"Transaction Tracking System for Small Business Owners"

Bal, Kevin Ranz J.
Bandigan, Christian Lee C.
Canoy, Hail B.
Pabalan, Hanz Rhayven M.

Technological Institute of the Philippines
Quezon City

November 2025

Table of Contents

Introduction

Small business owners mostly rely on notebooks, especially the older generations to record their sales, payments, and inventory. These are the traditional ways but are time consuming and inefficient. As the business grows it will have more sales and payments to track making it more difficult to do *(Giddh Blog, 2025).*

Many small business owners struggle to accurately track their sales, payments, and inventory because they only rely on manual, handwritten recording. Digital systems exist, but they are often perceived as expensive or too complicated to use, causing business owners to avoid them. As a result, manual tracking leads to frequent human errors, such as miscalculations, missing entries, and inaccurate inventory counts, which can cause product shortages or overstocking and negatively impact the store's income. Studies show that shifting to an automated system significantly reduces errors and speeds up inventory updates, proving that digital tools can make tracking more accurate and efficient (Wynn & Kuhn, 2021; Gestisoft, 2023; Retalon, 2025; Lee et al., 2021).

To address this problem, The team would like to propose a Transaction Tracking System and Management System that will simplify payment, and inventory tracking for business owners. Instead of relying on hand written notebooks our system will allow users to input transaction details digitally. And it will automatically generate an organized receipt that can be viewed or printed using notepad.

The Project

The proposed project is a Transaction Tracking and Management System that will help small business owners ease their tasks such as to simplify sales and payments, and track inventory.
Many small business owners still rely on traditional methods that can be time consuming and prone to errors. This system will replace the traditional methods with its simple, easy to use that is made using the C++ programming language. This project includes login, account management , inventory management, sales transaction, and receipt printing features.

The login is the first step in the Transaction Tracking and Management System. The user will be presented a menu login for Admin and Cashier, depending on what the user picks. For the login system if the user chose the cashier or admin the user will be asked to enter the correct admin password and username if it's not valid the program will display an error message. If the login is successful the system will direct the user to the chosen menu. If the user fails to input a valid username and password three times the program will stop. This login feature ensures that the system can only be accessed by authorized users.

The admin menu allows the user to manage cashier accounts more efficiently, easily and in an organized manner. Once logged in the user will be presented with three account management actions, the user can add a new cashier by entering the new cashier name and its password. The admin can also remove

existing cashiers just by simply entering their username. The admin will also be able to view all cashier accounts.

The cashier menu allows the cashier to record transactions or manage inventory. Once logged in the user can then access the menu. In the sales transaction feature the user can simply enter the product id and its quantity and the system will automatically check if the product exists and if not it will print an error message, if the product exists the user will then need to enter a quantity of the product. After entering the desired quantity it will check the inventory if it has enough stock. If there are insufficient stocks it will display the amount available and if the product is available it automatically computes for the subtotal. After entering a product it will ask the cashier if they want to add more products and will only stop once it is done. Following this, it will compute the total amount of all the products, ask the user to input payment, compute for change, update inventory and print out the receipt through notepad with all the product information, total amount, and change. The inventory allows the cashier to add new products by entering a unique id with its name, price, and stock. It will also allow the user to to update existing products by entering their Id to change the product information and to delete products when it's no longer available. After accessing its respective menu and transaction both the cashier and admin will have an option to do another transaction or to go back in the main menu to log out and exit the program.

Objectives

The objective of this project is to develop a secure system that manages inventory and simplifies sales transaction for small business owners that can be easily used without access or knowledge of technical tools.Specifically the project aims to:

1. Develop a system that:
    a. Provides secure access through a cashier and admin login feature.
    b. Create a user-friendly C++ system that simplifies product transactions.
    c. Manages inventory by adding, updating, deleting, and viewing products.
    d. Generates a receipt after each transaction.

2. Test and evaluate the system’s accuracy.

Flowchart of the System

This subsection explains the overall flow of the system through a flowchart,This flowchart helps to visualize the whole system of the user. The system starts at the Main Menu, where the user has three options to choose. Choosing Option 1 brings the user to the Admin Menu, where the user can add or delete cashier accounts and view all existing cashiers. Choosing option 2 brings up the Cashier Menu, which contains sales and inventory sections. In the sales section, the user can complete a transaction by entering a product ID, quantity, and payment and receiving a receipt. The inventory section allows the user to add, update, and delete products, and to see the updated list. Finally, selecting option 3 stops the program.

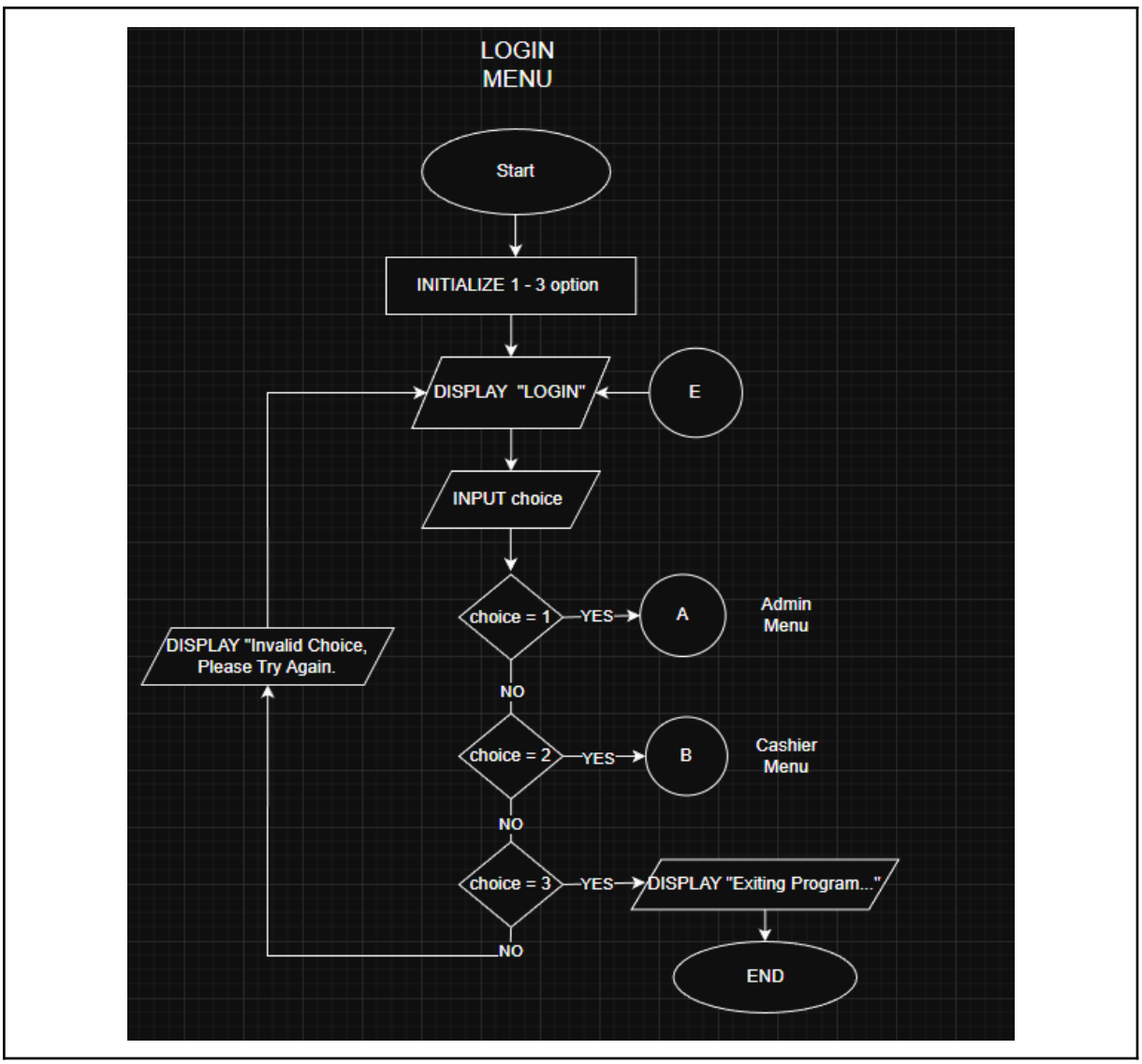

Figure 1: LOGIN MENU

In this figure, this is where the login interface appears. The user is given three options to choose from. If the user selects choice 1, they will be directed to the admin menu. If the user selects choice 2, they will be directed to the cashier menu. For choice 3, the program will exit and stop running. However, if the user enters an input that is not among the given choices, it will be considered invalid, and the program will prompt the user to choose again from the three available options.

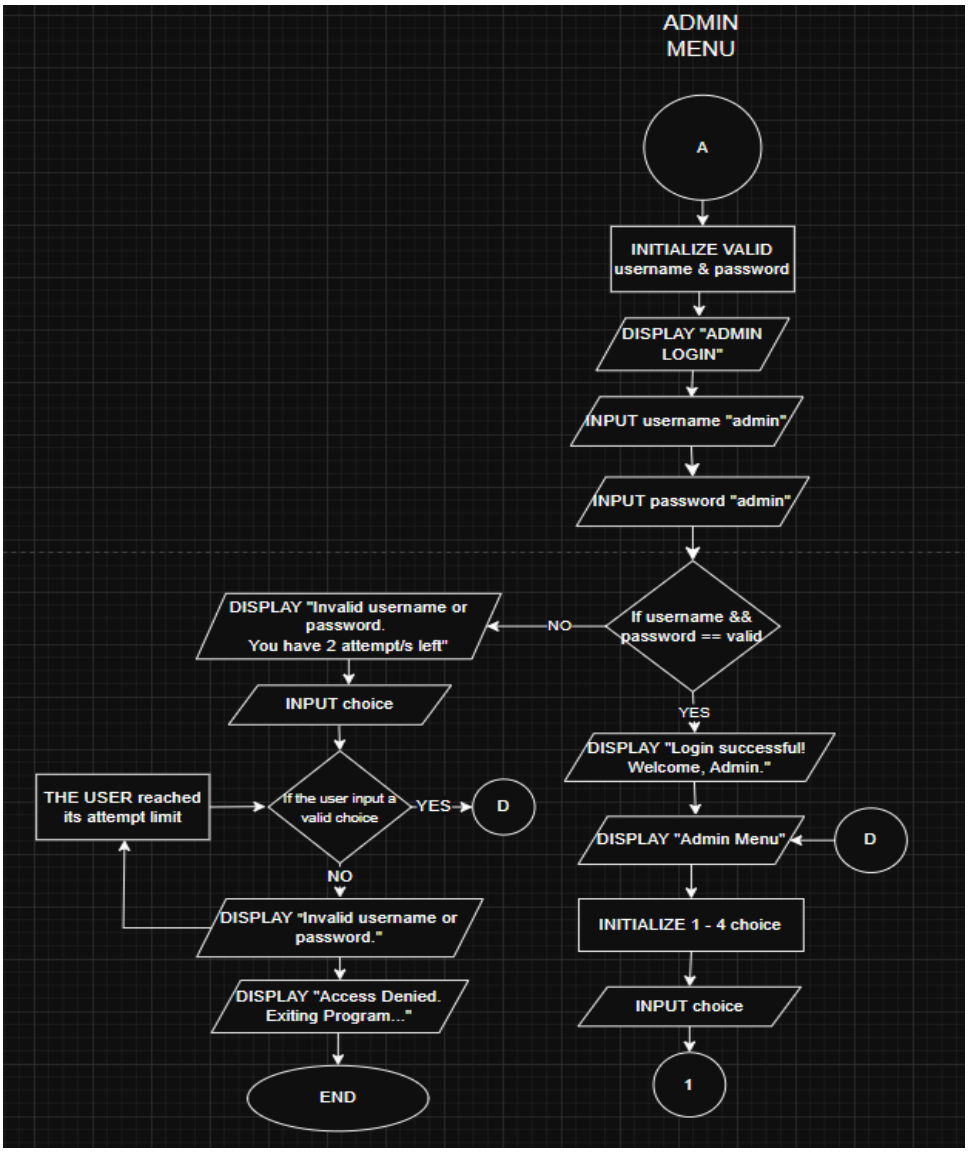

Figure 2: ADMIN MENU

In the second figure, the first step is the admin login, which requires entering a valid username and password. The system checks if the provided username and password are correct. If not, the system will display an invalid message and prompt the user to enter a valid username and password. If the user fails to enter the correct login information thrice, access will be denied, and the program will display "Access Denied. Exiting Program...". However, if the user enters valid credentials, the system will display "Login successful, Welcome, Admin." and proceed to show the admin section options, where the user can make a selection similar to the cashier section.

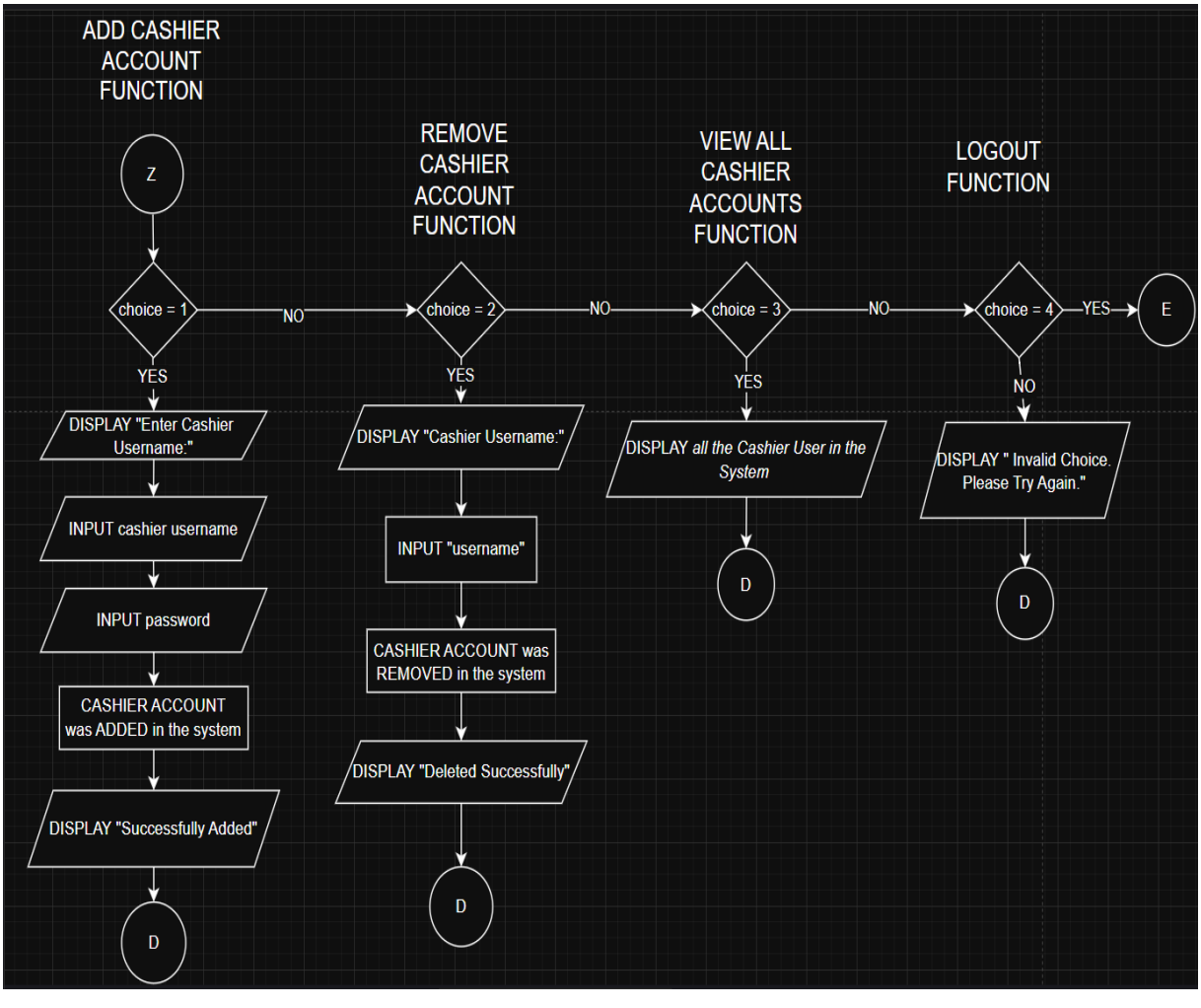

Figure 3: ADMIN SECTION SYSTEM

In this figure, there are four sections. The first section allows the user to add a cashier account, which requires entering a cashier's username and password. Once completed, the program will display "Successfully Added." The second option is to delete a cashier account, which only requires the cashier's username and will display "Successfully Deleted" once done. The third section allows the admin to view all cashier users in the system to check if any accounts have been added or removed. If the user enters all the required information correctly in any of these three sections, the system will automatically return to the admin menu for further actions if needed. Lastly, if the user chooses to log out from the admin section, the system will return to the main menu. However, if the user enters an invalid choice, the program will display "Invalid Choice, Please Try Again." and return to the Admin Menu Section.

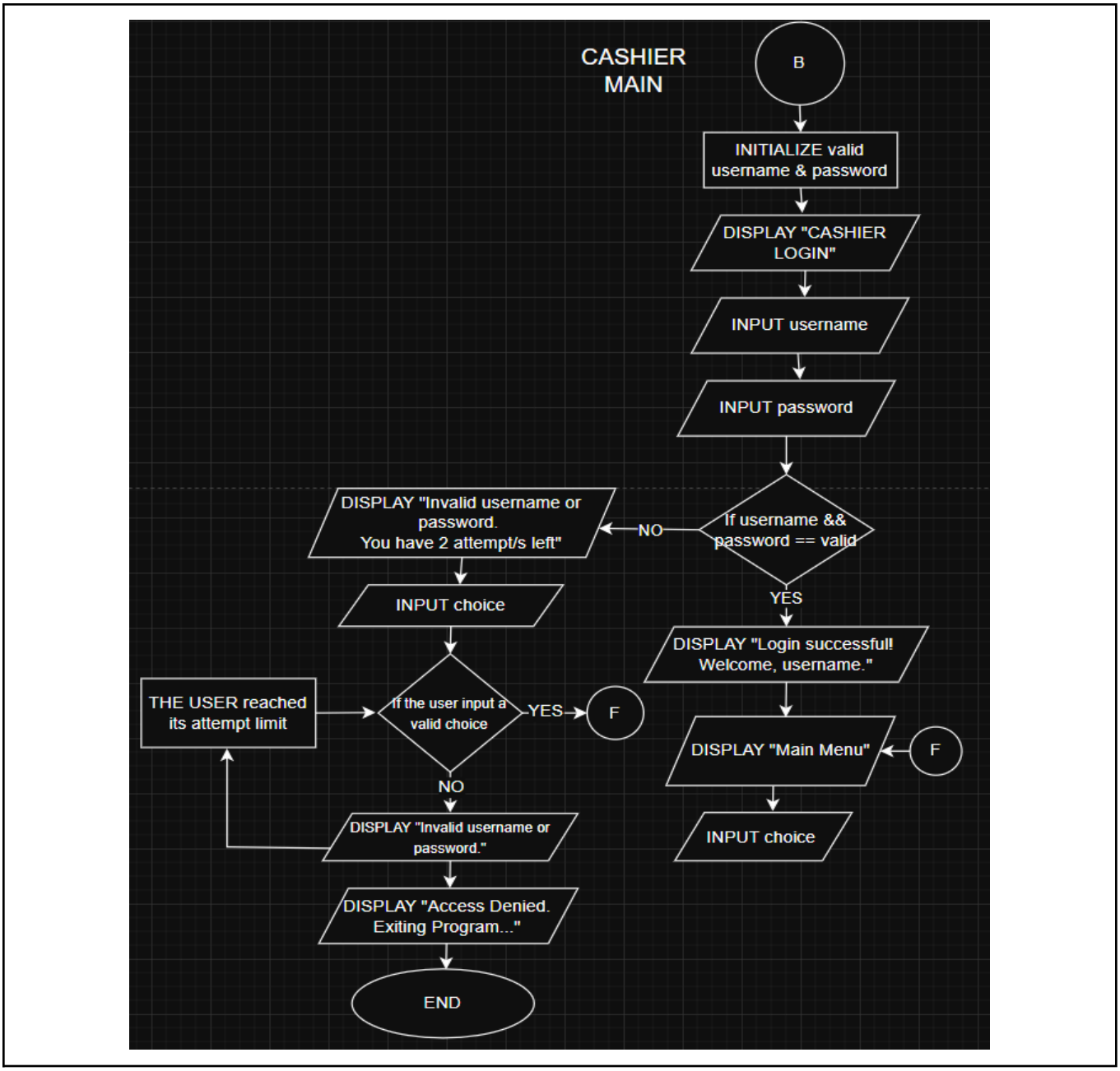

Figure 4: CASHIER MENU

In this figure, the first step is the cashier login, which requires entering a valid username and password. The system checks if the provided username and password are correct. If not, the system will display an invalid message and prompt the user to enter a valid username and password. If the user fails to enter the correct login information thrice, access will be denied, and the program will display "Access Denied. Exiting Program...". However, if the user enters valid credentials, the system will display "Login successful, Welcome, (cashier username)." and proceed to show the cashier section options, where the user can make a selection.

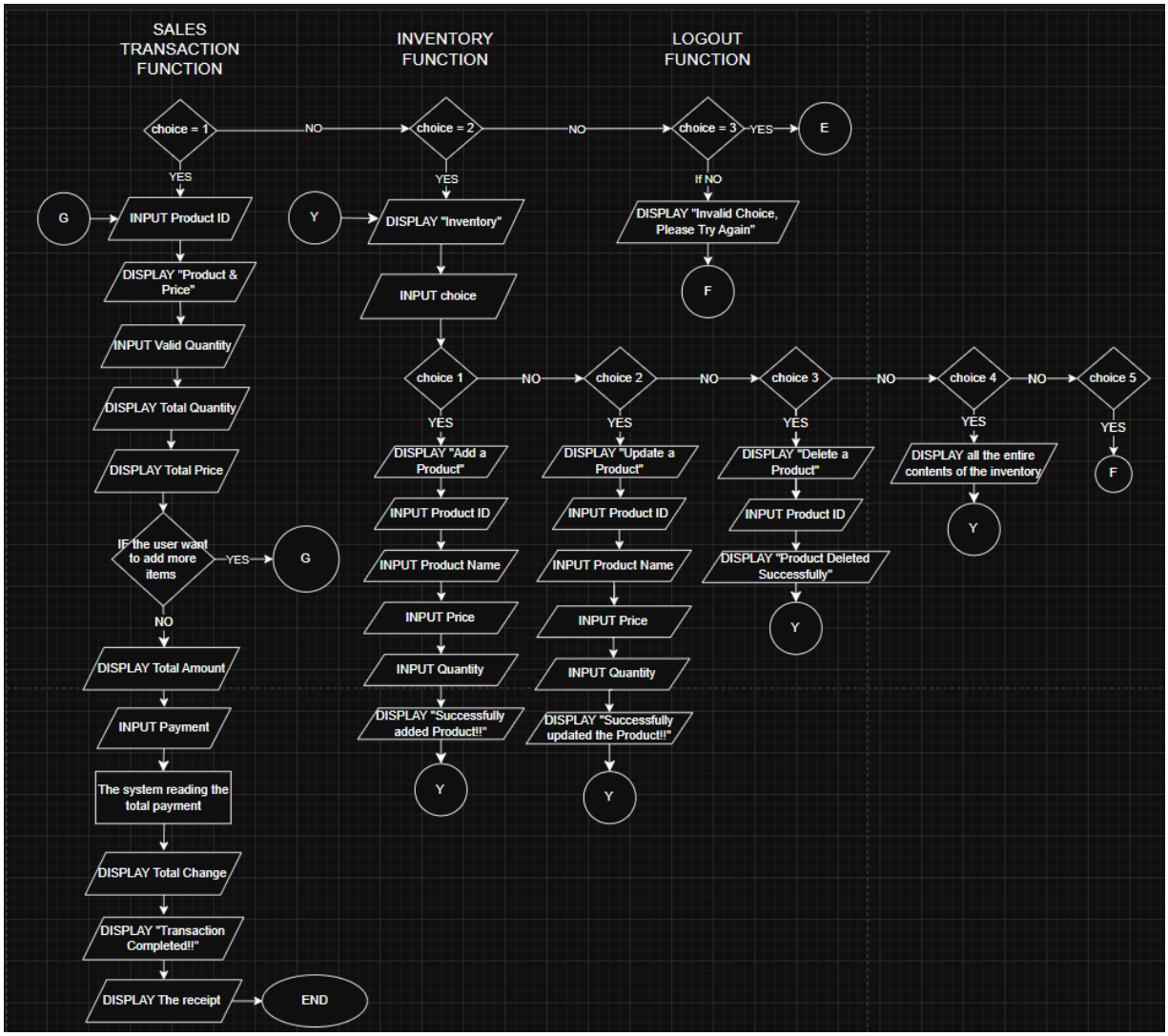

Figure 5: CASHIER SECTION SYSTEM

In this Figure, the cashier main section contains three parts. The first part is the sales transaction, where the user must enter the product ID, which serves as the identification of a product in the system. Once the product ID is entered, the product name and price will appear based on the data stored in the system. After that, the user must input a valid quantity of the product, and the total quantity and price will be shown. Then, a message will appear asking, "Add more items? (y/n):" where the user can choose to continue or stop. If the user selects yes, the process will repeat. If the user selects no, the process will stop, and the total amount will be displayed. Afterward, the user must enter the payment amount, and the system will process the transaction to show the total change. A message will then appear saying, "Transaction Completed!!" and the system will provide a receipt once the transaction is finished.

The second part is the inventory section, where the user has five choices. The first choice allows the user to add a product by entering the Product ID, Product Name, Price, and Quantity. Once all required

information is entered, the system will display "Successfully added Product!!". The second choice allows the user to update an existing product in the inventory by entering the Product ID and then providing the New product ID, Product Name, Price, and Quantity as desired. After the update, the system will display "Successfully updated the Product!!". The third choice is to delete a product from the inventory. The user must enter the Product ID, and once completed, the system will show "Product Deleted Successfully". The fourth choice displays all the Added, Updated, and Deleted products in the entire Inventory System. When the user correctly enters all the necessary information in any of the choices, the system will automatically return to the Inventory Section, including when choice 5 is selected.

Pseudocode

This subsection shows Figures 6 and 7 that represents the Admin or cashier Login and Management system of the transaction tracking and management system. This pseudocode lets the user visualize how the log in system for both admin and cashier works. The admin menu shows how the admin can add, remove, or view all cashier accounts. On the other hand the cashier menu demonstrates how transactions work. The cashier can check and update the stock, print receipts using notepad after each transaction. The cashier can also add, update, or view products in the inventory. After that the admin or cashier can go back to each menu and exit the program or execute another command.

```
SET correctUsername = [ “hanz”, “kevin”, “hail”,  “chris” ]
SET correctPassword = {“123”,”234”,”345”,”456”}
SET adminUsername = “admin”
SET adminPassword = “admin”

Login:
OUTPUT “- - - - - Welcome to Transaction Tracking and
Management System - - - - -”
OUTPUT "- - - - - LOGIN - - - - -"
OUTPUT “1. Admin”
OUTPUT “3. Exit”
INPUT 1

    IF userChoice = 1 THEN
        OUTPUT “Enter Admin Username: “
        INPUT correctAdminusername
        OUTPUT “Enter Admin Password: ”
        INPUT correctAdminpassword
            IF correctAdminusername = adminUsername AND
correctAdminpassword = adminPassword THEN
        GOTO AdminMenu
    ELSE
        OUTPUT “Invalid username or password”
    END IF

    ELSE
        OUTPUT “Exiting program…”
    END IF

AdminMenu:
REPEAT
    OUTPUT “- - - - - Admin Menu - - - - -”
    OUTPUT "1. “Add Cashier Accounts"
    OUTPUT "2. “Remove Cashier Accounts"
    OUTPUT "3. “View ALL Cashier Accounts"
```

```
    OUTPUT "4. "Log Out"
    OUTPUT "Enter Choice:"
    INPUT adminChoice

    IF adminChoice = 1 THEN
        OUTPUT "Enter Cashier Username: "
        INPUT cashierUsername
        OUTPUT "Enter Password: "
        INPUT cashierPassword


    ELSE IF
        adminChoice = 2 THEN
        OUTPUT "Enter Cashier Username: "
        INPUT cashier
        REMOVE cashier from correctUsername
        OUTPUT "Deleted Successfully"


    ELSE IF adminChoice = 3 THEN
        DISPLAY all correctUsername


    END IF
UNTIL adminChoice = 4
GOTO Login


OUTPUT "Exiting program…"

END PROGRAM
```

Figure 6: Pseudocode of Admin Login and Menu System.

The Figure 6 presents how the system first stores a coded username password for the admin and the cashiers then shows a greetings and the login menu for the user. After that the user should choose the admin option that will ask for the admin username and password if the admin enters a wrong password or username it will display an error if the user chose 3 it will display exiting the program and will exit the program. Once logged in, the admin is shown a menu for adding a new cashier account, removing a cashier account, viewing all cashier accounts, and to log out of the system. If the user's choice is one the admin can create a new cashier account by entering a new password and username island it will be added as a value in correctUsername and correctPassword. If the user chooses 2 the user can remove a cashier account by entering the cashiers username it will remove it from the correctUsername and will remove its password from correctPassword. If the user enters 3 then it will display all active cashiers or the correctUsername. All this step will be repeated until the users enter number 4 which is to logout and go

back to the login menu. Once the user logs out it can choose an option again if to exit or to login as an admin, if it chose 3 it will display exiting the program and will exit the program.

```
SET correctUsername = [ “hanz”, “kevin”, “hail”,  “christian” ]
SET correctPassword = {“123”,”234”,”345”,”456”}
SET inventoryCount = 0
SET totalProducts = 0

Login:
OUTPUT “- - - - - Welcome to Transaction Tracking and
Management System - - - - -”
OUTPUT "- - - - - LOGIN - - - - -"
OUTPUT “2. Cashier”
OUTPUT “3. Exit”
INPUT 2

ELSE IF choice = 2 THEN
    OUTPUT “Enter Cashier username: “
    INPUT cashierUsername
    OUTPUT “Enter Cashier password:”
    INPUT cashierPassword
    IF cashierUsername = correctUsername AND cashierPassword =
correctPassword THEN
        GOTO CashierMenu
    ELSE
        OUTPUT “Invalid username or password”
    END IF

ELSE
    OUTPUT “Exiting program…”
END IF


CashierMenu:
REPEAT
    OUTPUT ”- - - - - Main Menu - - - - -”
    OUTPUT ”1. Sales Transaction”
    OUTPUT ”2. Inventory”
    OUTPUT ”3. Log Out”
    OUTPUT “Enter Choice:”
    INPUT cashierChoice

 IF cashierChoice = 1 THEN
    REPEAT
        OUTPUT “Enter Product ID: ”
        INPUT productId
```

```
        IF productId found THEN
            OUTPUT “Enter Quantity:”
            INPUT quantity
            IF quantity <= stock THEN
                total = total + price * quantity
                stock = stock - quantity
        ELSE
            OUTPUT "Insufficient Stocks.”
        END IF
    ELSE
            OUTPUT “Product not found”
        END IF
        OUTPUT “Add more Items? (y/n):”
        INPUT addMore
        IF addMORE = “n” THEN
            PRINT RECEIPT ON NOTEPAD
        END IF

InventoryMenu:
REPEAT
    OUTPUT ”- - - - - Inventory - - - - -”
    OUTPUT "1. Add a Product"
    OUTPUT "2. Update a Product"
    OUTPUT "3. Delete a Product"
    OUTPUT "4. View Inventory"
    OUTPUT "5.  Exit”
    OUTPUT “Enter Choice:“
    INPUT choice


    IF choice = 1 THEN
        OUTPUT ”Enter Product ID:”
        INPUT id
        OUTPUT ”Enter product name:”
        INPUT name
        OUTPUT ”Enter price:”
        INPUT price
        OUTPUT ”Enter quantity:”
        INPUT quantity
        ADD to inventory
        OUTPUT "Successfully added product!!"


    ELSE IF choice = 2 THEN
        OUTPUT “Enter Product ID:”
        INPUT productId
```

```
        OUTPUT ”Enter New Product Name:”
        INPUT newName
        OUTPUT ”Enter New Price:”
        INPUT newPrice
        OUTPUT ”Enter New Quantity:”
        INPUT newQuantity
        OUTPUT "Successfully updated the product!!"


    ELSE IF choice = 3 THEN
        OUTPUT “Enter Product ID:”
        INPUT productId
        OUTPUT "Product Deleted Successfully"


    ELSE IF choice = 4 THEN
        DISPLAY  all Inventory products
    ELSE
        OUTPUT”Invalid choice”
UNTIL choice = 5
GOTO CashierMenu

OUTPUT “Exiting program…”

END PROGRAM
```

Figure 7: Pseudocode of Cashier Login and Menu System.

The figure 7 presents the list of cashiers and passwords and sets the total inventory and product count to 0. After that the menu shows the cashier option for the user to pick 2 and it will directly take the owner to the cashier login system but if the user picks 3 it will exit the program. The cashier login system will let the user enter the cashier username and password if the user entered a wrong password or username the system will print an error but if not it will take the user to the cashier menu. The cashier menu will have 3 things which are sales transaction, inventory and logout. If the user enters 1 then it will let the user enter the id of the product but if not found it will print product not found but if the entered product is correct it will automatically tell the name of the product and it will let the user enter the quantity of the product needed and if the quantity needed is less than the stock it will print insufficient stock but if the stock is greater than the quantity it will be a successful transaction and will ask if the user wants to add a product if not it will automatically print the receipt through notepad. Then if the user enters 2 the system will take it to the inventory menu which a user can add, update, delete, view products or exit the program. In the Inventory menu if the user chose 1 the user can then add a product and let the user enter the product id, name, price, and quantity. If the user chooses 2 the user can then update a products name, price, quantity by entering the product id. If the user enters 3 the user can delete a product by entering its ID. If the user enters 4 the program will then display all inventory products and lastly if the user enters 5 the user will be directed to the cashier menu, there if the user enters 3 it will exit the program.

Data Dictionary

The data dictionary provides a brief description of the variables that are needed for the system to function. It includes name, size, data type and brief description of the arrays, variables, constants, and structure used throughout the system. This can be used as a reference guide for other programmers and users to understand the variables used in the system.

Table 1 presents the list of global variables used in the system, including their name, size, data type, and a brief description. It includes variables needed for the login system, inventory management, and sales transactions. The table shows constant variables that are used for arrays to define the maximum amounts of products and user accounts, arrays the store user credentials and product information, and structure that contains the variables needed for each product.

Table 1: Data Dictionary

Data Name	Size	Data Type	Description
1. cashierAccount	4	const int	Constant number of possible cashier account for an array
2. activeacc	4	integer	Number of active cashier accounts
3. correctuser[ ]	240	string	Array that contains the usernames for the cashier accounts
4. correctpass [ ]	240	string	Array that contains the passwords for the cashier accounts
5. adminAccount	4	const int	Constant number of possible admin account for an array
6. adminUser [ ]	24	string	Array that contains the usernames for the admin account
7. adminPass [ ]	24	string	Array that contains the passwords for the admin account
8. no_items	4	const int	Constant number of possible items in the inventory for an array

9. stocks	4	integer	Number of active items in the inventory
10. quantity	4	integer	Number of a specific item a customer orders
11. change	4	integer	payment - Finaltotal
12. payment	4	integer	Stores the specific price
13. finaltotal	4	float	Final total price of what user bought
14. max_bought	4	const int	Number of possible items a user can buy for an array
15. no_bought	4	integer	Number of items a user buys
16. Product	36	struct	Structure to hold the information for each product
17. id	4	product : int	Member of the structure Product to hold the id for a product
18. name	24	product : string	Member of the structure Product to hold the name for a product
20. price	4	product : float	Member of the structure Product to hold the price for a product
21. quantity	4	product : int	Member of the structure Product to hold the quantity for a product
22. inventory [ ]	1800	product :	Structure variable to hold the information for each product in the

			inventory.
23. bought [ ]	720	product :	Structure variable to hold the information of a product that was bought

Code

This subsection presents the header files and the main function of the system. This includes all the functions needed for each feature of the system.The codes here shows how the system handles the login, transaction, inventory, receipt, and account management features. The cpp files for each of these header files can be found in the appendices.

```
#ifndef LOGIN_H
#define LOGIN_H

void greetUser(); //function declarations
int loginMenu();
bool cashierLogin();
bool adminLogin();

#endif
```

Figure 7. Login Header File

This figure shows the header file for the login menu, that contains the functions used for user authentication. The function greetUser just prints out a header to welcome the user to the system.

The function loginMenu that displays a menu that will let the user choose to log out or choose what kind of account to log into, either admin or cashier account. If the user inputs 1 it will call the adminLogin function inside an if structure that will run if adminLogin function returns false and will stop the program, else the loginMenu will return 1. It does the same thing when the user inputs 2 but calls the function cashierLogin instead. If the user inputs 3 it will stop the program.

The function adminLogin has a return type of bool that will return true if the username and password is valid and return false if the user inputs the wrong username or password three times. This can check if it is valid by comparing the user input username and password to the adminUser and adminPass array.

The function cashierLogin has a return type of bool that will return true if the username and password is valid and return false if the user inputs the wrong password three times. This can check if it is valid by using a for loop and if structure to compare the user input username and password to correctUser and correctPass array.

```
#ifndef ADMINMENU_H
#define ADMINMENU_H

void adminMenu(); //function declarations
void addAccount();
void removeAccount();

#endif
```

Figure 8. Admin Menu Header File

This figure shows the header file for the admin menu, this includes the functions used to manage cashier accounts. The void function adminMenu( ) displays the menu with available actions to manage cashier accounts that the administrator can choose from, this includes adding, removing, viewing of cashier accounts or logging out. This function also includes the feature to view all cashier accounts that just uses a for loop to print out all the elements in the correctUser and correctPass arrays. This function is inside a do-while loop to let the administrator choose when to log out.

The function addAccount( ) lets the administrator add a new cashier account and set their password. The function starts with a for loop that initializes to equal the active account to put the new cashier account to the next slot in the array. Inside the for loop the administrator is asked to input a username and that username will be checked to see if it's new using a for loop and if structure to compare it to the current usernames stored at correctUser[ ]. This part is in a do-while loop that will stop once the username is not taken and will store that username in the correctUser array. It then prompts the administrator to enter a password, which will be stored in the corresponding index of the correctPass array.

The function removeAccount( ) lets the administrator remove an existing cashier account. The administrator is prompted to input a username to know what account to remove and to check the currentUser array using a for loop to know if the account exists. This part is in a do-while loop that will stop once a valid username is entered. Once the username is found in the array it will display the current account and password and will be asked for confirmation to delete the account. If the administrator enters ‘n’ it will display “Action canceled” and will go back to the admin menu but when they enter ‘y’ it will delete the account using a for loop and that moves all username and passwords up one position in the array after the deleted one and decrement the “activeacc” variable to full delete the account.

```
int main(int argc, char** argv) {
	greetUser();
	int logchoice = loginMenu();
	do{
		switch(logchoice){
			case 1:
				adminMenu();
				break;
			case 2:
```

```
                        int choice;
                        do{
                              cout << "- - - - - Main Menu - - - - - \n";
                              cout << "1. Sales Transaction \n";
                              cout << "2. Inventory \n";
                              cout << "3. Log Out \n\n";
                              cout << "Enter Choice: ";
                              cin >> choice;
                              switch(choice){
                                    case 1:
                                          system("cls");
                                          salesTransaction();
                                          break;
                                    case 2:
                                          system("cls");
                                          inventoryMenu();
                                          break;
                                    case 3:
                                          system("cls");
                                          break;
                                    default:
                                          system("cls");
                                          cout << "Invalid Choice, Please
Try Again. \n";
                              }
                        }while(choice !=3);
                        break;
            }
            logchoice = loginMenu();
      }while(logchoice !=3);

      return 0;
}
```

Figure 9. Main Function

This figure shows the main function of the system and the main menu for the cashier. It starts with calling the function greetUser to display a welcome message, and then gets the user's login choice by calling the function loginMenu that will return an integer either 1, 2 or 3 that will be used for the switch case. If the loginMenu function returns 1 it will go to the adminMenu function that lets you manage cashier accounts. If it returns 2 it will display the main menu that the cashiers can choose from and will call out the corresponding functions, this includes sales transaction, inventory, or log out. All of these are in a do- while loop that will exit the program once the loginMenu returns 3, which is the "Exit". Moreover, the global variables are defined here so they can be used by different cpp files.

```
#ifndef SALESTRANSACTION_H
#define SALESTRANSACTION_H

void salesTransaction(); //function declaration


#endif
```

Figure 10. Sales Transaction Header File

This figure shows the header file for the sales transaction feature. The function salesTransaction starts with an if structure that checks if there is stock in the inventory and proceeds if there is stock and returns to the main menu when “stocks” is equal to 0. It will then enter a do-while loop that will allow for multiple items to be purchased and inside the loop it will ask for the id of the product that was bought and searches the inventory using a for loop and if structure to check if the product is in the inventory. If it is found in the inventory it will display the product name and its corresponding price, and the cashier will be prompted to enter the quantity that will be bought. This will then be checked if there is enough stock in the inventory. If there is not enough stock in the inventory it will print out the remaining stock and will be prompted if you want to add another item. If there is sufficient stock the information in the structure variable inventory[i] will be copied in the structure variable bought[no_bought] that will be used for printing the receipt and it will update the inventory. Then the total will be displayed and the cashier will be prompted if they want to add more items. Once the loop ends it will display the total amount and will be asked for the payment amount. Finally, it will call for the function printReceipt to display all the product bought, total amount, and change in notepad.

```
#ifndef INVENTORY_H
#define INVENTORY_H
#include <iostream>
using namespace std;

struct Product{
	int id;
	string name;
	float price;
	int quantity;
};

extern const int cashierAccount;
extern int activeacc;
extern string correctUser[];
extern string correctPass[];
```

```
extern const int adminAccount;
extern string adminUser[];
extern string adminPass [];

extern const int size;
extern Product inventory[];
extern int stocks;

extern int quantity,change, payment;
extern float finaltotal;
extern const int max_bought ;
extern Product bought[];
extern int no_bought;

void inventoryMenu();
void productAdd();
void productUpdate();
void deleteProduct();
void viewInventory();

#endif
```

Figure 11. Inventory Header File and Global Variables

This figure shows the header file for the inventory that contains all the functions that are needed for inventory management as well as the global variables that are used throughout the system. The global variables are declared here using “extern” in order to use the global variables throughout the different cpp files by only including the inventory header file. The function inventoryMenu displays the menu with available actions to manage the inventory that includes adding, updating, deleting, and viewing products in the inventory that depends on the user input.

The function productAdd allows the cashier to add products in the inventory. This uses a for loop that will input the new product and its corresponding name, price, and quantity to a new slot in the inventory array if the id doesn’t already exist.

The function productUpdate allows the cashier to update existing products in the inventory allowing them to change its name, price, and quantity.

The function deleteProduct allows the cashier to delete existing products in the inventory by searching for its id. This uses a for loop that moves all items up one position in the array after the deleted one and decrements the “stocks” variable to remove the duplicate/last element to delete a product from the inventory.

The function viewInventory allows the cashier to view all the existing products in the inventory using a for loop to print all the elements in the array.

```
#ifndef RECEIPT_H
#define RECEIPT_H

void printReceipt(); //function declaration

#endif
```

Figure 12. Receipt Header File

This figure shows the header file for the receipt, and the void printReceipt(). The printReceipt() function is designed to document and display a sales transaction. It first opens a file named receipt.txt to write the details. The function then iterates through all purchased items, listing each product's ID, name, price, and quantity. After itemizing the products, it prints the Payment, Change and final Total, It prints " Transaction Completed !! " and uses a system command to open receipt.txt in the notepad. Once the user closes the receipt in the notepad, the program returns to the main menu.

Results and Discussion

This subsection presents the output of code by implementing each feature the system has and a discussion on what is being shown on each figure. This includes all the possible output for each feature, including the login, sales transaction, and inventory system.

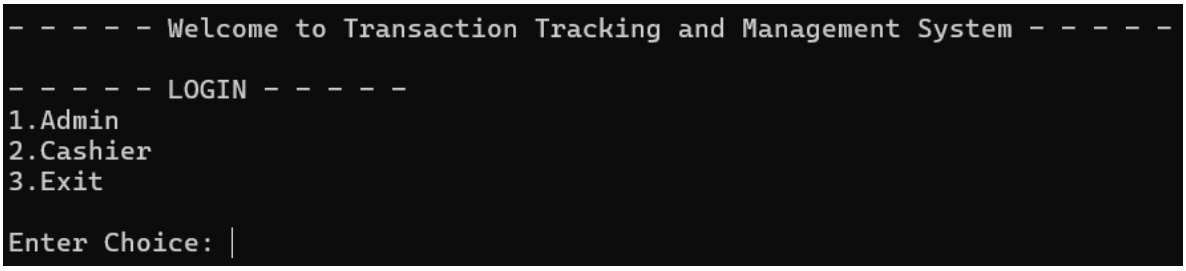

Figure 13. Welcome and Login Menu.

Figure 13 shows the welcome and login menu of the system. The user will be presented with three choices namely “1.Admin”, “2.Cashier”, and “3.Exit” and asked to input a number from 1 to 3.

Figure 14. Admin Login.

Figure 14 shows the admin login, The user will input username and password to direct in the admin menu dashboard. If the password or username is incorrect, the number of attempts you can log in will decrease by one.

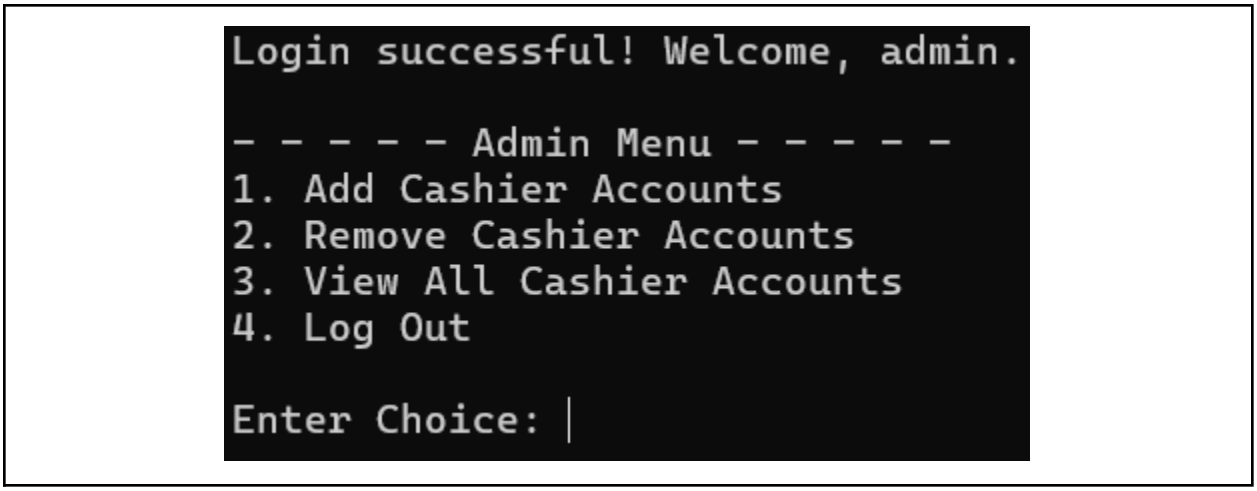

Figure 15. Admin Menu.

Figure 15 shows the admin dashboard where the user will input their choice, there's 4 choice namely 1.Add Cashier Account, 2.Remove Cashier Account, 3.View All Cashier Accounts, and lastly 4.Logout

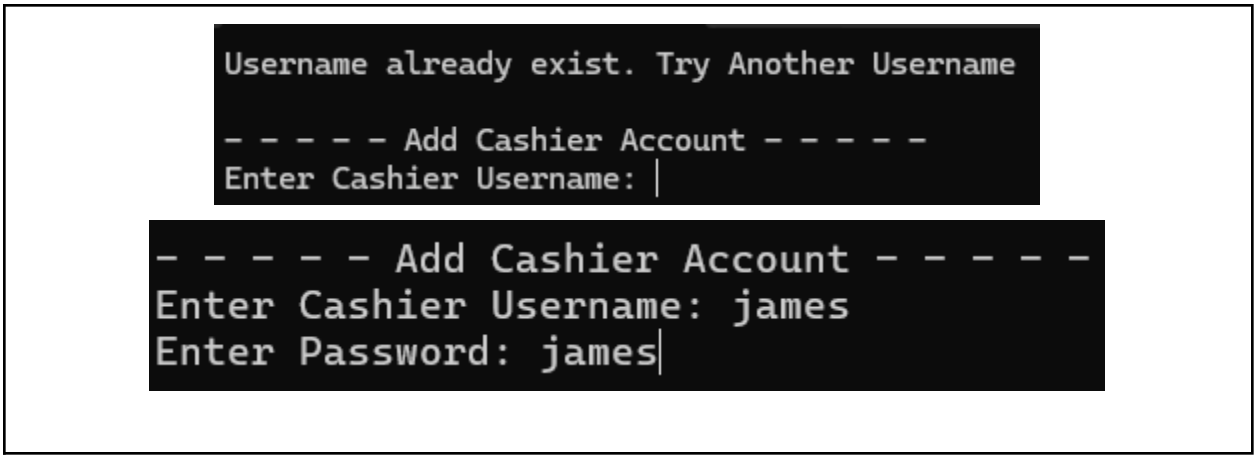

Figure 16. Add Cashier Accounts.

Figure 16 shows If the user inputs the first choice which is to add a cashier account the user will need to input a unique username for the new cashier account and the password for it.

.

Figure 17. Cashier Accounts

Figure shows that if the process is done correctly the program will display “Successfully Added Account”,Then it will go back to the Admin menu. Moreover, if the user inputs 3 it will show all the active cashiers and the one you recently added

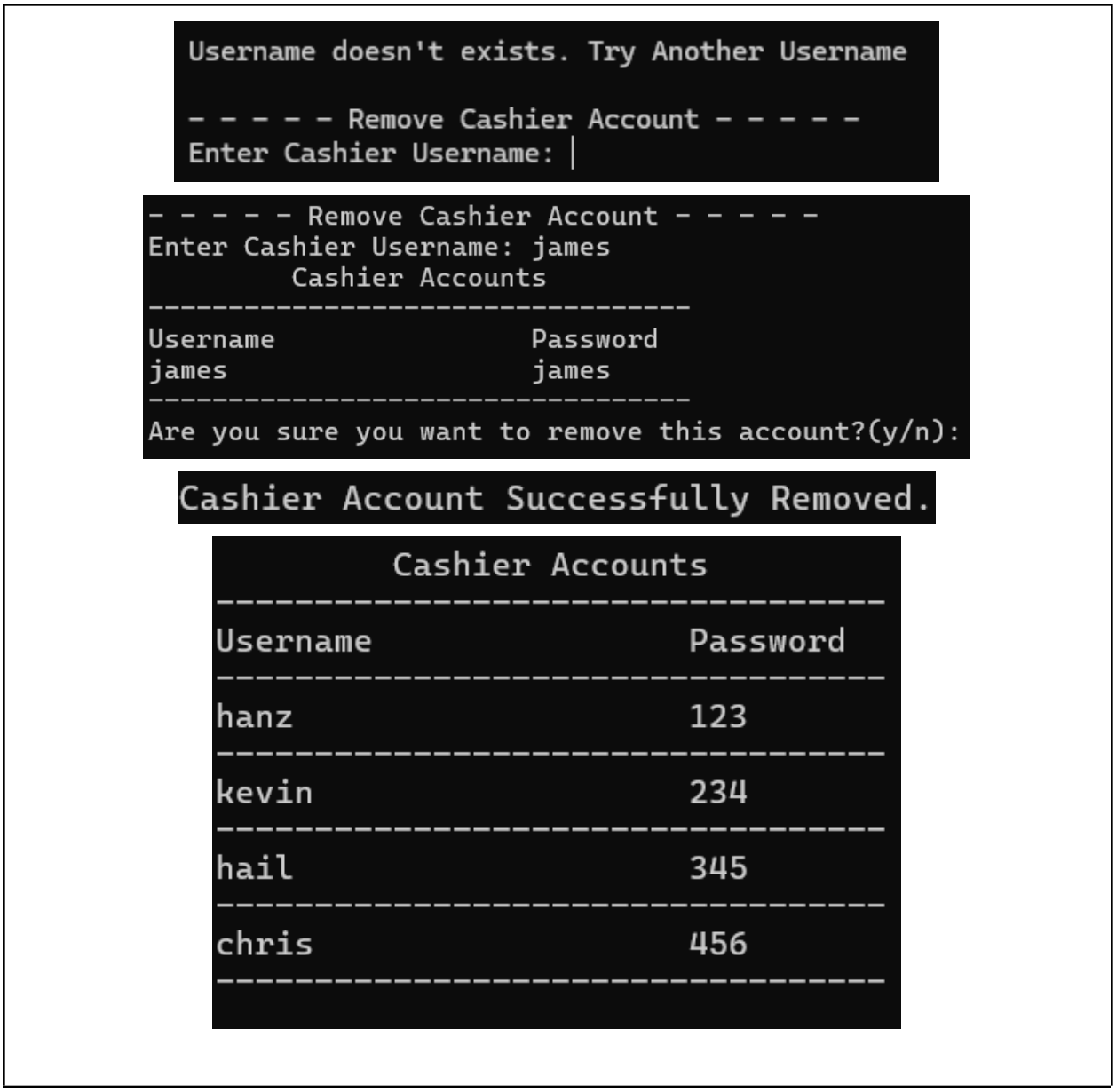

Figure 18. Remove Cashier Account.

Figure 18 shows the interface for removing accounts that shows when the user inputs 2 in the admin menu and the result afterwards. The user must enter an existing cashier account username to remove it, the user will be asked to choose yes or no, yes if you want to remove the account, or no if you want to cancel.

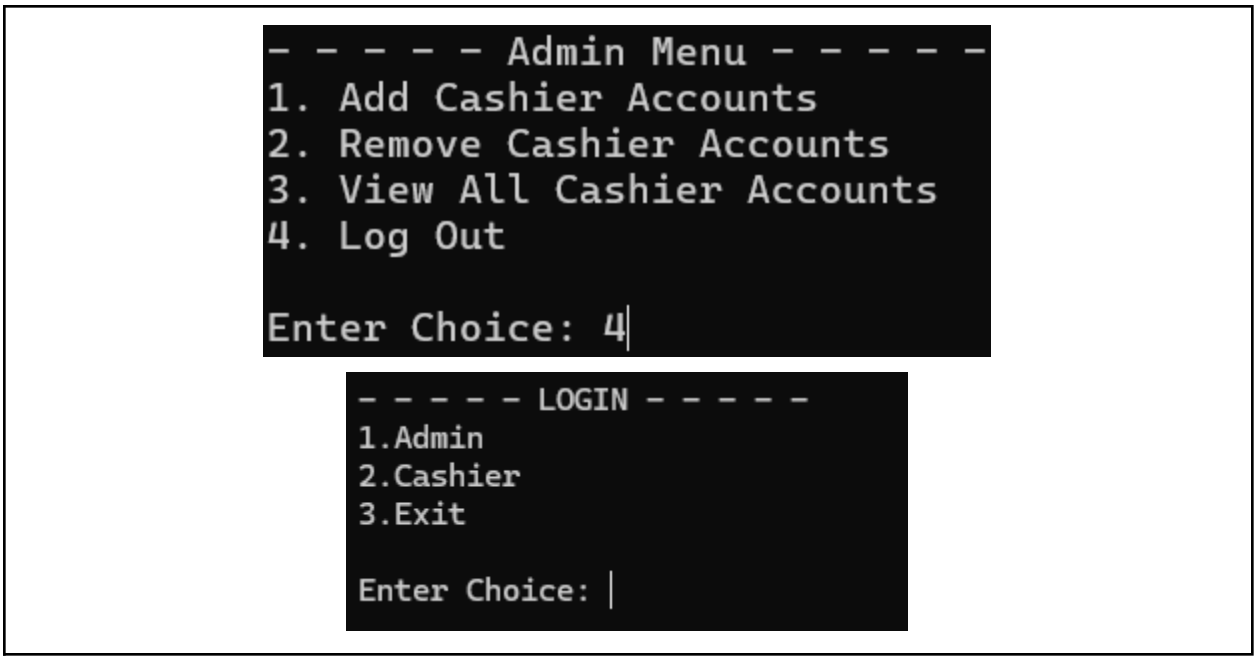

Figure 19. Logout.

Figure 19 shows that if the user inputs 4 the program logouts the admin account and goes back to the primary interface or the login menu.

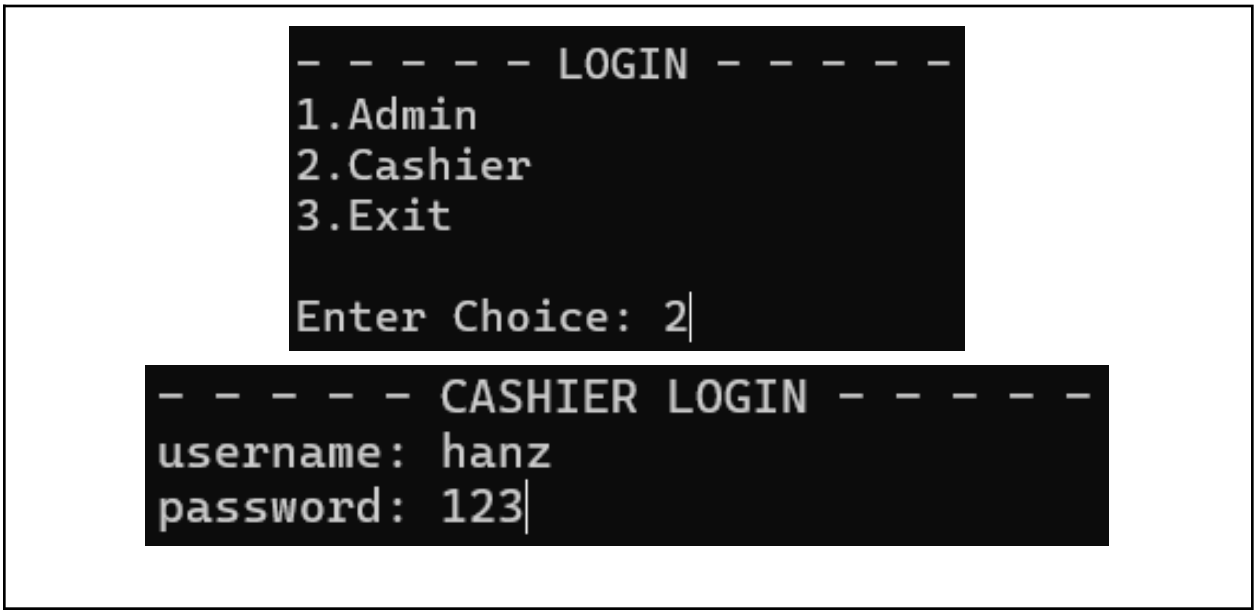

Figure 20. Cashier Login.

Figure 20 demonstrates the login process for the cashier, where the user is required to provide the correct username and password in order to proceed to the main menu.

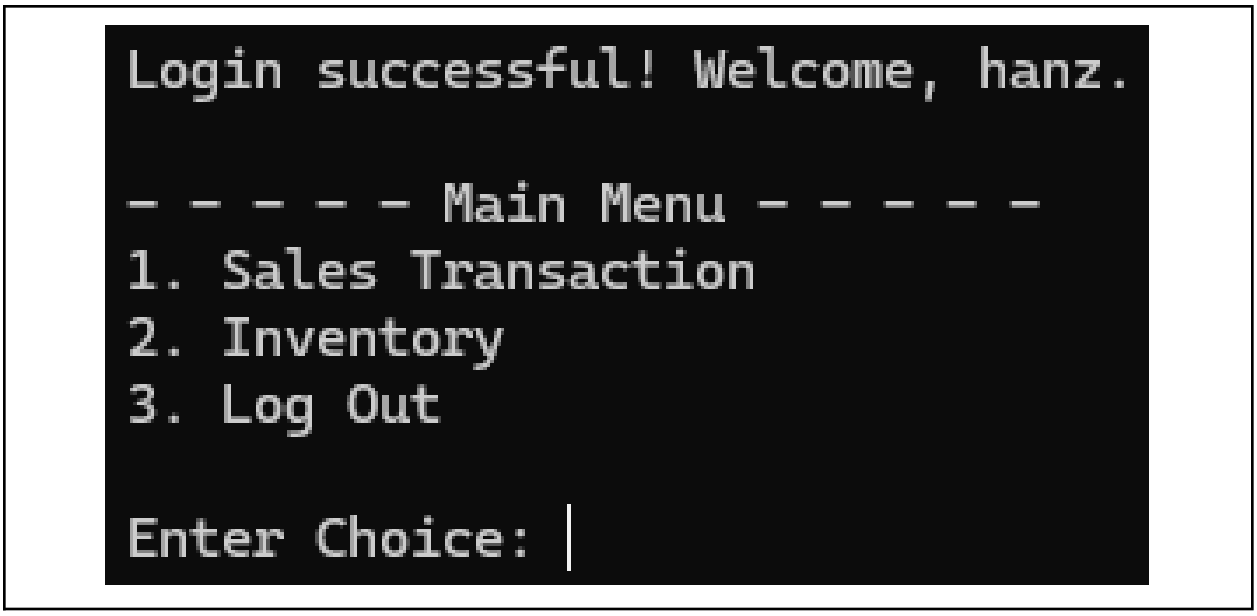

Figure 21. Main Menu.

Figure 21 shows the output “Login successful! Welcome, (name of the cashier)” if it's done correctly. After that the main menu for the cashier will show up. The user will be presented with three choices namely “1.Sales Transaction”, “2. Inventory”, and “3.Log Out” and asked to input a number from 1 to 3.

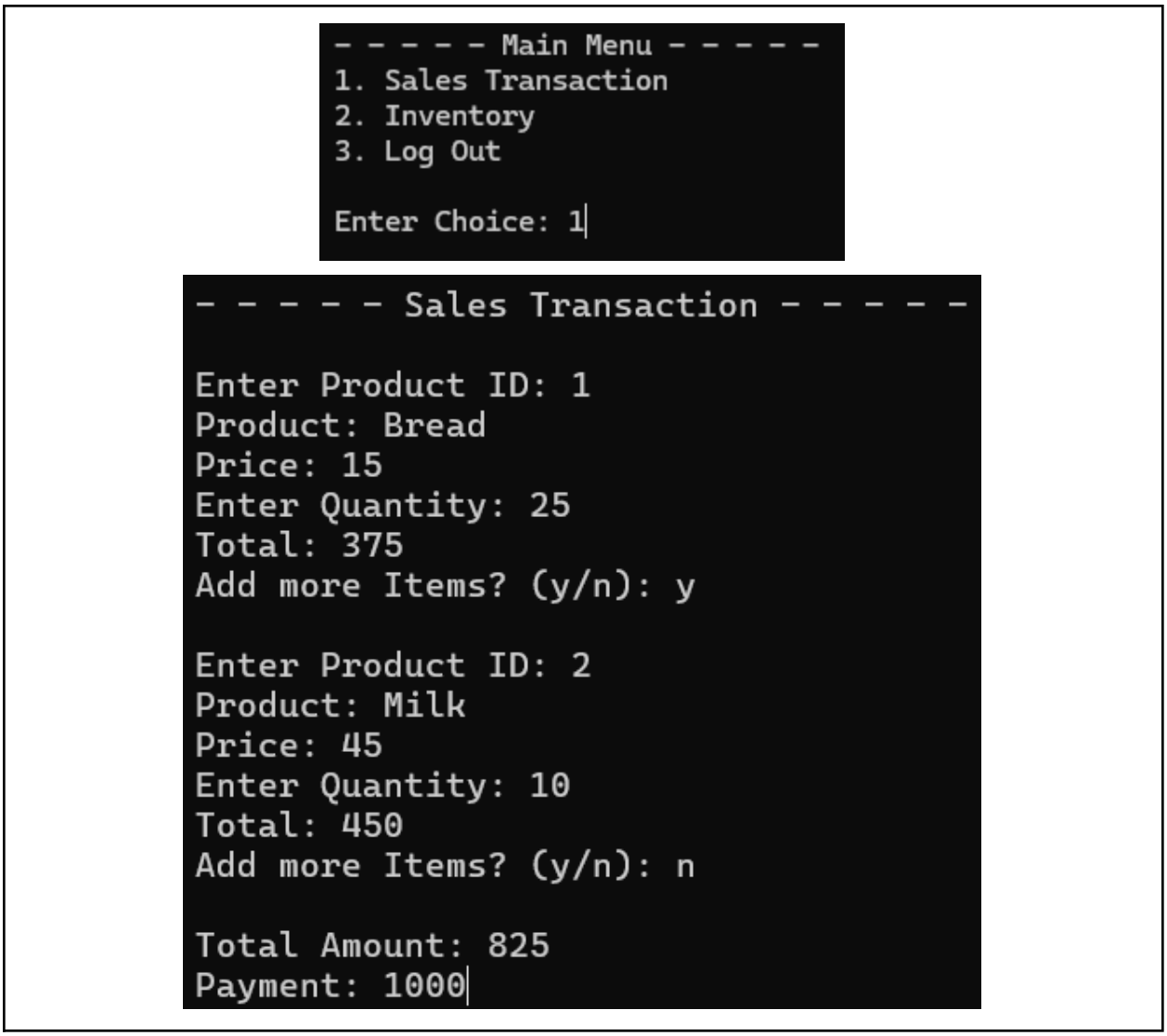

Figure 22 Sales Transaction.

Figure 2 shows that if the user chooses 1 the program will directly lead it to a sales transaction in which the user will enter the ID of the product and it will automatically tell the product name and price, then the user will be asked the quantity the customer will buy.This will loop until the user inputs ‘n’. The program will display the total amount and ask the user to input the payment of the customer then it will calculate the change.

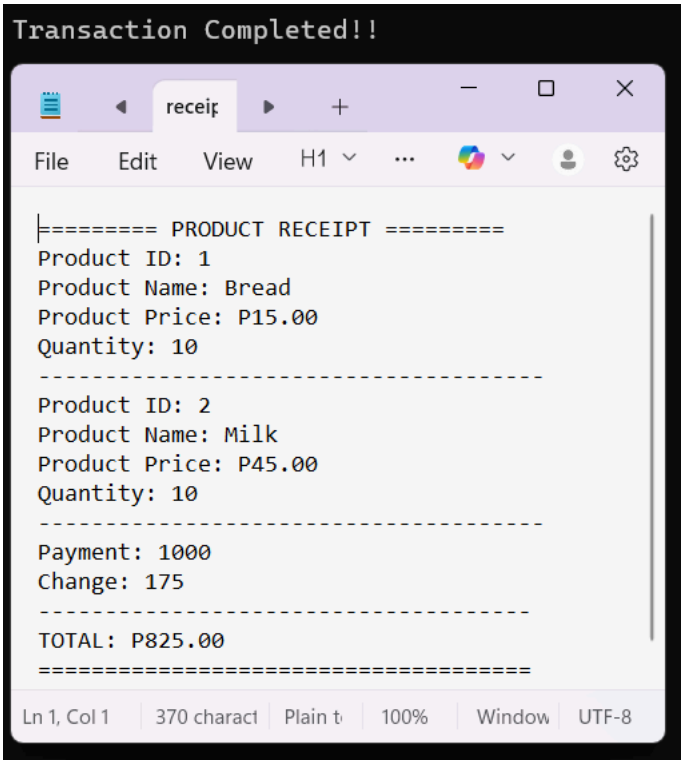

Figure 23. Printed Receipt

Figure 23 shows that after completing the transaction, the program automatically generates and prints a receipt through Notepad. This includes all the products bought, total, payment, and change.

Figure 15 The program will then print "Transaction completed!!" if the transaction is successfully done and will directly take the user back to the main menu, Then the user needs to enter number 2 to go to the inventory menu.

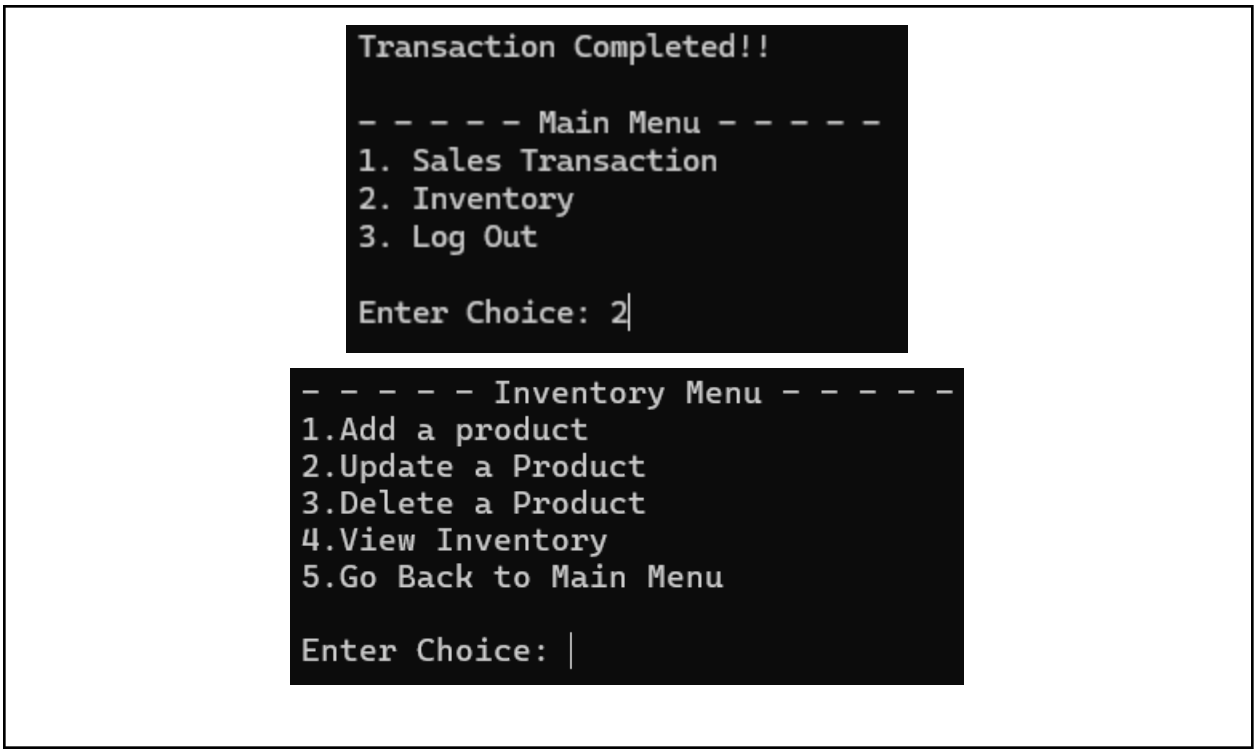

Figure 24. Inventory Menu.

Figure 24 shows that the program will then print "Transaction completed!!" If the transaction is successfully done and will directly take the user back to the main menu, then the user needs to enter number 2 to go to the inventory menu on which the user can add, update, delete products and view inventory.

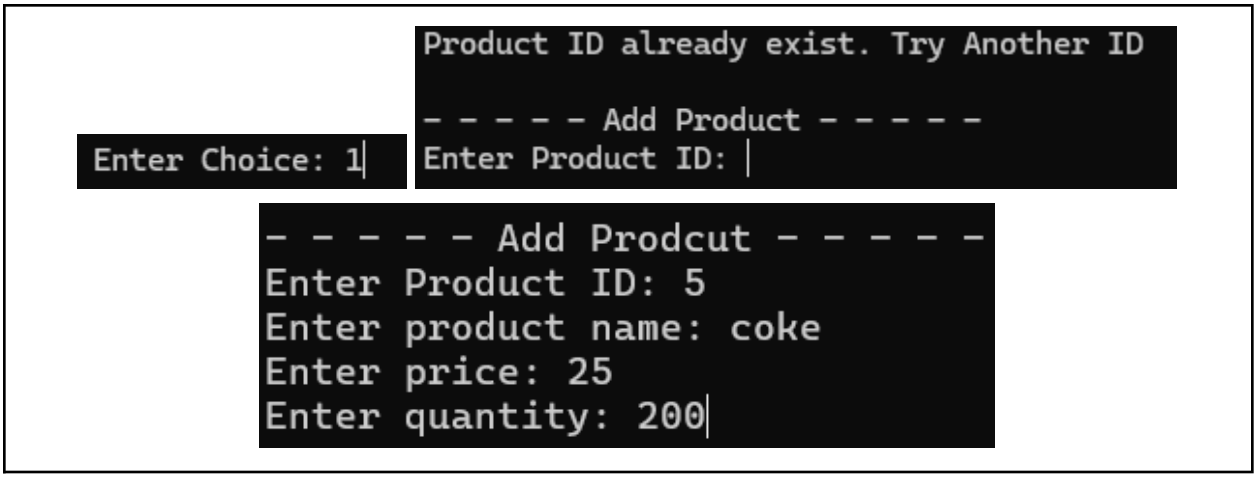

Figure 25. Add Product.

Figure 25 shows the add product feature that a user can access once they input 1 in the inventory menu. Enter a unique product ID and complete the required details, then click save to add the product.

Figure 26. Successfully Added Product.

Figure 26 shows that if done correctly the product will be saved and will display “Successfully added product !!”. If the user inputs 4 it will display the inventory containing all the current products including the newly added one with its corresponding id, name, price, and quantity

Figure 27 Update Product.

Figure 27 shows the update product interface and its result. It will prompt the user to input an existing product id. This will then show the current product that has not been updated yet and will ask the user for new information about the product. Moreover, the figure also shows the product after it has been successfully updated.

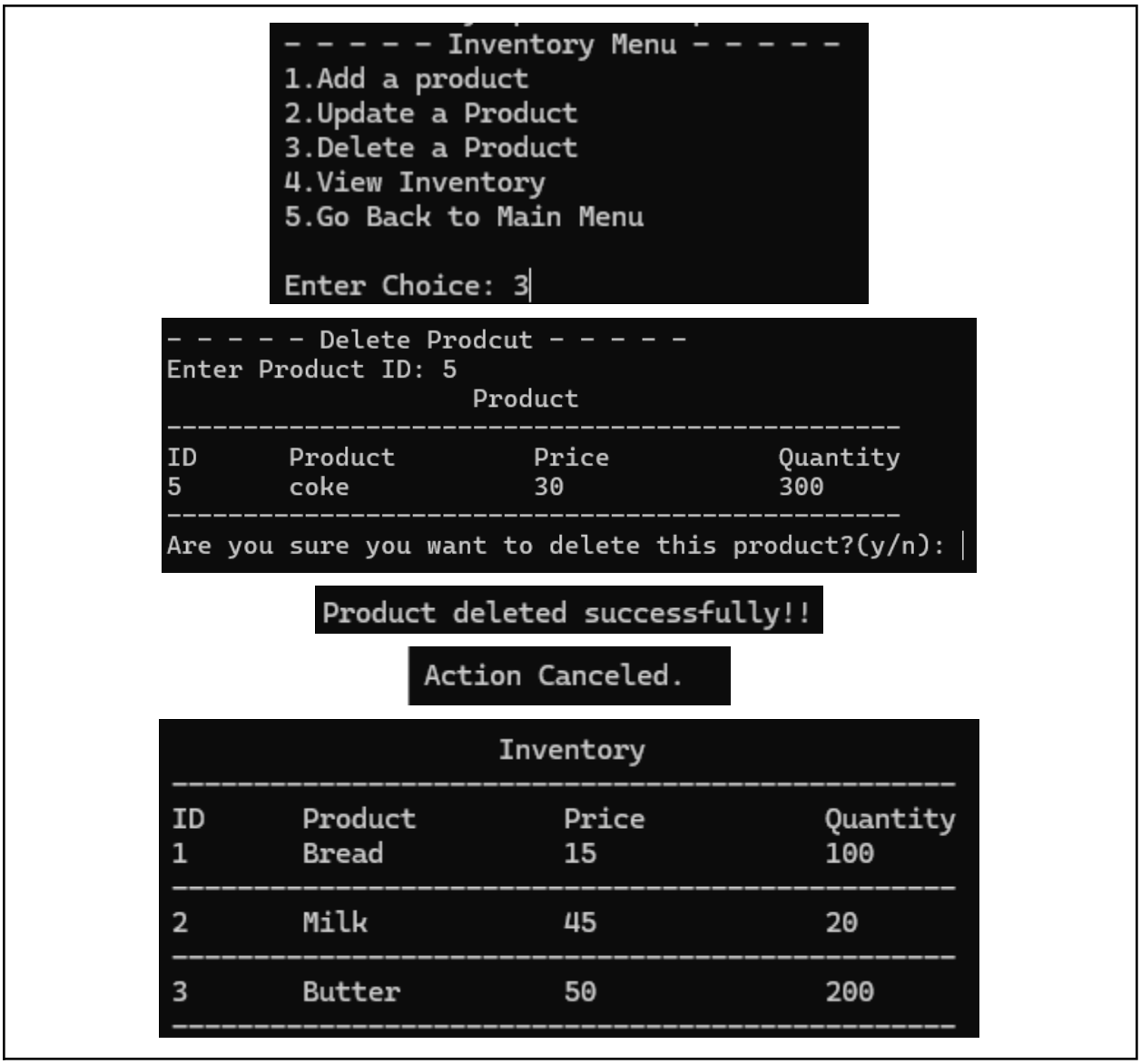

Figure 28. Delete Product.

Figure 28 shows the delete product interface. The user will be asked for the product ID that will be deleted. It will show the current product and then a prompt appears asking, “Are you sure you want to delete this product? (y/n).” If the user enters y, the product is deleted and a message saying “Product deleted successfully” is displayed. If the user enters n, the action is canceled, and the system returns to the Inventory menu. Moreover, the figure shows the updated inventory where it deleted the product.

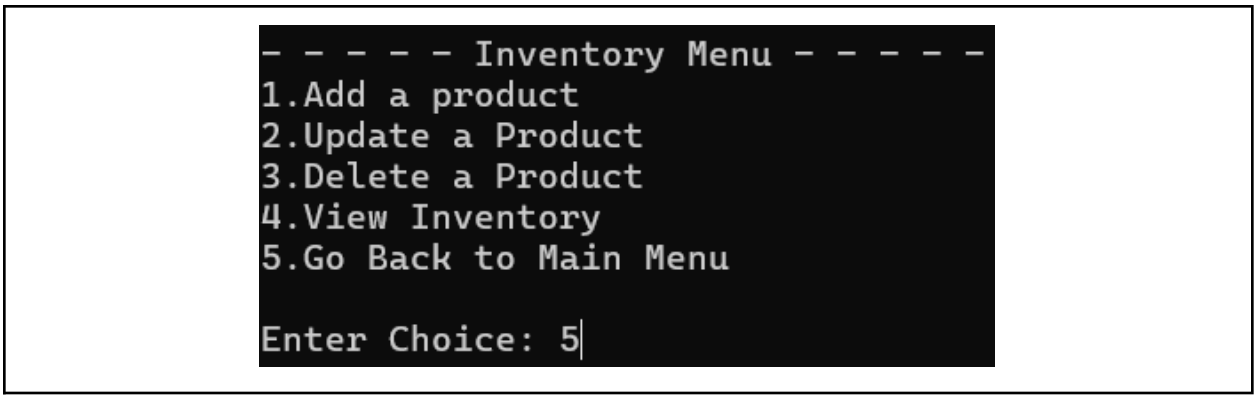

Figure 29. Back to the Main Menu.

Figure 29 shows that to return to the main menu, the user must enter the number 5. This action will navigate them back to the main menu, allowing them to access other available options and features.

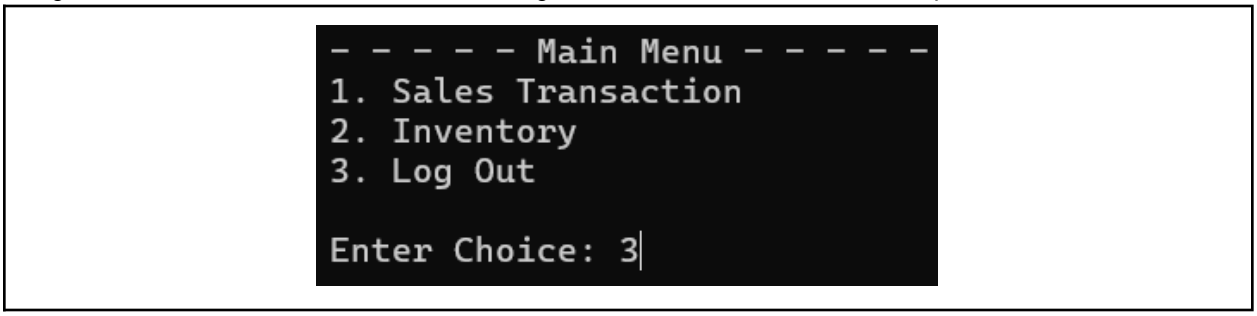

Figure 30. Main Menu and Logout.

Figure 30 shows that to logout of the cashier account and return to the login menu, the user is required to enter number 3.

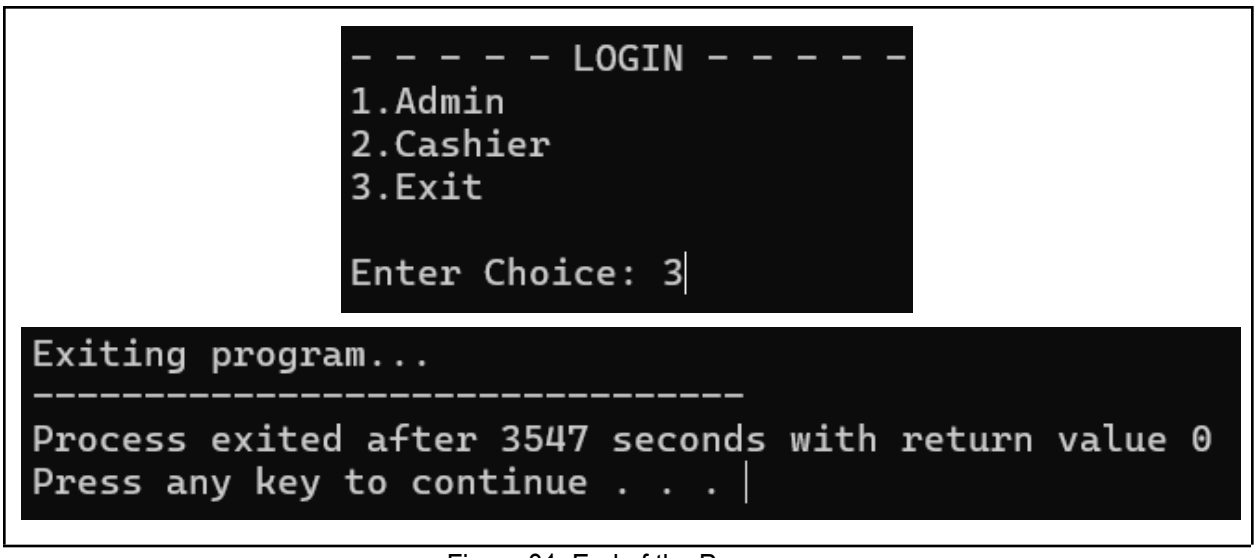

Figure 31. End of the Program.

Figure 31 shows the user is back at the primary interface, if the user inputs number 3 it will exit and close the program.

Conclusion

In conclusion the Transaction Tracking and Management System Project successfully achieves its main objective to help small business owners who still rely on manual and hand written records. The system includes important features such as a secure login access for both admin and cashier, account management, inventory management, sales transaction recording and receipt printing. This testing showed that the system can efficiently handle the transactions, accurate computation of totals and change, and can properly update the inventory section based on the user. All objectives of the project were achieved.
The system successfully provided user authentication for the account, simplified sales transactions, managed the product inventory, and generated receipts automatically after transaction. Overall, this project proved that even with a C++ program, small business operations can become more efficient and organized.
To further improve the system, future development on the system should include sales tracking and inventory shortage alert to improve data monitoring. Moreover, it should further develop a more structured Graphical User Interface (GUI) to improve the system's usability and make it more user friendly.

References

Gestisoft. (2023, April 5). How to avoid common inventory pitfalls with small business inventory management software. Gestisoft Blog. Avoid inventory pitfalls with small business management software | Gestisoft

Retalon. (2025, February 3). Common inventory management problems and how to solve them. Retalon Blog. Most Common Inventory Management Problems in 2025 (and Their Solutions) | Retalon

Wynn, J., & Kuhn, J. (2021). The financial impact of manual inventory record errors. International Journal of Business and Social Science, 12(10), 10–17. 2.pdf

Giddh Blog. (2025, September 22). 5 Reasons Why Small Businesses Prefer Bookkeeping Software. Giddh.5 Reasons Why Small Businesses Prefer Bookkeeping Software

Bookkeeper360. (n.d.). The Hidden Costs of Manual Bookkeeping and How Automation Can Save Your Business Thousands. Bookkeeper360.
The Hidden Costs of Manual Bookkeeping and How Automation Can Save Your Business Thousands | Bookkeeper360

Lee, K. L., et al. (2021). Elimination of Misconduct in Manual Counting Process as an Improvement of Inventory Accuracy in a Manufacturing Company. International Journal. (Proposal of automation to reduce human error)
(PDF) Elimination of Misconduct in Manual Counting Process as an Improvement of Inventory Accuracy in A Manufacturing Company

Appendices

Appendix A: login.cpp

```
#include "login.h"
#include "inventory.h"
#include <iostream>
#include <cstdlib>
using namespace std;

void greetUser(){
	cout << "- - - - - Welcome to Transaction Tracking and
		Management System - - - - -" << endl;
}

int loginMenu(){
	int choice;
	cout << "\n- - - - - LOGIN - - - - -\n";
	cout << "1.Admin \n";
	cout << "2.Cashier \n";
	cout << "3.Exit \n\n";
	cout << "Enter Choice: ";
	cin >> choice;
	system("cls");
	switch(choice){
		case 1:
			if (!adminLogin()) {
		cout << "\nAccess Denied. Exiting program...\n";
		exit(0);
			}else{
				return 1; //returns an integer for the switch
					case in the main menu
			}
			break;
		case 2:
			if (!cashierLogin()) {
		cout << "\nAccess Denied. Exiting program...\n";
		exit(0);
			}else{
				return 2; //returns an integer for the switch
					case in the main menu
```

```
                }
                break;
            case 3:
                cout << "Exiting program...";
                exit (0);
                break;
            default:
                system("cls");
                cout << "Invalid Choice, Please Try Again. \n";
                cout << "\n";
            }
}
bool adminLogin(){ //function returns false when all attempts are
                    used
    string username, password;
    int attempt = 3;

    do{
        cout<<"- - - - - ADMIN LOGIN - - - - -\n";
        cout<<"username: ";
        cin>>username;
        cout<<"password: ";
        cin>>password;
        attempt--;

        for(int i = 0; i < adminAccount; i++){
            if (username == adminUser[i] && password ==
                adminPass[i]) { //checks if both username and
password are valid
            system("cls");
            cout << "Login successful! Welcome, " << username
                 << ".  \n\n";
            return true;
                }
            }
        system("cls");
        cout << "Invalid username or password.\n";
        cout <<"You have "<< attempt<< " attempts left. \n\n";
     }while(attempt !=0);
```

```
    return false;
}

bool cashierLogin(){ //function returns false when all attempts are
                       used
    string username, password;
    int attempt = 3;
    bool found;
    char choice;
    do{
          cout<<"- - - - - CASHIER LOGIN - - - - -\n";
          cout<<"username: ";
          cin>>username;
          for(int i = 0;i<cashierAccount; i++){
           if(correctUser[i] == username){
                found = true;
                    cout<<"password: ";
                    cin>>password;
                    attempt--;

                    for(int j = 0; j <cashierAccount; j++){
                        if (username == correctUser[j] && password
                            == correctPass[j]) { //checks the array
if both username and password matches with a valid account
                        system("cls");
                        cout << "Login successful! Welcome, " <<
                                 username << ". \n\n";
                        return true;
                           }
                     }
                     system("cls");
                     cout << "Incorrect password.\n";
                     cout <<"You have "<< attempt<< " attempts left.
                           \n\n";
                }
          }
          if(!found){
                system("cls");
```

```
                cout << "Username doesn't exist. \n";
                cout<<"\n";
            }
      }while(attempt !=0);

    return false;
}
```

Appendix B: adminMenu.cpp

```
#include "adminMenu.h"
#include "inventory.h"
#include <iostream>
using namespace std;

void adminMenu(){
      int choice;
      string username;
      do{
            cout <<"- - - - - Admin Menu - - - - - \n";
            cout <<"1. Add Cashier Accounts \n";
            cout <<"2. Remove Cashier Accounts \n";
            cout <<"3. View All Cashier Accounts \n";
            cout <<"4. Log Out \n\n";
            cout << "Enter Choice: ";
            cin >> choice;
            switch(choice){
                  case 1:
                        addAccount();
                        system("cls");
                        cout << "Successfully Added Account. \n\n";
                        activeacc++;
                        break;
                  case 2:
                        system("cls");
                        removeAccount();
                        break;
                  case 3:
                        system("cls");
                        cout << "   Cashier Accounts \n";
                        cout << "---------------------------------\n";
                        cout << "Username          Password \n";
                        cout << "---------------------------------\n";
                        for(int i = 0; i < activeacc; i++){
                              cout << correctUser[i] << "           "<<
                                    correctPass[i]<<endl;
                              cout << "----------------------------\n";
                        }
```

```
                    cout << " \n";
                    break;
               case 4:
                    system("cls");
                    break;
               default:
                    cout << "Invalid Choice. Please Try Again. \n";
                    break;
          }
     }while(choice !=4);
}

void addAccount(){
     string check;
     bool New;
     system("cls");
     for(int i = activeacc; i < (activeacc + 1); i++){
          do{
               New = true;
               cout << "- - - - - Add Cashier Account - - - - -\n";
               cout << "Enter Cashier Username: ";
               cin >> check;
               for(int j = 0;j < activeacc;j++){
                    if(check == correctUser[j]){
                         system("cls");
                         cout << "Username already exists. Try
                                   Another Username \n\n";
                         New = false;
                    }
               }
          }while(!New);
          correctUser[i] = check;
          cout << "Enter Password: ";
          cin >> correctPass[i];
     }
}

void removeAccount(){
     bool found = false;
```

```
      string username;
      char choice;
      system("cls");
      do{
            cout << "- - - - - Remove Cashier Account - - - - - \n";
            cout << "Enter Cashier Username: ";
            cin >> username;
            for(int i = 0; i < activeacc; i++){ //searches for the
                                                account
                  if(correctUser[i] == username){
                        found = true;
                        cout << "   Cashier Accounts \n";
                        cout << "---------------------------------\n";
                        cout << "Username          Password \n";
                        cout << correctUser[i] << "           "<<
                                correctPass[i]<<endl;
                        cout << "---------------------------------\n";
                        cout << "Are you sure you want to remove this
                                 account?(y/n): ";
                        cin >> choice;
                        if(choice == 'y'){
                              for (int j = i; j < activeacc - 1; j++) {
                                    correctUser[j] = correctUser[j + 1];
//moves all username and passwords up one position in the array after
the deleted one
                              }
                              activeacc--; //removes the duplicate/last
element
                              system("cls");
                              cout << "Cashier Account Successfully
                                       Removed. \n";
                        }else{
                              system("cls");
                              return;
                        }
                  }
            }
            if(!found){
                  system("cls");
```

```
                cout << "Username doesn't exist. Try Another
Username. \n\n";
            }
      }while(!found);
}
```

Appendix C: main.cpp

```
#include <iostream>
#include <cstdlib>
#include "login.h"
#include "inventory.h"
#include "salesTransaction.h"
#include "adminMenu.h"
#include "receipt.h"
using namespace std;

const int cashierAccount = 10; //global variables
int activeacc = 4;
string correctUser[cashierAccount] = {"hanz", "kevin", "hail",
                                      “chris”};
string correctPass[cashierAccount] = {"123", "234", "345", "456"};

const int adminAccount = 1;
string adminUser[adminAccount] = {"admin"};
string adminPass [adminAccount] = {"admin"};

const int no_items = 50;
Product inventory[no_items] = {{1,"Bread",15,100},{2,"Milk",45,20},
                               {3,"Butter",50,200}};
int stocks = 3;

int quantity,change, payment;
float finaltotal = 0;
const int max_bought = 20;
Product bought[max_bought];
int no_bought = 0;

int main(int argc, char** argv) {
     greetUser();
     int logchoice = loginMenu();
     do{
          switch(logchoice){
               case 1:
                    adminMenu();
                    break;
```

```
                case 2:
                      int choice;
                      do{
                            cout << "- - - - - Main Menu - - - - -\n";
                            cout << "1. Sales Transaction \n";
                            cout << "2. Inventory \n";
                            cout << "3. Log Out \n\n";
                            cout << "Enter Choice: ";
                            cin >> choice;
                            switch(choice){
                                  case 1:
                                        system("cls");
                                        salesTransaction();
                                        break;
                                  case 2:
                                        system("cls");
                                        inventoryMenu();
                                        break;
                                  case 3:
                                        system("cls");
                                        break;
                                  default:
                                        system("cls");
                                        cout << "Invalid Choice, Please
                                                Try Again. \n";
                            }
                      }while(choice !=3);
                      break;
          }
          logchoice = loginMenu();
     }while(logchoice !=3);

     return 0;
}
```

Appendix D: salesTransaction.cpp

```
#include "salesTransaction.h"
#include "inventory.h"
#include "receipt.h"
#include <iostream>
using namespace std;


void salesTransaction(){
     if (stocks == 0) {
        cout << "No products available in inventory.\n";
        cout << " \n";
        return;
    }
     int id,total;
     char choice;
     cout << "- - - - - Sales Transaction - - - - -\n\n";
     do{
          bool found = false;
          cout << "Enter Product ID: ";
          cin >> id;
          for(int i = 0; i < stocks; i++){
               if(id == inventory[i].id){
                    found = true;
                    cout << "Product: "<< inventory[i].name<< endl;
                    cout << "Price: "<< inventory[i].price<< endl;
                    cout << "Enter Quantity: ";
                    cin >> quantity ;
                    if(inventory[i].quantity >= quantity){
                         bought[no_bought] = inventory[i]; // copy
                                        all information for receipt
                         bought[no_bought].quantity = quantity;
                         no_bought++;
                         total = quantity * inventory[i].price;
                         finaltotal += total;
                         cout << "Total: "<< total<< endl;
                    }else{
                         cout << "Insufficient Stocks. Only "<<
inventory[i].quantity<<" available. \n";
```

```
                    }
                }
                }if(!found){
                    cout << "Product Not Found. \n";
                }
                cout << "Add more Items? (y/n): ";
                cin >> choice;
                cout<< " \n";
     }while(choice == 'y');
          cout << "Total Amount: " << finaltotal << endl;
          cout << "Payment: ";
          cin >> payment;
          if(payment >= finaltotal){
               change = payment - finaltotal;
               cout << "Change: " << change<< endl;
               system("cls");
               cout << "Transaction Completed!! \n";
               for(int i = 0 ;i < stocks; i++){  // updates the
inventory
                    if(inventory[i].id == bought[i].id){
                         inventory[i].quantity-=bought[i].quantity;
                    }
               }
               printReceipt();
               cout << " \n";
          }else{
               cout << "Insufficient Payment. \n\n";
          }
}
```

Appendix E: Inventory.cpp

```
#include "inventory.h"
#include <iostream>
#include <cstdlib>
using namespace std;

void inventoryMenu(){
      int choice;
      do{
            cout << "- - - - - Inventory Menu - - - - - \n";
            cout << "1.Add a product \n";
            cout << "2.Update a Product \n";
            cout << "3.Delete a Product \n";
            cout << "4.View Inventory \n";
            cout << "5.Go Back to Main Menu \n\n";
            cout << "Enter Choice: ";
            cin >> choice;
            switch(choice){
                  case 1:
                        productAdd();
                        stocks++;
                        system("cls");
                        cout << "Successfully added product!! \n";
                        break;
                  case 2:
                        productUpdate();
                        system("cls");
                        cout << "Successfully updated the product. \n";
                        break;
                  case 3:
                        deleteProduct();
                        break;
                  case 4:
                        system("cls");
                        viewInventory();
                        break;
                  case 5:
                        system("cls");
                        break;
```

```
                default:
                        system("cls");
                        cout << "Invalid choice \n";
                        break;
            }
      }while(choice !=5);
}

void productAdd(){
      int check;
      bool New;
      system("cls");
      for(int i = stocks; i < (stocks + 1); i++){
            do{
                  New = true;
                  cout << "- - - - - Add Product - - - - - \n";
                  cout << "Enter Product ID: ";
                  cin >> check;
                  for(int j = 0;j < stocks;j++){ //searches if product
                                                    id is already taken
                        if(check == inventory[j].id){
                              system("cls");
                              cout << "Product ID already exist. Try
                                          Another ID \n\n";
                              New = false;
                        }
                  }
            }while(!New);
            inventory[i].id = check;
            cout << "Enter product name: ";
            cin >> inventory[i].name;
            cout << "Enter price: ";
            cin >> inventory[i].price;
            cout << "Enter quantity: ";
            cin >> inventory[i].quantity;
      }
}
void productUpdate(){
      int id;
```

```
	bool found = false;
	system("cls");
	do{
		cout << "- - - - - Update Product - - - - - \n";
		cout << "Enter Product ID: ";
		cin >> id;
		for(int i = 0;i < stocks;i++){ //searching for product
			if(id == inventory[i].id){
				cout << "		Current Product \n";
				cout <<
"------------------------------------------- \n";
				cout << "ID	Product		Price
Qty
					Quantity \n";
				cout << inventory[i].id <<""<<
inventory[i].name<< "	"<< inventory[i].price<< "	"<<
inventory[i].quantity << endl;
				cout <<
"------------------------------------------- \n";
				cout << "Enter New Name: ";
				cin >> inventory[i].name;
				cout << "Enter New Price: ";
				cin >> inventory[i].price;
				cout << "Enter New Quantity: ";
				cin >> inventory[i].quantity;
				found = true;
			}
		}
		if(!found){
			system("cls");
			cout << "Product ID doesn't exist. Try Another ID
\n\n";
		}
	}while(!found);
}
void deleteProduct(){
	int id;
	bool found = false;
	char choice;
	system("cls");
```

```
     do{
          cout << "- - - - - Delete Product - - - - - \n";
          cout << "Enter Product ID: ";
          cin >> id;
          for(int i = 0; i < stocks; i++){ //searching for product
               if(inventory[i].id == id){
                    found = true;
                    cout << "            Product \n";
                    cout <<
"---------------------------------------------- \n";
                    cout << "ID     Product          Price
Quantity \n";
                    cout << inventory[i].id <<""<<
inventory[i].name<< "      "<< inventory[i].price<< "       "<<
inventory[i].quantity << endl;
                    cout <<
"---------------------------------------------- \n";
                    cout << "Are you sure you want to delete this
                              product?(y/n): ";
                    cin >> choice;
                    if(choice == 'y'){
                         for (int j = i; j < stocks - 1; j++) {
//moves all items up one position in the array after the deleted one
                         inventory[j] = inventory[j + 1];
                     }
                     stocks--; //removes the duplicate/last element
                     system("cls");
                          cout << "Product deleted successfully.
\n\n";
                  }else{
                     system("cls");
                     cout << "Action Canceled. \n\n";
                     }
               }
          }
          if(!found){
               system("cls");
               cout << "Product ID doesn't exist. Try Another ID
\n\n";
```

```
            }
      }while(!found);
}
void viewInventory(){
      cout << "            Inventory \n";
      cout << "------------------------------------------------ \n";
      cout << "ID      Product          Price      Quantity \n";
      for(int i = 0; i < stocks; i++){
            cout << inventory[i].id <<" "<< inventory[i].name<< "
      "<< inventory[i].price<< "       "<< inventory[i].quantity <<
endl;
            cout << "------------------------------------------------
\n";
      }
}
```

Appendix E: Receipt.cpp

```
#include "receipt.h"
#include "salesTransaction.h"
#include "inventory.h"
#include <fstream>
#include <iostream>
#include <iomanip>
using namespace std;

void printReceipt() {

    Product p = inventory[stocks - 1];

    ofstream receipt("receipt.txt");
    receipt << fixed << setprecision(2);

    receipt << "========= PRODUCT RECEIPT =========\n";
    for(int i = 0; i < no_bought; i++){
      receipt << "Product ID: " << bought[i].id << "\n";
      receipt << "Product Name: " << bought[i].name << "\n";
           receipt << "Product Price: P" << bought[i].price << "\n";
      receipt << "Quantity: " << quantity << "\n";
      receipt << "------------------------------------ \n";
      }
      receipt << "Payment: " << payment << endl;
    receipt << "Change: " << change << "\n";
    receipt << "------------------------------------\n";
    receipt << "TOTAL: P" << finaltotal << "\n";
    receipt << "====================================\n";

    receipt.close();

    system("notepad receipt.txt");
}
```