if it is stored to two decimal places	Select New Sale Item Mash Potato 4.00 Food Potato,Butter	Price (4.00) is displayed in the SQL table to 2 decimal places	Price (4.00) is displayed in the SQL table See **sale item price type update.mp4**

Completing A Transaction

I will first need to pass the transaction ID to the processing file once the complete transaction button has been pressed in the point-of-sale system. I will do this using hidden input fields.

```
132             /*Complete Transaction Button */
133             echo "<form action='process_completeTransaction.php'>
134                   <button type='submit'>Complete</button>
135                   <input type='hidden' name='transactionID' value='$transactionID'>
136                   </form>";
137
```

I will now need to retrieve the transactionID in the processing file and run a MySQLi query, selecting everything from the table for the current transaction.

```
8   // Initiate variables
9   $transactionID = $_GET("transactionID");
10  $companyName = $_SESSION["companyName"];
11  $tableName = "transaction_".$transactionID."_".$companyName;
12
13  // Select everything contained within the transaction table
14  $query = "SELECT * FROM $tableName";
15  $result = mysqli_query($connection,$query);
16
```

I now need to loop through each sale item. Within this loop, I need to loop through the products that the particular sale item contains, check there are sufficient amounts of that product, and deduct stock from the stock management system. If a product is not in stock or there is insufficient stock for a product, the transaction cannot be complete. Therefore, the stock level will be re-instated to its state prior to the complete transaction button being pressed.

```php
/*
 * This function returns false if an item is out of stock,
 * and true if a;; saleItems are in stock
 * $DBconnection stores the connection to the SQL database
 * $companyName is the name of the company of the current user
 * $decuctFromStock is a boolean value to show whether or not you want stock levels to be decreased, where:
 * true = you would like the stock level to be decreased
 * false = you would not like the stock level to be decreased
 */
function updateStockManagementSystem($DBconnection,$companyName){
    // initiate variables
    $tableName = "transaction_" . $_GET["transactionID"] . "_" . $_SESSION['companyName'];
    $connection = $DBconnection;
    $returnValue = true;

    // Select everything contained within the transaction table
    $query = "SELECT * FROM $tableName";
    $result = mysqli_query($connection, $query);

    $numRows = mysqli_num_rows($result); //returns the number of rows in the table

    // Loop through sale items
    for ($i = 0; $i < $numRows; $i++) {
        // fetch a new row from the SQL result
        $row = mysqli_fetch_assoc($result);

        // Store the table name for the contents of the current sale item
        $saleItemContentsTableName = "SaleItem_Contents_" . $companyName . "_" . $row['ID'];

        // Select all rows for the contents of the current sale item
        $query = "SELECT * FROM $saleItemContentsTableName";
        $saleItemContentsResult = mysqli_query($connection, $query);

        // Loop through each product of a sale item
        for ($k = 0; $k < mysqli_num_rows($saleItemContentsResult); $i++) {
            $saleItemContentsRow = mysqli_fetch_assoc($saleItemContentsResult);

            // Select product name and current stock value from the stockmanagement table for the current product.
            // Stores result in $productQueryResult
            $stockManagementTableName = "stockmanagementtable_$companyName";
            $productName = $saleItemContentsRow['Product'];
            $query = "SELECT ProductName,CurrentStockValue FROM $stockManagementTableName
                      WHERE ProductName == '$productName'";
            $productQueryResult = mysqli_query($connection,$query);

            // Produce an error if more than one result is returned
            // Produce an error if no results are returned
            if(mysqli_num_rows($productQueryResult) == 1) {

                // Checks if the stock management system has a great enough stock quantity of the product
                if ($saleItemContentsRow['Quantity'] - $productQueryResult['CurrentStockValue'] < 0) {

                    $returnValue = false;
                }
```

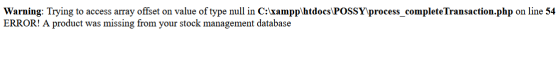
```
66
67                  // Deducts from the stock management system
68                  $productQuantity = $saleItemContentsRow['Quantity'];
69                  $query = "UPDATE $stockManagementTableName
70                            SET CurrentStockValue = 'CurrentStockValue - $productQuantity'
71                            WHERE ProductName = '$productName'";
72                  mysqli_query($connection,$query);
73
74              }
75              else if(mysqli_num_rows($productQueryResult) == 0){
76                  echo "ERROR! A product was missing from your stock management database";
77              }
78              else{
79                  echo "ERROR! Duplicated product names! Remove duplicates from your stock management database";
80              }
81
82          }
83
84      }
85
86      /*
87       *  If a product is not in stock, all products quantities should be added back to the stock management
88       *  database, as the transaction will not be completed.
89       *
90       * This section could be severely simplified, probably by using a while loop and a selection statement
91       * in the previous section
92       */
93      if ($returnValue == false){
94          // Loop through sale items
95          for ($i = 0; $i < $numRows; $i++) {
96              // fetch a new row from the SQL result
97              $row = mysqli_fetch_assoc($result);
98
99              // Store the table name for the contents of the current sale item
100             $saleItemContentsTableName = "SaleItem_Contents_" . $companyName . "_" . $row['ID'];
101
102             // Select all rows for the contents of the current sale item
103             $query = "SELECT * FROM $saleItemContentsTableName";
104             $saleItemContentsResult = mysqli_query($connection, $query);
105
106             // Loop through each product of a sale item
107             for ($k = 0; $k < mysqli_num_rows($saleItemContentsResult); $i++) {
108                 $saleItemContentsRow = mysqli_fetch_assoc($saleItemContentsResult);
109
110                 // Select product name and current stock value from the stockmanagement table for the current product.
111                 // Stores result in $productQueryResult
112                 $stockManagementTableName = "stockmanagementtable_$companyName";
113                 $productName = $saleItemContentsRow['Product'];
114                 $query = "SELECT ProductName,CurrentStockValue FROM $stockManagementTableName
115                           WHERE ProductName == '$productName'";
116                 $productQueryResult = mysqli_query($connection,$query);
117
118                 // Produce an error if more than one result is returned
119                 // Produce an error if no results are returned
120                 if(mysqli_num_rows($productQueryResult) == 1) {
121
122                     // Increases quantities in the stock management system
123                     $productQuantity = $saleItemContentsRow['Quantity'];
124                     $query = "UPDATE $stockManagementTableName
```

```
                        SET CurrentStockValue = 'CurrentStockValue + $productQuantity'
                        WHERE ProductName = '$productName'";
                    mysqli_query($connection,$query);

                }
                else if(mysqli_num_rows($productQueryResult) == 0){
                    echo "ERROR! A product was missing from your stock management database";
                }
                else{
                    echo "ERROR! Duplicated product names! Remove duplicates from your stock management database";
                }

            }

        }
    }


    return $returnValue;
}
```

I now need to end the transaction and automatically open a new transaction. To do this, I need to update the transaction history table in the database, marking the current transaction as complete. Then I will need to create a new transaction table in the database.

```
    if($returnValue == true){
        /* Set current transaction to complete */
        $tableName = "transactionhistory_".$companyName;
        $query = "UPDATE $tableName SET Complete = 'yes' WHERE TransactionID = '$transactionID'";
        mysqli_query($connection,$query);

        /* Open a new transaction */
        $query = "INSERT INTO transactionhistory_$companyName(Complete) VALUES('no')";
        mysqli_query($connection,$query);
    }

    return $returnValue;
}
```

I will now run the following tests to ensure this section of the project is working as intended.

Test No.	Description	Test Data	Expected Result	Actual Result
1d	Add the sale item “Cooked Potato” to the transaction	Press on the “Cooked Potato” button in the point of sale system	The sale item is added to the transaction menu	The sale item is added to the transaction menu
1d	Add the sale item “Baked Potato” to the transaction	Press on the “Baked Potato” button in the point of sale system	The sale item is added to the transaction menu	The sale item is added to the transaction menu 
1e	Complete the current transaction	Press the “complete” button in the transaction menu	The current transaction is completed, and a new transaction ID is shown on the point of sale system web page	An error is produced more than 100 times: 

The sale items were added well to the transaction. However, this was not the aim of this test. This test aimed to test that a transaction is completed when the complete transaction button is pressed. The transaction is not completed, instead, over 100 of the following error was produced:

Warning: Trying to access array offset on value of type null in **C:\xampp\htdocs\POSSY\process_completeTransaction.php** on line **54**
ERROR! A product was missing from your stock management database

As the same error was produced many times, I assume that an infinite loop was formed, and I found the following culprit:

```
for ($k = 0; $k < mysqli_num_rows($saleItemContentsResult); $i++) {
```

The wrong variable is incremented, leading to the loop being infinite. To fix this issue, I need to change the variable updated inside the for loop from $i to $k

```
for ($k = 0; $k < mysqli_num_rows($saleItemContentsResult); $k++) {
```

The transaction should now be able to be completed, and a new transaction is automatically opened.

I then ran a test, See **complete tranasction test.mp4**

The current transaction was completed in this test, and the transaction ID was increased. However, when returning to the point-of-sale system, the following error was produced:

Fatal error: Uncaught TypeError: mysqli_num_rows(): Argument #1 ($result) must be of type mysqli_result, bool given in C:\xampp\htdocs\POSSY\pointOfSale.php:111 Stack trace: #0 C:\xampp\htdocs\POSSY\pointOfSale.php(111): mysqli_num_rows(false) #1 {main} thrown in **C:\xampp\htdocs\POSSY\pointOfSale.php** on line **111**

To fix this, I will need only to run the mysqli_num_rows function if the result of the MySQLi query is not false. I can do this using if statements.

Before:

```
107         $tableName = "transaction_".$transactionID."_".$companyName;
108         $query = "SELECT SaleItemName,Cost,Quantity FROM $tableName";
109         $result = mysqli_query($connection,$query);
110
111         for($i=0;$i<mysqli_num_rows($result);$i++){
112             $row = mysqli_fetch_assoc($result);
113             $saleItemName = $row['SaleItemName'];
114             $saleItemCost = "£".$row['Cost'];
115             $quantity = $row['Quantity'];
116
117             echo "<p align='center'>$saleItemName <br> Quantity: $quantity <br> $saleItemCost</p>";
118         }
119
120         /* Output total cost of transaction */
121         $query = "SELECT Quantity, Cost FROM $tableName";
122         $result = mysqli_query($connection, $query);
123
124         $total = 0;
125         for($i=0;$i<mysqli_num_rows($result);$i++){
126             $row = mysqli_fetch_assoc($result);
127             $total += $row['Cost'] * $row['Quantity'];
128
129         }
130         echo "<h3 align='center'>Total: $total</h3>";
131
```

After:

```
110
111         $total = 0; // Stores the total for the current transaction
112
113         /* Only runs if the SQL query produces results */
114         if($result != false) {
115
116             /* Loops through each row of the results from the SQL table */
117             for ($i = 0; $i < mysqli_num_rows($result); $i++) {
118                 $row = mysqli_fetch_assoc($result);
119                 $saleItemName = $row['SaleItemName'];
120                 $saleItemCost = "£" . $row['Cost'];
121                 $quantity = $row['Quantity'];
122
123                 echo "<p align='center'>$saleItemName <br> Quantity: $quantity <br> $saleItemCost</p>";
124             }
125
126             /* Calculate total cost of transaction */
127             $query = "SELECT Quantity, Cost FROM $tableName";
128             $result = mysqli_query($connection, $query);
129
130             for ($i = 0; $i < mysqli_num_rows($result); $i++) {
131                 $row = mysqli_fetch_assoc($result);
132                 $total += $row['Cost'] * $row['Quantity'];
133
134             }
135         }
136         /* Output total cost of transaction */
137         echo "<h3 align='center'>Total: $total</h3>";
```

I will now re-run the tests in the new transaction, checking everything is working correctly.

Test No.	Description	Test Data	Expected Result	Actual Result
1d	Add the sale item “Cooked Potato” to the transaction	Press on the “Cooked Potato” button in the point of sale system	The sale item is added to the transaction menu	An error is produced
1d	Add the sale item “Baked Potato” to the transaction	Press on the “Baked Potato” button in the point of sale system	The sale item is added to the transaction menu	An error is produced
1e	Complete the current transaction	Press the “complete” button in the transaction menu	The current transaction is completed, and a new transaction ID is shown on the point of sale system web page	An error is produced

These tests were completely unsuccessful. However, all errors stem from the same issue, the wrong type being input into the function mysqli_num_rows().

Fatal error: Uncaught TypeError: mysqli_num_rows(): Argument #1 ($result) must be of type mysqli_result, bool given in C:\xampp\htdocs\POSSY\process_completeTransaction.php:33 Stack trace: #0 C:\xampp\htdocs\POSSY\process_completeTransaction.php(33): mysqli_num_rows(false) #1 C:\xampp\htdocs\POSSY\process_completeTransaction.php(161): updateStockManagementSystem(Object(mysqli), 'company') #2 {main} thrown in **C:\xampp\htdocs\POSSY\process_completeTransaction.php** on line **33**

```
33      $numRows = mysqli_num_rows($result); //returns the number of rows in the table
```

This is an easy fix. I just need to implement if statements at any point this error could occur, checking that the parameter being input into the function is not false.

```
33      if ($result != false) {
34          $numRows = mysqli_num_rows($result); //returns the number of rows in the table
35      }
36      else{
37          $numRows = 0;
38      }
```

This function has also been used numerous times in other files of the program. Therefore in future tests, I will need to make sure that I test this, ensuring that they do not throw errors.

I then tried to run the following tests:

Test No.	Description	Test Data	Expected Result	Actual Result
1d	Add the sale item “Cooked Potato” to the transaction	Press on the “Cooked Potato” button in the point of sale system	The sale item is added to the transaction menu	The sale item is added to the transaction menu
1d	Add the sale item “Baked Potato” to the transaction	Press on the “Baked Potato” button in the point of sale system	The sale item is added to the transaction menu	The sale item is added to the transaction menu
1e	Complete the current transaction	Press the “complete” button in the transaction menu	The current transaction is completed, and a new transaction ID is shown on the point of sale system web page	The current transaction is completed, and a new transaction ID is shown on the point of sale system web page
2a	Add the sale item “Cooked Potato” to the transaction	Press on the “Cooked Potato” button in the point of sale system	The sale item is added to the transaction menu	The sale item is added to the transaction menu
2b	Add the sale item “Baked Potato” to the transaction	Press on the “Baked Potato” button in the point of sale system	The sale item is added to the transaction menu	The sale item is added to the transaction menu
2c	Complete the current transaction	Press the “complete” button in the transaction menu	The current transaction is completed, and a new transaction ID is shown on the point of sale system web page	The current transaction is completed, and a new transaction ID is shown on the point of sale system web page

All tests have now passed, and therefore this feature is complete. The transaction column is well placed, and the current transaction ID is clear to the system operator. The contents of the current transaction are shown clearly to the system operator, displaying all sale items the current transaction contains, their price, and their quantity. The complete transaction button functions as intended but is not designed as was initially intended. The button should be coloured red, larger, centred, and at the bottom of the transaction column. However, these features are not essential to the functionality of the project, and therefore shall not be rectified at this stage. Perhaps I will return to this towards the end of this stage if time permits.

Triggering low-stock alerts

The function to trigger a low stock alert is in the stockmanagement.php folder, but I need to run this function from the pointOfSale.php file. Therefore, I need to require the use of the pointOfSale.php file once to run the function. To do this, I will use the following statement:

```
198     /* Check for low stock, and trigger an alert if necissary */
199     require_once("stockManagement.php");
200     checkForLowStock($connection,$companyName);
201     ?>
```

I then tried testing this, as seen in **low stock alert testing.mp4**

The following errors were produced when trying to complete a transaction

Warning: Trying to access array offset on value of type null in **C:\xampp\htdocs\POSSY\process_completeTransaction.php** on line **114**

Fatal error: Uncaught TypeError: mysqli_num_rows(): Argument #1 ($result) must be of type mysqli_result, bool given in C:\xampp\htdocs\POSSY\process_completeTransaction.php:121 Stack trace: #0 C:\xampp\htdocs\POSSY\process_completeTransaction.php(121): mysqli_num_rows(false) #1 C:\xampp\htdocs\POSSY\process_completeTransaction.php(182): updateStockManagementSystem(Object(mysqli), 'company') #2 {main} thrown in **C:\xampp\htdocs\POSSY\process_completeTransaction.php** on line **121**

These errors are not related to low-stock alerts, as the low stock alert should be triggered once redirected away from this page.

```
113             // Store the table name for the contents of the current sale item
114             $saleItemContentsTableName = "SaleItem_Contents_" . $companyName . "_" . $row['ID'];
```

```
120             // Loop through each product of a sale item
121             for ($k = 0; $k < mysqli_num_rows($saleItemContentsResult); $i++) {
122                 $saleItemContentsRow = mysqli_fetch_assoc($saleItemContentsResult);
123
```

I have forgotten to include a copy of the original query selecting everything from the transaction table when looping back through the sale items. This means that I have already fetched the final row of the results, and therefore when they try being fetched again, there are no more results. To fix this, I need to duplicate the query onto lines 107 – 111, as shown below:

```
100 /*
101  *  If a product is not in stock, all products quantities should be added back to the stock management
102  *  database, as the transaction will not be completed.
103  *
104  * This section could be severely simplified, probably by using a while loop and a selection statement
105  * in the previous section
106  */
107
108 // Select everything contained within the transaction table
109 $query = "SELECT * FROM $tableName";
110 $result = mysqli_query($connection, $query);
111
112 if ($returnValue == false) {
113     // Loop through sale items
114     for ($i = 0; $i < $numRows; $i++) {
115         // fetch a new row from the SQL result
116         $row = mysqli_fetch_assoc($result);
117
118         // Store the table name for the contents of the current sale item
119         $saleItemContentsTableName = "SaleItem_Contents_" . $companyName . "_" . $row['ID'];
120
121         // Select all rows for the contents of the current sale item
122         $query = "SELECT * FROM $saleItemContentsTableName";
123         $saleItemContentsResult = mysqli_query($connection, $query);
124
125         // Loop through each product of a sale item
126         for ($k = 0; $k < mysqli_num_rows($saleItemContentsResult); $i++) {
127             $saleItemContentsRow = mysqli_fetch_assoc($saleItemContentsResult);
```

Although this did not fix the problem, the same errors were still occurring in the same places of the program:

Warning: Undefined array key "ID" in **C:\xampp\htdocs\POSSY\process_completeTransaction.php** on line **118**

Fatal error: Uncaught TypeError: mysqli_num_rows(): Argument #1 ($result) must be of type mysqli_result, bool given in C:\xampp\htdocs\POSSY\process_completeTransaction.php:125 Stack trace: #0 C:\xampp\htdocs\POSSY\process_completeTransaction.php(125): mysqli_num_rows(false) #1 C:\xampp\htdocs\POSSY\process_completeTransaction.php(186): updateStockManagementSystem(Object(mysqli), 'company') #2 {main} thrown in **C:\xampp\htdocs\POSSY\process_completeTransaction.php** on line **125**

I then realised that the error on line 125 is likely caused by the error on line 118, as it requires the correct query to run on line 118. Therefore, I focused on solving the error on line 118. This was when I discovered that the stock quantity of a product is set to zero when a transaction is completed, rather than only deducting the necessary amount. This is shown in **Stock Quantity Cleared.mp4.** I believe the following query causes this:

```
83              // Deducts from the stock management system
84              $productQuantity = $saleItemContentsRow['Quantity'];
85              $query = "UPDATE $stockManagementTableName
86                        SET CurrentStockValue = 'CurrentStockValue - $productQuantity'
87                        WHERE ProductName = '$productName'";
88              mysqli_query($connection, $query);
89
```

I think that 'CurrentStockValue - $productQuantity' is not valid. To fix this, I will insert an MySQLi query prior to this one, selecting CurrentStockValue for the current ProductName. I will then calculate the new stock value, storing it in a variable $newStockValue. This variable will then be passed into the existing query.

```
83              // Calculate new stock value of product
84              $productQuantity = $saleItemContentsRow['Quantity'];
85
86              $query = "SELECT CurrentStockValue FROM $stockManagementTableName WHERE ProductName = '$productName'";
87              $currentStockValueQueryResult = mysqli_query($connection,$query); // Run SQL query
88
89              $currentStockValueQueryRow = mysqli_fetch_assoc($currentStockValueQueryResult);
90              $newStockQuantity = $currentStockValueQueryRow['CurrentStockValue'] - $productQuantity;
91
92              // Deducts from the stock management system
93              $query = "UPDATE $stockManagementTableName
94                        SET CurrentStockValue = '$newStockQuantity'
95                        WHERE ProductName = '$productName'";
96              mysqli_query($connection, $query);
97
```

The same code appears later in the file for addition rather than subtraction, which will also set the CurrentStockValue to an incorrect number if run.

```
146             // Produce an error if more than one result is returned
147             // Produce an error if no results are returned
148             if ($productQueryResult == false) {
149                 echo "ERROR! A product was missing from your stock management database";
150             }
151             else if (mysqli_num_rows($productQueryResult) == 1) {
152
153                 // Increases quantities in the stock management system
154                 $productQuantity = $saleItemContentsRow['Quantity'];
155                 $query = "UPDATE $stockManagementTableName
156                           SET CurrentStockValue = 'CurrentStockValue + $productQuantity'
157                           WHERE ProductName = '$productName'";
158                 mysqli_query($connection, $query);
159
160             }
161             else {
162                 echo "ERROR! Duplicated product names! Remove duplicates from your stock management database";
163             }
164
```

I will include the same corrections to this section of code:

```
152             else if (mysqli_num_rows($productQueryResult) == 1) {
153
154                 // Calculate new stock value of product
155                 $productQuantity = $saleItemContentsRow['Quantity'];
156
157                 $query = "SELECT CurrentStockValue FROM $stockManagementTableName WHERE ProductName = '$productName'";
158                 $currentStockValueQueryResult = mysqli_query($connection,$query); // Run SQL query
159
160                 $currentStockValueQueryRow = mysqli_fetch_assoc($currentStockValueQueryResult);
161                 $newStockQuantity = $currentStockValueQueryRow['CurrentStockValue'] + $productQuantity;
162
163                 // Increases quantities in the stock management system
164                 $query = "UPDATE $stockManagementTableName
165                           SET CurrentStockValue = '$newStockQuantity'
166                           WHERE ProductName = '$productName'";
167                 mysqli_query($connection, $query);
168
169             }
```

Although this will fix the incorrect stock value from being deducted, it will not fix the errors we were previously getting relating to undefined array keys and incorrect data types. The undefined array key was caused by the incorrect name for a table column heading being used to access the data within the $row variable. ID was being used instead of SaleItemID.

```
129             // Store the table name for the contents of the current sale item
130             $saleItemContentsTableName = "SaleItem_Contents_" . $companyName . "_" . $row['ID'];
131
```

To fix this, I need to change $row[‘ID’] to $row[‘SaleItemID’]

```
129             // Store the table name for the contents of the current sale item
130             $saleItemContentsTableName = "SaleItem_Contents_" . $companyName . "_" . $row['SaleItemID'];
131
```

The incorrect data type error was caused by the previous error, but also the if statement incremented the incorrect variable:

```
134             // Loop through each product of a sale item
135             for ($k = 0; $k < mysqli_num_rows($saleItemContentsResult); $i++) {
136                 $saleItemContentsRow = mysqli_fetch_assoc($saleItemContentsResult);
137
```

To fix this, I need to increment the correct variable $k

```
134             // Loop through each product of a sale item
135             for ($k = 0; $k < mysqli_num_rows($saleItemContentsResult); $k++) {
136                 $saleItemContentsRow = mysqli_fetch_assoc($saleItemContentsResult);
137
```

I will now return to resolving the issue of low-stock alerts not working. I suspect that the function is not being called from the external file. Therefore, I will try duplicating the low stock trigger function into the pointOfSale.php file and calling it.

```php
<?
/* Check for low stock, and trigger an alert if necessary */
checkForLowStock($connection,$companyName);

function checkForLowStock($connection,$companyName){
    $tableName = "stockmanagementtable_".$companyName;
    $query = "SELECT ProductID,ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone
        FROM $tableName WHERE CurrentStockValue < MinimumStockValue AND Ordered='off'";
    $result = mysqli_query($connection, $query);

    if(mysqli_num_rows($result) > 0) {
        $_SESSION['returnedRows'] = mysqli_fetch_assoc($result);
        $_SESSION['companyName'] = $companyName;
        $_SESSION['tableName'] = $tableName;

        header( header: 'Location: lowStock.php');// redirect user
        exit;

    }
}
?>
```

I then tried adding a sale item to a transaction and completing it, but I got the following error:

ERROR! A product was missing from your stock management database

I then output the query being run by the program. The name of the product is not being fed into the query:

SELECT ProductName,CurrentStockValue FROM stockmanagementtable_company WHERE ProductName = '

This is due to this query being incorrect

```php
        // Store the table name for the contents of the current sale item
        $saleItemContentsTableName = "SaleItem_Contents_".$companyName."_".$row['SaleItemID'];
```

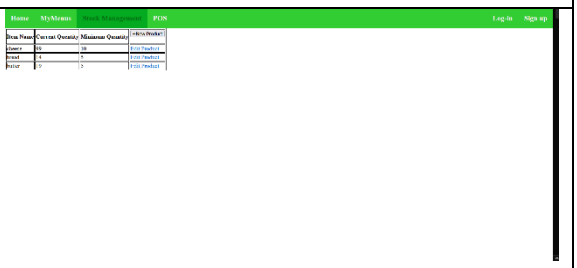
$row['SaleItemID'] gets the ID of the sale item in the current transaction. However, I need the ID of the sale item from the menu table as the table name. Therefore, I need to create a session variable to store the menu Id when opening an existing menu.

```php
$query = "SELECT MenuName,MenuID FROM menus WHERE CompanyName = '$companyName' AND menuName = '$menuName'";
$result = mysqli_query($connect,$query);

$row = mysqli_fetch_assoc($result);
$_SESSION['menuID'] = $row['MenuID'];
```

I then need to retrieve the menu ID from the session variable, calculate the table name for the current menu, and then use an SQL query to search the current menu for the ID of the current sale item.

```php

    // Get the table name for the current menu
    $menuID = $_SESSION['menuID'];
    $name = "menu_".$companyName."_".$menuID;
    $currentSaleItemName = $row['SaleItemName'];

    // Get the sale item id of the current product from the current menu table
    $query = "SELECT ID FROM $name WHERE saleItemName = '$currentSaleItemName'";
    $queryResult = mysqli_query($connection,$query);
    $rowID = mysqli_fetch_assoc($queryResult);

```

I will then test this:

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Add the sale item “testing” to a transaction until one of its products (bread) is below its minimum stock quantity	Set the current stock level of bread to 15 and the minimum quantity to 5. Press on the “testing” sale item button in the point of sale system 12 times, then close the transaction	The transaction is closed, and the transaction ID is updated. The current stock level of bread is decreased from 15 to 3. A low-stock alert is triggered	 • The current transaction is closed, and the transaction id is incremented. • The current stock level is decreased from 15 to 14 • No low-stock alert is triggered

The test failed – the stock level of bread was not decreased by the correct amount. The stock level should have been decreased by 12 but was only decreased by 1. The removal value of the product is not being multiplied by the number of sale items in a transaction. To fix this, I need to multiply the quantity of a product taken away by the quantity of the sale item in the current transaction.

```php
if ($productRow['CurrentStockValue'] - ($saleItemContentsRow['Quantity'] * $row['Quantity']) < 0) {

    $returnValue = false;
}
```

```php
$newStockQuantity = $currentStockValueQueryRow['CurrentStockValue'] - ($productQuantity * $row['Quantity']);

// Deducts from the stock management system
$query = "UPDATE $stockManagementTableName
          SET CurrentStockValue = '$newStockQuantity'
          WHERE ProductName = '$productName'";
mysqli_query($connection, $query);
```

```php
$newStockQuantity = $currentStockValueQueryRow['CurrentStockValue'] + ($productQuantity * $row['Quantity']);

// Increases quantities in the stock management system
$query = "UPDATE $stockManagementTableName
          SET CurrentStockValue = '$newStockQuantity'
          WHERE ProductName = '$productName'";
mysqli_query($connection, $query);
```

Upon refreshing the point-of-sale system, I received the following error:

Warning: Cannot modify header information - headers already sent by (output started at C:\xampp\htdocs\POSSY\pointOfSale.php:185) in **C:\xampp\htdocs\POSSY\pointOfSale.php** on line **218**

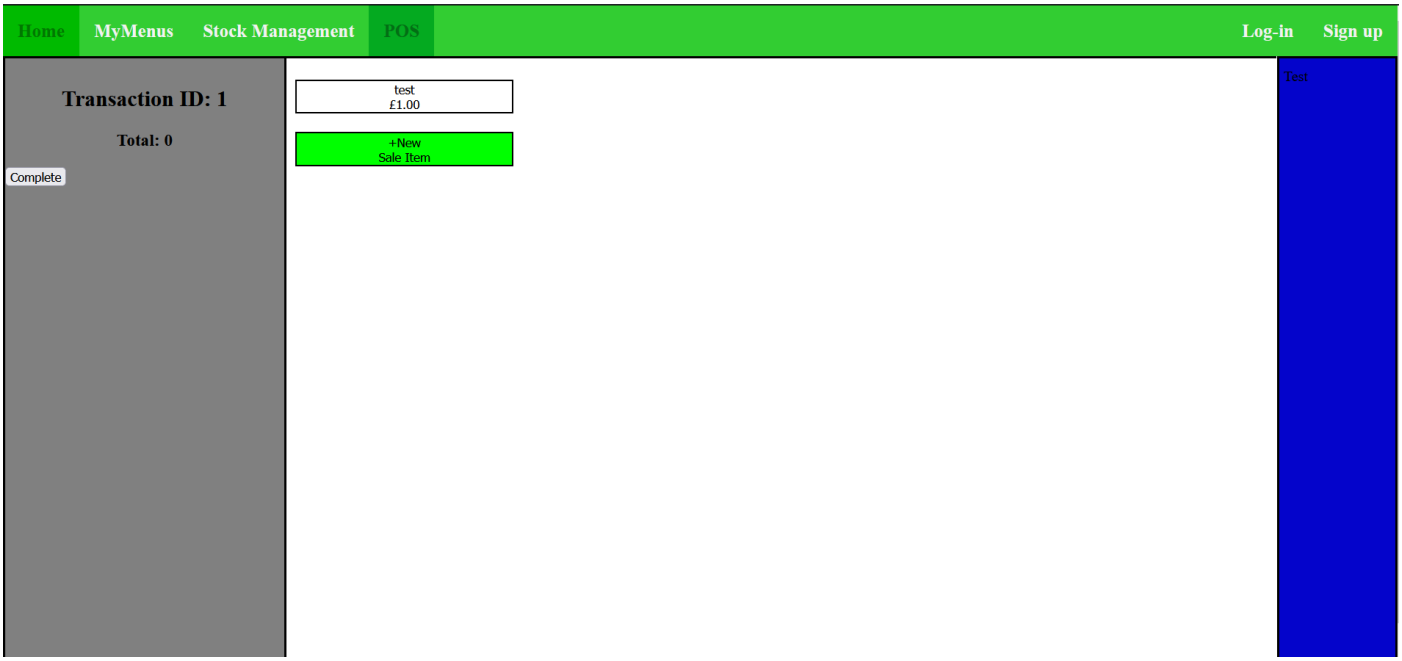
This error occurs as the HTML header is altered after values have been output to the web page using PHP. To fix this, I need to place this check before any outputs are made to the web page using php.

```
89          session_start();
90          $companyName = $_SESSION['companyName'];
91
92          /* Open Database Connection */
93          include "database_connect.php";
94          $connection = openConnection();
95
96          /* Check for low stock, and trigger an alert if necessary */
97          checkForLowStock($connection,$companyName);
```

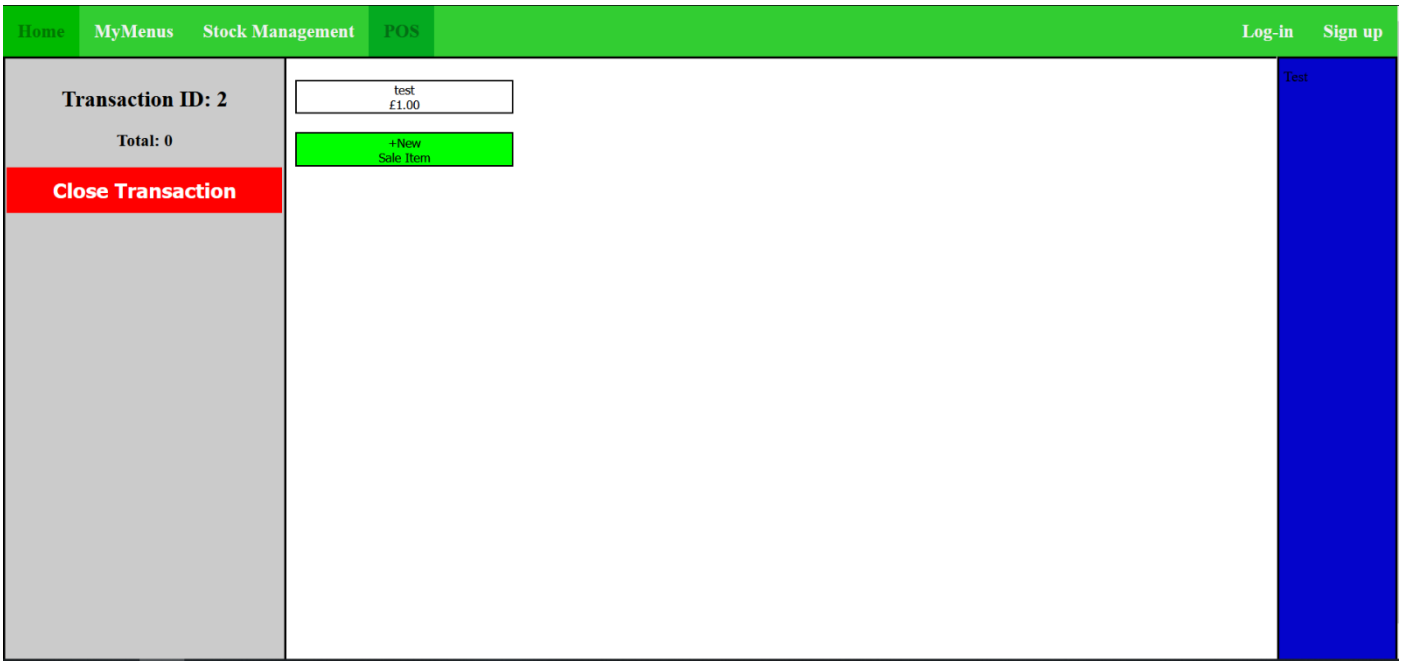
Making some changes

The background colour of the point-of-sale system is yellow and has a blue border. This was temporary,

The left-hand transaction column is well-sized but should have a black border and a grey background. The centre column containing the sale item buttons should be white and only have a top and bottom black border. The right-hand column should have a black border. My designs suggested there should be no space between buttons in the right-hand column. However, the way in which this column was designed in the project requires space between the buttons. Therefore, the background will be dark blue. Sale item buttons will have a black border but will remain with a white background for now, as their design is not fully implemented at this stage. The +New Sale Item button will remain green and will also have a black border



I like the colours of both the centre and right-hand columns, and the buttons are of a good size. However, the left-hand column is too dark of a grey colour. I will change this to a lighter grey. I also do not like that the Complete transaction button is not centred, is not red, and is not labelled “close transaction” rather than just “complete”, and the text colour should be white and in a larger font.



I am content with this design. The only issue is that the cost total is not displayed always to two decimal places. However, the total cost is displayed to 2 decimal places when necessary. Therefore, I do not see this as a problem, and it will remain unfixed.

Another problem is that the “close transaction” button is not displayed at the bottom of the transaction column, it instead moves down with the greater the number of sale items in the transaction. Although this is not ideal, it is not a problem therefore will remain unfixed unless I have time to fix this at the end of the project

General Testing Of This Stage

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Does the database store the containing products of each sale item button in a table?	(Press Create sale item button) Cheese sandwich 3.00 Food cheese,bread,bread,butter	Test data displays correctly in the menus database	Test data displays correctly in the menus database See **general testing stage 5.mp4**
2a	Does pressing a sale item button to add that item to the current transaction?	Press any sale item button	Sale item pressed is added to the current transaction	Sale item pressed is added to the current transaction See **general testing stage 5.mp4**
4a	Is the total cost of a transaction increased by the correct amount after a sale item is added to the transaction?	Press any sale item button	Total cost increases by the correct amount	Total cost increases by the correct amount See **general testing stage 5.mp4**
5a	Does pressing the "complete transaction" button close the current transaction?	Press the "close transaction" button	Current transaction closes	The current transaction is closed See **general testing stage 5.mp4**
5b	Does completing a transaction decrease the product stock level from the stock management system by the correct amount?	Press any sale item button three times	Correct product quantities are deducted from the stock management database table	Correct product quantities are deducted from the stock management database table See **general testing stage 5.mp4**
5c	Can you add multiple of the same sale item to a transaction?	Add multiple of the same sale item button to a transaction	Multiple of the same sale item are contained within the transaction	Multiple of the same sale item can be contained within the same transaction See **general testing stage 5.mp4**
5d	Can you complete an empty transaction?	Press complete on the current transaction	The transaction is not complete, and the transaction ID remains the same	The empty transaction is completed See **general testing stage 5.mp4**
6a	Does a transaction remain open if the products to create a sale item are out of stock?	Add a sale item to the transaction until it will place a product out of stock	The transaction ID remains the same	The transaction and its id remains the same
6b	Is a low stock alert triggered if a product goes below a minimum stock level after a transaction is completed?	Add a sale item to a transaction until it is below its minimum stock level, but above or equal to 0	A low stock alert is triggered	A low stock alert is triggered See **general testing stage 5.mp4**

Overall – this test went well. I had made a couple of changes since the last test; therefore, I expected some related bugs. However, this was not really the case. However, a transaction can be completed when it is empty. I will need to include an SQL statement to fix this, selecting everything from the current transaction table. If results are returned, the stock management system should be updated. Otherwise, an error should be output.

```php
/* Select everything from the current transaction table */
$tableName = "transaction_" . $_GET["transactionID"] . "_" . $_SESSION['companyName'];
$query = "SELECT * FROM $tableName";
$result = mysqli_query($connection,$query);

/*
 *   Only attempt to update the stock management system
 *   if are sale items contained within the current transaction
*/
if($result != false) {
    if(mysqli_num_rows($result) != 0) {
        updateStockManagementSystem($connection, $companyName);
    }
    else{
        echo "Error! Transaction is empty";
    }
}
else{
    echo "Error! Transaction is empty";
}
```

End of stage testing

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Does the database store the containing products of each sale item button in a table?	(Press Create sale item button) Cheese sandwich 3.00 Food cheese,bread,bread,butter	Test data displays correctly in the menus database	Test data displays correctly in the menus database See **general testing stage 5.mp4**
2a	Does pressing a sale item button to add that item to the current transaction?	Press any sale item button	Sale item pressed is added to the current transaction	Sale item pressed is added to the current transaction See **general testing stage 5.mp4**
4a	Is the total cost of a transaction increased by the correct amount after a sale item is added to the transaction?	Press any sale item button	Total cost increases by the correct amount	Total cost increases by the correct amount See **general testing stage 5.mp4**
5a	Does pressing the "complete transaction" button close the current transaction?	Press "close transaction" button	Current transaction closes	The current transaction is closed See **general testing stage 5.mp4**
5b	Does completing a transaction decrease the product stock level from the stock management system by the correct amount?	Press any sale item button three times	Correct product quantities are deducted from the stock management database table	Correct product quantities are deducted from the stock management database table See **general testing stage 5.mp4**
5c	Can you add multiple of the same sale item to a transaction?	Add multiple of the same sale item button to a transaction	Multiple of the same sale item are contained within the transaction	Multiple of the same sale item can be contained within the same transaction See **general testing stage 5.mp4**
5d	Can you complete an empty transaction?	Press complete on the current transaction	The transaction is not complete, and the transaction ID remains the same	The empty transaction is not completed – transaction id remains the same See **final test stage 5.mp4**
6a	Does a transaction remain open if the products to create a sale item are out of stock?	Add a sale item to the transaction until it will place a product out of stock	The transaction ID remains the same	The transaction and its id remains the same See **final test stage 5.mp4**
6b	Is a low stock alert triggered if a product goes below a minimum stock level after a transaction is completed?	Add a sale item to a transaction until it is below its minimum stock level, but above or equal to 0	A low stock alert is triggered	A low stock alert is triggered See **general testing stage 5.mp4**

The development of this stage took a lot longer than expected. This is mainly due to the various bugs that kept revealing other bugs during the debugging process. Despite this, I am happy with the results of this stage. The point-of-sale system is functioning well, and all intended features during this stage have been implemented. The project is coming together well and meets the bare minimums of its intended uses at the current stage.

This stage met the following success criteria:

- Clickable button to add sale items to transactions as described in requirements 2.2.0
- A sale item database table as outlined in requirements 2.16
- Stock management database table automatically edited after a transaction is complete as outlined in requirements 3.4
- Low stock level alerts output to the operator as detailed in requirements 3.6.0
- Ability to create a new transaction as outlined in requirements 2.5

It has begun to meet:

- A point-of-sale system user interface as described in requirements 2.1
- The ability for new sale item buttons to be added to the system as described in requirements 2.2.3

I would like to return to some areas of this stage before the end of my project. These include:

The sale item buttons in the point-of-sale system are aligned incorrectly. My designs show the point-of-sale buttons being aligned in four columns, as well as rows. Currently, sale item buttons are only displayed in a single column due to a bug. I would like to return to this, as it affects the system operator's efficiency when using the program, which detracts from the aim of my project, making the process of a transaction as fast as possible.

The total cost of a transaction is not always displayed to 2 decimal places. The cost of a transaction will only be displayed to two decimal places if the total requires two decimal places. Money is conventionally displayed to 2 decimal places, and my designs of the point-of-sale system also show this total being displayed to 2 decimal places. This means that this feature is not as professional as it could be; however, this bug does not affect the usability of this program; therefore, it is unlikely to be fixed.

Stage 6 – Drop-down Menu & Refunds (2.5hours)

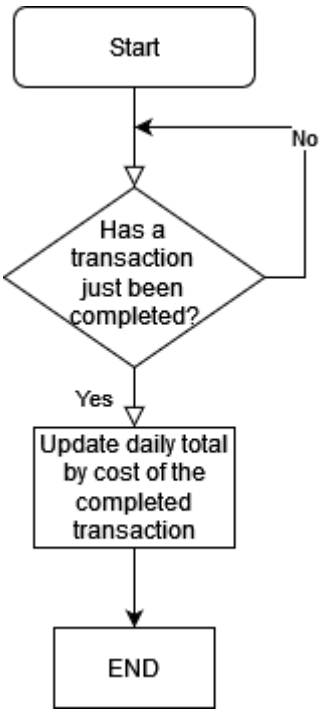
The drop-down menu is essential as it contains the final essential feature of my project – the daily total. This menu will also contain the ability to refund purchases and has the ability to host many more features in the future, such as the toggling of alerts.

The refund page will allow for purchases to be refunded in a calculator format. This is important to include at this stage of the project, as on average, 5 to 10 per cent of in-store purchases are refunded. Therefore, it is highly likely that this feature will be used by most companies registered on the website.

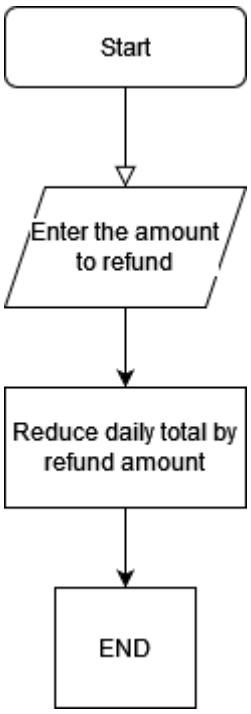
Features/ functionality:

1. A drop-down menu GUI
2. A refunds web page
3. An algorithm to calculate the daily total to be displayed in the drop-down menu
4. An algorithm to calculate the refunded amount and deduct it from the daily total

Test No.	Description	Type Of Test	Test Data	Expected Result	Actual Result
1a	Does the drop-down menu generate correctly sized		Press the drop-down menu button	The drop-down menu opens and is sized correctly	
2a	Does the "Refunds" button redirect the user to the refunds page?		Press the "Refunds" button	Refunds page opens	
3a	Does the Refunds page generate correctly sized?		Press the "Refunds" button	Refunds page opens and is sized correctly	
4a	Does refunding an amount deduct the amount from the daily total?		5.00	The daily total is reduced by £5.00	



This shows the process of a transaction being completed. *The system will continuously check to see if a transaction has been completed, this prevents the user from having to press this button multiple times before the click is registered. If a transaction is complete, the daily total will be updated (contained within the drop-down menu)*



This shows the process behind refunding a transaction. First the user must enter the amount they would like to refund. The daily total is then reduced by this amount by adding a transaction to the transaction table with a negative value.

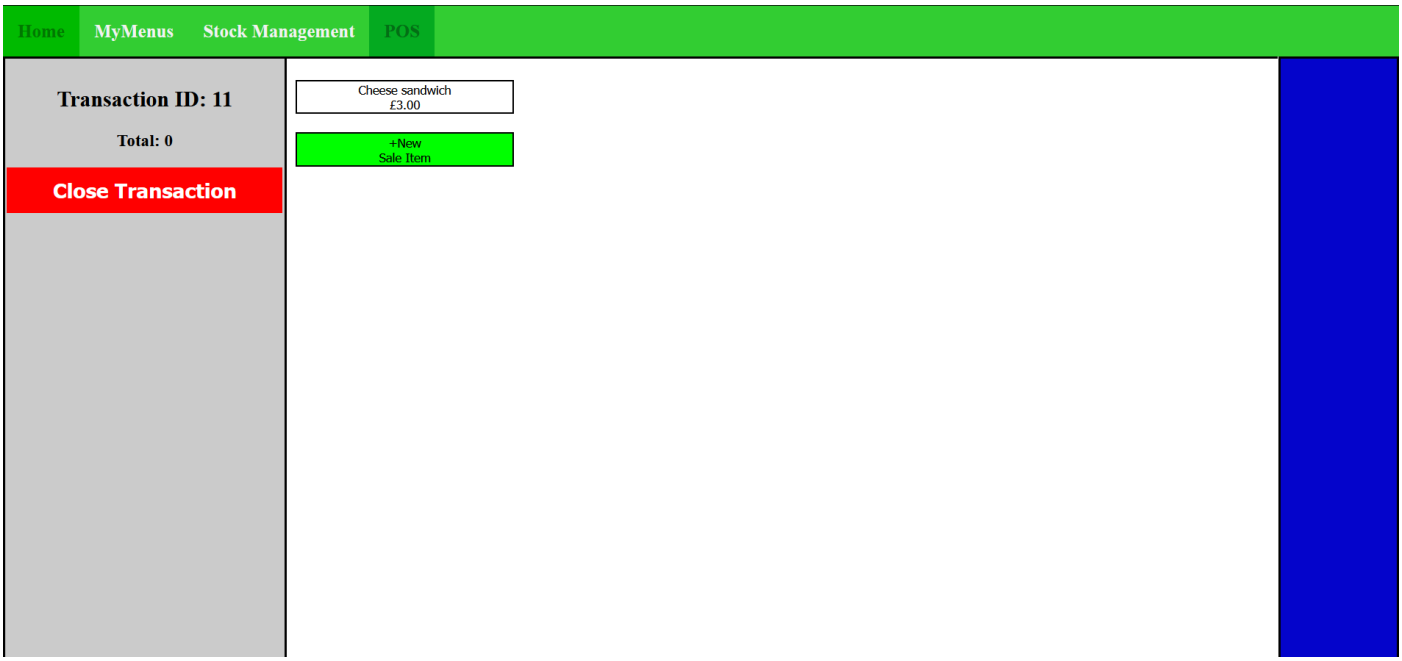
A drop-down menu GUI

My original design included a profile picture. This will not be implemented at this stage, as it is unnecessary and will require a lot of time to implement. Instead, the user's username displayed on the right-hand side of the taskbar will trigger a drop-down menu when hovered over. This will be included in the point-of-sale system only.

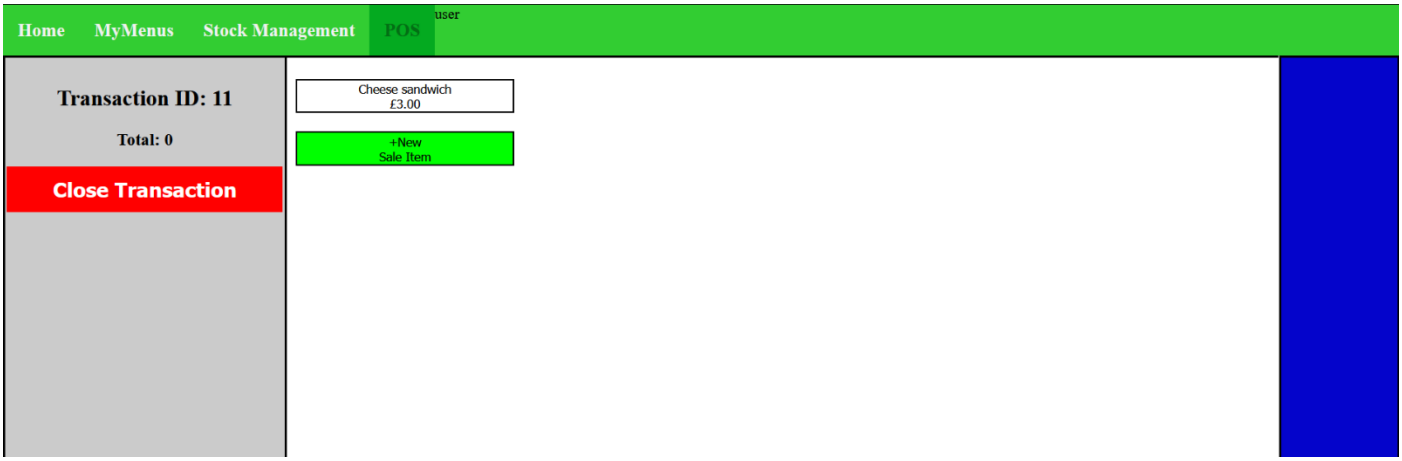
I will begin by removing the "log-in", and "sign-up" buttons from the taskbar in the point-of-sale system as these are not necessary.

Before:

After:



I will now insert the current user's username where the login and signup buttons were previously displayed. I will access the users' username using a session variable initiated during the login process.

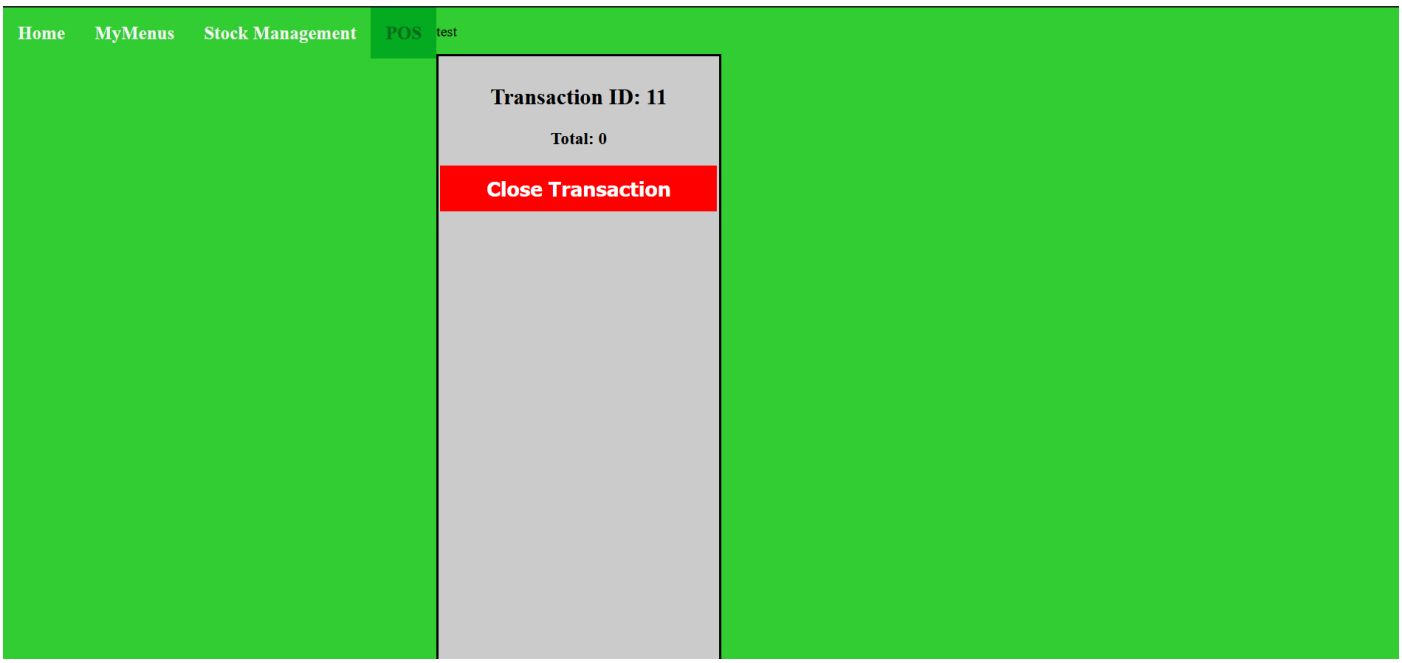


I now need to add some styling, changing the font colour to white and float the username from the right-hand side of the taskbar.

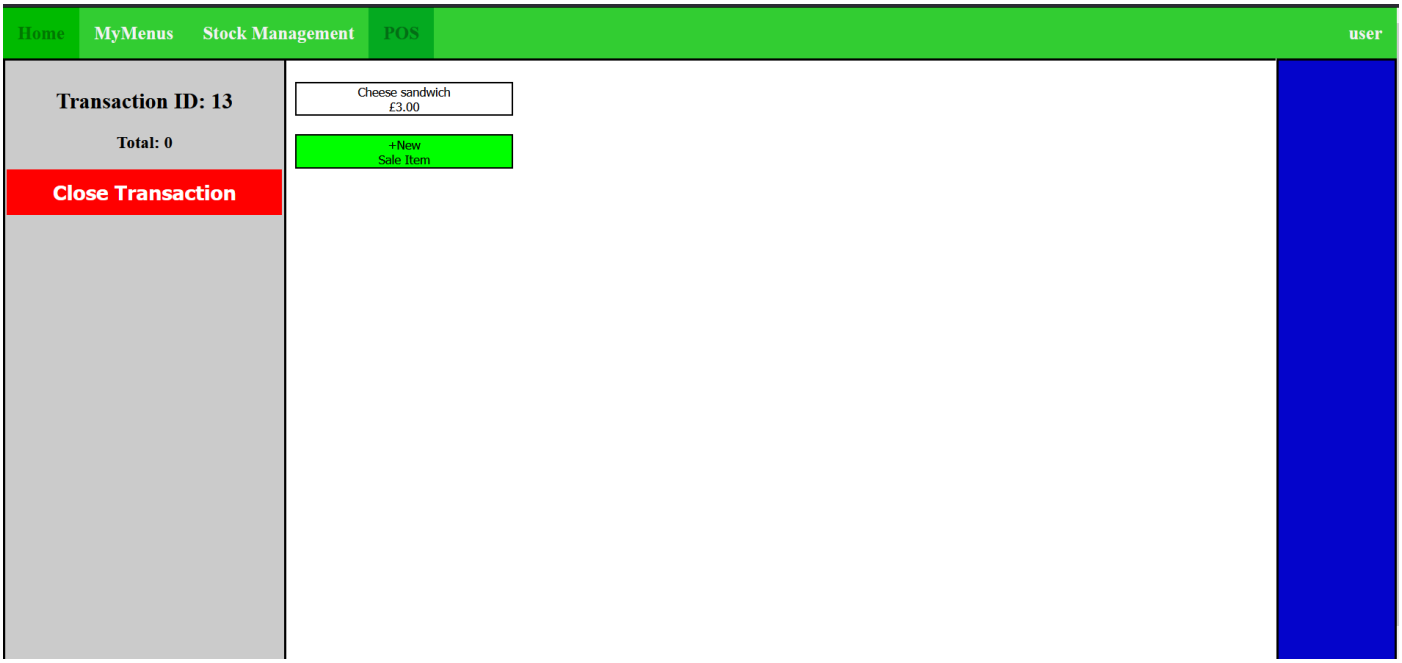
I am having problems trying to create a drop-down menu. I wrote the following:

```php
<?php
session_start();

$username = $_SESSION['username']; // Get the current user's username
?>

<!-- Display drop-down menu -->
<div class='dropdown'>
    <span style='float: right;'><?php $username ?></span>
    <div class='content'>
        <p>test</p>
    </div>
</div>
```

And my page looked like this:



Therefore I will no longer be creating a drop-down menu for easiness. Instead, I will create a new web page that will contain the contents of the drop-down menu. When the user's username is pressed, they will be re-directed to the web page, which contains what would have been the drop-down menu. This also requires that the drop-down menu has a link back to the point of sale system before, where this was not the case if following my original design.



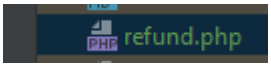
This is as I had expected after the design change. I hope to be able to create the drop-down menu in the future, but I doubt that I will have enough time to do so.

Creating the ability to refund an amount

I will first need to create a button in the drop-down menu to redirect the user to the refund web page

```
1  <form action="refund.php">
2      <button type="submit" name="refund">Refund</button>
3  </form>
```

I will now create a refund.php file



Initially, I had planned on a calculator style to input the amount to refund. However, I decided that I would not do this for simplicity. Instead, I will use an input box to input an amount and a submit button to send this amount to a processing page where the amount will be deducted from the daily total.

After writing the above, I realised that I could not code this section of the program without having first created the daily total. Therefore, I will do this first then return back to creating the refund page.

Creating a daily total

First, I will need to calculate the daily total. I had initially planned on passing through each transaction and calculating the total cost of the transaction, then totalling these together. However, for simplicity, I will instead add a new column to the transaction history table for each company. Not only will this make calculating the total easier, it will also make refunds easier as a refund can be displayed as a transaction with a negative total, and therefore when the daily total is added up, it can be done so easily. This was not initially planned to happen this way in my project, although I feel this will be easier to code but also easier to maintain after the project is complete.

My first step is to add a column to the transaction history table

```
111         /* Create A Transaction History Table For This Company */
112         $query = "CREATE TABLE `users`.`transactionhistory_$companyName` (
113                   `TransactionID` INT(9) NOT NULL AUTO_INCREMENT , `Complete` TEXT NOT NULL DEFAULT 'no' ,
114                   PRIMARY KEY (`TransactionID`))";
115         mysqli_query($connection,$query);
```

for each company called total. This involves altering the SQL query that creates this table during sign-up of a new account, as shown below.

```
111         /* Create A Transaction History Table For This Company */
112         $query = "CREATE TABLE `users`.`transactionhistory_$companyName` (
113                   `TransactionID` INT(9) NOT NULL AUTO_INCREMENT , `Complete` TEXT NOT NULL DEFAULT 'no' ,
114                   `Total` DECIMAL(10,2) NOT NULL, PRIMARY KEY (`TransactionID`))";
115         mysqli_query($connection,$query);
```

The total cost of each transaction then needs to be added to this table during the completion of every transaction. This means that I need to alter the query updating this table upon the completion of each transaction to include the total transaction price.

First, I have to implement the total of the current transaction into a session variable:

```
142             /* Output total cost of transaction */
143             echo "<h3 align='center'>Total: $total</h3>";
144             $_SESSION['total'] = $total;
145
```

Then I need to update the query updating the transaction history table

Before:

```
188     /*
189      *  End the current transaction
190      */
191     if ($returnValue == true) {
192         /* Set current transaction to complete */
193         $tableName = "transactionhistory_" . $companyName;
194         $query = "UPDATE $tableName SET Complete = 'yes' WHERE TransactionID = '$transactionID'";
195         mysqli_query($connection, $query);
196
```

After:

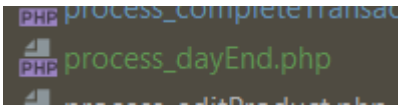
```php
/*
 *  End the current transaction
 */
if ($returnValue == true) {
    /* Set current transaction to complete */
    $transactionTotal = $_SESSION['total'];
    $tableName = "transactionhistory_" . $companyName;
    $query = "UPDATE $tableName SET Complete = 'yes', Total = '$transactionTotal' WHERE TransactionID = '$transact
    mysqli_query($connection, $query);

```

I now need to include a button “end the day” in the drop-down menu that will end the current day. This button will clear the transaction history of a company and all of the transaction tables for that particular company. This allows the daily total to be correct every day. This was not initially as I had planned for this to work, I had planned to use the system date and time to reset the day at 12 am every day. However, it then occurred to me that some business days may end after 12 am, and therefore their daily total would be incorrect. Allowing the user to end their day manually prevents this.

Creating a button “end of day”

```php
<form action="process_dayEnd.php">
    <button type="submit" name="endOfDay">End Of Day</button>
</form>
```

I will now need to create a processing file, process_dayEnd.php



I will now need to create a MySQLi query Selecting all from the current company’s transaction history table

```php
/* Select all from company transaction history */
$companyName = $_SESSION['companyName'];
$query = "SELECT * FROM transactionhistory_$companyName";
$queryResult = mysqli_query($connection, $query);
```

Using the results of this query, I will then need to delete all of the transaction tables for that specific company. I will do this using a for loop and a MySQLi query.

```php
/* Calculate number of rows returned by query */
if($queryResult == false){ // If no rows are returned by function
    $numRows = 0;
}
else{
    $numRows = mysqli_num_rows($queryResult); // Returns number of rows
}

/* Delete all transaction tables */
for ($i=0; $i<$numRows; $i++){
    $rowResult = mysqli_fetch_assoc($queryResult);
    $transactionNumber = $rowResult['TransactionID']; // Get transaction ID

    /* Delete transaction tables for this company */
    $tableName = "transaction_".$transactionNumber."_".$companyName; // Calculate name of table
    $query = "DROP TABLE `$tableName`"; // Delete the table from DB Query
    mysqli_query($connection,$query);
}
```

I will then include within this for loop the deletion of the transaction from the transaction history table for this company after the current contents of the for loop.

```php
/* Delete all transaction tables */
for ($i=0; $i<$numRows; $i++) {
    $rowResult = mysqli_fetch_assoc($queryResult);
    $transactionNumber = $rowResult['TransactionID']; // Get transaction ID

    /* Delete transaction tables for this company */
    $tableName = "transaction_" . $transactionNumber . "_" . $companyName; // Calculate name of table
    $query = "DROP TABLE `$tableName`"; // Delete the table from DB Query
    mysqli_query($connection, $query);

    /* Delete corresponding row in transaction history table */
    $tableName = "transactionhistory_" . $companyName;
    $query = "DELETE FROM $tableName WHERE TransactionID = '$transactionNumber'";
    mysqli_query($connection, $query);
}
```

I will then test this by ending the current day as shown in **end of day test.mp4**

I got the following errors:

Warning: Undefined variable $_SESSION in **C:\xampp\htdocs\POSSY\process_dayEnd.php** on line 7

Warning: Trying to access array offset on value of type null in **C:\xampp\htdocs\POSSY\process_dayEnd.php** on line 7

I have not started the session in this file, causing the errors on line 7.

```php
<?php
/* Open DB connection */
include "database_connect.php";
$connection = openConnection();

/* Select all from company transaction history */
$companyName = $_SESSION['companyName'];
$query = "SELECT * FROM transactionhistory_$companyName";
$queryResult = mysqli_query($connection, $query);
```

To fix this, I will start the session before line 7 is run:

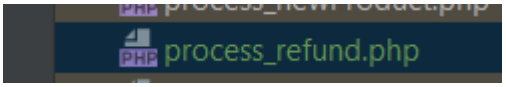
```php
<?php
/* Open DB connection */
include "database_connect.php";
$connection = openConnection();

/* Start the session */
session_start();

/* Select all from company transaction history */
$companyName = $_SESSION['companyName'];
$query = "SELECT * FROM transactionhistory_$companyName";
$queryResult = mysqli_query($connection, $query);
```

Creating a refund page

Creating an input box to enter the amount to refund and a submit button. When submitted, the user will be redirected to process_refund.php:

```
<form action = "process_refund.php">
    <label>Refund Amount:<input type="number" step="0.01" value = 0 name="refundAmount"></label>
    <button type="submit" name="submit">Submit</button>
</form>
```

Creating the process_refund.php file:



In this file, I will first need to open the session and open the connection to the database:

```
/* Open DB connection */
include "database_connect.php";
$connection = openConnection();

/* Start the session */
session_start();
```

I will then need to use $_GET to get the value of the amount to refund.

```
/* Retrieve amount to refund */
$refundAmount = $_GET['refundAmount'];
```

I will then need to access the name of the company from a session variable:

```
/* Retrieve companyName */
$companyName = $_SESSION['companyName'];
```

I will then need an SQL query inserting the value of the refund as a negative value into the company's transaction history table with the transaction marked as complete.

```
/* Add refund as a negatively valued transaction to the transaction history table */
$refundAmount = -$refundAmount; // Set refundAmount to be a negative value
$tableName = "transactionhistory_".$companyName;
$query = "INSERT INTO $tableName(Complete,Total) VALUES('yes','$refundAmount')";
mysqli_query($connection,$query);
```

A toolbar

A toolbar must be placed on this page to easily allow the user access to other pages. I will copy & paste the toolbar from the pointOfSale.php page and alter the current page from the point-of-sale page to the drop-down menu page as being the active page.

I will then also copy & paste this toolbar into the drop-down menu as well for the same reasons.

Calculating & Displaying the daily total

I will need to calculate the daily total. To do this, I will need a for loop to loop through each row in the transaction history table where the transaction is complete and add this to a running total variable.

Therefore, I will first need to open a connection to the database and start the session.

```
28    /* Open DB connection */
29    include "database_connect.php";
30    $connection = openConnection();
31
32    /* Start the session */
33    session_start();
```

I will then need to retrieve the company name from the session variable

```
35    /* Retrieve company name */
36    $companyName = $_SESSION['companyName'];
```

I will then need to create an SQL query selecting all from the transaction history table for this company where the transaction is complete

```
38  /* Retrieve data from transaction history table where transaction is complete */
39  $tableName = "transactionhistory_".$companyName; // Calculate table name
40  $query = "SELECT * FROM $tableName WHERE Complete = 'yes'";
41  $transactionhistoryResult = mysqli_query($connection,$query);
```

I will now need to calculate the number of rows returned by that table.

```
43    /* Calculate number of returned rows */
44    $numRows = 0;
45    if($transactionhistoryResult == false){
46        $numRows = 0;
47    }
48    else{
49        $numRows = mysqli_num_rows($transactionhistoryResult);
```

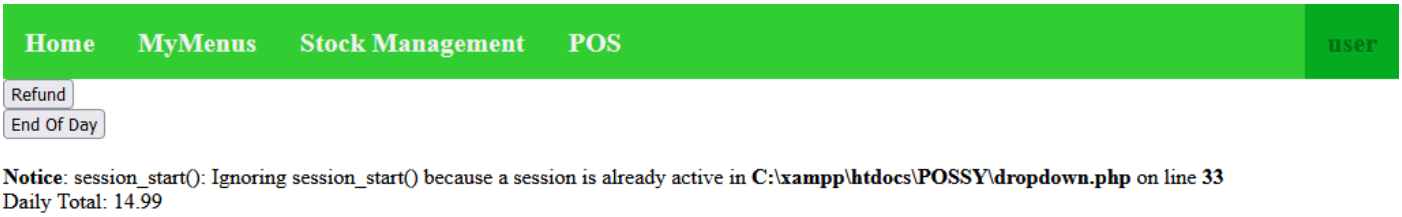
I will then need to use a for loop, looping through each row returned by the query and incrementing the total by the total cost of each completed transaction

```
52  /* Loop through each row of the results to the query, adding to the variable total the total cost of each
53   * transaction in the transaction history table that is complete
54   */
55  $total = 0;
56  for ($i=0; $i<$numRows; $i++){
57      $tableRow = mysqli_fetch_assoc($transactionhistoryResult);
58
59      $total = $total + $tableRow['Total'];
60  }
```

Finally, I will need to output the daily total to the system operator

```
62    echo "Daily Total: ".$total;
63
```

Upon running the program, I received the following error:

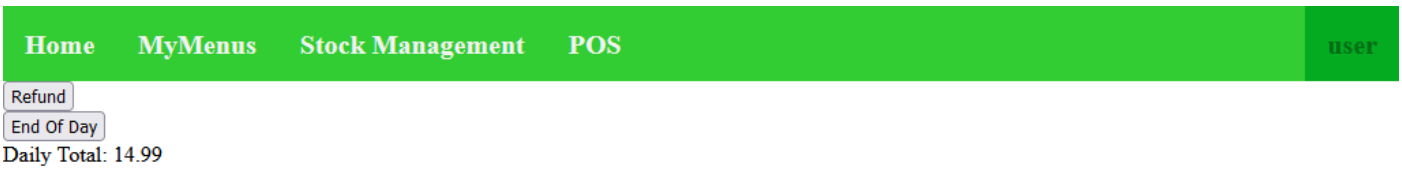



```
14    <?php session_start();
15    $username = $_SESSION['username']; // get username from login process
16    ?>
```

I had declared the session to be started twice.

```
32    /* Start the session */
33    session_start();
```

To fix this, I will remove the session being started on line 33



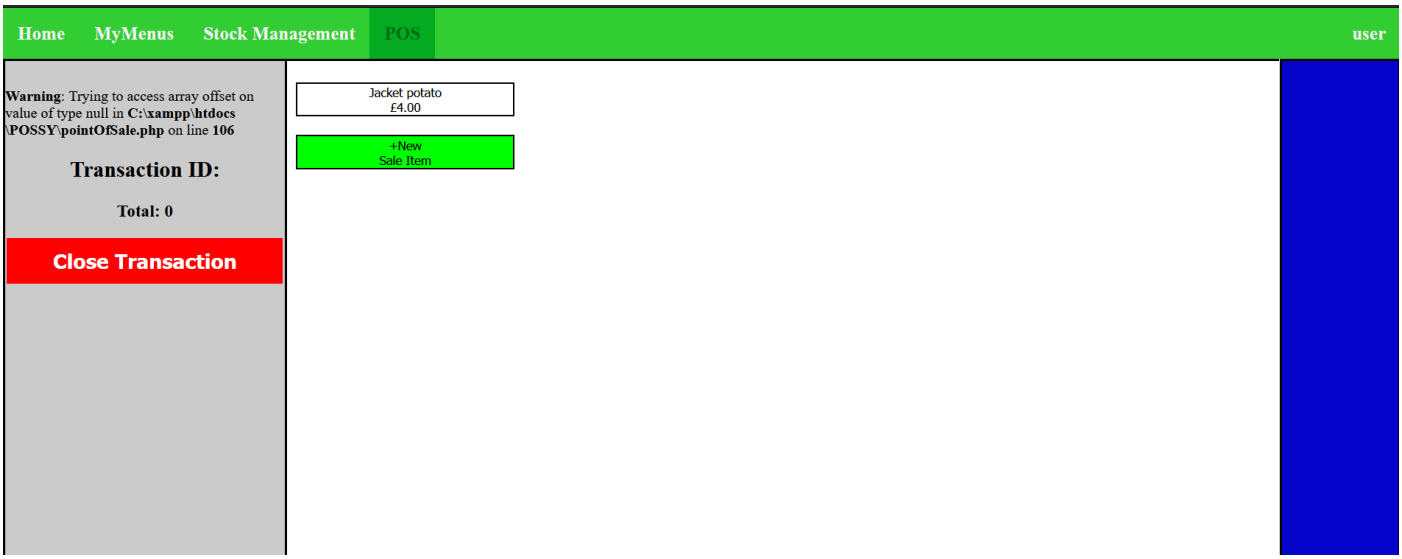
According to the current transactions within the transaction history table of the database, the daily total is working as intended.

	TransactionID	Complete	Total
Edit Copy Delete	1	yes	4.00
Edit Copy Delete	2	yes	8.00
Edit Copy Delete	3	yes	4.00
Edit Copy Delete	4	yes	-0.01
Edit Copy Delete	5	yes	-1.00
Edit Copy Delete	6	no	0.00

Click the drop-down arrow to toggle column's visibility.

The daily total is also reset when the day is ended manually by the system operator, as shown in **resetting the daily total.mp4** It was not initially planned for this to be a manual process, however with some thought I decided that it is better this feature is manual. I decided this because some venues may close after 12am, therefore the day would be ended before they had closed, and transactions would be lost if the process was completed automatically.

I then returned to the point-of-sale system intending to start my end of stage testing when I was met with an error in the transaction column:



Once a day has been ended, there are no rows within the transaction history table of the database. This means that a transaction ID cannot be found, and therefore a new transaction cannot be opened. To fix this, after a day is ended, there must be a MySQLi query inserting a row into the transaction history table, and a new transaction table must also be created for this transaction ID.

First, I need to insert the new transaction into the transaction history table:

```
39  /* Create a new transaction in the transaction history table,
40   * and create a new transaction table for the transaction.
41   */
42
43  /* Insert new transaction table name into transaction history table */
44  $tableName = "transactionhistory_".$companyName;
45  $query = "INSERT INTO $tableName(Complete) VALUES('no')";
46  mysqli_query($connection,$query);
```

I then need to use an MySQLi query to find the ID of this new transaction:

```
48    /* Retrieve transaction ID */
49    $query = "SELECT TransactionID FROM $tableName WHERE Complete = 'no'";
50    $queryResult = mysqli_query($connection,$query);
51    $transactionHistoryRow = mysqli_fetch_assoc($queryResult);
52    $transactionID = $transactionHistoryRow['TransactionID'];
53
```

I then need to create a new transaction table using the Id as $transactionID

```
54    /* Create Initial Transaction Table */
55    $tableName = "transaction_".$transactionID."_".$companyName;
56    $query = "CREATE TABLE `users`.`$tableName` ( `SaleItemID` INT(9) NOT NULL AUTO_INCREMENT ,
57                  `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT NOT NULL ,
58                   PRIMARY KEY (`SaleItemID`))";
59    mysqli_query($connection,$query);
```

End of stage testing

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Does the drop-down menu web page generate correctly sized	Press your username on the task-bar	Drop-down menu webpage opens and is sized correctly	Drop-down menu webpage opens and is sized correctly See **end of stage 6 testing.mp4**
2a	Does the "Refunds" button redirect the user to the refunds page?	Press the "Refunds" button	Refunds page opens	Refunds page opens See **end of stage 6 testing.mp4**
3a	Does the Refunds page generate correctly sized?	Press the "Refunds" button	Refunds page opens and is sized correctly	Refunds page opens and is sized correctly See **end of stage 6 testing.mp4**
4a	Does refunding an amount deduct the amount from the daily total?	1.00	The daily total is reduced by £1.00	The daily total is reduced by the correct amount See **end of stage 6 testing.mp4**
4b	Does refunding a decimal amount deduct the amount from the daily total?	0.01	The daily total is reduced by £0.01	The daily total is reduced by the correct amount See **end of stage 6 testing.mp4**
4c	Can you enter invalid values into the refund input box?	0.001	Invalid number – refund is not completed	The daily total cannot be reduced as the value is not valid See **end of stage 6 testing.mp4**
5a	Able to create a new transaction after ending the day	Press any sale item button on the point-of-sale system web page 2 times Then close the transaction	2 of the selected product is added to the transaction The transaction is closed The daily total is updated by the correct amount	2 of the product is added to the transaction The transaction is closed The daily total is updated by the correct amount See **end of stage 6 testing.mp4**

This stage was completed much faster than expected. The development process was fast and efficient with only minor bugs, and all features intended to be implemented at this stage were implemented with all intended uses working.

This stage has met the following success criteria:

- Ability to refund a transaction as outlined in requirements 2.4

As this stage is fully functional, with no major issues, it is unlikely that I will return to this section. If I were to return to this section, I would first like to produce a drop-down menu as was initially intended, rather than a further webpage. I would also like to have my refund page displayed in a calculator form as shown in my original designs, rather than a simple input box as it is currently. Failing this, I would like to style the buttons and input boxes away from the default styling.

Stage 7 – Point-of-sale system – DESIREABLES (4 + 2 hours)

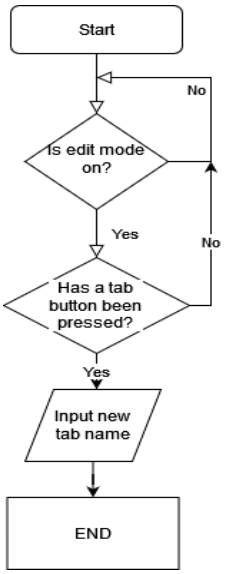
The desirable user requirements for the point-of-sale system will be included in this stage. This includes the use of increment buttons to add multiple of the same sale item at one time, a search bar to decrease the length of each transaction, the ability to customize items added to a transaction, and the use of categories to enable more sale items to be displayed at any one time via the use of tabs on the user interface.

Features/functionality:

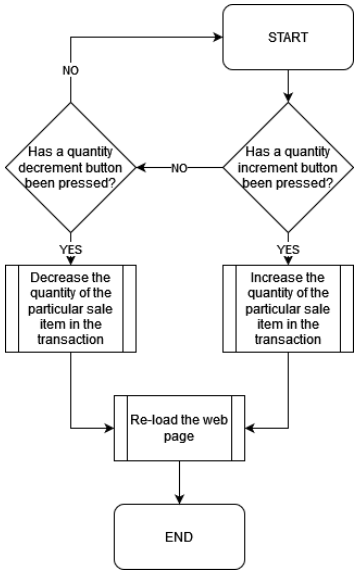
1. Implementation of categories into the menus database
2. Implementation of categories when creating a menu sale item
3. Implementation of tabs along the right-hand side of the user interface
4. An algorithm to only show relevant items when a tab is pressed in the point-of-sale system
5. Ability to edit the name of each tab in the point-of-sale system
6. Increment buttons next to each transaction item
7. Ability to set the sale item button to a custom image
8. Inclusion of the price of each sale item under the button
9. Inclusion of the name of each sale item under the button
10. A search bar to search for sale item buttons
11. Customise sale item contents*Indecisive on this feature

Test No.	Description	Type Of Test	Test Data	Expected Result	Actual Result
1a	Does pressing a tab button open a different set of sale item buttons?		Press “Food” tab	Sale item buttons categorized with “Food” are displayed	
2a	Can the name of tabs be edited?		Edit a tab to be named “test tab”	Tab renamed “test tab”	
2b	Does editing the name of a tab create a new category?		Edit a tab to be named “category”	A category should appear named “category”	
3a	Can a sale item be written into the search bar and pressed on to get added to the current transaction?		Enter “cheese sandwich” into the search bar and press on the result	A cheese sandwich should be added to the transaction	
4a	Can a custom image be set as a sale item button?		Upload image to sale item button via the edit menu	Button shows as an image	

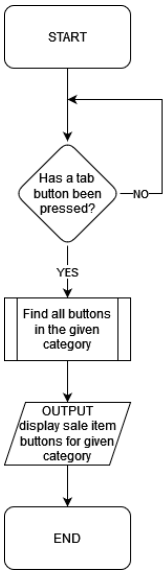
Flow Charts



This shows the ability to edit a category button name



This shows the process behind the increment decrement buttons



This shows the processes behind a category button being pressed

SQL Statements

SELECT SaleItemName, ID, Price FROM (company's menu table) WHERE Category = '(category of category button)'

SELECT SaleItemName, ID, Price FROM (company's menu table) WHERE Category LIKE '%(entry into search bar)%'

SELECT Quantity FROM (current transaction table) WHERE SaleItemName = '(given sale item name)'

Implementation of categories into menus database

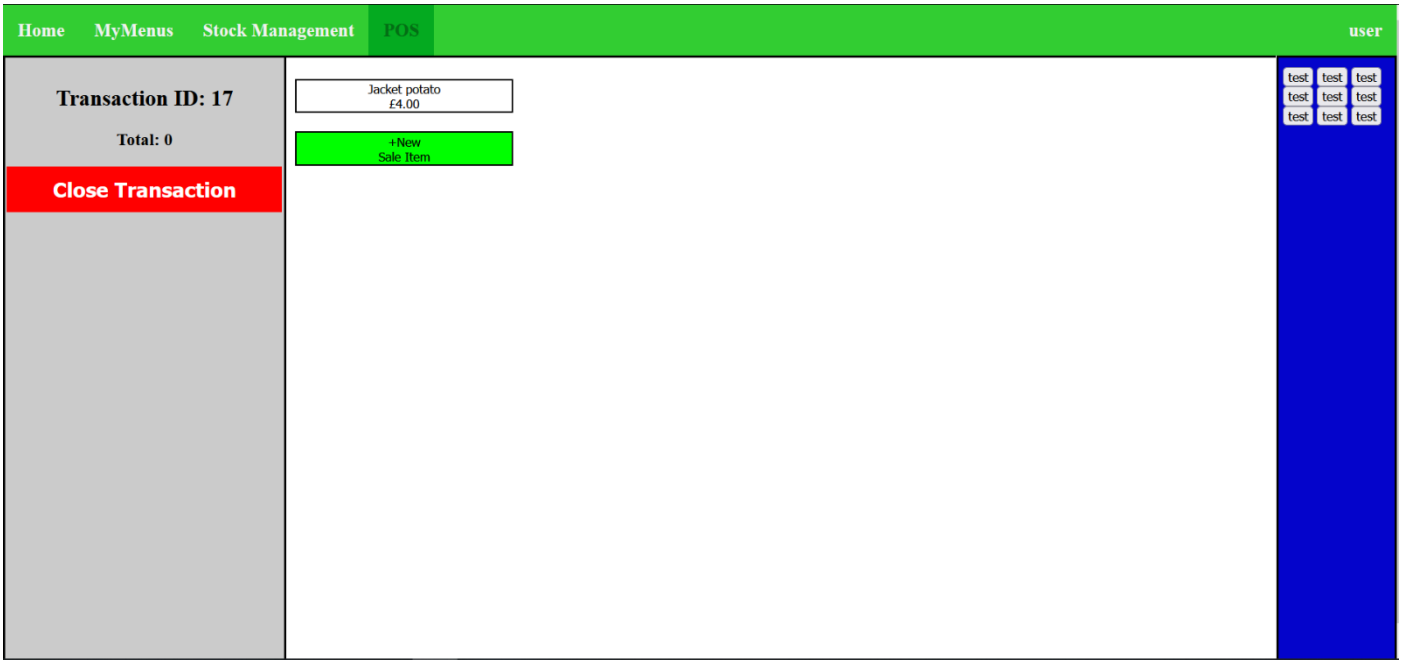
Categories were already implemented into the menus database in a previous stage when the database was created.

Implementation of categories when creating a menu sale item

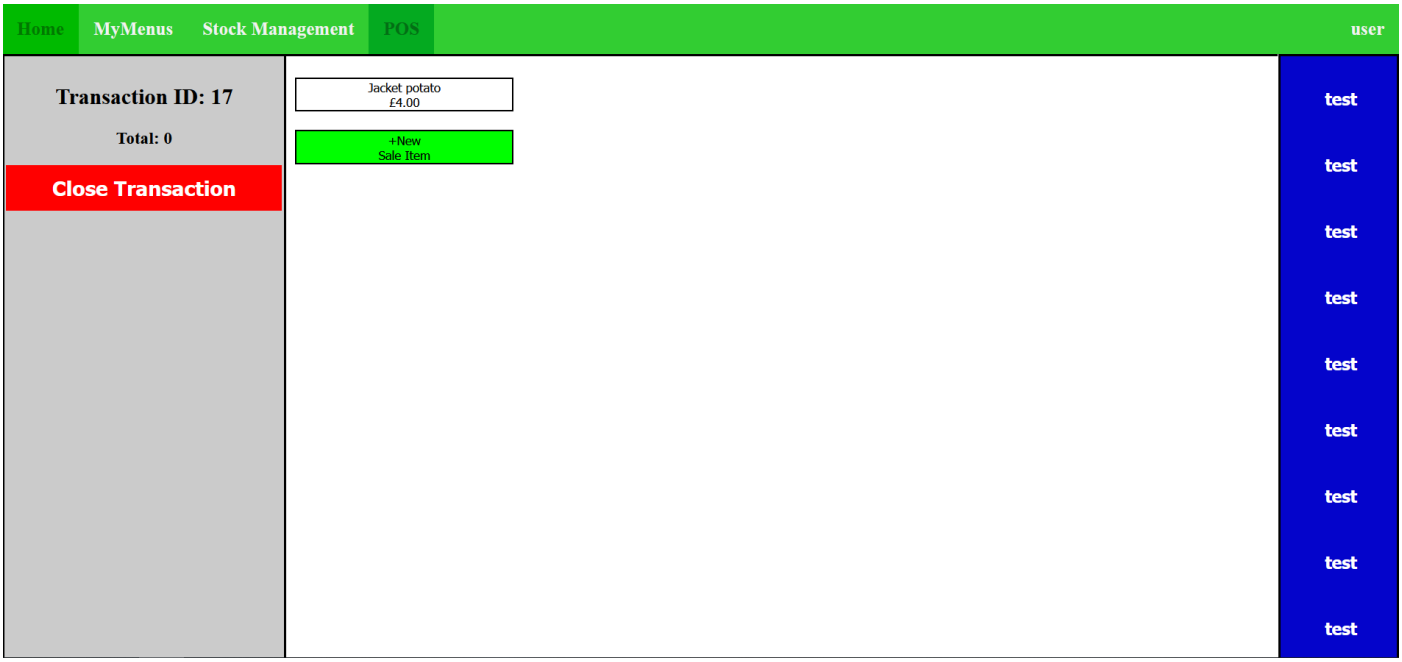
Categories when creating a sale item were already implemented in a previous stage when editing sale items was implemented

Implementation of tabs along the right-hand side of the point-of-sale system

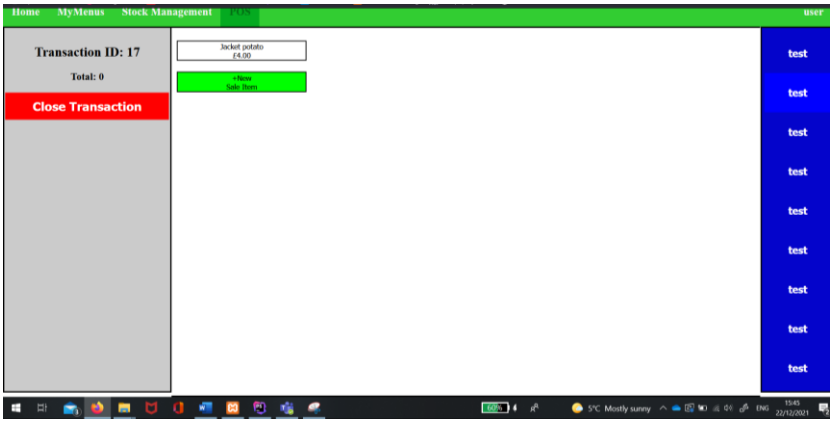
```
214     <!-- Category Column -->
215     <span class='rightGUI'>
216         <?php
217
218         /* Create 9 category buttons */
219         for($i = 0; $i<9; $i++)
220             echo "
221                 <button type='submit'>test</button>
222             ";
223
224         ?>
225
226     </span>
227
228 </div>
```



The correct number of buttons are displayed; however, they need to be styled to be the same colour blue as the background. The buttons also need to be larger so that they are displayed in a line rather than a grid.



This is good, but I would like the button to change colour when hovered over, making it clear when your mouse is hovering over one of the buttons.



I like that the buttons now change colour when hovered. However, the buttons do not span to the left-hand edge. Therefore, there is a gap in the colour change to the left-hand side when the user hovers over the button. This is not a huge problem; therefore, it will be left and perhaps fixed at a later stage if time permits.

Final styling:

```
<span class='rightGUI'>
    <style>
        .category{
            background-color: #0404cb;
            border: none;
            width: 100%;
            height: 11%;
            color: white;
            font-weight: bold;
            font-size: 18px;
        }

        .category:hover{
            background-color: blue;
        }
    </style>
```

I now need to label each button with the correct category. To do this, I will use an SQL query searching through the menu table of the current company. It will include the first eight unique categories it finds. If there are less than eight unique categories, the remaining category buttons will remain without a label.

I will first need to calculate the number of rows returned by the query:

```
246         /* Calculate number of rows returned by query */
247         $numRows = 0;
248         if($categoryResult == false){
249             $numRows = 0;
250         }
251         else{
252             $numRows = mysqli_num_rows($categoryResult);
253         }
```

Then alter the for loop from $i<8 to $i<$numRows:

```
255         /* Display category tabs */
256         for($i = 0; $i<$numRows; $i++) {
257 
258             echo "
259                 <button class='category' type='submit'>test</button>
260             ";
261         }
```

Then fetch the results of each row, outputting the name of the category inside each button

```
255         /* Display category tabs */
256         for($i = 0; $i<$numRows; $i++) {
257 
258             $categoryRow = mysqli_fetch_assoc($categoryResult); // fetch the next row from results
259 
260             $tabName = $categoryRow['Category'];
261             echo "
262                 <button class='category' type='submit'>$tabName</button>
263             ";
264         }
```

I now need to create a HTML form with action to return to the point-of-sale page. I then need a hidden input with the value of the category name to post the category name to the file when a category button is pressed.

```
237         // Display default tab
238         echo "
239             <form action='pointOfSale.php'>
240             <input type='hidden' value='all' name='category'>
241             <button class='category' type='submit'>All</button>
242             </form>";
```

```
255     /* Display category tabs */
256     for($i = 0; $i<$numRows; $i++) {
257
258         $categoryRow = mysqli_fetch_assoc($categoryResult); // fetch the next row from results
259
260         $tabName = $categoryRow['Category'];
261         echo "
262             <form action='pointOfSale.php'>
263             <input type='hidden' value='$tabName' name='category'>
264             <button class='category' type='submit'>$tabName</button>
265             </form>
266         ";
267     }
268     ?>
```

I now need to change which sale items are displayed depending on which categories are pressed. To do this, I need to change the query run when selecting items from the menu table of the current menu. Currently, only all of the buttons are shown in the point-of-sale system (default). The query needs to include a where clause if the chosen category is not the default. I will do this using if statements.

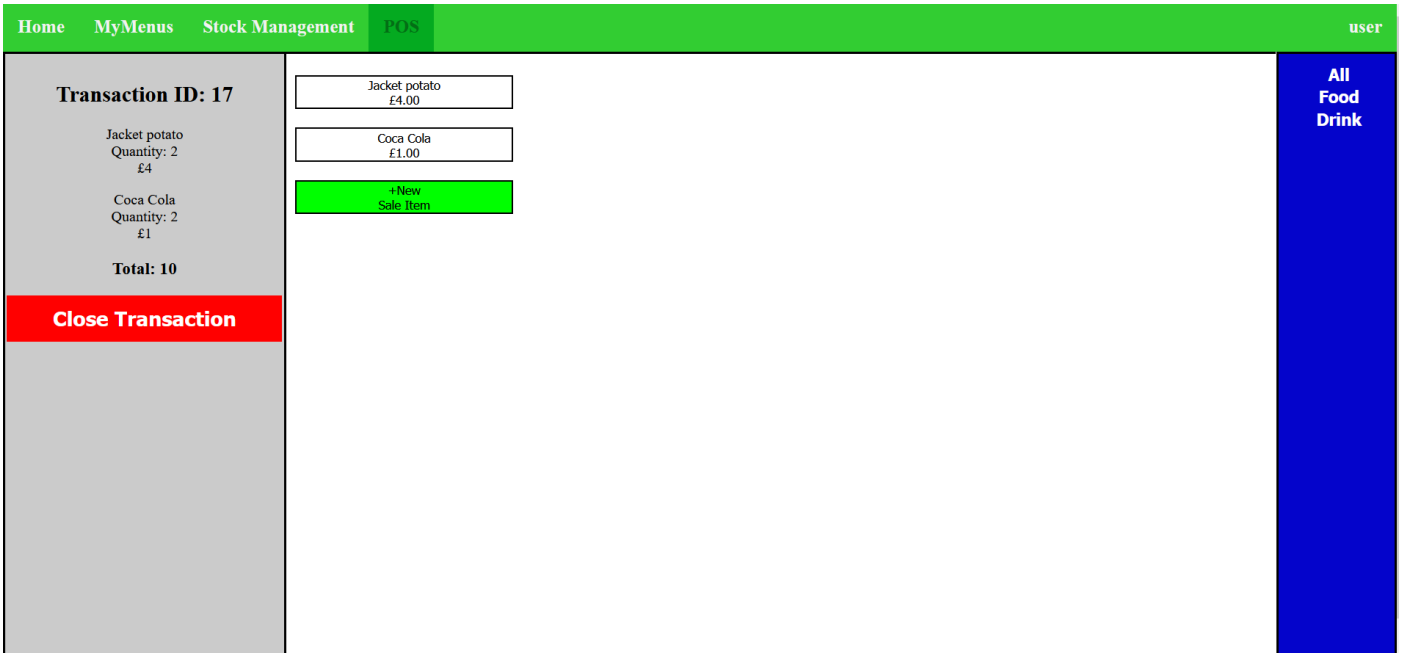
Before:

```
187         $query = "SELECT saleItemName,ID,price FROM $tableName";
188         $result = mysqli_query($connection, $query);
189         $numRows = mysqli_num_rows($result);
```

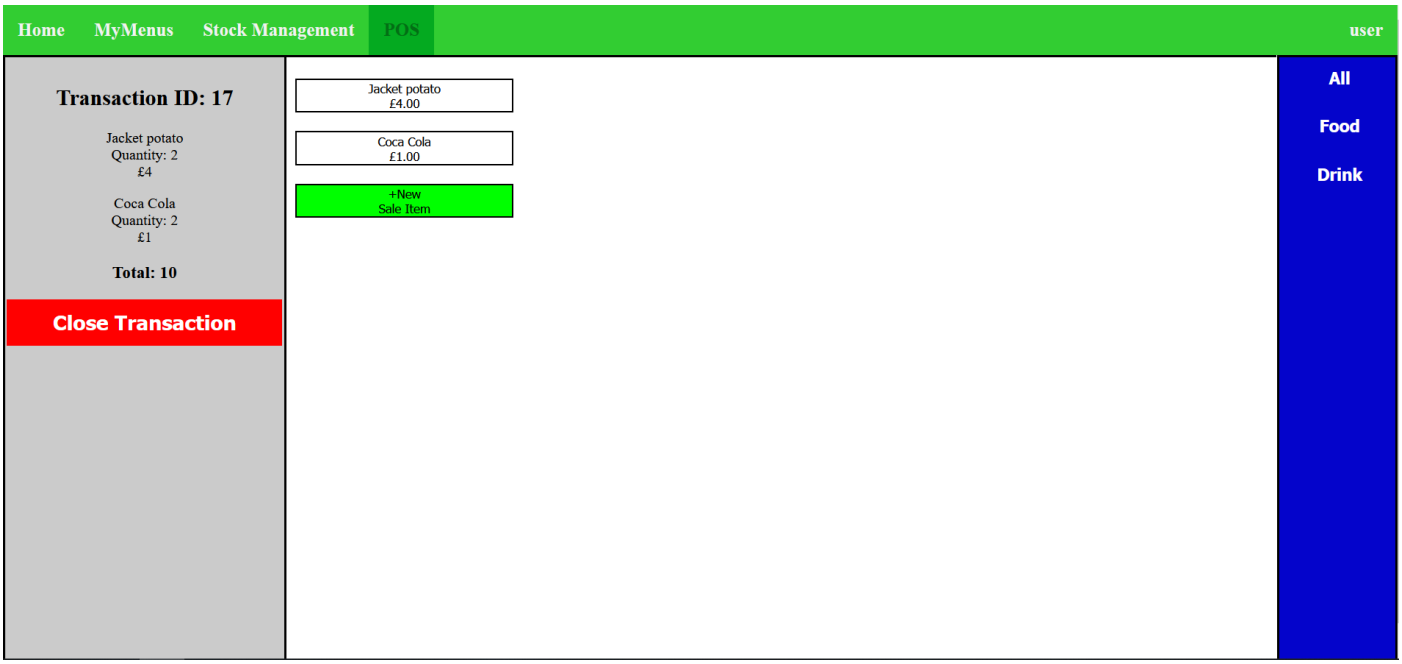
After:

```
187     /* Checks if a category has been selected that is not all */
188     if(!empty($_GET) && $_GET['category'] != "all"){
189         $category = $_GET['category'];
190         // Only show results for given category
191         $query = "SELECT saleItemName,ID,price FROM $tableName WHERE Category = '$category'";
192     }
193     else{
194         // show results for all categories
195         $query = "SELECT saleItemName,ID,price FROM $tableName";
196     }
197     /* Run query selecting which sale items to display */
198     $result = mysqli_query($connection, $query);
199     $numRows = mysqli_num_rows($result);
```

The tests (category button testing) went really well – the category buttons function as intended and are really well laid out.



However, I would like to add a little spacing between the category buttons.



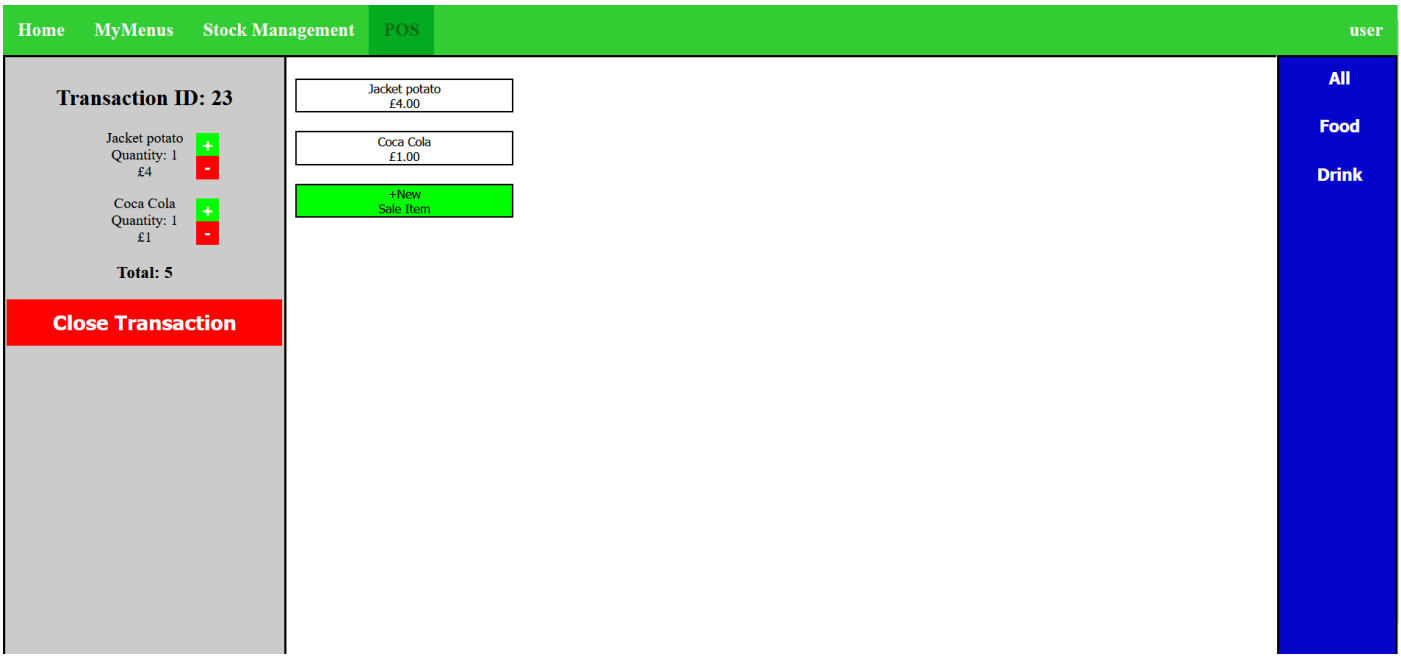
Ability to edit the name of each tab

With the way that tabs were implemented, this feature would not make sense; therefore, it will no longer be implemented into the project

Increment buttons next to each transaction item

My design implemented these as input buttons, with each increment increasing the value of the number of that sale item added to the transaction. However, this will not work because the database will not be updated each time the input field is incremented/ decremented. Therefore, when the transaction is closed, the incorrect product items will be contained within the transaction. To fix this, I will instead implement this using a + and a – submit button, which will reload the page and update the database with each click of the buttons.

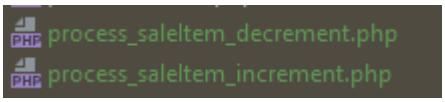
First, I will design the + and – buttons. I will implement these next to the item quantity value in the transaction column next to each sale item.



Then I need to send the system operator to a processing page when the buttons are clicked using a HTML form

```
        echo "
        <p align='center'>$saleItemName <br>
        Quantity: $quantity <br>
        $saleItemCost</p>

        <style>
        .button{
        margin: 0px;
        left: 50%;
        position: absolute;
        border: none;
        font-weight: bold;
        color: white;
        width: 25px;
        height: 25px;
        font-size: 16px;
        }

        </style>

        <form action='process_saleItem_increment'><button class='button'
        style='transform: translate(-2200%, -265%); background-color: lime;' type='submit'>+</button></form>
        <form action='process_saleItem_decrement'><button class='button'
        style='transform: translate(-2200%, -165%); background-color: red' type='submit'>-</button></form>";

    }
```

I now need to create both files:



I will now need to add hidden input fields into both forms containing the sale item that each increment/ decrement button represents.

```
149     <!-- Increment Button -->
150     <form action='process_saleItem_increment'>
151     <button class='button' style='transform: translate(-2200%, -265%);
152     background-color: lime;' type='submit'>+</button>
153     <input type='hidden' value='$saleItemName' name='saleItem'>
154     </form>
155
156     <!-- Decrement button -->
157     <form action='process_saleItem_decrement'>
158     <button class='button' style='transform: translate(-2200%, -165%);
159     background-color: red' type='submit'>-</button>
160     <input type='hidden' value='$saleItemName' name='saleItem'>
161     </form>";
```

Process increment click

I will first need to create a connection to the database and open the current session

```php
<?php
/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

session_start();
```

I then need to store needed session variables into standard variables

```php
session_start();
$transactionID = $_SESSION['transactionID'];
$companyName = $_SESSION['companyName'];
```

I then need to retrieve the passed item name value into a standard variable from a $_GET variable

```php
$saleItemName = $_GET['saleItem'];
```

I then need to access the transaction table for the current transaction, selecting the quantity of the sale item in the transaction where the sale item = the current sale item

```php
$tableName = "transaction_".$transactionID."_".$companyName;
$query = "SELECT Quantity FROM $tableName WHERE SaleItemName = '$saleItemName'";
$result = mysqli_query($connection,$query);
```

I then need to fetch the quantity found from the query and add 1 to that value

```php
// update the quantity of that sale item in the transaction
$row = mysqli_fetch_assoc($result);
$quantity = $row['Quantity'];
$newQuantity = $quantity + 1;
```

I now need to run an SQL query, updating the database table of the current transaction to the new quantity value

```php
$query = "UPDATE $tableName
              SET Quantity = '$newQuantity'
              WHERE SaleItemName = '$saleItemName' ";
mysqli_query($connection,$query);
```

Process decrement click

I first need to copy the code for the increment processing but change the addition to a negative.

```php
<?php
/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

session_start();
$transactionID = $_SESSION['transactionID'];
$companyName = $_SESSION['companyName'];
$saleItemName = $_GET['saleItem'];

$tableName = "transaction_".$transactionID."_".$companyName;
$query = "SELECT Quantity FROM $tableName WHERE SaleItemName = '$saleItemName'";
$result = mysqli_query($connection,$query);

// update the quantity of that sale item in the transaction
$row = mysqli_fetch_assoc($result);
$quantity = $row['Quantity'];
$newQuantity = $quantity - 1;

$query = "UPDATE $tableName
            SET Quantity = '$newQuantity'
            WHERE SaleItemName = '$saleItemName' ";
mysqli_query($connection,$query);
```

I now need to make sure that if the quantity is equal to zero when decremented, the sale item is removed from the current transaction via an SQL DELETE query.

```php
if($newQuantity >0) {
    $query = "UPDATE $tableName
                SET Quantity = '$newQuantity'
                WHERE SaleItemName = '$saleItemName' ";
    mysqli_query($connection, $query);
}
else{
    $query = "DELETE FROM $tableName WHERE SaleItemName = '$saleItemName'";
    mysqli_query($connection,$query);
}
```

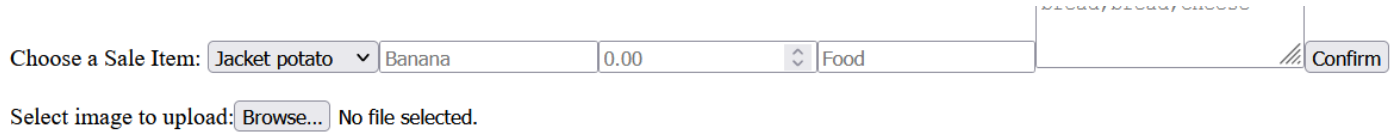
This can be seen working in **testing increment decrement.mp4**

Including image behind sale item

First, I need to style the sale item button to include a background image

```
65      button.saleItem{
66          display: inline-grid;
67          width: 22%;
68          height: 15%;
69          border: 2px solid black;
70          background-color: white;
71          margin: 10px;
72          text-align: center;
73          background: #f0f0f0 url(<?php echo $img; ?>);
74      }
```

I now need to include the ability for the user to upload an image to a sale item on creation. To do this, I need to create an image upload box



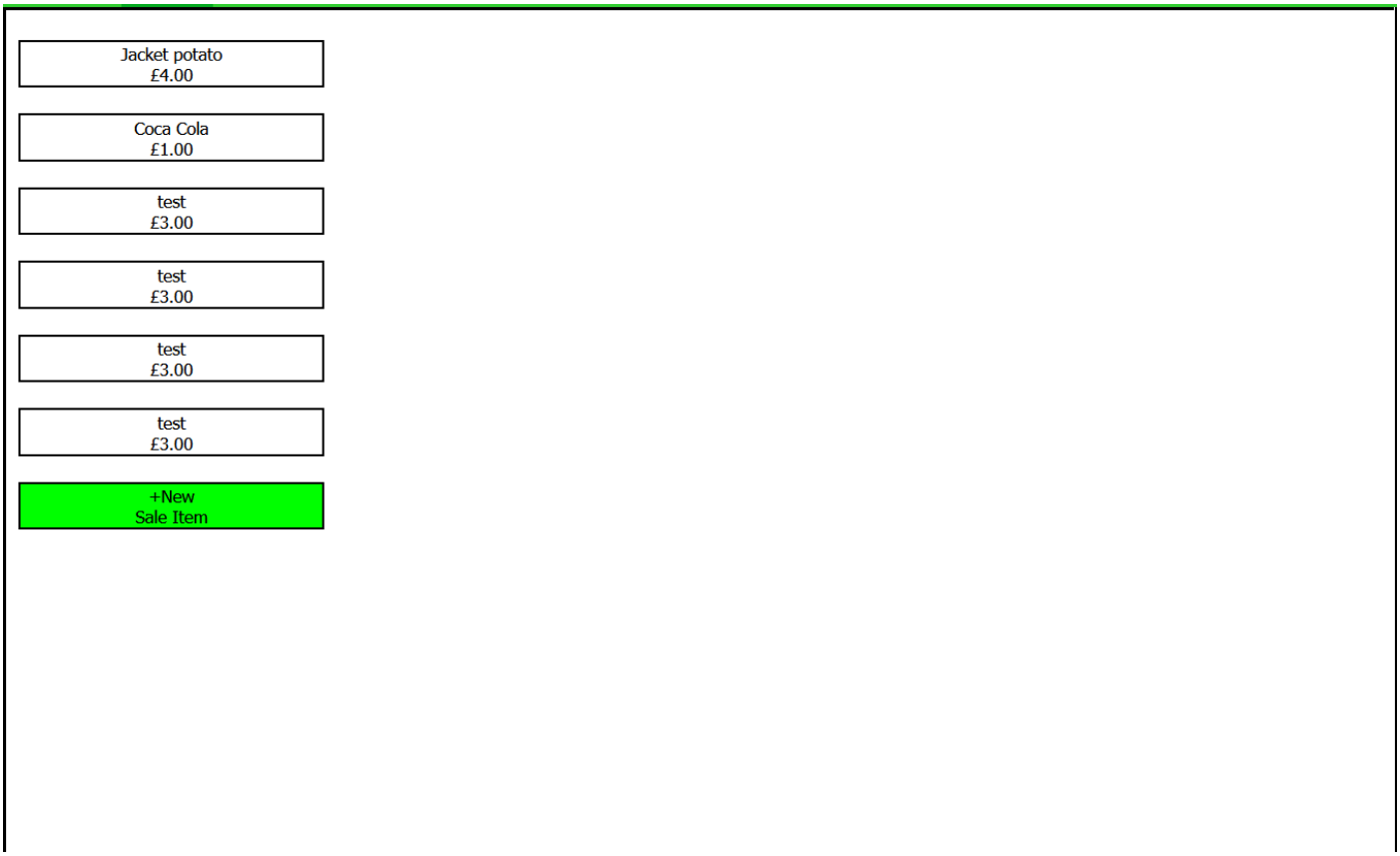

```
50      /* Input image */
51      echo "
52          <br><br>Select image to upload:<input type='file' name='img' accept'image/*'>
53
```

This is when I realised that I am unable to include this feature. I do not know how to upload the image files into my project, and you cannot store files (that are not .json) in a database table; therefore, I do not know how to implement this feature. I will therefore revert my code to a version prior to the implementation of this feature. This feature will no longer be included in my project.

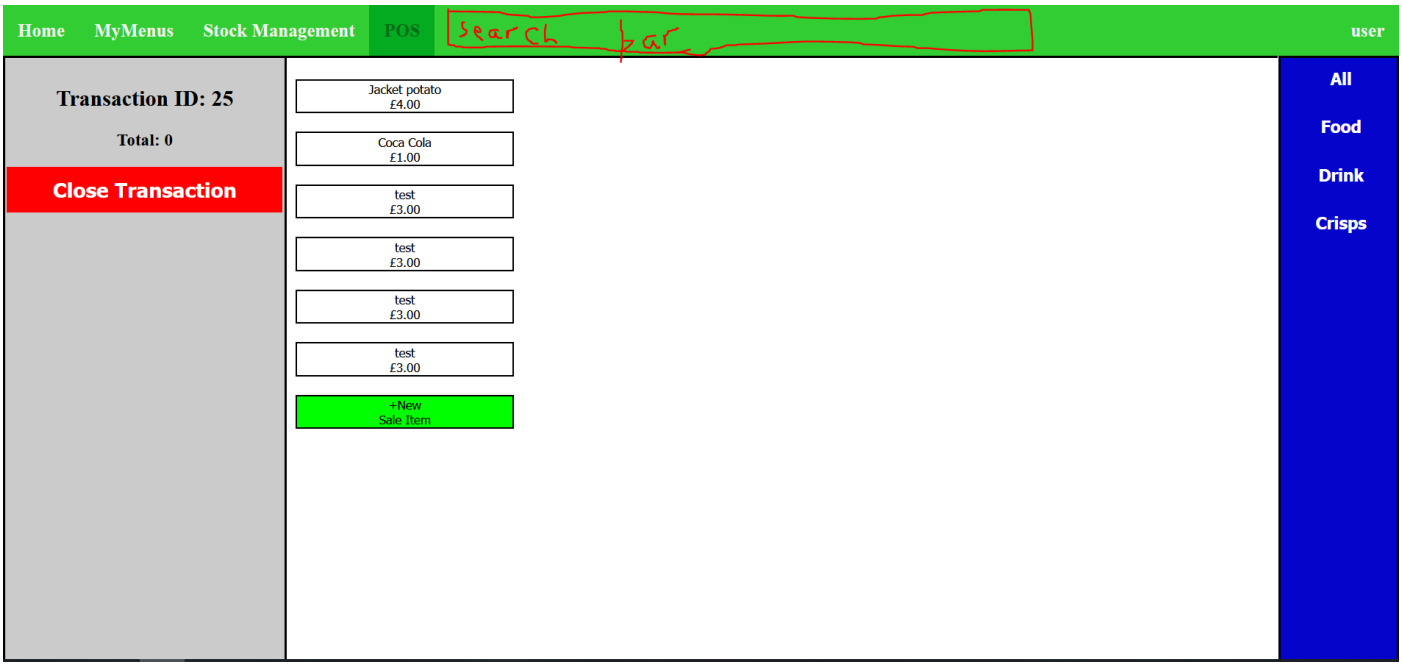
Features 8 & 9 will also not be included as these were only needed due to images being included in the sale item buttons.

Creating a search bar

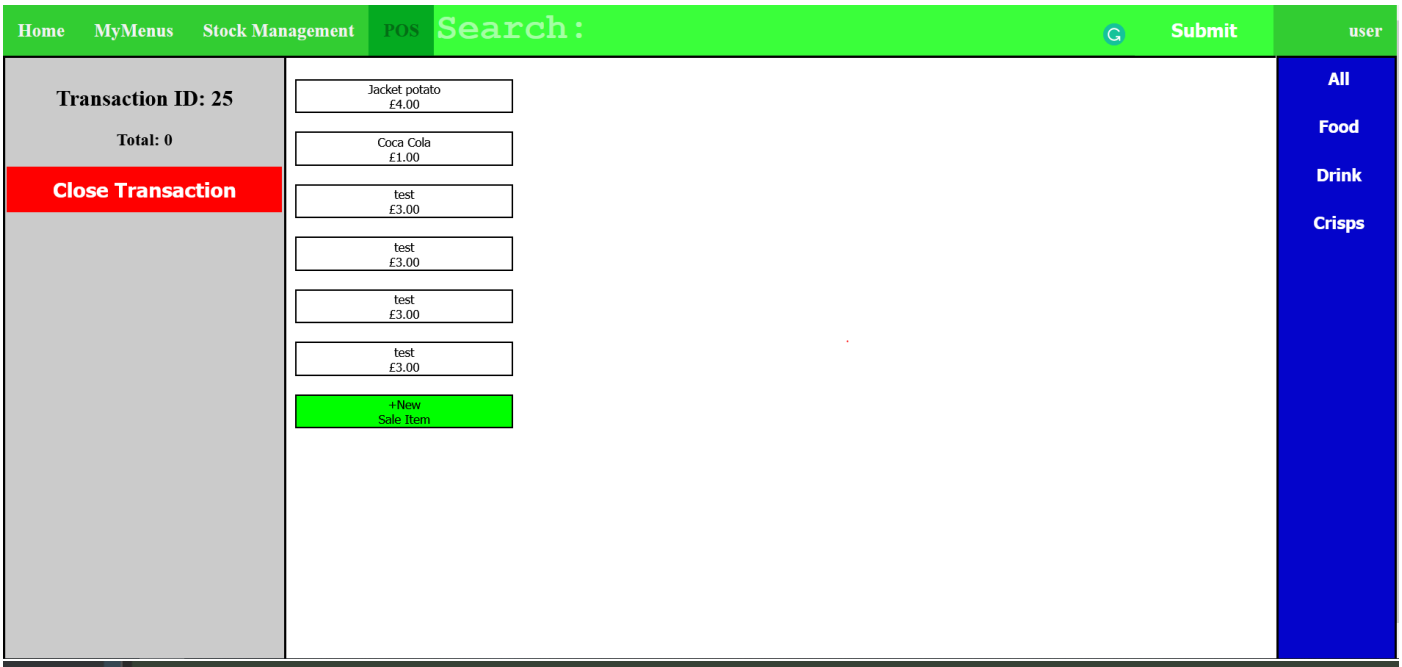
My original design places the search bar within the top of the “mid-GUI” of the point-of-sale page.



Although this would be a good area to place the search bar, I believe it will now be better placed within the taskbar, after the ‘POS’ button (area highlighted below)



I need to begin by creating a text area in the toolbar which can input text, alongside a submit button

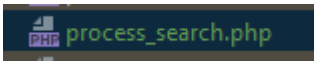



```
.searchSubmit{
    width: 10%;
    height: 55px;
    background-color: #3aff3a;
    border: none;
    outline: none;
    resize: none;
    color: white;
    font-weight: bold;
    font-size: 20px;
    text-align: center;
    float: left;
    overflow; hidden:

}

.searchSubmit:hover{
    background-color: #00ba00;
}

</style>
<form style='height: 0; width: 0; display: inline'  action='process_search.php'>
<textarea class='searchBar' name='searchBar' placeholder='Search:'></textarea>
<button class='searchSubmit' type='submit'>Submit</button>
</form>
";
```

```
    echo "
    <style>
    .searchBar{
        width: 50%;
        height:50px;
        background-color: #3aff3a;
        border: none;
        outline: none;
        resize: none;
        color: white;
        font-weight: bold;
        font-size: 40px;
        margin: 0 auto;
        float: left;
        overflow: hidden;
    }

    .searchBar:hover{
    background-color: #00ba00;
    }
```

I

now need to create the search processing page



First, I need to open the database connection and start a new session in the new file

```
session_start();

/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();
```

I then need to retrieve the search parameters using $_GET

```
/* Get the search value */
$searchValue = $_GET['searchBar'];
```

I now need to create a MySQLi query, searching for the results of the current search in the current menu table. The result should be entered into a session variable $_SESSION[‘searchReturnObject’]

```
/* Search for query in current menu table */
$tableName = "menu_".$companyName."_".$menuID;
$menuQuery = "SELECT * FROM $tableName WHERE saleItemName LIKE '%$searchValue%'";
$_SESSION['searchReturnObject'] = mysqli_query($connection,$menuQuery);
```

I now need a session variable to know whether I should use the search function currently or not. To do this, I will use a session variable $_SESSION[‘shouldSearch’] which is false when the program should not search via the value in the search bar and true when it should. This was not initially planned in my design. The reason to include this is that if this is not included and the user searches using the search bar, the current search will continue to happen until the search bar is emptied and the submit search button is pressed. This is not user friendly as if any other buttons that change the displayed sale items are pressed, they will not work until the search bar is emptied and submitted.

Therefore, I will first need to initate the session variable upon the user logging in

```
38      $_SESSION['shouldSearch'] = false; // set POS search bar as not actively searching
```

I will now need to create an if statement, stating that if $_SESSION[‘shouldSearch’] is true only sale items with the entered character values contained within are displayed in the point-of-sale system. This will take precedence over the category buttons.

Here I realised that it was infact not a result object that I needed. Instead, I just needed the query to be ran. I therefore removed the following line from the search processing file:

```
18  $_SESSION['searchReturnObject'] = mysqli_query($connection,$menuQuery);
```

And turned the following variable containing the query into a session variable:

```
17  $menuQuery = "SELECT * FROM $tableName WHERE saleItemName LIKE '%$searchValue%'";
```

```
17  $_SESSION['menuQuery'] = "SELECT * FROM $tableName WHERE saleItemName LIKE '%$searchValue%'";
```

I can now pass the query into the pointOfSale file

```
275     /* Checks if a category has been selected that is not all */
276     if($_SESSION['shouldSearch'] == true){
277         // Retrieve query from process_search.php
278         // only shows research for entered search
279         $query = $_SESSION['menuQuery'];
280     }
```

I now need to set $_SESSION[‘shouldSearch’] = true when the process_search.php file is used

```
19    /* Set search to be true when the POS system is next opened */
20    $_SESSION['shouldSearch'] = true;
```

And I need to set $_SESSION[‘shouldSearch’] = false when the search query has been entered into the pointOfSale.php file

```
275         /* Checks if a category has been selected that is not all */
276         if($_SESSION['shouldSearch'] == true){
277             // Retrieve query from process_search.php
278             // only shows research for entered search
279             $query = $_SESSION['menuQuery'];
280             $_SESSION ['shouldSearch'] = false; // set POS system not to search on next reload
281                                                 // unless search is re-submitted
282         }
```

This feature is now fully implemented. I am happy with the design of the search bar, and I am happy that it functions well and as expected.

Customise menu items

I have decided that this feature will take a lot of time to implement as bugs are likely, and it will require a lot of implementation, e.g. further database tables that were an oversight. Therefore, I will not be implementing this feature.

End of stage testing

Test No.	Description	Type Of Test	Test Data	Expected Result	Actual Result
1a	Does pressing a tab button open a different set of sale item buttons?		Press "Food" tab	Sale item buttons categorized with "Food" are displayed	Sale items categorized as Food are displayed See **end of stage 7 testing.mp4**
2a	Can the name of tabs be edited?	This was not included	Edit a tab to be named "test tab"	Tab renamed "test tab"	This was not included
2b	Does editing the name of a tab create a new category?	This was not included	Edit a tab to be named "category"	A category should appear named "category"	This was not included
3a	Can a sale item be written into the search bar and pressed on to get added to the current transaction?		Enter "cheese sandwich" into the search bar and press on the result	Cheese sandwich should be added to the transaction	Cheese sandwich is added to the transaction See **end of stage 7 testing.mp4**
4a	Can a custom image be set as a sale item button?	This was not included	Upload image to the sale item button via the edit menu	Button shows as an image	This was not included

This stage consisted of quite a few problems – mainly oversights. I have not managed to include all features that were intended. This includes the ability to edit the name of each tab in the point-of-sale system – due to the way that this feature was implemented, it did not make sense to be able to edit the name of tabs as the tab names are the categories of items themselves, therefore renaming should not be necessary. If you want a tab to be a different name, change the category of those products. This means that this feature is still included in the project; however, it was not implemented as initially intended. The category tabs are less customisable than I initially intended them to be, as sale items are now assigned a tab according to their category.

The ability to set sale item buttons to have custom images was not implemented. This was because I was unable to upload the image file or store the image file in a database. This meant that only hard-coded images would have worked, which was not the aim of this feature. Therefore this feature was removed and will no longer be included in the project.

Due to the image feature not being included, the price and name under the sale item button were not included either. The price and sale item name are already included within the sale item button; therefore, it is not necessary to implement twice. The price and name **under** the button were initially intended to make the name and price of a sale item visible if an image was included within the sale item button. As both features are available in the projects current state, the user-friendliness of the project will not be affected.

Custom menu items were not included. I was already indecisive on this feature during the design phase of this project. With the current layout of my project, I am unsure that this will fit well in the transaction column of the point-of-sale system. Other than this, there were many oversights in this feature, for example, the need for further database tables. One database table would be needed for each sale item to keep track of which products a particular sale item contains and which it does not. Therefore, I will not include this feature in my project.

I did not meet any of the success criteria during this stage, as all included features were desirable rather than essential. Hence, it was no problem not including a couple of features in this stage due to oversights, and my stakeholders have agreed that they are happy for these features to not be present in this version of my project. Despite several hiccups during this stage, the search bar to search for sale items turned out perfect, and my stakeholders and I are very happy with both the design and the functionality of the search bar.

Stage 8 – Stock management system – DESIREABLES (3 hours)

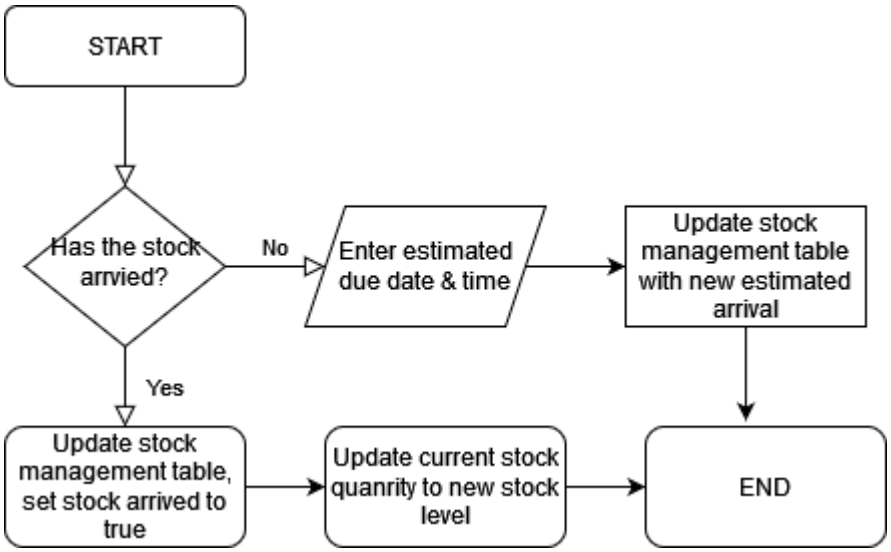
This stage of the project includes some of the finishing touches to allow for greater customizability of the entire project. This includes the introduction of new forms of low stock alerts via emails & SMS, the ability to set a due date for the ordering of stock, allowing for automatic stock updates & the ability to resend low stock alerts after a 20minute interval if the initial alert appeared poorly timed.

Features/functionality:

1. Implementation of date & time stock is due into the stock management database table
2. Implementation of the amount of stock due into the stock management database table
3. Implementation of the ability to enter the amount, date, and time when a low stock alert appears
4. A function to update the order status of a product in the stock management database table
5. A function to update the due date of a product in the stock management database table
6. A stock arrived yes/no alert GUI
7. Implementation of the resending of alert messages

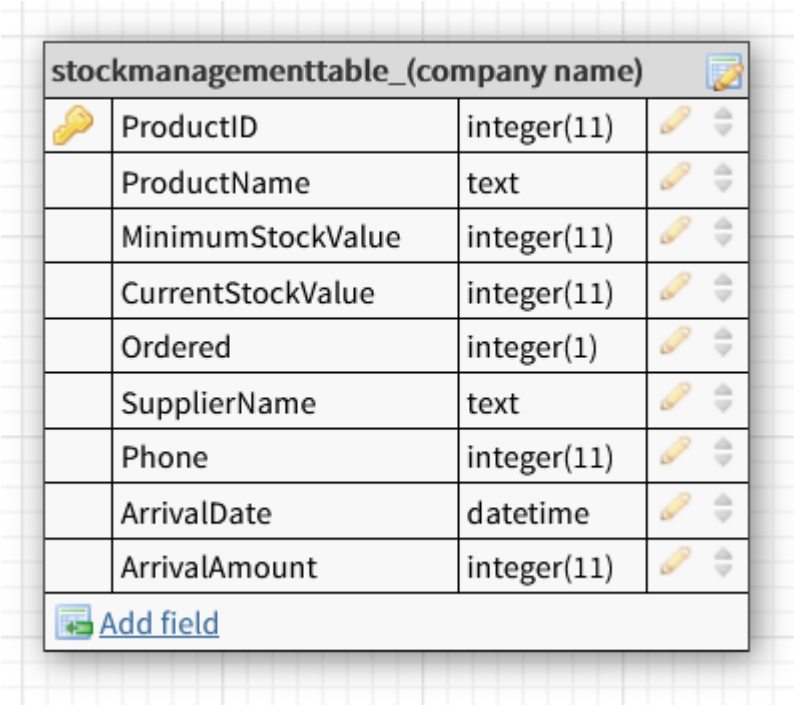
Test No.	Description	Test Data	Expected Result	Actual Result
1a	When a low-stock alert is sent, the alert can be deferred for 20 minutes	Press “unordered” when a low-stock alert appears	Alert reappears after 20 minutes	
2a	Does the order status get set to unordered after stock has been set as arrived	Press “yes” in stock arrived alert	The stock management database field for the order status of the particular item is set to unordered	
3a	When a low-stock alert is sent, the estimated due time & date can be entered	Trigger a low stock alert by setting a product’s due date & time to the current date & time + 1 minute Open the point-of-sale web page 11:00 01/23/2050 Click ordered	Stock is set as being ordered	
3b	When a due date & time is entered, the database table is updated to the correct value		The due date & time is updated correctly in the company’s stock management database table	

Flow Charts



This shows the process behind stock arrival alerts. Whenever this code is called, the system will check if the stock of a product has arrived. If it has, the stock arrived field for this product within the stock management table will be set to true, and the current stock quantity of the product inside this table will be updated to the new stock quantity. Otherwise, the user will be asked to enter the new estimated due date and time, and the estimated arrival date and time field within the stock management table will be updated with the new estimated due date and time of the stock.

Database tables



stockmanagementtable_(company name)

ProductID	integer(11)	
ProductName	text	
MinimumStockValue	integer(11)	
CurrentStockValue	integer(11)	
Ordered	integer(1)	
SupplierName	text	
Phone	integer(11)	
ArrivalDate	datetime	
ArrivalAmount	integer(11)	

Add field

Implementing Data & Time Into Stock Management database table

Each product needs a date & time that it is due to arrive back in stock after being ordered. This will be done by altering the creation of a company's stock management database during sign-up to include the date and time as a column of the database.

Before:

```
/* Create A Stock Management Table For This Company */
mysqli_query($connection,
    query: "CREATE TABLE `users`.`$tableName` ( `ProductID` INT NOT NULL AUTO_INCREMENT ,
            `ProductName` TEXT NOT NULL , `MinimumStockValue` INT NOT NULL , `CurrentStockValue` INT NOT NULL,
            `Ordered` TEXT NOT NULL DEFAULT 'off' ,`SupplierName` TEXT NULL DEFAULT NULL , `Phone` INT(11) NULL,
             PRIMARY KEY (`ProductID`)); ");
```

After:

```
/* Create A Stock Management Table For This Company */
mysqli_query($connection,
    query: "CREATE TABLE `users`.`$tableName` ( `ProductID` INT NOT NULL AUTO_INCREMENT ,
            `ProductName` TEXT NOT NULL , `MinimumStockValue` INT NOT NULL , `CurrentStockValue` INT NOT NULL,
            `Ordered` TEXT NOT NULL DEFAULT 'off' ,`SupplierName` TEXT NULL DEFAULT NULL , `Phone` INT(11) NULL,
            `ArrivalDate` DATETIME NULL, PRIMARY KEY (`ProductID`)); ");
```

Implementing Arrival stock amount Into Stock Management database table

Each product needs the amount of stock that it is due after being ordered. This will be done by altering the creation of a company's stock management database table during sign-up to include the ArrivalAmount in a column of the database.

Before:

```
/* Create A Stock Management Table For This Company */
mysqli_query($connection,
    query: "CREATE TABLE `users`.`$tableName` ( `ProductID` INT NOT NULL AUTO_INCREMENT ,
            `ProductName` TEXT NOT NULL , `MinimumStockValue` INT NOT NULL , `CurrentStockValue` INT NOT NULL,
            `Ordered` TEXT NOT NULL DEFAULT 'off' ,`SupplierName` TEXT NULL DEFAULT NULL , `Phone` INT(11) NULL,
            `ArrivalDate` DATETIME NULL, PRIMARY KEY (`ProductID`)); ");
```

After:

```
/* Create A Stock Management Table For This Company */
mysqli_query($connection,
    query: "CREATE TABLE `users`.`$tableName` ( `ProductID` INT NOT NULL AUTO_INCREMENT ,
            `ProductName` TEXT NOT NULL , `MinimumStockValue` INT NOT NULL , `CurrentStockValue` INT NOT NULL,
            `Ordered` TEXT NOT NULL DEFAULT 'off' ,`SupplierName` TEXT NULL DEFAULT NULL , `Phone` INT(11) NULL,
            `ArrivalDate` DATETIME DEFAULT NULL, `ArrivalAmount` INT DEFAULT NULL, PRIMARY KEY (`ProductID`)); ");
```

Creating a new account & company to check this is working: **testing SM table.mp4**. The test went as expected, the correct columns are included in the database table; therefore, I can now continue onto the next feature.

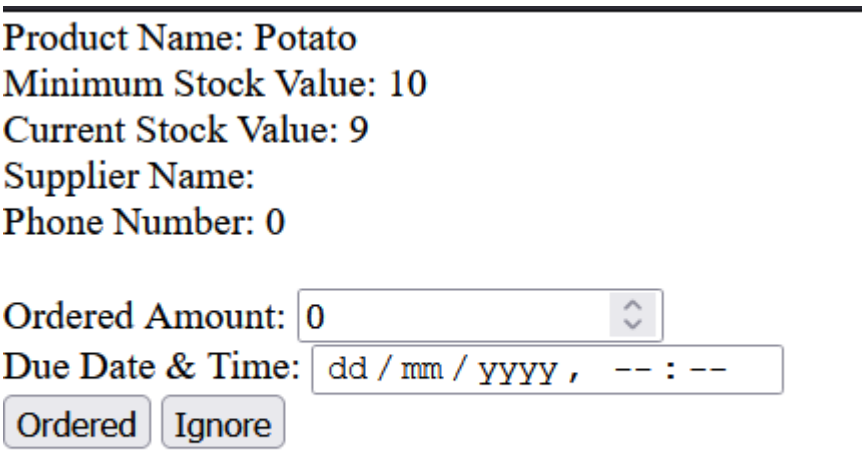
Implement stock due date, time, and amount into low stock alert

At this stage, I noticed that my designs for the low-stock alert had used two input boxes, one for time due and another for date due. This should not be the case. The first input box should be the amount of stock due, and the second box should be the date & time due.

As mentioned previously, this alert will also be its own web page as I could not create it using a pop-out box as was initially intended.

First, I will need to include both the amount of stock due & data and time input boxes into the low stock alert web page:

```
/* Output:
 Time & Date due input box
 * Amount due input box
 * Ordered form submit button
*/
echo "
    <form style='display: inline' action='process_lowStock.php'>
    <br><br><label>Ordered Amount: <input type='number' step='1' value='0'></label><br>
    <label>Due Date & Time: <input type='datetime-local'></label><br>
    <button type='submit'>Ordered</button> <!-- Output submit button -->
    </form>
    ";
```



I then realised that I had not named the input boxes, and that is needed to process the data; therefore, I named each input box:

```
echo "
    <form style='display: inline' action='process_lowStock.php'>
    <br><br><label>Ordered Amount: <input type='number' name='orderAmount' step='1' value='0'></label><br>
    <label>Due Date & Time: <input type='datetime-local' name='dateTime'></label><br>
    <button type='submit'>Ordered</button> <!-- Output submit button -->
    </form>
    ";
```

I can now use the process_lowStock.php file to process the inputs. Processing first includes retrieving the values orderAmount and dateTime using $_GET methods

```php
/* Fetch dateTime and orderAmount from lowStock.php */
$dateTime = $_GET['dateTime'];
$orderAmount = $_GET['orderAmount'];
```

I now need to check that the date and time is greater than the current date and time. To get the current date and time, I will use php's date function and format it correctly:

```php
/* Validate date & time */
date_default_timezone_set( timezoneId: "GMT");
$currentDateLong = date( format: "c");
$currentDateShort = substr($currentDateLong, offset: 0, length: 16);
```

I now need to check that this date is greater than the date entered.

```php
/* Check that the entered date & time is greater than today's date & time*/
if($dateTime > $currentDateShort){
    /* Update the SM table */
    $query = "UPDATE ".$tableName." SET Ordered = 'on' WHERE ProductID = ".$_SESSION['ProductID']."";
    mysqli_query($connection,$query);
}
```

I now need to check that the order amount is greater than zero

```php
/* Check that the entered date & time is greater than today's date & time*/
if($dateTime > $currentDateShort AND $orderAmount > 0){
    /* Update the SM table */
    $query = "UPDATE ".$tableName." SET Ordered = 'on' WHERE ProductID = ".$_SESSION['ProductID']."";
    mysqli_query($connection,$query);
}
```

And I need to output an error if the conditions are not met

```php
/* Check that the entered date & time is greater than today's date & time*/
if($dateTime > $currentDateShort AND $orderAmount > 0){
    /* Update the SM table */
    $query = "UPDATE ".$tableName." SET Ordered = 'on' WHERE ProductID = ".$_SESSION['ProductID']."";
    mysqli_query($connection,$query);
}
else{
    echo "error! Invalid input"; // Output error
```

I then need to change the query, to include updating the currentDate and orderAmount.

```php
    $query = "UPDATE $tableName SET Ordered = 'on', ArrivalDate = '$dateTime', ArrivalAmount = '$orderAmount'
              WHERE ProductID = '$productID'";
```

I then completed some testing, shown in **testing dateTime.mp4**. I noticed that the ArrivalDate is stored incorrectly in the stock management table.

←T→	ProductID	ProductName	MinimumStockValue	CurrentStockValue	Ordered	SupplierName	Phone	ArrivalDate	ArrivalAmount
☐ Edit Copy Delete	1	Potato	10	9	on		0	2022-12-17 10:00:00	10

The ArrivalDate column contains seconds – whereas it should only contain hours and minutes. Although this was not planned, it does not matter as the seconds will always be 00, and it is impossible to remove this from the database. Therefore, I just need to consider that the seconds exist in the database when selecting data from this table.

Including due date & arrival stock quantity into product edit web page

First, I will need to include input boxes into the existing web page:

Product Name: Carrot
Current Stock: 100
Product Name: 10
On order:
Supplier Name: Supplier
Phone Number: 07999123456
Done

For both date (including time) and quantity due to arrive.

```
    <label for='dueDate'>Restock Date:</label>
    <input type='datetime-local' name='dueDate'>

    <br>

    <label for='Quantity due:'>Restock Amount:</label>
    <input type='number' name='restockQuantity' placeholder='0' step='1'>

    <br>
    <button type='submit'>Done</button>
</form>";
```

Product Name: Carrot
Current Stock: 100
Product Name: 10
On order:
Supplier Name: Supplier
Phone Number: 07999123456
Restock Date: dd / mm / yyyy , -- : --
Restock Amount: 0
Done

This is good, but it will not display the current Restock date and restock amount values in the database, which could lead to restocks being overwritten accidentally. I had not initially planned for this. However, it is something that came to mind when thinking about the potential bugs of the solution. I will include a MySQLi query to fix this, retrieving the value of date and restock amount from the company's stock management table in the database. The retrieved values will then be used as the values in the input boxes.

```php
/* Open DB connection */
include "database_connect.php";
$connection = openConnection();

/* GET EXISTING RESTOCK DATE FROM STOCK MANAGEMENT TABLE */
$tableName = 'stockmanagementtable_'.$_SESSION['companyName'];
$productID = $_GET['ProductID'];
$query = "SELECT ArrivalDate,ArrivalAmount FROM $tableName WHERE ProductID = '$productID' ";
$queryResult = mysqli_query($connection,$query);

/* Enter fetched values into variables */
$tableRow = mysqli_fetch_assoc($queryResult);
$arrivalDate = $tableRow['ArrivalDate'];
$arrivalAmount = $tableRow['ArrivalAmount'];
```

```php
    <!-- restock due date input field of type date -->
    <label for='dueDate'>Restock Date:</label>
    <input type='datetime-local' name='dueDate' value='$arrivalDate'>

    <br>

    <!-- restock quantity input field of type number -->
    <label for='Quantity due:'>Restock Amount:</label>
    <input type='number' name='restockQuantity' placeholder='0' step='1' value='$arrivalAmount'>
```

Once submitted, the data will be sent to the processing page before being entered into the database. The processing page does not yet enter the updated values into the database table. Therefore the next step is to edit the existing query on the processing page:

```php
$query = "UPDATE $tableName
          SET ProductName='$productName',MinimumStockValue='$minimumStockValue',CurrentStockValue='$currentStockValue',
          Ordered='$ordered',SupplierName='$supplierName',Phone='$phoneNumber'
          WHERE productID='$productID'";
mysqli_query($connection,$query);
```

To also include the values of the amount due to be restocked and the date the restock is due.

```php
$arrivalDate = $_GET['dueDate'];
$arrivalAmount = $_GET['restockAmount'];

$query = "UPDATE $tableName
          SET ProductName='$productName',MinimumStockValue='$minimumStockValue',CurrentStockValue='$currentStockValue',
          Ordered='$ordered',SupplierName='$supplierName',Phone='$phoneNumber', ArrivalDate='$arrivalDate',
          ArrivalAmount='$arrivalAmount' WHERE productID='$productID'";
mysqli_query($connection,$query);
```

This will work as intended; however, there is currently no validation. This means that negative stock values could be entered. To fix this, I will make sure that the arrival amount is greater than 0 and the arrival date is greater than today's date using an if statement.

```php
/* Validate date & time */
date_default_timezone_set( timezoneId: "GMT");
$currentDateLong = date( format: "c");
$currentDateShort = substr($currentDateLong, offset: 0, length: 16);

/* Check that the entered date & time is greater than today's date & time*/
if(($arrivalDate > $currentDateShort OR $arrivalDate == null) AND ($arrivalAmount > 0 OR $arrivalDate == null)) {
    $query = "UPDATE $tableName
            SET ProductName='$productName',MinimumStockValue='$minimumStockValue',CurrentStockValue='$currentStockValue',
            Ordered='$ordered',SupplierName='$supplierName',Phone='$phoneNumber', ArrivalDate='$arrivalDate',
            ArrivalAmount='$arrivalAmount' WHERE productID='$productID'";
    mysqli_query($connection, $query);
}
else{
    echo "ERROR! Invalid date or restock quantity";
```

Testing of this feature can be seen in **testing edit product datetime.mp4**

The test was short, as the following error was produced:

Warning: Undefined array key "restockAmount" in **C:\xampp\htdocs\POSSY\process_editProduct.php** on line **17**
ERROR! Invalid date or restock quantity

```php
17    $arrivalAmount = $_GET['restockAmount'];
```

The array key should be restockQuantity, not restock amount

```php
17    $arrivalAmount = $_GET['restockQuantity'];
```

I then tested this again, as seen in **testing edit product datetime 2.mp4**

The page is functioning well, and the correct values are updated in the database. However, the user is not automatically redirected to the stock management system. I need to include a header change at the bottom of the processing file redirecting the user to fix this.

```php
/* Check that the entered date & time is greater than today's date & time*/
if(($arrivalDate > $currentDateShort OR $arrivalDate == null) AND ($arrivalAmount > 0 OR $arrivalDate == null)) {
    $query = "UPDATE $tableName
            SET ProductName='$productName',MinimumStockValue='$minimumStockValue',CurrentStockValue='$currentStockValue',
            Ordered='$ordered',SupplierName='$supplierName',Phone='$phoneNumber', ArrivalDate='$arrivalDate',
            ArrivalAmount='$arrivalAmount' WHERE productID='$productID'";
    mysqli_query($connection, $query);
    header( header: 'Refresh: 0; URL=stockManagement.php');
    exit;
}
else{
    echo "ERROR! Invalid date or restock quantity";
    header( header: 'Refresh: 3; URL=stockManagement.php');
    exit;
```

The user is now redirected to the stock management page after processing

A stock arrived yes/no alert GUI

When stock is due to have arrived, the user will need to be alerted and will need to confirm whether the stock did in fact arrive or not. If the stock has arrived, the current stock value of that product will need to be updated. If the stock has not arrived, a new due date and quantity must be set.

I will first need to trigger the user to be re-directed to this web page if the current date and time are less than or equal to the due date and time of the arrival of new stock. First, I will need to create an SQL query selecting all rows from the current user's stock management table.

```
72      /* Check for re-stock having arrived */
73      $tableName = "stockmanagementtable_".$_SESSION['companyName'];
74      $query = "SELECT * FROM $tableName";
75      $stockManagementTableResult = mysqli_query($connection,$query);
76
```

I will then need to use an if statement and the built-in mysqli_num_rows() function to check how many rows the query result returned. This will be used within a for loop later in this feature.

```
77      /* Calculate the number of rows returned by query */
78      $numRowsReturnedByQuery = 0;
79      if ($stockManagementTableResult == false){
80          $numRowsReturnedByQuery = 0;
81      }
82      else{
83          $numRowsReturnedByQuery = mysqli_num_rows($stockManagementTableResult);
84      }
```

I will then need to calculate the current date and time to be compared with the date and time of due stock of products later in this feature in the format YYYY-MM-DD HH:MM

```
86      /* Calculate current date & time */
87      date_default_timezone_set( timezoneId: "GMT");
88      $currentDateLong = date( format: "c");
89      $currentDateShort = substr($currentDateLong,  offset: 0,  length: 16);
90
```

I will then need to create a for loop, searching through each result of the query using the built-in mysqli_fetch_assoc() function, and within this function, use an if statement checking that for all products whether the due date of a re-stock is less than or equal to the current date & time. If less than the current date and time, the user needs to be sent to the web page "hasStockArrived.php"

```php
for($i=0; $i<$numRowsReturnedByQuery; $i++) {
    /* Fetch new row of the query result */
    $row = mysqli_fetch_assoc($stockManagementTableResult);

    /* Fetch arrival date and time from the current row */
    $arrivalDate = $row['ArrivalDate'];

    /* if date&time is not null, check whether the date and time of stock due to arrive is less than the
       current date and time */
    if (($arrivalDate < $currentDateShort and $arrivalDate != null)) {

        header( header: 'refresh: 0; URL = hasStockArrived.php');// redirect to home page
        exit;
    }
}
```

This is good; however, no data about the product which should have arrived is sent to the hasStockArrived.php file. This means that the same code would have to be re-written in that file to determine which product to update the stock value or arrival date & time of. To prevent writing more code than necessary, I will pass values into the redirect URL to the hasStockArrived.php file, containing the ProductID, ProductName, ArrivalDate, and ArrivalAmount of the product.

```php
    if (($arrivalDate < $currentDateShort and $arrivalDate != null)) {

        /* Fetch ArrivalQuantity & ProductID & ProductName */
        $arrivalQuantity = $row['ArrivalAmount'];
        $productID = $row['ProductID'];
        $productName = $row['ProductName'];

        /* Calculate redirect URL */
        $redirectURL = "hasStockArrived.php?productID=".$productID."&productName=".$productName.
                       "&arrivalDate=".$arrivalDate."arrivalQuantity=".$arrivalQuantity;

        /* Redirect the user */
        header( header: 'refresh: 0; URL = '.$redirectURL);// redirect to home page
        exit;
    }
}
```

I then need to create the file “hasStockArrived.php”:

hasStockArrived.php

I was then looking for the design of this web page when I realised that a design was not made for this page during the design process. This page should contain the product that is due to have arrived, the quantity that is due to have arrived and then two buttons, a “Stock has arrived”, and a “stock hasn’t arrived” button.

First, I need to create the two buttons, the first button “stock has arrived” redirecting the user to the processing file “process_stockArrival.php” and a second button “stock hasn’t arrived”, redirecting the user to the stock management page of the particular sale item, where they will be able to update the due date and quantity of the product.

```php
<!-- submit user to processing page if stock has arrived -->
<form action='process_stockArrived.php'>
<button type='submit' value=''>Stock Has Arrived</button>
</form>

<!-- redirect user to the product edit page for a product if a re-stock has not yet arrived -->
<form action='EditProduct.php'>
<button type='submit' value=''>Stock Hasn't Arrived</button>
</form>
```

Stock Has Arrived

Stock Hasn't Arrived

I now need to output a message to the user above these buttons, containing the name of the product and a brief description

```
Has the re-stock of product:
<?php echo "<br>".$_GET['productName']."<br>"; ?>
Arrived? <br>
Quantity Due: <?php echo $_GET['arrivalQuantity']; ?>
```

Has the re-stock of product:
banana
Arrived?
Quantity Due: 103

Stock Has Arrived

Stock Hasn't Arrived

This is good, but I believe that the two buttons should be placed next to each other horizontally rather than vertically. I will add the CSS attribute display: inline-block to both form elements to fix this.

```
<style>
.displayInline {
    display: inline-block;
}
</style>

<!-- submit user to processing page if stock has arrived -->
<form action='process_stockArrived.php' class="displayInline">
<button type='submit' value=''>Stock Has Arrived</button>
</form>

<!-- redirect user to the product edit page for a product if a re-stock has not yet arrived -->
<form action='EditProduct.php' class="displayInline">
<button type='submit' value=''>Stock Hasn't Arrived</button>
</form>
```

Has the re-stock of product:
banana
Arrived?
Quantity Due: 103
Stock Has Arrived | Stock Hasn't Arrived

I now need to pass hidden inputs into both forms, allowing the respective redirect file to access the needed data.

First, I will include hidden input values for the following in the “Stock Has Arrived” button form: productID

```
<!-- submit user to processing page if stock has arrived -->
<form action='process_stockArrived.php' class="displayInline">
    <input name='productID' type="hidden" value="<?php echo $_GET['productID']; ?>">
    <button type='submit' value=''>Stock Has Arrived</button>
</form>
```

I will then include hidden input values for the following in the “Stock Hasn’t Arrived” button form:

ProductID, TableName (the company’s stock management table), ProductName, CurrentStockValue, MinimumStockValue, Ordered, SupplierName, PhoneNumber

These values will be needed in order for the edit page to load correctly with the intended values contained within the input boxes.

Whilst including this, I noticed that I did not have the data of CurrentStockValue, MinimumStockValue, Ordered, SupplierName, and PhoneNumber in the current file.

To fix this, I will pass these values into the URL that the user uses to redirect to the hasStockArrived.php page, as the required query has already been run in the pointOfSale.php file.

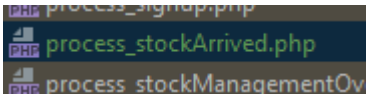
```
            /* Fetch data from each column of the result into variables */
            $arrivalQuantity = $row['ArrivalAmount'];
            $productID = $row['ProductID'];
            $productName = $row['ProductName'];
            $currentStock = $row['CurrentStockValue'];
            $minimumStock = $row['minimummStockValue'];
            $ordered = $row['Ordered'];
            $supplierName = $row['SupplierName'];
            $phoneNumber = $row['Phone'];

            /* Calculate redirect URL */
            $redirectURL = "hasStockArrived.php?productID=".$productID."&productName=".$productName.
                           "&arrivalDate=".$arrivalDate."&arrivalQuantity=".$arrivalQuantity.
                           "&currentStock=".$currentStock."&minimumStock=".$minimumStock.
                           "&ordered=".$ordered."&supplierName=".$supplierName."&phoneNumber=".$phoneNumber;

```

I will now continue to implement hidden inputs in the hasStockArrived.php file

```php
<!-- redirect user to the product edit page for a product if a re-stock has not yet arrived -->
<form action='EditProduct.php' class="displayInline">
    <input name='ProductID' type="hidden" value="<?php echo $_GET['productID']; ?>">
    <input name='TableName' type="hidden" value="<?php echo "stockmanagementtable_".$_SESSION['companyName']; ?>">
    <input name='ProductName' type="hidden" value="<?php echo $_GET['productName']; ?>">
    <input name='CurrentStockValue' type="hidden" value="<?php echo $_GET['currentStock']; ?>">
    <input name='MinimumStockValue' type="hidden" value="<?php echo $_GET['minimumStock']; ?>">
    <input name='Ordered' type="hidden" value="<?php echo $_GET['ordered']; ?>">
    <input name='SupplierName' type="hidden" value="<?php echo $_GET['supplierName']; ?>">
    <input name='PhoneNumber' type="hidden" value="<?php echo $_GET['phoneNumber']; ?>">
    <button type='submit' value=''>Stock Hasn't Arrived</button>
</form>
```

The “Stock Hasn’t Arrived” button should now be working as intended. Therefore, I can move on to processing stock arrival.

First, I will need to create a file, “process_stockArrived.php”



I will now need to start the session and establish a connection with the database

```php
session_start();

/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();
```

I will now need to create a MySQLi query updating the company's stock management database table with the updated values and setting the due date and amount to null.

This is when I realised that my designed SQL query for this would not work as previously seen in my project, adding together values inside the SQL query does not work. Therefore, I will calculate the new value of currentStockValue first before running the update query.

To do this, I will need to implement two hidden input values in the hasStockArrived.php file containing the current stock value of the product and the ArrivalAmount in order to pass them into the URL when being redirected to the process_stockArrived.php file.

```php
<!-- submit user to processing page if stock has arrived -->
<form action='process_stockArrived.php' class="displayInline">
    <input name='productID' type="hidden" value="<?php echo $_GET['productID']; ?>">
    <input name='currentStockValue' type="hidden" value="<?php echo $_GET['currentStock']; ?>">
    <input name='arrivalQuantity' type="hidden" value="<?php echo $_GET['arrivalQuantity']; ?>">
    <button type='submit' value=''>Stock Has Arrived</button>
</form>
```

I can then retrieve this value in the processing file using $_GET to then calculate the new stock value

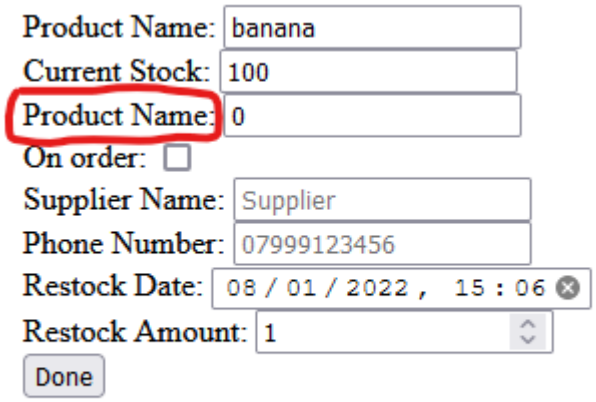
```php
/* Calculate new stock value */
$priorStockValue = $_GET['currentStockValue'];
$arrivalStockValue = $_GET['arrivalQuantity'];
$newStockValue = $priorStockValue + $arrivalStockValue;
```

Now I can create the SQL query

```php
/* Update the values of the product in the company's stock management database */
$tableName = "stockmanagementtable_".$_SESSION['companyName'];
$query = "UPDATE $tableName SET CurrentStockValue = '$newStockValue', Ordered = 'off', ArrivalDate= NULL, ArrivalAmount = NULL";
mysqli_query($connection,$query);
```

Testing can be found in **testing stock arrival acceptance.mp4**

Testing presented quite a few errors despite the stock management table updating correctly. The first error is a minor error from a prior stage. The input box that should be named “minimum stock quantity” in the EditProduct.php file is named “Product Name”:



This is fixed by updating the name of the label of this input box in the EditProduct.php file from “Product Name:” to “Minimum Stock:”

```php
    <!-- Display minimum stock value input field -->
    <label for='MinimumStockValue'>Minimum Stock:</label>
    <input type='text' name='MinimumStockValue' value='$_GET[MinimumStockValue]' required>
```

Another issue is that there is a WHERE condition missing from the MySQLi query updating the company's stock management table when stock has arrived, causing all products to be updated to the specified values, not just the intended value. To fix this, I will include a where condition in this query within the process_stockArrived.php file:

```php
/* Update the values of the product in the company's stock management database */
$productID = $_GET['productID'];
$tableName = "stockmanagementtable_".$_SESSION['companyName'];
$query = "UPDATE $tableName SET CurrentStockValue = '$newStockValue', Ordered = 'off', ArrivalDate= NULL,
          ArrivalAmount = NULL WHERE ProductID = '$productID'";
mysqli_query($connection,$query);
```

The next error is that when the date & time is compared, only the date is considered when checking whether the date and time is less than the current date and time, causing the hasStockArrived alert to sometimes trigger too early. This is due to the current time being in the incorrect format – it is in the format “YYYY-MM-DDTHH:MM” rather than “YYYY-MM-DD HH:MM”. To fix this, I will use the PHP str_replace() function to replace the letter T with a space character.

```php
$currentDateShort = str_replace( search: "T", replace: " ", subject: "$currentDateShort"); /* remove the letter T from the
                                                                                              string and replace with a space
                                                                                              character */
```

The arrival date and time is also in the incorrect format. It contains seconds when it should only contain hours and minutes. Therefore I will use the substring function to shorten the date and time to its correct length.

```php
        $arrivalDate = substr($arrivalDate, offset: 0, length: 16); // shorten arrival date to remove seconds
```

Stock levels will now be triggered at the correct timing. Therefore, they now work as was initially intended.

The final problem is that the minimum stock value is being incorrectly retrieved when editing a product, saying that the stock has not arrived. When reviewing this problem, I noticed an error in the URL passing the user to the EditProduct.php page from the hasStockArrived.php page:

r+%2F>%0D%0A<b>Warning<%2Fb>%3A++Undefined+variable+%24_SESSION+in+<b>C%3A\xampp\htdocs\POSSY\hasStockArrived.php<%2Fb>+on+li

+on+line+<b>23<%2Fb><br+%2F>%0D%0A<br+%2F>%0D%0A<b>Warning<%2Fb>%3A++Trying+to+access+array+offset+on+value+of+type+null+in+

in+<b>C%3A\xampp\htdocs\POSSY\hasStockArrived.php<%2Fb>+on+line+<b>23<%2Fb><br+%2F>%0D%0Astockmanagementtable_&ProductName=ban

I have attempted to use a session variable in the hasStockArrived.php file without starting the session. To fix this, I will start the session in the hasStockArrived.php file:

```php
<?php
session_start(); // start the current session
?>
```

Also, n incorrect array key is being used when trying to access the data within a row of the SQL query result, where an extra m was accidentally placed in the array key value within the pointOfSale.php file.

```php
$minimumStock = $row['minimummStockValue'];
```

To fix this, I will enter the correct array key:

```php
$minimumStock = $row['MinimumStockValue'];
```

The final issue is that the web page does not reload after processing in the process_stockArrived.php page has been completed. To fix this, I will refresh the header at the end of the file to redirect the user.

```php
/* Redirect the user */
header( header: 'refresh: 0; URL = pointOfSale.php');// redirect to home page
exit;
```

Implementation of the resending of alert messages

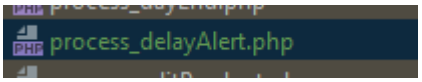
First, I will need to implement a button into the lowStock.php file, “delay alert”, within a form that re-directs the user to the file “process_delayAlert.php”.

```php
<!-- Output delay alert button to delay a low-stock alert by 20 minutes -->
<form style='display: inline'  class='lowStock' action='process_delayAlert.php'>
    <button type='submit' >Delay Alert</button>
</form>
```

Product Name: Carrot
Minimum Stock Value: 200
Current Stock Value: 102
Supplier Name:
Phone Number: 0

Ordered Amount: 1
Due Date & Time: dd / mm / yyyy , -- : --
Ordered | Ignore | Delay Alert

I now need to create the file “process_delayAlert.php”



Within this file, I first need to start the session and establish a connection with the database

```php
<?php
session_start();

/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

```

I first need to retrieve the value of the productID from a session variable

```php

/* Retrieve data from session variables */
$productID = $_SESSION['ProductID'];

```

I then need to calculate the current date and time in the format YYYY:MM:DD HH:MM

```php
/* Calculate current date & time */
date_default_timezone_set( timezoneId: "GMT");
$currentDateLong = date( format: "c");
$currentDateShort = substr($currentDateLong, offset: 0, length: 16);
$currentDateShort = str_replace( search: "T", replace: " ", subject: "$currentDateShort"); /* remove the letter T from the
                                                                                              string and replace with a space
                                                                                              character */

```

I then need to take a substring of this variable, get only the number of minutes, add 20 to this value, the specified time in the user requirements for a period of delay.

```php

/* add 20 minutes to the current number of minutes from the time and store in variable $newTimeMinutes */
$newTimeMinutes = (int) substr($currentDateShort, offset: 11, length: 2) + 20;

```

I then need to validate that this value is less than 60. If the value is greater than or equal to 60, I will need to do the same process for hours. I will implement this using an if statement. After writing this, I noticed that I would then need to do this for the number of hours, which would not be a problem. However, I would then need to implement this for the number of days. Since each month has a different number of days, I would need to have a large quantity of if statements calculating the value of the date & time depending on the month. I would also need to calculate whether or not there was a leap year, leading to a much bigger task than was previously expected. Therefore, I will revert these changes and no longer include this in my project due to the complexity of this task & time constraints.

End Of Stage Testing

Test No.	Description	Test Data	Expected Result	Actual Result
1a	When a low-stock alert is sent, the alert can be deferred for 20 minutes	Press "unordered" when a low-stock alert appears	Alert reappears after 20 minutes	**This was not implemented – explained why within the stage**
2a	Does the order status get set to unordered after stock has been set as arrived	Press "yes" in stock arrived alert	The stock management database field for the order status of the particular item is set to unordered	The stock management database field for the order status of the particular item is set to unordered See **end of stage 8 testing.mp4**
3a	A stock arrival alert is triggered when the current time is less than or equal to the due date of stock	Trigger a low stock alert by setting a product's due date & time to the current date & time + 1 minute And its due stock quantity to 10 Open the point-of-sale web page	A stock arrival alert is triggered at the date & time specified	Stock is given a due date and time in the company's stock management table in the database See **end of stage 8 testing.mp4**
3b	When a due date & time is entered, the database table is updated to the correct value	Current time + 1 min 10/01/2022 Due quantity: 10 Click ordered	The due date & time is updated correctly in the company's stock management database table	The due date & time is updated correctly in the company's stock management database table See **end of stage 8 testing.mp4**

All tests that could be carried out passed the tests and are working as intended. Therefore I am happy to begin the process of completing this stage of the project.

The majority of features intended to be implemented at this stage were implemented to a good standard. However, one intended feature was not implemented at all in this stage.
During the process of implementation of the re-sending of alert messages, it came to my attention that this would be a much larger task than was previously expected. I had not previously considered the need for adjusting the month, which posed the problem of knowing how many days were in each month and whether the current year was a leap year. This would have taken a very long time to implement and would likely include many bugs as this is not something I have had previous experience with. This feature was also not included in the success criteria, as it was only a desirable feature in the project, not an essential feature.

I made sure to consult my stakeholders as to whether they were happy for me to go ahead with the project without this feature, and they agreed that despite the feature having potential uses, it is not guaranteed that they would use it; therefore, there is no problem that it has not being included.

In general, this stage took a little longer than expected as there were many bugs found during the development process. Despite this, I am very happy with the features implemented in this stage and the standard that the features are completed to. If I have time left at the end of this project, I would like to return to this stage for a short period of time to style the input boxes and form submit buttons using CSS to give the web pages created during this stage a professional look, as was previously intended according to my initial GUI designs.

This stage has met the following success criteria:

- Stock arrival date contained within the stock management database table used to prompt operators whether the stock has arrived as outlined in requirements 3.5

One improvement I would have liked to make during the design of this stage is to think more in-depth about the necessary SQL queries needed. During this stage, I required SQL queries that were not identified during my design phase. If I had not made these mistakes, it would have made the implementation process slightly faster, as I would be able to code continuously without the need for much thought processing. This would increase efficiency in my programming, as I will not lose my train of thought.

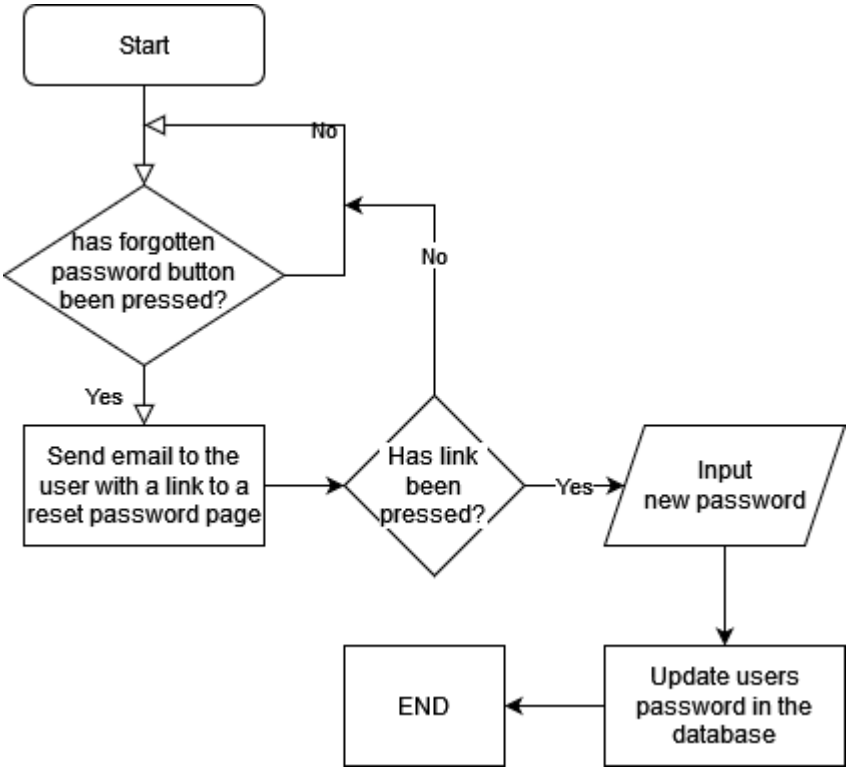
Stage 9 – User Quality of Life (12 hours)

This is the final stage of my project, which aims to introduce features to benefit the system operators and widen the target audience via allowing compatibility across multiple device types. This stage also includes the ability to reset a forgotten password via email, the introduction of account types, each with different permissions, an options menu to allow users to toggle alert types on/off & the ability for multiple transactions to be open at any one time.

Features/functionality:

1. Further additions to the drop-down menu
    a. Toggleable low-stock email alerts
    b. Toggleable low-stock phone alerts
2. Ability to reset a forgotten password
3. Alerts to prompt system operators of phrases to ask customers upon ordering a specific item
4. The ability to open multiple transactions at one time
5. Hosting the system on a server
6. Cross-device compatibility (mobile/pc)
7. Implementation of account types with different permissions
8. Creation & embedding of the tutorial video into the home page

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Can users reset their passwords via their email accounts	Press “forgotten password” and enter a registered email and a new password	The users' password is reset	
2a	Does the website load correctly sized according to your device type?	Open website on pc Open website on mobile	The website opens correctly sized and laid out on both device types	
4a	Can the system operator toggle their low-stock email alerts?	Toggle off email alerts in the settings menu and wait for a low stock alert to appear	An email alert is not sent to the system operator when a low stock alert appears	
4b	Can the system operator toggle their low-stock phone alerts?	Toggle off phone alerts in the settings menu and wait for a low stock alert to appear	A phone alert is not sent to the system operator via text	
6a	Can system operators open multiple transactions at one time?	Open two new transactions	Both transactions should be accessible	



This shows the process of resetting a forgotten password. The user will first need to press a hyperlink contained within the login page. They will then get send an email with a link to a reset password page. If this link is pressed, the user should input a new password, and this new password should update the password currently contained within the credentials table of the users database for their account. If the link is not pressed, nothing will happen and their password will remain the same.

Implementation of low stock alerts via email & phone

I will first need to test the PHP mail() function, ensuring that the parameters I am using are correct and working

```php
<?php
mail( to: "possynea@gmail.com", subject: "MySubject", message: "myMessage");
?>
```

Upon opening the web page to run this file, I met the following error:

Warning: mail(): Bad Message Return Path in **C:\xampp\htdocs\POSSY\testDELETE.php** on line **2**

I do not have a return path. This was initially seen as a simple feature to implement. However, upon further inspection, I do not have features that are required to implement this, such as a mail server. Therefore, the low stock alerts via email will no longer be included in my project, as this is far outside my area of knowledge. Low stock alerts via phone were also supposed to take place using emails, ads you are able to send a message to an email, and a phone number receive the email contents via text. As I cannot get email alerts to work, I will also not be able to get text alerts to work, and therefore this feature as a whole will no longer be implemented in my final project.

This also means what is the “drop-down” menu will nolonger act as a settings menu as previously intended. This is because there are nolonger any settings that can be displayed in that menu, only the refund transaction and daily total is stored within that menu.

Ability to Reset A Forgotten Password

This feature cannot be implemented. It is reliant on the user who has forgotten their password receiving an email re-directing them to a web page where they can reset their password. As I am unable to send emails, the user cannot receive a link to the web page for them to reset their forgotten password. Therefore, users will not be able to reset a forgotten password in my project.

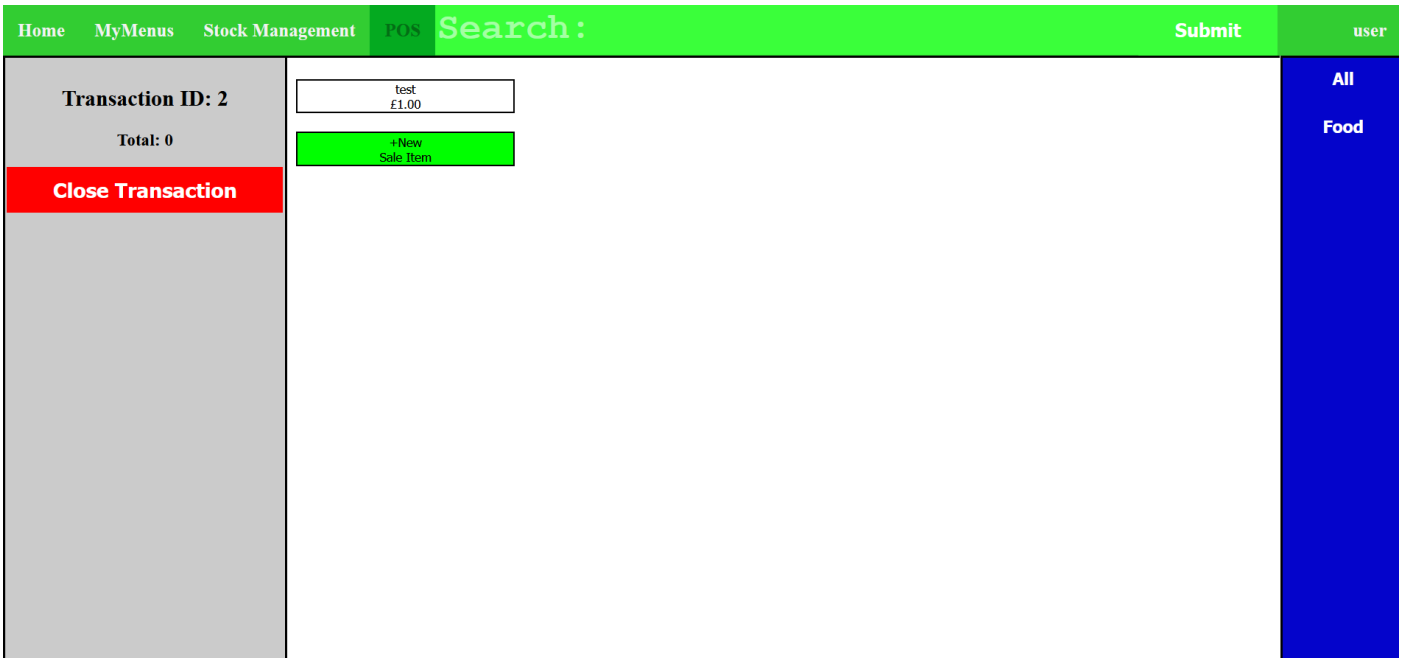
System Operator Phrase Assistance

As I chose not to implement the following feature: “Customise sale item contents” during stage 7 of development, this feature no longer makes sense to implement, as there would be very few phrases that the system operator could be prompted with, and therefore I no longer believe that this feature would be beneficial to the end project.

The Ability To Open Multiple Transactions At One Time

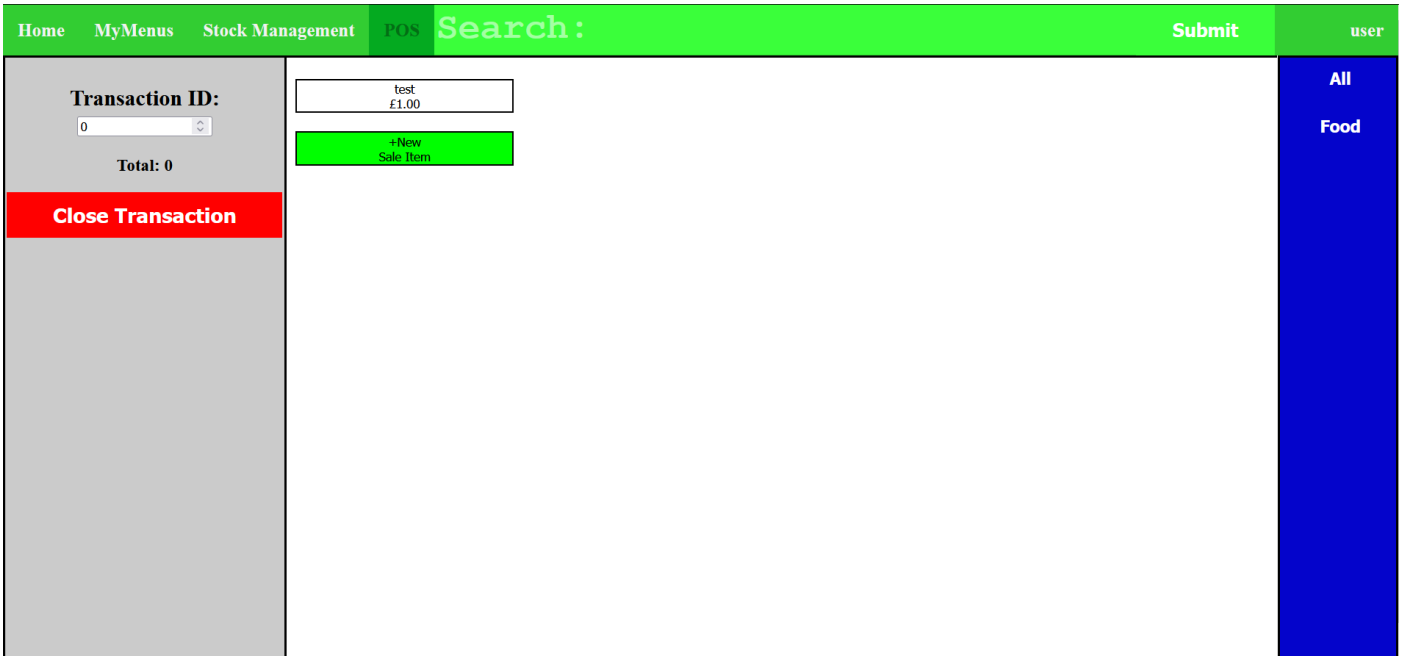
First, I will need to implement the ability to change the current transaction number. Currently, the transaction number is displayed as text and is output onto the left-GUI within a HTML header tag using the PHP echo function.

```
221            <h2 align='center'> Transaction ID: <?php echo"$transactionID"; ?></h2>
```



To have the ability to change this number, I will need to instead output this value inside a number input field.

```
220    ?>
221            <h2 align='center'> Transaction ID:
222            <form>
223                <input type='number' value='<?php echo $transactionID; ?>' step='1'>
224            </form>
225            </h2>
226
227    <?php
```

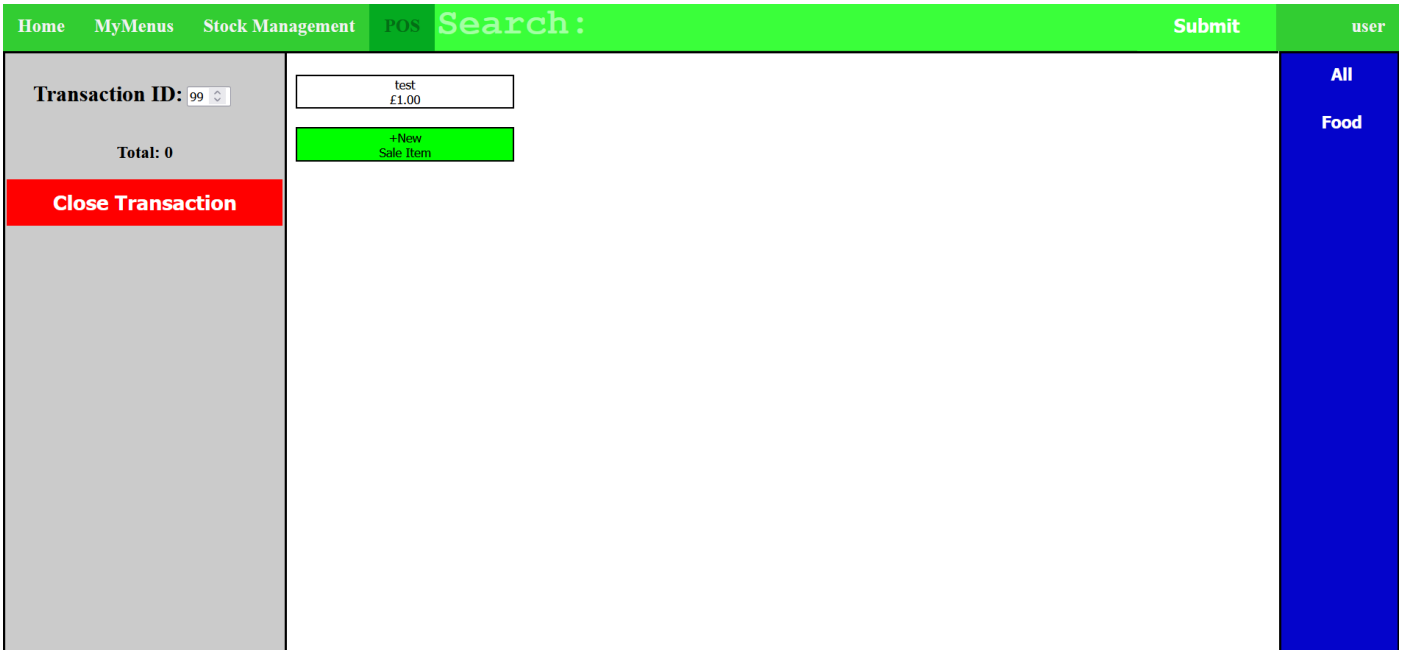


This is good, but the transaction number is not next to “Transaction ID: ”. To fix this, I will create a class that displays in an inline-block using CSS and apply this class to both HTML elements. I would also like to reduce the width of the transaction ID box to fit approximately 2, possibly three digits in it.

```
220  ?>
221
222              <style>
223                  .displayInline{
224                      display: inline-block;
225                      text-align: center;
226                  }
227
228              </style>
229
230              <h2 align='center' class="displayInline" > Transaction ID: </h2>
231              <form class="displayInline">
232                  <input class="displayInline" type='number' value='<?php echo $transactionID; ?>' step='1' size='3'>
233              </form>
234
235
236  <?php
```

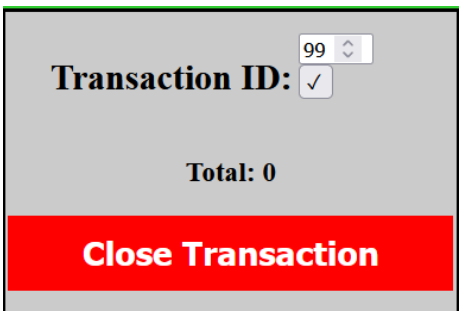
This is good, but it has now caused the text to no longer be central. To fix this, I will add padding to the styling.

```
220 ?>
221
222         <style>
223             .displayInline{
224                 display: inline-block;
225                 padding-right: 10%;
226             }
227
228             .displayInlineH2{
229                 padding-left: 10%;
230                 display: inline-block;
231             }
232         </style>
233
234         <h2 class="displayInlineH2" > Transaction ID: </h2>
235         <form class="displayInline">
236             <input class="displayInline" type='number' value='<?php echo $transactionID; ?>' step='1' size='3'>
237         </form>
238
```

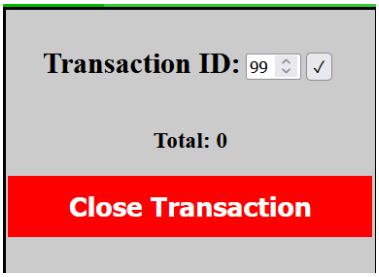


I now need to include a submit button next to the input field, allowing the form to be submitted for the transaction id to change.

```
234         <h2 class="displayInlineH2" > Transaction ID: </h2>
235         <form class="displayInline">
236             <input class="displayInline" type='number' value='<?php echo $transactionID; ?>' step='1' size='3'>
237             <button class="displayInline" type='submit'>✓</button>
238         </form>
```



This is good, and I would be able to create a working solution out of this. However, I would like the submit button to be placed on the right-hand side of the input box rather than underneath. I will remove the padding from the input field styling to fix this.



I would like for the tick-box to be green and the tick to be white and bold. However, this is not necessary for the functionality of my program. Therefore, it will only be implemented if time is left at the end of the project.

I now need to set the form to be sent by method post when submitted, as if I use the method get as I would have previously, it would interfere with the code for categories, breaking the code. I will also set the submit button's name to “transactionID”.

```
            <h2 class="displayInlineH2" > Transaction ID: </h2>
            <form class="displayInline" method="post">
                <input class="displayInline" type='number' value='<?php echo $transactionID; ?>' step='1' size='3'>
                <button class="displayInline" type='submit' name="transactionID">✓</button>
            </form>


<?php
```

I now need to use if statements to check whether there is a value that has been posted and if a value has been posted to update the value of the variable $transactionID

```
<?php
/* Check if any transactionID values have been posted */
if(!empty($_POST)){
    $transactionID = $_POST['transactionID'];

}
```

I then tested this, as seen in **multiple transactions bug 1.mp4**

The test did not go as expected. When a transaction ID is chosen and submitted, the web page is reloaded as expected, but it does not load a transactionID.

The problem is that the form input for the transactionID number was not given a name.

```
            <!-- Outpupt the transaction ID -->
            <h2 class="displayInlineH2" > Transaction ID: </h2>
            <form class="displayInline" method="post">
                <input class="displayInline" type='number' value='<?php echo $transactionID; ?>' step='1' size='3'>
                <button class="displayInline" type='submit' name="transactionID">✓</button>
            </form>
```

instead, the submit button was given a name. To fix this, I will remove the name “transactionID” from the button and instead give the input element the name “transactionID”.

```
            <!-- Outpupt the transaction ID -->
            <h2 class="displayInlineH2" > Transaction ID: </h2>
            <form class="displayInline" method="post">
                <input class="displayInline" type='number' value='<?php echo $transactionID; ?>' step='1' size='3' name="transactionID">
                <button class="displayInline" type='submit'>✓</button>
            </form>
```

I then tested this, as seen in **testing multiple transactions 2.mp4**

Although users could now browse through multiple transactions, a couple of problems were found. Users could view transactions that have already been closed. I do not view this as a huge problem, and I am sure that this could potentially be

useful in some cases. For example, it could be good to see what a closed transaction contained if all or part of the transaction needed to be refunded. However, it was not initially intended. As this problem is not project breaking and could potentially be a useful feature all be it unintended, It will be left in the project for now.

However, a problem found that will need to be fixed is that the current transactionID can be incorrect. If the user has chosen to view a transaction that is not the default transaction (the transaction with the lowest transactionID that has not been closed), the default transactionID will be shown despite the current transaction not being the default transaction. This is because the code to output the current transactionID appears before the code in the pointOfSale.php file changing the transactionID if the user chooses to view a transaction that is not the default transaction.

```
236             <!-- Outpupt the transaction ID -->
237             <h2 class="displayInlineH2" > Transaction ID: </h2>
238             <form class="displayInline" method="post">
239                 <input class="displayInline" type='number' value='<?php echo $transactionID; ?>' step='1' size='3' name="transactionID">
240                 <button class="displayInline" type='submit'>✓</button>
241             </form>
242 
243         <?php
244         /* Check if any transactionID values have been posted */
245         if(!empty($_POST)){
246             $transactionID = $_POST['transactionID'];
247 
248         }
249 
250         ?>
```

To fix this, I will swap the two pieces of code, so that the transactionID is output after the transactionID is re-calculated if necessary.

```
236             <?php
237             /* Check if any transactionID values have been posted */
238             if(!empty($_POST)){
239                 $transactionID = $_POST['transactionID'];
240 
241             }
242             ?>
243 
244             <!-- Outpupt the transaction ID -->
245             <h2 class="displayInlineH2" > Transaction ID: </h2>
246             <form class="displayInline" method="post">
247                 <input class="displayInline" type='number' value='<?php echo $transactionID; ?>' step='1' size='3' name="transactionID">
248                 <button class="displayInline" type='submit'>✓</button>
249             </form>
```

I then tested this new code structure, as seen in **testing multiple transactions 3.mp4**

This feature is now working similarly to what I initially intended. The correct transaction ID is now shown for the current transaction, and you can view transactions other than the default transaction—however, two issues arose when testing.

The first issue is that after a transaction that is not the default transaction has been edited, the page refreshes to the default transaction. To fix this, I will use a $_SESSION variable to track the current transactionID.

I will first initiate a $_SESSION variable with array key ‘transactionID’ in the process_login.php file and set it equal to an empty string when a user has been given access to their account during sign-up.

```
39      $_SESSION['transactionID'] = "";
```

Then, I will use an if statement to check the status of the $_SESSION[‘transactionID’] variable in the pointOfSale.php file. If the contents is still an empty string, the default transaction will be opened.

```php
        //Set the transactionID to the default transaction if there is not an already existing transactionID
        if($_SESSION['transactionID'] == "") {
            $query = "SELECT TransactionID FROM transactionhistory_$companyName WHERE Complete = 'no'";

            $result = mysqli_query($connection, $query);
            $row = mysqli_fetch_assoc($result);

            $transactionID = $row['TransactionID'];
        }
?>
```

The $_SESSION variable will also need to be updated if the user has manually selected a transaction to view, and then the variable $transactionID needs to be set equal to $_SESSION[‘transactionID’]

```php
            <?php
            /* Check if any transactionID values have been posted */
            if(!empty($_POST)){
                /* Update the transactionID session variable with the posted transactionID */
                $_SESSION['transactionID'] = $_POST['transactionID'];

            }
            // Set the transactionID to the required transactionID */
            $transactionID = $_SESSION['transactionID'];

            ?>
```

I then tested the results of this fix in **testing multiple transactions 4.mp4**

I checked my user requirements, ensuring that this feature had been implemented as intended when I noticed a discrepancy. My user requirements for this feature state, “If a transaction number is chosen that does not already exist, a transaction should be opened using the entered transaction ID”. This is not something I have yet implemented.

I will need to check whether or not a transaction with the given transactionID already exists. I will do this first using an SQL query, searching for the provided transactionID in the transactionhistory table for the current company. If no results are shown, a new transaction will be opened, and a new transaction table will be created with the correct transactionID

```php
        /* search for a transaction with the current transactionID */
        $transactionHistoryTable = "transactionhistory_".$companyName;
        $query = "SELECT TransactionID FROM $transactionHistoryTable WHERE TransactionID = '$transactionID'";

        /* If no transaction ID is returned, create a new transaction with the given transactionID */
        if (mysqli_query($connection,$query) == false){
            /* Open a new transaction with the provided transactionID */
            $query = "INSERT INTO transactionhistory_$companyName(TransactionID,Complete) VALUES('$transactionID','no')";
            mysqli_query($connection, $query);

            // Create a new transaction table
            $newTransactionID = $transactionID;
            $newTableName = "transaction_".$newTransactionID."_".$companyName;
            $query = "CREATE TABLE `users`.`$newTableName` ( `SaleItemID` INT(9) NOT NULL AUTO_INCREMENT ,
            `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT NOT NULL ,
             PRIMARY KEY (`SaleItemID`))";
            mysqli_query($connection,$query);
        }

        ?>
```

I tested this in **testing multiple transactions 5.mp4**

This test showed that although transactions are now created when a new transaction ID is searched for, there is no default transaction ID when loading the page, causing an error. This is caused by tbe transaction ID being updated outside of an if statement, rather than within an if statement. This has caused the correct value of transaction ID to be overriden with an empty string. This meant that the transaction ID is not set to the default transaction ID upon loading in to the point-of-sale system.

```php
<?php
/* Check if any transactionID values have been posted */
if(!empty($_POST)){
    /* Update the transactionID session variable with the posted transactionID */
    $_SESSION['transactionID'] = $_POST['transactionID'];

}
// Set the transactionID to the required transactionID */
$transactionID = $_SESSION['transactionID'];
```

To fix this, I will place lines 244 and 245 into the if statement.

```php
/* Check if any transactionID values have been posted */
if(!empty($_POST)){
    /* Update the transactionID session variable with the posted transactionID */
    $_SESSION['transactionID'] = $_POST['transactionID'];

    // Set the transactionID to the required transactionID */
    $transactionID = $_SESSION['transactionID'];

}
```

I then tested this in **testing multiple transactions 6.mp4**

This test produced an abundance of warnings:

Because I have now placed the variable $transactionID within the if statement, it has nolonger been initiated, hence producing the errors. To fix this, I will initate the variable before its first use.

I then checked the change was working in **testing multiple transactions 7.mp4**

This test presented an unusual bug that I had not noticed previously. After adding an item to the transaction, the transaction ID goes blank. If a sale item is then added to the transaction, the quantity of that item is not increased in the transaction but the transaction ID is updated again to the correct value on page refresh. This is because the transactionID is initiated to blank in the pointOfSale.php file. To fix this, I will instead calculate the current transaction ID and store it in a session variable, and only initiate the $transactionID variable once the ID of the transaction has already been determinded.

It should also not be possible to open a transaciton if the value of the transaction ID is blank. This is because the transaction must be easily identifyable by the system operator. To fix this, I will include some validation to ensure that a blank transaction ID can nolonger be entered. (lines 240 and 243 were added, and lines 241 and 242 nested within this added if statement)

```php
            /* Check if any transactionID values have been posted */
            if(!empty($_POST)){
                if($_POST['transactionID'] != "") {
                    /* Update the transactionID session variable with the posted transactionID */
                    $_SESSION['transactionID'] = $_POST['transactionID'];
                }
            }
            $transactionID = $_SESSION['transactionID'];
```

I am happy that this feature is working, however I have decided that it should not be possible to open a closed transaction, as this may be confusing for the system operator as in the programs current state, it is impossible to tell which transactions are open and which are not. To fix this, I will implement a MySQLi query that I had not initially planned to implement. This query should search for whether the transaction posted to the pointOfSale.php file is closed. If the requested transaction is closed, the default transaction should instead be displayed.

```php
        /* Check if any transactionID values have been posted */
        if(!empty($_POST)){
            if($_POST['transactionID'] != "") {
                /* Search in the transaction history table for a transaction with the requested transactionID that is open */
                $tableName = "transactionhistory_".$companyName; //calculate the name of the table to insert into the SQL query
                $requestedTransactionID = $_POST['transactionID'];
                $query = "SELECT * FROM $tableName WHERE
                          TransactionID = '$requestedTransactionID' AND Complete = 'yes'";
                $result = mysqli_query($connection,$query);

                /* Calculate the number of rows returned by the query */
                $numberOfRows = 0;
                if($result == false){
                    $numberOfRows = 0;
                }
                else{
                    $numberOfRows = mysqli_num_rows($result);
                }

                /* If there were results found, update the current transactionID */
                if($numberOfRows == 0) {
                    /* Update the transactionID session variable with the posted transactionID */
                    $_SESSION['transactionID'] = $_POST['transactionID'];
                }
            }
        }
        $transactionID = $_SESSION['transactionID'];
```

Feedback from the stakeholder

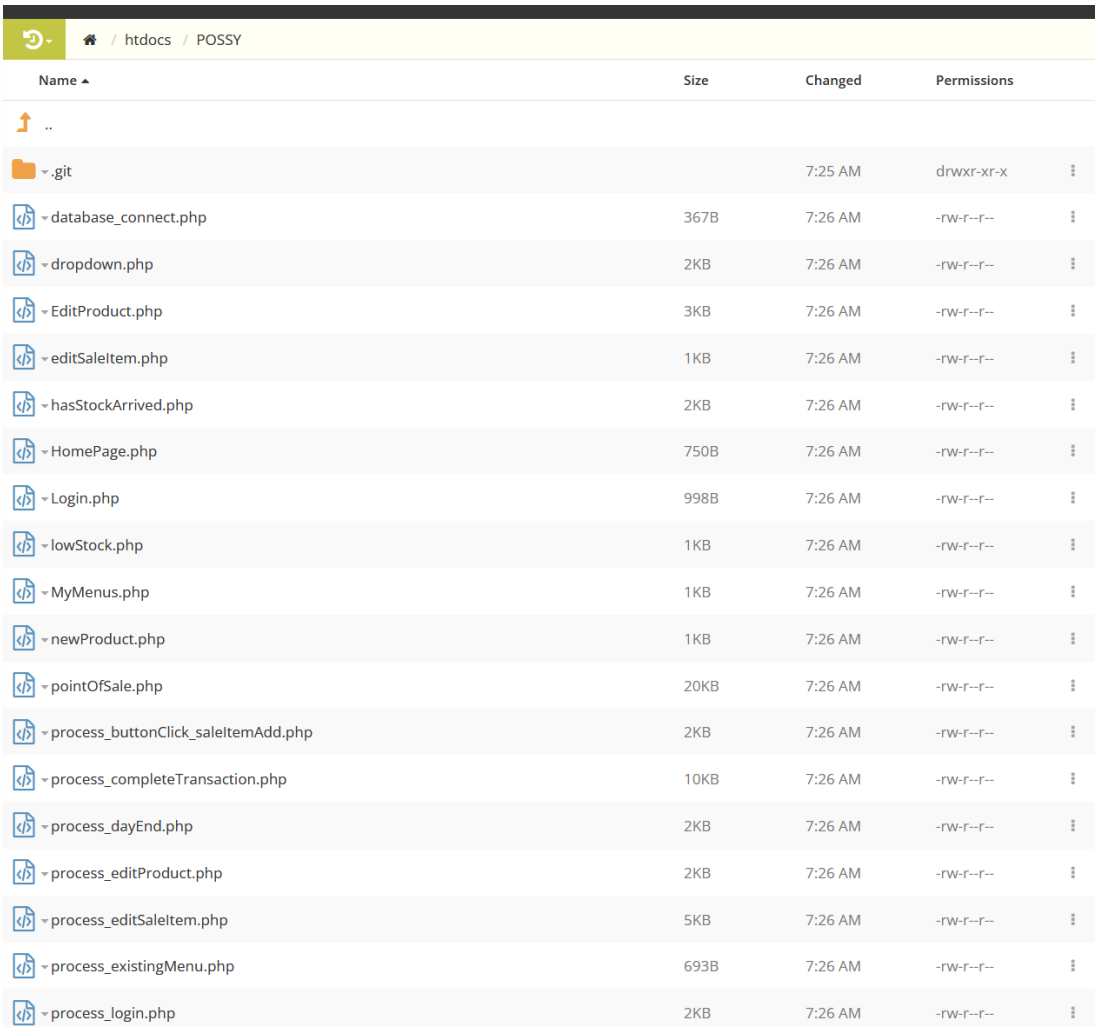
After implementing this feature, I felt that my project had really come together and that my stakeholders would be more than happy with the current program. Therefore, I decided to contact them for some feedback, below are the main outcomes of the discussion:

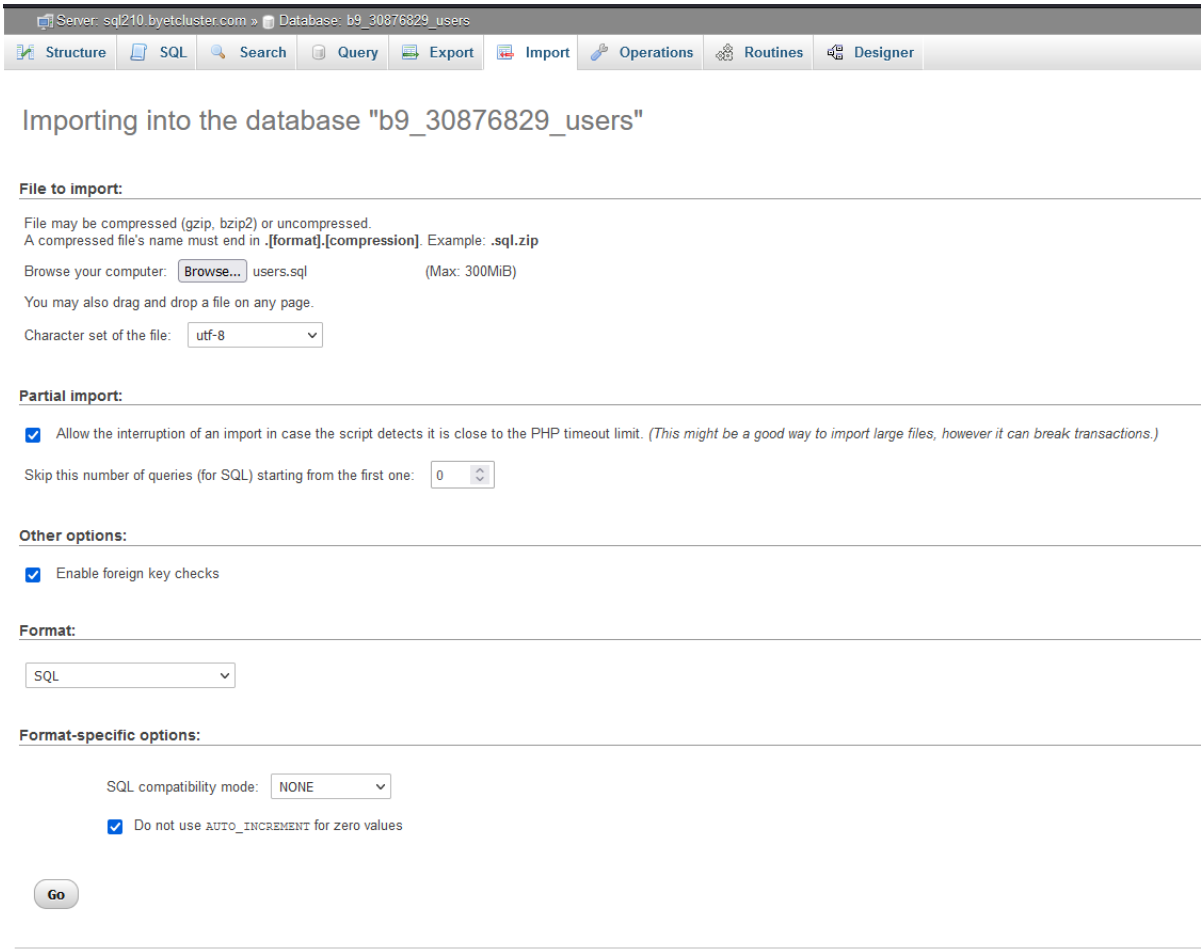
- The point-of-sale system as a whole is working perfect and just as expected
- However, it would be nice if the sale item buttons could be aligned in a grid as previously intended, as it will increase both the efficiency and look of the program, and will more closely resemble the initial design.
- It is great that the total at the end of the day can be displayed
- However, it would be good if once implemented, the total was displayed alongisde the total for cash and the total for card (this was not initially in the user requirements)
- It would be nice to see some styling added to many of the buttons and input boxes to give the program a professional look, however they are more than happy with the design currently
- Some input boxes could contain slightly more detail explaining what should be placed in them, a good example of this given was the input box for entering the products that a sale item contains, as it was not quite clear that if you wanted multiple of the same product included, you must write the name of the product twice with each product seperated by a comma, rather than a number next to the name of the product

- The stock management system works excellently, and will be great for the ordering and control of their stock levels
- The layout of the toolbar throughout the page is clear and presents an easy way to navigate throughout all pages on the website.

Creating a server host

I first need to create an account on byethost9.com to get access to a free server. To set up the server with my project, I will need to upload the project files to the server (url = http://possy.byethost9.com/POSSY/HomePage.php) and then I will need to create a datbase on the server ()import my databases into phpmyadmin





I then need to update the database connection credentials within the files on the server to enable the database to work.

```php
<?php
/* CONNECT TO LOCAL HOST */
function openConnection()
{
    $servername = "sql210.byethost9.com";
    $username = "b9_30876829";
    $password = "b9_30876829";
    $database = "b9_30876829_users";

    $connection = new mysqli($servername, $username, $password, $database);

    return $connection;
}

function closeConnection($connection){
    $connection -> close();
}
?>
```

The name of the database also needs to be updated throughout the file where SQL CREATE TABLE queries are used. IF these are not updated, no new tables will be created when necissary causing a lot of the program not to function.

Before:

```
/* Create A Stock Management Table For This Company */
mysqli_query($connection,
    "CREATE TABLE `users`.`$tableName` ( `ProductID` INT NOT NULL AUTO_INCREMENT ,
            `ProductName` TEXT NOT NULL , `MinimumStockValue` INT NOT NULL , `CurrentStockValue` INT NOT NULL,
            `Ordered` TEXT NOT NULL DEFAULT 'off' ,`SupplierName` TEXT NULL DEFAULT NULL , `Phone` INT(11) NULL,
            `ArrivalDate` DATETIME DEFAULT NULL, `ArrivalAmount` INT DEFAULT NULL, PRIMARY KEY (`ProductID`)); ");

/* Insert the Table Name and Company Name into the Stock Management Hub table */
$query = "INSERT INTO stockmanagementhub(TableName,CompanyName)
        VALUES('$tableName', '$companyName')";
mysqli_query($connection,$query);

/* Create A Transaction History Table For This Company */
$query = "CREATE TABLE `users`.`transactionhistory_$companyName` (
        `TransactionID` INT(9) NOT NULL AUTO_INCREMENT , `Complete` TEXT NOT NULL DEFAULT 'no' ,
        `Total` DECIMAL(10,2) NOT NULL, PRIMARY KEY (`TransactionID`))";
mysqli_query($connection,$query);

/* Create Initial Transaction Table */
$query = "CREATE TABLE `users`.`transaction_1_$companyName` ( `SaleItemID` INT(9) NOT NULL AUTO_INCREMENT ,
        `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT NOT NULL ,
        PRIMARY KEY (`SaleItemID`))";
mysqli_query($connection,$query);
```

After:

```
95      if($numResults == 1){
96          /* Calculate Table Name For This Company */
97          $tableName = "stockmanagementTable_".$companyName;
98
99          /* Create A Stock Management Table For This Company */
100         mysqli_query($connection,
101             "CREATE TABLE `b9_30876829_users`.`$tableName` ( `ProductID` INT NOT NULL AUTO_INCREMENT ,
102                     `ProductName` TEXT NOT NULL , `MinimumStockValue` INT NOT NULL , `CurrentStockValue` INT NOT NULL,
103                     `Ordered` TEXT NOT NULL DEFAULT 'off' ,`SupplierName` TEXT NULL DEFAULT NULL , `Phone` INT(11) NULL,
104                     `ArrivalDate` DATETIME DEFAULT NULL, `ArrivalAmount` INT DEFAULT NULL, PRIMARY KEY (`ProductID`)); ");
105
106         /* Insert the Table Name and Company Name into the Stock Management Hub table */
107         $query = "INSERT INTO stockmanagementhub(TableName,CompanyName)
108                 VALUES('$tableName', '$companyName')";
109         mysqli_query($connection,$query);
110
111         /* Create A Transaction History Table For This Company */
112         $query = "CREATE TABLE `b9_30876829_users`.`transactionhistory_$companyName` (
113                 `TransactionID` INT(9) NOT NULL AUTO_INCREMENT , `Complete` TEXT NOT NULL DEFAULT 'no' ,
114                 `Total` DECIMAL(10,2) NOT NULL, PRIMARY KEY (`TransactionID`))";
115         mysqli_query($connection,$query);
116
117         /* Create Initial Transaction Table */
118         $query = "CREATE TABLE `b9_30876829_users`.`transaction_1_$companyName` ( `SaleItemID` INT(9) NOT NULL AUTO_INCREMENT ,
119                 `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT NOT NULL ,
120                 PRIMARY KEY (`SaleItemID`))";
121         mysqli_query($connection,$query);
122
123         /* Insert new transaction table name into transaction history table */
124         $query = "INSERT INTO transactionhistory_$companyName(Complete)
125                 VALUES('no')";
126         mysqli_query($connection,$query);
127
128     }
```

The server host is now run on the url http://possy.byethost9.com/POSSY/HomePage.php

(please note that all testing has occurred using a localhost, ran using XAMPP. This means that bugs may appear on the server host that do not appear on the localhost. Therefore, it is not reccommended to be used at this moment in time as it has not been tested. It does however allow for users to access the POSSY website from multiple devices remotely at their own risk. It also uses a separate database to that of a local host, therefore data will not be saved between the server host and the local host. I can also not guaruntee uptime of this website, as it is hosted using a free hosting service)

Cross-device compatibility

Due to the inclusion of a server host, it is now possible to use POSSY across almost any device via a search browser. However, the whole program has been set to be run on a computer with resolution 1920px x 1080px therefore the user interface may not appear correctly on other devices. Although it is possible to resolve this problems, it would take many hours to implement and would not be of much benefit to my project. Therefore, although cross-device compatiblity has been implemented, I will not optimise for devices such as mobile phones as was initially mentioned.

Implementation of account types with different permissions

This feature will still be implemented, however not quite as outlined in the user requirements. The user requirements state that the account type should be chosen during sign-up, however I do not believe this should be the case, as users would be able to select their account type to be admin during sign-up when they should not be able to do so, therefore instead the default account type for users that are not the original registered user of a company will be set to unknown. The user will need to be confirmed to be a part of the company by a user with owner permissions via the drop-down menu to set their account type to "standard". I also believe that the account type previously described as "admin" should instead be called "owner".

To keep track of the permissions of each user, I will need to implement a column into the creedentials table of the database "accountType" to keep track of the type of each account during sign-up. This was not initially mentioned in the design of my project as it was overlooked. An account created with a unique company name will be given the account type "owner" otherwise, users will be given the account type "unknown". To implement this, I will need to alter the MySQLi queries during sign-up that insert a users cridentials into the credentials table of the database.

```
ALTER TABLE `credentials` ADD `accountType` TEXT NOT NULL
AFTER `Phone`;
```

```
/* submit details to Credentials table if username is not in use and email is valid*/
if ($numResults == 0 && validateEmail($email) == true) {
    echo "Username is valid\n";// temp

    if ($phoneNumber == null and $email != null) {
        mysqli_query($connection, query: "INSERT INTO Credentials (Username, Password, Company, Email, accountType)
            VALUES ('$username', '$password', '$companyName', '$email','unknown')");
    } else if ($phoneNumber != null and $email == null) {
        mysqli_query($connection, query: "INSERT INTO Credentials (Username, Password, Company, Phone, accountType)
            VALUES ('$username', '$password', '$companyName', '$phoneNumber','unknown')");
    } else if ($phoneNumber == null and $email == null) {
        mysqli_query($connection, query: "INSERT INTO Credentials (Username, Password, Company, accountType)
            VALUES ('$username', '$password', '$companyName','unknown')");
    } else {
        mysqli_query($connection, query: "INSERT INTO Credentials (Username, Password, Company, Email, Phone, accountType)
            VALUES ('$username', '$password', '$companyName', '$email', '$phoneNumber','unknown')");
    }

```

I will then need to run a MySQLi query that was not previously overlooked, which will update the account type to owner if it is the first time a company name has been registered to the program.

```php
/* Check if stock management table already exists for that company, if not create one. */
$result = mysqli_query($connection, query: "SELECT Company FROM Credentials WHERE Company = '$companyName'");
$numResults = mysqli_num_rows($result);

/* This is the first time the company has been registered to the system */
if($numResults == 1){
    /* Calculate Table Name For This Company */
    $tableName = "stockmanagementTable_".$companyName;

    /* Create A Stock Management Table For This Company */
    mysqli_query($connection,
        query: "CREATE TABLE `users`.`$tableName` ( `ProductID` INT NOT NULL AUTO_INCREMENT ,
            `ProductName` TEXT NOT NULL , `MinimumStockValue` INT NOT NULL , `CurrentStockValue` INT NOT NULL,
            `Ordered` TEXT NOT NULL DEFAULT 'off' ,`SupplierName` TEXT NULL DEFAULT NULL , `Phone` INT(11) NULL,
            `ArrivalDate` DATETIME DEFAULT NULL, `ArrivalAmount` INT DEFAULT NULL, PRIMARY KEY (`ProductID`)); ");

    /* Insert the Table Name and Company Name into the Stock Management Hub table */
    $query = "INSERT INTO stockmanagementhub(TableName,CompanyName)
            VALUES('$tableName', '$companyName')";
    mysqli_query($connection,$query);

    /* Create A Transaction History Table For This Company */
    $query = "CREATE TABLE `users`.`transactionhistory_$companyName` (
            `TransactionID` INT(9) NOT NULL AUTO_INCREMENT , `Complete` TEXT NOT NULL DEFAULT 'no' ,
            `Total` DECIMAL(10,2) NOT NULL, PRIMARY KEY (`TransactionID`))";
    mysqli_query($connection,$query);

    /* Create Initial Transaction Table */
    $query = "CREATE TABLE `users`.`transaction_1_$companyName` ( `SaleItemID` INT(9) NOT NULL AUTO_INCREMENT ,
            `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT NOT NULL ,
            PRIMARY KEY (`SaleItemID`))";
    mysqli_query($connection,$query);

    /* Insert new transaction table name into transaction history table */
    $query = "INSERT INTO transactionhistory_$companyName(Complete)
            VALUES('no')";
    mysqli_query($connection,$query);

    /* Update the account type of the current user to owner, as they are the user who registered their company to
       the program */
    $query = "UPDATE Credentials SET accountType = 'owner'";
    mysqli_query($connection,$query);
```

I now need to create a session variable for the current users account type $_SESSION[‘accountType’] that is stored after log-in of a user. I will retrieve this value using the MySQLi query “SELECT accountType FROM Credentials WHERE Username = (current user’s username)” then selecting the account type from the returned row.

```php
/* Store the current user's account type in a session variable */
$query = "SELECT accountType FROM Credentials WHERE Username = '$username'";
$result = mysqli_query($connection,$query);
$_SESSION['accountType'] = $result->fetch_row()[0];
```

I then need to implement access to certain web pages according to account type. The owner account type should have access to all web pages, standard account types should have access to only the point-of-sale system and stock management pages, and unknown account types should only have access to the home page, login page, and sign-up page. To avoid having to write the permissions code multiple times, I will create a file with a function in that takes account type and file name in as parameters, and refreshes the user back to the home page if they are not allowed on a page they have tried to access. The user will be displayed an alert saying that their account does not permit them to be on the page.

```php
<?php

function isUserAllowedAccessToThisPage($accountType,$fileName){

    /* Allows only owner account types can access the dropdown.php file */
    if($fileName == "dropdown.php" and ($accountType == "unknown" or $accountType == "standard")){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header( header: 'Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }

    /* Allows only owner account types can access the EditProduct.php file */
    if($fileName == "EditProduct.php" and ($accountType == "unknown" or $accountType == "standard")){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header( header: 'Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }

    /* Allows only owner account types can access the editSaleItem.php file */
    if($fileName == "editSaleItem.php" and ($accountType == "unknown" or $accountType == "standard")){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header( header: 'Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }

    /* Allows only owner account types can access the refund.php file */
    if($fileName == "refund.php" and ($accountType == "unknown" or $accountType == "standard")){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header( header: 'Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }

    /* Allows owner and standard account types to access the MyMenus.php file */
    if($fileName == "MyMenus.php" and $accountType == "unknown"){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header( header: 'Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }

    /* Allows owner and standard account types to access the stockManagement.php file */
    if($fileName == "stockManagement.php" and $accountType == "unknown"){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header( header: 'Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }



}
```

```php
require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"dropdown.php");
```

```php
require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"EditProduct.php");
```

```php
require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"editSaleItem.php");
?>
```

```php
<?php
require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"refund.php");
?>
```

```php
<?php
session_start();

require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"MyMenus.php");
?>
```

```php
require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"stockManagement.php");
?>
```

Now I need to implement the confirmation of newly signed up accounts within the drop-down menu for users with owner permissions. This was not initially planned within my designs, as it was overlooked during the designed however mentioned as a feature in the user requirements. The username of each user with company name equal to the current user's company and account type unkown will be displayed in this drop-down menu within a submit button. Pressing the submit button will send the current user to a processing page where the selected user's account type will be updated from unknown to standard. One problem with this is that the account type of a user can never be updated to or changed from owner. However, this was not mentioned in the user requirements therefore will not be implemented.

```php
/* Output users requested to your company */
$query = "SELECT * FROM Credentials WHERE Company = '$companyName' AND accountType = 'unknown'";
$result = mysqli_query($connection,$query);

/* Calculate the number of rows returned by the query */
$numRowsReturned = 0;
if($result == false){
    $numRowsReturned = 0;
}
else{
    $numRowsReturned = mysqli_num_rows($result);
}

/* Loop through each row of the query result outputting the username and a button to confirm that their account type
   can be updated from unkown to standard, allowing them access to the company's systems with standard permissions. */
for($i = 0; $i<$numRowsReturned; $i++) {
    $resultRow = mysqli_fetch_assoc($result);

    ?>
    <!-- Output a button that when pressed updates the intended user's account type from unkown to standard -->
    <!-- echo $resultRow['Username'] outputs the username of a user with account type unknown and company = the current user's company -->
    <form action="process_updateAccountType.php">
        <button type="submit">Allow User <?php echo $resultRow['Username'] ?> Access</button>
        <!-- Create a hidden input field to post the username of a user to the processing file -->
        <input type="hidden" value="<?php echo $resultRow['Username'] ?>" name="usernameToUpdate">
    </form>

    <?php
}
```

I now need to create a processing page to update the selected user's permissions in the database table. As usual with a processing file, I will need to start the session allowing session variables to used, and connect to the database to be able to update the value of a user's accountType. I will then need to retrieve the username posted to the processing file, and run the SQL query: UPDATE Credentials SET accountType = 'standard' WHERE Username ='(username posted to processing file)'.

```php
<?php
session_start(); //start the current session allowing session variables to be used

/* Open DB connection */
include "database_connect.php";
$connection = openConnection();

/* Retrieve the required user to update their account type using $_GET */
$usernameToUpdateAccountType = $_GET['usernameToUpdate'];

/* Update the retrieved user's account permissions to standard */
$query = "UPDATE Credentials SET accountType = 'standard' WHERE Username = '$usernameToUpdateAccountType'";
mysqli_query($connection, $query);

```

Refund
End Of Day
Daily Total: 41
Allow User username Access

I like this, but I would like this section to be given a title and I would also like more description in the submit button, as right now it may not be clear to new users what the button is used for.

Refund
End Of Day
Daily Total: 41

The Following Users Request Standard Permissions To Your Company:

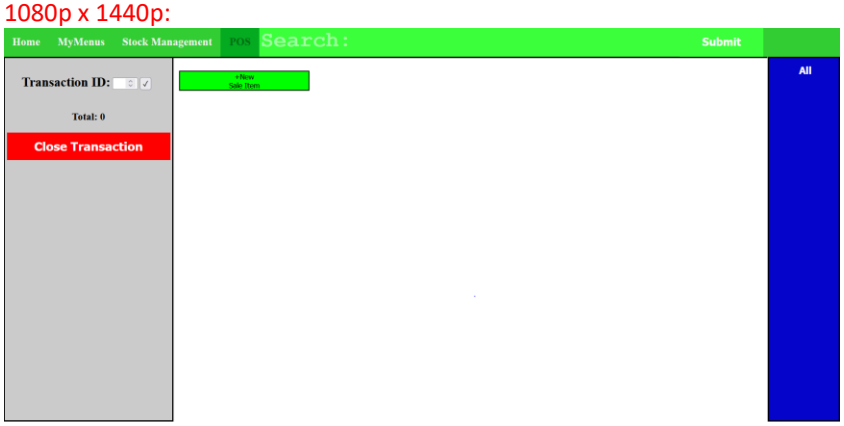
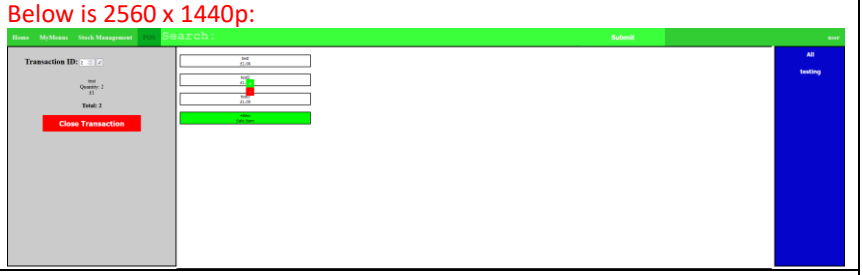
Allow User "username" Standard Access To Your Company's System

This is better, it is clear what the current user is doing if the button is pressed and the web page is more presentable as it is now clear that the implemented section is separate from the buttons above (Refund, End Of Day).

Tutorial Video

The tutorial video can be found in tutorialVideo.mp4

End Of Stage Testing

Test No.	Description	Test Data	Expected Result	Actual Result
1a	Can users reset their passwords via their email accounts	Press “forgotten password” and enter a registered email and a new password	The users' password is reset	**Unable to test as this was not implemented**
2a	Does the website load correctly sized according to your device type?	Open website on pc Open website on mobile	The website opens correctly sized and laid out on both device types	The website loads, however loads incorrectly sized when the device is not 1080p x 1440p:  Below is 2560 x 1440p: 
4a	Can the system operator toggle their low-stock email alerts?	Toggle off email alerts in the settings menu and wait for a low stock alert to appear	An email alert is not sent to the system operator when a low stock alert appears	**Unable to test as this was not implemented**
4b	Can the system operator toggle their low-stock phone alerts?	Toggle off phone alerts in the settings menu and wait for a low stock alert to appear	A phone alert is not sent to the system operator via text	**Unable to test as this was not implemented**
6a	Can system operators open multiple transactions at one time?	Open two new transactions	Both transactions should be accessible	System operators are able to open 2 transactions at one time and can access them both This can be seen in **testing multiple transactions final.mp4**

I am not very happy with this stage – many of the features intended to be implemented could not be implemented. This includes the ability to send alerts via SMS & email and due to this also the ability to reset a forgotten password. The ability for system operators to shown phrases to ask was also unable to be included due to a prior feature not being included. Despite this, some important features were added during this stage.

This stage implemented the use of multiple account types – I really like this feature, as it gives company owners the ability to control who can access their menus, and which aspects they can access. If I was to re-make this feature however, I would like to add the ability to add co-owner accounts, providing more access to some users.

This stage also implemented a server host, therefore allowing the ability for the program to be compatible with multiple device type. This was a really big think, as part of the aim of this program was to provide users easy access to a point of sale and stock management system from anywhere, and the compatibility of multiple device types really contributes to this. However, I would like to have optmised more devices such as mobile phones for this program, as it has come to my attention that some devices may not show the web pages correctly. This is something I would like to improve if I were to re-create this feature.

I was happy with the ability to open multiple transactions implemented in this stage, it allows system operators to open a transaction with a specific ID, and access multiple transactions without closing them. This feature will really open my target audience as this project is now suitable for many establishments that do not require customers to pay when they order.

Success criteria met during this stage:

- The project is hosted by a server as outlined in requirements 4.3
- Ability to create a new transaction as outlined in requirements 2.5

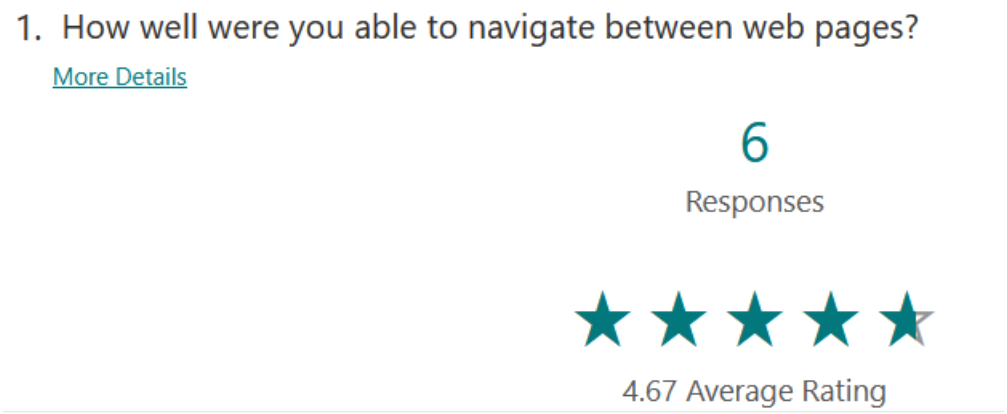
Post Development Plan

Beta testing

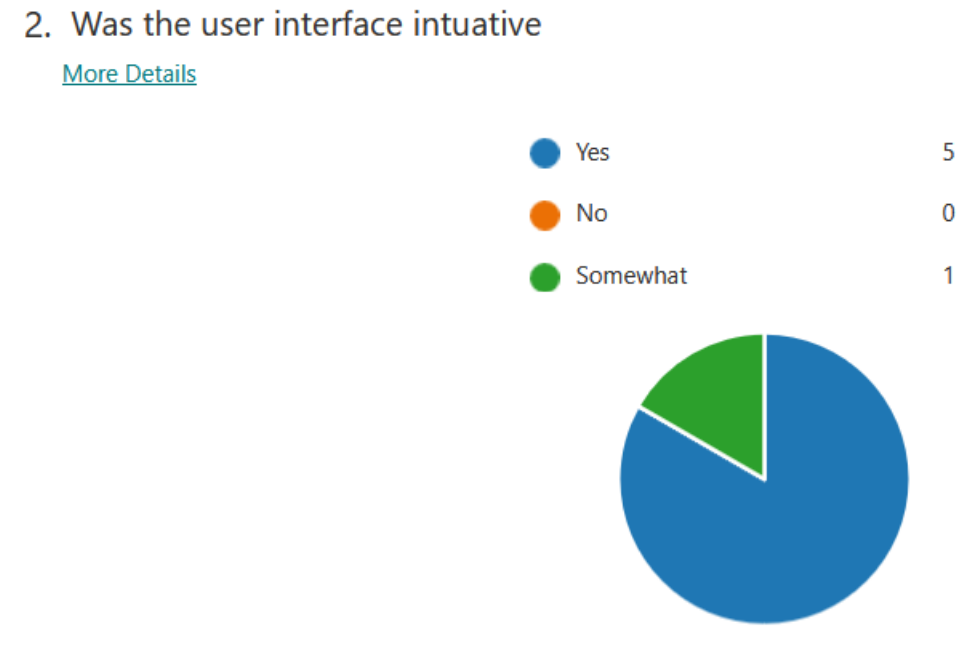
I will provide testers a survey to complete. The survey will focus on useability as well as the quality of the solution – does it work? The survey will contain the results of each acceptance test and a place to write some feedback if things did not go according to the expected result.
The following questions will be asked:

- How well were you able to navigate between web pages? (1-5)
- Do you think that the user interface is intuative? (yes/no/somewhat)
- How readable was any text? (1-5)
- Were buttons ideally coloured and sized? If not what would your preference be?
- Did you encounter any bugs whilst testing the system?
- Were there any features of the system you particularly liked?
- What features do you believe could be added to improve the system?
- How well do you believe the success criteria have been met (there will be an image of the success criteria attatched)

I provided this survey to my stakeholders, achieving 6 responses. A summary of the responses is shown below.



Almost all beta testers found the website easy to navigate – this is something that I am very proud of, as it means that users will be able to get the most out of my project.



I would have liked for all users to have found the user interface intuative, however the vast majority did thefore I am still happy with the result, although would like to request more details from the stakeholder that answered somewhat to further imprve my project in the future.

3. How readable was any text?

More Details

6

Responses

4.17 Average Rating

To me this result was disappointing. Although an average of 4.17 is good, readability is very important to the efficiency of this program, therefore if I were to re-make this project, I would focus on improving this making my project suitable for more users.

5. Did you encounter any bugs whilst testing the system?

4 Responses

ID ↑	Name	Responses
1	anonymous	The sale item buttons are displayed only vertically, leading to a lot of lost space in the system

This question received four responses, 3 of which were relating to a known bug where the sale item buttons would only align vertically. I am really disappointed that I could not fix this bug before the end of the project, as I also believe it is rather annoying. However, the 3 responses mentioning this only contained a similar response, therefore I am quite happy that relatively few bugs were found.

3	anonymous	The point of sale page does not load correctly. The columns are not quite large enough for the contents.

One user (on a mobile device) had problems loading the point of sale page correctly. Although this is a very big problem, the program was never optimised for mobile use therefore I am not overly bothered about this being an issue, as the project is primarily designed for PC users.

6. Were there any features of the system you particularly liked?

5 Responses

This question received a multitude of different responses, some of which felt very rewarding after the very long task of programming this project. Several users really liked the accessibility of the system over the internet, meaning that if they were travelling they would not have to bring any, or very little equipment with them to use the system and would not need to download any applications.

7. What features do you believe could be added to improve the system?

More Details

4

Responses

Latest Responses

"ability to edit the contents of a sale item"

"A little more description on the new sale item web page to explain the purpose of each input field."

I was quite disappointed that some of the buttons lacked description, but since reviewing this I also agree that some buttons in the system could have been described better. I did not initially see this as I was the one to design the entire system hence knew how the program worked inside and out.

8. How well do you believe the success criteria have been met?

More Details

6

Responses

4.33 Average Rating

I am very proud of this result – considering there were a few features that I could not implement I am delighted that a large number of users rated this a 5/5.

Post Development Robustness Tests

In these tests I will test the project with inputs designed to cause errors.

Test No.	Description	Type Of Test	Test Data	Expected Result	Actual Result
1.1	Fill the credentials table of the database with an excess number of users, attempting to crash the database.		Add 100 new accounts to the database using: Name: username(1-100) Password: password Company: myCompany Email: company@email.com phone no: 1234567891	The database does not crash	SUCCESSFUL The database does not crash
2.1	Check that values cannot be included into input boxes that should not be allowed, for example an email address with two @ symbols, and a phone number of length 4 digits.		Username: newUser Password: password Company name: newCompany Email: Firstname@ Lastname@ gmail.com Phone no: 1234	The input is invalid	Semi-successful An email with two @ symbols cannot be entered, passing the test However, a phone number with only 4 digits can be entered, failing the test See **Post dev test 2.1.mp4**
3.1	Use SQL injection to log in to an account that is not yours.		Username: " or 1='1' Password: " or 1='1'	The log-in details are invalid	This test failed. Despite producing an error, an account was accessed therefore the project is susceptible to SQL injections See **SQL inject.mp4**
4.1	Can the program be run on other operating systems?		Open and use the POSSY system on other operating systems (windows & linux)	The system should run with no problems on any operating system	The project opens the same on a linux device of the same resolution SUCCESSFUL

5.1	Can the program be run on different machines?		Open and use the POSSY system on different machines of the same operating system (Windows 10)	The system should run the same on each machine	Semi-successful The project works on all machines, however on devices with different resolutions the project will be displayed differently. See images 1.1 and 1.2 below
6.1	Can the program be run on different browsers?		Open and use the POSSY system in edge & firefox	The system should run the same on all browsers	Successful, the system runs the same on all tested browsers See images 2.1 and 2.2

Image 1.1 - 2560 x 1440

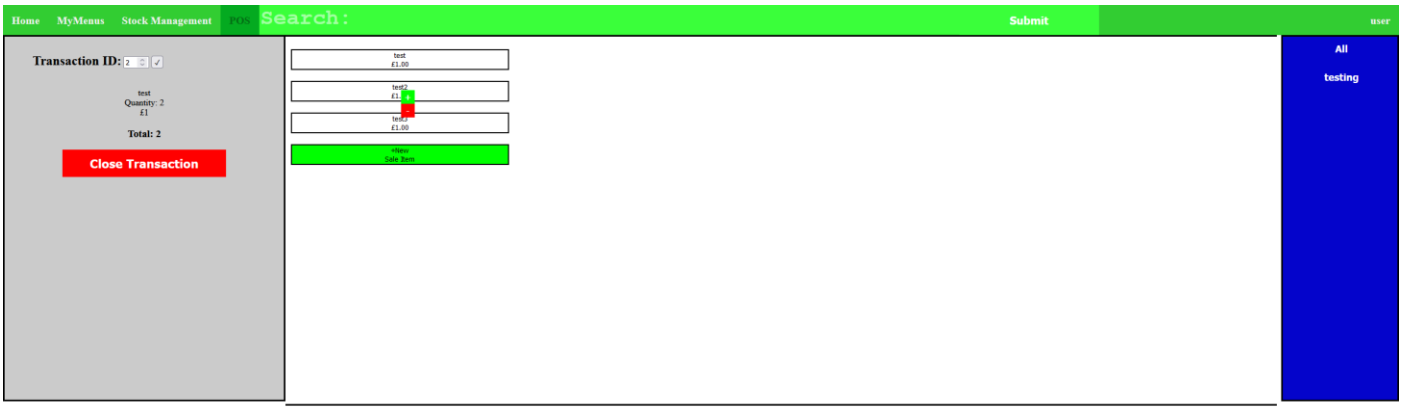


Image 1.2 – standard resolution 1080 x 1440

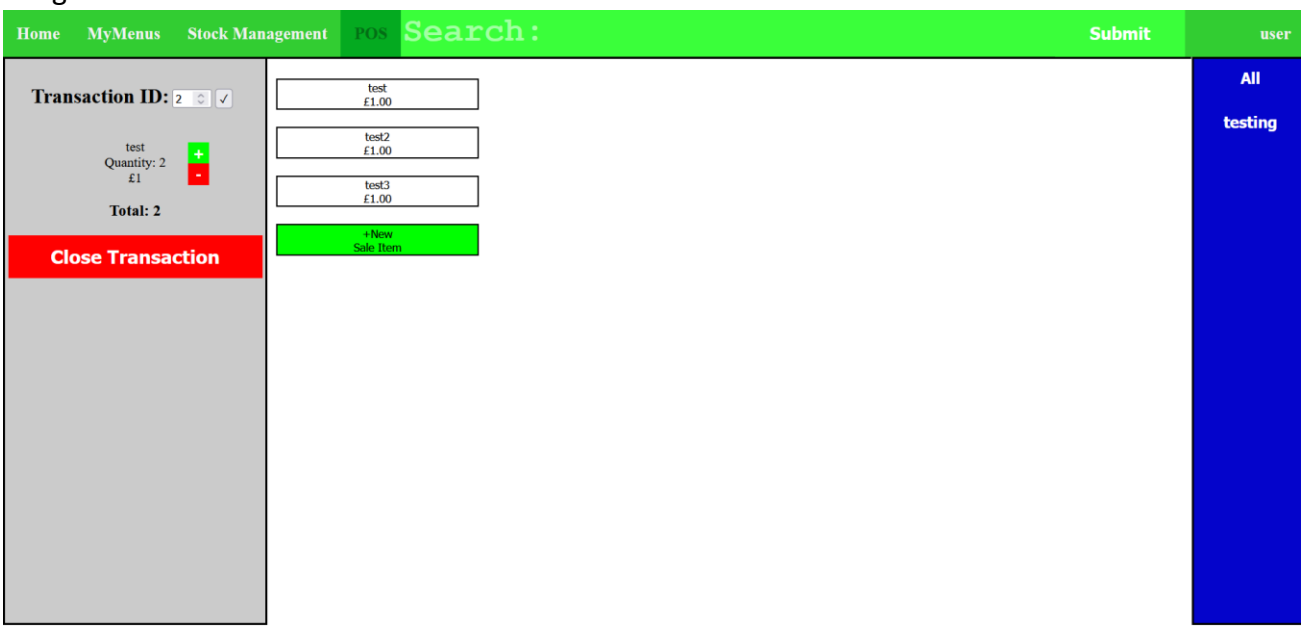


Image 2.1 – Microsoft Edge

Image 2.2 – Firefox

EVALUATION

Success Criteria Bullet point 1

- *A main menu as described in requirements 1.1*
  - Links to log-in, sign-up and 'My Menu' pages

PARTIALLY SUCCESSFUL

The homepage is the page that the user first loads to and provides access to all necessary web pages via a toolbar. This is the main aspect of the home page and is entirely successful. I am 100% happy with the design of the home page and it does not differ from my initial designs at all. (this can be seen in the end of stage testing within stage 1 of my project)

However, after creating the tutorial video, I forgot to embed the video within the home page, therefore the homepage does not currently serve the purpose of educating users on the system as initially intended. This is unfortunate as this video was shown to users during beta testing, and it was proven useful. I was therefore unsuccessful in completing this aspect of the project; hence this bullet point was not entirely met.

Success Criteria Bullet point 2

- A 'credentials' database table containing the account information of users after they have completed sign-up. (as outlined in requirements 1.5) It should contain:
  - Username (required)
  - Password (required)
  - Company Email (Required)
  - Phone number

SUCCESSFUL

The credentials database table stores the details of all signed-up users to the project including their username, password, company email, and phone number. This allows the user to login to their account and their details be saved for the next time that they use the project. It also allows for some of the projects features to be personalised, for example the drop-down menu where the current user's username is displayed. I am very happy as this works entirely as intended, I would make no changes to this database table if this project were re-created. The credentials database table is shown as working in the end of stage 2 testing test 2a.

Success Criteria Bullet point 3

- A sign-up page as described in requirements 1.2.0
  - User's entered details are stored in the 'Users' database

SUCCESSFUL

Users are able to sign up to the system using a username, email, password, company name and password. Users cannot create an account with a username that already exists. When the user has signed up, their details are stored in the credentials table of the database. This is a very key feature of the project, as it lets the user to login to their account, allowing them to access key aspects of the system. This feature is showcased as working in the end of stage 2 testing tests 1b, 2a, 3a, 3b, 3c

Despite this, I am not entirely happy with this page. If I were to program this again, I would like the web-page to look like the web-page I designed in my design phase, where the buttons have been styled, and the input fields are central to the page. This would make the project look more professional and more user friendly, potentially widening the audience for this system.

Success Criteria Bullet point 4

- A log-in page as described in requirements 1.3.0
    - User is allowed access if details entered match a user in the 'Users' database

SUCCESSFUL

Users are able to successfully login to their accounts using the details stored in the credentials table of the database. Users can only sign-in to an account with a matching username and password. This was showcased in the end of stage 2 testing in tests 1a, 4a, 4b, 4c

Despite the functionality being successful I am not 100% happy with this section of the project. The design of the web page is not exactly to my prior designs, as the buttons and input boxes are not styled, and the input boxes and submit button are displayed on the left-hand side of the page not in the center.

The login page is also prone to SQL injection attacks. It was known at the beginning of my project that security was a limitation, therefore this is not a huge issue, however it is something that I would like to fix if I were to re-create this project. This could make the program unsuitable for many potential end users, as their access credentials are at risk. This was uncovered during my post development robustness tests, test 3.1.

Success Criteria Bullet point 5

- A point-of-sale system user interface as described in requirements 2.1

SEMI-SUCCESSFUL

The user interface provides the system operator with an area to add any item to a transaction within 10 seconds. (Showcased in **general testing stage5.mp4** between 1:58 and 2:04 (8seconds)). Each sale item has a button that can be clicked to add it to a transaction. If the transaction is closed, but there is not enough of that product in stock, the transaction is re-opened, and no products are deducted from the stock management system. The total cost of the current transaction and the items that it contains is displayed within a college on the left-hand side of the page that takes up approximately ¼ of the page width. This is showcased in **general testing stage5.mp4**.

This web page is semi-successful as the transaction column is not exactly as expected. The close transaction button and the total cost is displayed below the lowest item in the transaction, rather than in a constant position at the bottom of the column as shown in my designs. This could retract from the efficiency of the program, as the system operator will need to identify the location of the button at the end of each transaction, rather than pressing in a constant area of the system to complete a transaction.

Another issue with this web page is that it loads incorrectly on some device types. I never intended on optimising this program for multiple devices, however I did intend that the program would load correctly on the vast majority of device types and would be usable. However, on some device types (particularly smart phones) the category buttons on the right-hand side of the interface load incorrectly, displaying below the other elements. This mis-shapes the point-of-sale system making it very difficult to use on these devices. If I were to create this program again, this would be a key area that I would like to fix.

The final issue with the point-of-sale system is the fact that I was unable to implement the ability to customise a dish that is being sold. This had proven to be a difficult task as much of the design for this feature was over sought, such as the need for further database tables in order to display the contents added and removed from a dish on a transaction. I believe this would have been a very useful and only moderately challenging feature to implement, had it been correctly thought out during my design phase of the project.

Success criteria bullet point 6

- Clickable button to add sale items to transactions as described in requirements 2.2.0

SUCCESSFUL

Each sale item button can be clicked and that item is added to the current transaction and the total cost is increased by a pre-defined amount.

I would be 100% happy with this feature, but there is one quite large let down. The sale item buttons are not displayed as initially intended. These buttons are displayed only vertically, rather than being in a grid pattern. This means that there is a large amount of wasted space on the point-of-sale web page which could contain more sale items, hence making my project appeal to a greater audience. This is something that I would most definitely fix if I were to re-make this project as I believe that it makes the system less appealing compared to similar systems.

Success criteria bullet point 7

- The ability for new sale item buttons to be added to the system as described in requirements 2.2.3

SEMI-SUCCESSFUL

The user is able to press the new sale item button and select the button they would like to edit from the drop-down menu and edit the data held by that sale item button. This is shown in end of stage 5 testing test 1a.

This is not implemented as initially intended in my design – the user was supposed to enter an edit mode where they could select a sale item button and edit the details from there, without the need for a drop-down menu. I believe this approach would have been much more user friendly, however I was entirely unsure how to implement this into the project, as I have zero prior experience with such a feature. If I were to make this project again, I would like to implement this feature in the intended way giving the program a more professional feel.

Success criteria bullet point 8

- The ability to refund a transaction as outlined in requirements 2.4

SEMI-SUCCESSFUL

The feature itself is 100% successful, the system operator can refund any amount using a text input field and that amount will be deducted from the daily total. However, this was not implemented as shown by my initial designs. This current implementation is shown in end of stage 6 testing tests 2a. 3a. 4a. 4b. 4c.

My initial designs used a webpage of a calculator design where the system operator would press the buttons on the calculator to input the amount to refund and the refund amount would be displayed to the system operator, where they could then press an enter key to confirm. However, during the implementation phase of this feature I decided that I did not really know how I would implement this, and I believe that I was over-complicating a simple task.

If I were to program this project again, I would implement this feature similar to its current implementation, totally disregarding the calculator design initially intended. I would however like to add some styling to the input box and submit button, and I would also like to make this fill the webpage a lot more, as currently it is very empty making the program feel unprofessional.

Success criteria bullet point 9

- Ability to create a new transaction as outlined in requirements 2.5

SUCCESSFUL

This feature was 100% successful. The system operator can either open a new transaction manually alongside other transactions, or if there is only one transaction left and it is closed, the system will automatically open a new transaction. I am very happy with the way that this feature works and would not implement it any different if I were to re-create this project. This feature is showcased in stage 9 test 6a.

Success criteria bullet point 10

- A sale item database table as outlined in requirements 2.16

SEMI-SUCCESSFUL

The "sale item" database table contains the name of each sale item, its price, and a category.

This feature is semi-successful as its functionality exists entirely as intended, although it was not implemented as initially intended. The "sale item" database table according to my designs should have contained a list of all of the products that that sale item contains. However, this was not included within the "sale item" database table. Instead, the sale item database table was re-named as the menu database table of each company and the products of each sale item was stored in its own database table called sale item contents. This was not included in my design. If I was to program this again, I would like to make use of linking database tables to link these tables together. Despite this, these two+ tables serve the same purpose as the sale item database table in my designs. This is showcased in the end of stage 5 testing test 1a

Success criteria bullet point 11

- A stock management database table as outlined in requirements 3.1

SUCCESSFUL

The stock management database table contains the name of all products, their current stock level, minimum stock level, maximum stock level, whether they are ordered or not, a supplier name, a supplier phone number, an amount due and a date due. This is everything that was intended to be included within this database table. Therefore, it was entirely successful, and I would not change anything about this. This is showcased in the end of stage 4 testing tests 1b, 1c.

Success criteria bullet point 12

- The ability to add new products to the stock management database table as outlined in requirements 3.3

SUCCESSFUL

A +new product button is displayed in the corner of the overview table on the stock management page. When pressed, the user is redirected to a web page in which they can input the details of the new sale item, and upon pressing submit it will be inserted into the stock management database table. This feature was implemented entirely as intended, and I am 95% happy with it. The only change that I would make with this feature is that I would like to add some styling to the edit button, making it fit nicely into the corner of the overview table giving my program a more professional look. This is showcased in the end of stage 4 testing tests 1b, 1a.

Success criteria bullet point 13

- Stock management database table automatically edited after a transaction is complete as outlined in requirements 3.4

SUCCESSFUL

When the complete transaction button is pressed, so long there are enough of a product to fulfil that transaction, the products within the transaction are deducted from the stock management database. I am 100% happy with this feature and would not change anything about it if I were to re-make this project. This is showcased in the end of stage 5 testing test 5b.

Success criteria bullet point 14

- Manual edits to the stock management database table to compensate for out of the box factors such as wastages as outlined in requirements 3.2.1

SEMI-SUCCESSFUL

An edit hyperlink is displayed in the final column of each product in the stock management overview table. This hyperlink re-directs the system operator to a new web page where they can edit the current values of the product. Although this feature is working, it was not implemented as initially intended.

Instead of a button being used to redirect the user, instead, a hyperlink is used. This was to easily relay data between the two files. If I were to program this project again, I would keep this as it is currently. However, I would like to style the hyperlink to appear like a button, much like those on the toolbar. This would give the program a more professional look, widening the target audience. The feature as currently implemented is shown in the end of stage 4 testing test 2a.

I would also like to have styled the buttons on the edit product page, as these buttons are currently default and only take up a minute area of the web page. Fixing these issues would make the program more efficient as it would require a less precise mouse click to click the input boxes and make the page appear more professional.

Success criteria bullet point 15

- Stock arrival date contained within the stock management database used to prompt operators whether the stock has arrived as outlined in requirements 3.5

SUCCESSFUL

The due date & time and due quantity of stock is held within the stock management database table. When the date and time in the stock management table and the current date and time is less than or equal to the current time (in GMT), an alert is sent to the system operator asking whether the stock has been delivered or not. If stock is delivered, the arrival quantity is updated in the stock management database table. Otherwise the system operator is sent to edit this product, where they can then alter the due amount and due date.

This works entirely as expected and I am 100% happy with this. If I were to re-make this project, I do not expect that I would alter any of this feature.

Success criteria bullet point 16

- Low stock level alerts output to the operator as detailed in requirements 3.6.0

SUCCESSFUL

An alert is sent to the point-of-sale system user interface whenever a product is below its minimum stock level in the stock management database. The alert contains the name of the product that is low on stock, the name of it's supplier and their contact details. The alert has two buttons, 'ordered' and 'Ignore'. After inputting the amount to order and the date and time that product is due, the stock management system will mark the product as ordered, and will update the stock management database table with this information. The ignore button will ignore the alert, it will not be triggered again. This product will have to then be manually ordered.

This was implemented entirely intended, and I am 100% happy with the functionality of this feature. However, I would like to make the user interface appear more professional by styling the buttons using CSS, moving away from the default theme of the user's browser.

Limitations

Although I knew that security would be a limitation during the design phase of my project, I was disappointed that my program failed the SQL injection. This is a huge security risk that could lead to accounts being accessed on my own web page as well as others. This will be a very high priority fix during the maintenance of this program in the future to keep users and their accounts safe.

My project is also not very aesthetically pleasing. This was identified in my initial limitations; however, the final project included a lot more limitations on design than initially intended. Very few pages contain styled buttons making the project appear much less professional than it could be. This will likely narrow my target audience.

Although the initial goal of having a system operator apply any sale item to a transaction within 10 seconds was successful, the sale item buttons on the point-of-sale page are not displayed as initially intended. This severely decreases efficiency and means that there is a lot of empty space on this page. This is likely to narrow my target audience as the efficiency of the system may not meet the need of some potential end users.

Maintenance

The greatest problem with this project in its current state is that the point-of-sale system is much less efficient than it was intended to be. The sale item buttons are not displayed in a grid, they are only displayed in a vertical column. This means that a lot fewer sale item buttons are presented to the system operator at one time, hence decreasing the efficiency of the system. It also makes this specific web page look unprofessional as there is a lot of empty space.

Future maintenance would fix this web page, where the sale items are displayed in a grid filling the web page, making the system more efficient and appear to be more professional.

If the stakeholders with to add or remove features, this will be easy in many cases as a lot of features are contained within their own web page, with the vast majority of the feature contained within its own files. The important variables within the program are all $_SESSION variables (and therefore global) so can easily be accessed if a new feature is added. It may be beneficial to create a single file of which functions are contained within in the future to make the project more modular, and therefore easier to maintain.

A lot of the styling for the web pages are contained within a style sheet. This makes it easy for the stakeholder to re-design the system colours and theme if they wish to in the future.

The code contains a lot of annotations, making it easier for another programmer to understand what is happening in each stage of the project. Where possible, functions have been used to make lengthy pieces of code easily identifiable by a small phrase, and each of these functions have a good description of what they do, what their inputs should be, and what its output should be above the function declaration.

Future versions of this project should include increased account security, preventing malicious attacks against this system and should include a version of this program suitable for mobile devices to not only widen the target audience for this system, but also improve accessibility.

Future Developments

I did not have the time nor ability to implement some features I would have liked to have seen in this project. There are several features which could have improved the efficiency of the project, hence making the program a better solution to the task and I feel that the project could have been more professional in many aspects. Below is a summary of changes I would like to make in this project:

- Implement images inside sale item buttons – This is a feature I had planned to include earlier in the project but lacked the knowledge on uploading and storing image files inside a database table to implement this. These images should have complemented the name of each sale item button improving efficiency of the program as outlined in user requirements 2.2.1
- Implementation of sale item buttons in a grid fashion – due to a bug with the program sale item buttons are only displayed in a vertical fashion. It was intended that the buttons were displayed in a grid fashion, allowing the page to be filled. This increases efficiency of the program, as more sale item buttons can be seen at one time by the system operator. This feature is displayed in my user interface design of the point-of-sale system
- Including the ability to customise a sale item – currently a sale item cannot be modified. This means that the stock management system must be manually updated if the system operator would like to add, remove, or exchange a product within a sale item dish. Such a feature to fix this problem was initially included in my project (user requirements 2.14.0), however I quickly realised that this would be a very time-consuming task. Implementing this feature would add to the program, widening the target audience drastically.
- Alerts messages via other methods – Currently, alert messages are only sent within the system. I would like to have other methods of sending these alerts, as originally intended in the project with SMS and email alerts (as in user requirements 3.6.2 and 3.6.3). These could not be implemented as it was a much larger task than I initially expected, and I was clueless on where to start with this feature. It would however widen my target audience by offering features that similar programs do not offer.

Conclusion

I believe that this program has been a very big success. Despite some features not being included within the project and a few bugs, the program solves the task that I intended to solve at the beginning of the project. It is efficient, intuitive, and can help companies improve their profitability by requiring less workers due to a more efficient check-out process and provide a much easier task for those using the system. While adding some of the improvements mentioned in the future development section would improve my project by increasing efficiency and widening the target audience, this solution has far exceeded the expectations of my stakeholders therefore I am happy with the result.

Code

Database_connect.php

```php
<?php
/* CONNECT TO LOCAL HOST */
function openConnection()
{
    $servername = "localhost"; //name of the server
    $username = "username"; //server login username
    $password = "password"; //server login password
    $database = "Users"; //database to connect to name

    $connection = new mysqli($servername, $username, $password, $database);

    return $connection;
}

/* Close local host */
function closeConnection($connection){
    $connection -> close();
}
?>
```

Dropdown.php

```
<!DOCTYPE html>
<head>
    <link rel="stylesheet" href="Styles.css" type="text/css"> <!-- link HomePage to
style sheet -->
</head>

<html>
<body>

<div class="toolbar">
    <a href="HomePage.php">Home</a>
    <a href="MyMenus.php">MyMenus</a>
    <a href="stockManagement.php">Stock Management</a>
    <a href="pointOfSale.php">POS</a>
    <?php session_start();
    $username = $_SESSION['username']; // get username from login process
    ?>
    <a style="float: right;" class="current" href="dropdown.php"> <?php echo $username;
?></php></a>
</div>

<form action="refund.php">
    <button type="submit" name="refund">Refund</button>
</form>

<form action="process_dayEnd.php">
    <button type="submit" name="endOfDay">End Of Day</button>
</form>

<?php
/* Open DB connection */
include "database_connect.php";
$connection = openConnection();

/* Retrieve company name */
$companyName = $_SESSION['companyName'];

/* Retrieve data from transaction history table where transaction is complete */
$tableName = "transactionhistory_".$companyName; // Calculate table name
$query = "SELECT * FROM $tableName WHERE Complete = 'yes'";
$transactionhistoryResult = mysqli_query($connection,$query);

/* Calculate number of returned rows */
$numRows = 0;
if($transactionhistoryResult == false){
    $numRows = 0;
}
else{
    $numRows = mysqli_num_rows($transactionhistoryResult);
}

/* Loop through each row of the results to the query, adding to the variable total the
total cost of each
 * transaction in the transaction history table that is complete
 */
$total = 0;
for ($i=0; $i<$numRows; $i++){
    $tableRow = mysqli_fetch_assoc($transactionhistoryResult);

    $total = $total + $tableRow['Total'];
}

echo "Daily Total: ".$total; // Ouput the daily total
```

```
/* Output users requested to your company */
?>
<br><br><h3> The Following Users Request Standard Permissions To Your Company:</h3>
<?php
$query = "SELECT * FROM Credentials WHERE Company = '$companyName' AND accountType =
'unknown'";
$result = mysqli_query($connection,$query);

/* Calculate the number of rows returned by the query */
$numRowsReturned = 0;
if($result == false){
    $numRowsReturned = 0;
}
else{
    $numRowsReturned = mysqli_num_rows($result);
}

/* Loop through each row of the query result outputting the username and a button to
confirm that their account type
   can be updated from unkown to standard, allowing them access to the company's
systems with standard permissions. */
for($i = 0; $i<$numRowsReturned; $i++) {
    $resultRow = mysqli_fetch_assoc($result);

    ?>
    <!-- Output a button that when pressed updates the intended user's account type
from unkown to standard -->
    <!-- echo $resultRow['Username'] outputs the username of a user with account type
unknown and company = the current user's company -->
    <form action="process_updateAccountType.php">
        <button type="submit">Allow User "<?php echo $resultRow['Username'] ?>"
Standard Access To Your Company's System</button>
        <!-- Create a hidden input field to post the username of a user to the
processing file -->
        <input type="hidden" value="<?php echo $resultRow['Username'] ?>"
name="usernameToUpdate">
    </form>

    <?php
}

/* Check that the current account type is allowed on this web page */
require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"dropdown.php");

?>
```

editProduct.php

```
<?php

/* Start and initialise session */
session_start();
$_SESSION['tableName'] = $_GET['TableName'];
$_SESSION['productID'] = $_GET['ProductID'];

/* Open DB connection */
include "database_connect.php";
$connection = openConnection();

/* GET EXISTING RESTOCK DATE FROM STOCK MANAGEMENT TABLE */
$tableName = 'stockmanagementtable_'.$_SESSION['companyName'];
$productID = $_GET['ProductID'];
$query = "SELECT ArrivalDate,ArrivalAmount FROM $tableName WHERE ProductID =
'$productID' ";
$queryResult = mysqli_query($connection,$query);

/* Enter fetched values into variables */
$tableRow = mysqli_fetch_assoc($queryResult);
$arrivalDate = $tableRow['ArrivalDate'];
$arrivalAmount = $tableRow['ArrivalAmount'];

echo "
<!-- print form -->
    <!-- Display product Name input field -->
    <form action='process_editProduct.php'>
    <label for='ProductName'>Product Name:</label>
    <input type='text' name='ProductName' value='$_GET[ProductName]' required>

    <br>

    <!-- Display current stock value input field -->
    <label for='CurrentStockValue'>Current Stock:</label>
    <input type='text' name='CurrentStockValue' value='$_GET[CurrentStockValue]'
required>

    <br>

    <!-- Display minimum stock value input field -->
    <label for='MinimumStockValue'>Minimum Stock:</label>
    <input type='text' name='MinimumStockValue' value='$_GET[MinimumStockValue]'
required>

    <br>

    <!-- Display checkbox input field for whether or not a product has been ordered -->
    <label for='Ordered'>On order:</label>
    <input type='hidden' value='off' name='Ordered'> <!-- to solve not being posted if
off -->
    <input type='checkbox' name='Ordered'>

    <br>

    <!-- Supplier name text input field -->
    <label for='SupplierName'>Supplier Name:</label>
    <input type='text' name='SupplierName' placeholder='Supplier'
value='$_GET[SupplierName]'>

    <br>

    <!-- phone number input field of type telephone -->
    <label for='PhoneNumber'>Phone Number:</label>
    <input type='tel' name='PhoneNumber'  pattern='[0-9]{11}' placeholder='07999123456'
value='$_GET[PhoneNumber]'>
```

```
    <br>

    <!-- restock due date input field of type date -->
    <label for='dueDate'>Restock Date:</label>
    <input type='datetime-local' name='dueDate' value='$arrivalDate'>

    <br>

    <!-- restock quantity input field of type number -->
    <label for='Quantity due:'>Restock Amount:</label>
    <input type='number' name='restockQuantity' placeholder='0' step='1'
value='$arrivalAmount'>

    <br>
    <button type='submit'>Done</button>
</form>";

require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"EditProduct.php");

?>
```

editSaleItem.php

```php
<?php
/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

session_start();
$tableName = $_SESSION['companyMenuTableName'];

$query = "SELECT SaleItemName FROM $tableName";
$result = mysqli_query($connection,$query);

echo "<form action='process_editSaleItem.php'>";
echo "<label for='saleItemToEdit'>Choose a Sale Item: </label>";
echo "<select name='saleItemToEdit' id='saleItemToEdit'>";

for ($i=0;$i<mysqli_num_rows($result);$i++){
    $row = mysqli_fetch_assoc($result);
    $saleItem = $row['SaleItemName'];
        echo "<option value='$saleItem'>$saleItem</option>";
}
echo "<option value='+new'>New Sale Item</option>";
echo "</select>";

echo "<input type='text' name='saleItemName' placeholder='Banana'>";
echo "<input type='number' name='saleItemPrice' placeholder='0.00' step='0.01'
min='0.00' minlength='3'>";
echo "<input type='text' name='saleItemCategory' placeholder='Food'>";
echo "<textarea name='products' placeholder='bread,bread,cheese'></textarea>";
echo "<button type='submit'>Confirm</button>";

require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"editSaleItem.php");
?>
```

Functions.php

```
<?php

function isUserAllowedAccessToThisPage($accountType,$fileName){

    /* Allows only owner account types can access the dropdown.php file */
    if($fileName == "dropdown.php" and ($accountType == "unknown" or $accountType == "standard")){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header('Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }

    /* Allows only owner account types can access the EditProduct.php file */
    if($fileName == "EditProduct.php" and ($accountType == "unknown" or $accountType == "standard")){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header('Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }

    /* Allows only owner account types can access the editSaleItem.php file */
    if($fileName == "editSaleItem.php" and ($accountType == "unknown" or $accountType == "standard")){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header('Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }

    /* Allows only owner account types can access the refund.php file */
    if($fileName == "refund.php" and ($accountType == "unknown" or $accountType == "standard")){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header('Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }

    /* Allows owner and standard account types to access the MyMenus.php file */
    if($fileName == "MyMenus.php" and $accountType == "unknown"){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header('Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }

    /* Allows owner and standard account types to access the stockManagement.php file */
    if($fileName == "stockManagement.php" and $accountType == "unknown"){

        echo "<script>alert('your account type does not permit you to be here!')</script>"; //Outputs alert box message
        header('Refresh: 0; URL=HomePage.php'); //Redirects the user away from the page
        exit;
    }
}
?>
```

hasStockArrived.php

```
<?php
session_start(); // start the current session
?>

Has the re-stock of product:
<?php echo "<br>".$_GET['productName']."<br>"; ?>
Arrived? <br>
Quantity Due: <?php echo $_GET['arrivalQuantity']."<br>"; ?>

<style>
.displayInline {
    display: inline-block;
}
</style>

<!-- submit user to processing page if stock has arrived -->
<form action='process_stockArrived.php' class="displayInline">
    <input name='productID' type="hidden" value="<?php echo $_GET['productID']; ?>">
    <input name='currentStockValue' type="hidden" value="<?php echo
$_GET['currentStock']; ?>">
    <input name='arrivalQuantity' type="hidden" value="<?php echo
$_GET['arrivalQuantity']; ?>">
    <button type='submit' value=''>Stock Has Arrived</button>
</form>

<!-- redirect user to the product edit page for a product if a re-stock has not yet
arrived -->
<form action='EditProduct.php' class="displayInline">
    <input name='ProductID' type="hidden" value="<?php echo $_GET['productID']; ?>">
    <input name='TableName' type="hidden" value="<?php echo
"stockmanagementtable_".$_SESSION['companyName']; ?>">
    <input name='ProductName' type="hidden" value="<?php echo $_GET['productName'];
?>">
    <input name='CurrentStockValue' type="hidden" value="<?php echo
$_GET['currentStock']; ?>">
    <input name='MinimumStockValue' type="hidden" value="<?php echo
$_GET['minimumStock']; ?>">
    <input name='Ordered' type="hidden" value="<?php echo $_GET['ordered']; ?>">
    <input name='SupplierName' type="hidden" value="<?php echo $_GET['supplierName'];
?>">
    <input name='PhoneNumber' type="hidden" value="<?php echo $_GET['phoneNumber'];
?>">
    <button type='submit' value=''>Stock Hasn't Arrived</button>
</form>
```

HomePage.php

```
<!DOCTYPE html>
<head>
    <link rel="stylesheet" href="Styles.css" type="text/css"> <!-- link HomePage to
style sheet -->
</head>

<html>
<body>

<!-- toolbar -->
<div class="toolbar">
    <a class="current" href="HomePage.php">Home</a>
    <a href="MyMenus.php">MyMenus</a>
    <a style="float: right;" href="Signup.php">Sign up</a>
    <a style="float: right;" href="Login.php">Log-in</a>
</div>

<h6> TUTORIAL </h6>

<!-- embed a youtube video into the centre of the HomePage -->
<div id="embeddedVideo">
    <iframe width="60%" height="500" style="border: none;" allowfullscreen

src="https://www.youtube.com/embed/j5a0jTc9S10?rel=0&amp;controls=0&amp;showinfo=0">
    </iframe>
</div>

</body>
</html>
```

Login.php

```
<?php
?>

<head>
    <link rel="stylesheet" href="Styles.css" type="text/css"> <!-- link Login Page to
style sheet -->
</head>

<form action="process_login.php"> <!-- send entered data into process_login.php for
processing -->

    <!-- toolbar -->
    <div class="toolbar">
        <a href="HomePage.php">Home</a>
        <a href="MyMenus.php">MyMenus</a>
        <a style="float: right;" href="Signup.php">Sign up</a>
        <a class="current" style="float: right;" href="Login.php">Log-in</a>
    </div>

    <div class ="loginContainer">

        <h1>Log in</h1>

        <label for="username">Username</label>
        <input type="text" name="username" placeholder="Enter Your Username Here"
required>

        <br>

        <label for="password">Password</label>
        <input type="text" name="password" placeholder="Enter Your Password Here"
required>

        <br>

        <button type="submit">Log in</button>
    </div>

</form>

</body>
</html>
```

LowStock.php

```php
<?php
session_start(); // start the current session

$row = $_SESSION['returnedRows'];
$_SESSION['ProductID'] =  $row['ProductID']; // get the id of the current product

echo "<head>
        <link rel='stylesheet' href='Styles.css' type='text/css'>
      </head>";

/* Output data regarding product & supplier */
echo "Product Name: ".$row['ProductName']."<br>Minimum Stock Value:
".$row['MinimumStockValue']."<br>".
     "Current Stock Value: ".$row['CurrentStockValue']."<br>Supplier Name:
".$row['SupplierName']."<br>".
     "Phone Number: ".$row['Phone'];

/* Output:
 * Time & Date due input box
 * Amount due input box
 * Ordered form submit button
*/
echo "
    <form style='display: inline' action='process_lowStock.php'>
    <br><br><label>Ordered Amount: <input type='number' name='orderAmount' step='1'
value='0'></label><br>
    <label>Due Date & Time: <input type='datetime-local' name='dateTime'></label><br>
    <button type='submit'>Ordered</button> <!-- Output submit button -->
    </form>
    ";

/* Output ignore submit button */
echo "<form style='display: inline'  class='lowStock' action='pointOfSale.php'>
      <button type='submit' >Ignore</button>
      </form>";

?>
```

MyMenus.php

```
<?php
?>

<!DOCTYPE html>
<head>
    <link rel="stylesheet" href="Styles.css" type="text/css"> <!-- link HomePage to
style sheet -->
</head>

<html>
<body>

<!-- toolbar -->
<div class="toolbar">
    <a class="current" href="HomePage.php">Home</a>
    <a href="MyMenus.php">MyMenus</a>
    <a style="float: right;" href="Signup.php">Sign up</a>
    <a style="float: right;" href="Login.php">Log-in</a>
</div>

<form action="process_existingMenu.php">
    <div class = "menuButtonContainer">

        <label for="existingMenuName">Existing Menu Name:
        <input type="text" name="existingMenuName" id="existingMenuName"
placeholder="Enter pre-existing menu name" required></label>

        <button type="submit" name="submit">Confirm</button>
    </div>
</form>

<form action="process_newMenu.php"> <!-- send entered data into process_myMenu.php for
processing -->
    <div class = "newMenuButtonContainer">

        <label for="MenuName">New Menu Name:
        <input type="text" name="MenuName" placeholder="Menu1" required></label>

        <button type="submit" name="submit">+New Menu</button>
    </div>
</form>


</body>
</html>

<?php
session_start();

require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"MyMenus.php");
?>
```

newProduct.php

```
<html>
<form action='process_newProduct.php'>
    <label for='ProductName'>Product Name:</label>
    <input type='text' name='ProductName' placeholder='Carrot' required>

    <br>

    <label for='CurrentStockValue'>Current Stock:</label>
    <input type='text' name='CurrentStockValue' placeholder='0' required>

    <br>

    <label for='MinimumStockValue'>Minimum Stock:</label>
    <input type='text' name='MinimumStockValue' placeholder='0' required>

    <br>

    <label for='Ordered'>On order:</label>
    <input type='hidden' value='off' name='Ordered'> <!-- to solve not being posted if
off -->
    <input type='checkbox' name='Ordered' value='on'>

    <br>

    <label for='SupplierName'>Supplier Name:</label>
    <input type='text' name='SupplierName' placeholder='Supplier'>

    <br>

    <label for='PhoneNumber'>Phone Number:</label>
    <input type='tel' name='PhoneNumber'  pattern='[0-9]{11}'
placeholder='07999123456'>

    <br>
    <button type='submit'>Done</button>
</form>
```

pointOfSale.php

```
<!DOCTYPE html>
<head>
    <link rel="stylesheet" href="Styles.css" type="text/css"> <!-- link HomePage to
style sheet -->
</head>

<html>
<body>

<div class="toolbar">
    <a href="HomePage.php">Home</a>
    <a href="MyMenus.php">MyMenus</a>
    <a href="stockManagement.php">Stock Management</a>
    <a class="current" href="pointOfSale.php">POS</a>
    <!-- Search Bar -->
    <?php
    session_start();

    /* Open Database Connection */
    include "database_connect.php";
    $connection = openConnection();

    /* Search Bar*/
    echo "
    <style>
    .searchBar{
        width: 50%;
        height:50px;
        background-color: #3aff3a;
        border: none;
        outline: none;
        resize: none;
        color: white;
        font-weight: bold;
        font-size: 40px;
        margin: 0 auto;
        float: left;
        overflow: hidden;
    }

    .searchBar:hover{
    background-color: #00ba00;
    }

    .searchSubmit{
        width: 10%;
        height: 55px;
        background-color: #3aff3a;
        border: none;
        outline: none;
        resize: none;
        color: white;
        font-weight: bold;
        font-size: 20px;
        text-align: center;
        float: left;
        overflow; hidden:

    }

    .searchSubmit:hover{
        background-color: #00ba00;
    }

    </style>
    <form style='height: 0; width: 0; display: inline'  action='process_search.php'>
```

```
    <textarea class='searchBar' name='searchBar' placeholder='Search:'></textarea>
    <button class='searchSubmit' type='submit'>Submit</button>
    </form>

    ";

    /* Check for re-stock having arrived */
    $tableName = "stockmanagementtable_".$_SESSION['companyName'];
    $query = "SELECT * FROM $tableName";
    $stockManagementTableResult = mysqli_query($connection,$query);

    /* Calculate the number of rows returned by query */
    $numRowsReturnedByQuery = 0;
    if ($stockManagementTableResult == false){
        $numRowsReturnedByQuery = 0;
    }
    else{
        $numRowsReturnedByQuery = mysqli_num_rows($stockManagementTableResult);
    }

    /* Calculate current date & time */
    date_default_timezone_set("GMT");
    $currentDateLong = date("c");
    $currentDateShort = substr($currentDateLong, 0, 16);
    $currentDateShort = str_replace("T"," ","$currentDateShort"); /* remove the letter
T from the

string and replace with a space

character */

    /* Search through all rows returned by the query, with each result checking whether
stock has been ordered
       and if it has been ordered, whether the arrival date of new stock is less than
or equal to the current
       date and time */
    for($i=0; $i<$numRowsReturnedByQuery; $i++) {
        /* Fetch new row of the query result */
        $row = mysqli_fetch_assoc($stockManagementTableResult);

        /* Fetch arrival date and time from the current row */
        $arrivalDate = $row['ArrivalDate'];
        $arrivalDate = substr($arrivalDate, 0, 16); // shorten arrival date to remove
seconds

        /* if date&time is not null, check whether the date and time of stock due to
arrive is less than the
           current date and time */
        if (($arrivalDate <= $currentDateShort and $arrivalDate != null)) {

            /* Fetch data from each column of the result into variables */
            $arrivalQuantity = $row['ArrivalAmount'];
            $productID = $row['ProductID'];
            $productName = $row['ProductName'];
            $currentStock = $row['CurrentStockValue'];
            $minimumStock = $row['MinimumStockValue'];
            $ordered = $row['Ordered'];
            $supplierName = $row['SupplierName'];
            $phoneNumber = $row['Phone'];

            /* Calculate redirect URL */
            $redirectURL =
"hasStockArrived.php?productID=".$productID."&productName=".$productName.

"&arrivalDate=".$arrivalDate."&arrivalQuantity=".$arrivalQuantity.

"&currentStock=".$currentStock."&minimumStock=".$minimumStock.
```

```
"&ordered=".$ordered."&supplierName=".$supplierName."&phoneNumber=".$phoneNumber;

            /* Redirect the user */
            header('refresh: 0; URL = '.$redirectURL);// redirect to home page
            exit;
        }
    }


    ?>

    <?php
    $username = $_SESSION['username']; // get username from login process
    ?>
    <a style="float: right;" href="dropdown.php"> <?php echo $username; ?></php></a>
</div>

<style>
    span.leftGUI{
        display: inline-block;
        float: left;
        width: 20%;
        height: 655px;
        padding-top: 10px;
        border: 3px solid black;
        background-color: #cbcbcb;
    }

    button.newSaleItem{
        display: inline-grid;
        width: 22%;
        height: 15%;
        border: 2px solid black;
        background-color: lime;
        margin: 10px;
    }

    span.midGUI{
        margin: auto;
        display: inline-block;
        width: 71%;
        height: 655px;
        padding-bottom: 5px;
        padding-top: 13px;
        padding-left 10px;
        border-top: 3px solid black;
        border-bottom: 3px solid black;
        background-color: white;
        font-size: 18px; /* Change to zero, testing with above 0 */
    }

    span.rightGUI{
        display: inline-block;
        width: 8%;
        height: 655px;
        padding-top: 10px;
        float:right;
        border: 3px solid black;
        background-color: #0404cb;
        padding-left: 5px;
    }

    button.saleItem{
        display: inline-grid;
        width: 22%;
        height: 15%;
```

```
        border: 2px solid black;
        background-color: white;
        margin: 10px;
    }

    button.completeTransaction{
        background-color: red;
        border: none;
        width: 300px;
        height: 50px;
        color: white;
        font-size: larger;
        font-weight: bold;
    }
</style>

<div>

    <!-- Transaction Column -->
    <span class='leftGUI'>
        <?php
            $companyName = $_SESSION['companyName']; // initiate $companyName

            /* Check for low stock, and trigger an alert if necessary */
            checkForLowStock($connection,$companyName);

            //Set the transactionID to the default transaction if there is not an
already existing transactionID
            if($_SESSION['transactionID'] == "") {
                $query = "SELECT TransactionID FROM transactionhistory_$companyName
WHERE Complete = 'no'";
                $result = mysqli_query($connection, $query);

                $row = mysqli_fetch_assoc($result);

                $_SESSION['transactionID'] = $row['TransactionID'];
            }
?>

            <style>
                .displayInline{
                    display: inline-block;
                }

                .displayInlineH2{
                    padding-left: 10%;
                    display: inline-block;
                }
            </style>

            <?php
            /* Check if any transactionID values have been posted */
            if(!empty($_POST)){
                if($_POST['transactionID'] != "") {
                    /* Search in the transaction history table for a transaction with
the requested transactionID that is open */
                    $tableName = "transactionhistory_".$companyName; //calculate the
name of the table to insert into the SQL query
                    $requestedTransactionID = $_POST['transactionID'];
                    $query = "SELECT * FROM $tableName WHERE
                              TransactionID = '$requestedTransactionID' AND Complete =
'yes'";
                    $result = mysqli_query($connection,$query);

                    /* Calculate the number of rows returned by the query */
                    $numberOfRows = 0;
                    if($result == false){
```

```
                    $numberOfRows = 0;
                }
                else{
                    $numberOfRows = mysqli_num_rows($result);
                }

                /* If there were results found, update the current transactionID */
                if($numberOfRows == 0) {
                    /* Update the transactionID session variable with the posted
transactionID */
                    $_SESSION['transactionID'] = $_POST['transactionID'];
                }
            }
        }
        $transactionID = $_SESSION['transactionID'];

        /* search for a transaction with the current transactionID */
        $transactionHistoryTable = "transactionhistory_".$companyName;
        $query = "SELECT TransactionID FROM $transactionHistoryTable WHERE
TransactionID = '$transactionID'";
        $result = mysqli_query($connection,$query);

        /* Calculate the umber of rows returned by the query */
        $numRows = 0;
        if($result == false){
            $numRows = 0;
        }
        else{
            $numRows = mysqli_num_rows($result);
        }

        /* If no transaction ID is returned, create a new transaction with the
given transactionID */
        if ($numRows == 0){
            /* Open a new transaction with the provided transactionID */
            $query = "INSERT INTO
transactionhistory_$companyName(TransactionID,Complete) VALUES('$transactionID','no')";
            mysqli_query($connection, $query);

            // Create a new transaction table
            $newTransactionID = $transactionID;
            $newTableName = "transaction_".$newTransactionID."_".$companyName;
            $query = "CREATE TABLE `users`.`$newTableName` ( `SaleItemID` INT(9)
NOT NULL AUTO_INCREMENT ,
            `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT
NOT NULL ,
             PRIMARY KEY (`SaleItemID`))";
            mysqli_query($connection,$query);
        }

        ?>

        <!-- Outpupt the transaction ID -->
        <h2 class="displayInlineH2" > Transaction ID: </h2>
        <form class="displayInline" method="post">
            <input class="displayInline" type='number' value='<?php echo
$transactionID; ?>' step='1' size='3' name="transactionID">
            <button class="displayInline" type='submit'>✓</button>
        </form>

<?php


        //Display the current items on the transaction
        $tableName = "transaction_".$transactionID."_".$companyName;
        $query = "SELECT SaleItemName,Cost,Quantity FROM $tableName";
        $result = mysqli_query($connection,$query);
```

```
            $total = 0; // Stores the total for the current transaction

            /* Only runs if the SQL query produces results */
            if($result != false) {

                /* Loops through each row of the results from the SQL table */
                for ($i = 0; $i < mysqli_num_rows($result); $i++) {
                    $row = mysqli_fetch_assoc($result);
                    $saleItemName = $row['SaleItemName'];
                    $saleItemCost = "£" . $row['Cost'];
                    $quantity = $row['Quantity'];

                    ?>
                    <p align='center'><?php echo $saleItemName; ?><br>
                    Quantity: <?php echo $quantity; ?> <br>
                    <?php echo $saleItemCost; ?> </p>

                    <style>
                    .button{
                    margin: 0;
                    left: 50%;
                    position: absolute;
                    border: none;
                    font-weight: bold;
                    color: white;
                    width: 25px;
                    height: 25px;
                    font-size: 16px;
                    }

                    </style>

                    <!-- Increment Button -->
                    <form action='process_saleItem_increment.php'>
                    <button class='button' style='transform: translate(-2200%, -265%);
                    background-color: lime;' type='submit'>+</button>
                    <input type='hidden' value='<?php echo $saleItemName; ?>'
name='saleItem'>
                    </form>

                    <!-- Decrement button -->
                    <form action='process_saleItem_decrement.php'>
                    <button class='button' style='transform: translate(-2200%, -165%);
                    background-color: red' type='submit'>-</button>
                    <input type='hidden' value='<?php echo $saleItemName; ?>'
name='saleItem'>
                    </form>
                    <?php
                }

                /* Calculate total cost of transaction */
                $query = "SELECT Quantity, Cost FROM $tableName";
                $result = mysqli_query($connection, $query);

                for ($i = 0; $i < mysqli_num_rows($result); $i++) {
                    $row = mysqli_fetch_assoc($result);
                    $total += $row['Cost'] * $row['Quantity'];

                }
            }
            /* Output total cost of transaction */
            ?>
            <h3 align='center'>Total: <?php echo $total; ?></h3>
            <?php
            $_SESSION['total'] = $total;
```

```
            /*Complete Transaction Button */
            $_SESSION['transactionID'] = $transactionID;
            ?>
            <h3 align='center'><form action='process_completeTransaction.php'>
                  <button class='completeTransaction' type='submit'>Close
Transaction</button>
                  <input type='hidden' name='transactionID' value='<?php echo
$transactionID; ?>'>
                  </form></h3>

            <?php
            function checkForLowStock($connection,$companyName){
                $tableName = "stockmanagementtable_".$companyName;
                $query = "SELECT
ProductID,ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone
                    FROM $tableName WHERE CurrentStockValue < MinimumStockValue AND
Ordered='off'";
                $result = mysqli_query($connection, $query);

                /* Calculate the umber of rows returned by the query */
                $numRows = 0;
                if($result == false){
                    $numRows = 0;
                }
                else{
                    $numRows = mysqli_num_rows($result);
                }

                if($numRows > 0) {
                    $_SESSION['returnedRows'] = mysqli_fetch_assoc($result);
                    $_SESSION['companyName'] = $companyName;
                    $_SESSION['tableName'] = $tableName;

                    header('Location: lowStock.php');// redirect user
                    exit;

                }
            }
            ?>
    </span>

    <!-- Sale Item Column -->
    <span class = 'midGUI' >
        <?php
        $menuName = $_SESSION['menuName'];

        $query = "SELECT MenuID FROM menus WHERE MenuName = '$menuName' AND CompanyName
= '$companyName'";
        $result = mysqli_query($connection, $query);

        $row = mysqli_fetch_assoc($result);
        $menuID = $row['MenuID'];

        $tableName = "menu_".$companyName."_".$menuID;
        $_SESSION['companyMenuTableName'] = $tableName;

        /* Checks if a category has been selected that is not all */
        if($_SESSION['shouldSearch'] == true){
            // Retrieve query from process_search.php
            // only shows research for entered search
            $query = $_SESSION['menuQuery'];
            $_SESSION ['shouldSearch'] = false; // set POS system not to search on next
reload
                                                // unless search is re-submitted
        }
        else if(!empty($_GET) && $_GET['category'] != "all"){
            $category = $_GET['category'];
```

```
            // Only show results for given category
            $query = "SELECT saleItemName,ID,price FROM $tableName WHERE Category =
'$category'";
        }
        else{
            // show results for all categories
            $query = "SELECT saleItemName,ID,price FROM $tableName";
        }
        /* Run query selecting which sale items to display */
        $result = mysqli_query($connection, $query);
        $numRows = mysqli_num_rows($result);

        for($i=0;$i<$numRows; $i++){

            $row = mysqli_fetch_assoc($result);
            $saleItemName = $row['saleItemName'];
            $saleItemID = $row['ID'];
            $displayPrice = "£".$row['price'];
            $price = $row['price'];

            echo "<form action = 'process_buttonClick_saleItemAdd.php'>
                    <input type='hidden' name='transactionID' value='$transactionID'>
                    <input type='hidden' name='saleItemName' value='$saleItemName'> <!-
- type= hidden means this is not visable but can send necissary data to processing--->
                    <input type='hidden' name='saleItemID' value='$saleItemID'>
                    <input type='hidden' name='saleItemCost' value='$price'>
                    <button class= 'saleItem' type='submit'>$saleItemName <br>
$displayPrice</button>

                </form>";
        }

        echo "<form action='editSaleItem.php'>
              <button class= 'newSaleItem' type='submit'>+New <br> Sale
Item</button></form>";
        ?>
    </span>

    <!-- Category Column -->
    <span class='rightGUI'>
        <style>
            .category{
                background-color: #0404cb;
                border: none;
                width: 100%;
                height: 8%;
                color: white;
                font-weight: bold;
                font-size: 18px;
            }

            .category:hover button{
                background-color: blue;
                height: 50%;
            }



        </style>
        <?php



        /* Create 9 category buttons */

        // Display default tab
        echo "
```

```
            <form class='category' action='pointOfSale.php'>
            <input type='hidden' value='all' name='category'>
            <button class='category' type='submit'>All</button>
            </form>";


        /* Search for all categories */
        $tableName = "menu_".$companyName."_".$menuID;
        $query = "SELECT DISTINCT Category FROM $tableName"; // Select only unique
values
        $categoryResult = mysqli_query($connection,$query);

        /* Calculate number of rows returned by query */
        $numRows = 0;
        if($categoryResult == false){
            $numRows = 0;
        }
        else{
            $numRows = mysqli_num_rows($categoryResult);
        }

        /* Display category tabs */
        for($i = 0; $i<$numRows; $i++) {

            $categoryRow = mysqli_fetch_assoc($categoryResult); // fetch the next row
from results

            $tabName = $categoryRow['Category'];
            echo "
                <form class ='category' action='pointOfSale.php'>
                <input type='hidden' value='$tabName' name='category'>
                <button class='category' type='submit'>$tabName</button>
                </form>
            ";
        }
        ?>

    </span>

</div>


</body>
</html>
```

Process_buttonclick_saleItemAdd.php

```php
<?php
/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

session_start();
$transactionID = $_SESSION['transactionID'];
$companyName = $_SESSION['companyName'];

$saleItemName = $_GET['saleItemName'];
$cost = $_GET['saleItemCost'];

$tableName = "transaction_".$transactionID."_".$companyName;

$query = "SELECT Quantity FROM $tableName WHERE SaleItemName = '$saleItemName'";
$result = mysqli_query($connection,$query);

echo $query;
// This fixed an unusual bug with mysqli_query sometimes returning 0 rows but not being
false
$numRows = 0;
if($result != false){
    $numRows = mysqli_num_rows($result);
}

// Add sale item into current transaction
if ($numRows == 0){
    // Inserts a new row into the current transaction table if the sale item does not
already exist in the
    // current transaction
    $query = "INSERT INTO "."$tableName"."(SaleItemName,Cost,Quantity)
VALUES('$saleItemName','$cost','1')";
    mysqli_query($connection,$query);

}
else{
    // If the sale item exists in the current transaction,
    // update the quantity of that sale item in the transaction
    $row = mysqli_fetch_assoc($result);
    $quantity = $row['Quantity'];

    $newQuantity = $quantity + 1;
    $query = "UPDATE $tableName
              SET Quantity = '$newQuantity'
              WHERE SaleItemName = '$saleItemName' ";
    mysqli_query($connection,$query);

}



header('Location: pointOfSale.php');// redirect to home page
exit;
?>
```

Process_CompleteTransaction.php

```php
<?php
session_start();

/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

// Initiate variables
$transactionID = $_GET["transactionID"];
$companyName = $_SESSION["companyName"];

/*
 * This function returns false if an item is out of stock,
 * and true if all saleItems are in stock
 * $DBconnection stores the connection to the SQL database
 * $companyName is the name of the company of the current user
 * $decuctFromStock is a boolean value to show whether or not you want stock levels to
be decreased, where:
 * true = you would like the stock level to be decreased
 * false = you would not like the stock level to be decreased
 */
function updateStockManagementSystem($DBconnection,$companyName)
{
    // initiate variables
    $tableName = "transaction_" . $_GET["transactionID"] . "_" .
$_SESSION['companyName'];
    $connection = $DBconnection;
    $transactionID = $_GET["transactionID"];
    $returnValue = true;

    // Select everything contained within the transaction table
    $query = "SELECT * FROM $tableName";
    $result = mysqli_query($connection, $query);

    if ($result != false) {
        $numRows = mysqli_num_rows($result); //returns the number of rows in the table
    }
    else{
        $numRows = 0;
    }

    // Loop through sale items
    for ($i = 0; $i < $numRows; $i++) {
        // fetch a new row from the SQL result
        $row = mysqli_fetch_assoc($result);

        // Get the table name for the current menu
        $menuID = $_SESSION['menuID'];
        $name = "menu_".$companyName."_".$menuID;
        $currentSaleItemName = $row['SaleItemName'];

        // Get the sale item id of the current product from the current menu table
        $query = "SELECT ID FROM $name WHERE saleItemName = '$currentSaleItemName'";
        $queryResult = mysqli_query($connection,$query);
        $rowID = mysqli_fetch_assoc($queryResult);

        // Store the table name for the contents of the current sale item
        $saleItemContentsTableName =
"SaleItem_Contents_".$companyName."_".$rowID['ID'];

        // Select all rows for the contents of the current sale item
        $query = "SELECT * FROM $saleItemContentsTableName";
        $saleItemContentsResult = mysqli_query($connection, $query);

        // If result is false, number of rows = 0
        if($saleItemContentsResult == false){
```

```
            $numberOfRows = 0;
        }
        else{
            $numberOfRows = mysqli_num_rows($saleItemContentsResult);
        }

        // Loop through each product of a sale item
        for ($k = 0; $k < $numberOfRows; $k++) {
            $saleItemContentsRow = mysqli_fetch_assoc($saleItemContentsResult);

            // Select product name and current stock value from the stockmanagement
table for the current product.
            // Stores result in $productQueryResult
            $stockManagementTableName = "stockmanagementtable_$companyName";
            $productName = $saleItemContentsRow['Product'];
            $query = "SELECT ProductName,CurrentStockValue FROM
$stockManagementTableName
                      WHERE ProductName = '$productName'";
            $productQueryResult = mysqli_query($connection, $query);
            $productRow = mysqli_fetch_assoc($productQueryResult);

            // Produce an error if more than one result is returned
            // Produce an error if no results are returned
            if (mysqli_num_rows($productQueryResult) == 1) {

                // Checks if the stock management system has a great enough stock
quantity of the product
                if ($productRow['CurrentStockValue'] -
($saleItemContentsRow['Quantity'] * $row['Quantity']) < 0) {

                    $returnValue = false;
                }

                // Calculate new stock value of product
                $productQuantity = $saleItemContentsRow['Quantity'];

                $query = "SELECT CurrentStockValue FROM $stockManagementTableName WHERE
ProductName = '$productName'";
                $currentStockValueQueryResult = mysqli_query($connection,$query); //
Run SQL query

                $currentStockValueQueryRow =
mysqli_fetch_assoc($currentStockValueQueryResult);
                $newStockQuantity = $currentStockValueQueryRow['CurrentStockValue'] -
($productQuantity * $row['Quantity']);

                // Deducts from the stock management system
                $query = "UPDATE $stockManagementTableName
                          SET CurrentStockValue = '$newStockQuantity'
                          WHERE ProductName = '$productName'";
                mysqli_query($connection, $query);

            } else if (mysqli_num_rows($productQueryResult) == 0) {
                echo "ERROR! A product was missing from your stock management
database";
            } else {
                echo "ERROR! Duplicated product names! Remove duplicates from your
stock management database";
            }

        }

    }

    /*
     *  If a product is not in stock, all products quantities should be added back to
the stock management
```

```
     *  database, as the transaction will not be completed.
     *
     * This section could be severely simplified, probably by using a while loop and a
selection statement
     * in the previous section
     */
    if ($returnValue == false) {

        // Select everything contained within the transaction table
        $query = "SELECT * FROM $tableName";
        $result = mysqli_query($connection, $query);

        // Loop through sale items
        for ($i = 0; $i < $numRows; $i++) {

            // fetch a new row from the SQL result
            $row = mysqli_fetch_assoc($result);

            // Store the table name for the contents of the current sale item
            $saleItemContentsTableName = "SaleItem_Contents_" . $companyName . "_" .
$row['SaleItemID'];

            // Select all rows for the contents of the current sale item
            $query = "SELECT * FROM $saleItemContentsTableName";
            $saleItemContentsResult = mysqli_query($connection, $query);

            // Loop through each product of a sale item
            for ($k = 0; $k < mysqli_num_rows($saleItemContentsResult); $k++) {
                $saleItemContentsRow = mysqli_fetch_assoc($saleItemContentsResult);

                // Select product name and current stock value from the stockmanagement
table for the current product.
                // Stores result in $productQueryResult
                $stockManagementTableName = "stockmanagementtable_$companyName";
                $productName = $saleItemContentsRow['Product'];
                $query = "SELECT ProductName,CurrentStockValue FROM
$stockManagementTableName
                          WHERE ProductName = '$productName'";
                $productQueryResult = mysqli_query($connection, $query);

                // Produce an error if more than one result is returned
                // Produce an error if no results are returned
                if ($productQueryResult == false) {
                    echo "ERROR! A product was missing from your stock management
database";
                }
                else if (mysqli_num_rows($productQueryResult) == 1) {

                    // Calculate new stock value of product
                    $productQuantity = $saleItemContentsRow['Quantity'];

                    $query = "SELECT CurrentStockValue FROM $stockManagementTableName
WHERE ProductName = '$productName'";
                    $currentStockValueQueryResult = mysqli_query($connection,$query);
// Run SQL query

                    $currentStockValueQueryRow =
mysqli_fetch_assoc($currentStockValueQueryResult);
                    $newStockQuantity = $currentStockValueQueryRow['CurrentStockValue']
+ ($productQuantity * $row['Quantity']);

                    // Increases quantities in the stock management system
                    $query = "UPDATE $stockManagementTableName
                              SET CurrentStockValue = '$newStockQuantity'
                              WHERE ProductName = '$productName'";
                    mysqli_query($connection, $query);
```

```
                }
                else {
                    echo "ERROR! Duplicated product names! Remove duplicates from your
stock management database";
                }

            }

        }
    }

    /*
     *  End the current transaction
     */
    if ($returnValue == true) {
        /* Set current transaction to complete */
        $transactionTotal = $_SESSION['total'];
        $tableName = "transactionhistory_" . $companyName;
        $query = "UPDATE $tableName SET Complete = 'yes', Total = '$transactionTotal'
WHERE TransactionID = '$transactionID'";
        mysqli_query($connection, $query);

        /* Open a new transaction */
        $query = "INSERT INTO transactionhistory_$companyName(Complete) VALUES('no')";
        mysqli_query($connection, $query);

        // Create a new transaction table
        $newTransactionID = $transactionID + 1;
        $newTableName = "transaction_".$newTransactionID."_".$companyName;
        $query = "CREATE TABLE `users`.`$newTableName` ( `SaleItemID` INT(9) NOT NULL
AUTO_INCREMENT ,
                `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT
NOT NULL ,
                 PRIMARY KEY (`SaleItemID`))";
        mysqli_query($connection,$query);
    }

}

/*
 * MAIN
*/

/* Select everything from the current transaction table */
$tableName = "transaction_" . $_GET["transactionID"] . "_" . $_SESSION['companyName'];
$query = "SELECT * FROM $tableName";
$result = mysqli_query($connection,$query);

/*
 *   Only attempt to update the stock management system
 *   if are sale items contained within the current transaction
*/
if($result != false) {
    if(mysqli_num_rows($result) != 0) {
        updateStockManagementSystem($connection, $companyName);
    }
    else{
        echo "Error! Transaction is empty";
    }
}
else{
    echo "Error! Transaction is empty";
}


header('Refresh: 2; URL=pointOfSale.php');// redirect user
exit;
```

Process_DayEnd.php

```php
<?php
/* Open DB connection */
include "database_connect.php";
$connection = openConnection();

/* Start the session */
session_start();

/* Select all from company transaction history */
$companyName = $_SESSION['companyName'];
$query = "SELECT * FROM transactionhistory_$companyName";
$queryResult = mysqli_query($connection, $query);

/* Calculate number of rows returned by query */
if($queryResult == false){ // If no rows are returned by function
    $numRows = 0;
}
else{
    $numRows = mysqli_num_rows($queryResult); // Returns number of rows
}

/* Delete all transaction tables */
for ($i=0; $i<$numRows; $i++) {
    $rowResult = mysqli_fetch_assoc($queryResult);
    $transactionNumber = $rowResult['TransactionID']; // Get transaction ID

    /* Delete transaction tables for this company */
    $tableName = "transaction_" . $transactionNumber . "_" . $companyName; // Calculate
name of table
    $query = "DROP TABLE `$tableName`"; // Delete the table from DB Query
    mysqli_query($connection, $query);

    /* Delete corresponding row in transaction history table */
    $tableName = "transactionhistory_" . $companyName;
    $query = "DELETE FROM $tableName WHERE TransactionID = '$transactionNumber'";
    mysqli_query($connection, $query);
}


/* Create a new transaction in the transaction history table,
 * and create a new transaction table for the transaction.
 */

/* Insert new transaction table name into transaction history table */
$tableName = "transactionhistory_".$companyName;
$query = "INSERT INTO $tableName(Complete) VALUES('no')";
mysqli_query($connection,$query);

/* Retrieve transaction ID */
$query = "SELECT TransactionID FROM $tableName WHERE Complete = 'no'";
$queryResult = mysqli_query($connection,$query);
$transactionHistoryRow = mysqli_fetch_assoc($queryResult);
$transactionID = $transactionHistoryRow['TransactionID'];

/* Create Initial Transaction Table */
$tableName = "transaction_".$transactionID."_".$companyName;
$query = "CREATE TABLE `users`.`$tableName` ( `SaleItemID` INT(9) NOT NULL
AUTO_INCREMENT ,
                `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT
NOT NULL ,
                 PRIMARY KEY (`SaleItemID`))";
mysqli_query($connection,$query);

?>
```

Process_EditProduct.php

```
<?php
session_start();

/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

$productID = $_SESSION['productID'];
$tableName = $_SESSION['tableName'];
$productName = $_GET['ProductName'];
$minimumStockValue = $_GET['MinimumStockValue'];
$currentStockValue = $_GET['CurrentStockValue'];
$ordered = $_GET['Ordered'];
$supplierName = $_GET['SupplierName'];
$phoneNumber = $_GET['PhoneNumber'];
$arrivalDate = $_GET['dueDate'];
$arrivalAmount = $_GET['restockQuantity'];

/* Validate date & time */
date_default_timezone_set("GMT");
$currentDateLong = date("c");
$currentDateShort = substr($currentDateLong,0,16);

/* Check that the entered date & time is greater than today's date & time*/
if(($arrivalDate > $currentDateShort OR $arrivalDate == null) AND ($arrivalAmount > 0
OR $arrivalDate == null)) {
    $query = "UPDATE $tableName
            SET
ProductName='$productName',MinimumStockValue='$minimumStockValue',CurrentStockValue='$c
urrentStockValue',
            Ordered='$ordered',SupplierName='$supplierName',Phone='$phoneNumber',
ArrivalDate='$arrivalDate',
            ArrivalAmount='$arrivalAmount' WHERE productID='$productID'";
    mysqli_query($connection, $query);
    header('refresh: 0; URL = stockManagement.php');// redirect to home page
    exit;
}
else{
    echo "ERROR! Invalid date or restock quantity";
    header('refresh: 2; URL = stockManagement.php');// redirect to home page
    exit;
}

?>
```

Process_editSaleItem.php

```php
<?php
/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

/* Start the session */
session_start();

/* Init Variables */
$saleItemToEdit = $_GET['saleItemToEdit'];
$saleItemName = $_GET['saleItemName'];
$itemPrice = $_GET['saleItemPrice'];
$category = $_GET['saleItemCategory'];
$menuTableName = $_SESSION['companyMenuTableName'];
$companyName = $_SESSION['companyName'];
$productString = $_GET['products'];

function productStringToArray($productString){

    $productArray = []; //An array that stores the name of each product
    $tempString = ""; //A temporary string to store a product name when being initially formed,
                      //to later be inserted into the $productArray

    for ($i=0; $i<strlen($productString); $i++){

        /* Search for a comma */
        if($productString[$i] == ","){

            /* If there is a space before the next comma, remove it from the tempString */
            while($tempString[strlen($tempString)-1] == " "){
                $tempString = substr($tempString,0,strlen($tempString)-1);
            }

            $productArray[] = $tempString; //Append product into array
            $tempString = ""; //Empty string
        }
        /* Do Nothing if a space is found after a comma */
        else if($productString[$i] == " " and $productString[$i - 1] == ","){
            //do nothing - BAD PRACTICE
        }
        else{

            $tempString = $tempString.$productString[$i]; //Append character to $productString
        }
    }
    $productArray[] = $tempString; //Append final product into array

    return $productArray;
}

/* Checks if a floating point number is to 2 decimal places */
/* Returns true if number is to 2 decimal places */
/* Returns false if number is not to 2 decimal places */
function isTwoDecimalPlaces($value){

    /* Calculate the length of $value */
    $value = (string) $value; //cast $value to string
    $lengthValue = strlen($value);

    /* loop through $stringValue until the decimal point is found */
    /* Store the number of decimal places into variable $numberOfPlacesAfterDecimal */
    $decimalPointFound = false;
    $numberOfPlacesAfterDecimal = 0;
```

```
    for($i=0; $i<$lengthValue; $i++){
        if ($value[$i] == "."){
            $decimalPointFound = true;
        }
        else if($decimalPointFound == true){
            $numberOfPlacesAfterDecimal += 1;
        }
    }

    /* Return false if number of decimals is not 2 */
    if($numberOfPlacesAfterDecimal != 2){
        return false;
    }
    else{
        return true;
    }
}

if($saleItemToEdit == '+new' AND isTwoDecimalPlaces($itemPrice) == true){

    /* SQL query creating new entry into the menu */
    $query = "INSERT INTO $menuTableName(saleItemName,price,category)
              VALUES('$saleItemName','$itemPrice','$category')";
    mysqli_query($connection,$query);

    /* SQL query selecting the ID of the newly created sale item
       from the company's menu table                              */
    $query = "SELECT ID FROM $menuTableName
              WHERE saleItemName = '$saleItemName' ";
    $result = mysqli_query($connection,$query);

    $row = mysqli_fetch_assoc($result);
    $id = $row['ID'];
    echo "ID:".$id;

    $tablename = "SaleItem_Contents_".$companyName."_".$id;

    $query = "CREATE TABLE `users`.`$tablename` ( `ID` INT(9) NOT NULL AUTO_INCREMENT ,
`Product` TEXT NOT NULL ,
            `Quantity` INT NOT NULL , PRIMARY KEY (`ID`))";
    mysqli_query($connection,$query);

    $productArray = productStringToArray($productString);

    /* Inserting products into SaleItem_Contents table for the particular sale item */
    $sizeOfProductArray = count($productArray);

    for($i=0; $i<$sizeOfProductArray; $i++){

        $query = "SELECT Product FROM $tablename WHERE Product = '$productArray[$i]'";
        $result = mysqli_query($connection,$query);
        $numResults = mysqli_num_rows($result); //Fetch the number of results the query
produced

        /* Insert product into SaleItem_Contents table if product does not already
exist in the table */
        if($numResults == 0) {
            $query = "INSERT INTO $tablename(Product,Quantity)
VALUES('$productArray[$i]', '1')";
            mysqli_query($connection,$query);

        }
        /* Otherwise, update the quantity of such item */
        else{
            $query = "UPDATE $tablename SET(Quantity = 'Quantity + 1') WHERE Product =
$productArray[$i]";
            mysqli_query($connection,$query);
```

```
        }
    }


}
else if(isTwoDecimalPlaces($itemPrice) == false){
    echo "Incorrect number of decimal places";
}
else{
    //SQL query updating current entry in the menu
    $query = "UPDATE $menuTableName SET
              saleItemName = $saleItemName,
              itemPrice = $itemPrice,
              category = $category
              WHERE saleItemName = $saleItemToEdit";
    mysqli_query($connection,$query);
}

?>
```

Process_existingmenu.php

```php
<?php
session_start();
include "database_connect.php";
$connect = openConnection();

$menuName = $_GET["existingMenuName"];
$companyName = $_SESSION['companyName'];

$_SESSION['menuName'] = $menuName;

$query = "SELECT MenuName,MenuID FROM menus WHERE CompanyName = '$companyName' AND
menuName = '$menuName'";
$result = mysqli_query($connect,$query);

$row = mysqli_fetch_assoc($result);
$_SESSION['menuID'] = $row['MenuID'];

if (mysqli_num_rows($result) !=0){

    header('Location: pointOfSale.php');// redirect to point-of-sale system
    exit;
}
else{
    header('Location: MyMenus.php');// redirect back to MyMenus page
    //display an alert
    exit;
}

?>
```

Process_login.php

```php
<?php
session_start();

/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

/* Get username and password from HTML form */
$username = $_GET['username'];
$password = $_GET['password'];

/* Run SQLi query to see if there is a match in the database to username and password
   If a match is found, a new object of type User is created */
$query = "SELECT Username, Password FROM Credentials WHERE Username = '$username' AND
Password = '$password'";
$result = mysqli_query($connection, $query);
$numResults = mysqli_num_rows($result);

if($numResults != 0){
    /* Get the values of companyName, email, and phoneNumber. If details are note
provided, false is placed into the variables */
    $query = "SELECT Company FROM Credentials WHERE Username = '$username'";
    $result = mysqli_query($connection,$query);
    $row = $result->fetch_row();
    $companyName = $row[0] ?? false;

    /* Store the current user's email in a session variable */
    $query = "SELECT Email FROM Credentials WHERE Username = '$username'";
    $result = mysqli_query($connection,$query);
    $email = $result->fetch_row()[0] ?? false;
    $_SESSION['email'] = $email;

    /* Store the current user's phone number in a session variable */
    $query = "SELECT Phone FROM Credentials WHERE Username = '$username'";
    $result = mysqli_query($connection,$query);
    $phoneNumber = $result->fetch_row()[0] ?? false;
    $_SESSION['phoneNumber'] = $phoneNumber;

    /* Store the current user's account type in a session variable */
    $query = "SELECT accountType FROM Credentials WHERE Username = '$username'";
    $result = mysqli_query($connection,$query);
    $_SESSION['accountType'] = $result->fetch_row()[0];

    /* Set session variables */
    $_SESSION['username'] = $username;
    $_SESSION['password'] = $password;
    $_SESSION['companyName'] = $companyName;
    $_SESSION['shouldSearch'] = false; // set POS search bar as not actively searching
    $_SESSION['transactionID'] = "";

    echo "Account accessed";
}
else{
    echo "this is not your account!";// temp
}

closeConnection($connection);

header('Refresh: 3; URL=HomePage.php');
exit;

?>
```

Process_lowstock.php

```php
<?php
session_start();

include "database_connect.php";
$connection = openConnection();

// Calculate table name
$companyName = $_SESSION['companyName'];
$tableName = "stockmanagementtable_".$companyName;

/* Fetch dateTime and orderAmount from lowStock.php */
$dateTime = $_GET['dateTime'];
$orderAmount = $_GET['orderAmount'];

/* Validate date & time */
date_default_timezone_set("GMT");
$currentDateLong = date("c");
$currentDateShort = substr($currentDateLong,0,16);

/* Check that the entered date & time is greater than today's date & time*/
if($dateTime > $currentDateShort AND $orderAmount > 0){
    /* Update the SM table */
    $productID = $_SESSION['ProductID'];
    $query = "UPDATE $tableName SET Ordered = 'on', ArrivalDate = '$dateTime',
ArrivalAmount = '$orderAmount'
              WHERE ProductID = '$productID'";
    mysqli_query($connection,$query);
}
else{
    echo "error! Invalid input"; // Output error
}

header('refresh: 0; URL = pointOfSale.php');// redirect to POS page
exit;
?>
```

Process_newMenu.php

```php
<?php
session_start();

include "database_connect.php";
$connection = openConnection();

// Calculate table name
$companyName = $_SESSION['companyName'];
$tableName = "stockmanagementtable_".$companyName;

/* Fetch dateTime and orderAmount from lowStock.php */
$dateTime = $_GET['dateTime'];
$orderAmount = $_GET['orderAmount'];

/* Validate date & time */
date_default_timezone_set("GMT");
$currentDateLong = date("c");
$currentDateShort = substr($currentDateLong,0,16);

/* Check that the entered date & time is greater than today's date & time*/
if($dateTime > $currentDateShort AND $orderAmount > 0){
    /* Update the SM table */
    $productID = $_SESSION['ProductID'];
    $query = "UPDATE $tableName SET Ordered = 'on', ArrivalDate = '$dateTime',
ArrivalAmount = '$orderAmount'
              WHERE ProductID = '$productID'";
    mysqli_query($connection,$query);
}
else{
    echo "error! Invalid input"; // Output error
}

header('refresh: 0; URL = pointOfSale.php');// redirect to POS page
exit;
?>
```

Process_newProduct.php

```
<?php
session_start();

(string)$companyName = $_SESSION['companyName'];
$_SESSION['TableName'] = "stockmanagementtable_".$companyName;
$tableName = $_SESSION['TableName'];
(string)$productName = $_GET['ProductName'];
(int)$minimumStockValue = $_GET['MinimumStockValue'];
(int)$currentStockValue = $_GET['CurrentStockValue'];
(string)$ordered = $_GET['Ordered'];
(string)$supplierName = $_GET['SupplierName'];
(int)$phoneNumber = $_GET['PhoneNumber'];

include "database_connect.php";
$connection = openConnection();

/* See if product name already exists */
/* Run SQL query searching for all products and compare product names to entered
product name */
function isProductInTable($connection,$tableName,$productName){
    $query = "SELECT ProductName FROM $tableName";
    $result = mysqli_query($connection, $query);

    for ($i = 0; $i < mysqli_num_rows($result); $i++) {
        $row = mysqli_fetch_assoc($result);
        if ($row['ProductName'] == $productName) {
            return true;
        }
    }
    return false;
}

/* Checks if a value is greater than  or equal to zero. */
/* Returns true if value is greater than or equal to zero */
/* Returns false if value is less then 0 */
function isGreaterThanOrEqualToZero($value){
    if($value >= 0){
        return true;
    }
    else {
        return false;
    }
}

/* Removes whitespace before first character,
   and after the last character of a string. */
/* Returns the resulting string */
function removeWhitespaceBeforeAndAfterString($string){

    while($string[0] == " "){
        $string = substr($string, 1);
    }
    while ($string[strlen($string)-1] == " "){
        $string = substr($string,0, strlen($string) - 1);
    }

    return $string;
}


function doesContainComma($string){

    $commaFound = false;
    for($i=0; $i<strlen($string); $i++){
        if($string[$i] == ","){
            $commaFound = true;
```

```
        }
    }

    return $commaFound;
}

/* If the product is not in the table, insert the product into the stock management
database */
if (isProductInTable($connection, $tableName, $productName) == false) {

    if(doesContainComma($productName) == false) {
        $productName = removeWhiteSpaceBeforeAndAfterString($productName);

        if (isGreaterThanOrEqualToZero($minimumStockValue) and
isGreaterThanOrEqualToZero($currentStockValue)) {
            $query = "INSERT INTO
$tableName(ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone)
          VALUES('$productName', '$minimumStockValue', '$currentStockValue',
'$ordered',
          '$supplierName', '$phoneNumber')";
            mysqli_query($connection, $query);
            echo "Product Successfully Added";
        } else {
            echo "Value Smaller Than Zero Entered, INVALID!";
        }
    }
    else{
        echo "Invalid character: Comma ',' in product name";
    }
}
else{
    echo "Product already exists!";
}

header('Refresh: 2; URL=StockManagement.php');

?>
```

Process_refund.php

```php
<?php
/* Open DB connection */
include "database_connect.php";
$connection = openConnection();

/* Start the session */
session_start();

/* Retrieve amount to refund */
$refundAmount = $_GET['refundAmount'];

/* Retrieve companyName */
$companyName = $_SESSION['companyName'];

/* Add refund as a negatively valued transaction to the transaction history table */
$refundAmount = -$refundAmount; // Set refundAmount to be a negative value
$tableName = "transactionhistory_".$companyName;
$query = "INSERT INTO $tableName(Complete,Total) VALUES('yes','$refundAmount')";
mysqli_query($connection,$query);
```

Process_saleItemDecrement.php

```php
<?php
/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

session_start();
$transactionID = $_SESSION['transactionID'];
$companyName = $_SESSION['companyName'];
$saleItemName = $_GET['saleItem'];

$tableName = "transaction_".$transactionID."_".$companyName;
$query = "SELECT Quantity FROM $tableName WHERE SaleItemName = '$saleItemName'";
$result = mysqli_query($connection,$query);

// update the quantity of that sale item in the transaction
$row = mysqli_fetch_assoc($result);
$quantity = $row['Quantity'];
$newQuantity = $quantity - 1;

if($newQuantity >0) {
    $query = "UPDATE $tableName
              SET Quantity = '$newQuantity'
              WHERE SaleItemName = '$saleItemName' ";
    mysqli_query($connection, $query);
}
else{
    $query = "DELETE FROM $tableName WHERE SaleItemName = '$saleItemName'";
    mysqli_query($connection,$query);
}

// redirect user
header('Refresh: 0; URL=pointOfSale.php');
exit;
```

Process_saleItemIncrement.php

```php
<?php
/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

session_start();
$transactionID = $_SESSION['transactionID'];
$companyName = $_SESSION['companyName'];
$saleItemName = $_GET['saleItem'];

$tableName = "transaction_".$transactionID."_".$companyName;
$query = "SELECT Quantity FROM $tableName WHERE SaleItemName = '$saleItemName'";
$result = mysqli_query($connection,$query);

// update the quantity of that sale item in the transaction
$row = mysqli_fetch_assoc($result);
$quantity = $row['Quantity'];
$newQuantity = $quantity + 1;

$query = "UPDATE $tableName
              SET Quantity = '$newQuantity'
              WHERE SaleItemName = '$saleItemName' ";
mysqli_query($connection,$query);

// redirect user
header('Refresh: 0; URL=pointOfSale.php');
exit;
```

Process_Search.php

```php
<?php
session_start();

/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

/* retrieve variables from session */
$companyName = $_SESSION['companyName'];
$menuID = $_SESSION['menuID'];

/* Get the search value */
$searchValue = $_GET['searchBar'];

/* Search for query in current menu table */
$tableName = "menu_".$companyName."_".$menuID;
$_SESSION['menuQuery'] = "SELECT * FROM $tableName WHERE saleItemName LIKE
'%$searchValue%'";

/* Set search to be true when the POS system is next opened */
$_SESSION['shouldSearch'] = true;

header('Refresh: 0; URL=pointOfSale.php');
exit;
```

Process_signup.php

```
<?php
/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

/* VALIDATE EMAIL */
/* If valid returns true */
/* If invalid returns false */
function validateEmail($email){
    $valid = false;

    /* search through every character in an email until @ is found */
    $count = 0;
    while ($count < strlen($email) AND $email[$count] != "@"){
        $count += 1;
    }
    if($email[$count] == "@") {
        $secondHalfOfEmail = substr($email, $count + 1, strlen($email) - $count);

        /* Search for dot (.) in second half of email */
        $count = 0;

        while ($count < strlen($secondHalfOfEmail) AND $secondHalfOfEmail[$count] !=
"."){
            $count += 1;
        }

        /* Check that count is not greater than the last index of the array of the
string $secondHalfOfEmail */
        if($count != strlen($secondHalfOfEmail)) {

            if ($secondHalfOfEmail[$count] == ".") {
                $emailAfterDot = substr($secondHalfOfEmail, $count + 1,
strlen($secondHalfOfEmail) - $count);

                /* Search for alphabetic character at the end of the email */
                $alphabetArray = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j",
"k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"];

                $i = 0;
                while ($i < 26 and $valid == false) {

                    $alphabeticCharacter = $alphabetArray[$i];

                    /* Search for lowercase version */
                    if ($emailAfterDot[strlen($emailAfterDot) - 1] ==
$alphabeticCharacter) {
                        $valid = true;
                    }
                    /* Search for uppercase version */
                    if ($emailAfterDot[strlen($emailAfterDot) - 1] ==
strtoupper($alphabeticCharacter)) {
                        $valid = true;
                    }

                    $i += 1;

                }
            }
        }
    }


    return $valid;
}
```

```
/* retrieve entered details from form */
$username = $_GET['username'];
$password = $_GET['password'];
$email = $_GET['email'];
$companyName = $_GET['companyName'];
$phoneNumber = $_GET['phone'];

/* Check if username already exists */
$query = "SELECT Username FROM Credentials WHERE Username = '$username'";
$result = mysqli_query($connection, $query);
$numResults = mysqli_num_rows($result);

/* submit details to Credentials table if username is not in use and email is valid*/
if ($numResults == 0 && validateEmail($email) == true) {
    echo "Username is valid\n";// temp

    if ($phoneNumber == null and $email != null) {
        mysqli_query($connection, "INSERT INTO Credentials (Username, Password,
Company, Email, accountType)
            VALUES ('$username', '$password', '$companyName', '$email','unknown')");
    } else if ($phoneNumber != null and $email == null) {
        mysqli_query($connection, "INSERT INTO Credentials (Username, Password,
Company, Phone, accountType)
            VALUES ('$username', '$password', '$companyName',
'$phoneNumber','unknown')");
    } else if ($phoneNumber == null and $email == null) {
        mysqli_query($connection, "INSERT INTO Credentials (Username, Password,
Company, accountType)
            VALUES ('$username', '$password', '$companyName','unknown')");
    } else {
        mysqli_query($connection, "INSERT INTO Credentials (Username, Password,
Company, Email, Phone, accountType)
            VALUES ('$username', '$password', '$companyName', '$email',
'$phoneNumber','unknown')");
    }

    /* Check if stock management table already exists for that company, if not create
one. */
    $result = mysqli_query($connection, "SELECT Company FROM Credentials WHERE Company
= '$companyName'");
    $numResults = mysqli_num_rows($result);

    /* This is the first time the company has been registered to the system */
    if($numResults == 1){
        /* Calculate Table Name For This Company */
        $tableName = "stockmanagementTable_".$companyName;

        /* Create A Stock Management Table For This Company */
        mysqli_query($connection,
            "CREATE TABLE `users`.`$tableName` ( `ProductID` INT NOT NULL
AUTO_INCREMENT ,
                    `ProductName` TEXT NOT NULL , `MinimumStockValue` INT NOT NULL ,
`CurrentStockValue` INT NOT NULL,
                    `Ordered` TEXT NOT NULL DEFAULT 'off' ,`SupplierName` TEXT NULL
DEFAULT NULL , `Phone` INT(11) NULL,
                    `ArrivalDate` DATETIME DEFAULT NULL, `ArrivalAmount` INT DEFAULT
NULL, PRIMARY KEY (`ProductID`)); ");

        /* Insert the Table Name and Company Name into the Stock Management Hub table
*/
        $query = "INSERT INTO stockmanagementhub(TableName,CompanyName)
                  VALUES('$tableName', '$companyName')";
        mysqli_query($connection,$query);

        /* Create A Transaction History Table For This Company */
        $query = "CREATE TABLE `users`.`transactionhistory_$companyName` (
                  `TransactionID` INT(9) NOT NULL AUTO_INCREMENT , `Complete` TEXT NOT
```

```
NULL DEFAULT 'no' ,
                    `Total` DECIMAL(10,2) NOT NULL, PRIMARY KEY (`TransactionID`))";
        mysqli_query($connection,$query);

        /* Create Initial Transaction Table */
        $query = "CREATE TABLE `users`.`transaction_1_$companyName` ( `SaleItemID`
INT(9) NOT NULL AUTO_INCREMENT ,
                `SaleItemName` TEXT NOT NULL , `Cost` DOUBLE NOT NULL , `Quantity` INT
NOT NULL ,
                 PRIMARY KEY (`SaleItemID`))";
        mysqli_query($connection,$query);

        /* Insert new transaction table name into transaction history table */
        $query = "INSERT INTO transactionhistory_$companyName(Complete)
                  VALUES('no')";
        mysqli_query($connection,$query);

        /* Update the account type of the current user to owner, as they are the user
who registered their company to
           the program */
        $query = "UPDATE Credentials SET accountType = 'owner'";
        mysqli_query($connection,$query);

    }

    echo "Username Created";// temp
}
else{
    /* Output error messages */
    if($numResults != 0 ) {
        echo "Username already in use\n";// temp
    }
    else{
        echo "Email is invalid\n";// temp
    }
}

header('refresh: 2; URL = HomePage.php');// redirect to home page
exit;
?>
```

Process_StockArrived.php

```php
<?php
session_start();

/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

/* Calculate new stock value */
$priorStockValue = $_GET['currentStockValue'];
$arrivalStockValue = $_GET['arrivalQuantity'];
$newStockValue = $priorStockValue + $arrivalStockValue;

/* Update the values of the product in the company's stock management database */
$productID = $_GET['productID'];
$tableName = "stockmanagementtable_".$_SESSION['companyName'];
$query = "UPDATE $tableName SET CurrentStockValue = '$newStockValue', Ordered = 'off',
ArrivalDate= NULL,
          ArrivalAmount = NULL WHERE ProductID = '$productID'";
mysqli_query($connection,$query);

/* Redirect the user */
header('refresh: 0; URL = pointOfSale.php');// redirect to home page
exit;
```

Process_UpdateAccountType.php

```php
<?php
session_start(); //start the current session allowing session variables to be used

/* Open DB connection */
include "database_connect.php";
$connection = openConnection();

/* Retrieve the required user to update their account type using $_GET */
$usernameToUpdateAccountType = $_GET['usernameToUpdate'];

/* Update the retrieved user's account permissions to standard */
$query = "UPDATE Credentials SET accountType = 'standard' WHERE Username =
'$usernameToUpdateAccountType'";
mysqli_query($connection, $query);

/* Refresh the user back to the drop-down menu */
header('refresh: 0; URL = dropdown.php'); //redirect the current user
exit;
```

Refund.php

```
<!DOCTYPE html>
<head>
    <link rel="stylesheet" href="Styles.css" type="text/css"> <!-- link HomePage to
style sheet -->
</head>

<html>
<body>

<div class="toolbar">
    <a href="HomePage.php">Home</a>
    <a href="MyMenus.php">MyMenus</a>
    <a href="stockManagement.php">Stock Management</a>
    <a href="pointOfSale.php">POS</a>
    <?php session_start();
    $username = $_SESSION['username']; // get username from login process
    ?>
    <a style="float: right;" class="current" href="dropdown.php"> <?php echo $username;
?></php></a>
</div>

<form action = "process_refund.php">
    <label>Refund Amount:<input type="number" step="0.01" value = 0
name="refundAmount"></label>
    <button type="submit" name="submit">Submit</button>
</form>

<?php
require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"refund.php");
?>
```

Signup.php

```
<!DOCTYPE html>
<html>

<?php
/* CONNECT TO LOCAL HOST */
$servername = "localhost";
$username = "username";
$password = "password";

// Create a connection to the local host
$connection = new mysqli($servername, $username, $password);

/* Check the connection
if ($connection->connect_error) {
    die("The connection has failed: " . $connection->connect_error);
}
echo "Connected successfully"; */

?>

<head>
    <link rel="stylesheet" href="Styles.css" type="text/css"> <!-- link Signup Page to
style sheet -->
</head>

<!-- toolbar -->
<div class="toolbar">
    <a href="HomePage.php">Home</a>
    <a href="MyMenus.php">MyMenus</a>
    <a class="current" style="float: right;" href="Signup.php">Sign up</a>
    <a style="float: right;" href="Login.php">Log-in</a>
</div>

<form action="process_signup.php"> <!-- send entered data into process_signup.php for
processing -->
    <div class ="signup_container">

        <h1>Sign up</h1>

        <label for="username">Username</label>
        <input type="text" name="username" placeholder="Enter Your Username Here"
required>

        <br>

        <label for="password">Password</label>
        <input type="text" name="password" placeholder="Enter Your Password Here"
required>

        <br>

        <label for="companyName">Company Name</label>
        <input type="text" name="companyName" placeholder="Enter Your Company Name
Here" required>

        <br>

        <label for="email">Email</label>
        <input type="email" name="email" placeholder="Enter Your Email Address Here">

        <br>

        <label for="phone">Phone</label>
        <input type="number" name="phone" placeholder="Enter Your Phone Number Here">

        <br>
```

```
        <button type="submit">Sign Up</button>
    </div>

</form>

</body>
</html>
```

StockManagement.php

```
<?php
session_start();

/* Open Database Connection */
include "database_connect.php";
$connection = openConnection();

/* Check If Any Product Are Low On Stock */
checkForLowStock($connection,$_SESSION['companyName']);

echo"
<head>
    <link rel='stylesheet' href='Styles.css' type='text/css'> <!-- link HomePage to
style sheet -->
</head>

<!-- Toolbar -->
<div class='toolbar'>
    <a href='HomePage.php'>Home</a>
    <a href='MyMenus.php'>MyMenus</a>
    <a href='stockManagement.php' class='current'>Stock Management</a>
    <a href='pointOfSale.php'>POS</a>
    <a style='float: right;' href='Signup.php'>Sign up</a>
    <a style='float: right;' href='Login.php'>Log-in</a>
</div>";

echo "<div class='Container_SM_Overview'>
    <table id='stockManagementOverview'>
        <!-- Set Table Headings -->
        <tr>
            <th>Item Name</th>
            <th>Current Quentity</th>
            <th>Minimum Quantity</th>
            <th><form action='newProduct.php'><button type='submit'>+New
Product</button></form></th>
        </tr>
        ".addEntriesToTable(strtolower($_SESSION['companyName']),$connection).

   "</table>
</div>";



/* Calculates the number of rows
   Returns number of rows found in the table
   if now results found, returns 0 */
function getNumRows($companyName, $DBconnection){
    $tableName = "stockmanagementtable_".$companyName;
    $query = "SELECT ProductName FROM $tableName";
    $result = mysqli_query($DBconnection, $query);
    if($result != false) {
        return mysqli_num_rows($result);
    }
    else{
        return 0;
    }
}

function addEntriesToTable($companyName, $connection){
    $tableName = "stockmanagementtable_".$companyName;
    $query = "SELECT
ProductID,ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone
              FROM $tableName";
    $result = mysqli_query($connection, $query);

    /* Ouput data of each row */
```

```
    $ret = "";
    for($i=0;$i<getNumRows($companyName,$connection);$i++){
        $row = mysqli_fetch_assoc($result);
        $ret = $ret.'<tr><td>'.$row["ProductName"].'</td>
                    <td> '.$row["CurrentStockValue"].'</td>
                    <td>'.$row["MinimumStockValue"].'</td>
                    <td><a
href="EditProduct.php?ProductID='.$row['ProductID'].'&TableName='.$tableName.

'&ProductName='.$row["ProductName"].'&CurrentStockValue='.$row["CurrentStockValue"].

'&MinimumStockValue='.$row["MinimumStockValue"].'&Ordered='.$row["Ordered"].

'&SupplierName='.$row["SupplierName"].'&PhoneNumber='.$row["Phone"].
                                '">Edit Product</a></td></tr>';
    }
    return $ret;
}

function checkForLowStock($connection,$companyName){
    $tableName = "stockmanagementtable_".$companyName;
    $query = "SELECT
ProductID,ProductName,MinimumStockValue,CurrentStockValue,Ordered,SupplierName,Phone
            FROM $tableName WHERE CurrentStockValue < MinimumStockValue AND
Ordered='off'";
    $result = mysqli_query($connection, $query);

    /* Calculate the number of rows returned by the query */
    $numRows = 0;
    if($result == false){
        $numRows = 0;
    }
    else{
        $numRows = mysqli_num_rows($result);
    }

    if($numRows > 0) {
        $_SESSION['returnedRows'] = mysqli_fetch_assoc($result);
        $_SESSION['companyName'] = $companyName;
        $_SESSION['tableName'] = $tableName;

        header('Location: lowStock.php');// redirect user
        exit;

    }
}

require_once('functions.php');
isUserAllowedAccessToThisPage($_SESSION['accountType'],"stockManagement.php");
?>
```

Styles.css

```
/*TOOLBAR*/
/* Remove margins from <body> tag */
body{
    margin: 0px;
}

/* Add green background to the toolbar */
.toolbar {
    background-color: limegreen;
    overflow: hidden;
}

/* Style the links in the toolbar */
.toolbar a {
    float: left; /* links move from the left to right. Some links have inline styling
to change this*/
    text-decoration: none; /* removes the underline from the hyperlink */
    text-align: center;
    color: #f2f2f2; /* change the colour of text on inactive web pages */
    font-size: 20px;
    font-weight: bold;
    padding: 16px 16px; /* increase the height of the colour above and below the text
*/
}

/* Add a color to the current web page title */
.toolbar a.current {
    background-color: #04aa20;
    color: #0000005A;
}

/* Change the colour of links when hovered over */
.toolbar a:hover {
    background-color: #00ba00;
    color: #0000005A;
}

/* HEADINGS */
h6{
    text-align: center;
    font-size: 26px;
    text-underline: black;
}



/* EMBEDDED VIDEO */
#embeddedVideo{
    text-align: center;
}



/* SM Overview Table */
table, th, td {
    border:1px solid black;
}
```