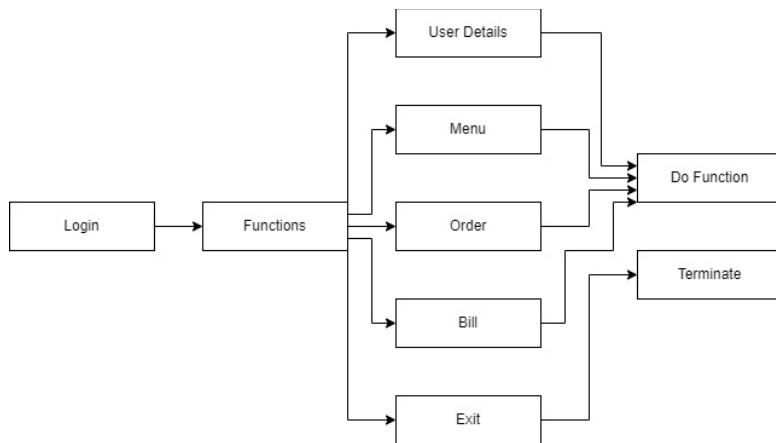
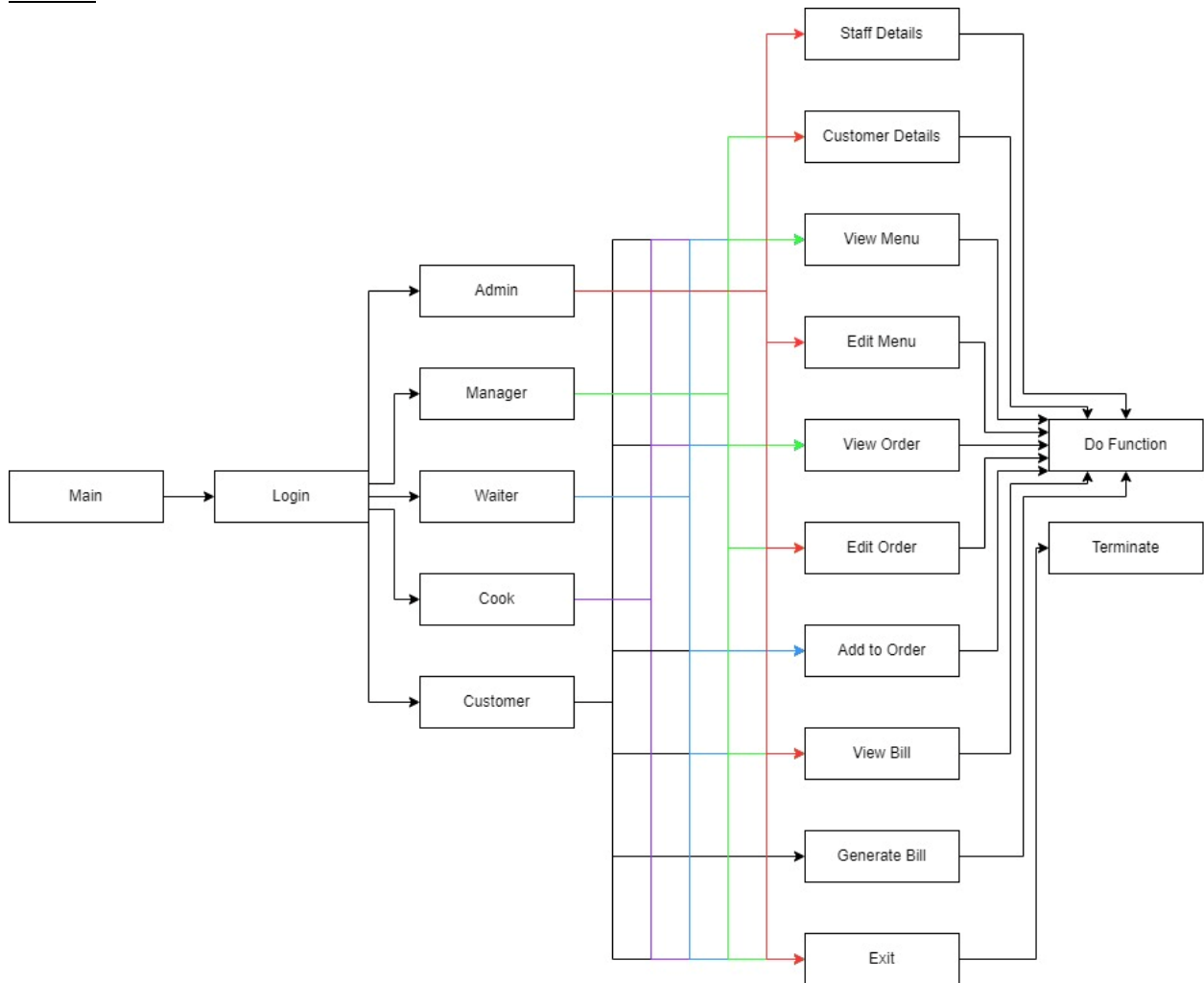


Block Diagram:

Level 1:

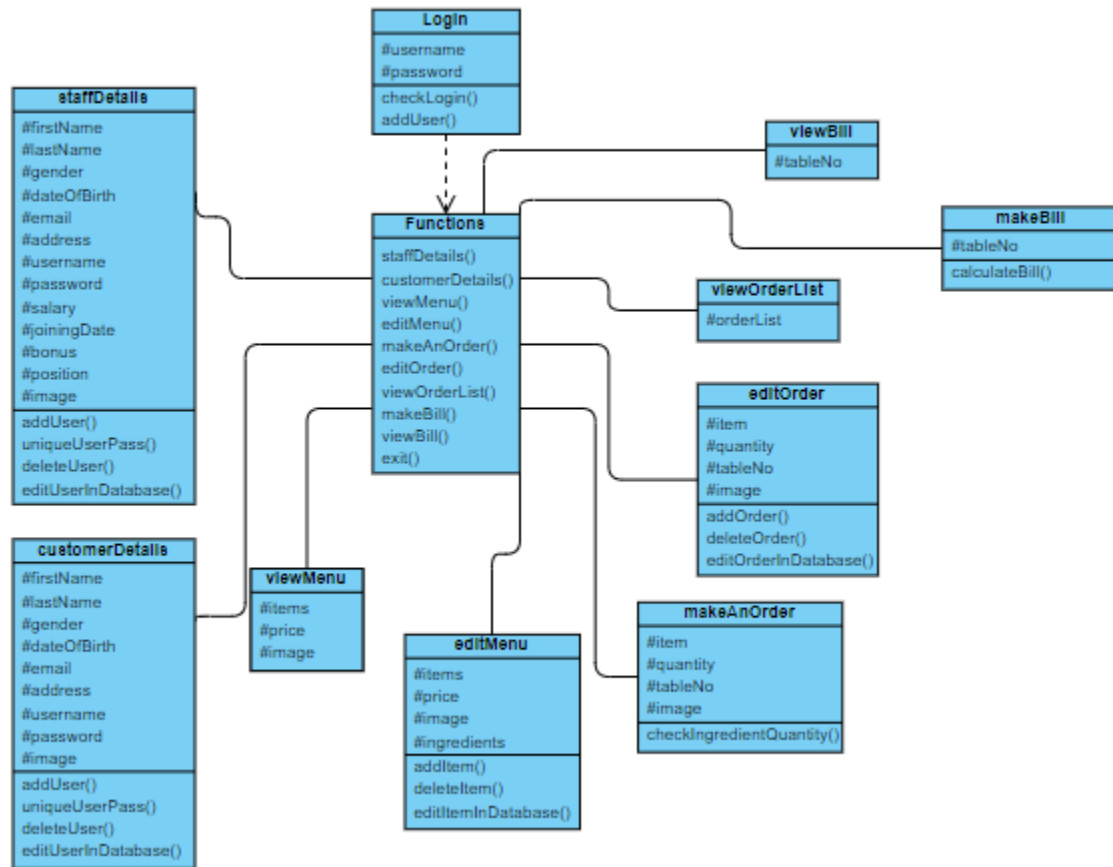


Level 2:



UML Diagram:

Level 1:



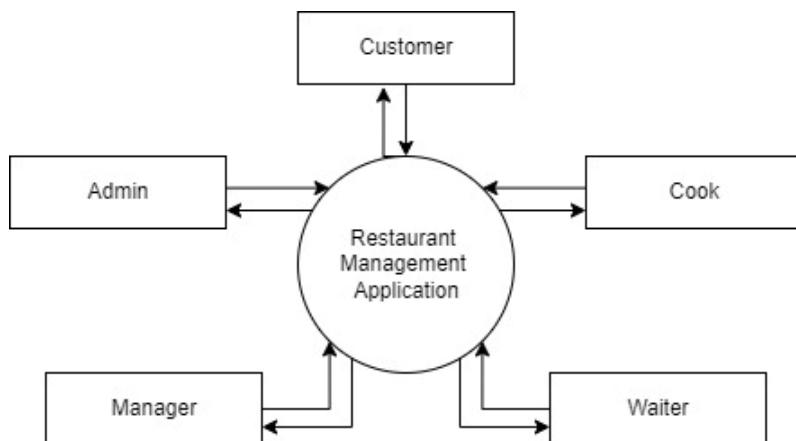
Class Functionality:

Class Name	Functionality
Login	This is where the user can enter the username and password, to get access to the rest of the functionalities.
staffDetails	This takes care of the staff details. This includes adding new staff and updating or deleting old staff.
customerDetails	This takes care of the customer details. This includes adding new customers and updating or deleting old customers.
viewMenu	This takes care of showing the user the menu.

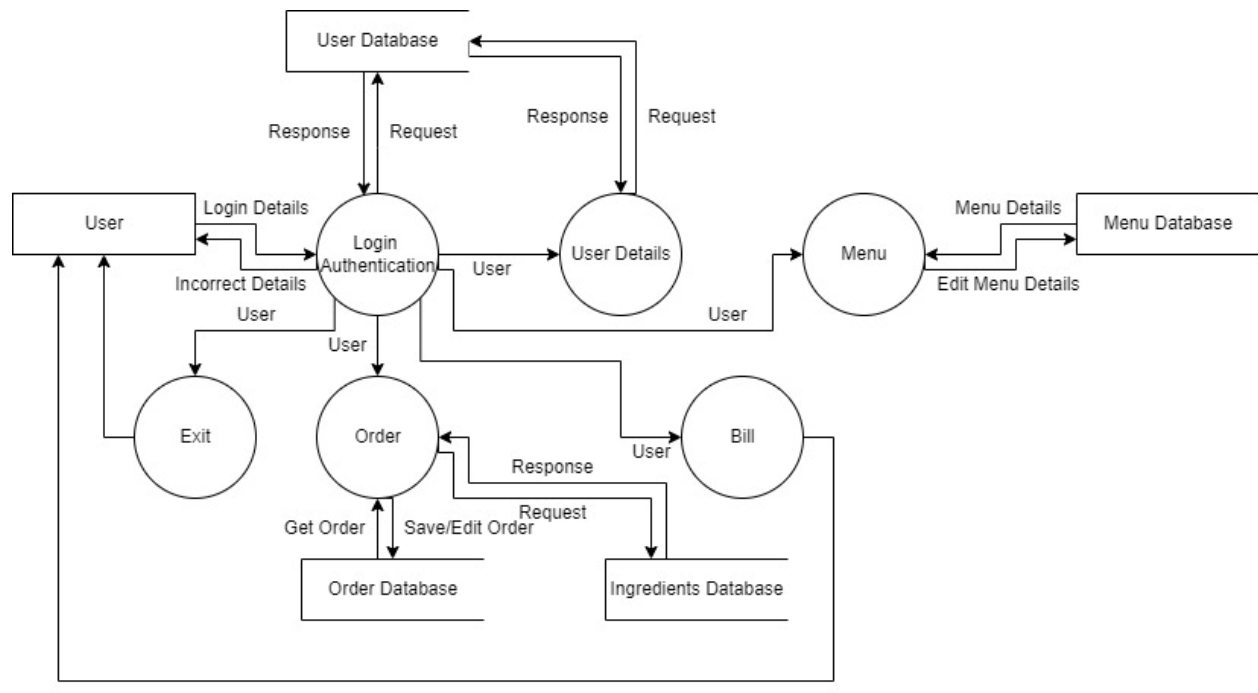
editMenu	This takes care of the menu. This includes adding new items and updating or deleting old items in the menu.
makeAnOrder	This takes care of adding new orders from the customer. It also checks to see if there are enough ingredients left in the inventory.
editOrder	This takes care of the order. This includes adding new orders and updating or deleting old orders.
viewOrderList	This takes care of showing the pending orders that are waiting to be made.
viewBill	This takes care of showing the bill and thus showing them their purchases.
makeBill	This takes care of showing the final bill and sending messages to the waiter/ manager for them to collect the money.

Data Flow Diagram:

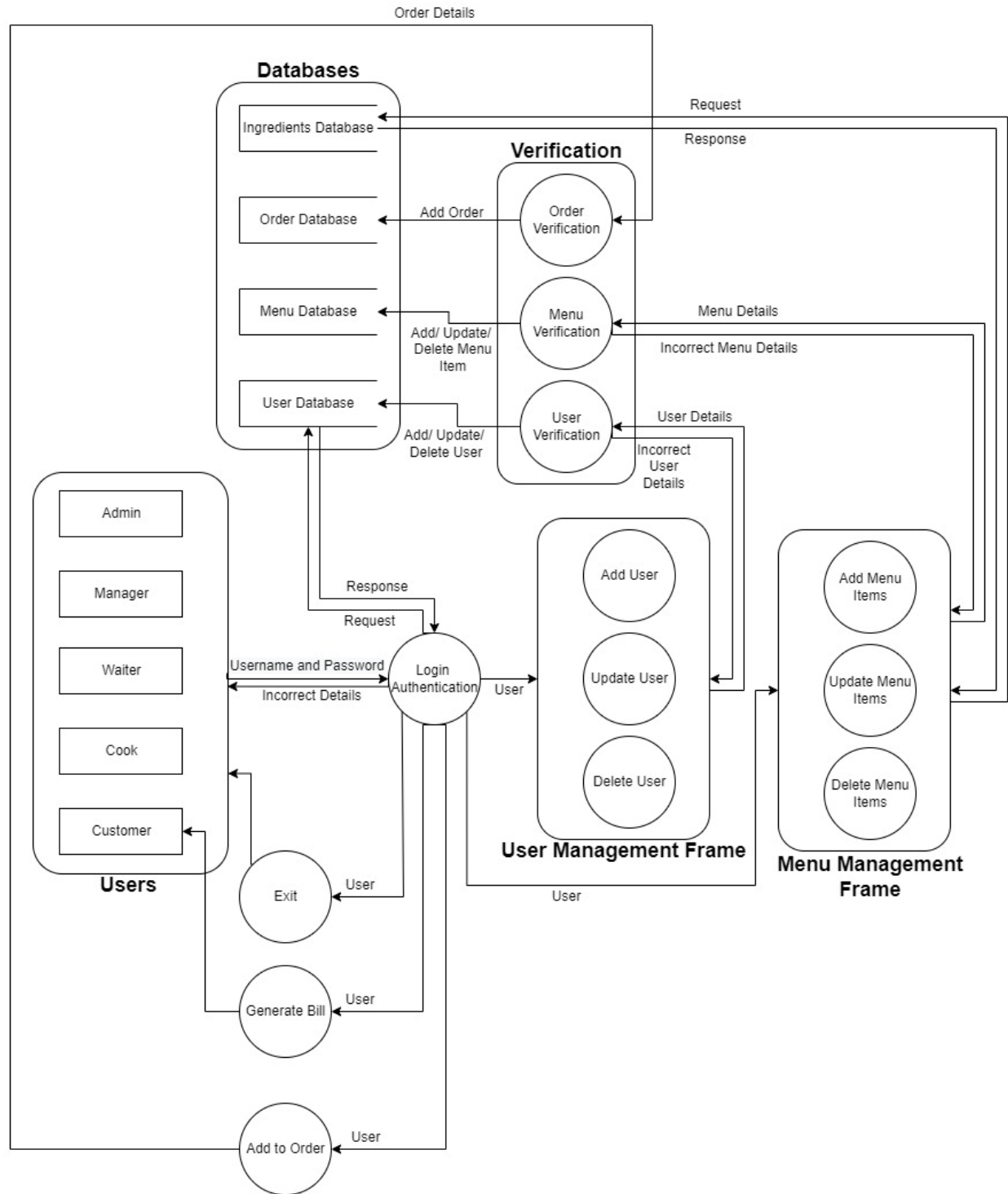
Level 0 (Context Level):



Level 1:

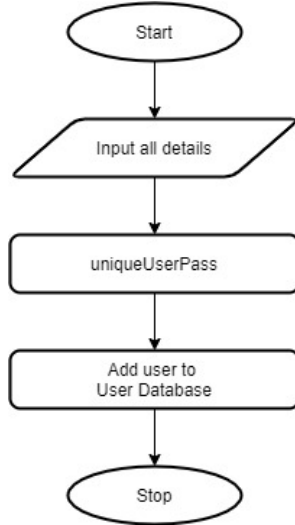


Level 2:



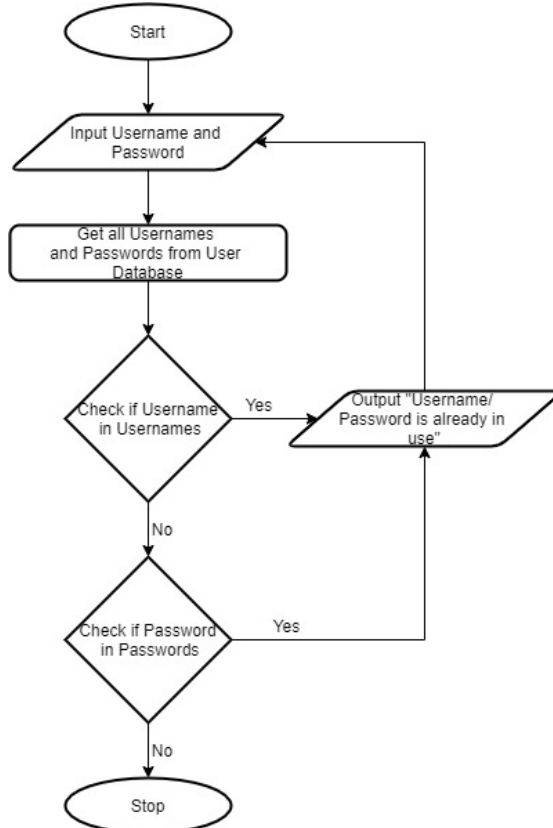
Flowcharts and Pseudocode:

1. Adding a new user to User Database.



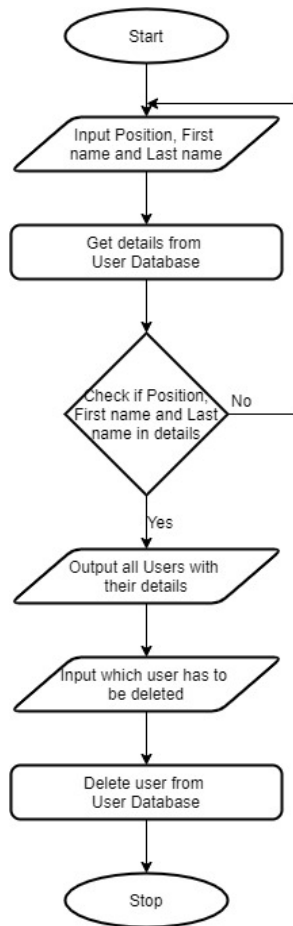
```
START
INPUT firstName, lastName, gender, DOB, email, address, username,
password, salary, joinDate, bonus, position
FUNCTION uniqueUserPass()
WRITE firstName, lastName, gender, DOB, email, address, username,
password, salary, joinDate, bonus, position to User Database
END
```

2. Checking if the user has a unique username and password.



```
START
INPUT username, password
READ usernames and passwords from User Database
IF username in usernames OR password in passwords THEN
OUTPUT "Username/ Password is already in use"
RETURN 0
ELSE
RETURN 1
END
```

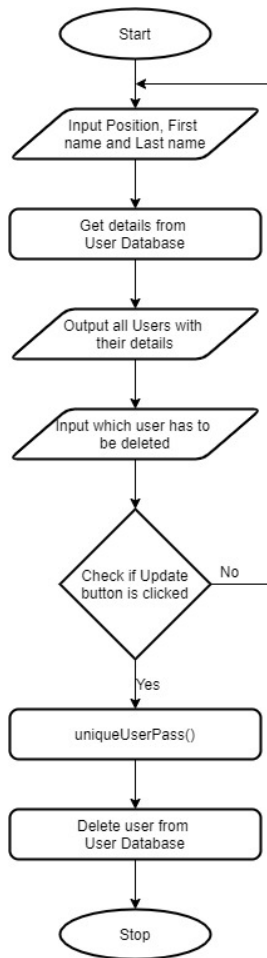
3. Deleting a user from User Database.



```

START
INPUT position, firstName, lastName
READ positions, firstNames and lastNames from User Database
IF position in positions AND firstName in firstNames AND lastName in lastNames THEN
    OUTPUT all those users in the User Database
    INPUT user that is to be deleted
    DELETE user from User Database
END
  
```

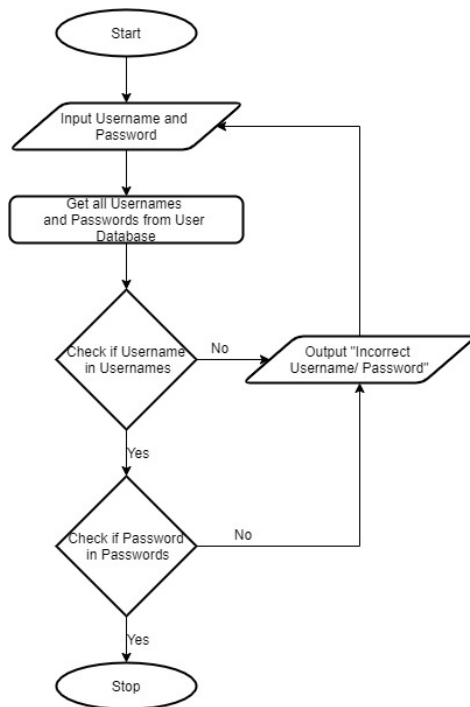
4. Editing a user in User Database.



```

START
INPUT position, firstName, lastName
READ positions, firstNames and lastNames from User Database
OUTPUT all users with the position, firstName and lastName in the User Database
INPUT user that is to be updated
DELETE user from User Database
IF updateButton.click() = True THEN
    FUNCTION uniqueUserPass()
        WRITE user to User Database
    END
END
  
```

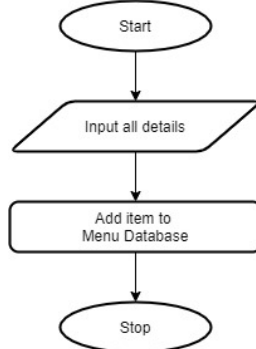
5. Checking if the login details match with the data in the User Database.



```

START
INPUT username, password
READ usernames and passwords from User Database
IF username in usernames OR password in passwords THEN
  RETURN 1
ELSE
  OUTPUT "Incorrect Username/ Password"
  RETURN 0
END
  
```

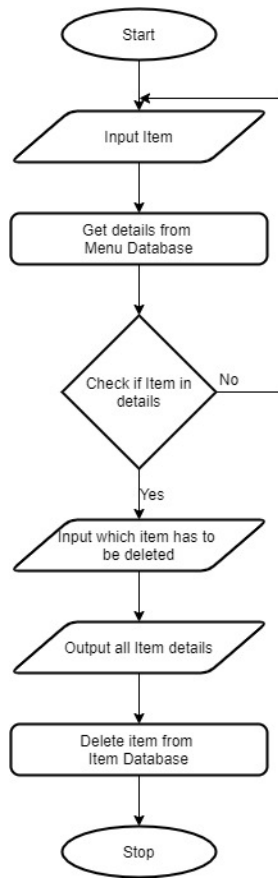
6. Adding a new item to the Menu Database.



```

START
INPUT itemName, price, ingredients, ingredientQuantity
WRITE itemName, price, ingredients, ingredientQuantity to Menu Database
END
  
```

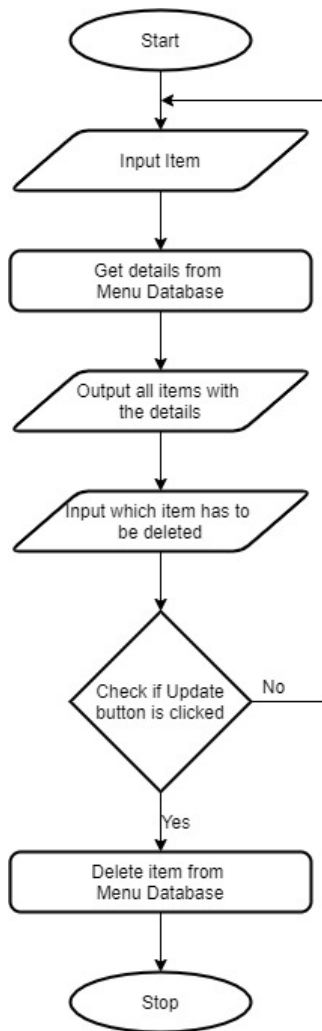
7. Deleting an item from Menu Database.



```

START
INPUT itemName
READ itemNames from Menu Database
IF itemName in itemNames THEN
    OUTPUT all those items in the Menu Database
    INPUT item that is to be deleted
    OUTPUT item details
    DELETE item from Menu Database
END
  
```

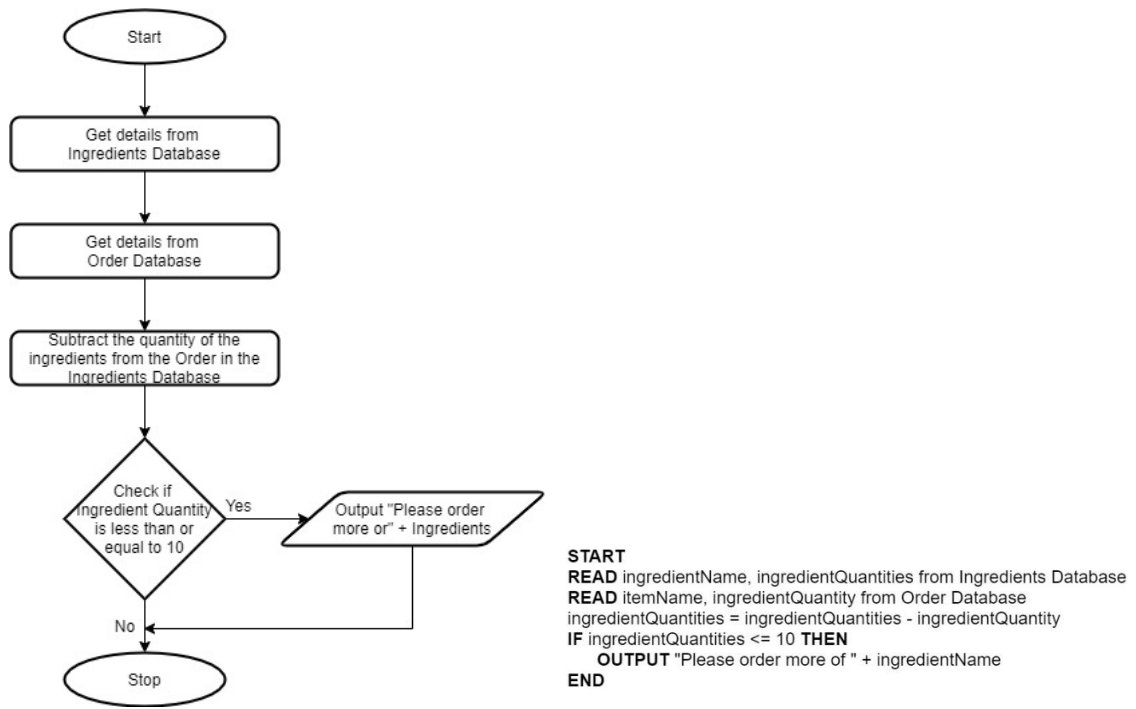
8. Editing an item in Menu Database.



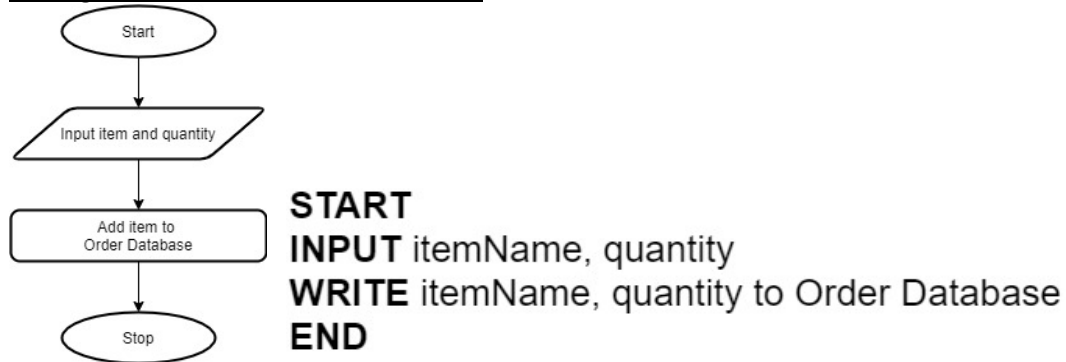
```

START
INPUT itemName
READ itemNames from Menu Database
OUTPUT all items with the itemName in the Menu Database
INPUT item that is to be updated
DELETE item from Menu Database
IF updateButton.click() = True THEN
    WRITE item to Menu Database
END
  
```

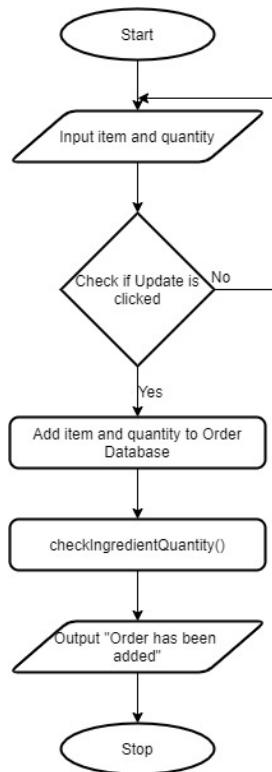
9. Checking the ingredient stock numbers.



10. Adding an order to the Order Database.



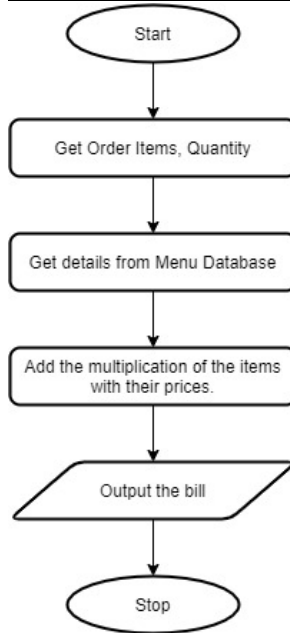
11. Editing an order in the Order Database.



```

START
INPUT itemQuantity for each itemName
DELETE order from Order Database
IF updateButton.click() = True THEN
    FUNCTION checkIngredientQuantity()
        WRITE itemName, itemQuantity to Order Database
    END
  
```

12. Calculating the final bill.



```

START
READ itemNames, itemQuantity from Order Database
READ itemNames, itemPrice from Menu Database
bill = sum of itemPrice * itemQuantity
OUTPUT bill
END
  
```

Design of Panels (Graphical User Interface):

Login

Subway

Login

Username:

Password:

User Sign Up

Subway

Sign Up

First Name:

Last Name:

Gender:

Date of Birth:

Email:

Address:

Username:

Password:

Upload Photo

Salary:

Joining Date:

Bonus:

Position:

User Functions

Admin

Subway	
Functions	
Staff Details	
Customer Details	
Edit Menu	
Edit Order	
View Bill	
Exit	

Manager

Subway	
Functions	
Customer Details	
View Menu	
View Order	
Edit Order	
View Bill	
Exit	

Waiter

Subway	
Functions	
View Menu	
View Order	
Add to Order	
View Bill	
Exit	

Cook

Subway	
Functions	
View Menu	
View Order	
Exit	

Customer


Subway	
Functions	
View Menu	
View Order	
Add to Order	
View Bill	
Generate Bill	
Exit	

Staff and Customer Details

Subway	
Functions	<div>Add User</div> <div>Update User</div> <div>Delete User</div>
Staff Details	
Customer Details	

Add User

Subway

Functions		Sign Up	
Staff Details	First Name: <input style="width: 90%;" type="text"/>		
Customer Details	Last Name: <input style="width: 90%;" type="text"/>	<input style="width: 100%;" type="button" value="Upload Photo"/>	
	Gender: <input style="width: 90%;" type="text"/>	Salary: <input style="width: 90%;" type="text"/>	
	Date of Birth: <input style="width: 90%;" type="text"/>	Joining Date: <input style="width: 90%;" type="text"/>	
	Email: <input style="width: 90%;" type="text"/>	Bonus: <input style="width: 90%;" type="text"/>	
	Address: <input style="width: 90%;" type="text"/>	Position: <input style="width: 90%;" type="text"/>	
	Username: <input style="width: 90%;" type="text"/>	<input style="width: 45%;" type="button" value="Submit"/> <input style="width: 45%;" type="button" value="Back"/>	
	Password: <input style="width: 90%;" type="text"/>		

Update User

Subway

Functions	Search Bar	Update User
Staff Details	<div>Index() Position: Name</div>	First Name: <input type="text"/>
Customer Details		Last Name: <input type="text"/>
		Gender: <input type="text"/>
		Date of Birth: <input type="text"/>
		<div><input type="text"/></div> <div>Upload Photo</div>
		Salary: <input type="text"/>
		Joining Date: <input type="text"/>
		Bonus: <input type="text"/>
		Position: <input type="text"/>
<div>View Details</div>		<div>Update</div> <div>Back</div>

Delete User

Subway		
Functions	<div>Search Bar</div> <div>Index) Position: Name</div> <div>Staff Details</div> <div>Customer Details</div> <div>View Details</div>	<div>Delete User</div> <div>First Name: <input type="text"/></div> <div>Last Name: <input type="text"/></div> <div>Gender: <input type="text"/></div> <div>Date of Birth: <input type="checkbox"/></div> <div>Email: <input type="text"/></div> <div>Address: <input type="text"/></div> <div>Username: <input type="text"/></div> <div>Password: <input type="text"/></div> <div><div><div></div></div><div>Upload Photo</div></div> <div>Salary: <input type="text"/></div> <div>Joining Date: <input type="checkbox"/></div> <div>Bonus: <input type="text"/></div> <div>Position: <input type="text"/></div> <div>Delete Back</div>

View Menu

Subway						
Functions	Menu					
View Menu	Item	Price	Item	Price	Item	Price

Edit Menu

```
graph TD
    subgraph Subway
        direction TB
        UC1[1. Add Item]
        UC2[2. Update Item]
        UC3[3. Delete Item]
    end
    subgraph Functions
        direction TB
        UC4[4. Edit Menu]
    end
```

The diagram illustrates the functional requirements for a Subway menu system. It is divided into two main sections: **Subway** and **Functions**.

The **Subway** section contains three use cases:

- 1. Add Item
- 2. Update Item
- 3. Delete Item

The **Functions** section contains one use case:

- 4. Edit Menu

Subway

Functions		Add Item																					
Edit Menu	Item: <input style="width: 90%;" type="text"/> Price: <input style="width: 90%;" type="text"/> Ingredients: <table border="1" style="display: inline-table; vertical-align: top;"> <tr><td style="width: 80%;"></td><td style="width: 20%; text-align: center;">Add</td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> <tr><td></td><td></td></tr> </table>		Add																			<div style="border: 1px solid black; width: 200px; height: 150px; margin: 0 auto;"></div> <div style="margin: 10px auto; width: 100px;">Upload Photo</div>	
	Add																						
		<div style="display: inline-block; margin-right: 20px;">Submit</div> <div>Back</div>																					

Subway

Functions	Search Bar	Update Item		
	Index Item	Item:	<input type="text"/>	<div style="border: 1px solid black; width: 100px; height: 100px; margin: 0 auto;"></div> <div style="border: 1px solid black; padding: 5px; margin: 10px auto; width: 80%;">Upload Photo</div> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <div style="border: 1px solid black; padding: 5px;">Update</div> <div style="border: 1px solid black; padding: 5px;">Back</div> </div>
Edit Menu		Price:	<input type="text"/>	
		Ingredients:	<input type="text"/> <div style="border: 1px solid black; padding: 2px 5px; float: right;">Add</div>	
		Quantity:	<input type="text"/> <div style="border: 1px solid black; padding: 2px 5px; float: right;">Add</div>	
			<input type="text"/>	
			<input type="text"/>	
			<input type="text"/>	
			<input type="text"/>	
View Details			<div style="border: 1px solid black; padding: 2px 5px; float: right;">Delete</div>	

The diagram illustrates the layout of a Subway mobile application. It features a top header with the 'Subway' logo. A left sidebar, labeled 'Functions', contains navigation options: 'Search Bar', 'Index Item', 'Edit Menu', and 'View Details'. The main content area, titled 'Delete Item', is divided into several sections. On the left, there are input fields for 'Item:', 'Price:', and 'Ingredients:'. The 'Ingredients' section includes a table with 'Quantity' and 'Add' columns. To the right of these fields is a large rectangular placeholder for a photo, with an 'Upload Photo' button positioned below it. At the bottom right of the main area are 'Delete' and 'Back' buttons.

[View Order](#)

View Individual Order

Subway								
Functions								
View Order								
	<table><tr><td>Table 1</td><td>Table 2</td><td>Table 3</td></tr><tr><td>Table 4</td><td>Table 5</td><td>Table 6</td></tr></table>		Table 1	Table 2	Table 3	Table 4	Table 5	Table 6
Table 1	Table 2	Table 3						
Table 4	Table 5	Table 6						

Subway						
Functions	Table Number: <input type="text"/>					
View Order						
	Item	Quantity	Item	Quantity	Item	Quantity

Add to Order

Subway

Functions

Add to Order

Table 1

Table 2

Table 3

Table 4

Table 5

Table 6

Subway

Functions

Add to Order

Table Number:

Item	Quantity	Item	Quantity	Item	Quantity
Item 1	<input type="text"/>				
<div>Add to Order</div>					

[View Bill](#)

Subway

Functions

Add to Order

Table 1

Table 2

Table 3

Table 4

Table 5

Table 6

Subway

Functions

View Bill

Bill

Date:

Table No:

Item	Price	Quantity	Amount

Total:

Generate Bill

Test Plan:

Success Criteria	Action to be Tested	Test Method	Expected Result
Login GUI Page which separates the customers, admin/ shareholders, manager, waiters, and cooks	Check whether the functionalities change when different users press the "login" button.	Login to the different user accounts.	There would be different functions that could be done by user according to the user account.

		<p>Login with wrong username/ password (Check case-sensitive)</p> <p>Login with empty username or password</p>	<p>If username, password, or both are wrong, a message would come up indicating that the username/password is incorrect.</p> <p>If username and password is correct, the user would get logged in.</p> <p>If the case of username/ password is wrong, a message would come up indicating that the username/password is incorrect.</p> <p>If the username or password is empty, a message would come up indicating that the username/password is incorrect.</p>
A layout of Sign-Up screen that changes according to the user being a staff member or customer.	Check whether the Sign-Up screen hides the salary, joining date, and bonus fields if "Add Customer" is clicked and shows the fields if "Add Staff" is clicked.	<p>Click the "Create Customer Account" in the login page to test for the hiding of the fields.</p> <p>Login as admin and click "Add Staff" in 'Staff Details' to test if the fields are showing.</p>	When "Create Customer Account" is clicked, the salary, joining date, and bonus fields would be hidden and when "Add Staff" is clicked, salary, joining date, and bonus fields would be visible.
A layout of Functional screen that changes according to the user being a staff member or customer.	Check whether the functionalities change when different users press the "login" button.	Login to the different user accounts.	There would be different functions that could be done by user according to the user account.
Validation to make sure the details entered/ edited in the Login page does not have any human error.	Check whether the details inserted by the user in the Login page is correct.	<p>Insert incorrect username and password.</p> <p>Insert username of one user and password or another.</p>	If username and password are both wrong or are from two different users, an error message would show up.

		Insert correct username and password	If username and password are correct and are of the same user, the user will be logged in.
Validation to make sure the details entered/ edited in the User Details page does not have any human error.	Check whether the details inserted by the user in the User Details page is correct.	<p>Insert email with and without “@”.</p> <p>Insert existing and non-existing username and password.</p> <p>Insert 2 and more decimal places in salary and bonus.</p> <p>Empty entry boxes when update is clicked.</p>	<p>If email is inserted without “@”, a message would come up indicating that the email is incorrect. If the email is inserted with “@”, and everything else is correct, the user would be added to the database.</p> <p>If username and password is inserted that is already present in the database, a message would come up indicating that the username/ password is incorrect. If the username and password inserted is unique, and everything else is correct, the user would be added to the database.</p> <p>If the decimal place in salary or bonus is more than 2, the program should automatically round it to 2 decimal places. If the decimal place is less than 2, it should add the correct number of 0s to make it 2 decimal places.</p> <p>When update is clicked, if any entry box is empty, an error will show up stating that not all fields are filled.</p>

Validation to make sure the details entered/ edited in the Menu Details page does not have any human error.	Check whether the details inserted by the user in the Menu Details page is correct.	<p>Insert item name field with and without numbers.</p> <p>Insert item price field with and without alphabets.</p> <p>Empty ingredients listbox with Submit is clicked.</p>	<p>If the item name field has numbers, an error message would show up. If the item name doesn't include numbers, and everything else is correct, item will be added to the database.</p> <p>If the item price field has alphabets, an error message would show up. If the item price doesn't include alphabets, and everything else is correct, item will be added to the database.</p> <p>If the ingredients listbox has a length of 0, an error message would show up. If the ingredients listbox has a length more than 0, and everything else is correct, then the item will be added to the database.</p>
Validation to make sure the details entered/ edited in the Order Details page does not have any human error.	Check whether the details inserted by the user in the Order Details page is correct.	Insert item quantity field with and without alphabets.	If the item quantity field has alphabets, an error message would show up. If the item quantity doesn't include alphabets, and everything else is correct, the order will be added to the database.
A layout of Menu screen that changes according to the number of menu items	Check whether the menu items change when menu items are added/deleted.	Add and delete menu items	<p>Create a new menu item and check if the View Menu page shows the new menu item.</p> <p>Delete the new menu item and check if View</p>

			Menu page shows the original list of menu items.
A layout of Order screen that changes as menu items are ordered.	Check whether the items possible to order change when menu items are added/deleted.	Add and delete menu items	<p>Create a new menu item and check if the Add to Order page shows the possibility of ordering the new menu item.</p> <p>Delete the new menu item and check if Add to Order page shows the original list of menu items to order from.</p>
A layout of Bill screen that changes as more orders are placed.	Check whether the bill changes when more items are ordered	Add menu items to order list	Order a menu item and check if the View Bill page shows the new order.