TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES

938 Aurora Boulevard, Cubao, Quezon City

COLLEGE OF ENGINEERING AND ARCHITECTURE

Computer Engineering Department

CPE 007

Programming Logic and Design

Automated Retail Inventory System for Efficiency (ARISE): A C++ Based Management Program

Submitted By:

Acala, Matt John
Arididon, Gabriel Dwyane
Astudillo, Thomas Joseph
Bungag, Marc Jhowell
Sevilla, Jullian Edrain R.

CPE11S2

Mrs. Marjorie Escanan
Instructor

In Partial Fulfillment of the Requirements for the Final Project at the Technological Institute of the Philippines – Cubao, Quezon City

November 2025

Introduction

Sari-sari stores play a vital role in many Filipino communities, serving as convenient neighborhood shops that provide easy access to everyday necessities such as snacks, beverages, and household products. These stores are often managed by individual owners who rely on daily sales to support their families and meet basic needs.

The client for this study is a sari-sari store owner who manages a small retail business to help provide for her family. She sells a variety of everyday items such as snacks, biscuits, instant coffee, shampoos, soaps, and cleaning detergents. Currently, she records her sales and tracks inventory using an Excel sheet. While this method is more organized than writing in a notebook, it is still prone to errors, time-consuming, and requires manual updating for each transaction, making it difficult to maintain accurate records.

To address these challenges, this project proposes a C++ based program named Automated Retail Inventory System for Efficiency (ARISE) to automate sales and inventory management. The system allows the store owner to record transactions efficiently, compute totals and change automatically, and update inventory in real time. By replacing the manual Excel-based method with a user-friendly automated system, the client can reduce errors, save time, and maintain accurate records. This study aims to develop and evaluate an automated retail inventory system tailored to the needs of small and medium enterprises. The goal of this system is to address the limitations of manual inventory processes and demonstrate how automation can improve accuracy, efficiency and scalability in retail operations.

Statement of the Problem

The issue noticed is that most small shop store owners continue using manual calculations to calculate their inventory and sales invoices, which results in inaccurate records, calculation mistakes, and inefficiency in inventory management. The manual process of calculations hampers the ability to track the availability of products, update the level of stocks, and keep accurate sales records, leading to problems like overstocking, stockout, and loss of profit. In addition, manual computation is inefficient and susceptible to human error, which may hinder business operations and decision-making. As a solution to these problems, this research seeks to create an Automated Retail Inventory System for Efficiency (ARISE) based on the C++ programming language that will automate inventory tracking, billing, and data recording. The system proposed aims to assist small retail shop owners in maintaining better accuracy, minimizing human mistakes, conserving time, and maximizing the overall efficiency of inventory and sales handling.

Benefits to the Users

- Efficiency and Time-Saving – The system automates calculations and inventory updates, making transactions faster.
- Accuracy and Error Reduction – It minimizes mistakes in computing totals and change.
- Real-Time Inventory Tracking – Each sale automatically updates product stock levels.

Objectives

- To develop a C++ system that automatically calculates and updates inventory in real-time.

- To provide an accessible and user-friendly interface for sari-sari stores to manage inventory digitally.
- To help small retail businesses owners in calculating and selling products efficiently.

Hardware Requirements

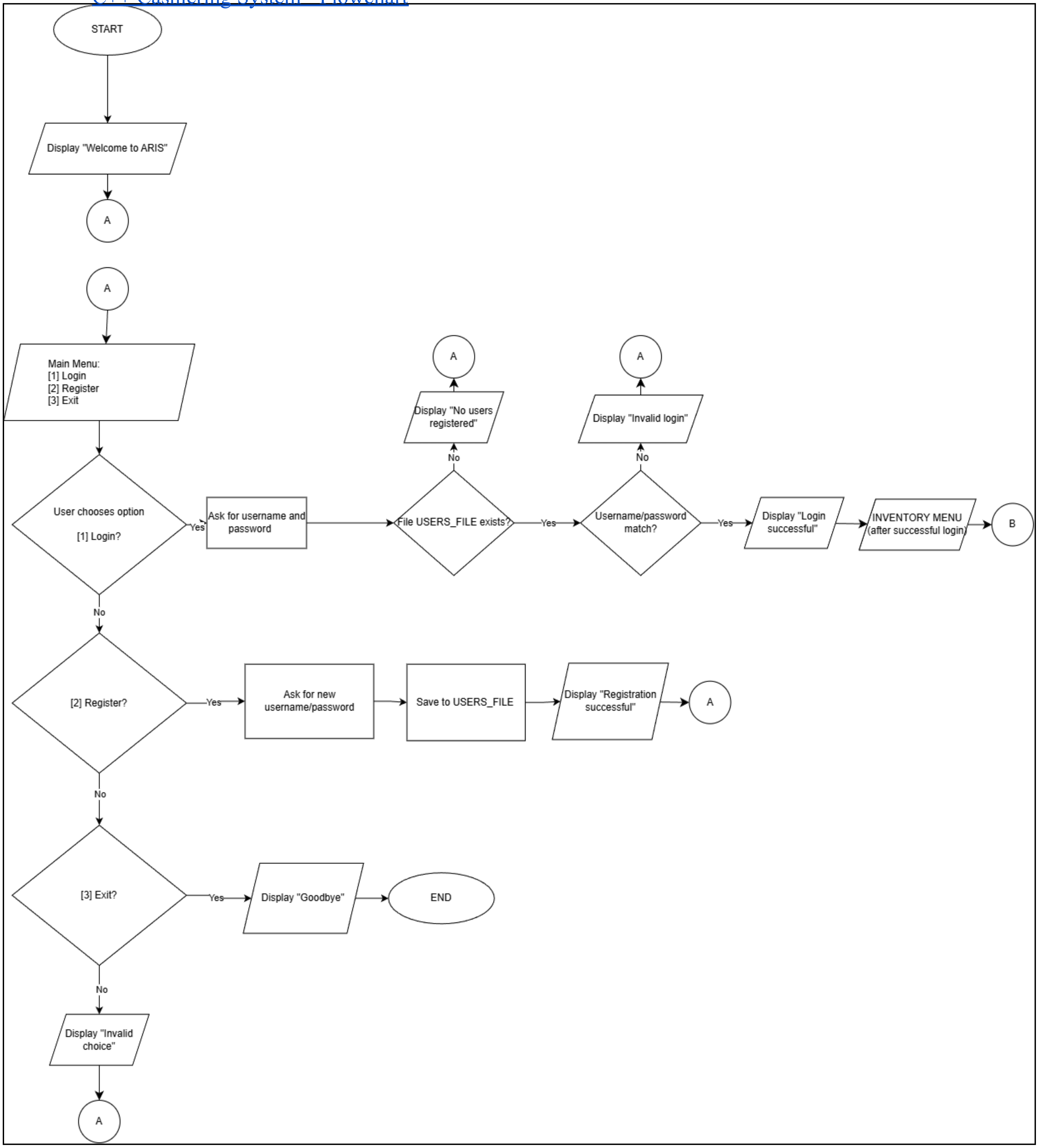
Component	Minimum Requirements	Recommended Requirements
Processor (CPU)	Intel Core I3 (or equivalent) and above	Intel Core i5/i7 or AMD Ryzen 5+
Memory (RAM)	2 GB	4 GB or higher
Storage	300 MB free disk space	500 MB free disk space
Display	800×600 resolution	1024×768 or higher
Input Devices	Standard Keyboard	Keyboard and mouse
Output Devices	Console/terminal screen	Console or IDE Terminal with color support

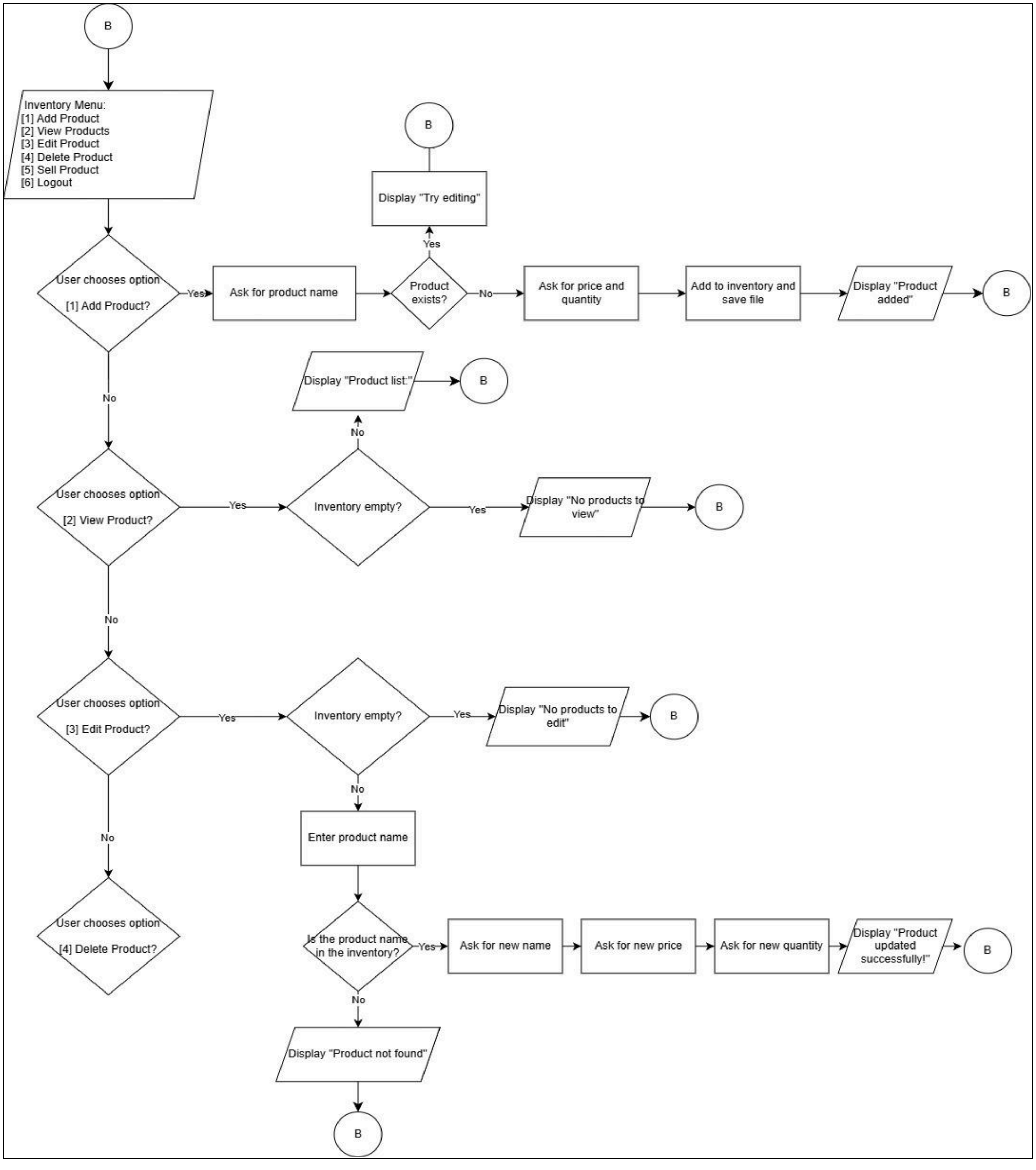
Software Requirements

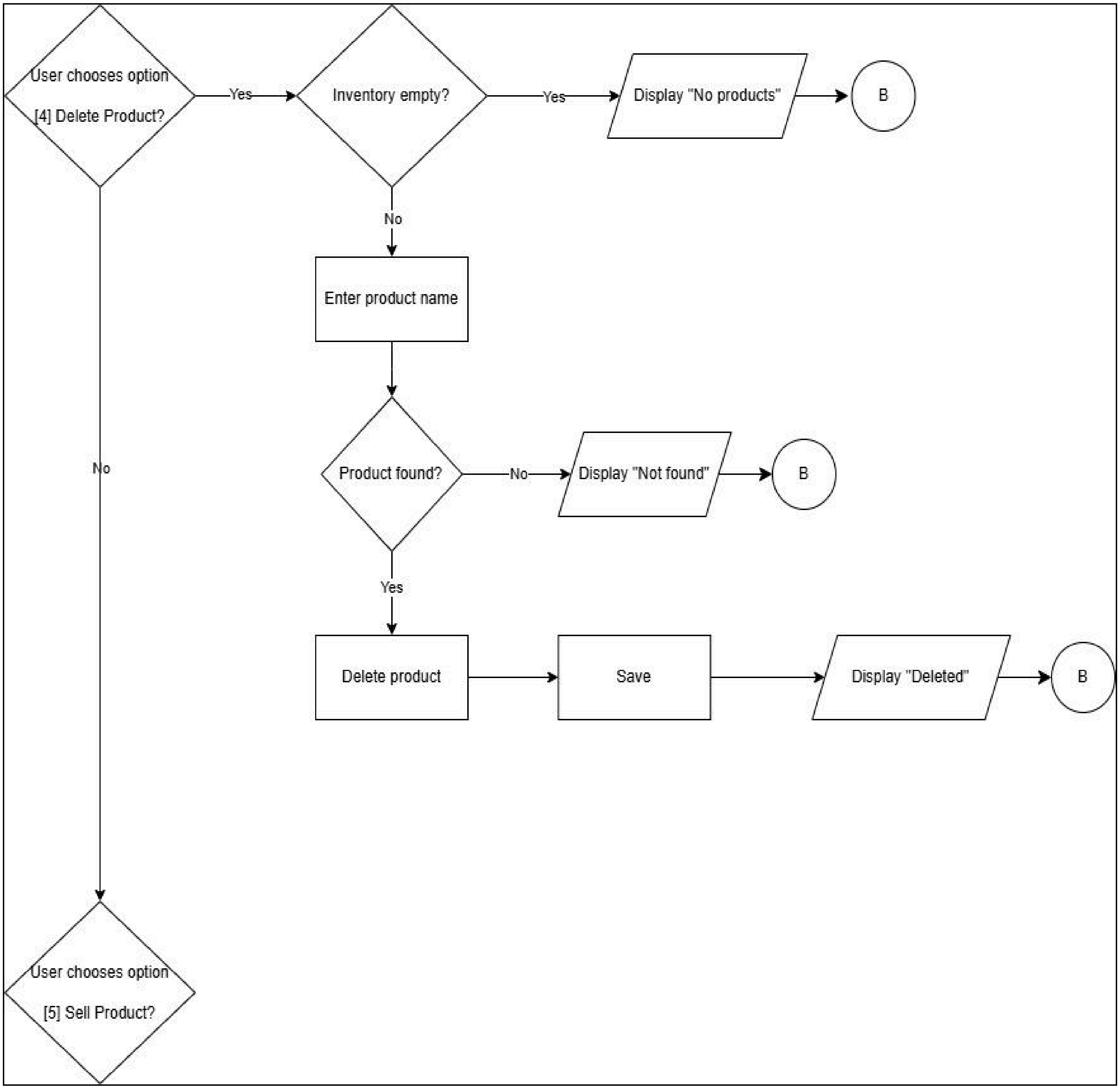
Category	Requirements
Operating System	Windows 7/8/10/11, Linux (Ubuntu, Fedora) macOS
Programming Language	C++ (Standard C++ 11 or later)
Compiler	GCC (MinGW for Windows), Clang or MSVC
Integrated Development Environment (IDE)	Code:Blocks, Dev-C++, Visual Studio or Visual Studio Code
Text Editor (for file viewing)	Microsoft Excel, Notepad, VS code, Sublime Text, or any text editor
File Storage	users.txt – stores usernames and passwords inventory.csv – stores product name, price, and quantity

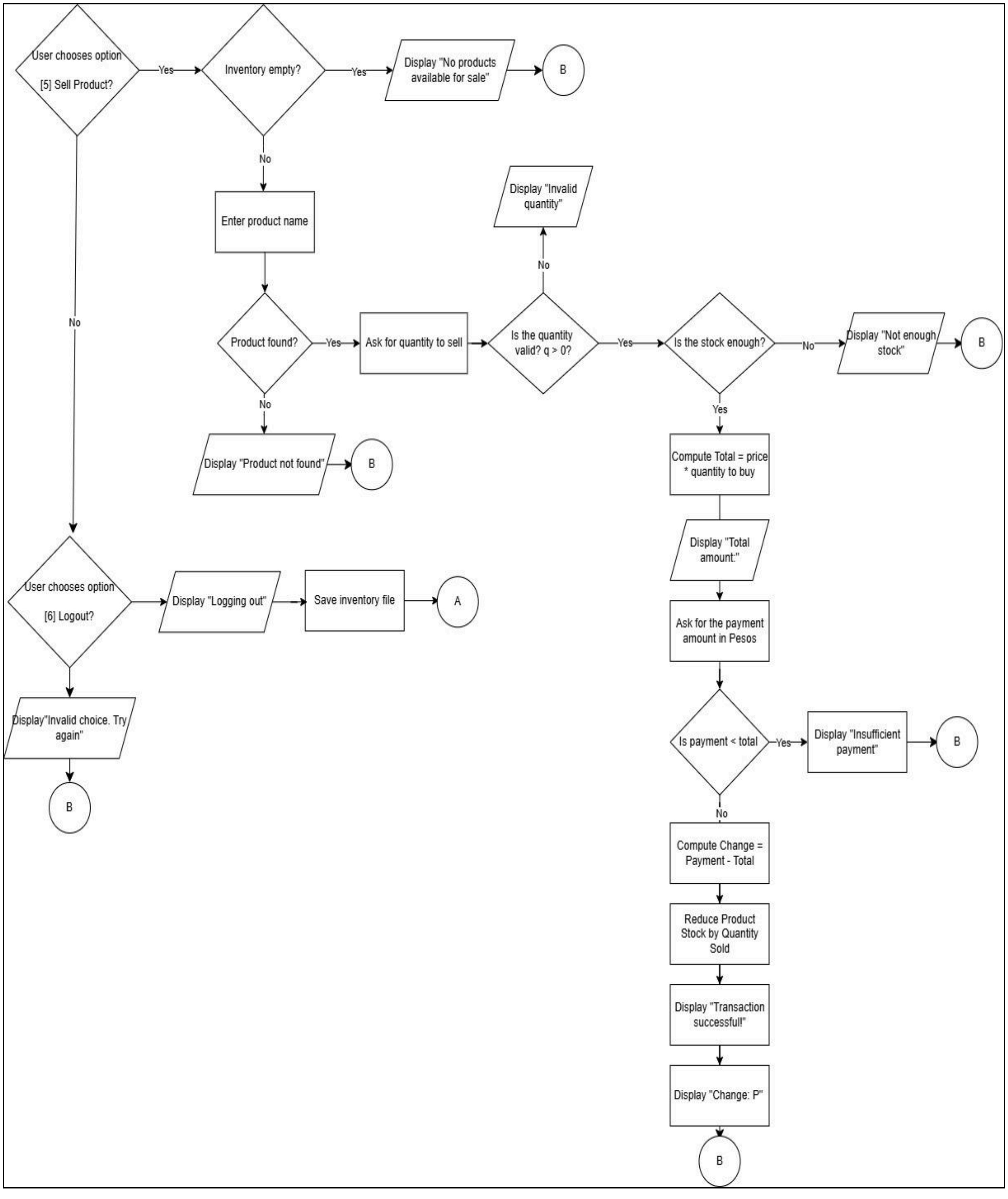
Flowchart of the System

C++ Cashiering System - Flowchart




B
Inventory Menu:
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
User chooses option [1] Add Product?
Yes
Ask for product name
Product exists?
Yes
Display "Try editing"
B
No
Ask for price and quantity
Add to inventory and save file
Display "Product added"
B
No
User chooses option [2] View Product?
Yes
Inventory empty?
No
Display "Product list:"
B
Yes
Display "No products to view"
B
No
User chooses option [3] Edit Product?
Yes
Inventory empty?
Yes
Display "No products to edit"
B
No
Enter product name
Is the product name in the inventory?
Yes
Ask for new name
Ask for new price
Ask for new quantity
Display "Product updated successfully!"
B
No
Display "Product not found"
B
No
User chooses option [4] Delete Product?


User chooses option [4] Delete Product?
Yes
Inventory empty?
Yes
Display "No products"
B
No
Enter product name
Product found?
No
Display "Not found"
B
Yes
Delete product
Save
Display "Deleted"
B
No
User chooses option [5] Sell Product?



User chooses option [5] Sell Product?
Yes
Inventory empty?
Yes
Display "No products available for sale"
B
No
Enter product name
Product found?
Yes
Ask for quantity to sell
Is the quantity valid? q > 0?
No
Display "Invalid quantity"
Yes
Is the stock enough?
No
Display "Not enough stock"
B
No
Display "Product not found"
B
Yes
Compute Total = price * quantity to buy
Display "Total amount:"
Ask for the payment amount in Pesos
Is payment < total
Yes
Display "Insufficient payment"
B
No
Compute Change = Payment - Total
Reduce Product Stock by Quantity Sold
Display "Transaction successful!"
Display "Change: P"
B
No
User chooses option [6] Logout?
Display "Logging out"
Save inventory file
A
Display"Invalid choice. Try again"
B

Pseudocodes

Login System

```
Display "=== Welcome to ARISE ==="
Display Options:
1. Login
2. Register
3. Exit

DO
   Input User_Choice

   IF User_Choice = 1 THEN
      Display "=== Login ==="
      Input Username
      Input Password

      Open User_File for Reading

      IF File not found THEN
         Display "No users registered yet. Please register first."
         Return to Menu
      ELSE
         Search the file for matching Username and Password
         IF Found THEN
            Display "Login Successful! Welcome " + Username
            Proceed to Inventory Menu
         ELSE
            Display "Invalid Username or Password."
            Return to Menu
         END IF
      END IF

   ELSE IF User_Choice = 2 THEN
      Display "=== Register ==="
      Input New_Username
      Input New_Password
      Open User_File in Append Mode
      Write Username and Password to file
      Close file
```

```
    Display "Registration Successful! You can now log in."
        Return to Menu

    ELSE IF User_Choice = 3 THEN
        Display "Goodbye!"
        RETURN 0

    ELSE
        Display "Invalid choice. Try again."
    END IF

REPEAT UNTIL User_Choice = 3
```

Inventory System
(After Successful Login)

```
Display "=== Inventory Menu ==="

Display:

1. Add Product

2. View Products

3. Edit Products

4. Delete Products

5. Sell Products

6. Log out



DO

    Input Inventory_Choice



    IF Inventory_Choice = 1 THEN

        Display "Enter Product name: "

        Input Product_Name
    Check if Product already exists
     IF Yes THEN
         Display "Product already exists. Try Editing."
         Return to Inventory Menu
     ELSE
         Display "Enter Price: "
         Input Price

         Display "Enter Quantity: "
         Input Quantity

         Add Product to Inventory
         Display "Product Added"
         Return to Inventory Menu
     END IF

  ELSE IF Inventory_Choice = 2 THEN
     Check if Inventory is Empty
     IF Yes THEN
         Display "No Products"
         Return to Inventory Menu
     ELSE
```

```
Display "Product List:"
    Output all Products
    Return to Inventory Menu
  END IF

ELSE IF Inventory_Choice = 3 THEN
  Check if Inventory is Empty
  IF Yes THEN
    Display "No Products to Edit"
    Return to Inventory Menu
  ELSE
    Display "Enter Product Name: "
    Input Product_Name

    Check if Product is Found
    IF No THEN
      Display "Product Not Found"
      Return to Inventory Menu
    ELSE
      Display "Enter New Name: "
      Input New_Name
      Display "Enter New Price: "
```

```
            Input New_Price
            Display "Enter New Quantity: "
            Input New_Quantity

            Save the Updated Product Info
            Display "Product Updated"
            Return to Inventory Menu
        END IF
    END IF

ELSE IF Inventory_Choice = 4 THEN
    Check if Inventory is Empty
    IF Yes THEN
        Display "No Products to Delete"
        Return to Inventory Menu
    ELSE
        Display "Enter Product Name: "
        Input Product_Name

        Check if Product is Found
        IF No THEN
        Display "Product Not Found"

        Return to Inventory Menu

      ELSE

        Delete Product from Inventory

        Save Updated Inventory

        Display "Product Deleted"

        Return to Inventory Menu

      END IF

    END IF


ELSE IF Inventory_Choice = 5 THEN

    Check if Inventory is Empty

    IF Yes THEN

      Display "No products available for sale."

      Return to Inventory Menu

    ELSE

      Display "Enter Product Name: "

      Input Product_Name


      Check if Product Exists

      IF Product Found THEN

     Display "Enter Quantity to Sell: "

             Input Quantity
```

```
IF Quantity <= 0 THEN
        Display "Invalid Quantity."
        Return to Inventory Menu
    END IF

    IF Quantity > Available_Stock THEN
        Display "Not enough stock available."
        Return to Inventory Menu
    END IF

    Total = Price * Quantity
    Display "Total Amount: " + Total

    Display "Enter Payment: "
    Input Payment

    IF Payment < Total THEN
        Display "Insufficient Payment."
        Return to Inventory Menu
    END IF

    Change = Payment - Total
```

```
  Reduce Product Stock by Quantity Sold
              Save Updated Inventory


              Display "Transaction Successful!"
              Display "Change: " + Change
              Return to Inventory Menu


          ELSE
              Display "Product not found."
              Return to Inventory Menu
          END IF
      END IF


   ELSE IF Inventory_Choice = 6 THEN
      Save Inventory File
      Display "Logging Out..."
      Return to Login Menu


   ELSE
      Display "Invalid Choice. Try Again."
   END IF


REPEAT UNTIL Inventory_Choice = 6
RETURN 0
```

Data Dictionary

Data Name	Data Type	Example Value	Description/Purpose
product	struct	{ "Coke", 25.00, 50 }	A structure that defines one product in the inventory.
name	string	“Fruit ninja”	Name of the product.
price	double	25.00	Price of the product.
inventory	vector<product>	[{"Coke",25.00,50}, {"Bread",15.00,20}]	A list that stores all product records currently in stock.
INVENTORY_FILE	const string	"inventory.txt"	The text file used to store all product information.
USERS_FILE	const string	"users.txt"	The text file used to store registered users and passwords.
username	string	admin	Username entered by the user during login or registration.
password	string	admin	Password entered by the user during login or registration.
choice	int	1	Username entered by the user during login or registration.
qtyToBuy	int	7	Quantity of the product purchased by the customer.
payment	double	100.00	Amount paid by the customer during a sale.

total	double	69,420.00	Computed total price for the transaction (price × qtyToBuy).
change	double	420.00	Amount of money returned to the customer (payment - total).
line	string	"Coke,25.00,50"	A single line read from inventory.txt during loading.
inFile	ifstream	N/A	File input stream used to read data from files.
outFile	ofstream	N/A	File output stream used to write data into files.

File Name	**File Type**	**File Content**	**Description/Purpose**
inventory.csv	Text File	Coke,25.00,50 Bread,15.00,20 Soap,10.00,30	Stores product details: name, price, and quantity separated by commas.
users.txt	Text File	admin,admin	Stores usernames and passwords separated by a space.

Conclusion

The system was considered successful because it met its main objectives, which are automating basic retail inventory operations such as adding, viewing, editing, deleting, and selling products. It also demonstrated proper file handling through the saving and loading of data in .csv and .txt formats. During the demonstration, the client shared that the system felt somewhat complex to use without any tutorial or guidance, especially for users who are not familiar with command-line programs. This feedback emphasized the need for a more user-friendly interface and clearer instructions. Overall, the ARISE program proved to be a functional prototype that can assist sari-sari store owners in managing their products more efficiently while also providing a foundation for future improvements such as a graphical user interface and mobile compatibility.

Recommendations

1. **Integration of a Graphical User Interface (GUI)**

   The researchers recommend developing a graphical user interface for better accessibility and understanding among business owners. A GUI would make the system more intuitive, user-friendly, and visually appealing, reducing the difficulty of reading plain text outputs especially for older sari-sari store owners.

2. **Enhancement of Pricing Flexibility**

   It is also recommended to modify the system to support batch-based pricing. Some store owners sell items such as candies or bubble gums in bundles (e.g., 3 pieces for P4). The current version of the system uses a single-price-per-item setup. Future versions should include a more flexible pricing structure to reflect real-world sari-sari store sales practices.

3. **Mobile Device Compatibility**

   The researchers suggest expanding the system to be compatible with mobile devices. Since many small business owners primarily use smartphones, making the program runnable on mobile platforms would enhance accessibility and widen its usability beyond laptops and desktop computers.

4. **Development of a Standalone Application**

   Lastly, it is recommended to develop a standalone application version of ARIS. This application could be downloaded and installed on various devices, eliminating the need to manually copy or run code. An app-based version can integrate both GUI elements and automation features, making the system more convenient and efficient for daily business operations.

Appendices

Appendix A: Source Code

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <iomanip>
using namespace std;

struct Product {
    string name;
    double price;
    int quantity;
};

const string INVENTORY_FILE = "inventory.csv";
const string USERS_FILE = "users.txt";


bool login();
void registerUser();
void addProduct(vector<Product> &inventory);
void viewProducts(const vector<Product> &inventory);
void editProduct(vector<Product> &inventory);
void deleteProduct(vector<Product> &inventory);
void sellProduct(vector<Product> &inventory);
void saveToFile(const vector<Product> &inventory);
void loadFromFile(vector<Product> &inventory);
void inventoryMenu(vector<Product> &inventory);

int main() {
    int choice;
    vector<Product> inventory;
    loadFromFile(inventory);

    cout << "=== Welcome to ARISE  (Automated Retail Inventory System for Efficiency)
===\n";

    do {
        cout << "\n[1] Login\n";
        cout << "[2] Register\n";
        cout << "[3] Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        if (choice == 1) {
            if (login()) {
                inventoryMenu(inventory);
            }
        }
        else if (choice == 2) {
            registerUser();
        }
        else if (choice != 3) {
            cout << "Invalid choice. Try again.\n";
        }

    } while (choice != 3);

    cout << "Goodbye!\n";
    return 0;
}

/bool login() {
    string username, password;
    cout << "\n=== Login ===\n";
    cout << "Username: ";
    cin >> username;
    cout << "Password: ";
    cin >> password;

    ifstream inFile(USERS_FILE);
    if (!inFile) {
        cout << "No users registered yet. Please register first.\n";
        return false;
    }
    string u, p;
    while (inFile >> u >> p) {
        if (u == username && p == password) {
            cout << "Login successful! Welcome, " << username << ".\n";
            return true;
        }
    }

    cout << "Invalid username or password.\n";
    return false;
}

void registerUser() {
    string username, password;
    cout << "\n=== Register ===\n";
    cout << "Enter new username: ";
    cin >> username;
    cout << "Enter new password: ";
    cin >> password;

    ofstream outFile(USERS_FILE, ios::app);
    if (!outFile) {
        cerr << "Error: Unable to open user file.\n";
        return;
    }

    outFile << username << " " << password << endl;
    outFile.close();
    cout << "Registration successful! You can now log in.\n";
}

void inventoryMenu(vector<Product> &inventory) {
```

```cpp
    int choice;
    do {
        cout << "\n=== Automated Retail Inventory System for Efficiency (ARISE) ===\n";
        cout << "[1] Add Product\n";
        cout << "[2] View Products\n";
        cout << "[3] Edit Product\n";
        cout << "[4] Delete Product\n";
        cout << "[5] Sell Product\n";
        cout << "[6] Logout\n";
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore(10000, '\n');

        switch (choice) {
            case 1: addProduct(inventory); saveToFile(inventory); break;
            case 2: viewProducts(inventory); break;
            case 3: editProduct(inventory); saveToFile(inventory); break;
            case 4: deleteProduct(inventory); saveToFile(inventory); break;
            case 5: sellProduct(inventory); break;
            case 6:
                saveToFile(inventory);
                cout << "Logging out...\n";
                break;
            default:
                cout << "Invalid choice. Try again.\n";
        }
    } while (choice != 6);
}


void addProduct(vector<Product> &inventory) {
    Product p;
    cout << "\nEnter Product Name: ";
    getline(cin, p.name);

    for (const auto &item : inventory) {
        if (item.name == p.name) {
            cout << "Product already exists. Try editing instead.\n";
            return;
        }
    }

    cout << "Enter Price: ";
    cin >> p.price;
    cout << "Enter Quantity: ";
    cin >> p.quantity;
    cin.ignore(10000, '\n');

    inventory.push_back(p);
    cout << "\nProduct added successfully!\n";
}

void viewProducts(const vector<Product> &inventory) {
    if (inventory.empty()) {
        cout << "\nNo products in inventory.\n";
        return;
    }

    cout << "\n=== Product List ===\n";
    cout << left << setw(20) << "Name"
         << setw(10) << "Price"
         << setw(10) << "Quantity" << endl;
    cout << "----------------------------------------\n";

    for (const auto &p : inventory) {
        cout << left << setw(20) << p.name
```

```
            << setw(10) << fixed << setprecision(2) << p.price
            << setw(10) << p.quantity << endl;
    }
}

void editProduct(vector<Product> &inventory) {
    if (inventory.empty()) {
        cout << "\nNo products to edit.\n";
        return;
    }

    string name;
    cout << "\nEnter Product Name to edit: ";
    getline(cin, name);

    for (auto &p : inventory) {
        if (p.name == name) {
            cout << "Editing product: " << p.name << endl;
            cout << "Enter new name: ";
            getline(cin, p.name);
            cout << "Enter new price: ";
            cin >> p.price;
            cout << "Enter new quantity: ";
            cin >> p.quantity;
            cin.ignore(10000, '\n');
            cout << "\nProduct updated successfully!\n";
            return;
        }
    }

    cout << "Product not found.\n";
}
```

```
void deleteProduct(vector<Product> &inventory) {
    if (inventory.empty()) {
        cout << "\nNo products to delete.\n";
        return;
    }

    string name;
    cout << "\nEnter Product Name to delete: ";
    getline(cin, name);

    for (size_t i = 0; i < inventory.size(); ++i) {
        if (inventory[i].name == name) {
            cout << "Deleting product: " << inventory[i].name << endl;
            inventory.erase(inventory.begin() + i);
            cout << "\nProduct deleted successfully!\n";
            return;
        }
    }

    cout << "Product not found.\n";
}

void sellProduct(vector<Product> &inventory) {
    if (inventory.empty()) {
        cout << "\nNo products available for sale.\n";
        return;
    }

    string name;
    int qtyToBuy;
    double payment;

    cout << "\nEnter Product Name to sell: ";
```

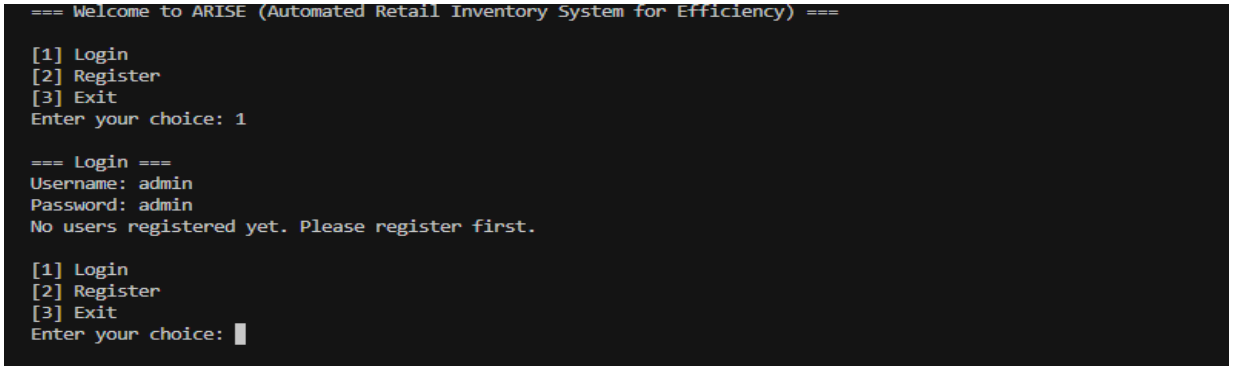
```
getline(cin, name);

for (auto &p : inventory) {
    if (p.name == name) {
        cout << "Enter quantity to sell: ";
        cin >> qtyToBuy;

        if (qtyToBuy <= 0) {
            cout << "Invalid quantity.\n";
            return;
        }

        if (qtyToBuy > p.quantity) {
            cout << "Not enough stock available.\n";
            return;
        }

        double total = p.price * qtyToBuy;
        cout << "Total amount: P" << fixed << setprecision(2) << total << endl;
        cout << "Enter payment: P";
        cin >> payment;

        if (payment < total) {
            cout << "Insufficient payment.\n";
            return;
        }

        double change = payment - total;
        p.quantity -= qtyToBuy;

        saveToFile(inventory);
        cout << "\nTransaction successful!\n";
        cout << "Change: P" << fixed << setprecision(2) << change << endl;
```

```
            return;
        }
    }

    cout << "Product not found.\n";
}



void saveToFile(const vector<Product> &inventory) {
    ofstream outFile(INVENTORY_FILE);
    if (!outFile) {
        cerr << "Error saving file.\n";
        return;
    }

 // for headers
    outFile << "Name,Price,Quantity\n";

    for (const auto &p : inventory) {
        outFile << p.name << "," << p.price << "," << p.quantity << "\n";
    }

    outFile.close();
}

void loadFromFile(vector<Product> &inventory) {
    ifstream inFile(INVENTORY_FILE);
    if (!inFile) {
        cout << "(No existing inventory file found — starting new.)\n";
        return;
    }

    inventory.clear();
    string line;
    while (getline(inFile, line)) {
        Product p;
        size_t pos1 = line.find(',');
        size_t pos2 = line.find(',', pos1 + 1);
        if (pos1 != string::npos && pos2 != string::npos) {
            p.name = line.substr(0, pos1);
            p.price = stod(line.substr(pos1 + 1, pos2 - pos1 - 1));
            p.quantity = stoi(line.substr(pos2 + 1));
            inventory.push_back(p);
        }
    }

    inFile.close();
}
```

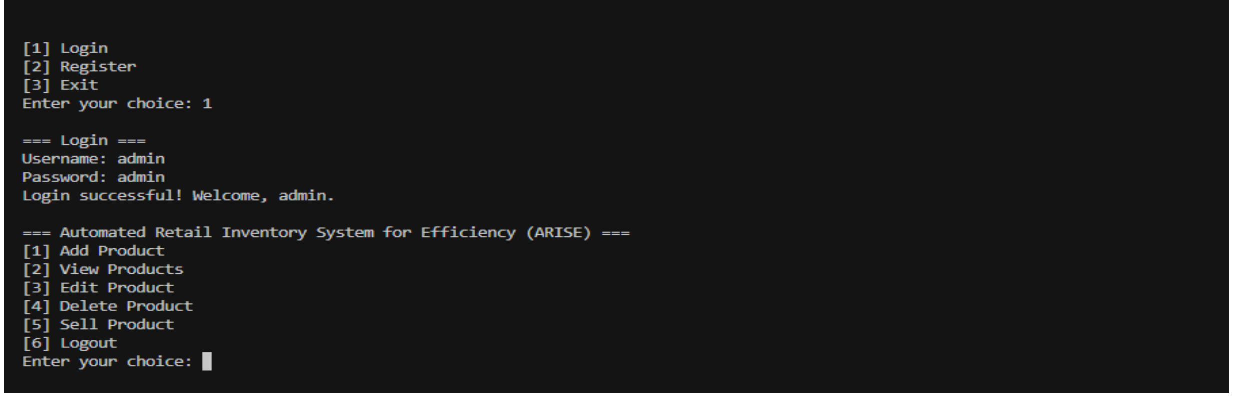
Appendix B: Program Outputs

Login System

Choice 1: Login



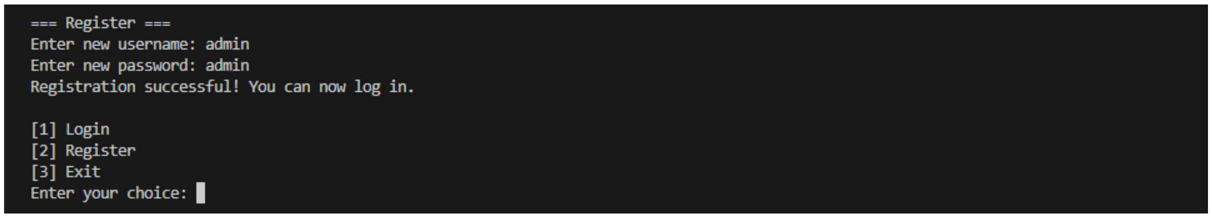
```
=== Welcome to ARISE (Automated Retail Inventory System for Efficiency) ===

[1] Login
[2] Register
[3] Exit
Enter your choice: 1

=== Login ===
Username: admin
Password: admin
No users registered yet. Please register first.

[1] Login
[2] Register
[3] Exit
Enter your choice:
```



```
[1] Login
[2] Register
[3] Exit
Enter your choice: 1

=== Login ===
Username: admin
Password: admin
Login successful! Welcome, admin.

=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice:
```

The login system reads data from users.txt. If the username and password match, access is granted. Otherwise, it prompts “No users registered yet. Please register first.”

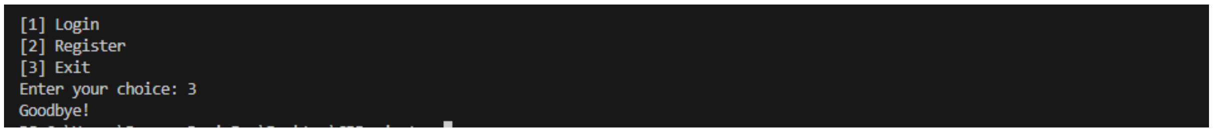
Choice 2: Register



```
=== Register ===
Enter new username: admin
Enter new password: admin
Registration successful! You can now log in.

[1] Login
[2] Register
[3] Exit
Enter your choice:
```

After registration, the system will ask if you want to login.

Choice 3: Exit

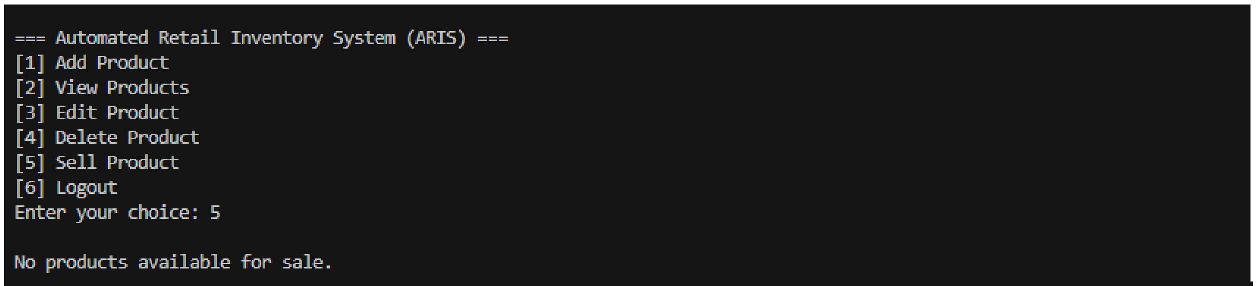


```
[1] Login
[2] Register
[3] Exit
Enter your choice: 3
Goodbye!
```

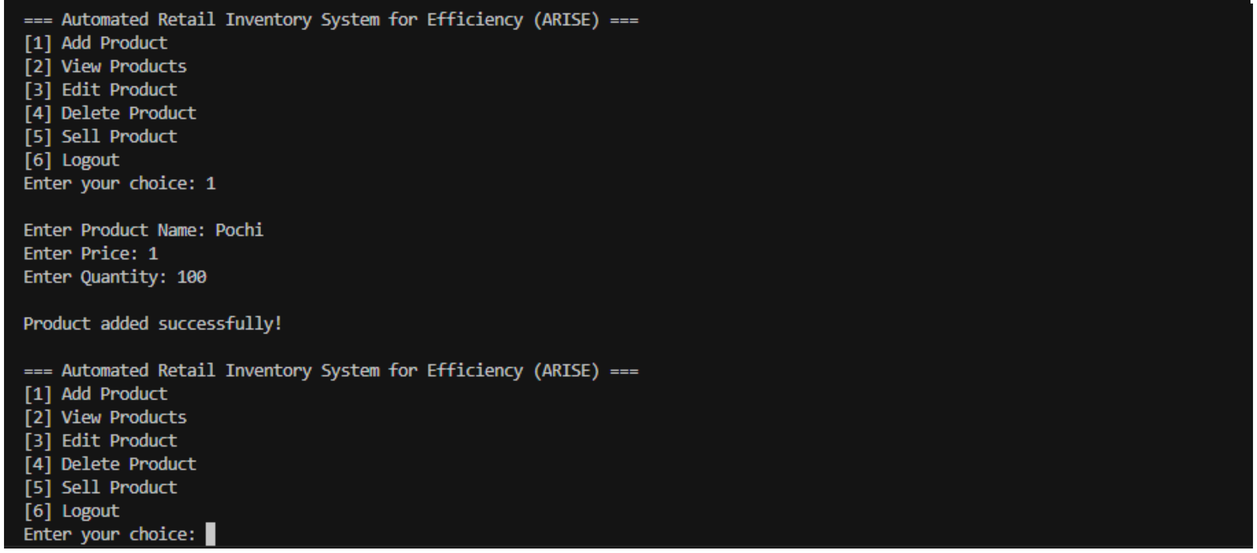
If the user chooses to exit, the program will end and display “Goodbye!.”

Inventory System

Choice 1: Add Product



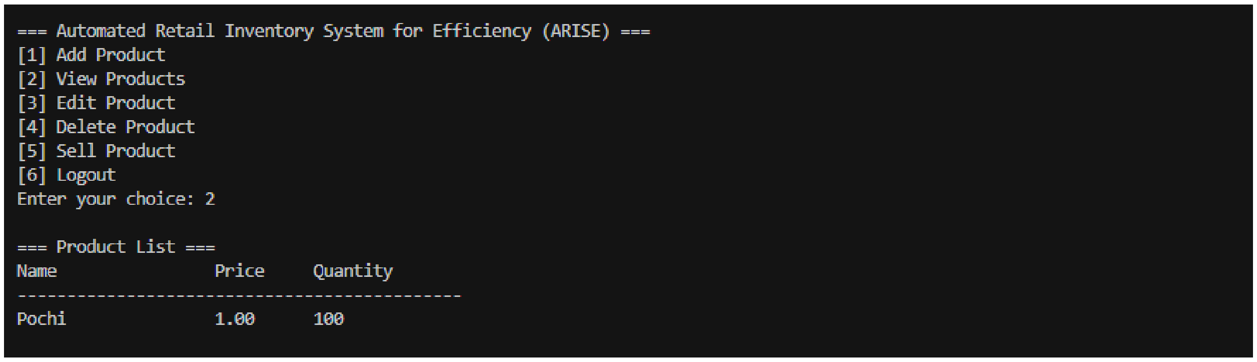
```
=== Automated Retail Inventory System (ARIS) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice: 5

No products available for sale.
```



```
=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice: 1

Enter Product Name: Pochi
Enter Price: 1
Enter Quantity: 100

Product added successfully!

=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice:
```

This function allows the user to add a new product to the inventory. It asks for the product name, price, and quantity. Before adding, it checks if the product name already exists to avoid duplicates. If valid, it pushes the new product into the inventory vector and confirms success.

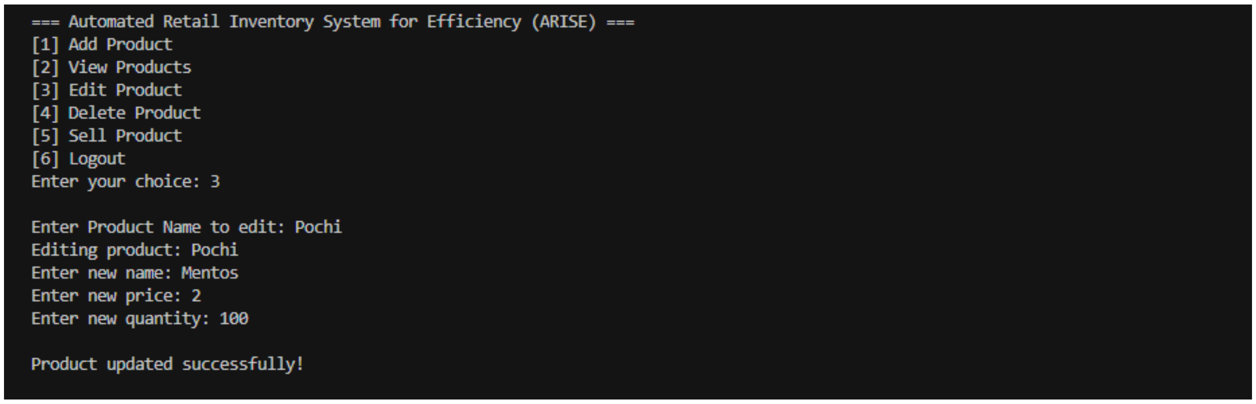
Choice 2: View Products



```
=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice: 2

=== Product List ===
Name                Price     Quantity
---------------------------------------------
Pochi               1.00      100
```

This function displays all products stored in the inventory in a clean table format.

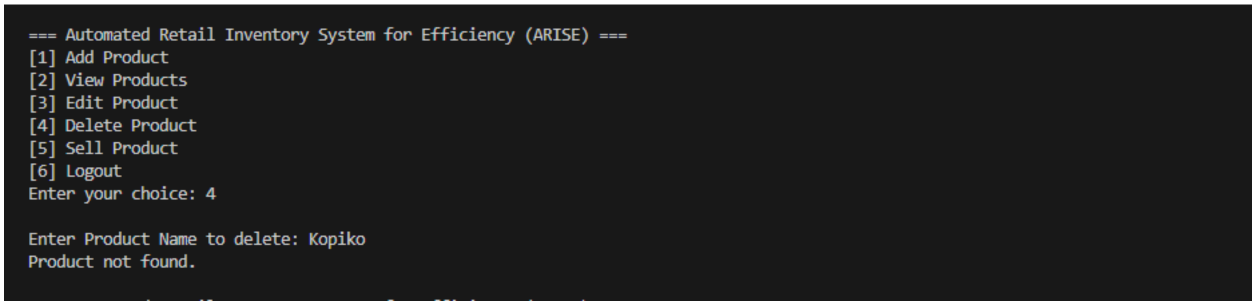
Choice 3: Edit Product



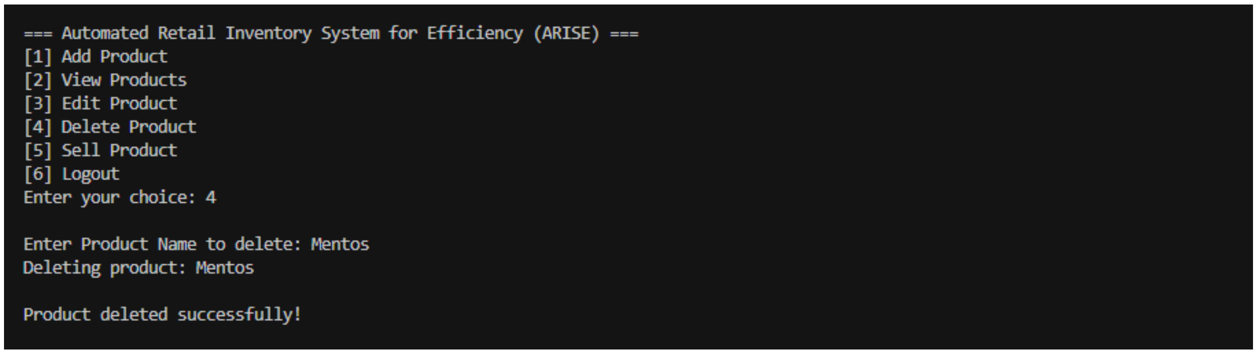
```
=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice: 3

Enter Product Name to edit: Pochi
Editing product: Pochi
Enter new name: Mentos
Enter new price: 2
Enter new quantity: 100

Product updated successfully!
```

This function lets the user edit the product. If the product is found in the inventory, the system will ask for its new name, price and quantity. Otherwise, it will display “Product not found.”

Choice 4: Delete product



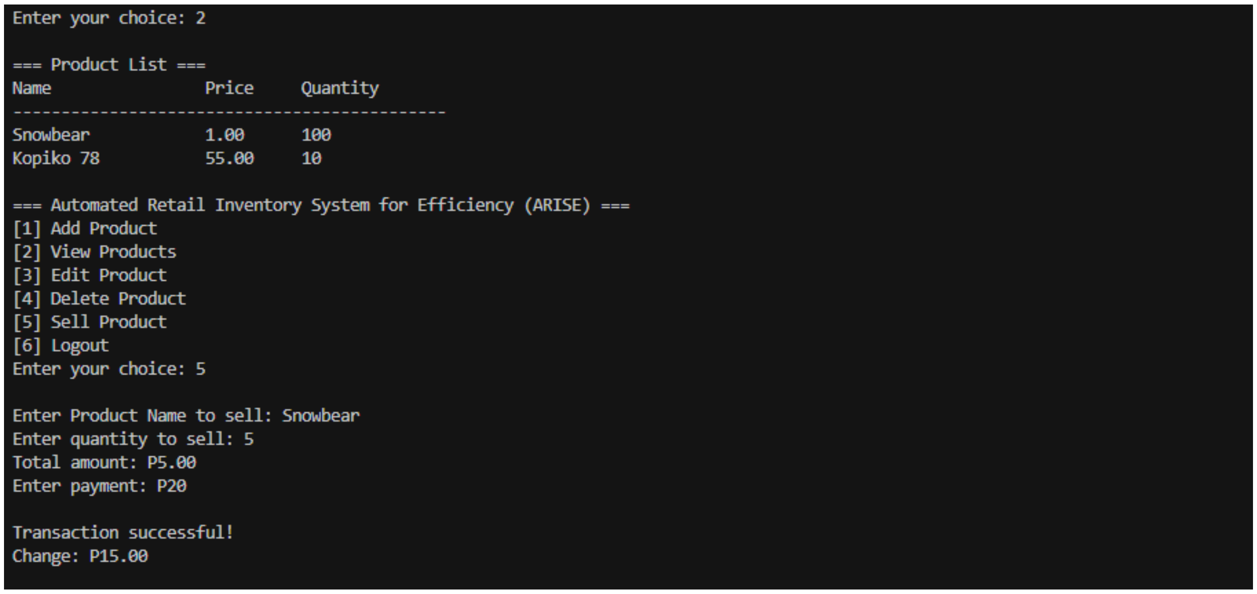
```
=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice: 4

Enter Product Name to delete: Kopiko
Product not found.
```



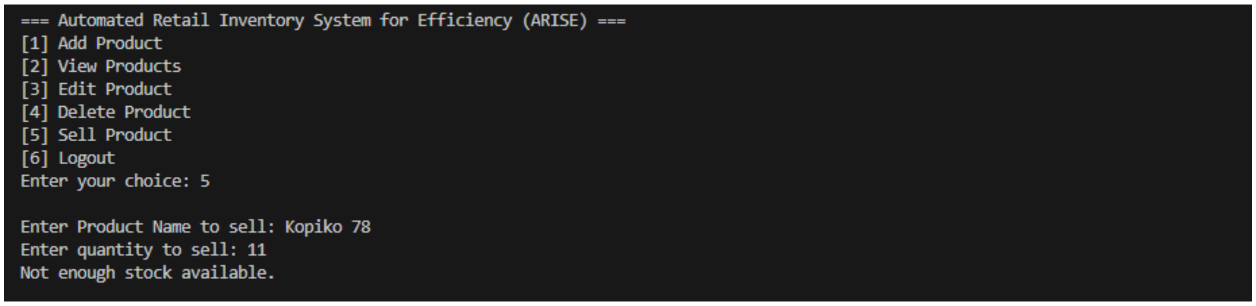
```
=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice: 4

Enter Product Name to delete: Mentos
Deleting product: Mentos

Product deleted successfully!
```

This function lets the user delete the product. It will first ask the name of the product to delete. Otherwise, it will display “Product not found.”

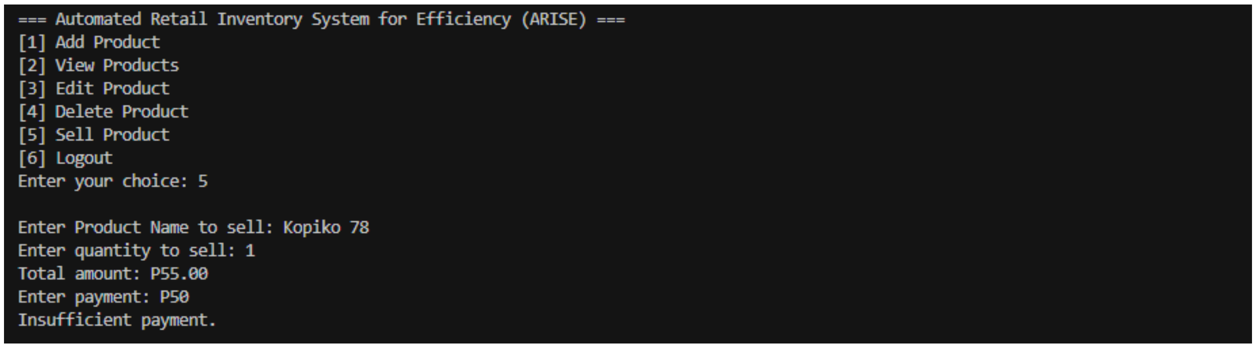
Choice 5: Sell Product



```
Enter your choice: 2

=== Product List ===
Name                Price     Quantity
---------------------------------------------
Snowbear            1.00      100
Kopiko 78           55.00     10

=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice: 5

Enter Product Name to sell: Snowbear
Enter quantity to sell: 5
Total amount: P5.00
Enter payment: P20

Transaction successful!
Change: P15.00
```



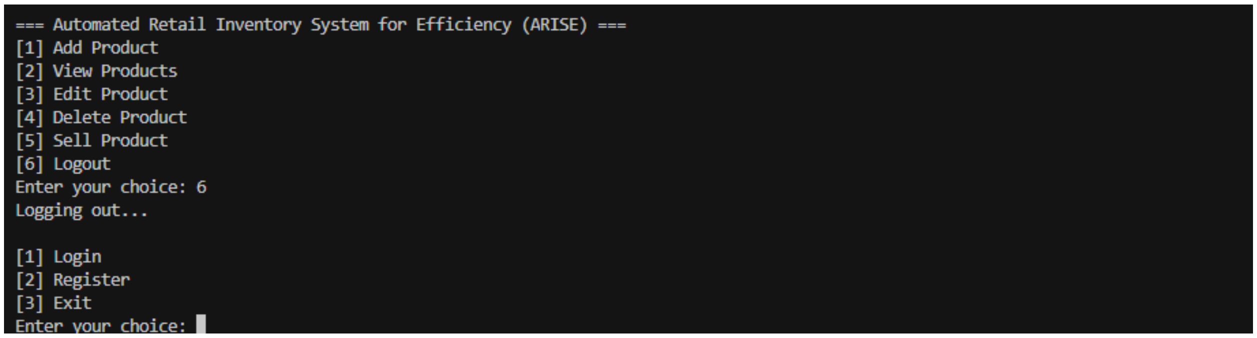
```
=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice: 5

Enter Product Name to sell: Kopiko 78
Enter quantity to sell: 11
Not enough stock available.
```



```
=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice: 5

Enter Product Name to sell: Kopiko 78
Enter quantity to sell: 1
Total amount: P55.00
Enter payment: P50
Insufficient payment.
```

This function allows the user to sell a product by entering its name and quantity. The program then displays the total cost and prompts the user to enter the payment amount. If the payment and quantity are sufficient, the transaction is completed successfully, and the change is displayed. Otherwise, if there is not enough stock, the program notifies the user that the product is out of stock. If the payment is insufficient, the program informs the user that the amount entered is not enough to complete the purchase.

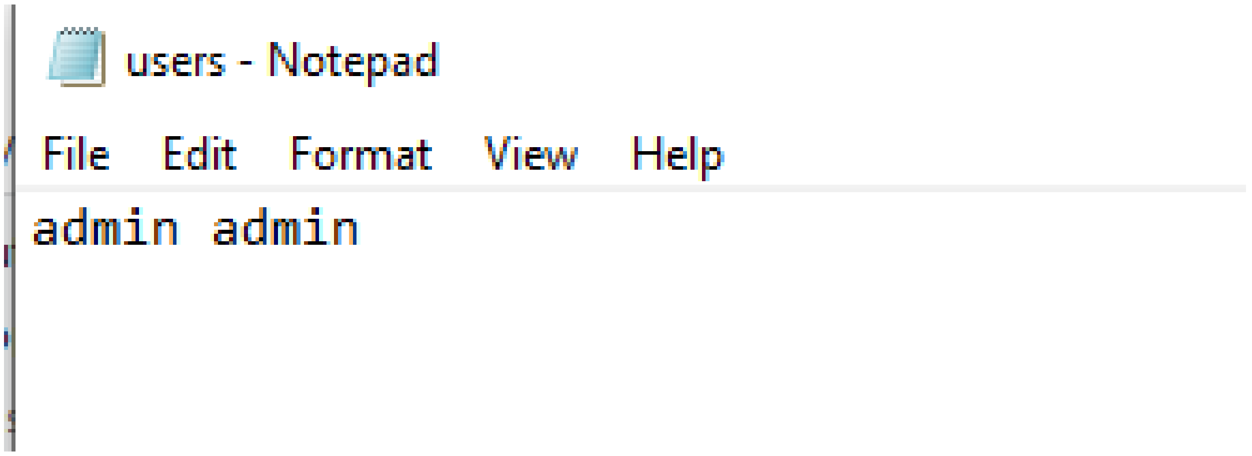
Choice 6: Logout



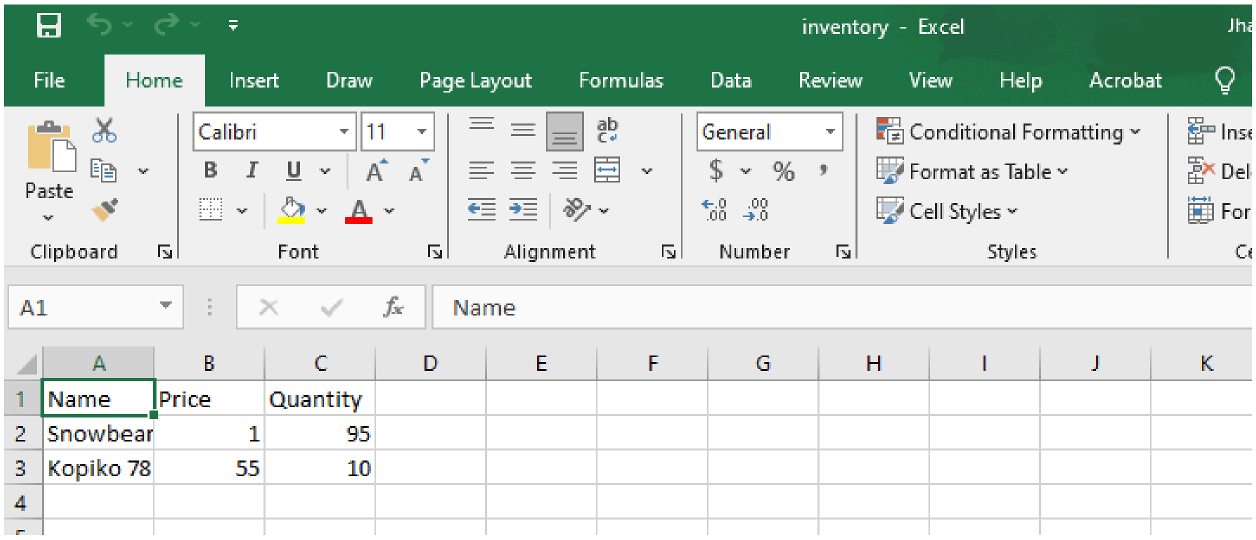
```
=== Automated Retail Inventory System for Efficiency (ARISE) ===
[1] Add Product
[2] View Products
[3] Edit Product
[4] Delete Product
[5] Sell Product
[6] Logout
Enter your choice: 6
Logging out...

[1] Login
[2] Register
[3] Exit
Enter your choice:
```

This function logs out the user by displaying "Logging out…" and then returns to the login screen. The program will only terminate when the user chooses the exit option.

Appendix C: Data Files



```
admin admin
```



inventory - Excel
File Home Insert Draw Page Layout Formulas Data Review View Help Acrobat
Clipboard Font Alignment Number Styles
Conditional Formatting
Format as Table
Cell Styles

A1 — Name

	A	B	C
1	Name	Price	Quantity
2	Snowbear	1	95
3	Kopiko 78	55	10
4			

All inputs from the program's login system and inventory system will be saved on a .txt file and .csv files in real-time.

Appendix D: Project Phase Distribution

The chart below shows the percentage distribution of each phase of the project. Each phase was assigned based on its duration within the project timeline.

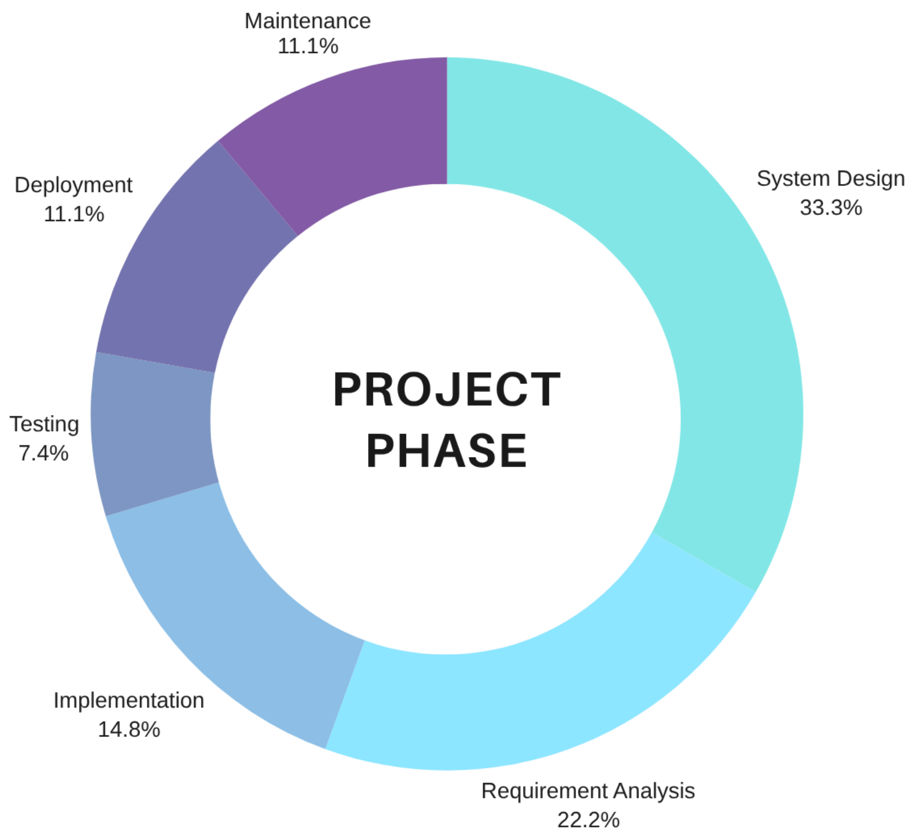


Figure D.1: Project Phase Distribution Pie Chart

As shown in the figure, the System Design Phase took the largest portion (33.3%) of the total timeline, followed by Requirement Analysis (22.2%), and Implementation (14.8%). Testing, Deployment, and Maintenance took smaller but essential parts of the development cycle.