Department of Artificial Intelligence

22AIE101: Problem Solving & C Programming, Fall 2023

Project Report

Inventory Management System Electronic Shop

Team Members:

Vishal Seshadri B	CB.SC.U4AIE23260
K S Venkatram	CB.SC.U4AIE23236
Sanggit Saaran K C S	CB.SC.U4AIE23247
Harivaarthan TD	CB.SC.U4AIE23228
Surya HA	CB.SC.U4AIE23267

Supervised By:

Dr. Vidhya Kamakshi V

Assistant Professor

Department of Artificial Intelligence

Amrita Vishwa Vidyapeetham

Date of submission: 28/12/2023

Signature of the Project Supervisor:

TABLE OF CONTENTS

1. Abstract

Inventory management is a critical aspect of business operations that involves overseeing and controlling the inflow and outflow of goods within an organization. An effective inventory management system is crucial for businesses of all sizes to optimize resources, reduce costs, and enhance overall operational efficiency. The primary goal is to strike a balance between maintaining sufficient stock levels to meet customer demand and minimizing excess inventory to avoid unnecessary holding costs.

Inventory Management is one of the basic problems in almost every company. Before the computer age and integration, paper tables, and paperwork solutions were being used as inventory management tools. These were far from being a solution, took so much time, and even needed employees just for this section of the organization. There was no efficient solution available in the many companies during these days. Every process was based on paperwork, the human fault rate was high, the process and the tracing of the inventory losses were not possible, and there were no efficient logging systems. After the computer age, every process started to be integrated into an electronic environment. And now we have qualified technology to implement new solutions to these problems. Software-based systems bring the advantages of having the most efficient control with less effort and employees. These developments provide new solutions for inventory management systems in this context. In this paper, a new solution for Inventory Management System (IMS) is designed and implemented.

2. Introduction

The Inventory Management system is based on the idea of providing efficient administration over shops, stores, or any other kind of shopping spot. Our inventory management system is based on the electronic shop which we have named Electroshock Pro. We are using C programming language for developing the inventory management system. In this, we have a total of 5 binary files for storing information. The first file is the item-master file which consists of all the details of all the items that the store maintains. The second file is about the suppliers from whom we purchase goods whenever our stock goes down. The third file is for the customers who visit our shop, we collect information about them and store it in the file. The fourth file is for the Sales transaction this is to store the information on all the sales which we are making. The fifth file is for the purchase transaction which we are using for storing all the data about the purchases we make with our suppliers. This entire project ensures that we make the process of managing a shop simpler rather than humans manually writing down records of all the data. Manually writing down all the data and performing calculations can result in a lot of errors and omissions, as well as taking up a lot of time. This is particularly true when dealing with multiple files. This system ensures faster and more accurate completion of tasks, while also reducing the likelihood of human errors.

3. Contributions

- **Vishal Seshadri B**: Implementing the creation of item and deletion of records
- **Sanggit Saaran K C S**: Implementing Display function to display all the records.
- **Harivaarthan TD**: Implementing writing to file function for storing and accessing the data from the binary file and text file creation.
- **VenkatRam K S**: Implementing the Customization function for customizing the existing records
- **Surya HA**: Implementing creation of suppliers, customers, purchase and sale Transaction.

4. Literature review

Inventory management systems are essential for efficient retail operations, and the C programming language is an ideal choice for optimizing performance and control. Electronic shops face unique challenges in managing their inventory due to the fast pace of technological advancements and product lifecycles.

Recent studies suggest that automation and technology integration play a critical role in improving inventory management efficiency. Barcode systems and RFID technology are crucial tools that can streamline tracking processes, reduce errors, and provide real-time visibility of stock levels. RFID technology, in particular, is highly efficient in managing entire shops [1].

concept involved in the inventory management of surgical supplies and sterile instruments in hospitals is the utilization of programming languages and database systems for efficient identification, tracking, optimization, and control of inventory. This includes employing coding methodologies like barcoding for surgical instrument set identification and computer systems for real-time tracking, emphasizing the efficient management of sterile instruments and disposable surgical supplies. [2].

Some research papers have explored alternative technologies, such as Ultrasonic systems for distance calculations using echoes, but the implementation and associated costs are challenging, making RFID technology a preferred and cost-effective solution. In this paper, the control of allotted storage cloud model is discussed, and lots of automation, intelligence and information technology are applied to successfully combine the sources successfully [3].

Inventory Support System for Retail Shop, they anticipated demand for the subsequent period which is crucial to ensure that customer needs are met without the burden of excess stock costs. They also Designed Standard Operating Procedure for the shop [4].

Recently on 2021, The Inventory Management system comprising the following modules: Login, Sales, Inventory, Settings, and Exit, utilizing a graphical user interface (GUI) in developed. The system is designed for a medical store, incorporating various medical products. GUIs facilitate user interaction through clarity and control. The primary goal of each GUI is to provide the level of clarity that enables users to quickly initiate meaningful interactions. While GUIs can be designed to introduce a sense of mystery and intrigue, it is essential to avoid confusion to ensure an effective and user-friendly interface [5].

The IoT-based Inventory Management System for Retail Stores focuses on incorporating cutting-edge technologies to optimize inventory control. This involves concepts such as integrating IoT devices and sensors to collect real-time data on stock levels, product movements, and other relevant information. The coding aspect revolves around developing algorithms for data processing, analysis, and interpretation. The system aims to provide a seamless and automated approach to inventory monitoring, eliminating the need for manual intervention. This includes coding for communication protocols, data storage, and analysis algorithms to ensure that the information gathered from IoT devices is processed efficiently. Furthermore, the concept emphasizes the development of a user-friendly interface to enable retailers to interact with the system, view inventory insights, and make informed decisions. The overall goal is to enhance operational efficiency, minimize stockouts and overstock situations, and improve the overall management of inventory in retail environments through innovative coding practices and IoT integration. [6].

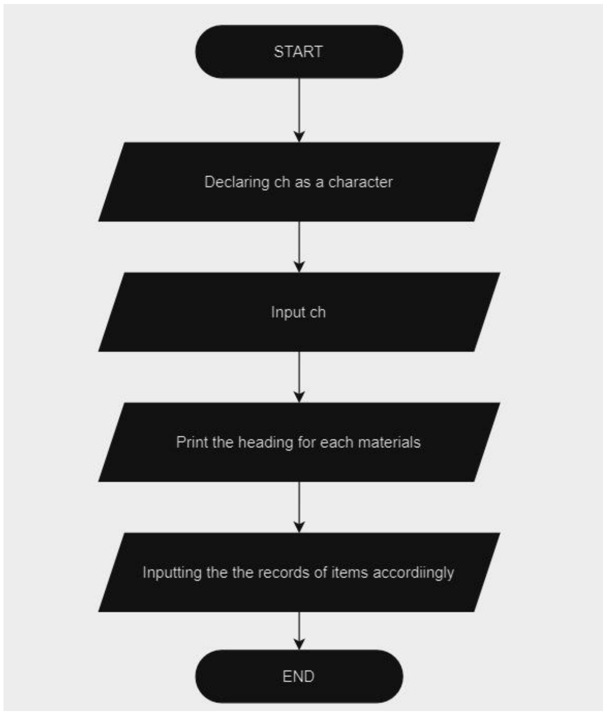
The paper "Inventory management for retail companies: A literature review and current trends" likely involves concepts such as inventory optimization, demand forecasting, supply chain management, and the use of technology in retail inventory systems. the paper could explore best practices in inventory control, vendor-managed inventory (VMI), and the integration of inventory management systems with point-of-sale (POS) systems for seamless retail operations. [7].

Research and Development of Inventory Management and Human Resource Management in ERP by Zhao, B. and Tu, This paper conducts an analysis of Enterprise Resource Planning (ERP) systems by examining their application in various aspects of business operations. It delves into the market characteristic information system and the application of ERP systems, extending from software functionalities to inventory management and human resource management for decision-making processes. The study identifies weaknesses in both the inventory management and human resource systems within ERP and subsequently proposes improvement strategies and measures. Furthermore, the paper suggests that the enhanced ERP can enable standardized and process-oriented management of daily operations, including processes related to production, procurement, and sales, facilitating collaborative processing of financial and business activities within the organization [8].

In the WEB BASED BOOKSTORE MANAGEMENT (WBBM) by Koralage, S.I. , The development process initially followed by the Rational Unified Process (RUP). HTML5 serves as the core markup language, CSS3 for styling, PHP for server-side scripting, and JavaScript, jQuery, and AJAX for client-side scripting. The database management system (DBMS) used is MySQL. WBBM is anticipated to enhance the efficiency and productivity of daily business functions while strengthening relationships with suppliers, businesses, and customers [9].

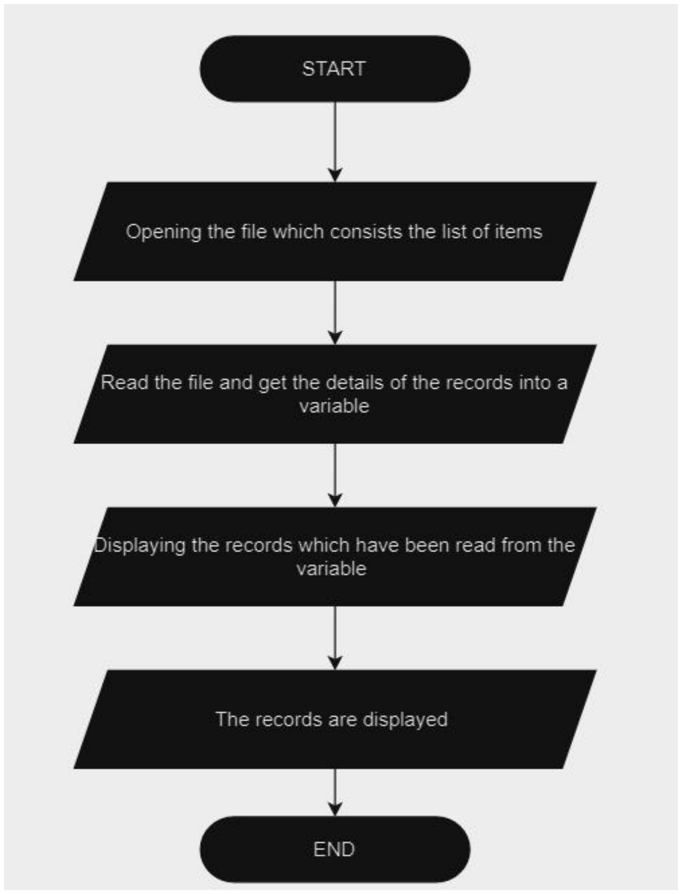
This research delves into how the implementation of an inventory management system influences a company's performance in a real-world context. The key ideas explored encompass the connection between adopting effective inventory practices and the overall success of the firm. It likely investigates how using modern inventory systems helps in maintaining optimal stock levels, cutting costs, ensuring smoother order fulfillment, and ultimately elevating customer satisfaction. The study is expected to employ practical data analysis methods, offering insights into how a well-functioning inventory management system positively impacts both the operational efficiency and financial performance of a business. [10].

5. Methodology


START
Declaring ch as a character
Input ch
Print the heading for each materials
Inputting the the records of items accordiingly
END

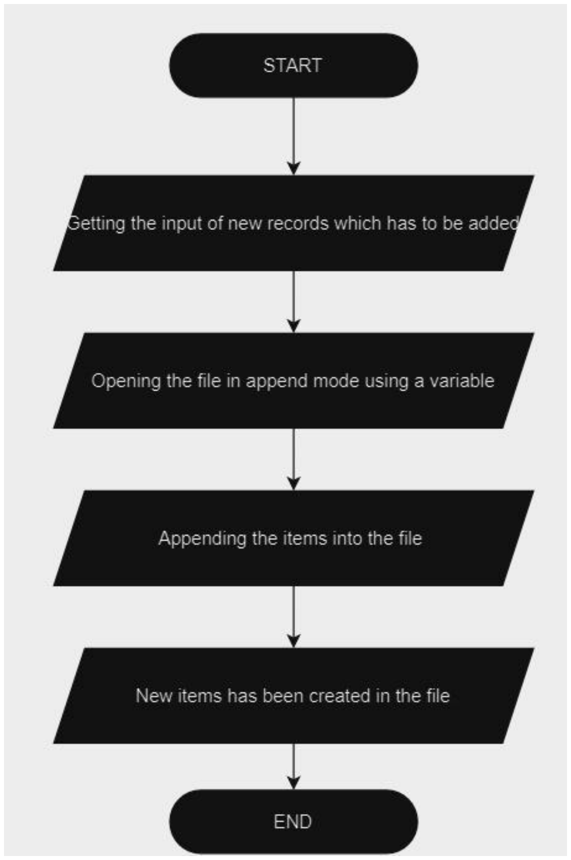
This part of the program is for the creation of each record. A similar kind of programming has been done for all the 5 files that we have. Initially, we keep a c variable “ch” as a character, and then we get the input of “ch”. We then add the titles of each material like the item code, item name, and so on. Then we input all the records inside and that makes our inventory creation of items the similar pattern has been followed for the suppliers, customers, purchase transactions, and sales transaction.

5.1. File Handling


START
Opening the file which consists the list of items
Read the file and get the details of the records into a variable
Displaying the records which have been read from the variable
The records are displayed
END

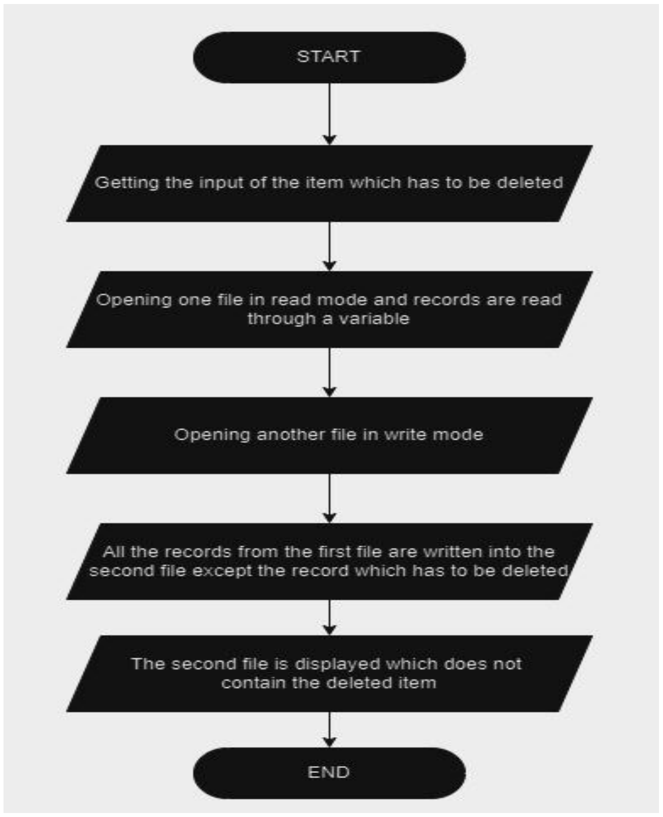
This part of the program talks about how we are using the concepts of files to display all the records. Initially, we are opening the file in which all the records have already been stored. We open the file in read binary mode and then the read part is stored in a variable. Next, the entire data stored in the variable is asked to be displayed and all the records are displayed in a more informative manner.

5.2 *Appending Binary Files*


START
Getting the input of new records which has to be added
Opening the file in append mode using a variable
Appending the items into the file
New items has been created in the file
END

In this part, we are getting the input of new records which must be appended to the binary file. For this we open the binary file in the append mode and store the contents of the opened binary file in a variable then the input which contains the details of the new record is added at the end of the binary file and the new item is created when we display the new item will also be displayed.

5.3 Deletion of Records


START
Getting the input of the item which has to be deleted
Opening one file in read mode and records are read through a variable
Opening another file in write mode
All the records from the first file are written into the second file except the record which has to be deleted
The second file is displayed which does not contain the deleted item
END

Here we are implementing the deletion of records where if a record is not needed the record is deleted. First, we get input of the name of the item which has to be deleted. We then open the binary file in read mode and store all the records in a variable. Next, we open another file in write mode and then store all the records of the binary file in the new file except the deleted record. Then the new file is displayed where we will not find the record which has been deleted.

All these are done for all 5 binary files and the same idea of thinking has been made for each of the files.

6. Experiments

Our inventory management system offers users the flexibility to customize every data element, tailoring the system to their unique needs. The comprehensive data display provides a detailed overview of the electronic shop inventory. Users can dynamically delete data, ensuring efficient management and a clean database. All data is stored in binary files, optimizing storage efficiency, and ensuring seamless integration with our user-friendly interface.

In our "Electroshock Pro" electronic Shop users are provided with a versatile set of options, each designed to facilitate efficient control and customization of data. The main menu offers three primary choices: Display Information, Perform Action, and Delete Records. This C-based system empowers users to interactively manage their electronic shop inventory. Let's delve into the comprehensive functionality of each major option within the system.

6.1. Binary Files:

The "Electro Shop" inventory system utilizes binary files to achieve efficient data persistence. Binary files store information in a compact, machine-readable format, enhancing data security and minimizing storage space. Five Binary Files have been used:

(1).Item_master.dat

(2).Purchases.dat

(3).Sales.dat

(4).Customer.dat

(5).Suppliers.dat

The binary files—Item_master.dat, Purchases.dat, Sales.dat, Customer.dat, and Suppliers.dat—serve as integral components in the "Electro Shop" inventory system. Item_master.dat stores item details, facilitating seamless addition and customization. Purchases.dat and Sales.dat log purchase and sale transactions, respectively, ensuring accurate financial tracking. Customer.dat and Suppliers.dat manage comprehensive customer and supplier databases, allowing efficient addition and modification of records. These binary files employ serialization for compact storage, direct access, and optimized retrieval. Their use enhances data security, system performance, and overall data integrity, contributing to the robustness and efficiency of the inventory management system.

```
int writeItemtofile(struct item rec) {
    FILE *wp, *rp;
    struct item readrec;
    int stat = 1; // Initialize stat
    rp = fopen("item_master.dat", "rb");
    while (stat == 1) {
        stat = fread(&readrec, sizeof(rec), 1, rp);
        if (stat == 1 && strncmp(readrec.code, rec.code, sizeof(readrec.code)) == 0) {
            printf(" The item code exists in the database \n");
            fclose(rp); // Close the file before returning
            return -1;
        }
    }
    fclose(rp);
    wp = fopen("item_master.dat", "ab");
    fwrite(&rec, sizeof(rec), 1, wp);
    printf(" Added Item... \n");
    fclose(wp);
    return 0;
}
```

6.2. Efficient Data Management and Deletion Capabilities in Electro Shop Inventory System:

6.2.1.Display Information Option (D/d):

The system initiates with a user-friendly main menu that welcomes users to "Electro Shop" and presents three main options: Display Information (D), Perform Action (A), and Delete Records (X), User input is captured through the “scanf” function to determine the chosen option. The following code prompts the user for their choice:

```
printf("\n\n\n");

    printf("          Electro Shock Pro      \n");

    printf("\n\n");

    printf("                MENU           \n");

    printf("         ====================  \n\n");

    printf("        D. Display Information\n");

    printf("        A. Perform Action\n");

    printf("        X. Delete Records\n");

    printf("\n\n\n");

    printf(" Option (D/A/X): ");

    scanf(" %c", &displayOption);


    if (displayOption == 'D' || displayOption == 'd') {

        printf("\nWhat information would you like to display?\n");

        printf("1. Inventory\n");

        printf("2. Supplier\n");

        printf("3. Customer\n");

        printf("4. Purchase Transactions\n");

        printf("5. Sale Transactions\n");
```

To ensure a tailored experience, the system dynamically calls the corresponding display function based on the user's selection. For instance, if the user chooses to display Inventory, the system invokes the displayInventory() function. This dynamic approach ensures efficient and organized information retrieval:

```
switch (option) {
        case '1':
          displayInventory();
          break;
        case '2':
          displaySupplier();
          break;
        case '3':
          displayCustomer();
          break;
        case '4':
          displayPurchaseTransactions();
          break;
        case '5':
          displaySaleTransactions();
          break;
        default:
          printf("Invalid option. Please choose again.\n");
          break;  }
```

Within each display function, the system adopts a structured and organized approach to present data. Whether showcasing inventory details, supplier information, or transaction records, the display functions utilize formatted output to enhance readability and comprehension.

For example:

```
printf(" %-15s %-30s %-10s %-10s\n", "Item Code", "Item Name", "Quantity", "Price");
```

Post-display, the system facilitates user interaction by prompting whether they would like to continue exploring information. This iterative process ensures a seamless and engaging user experience within "Electro Shop":

```
printf(" Would you like to Continue(Y/N):  ");
scanf(" %c", &option);
```

6.2.2. Performing Action Option (A/a):

Opting for "A/a" leads users to a sub-menu where they can perform various actions:

Entering a new item in the Item Master	:(createitem()).
Customizing item details	:(customizeitem()).
Creating a new supplier	:(createsupplier()).
Customizing supplier details	:(customizeSupplierDetails()).
Creating a new customer	:(createCustomer()).
Customizing customer details	:(customizeCustomerDetails()).
Recording a purchase transaction	:(recordPurchaseTransaction()).
Recording a sale transaction	:(recordSaleTransaction()).

These functions allow the user to add and customize the records of items, suppliers, and customers and modify the Sale and purchase transaction. The system employs a switch-case structure to invoke the corresponding functions based on the user's chosen action.

Opting for option "A" guide’s users to a sub-menu, presenting a spectrum of actions for efficient inventory management. From seamlessly adding new items and customizing item specifics to streamlining supplier and customer database entries, users can tailor their records with precision.

The system's switch-case structure ensures seamless execution, providing an intuitive and efficient user experience. This functionality empowers users to dynamically manage records, fostering personalized relationships with both suppliers and customers.

Additionally, the system facilitates the recording of purchase and sale transactions, ensuring accurate financial tracking and updating of inventory records. This cohesive approach supports comprehensive inventory management within the "Electro Shop" system.

6.2.3. Delete Records Option(X/x):

Selecting "X/x" empowers users to selectively delete records from specific categories, including Inventory, Supplier, Customer, Purchase Transactions, and Sale Transactions.

The system utilizes a switch-case structure to call deletion functions (e.g., deleteItem(), deleteSupplier()) corresponding to the chosen category.

6.3. Deleting Records:

Users can selectively delete records, enhancing the system's data management capabilities.

Upon choosing the "Delete Records" option, users are prompted to specify the category of records they wish to remove, such as Inventory, Supplier, Customer, Purchase Transactions, or Sale Transactions. The system employs a switch-case structure to call the appropriate deletion functions corresponding to the selected category.

For instance, deleting an item (deleteItem()) removes the specified item from the item master, ensuring that inventory records remain accurate and up-to-date.

Similarly, deleting a supplier (deleteSupplier()) or customer (deleteCustomer()) removes the selected entity from the respective master records, maintaining data integrity. Deletion of purchase transactions (deletePurchaseTransaction()) or sale transactions (deleteSaleTransaction()) allows users to manage financial records efficiently. The system provides clear confirmation messages upon successful deletion and ensures a user-friendly experience. Robust error-handling mechanisms are in place to guide users in the event of invalid inputs or attempts to delete non-existent records.

Overall, the "Delete Records" functionality complements the system's comprehensive features, offering users a powerful tool to maintain a clean and organized database. This capability enhances data accuracy, streamlining inventory management processes and supporting informed decision-making within the "Electro Shop" system.

6.4 Comprehensive Functionality Overview for Electro Shop Inventory System

```
int createitem();

void customizeitem();

int createsupplier();

void customizeSupplierDetails();

int createCustomer();

void customizeCustomerDetails();

int recordPurchaseTransaction();

int recordSaleTransaction();

int displayInventory();

int displayCustomer();

int displaySupplier();
int displaySaleTransactions();

int displayPurchaseTransactions();

int create_report_item(struct item rec);

int create_report_supplier(struct supplier rec);

int create_report_customer(struct customer rec);

int create_report_PurchaseTransaction(struct transaction rec);

int create_report_SaleTransaction(struct transaction rec);

int writeItemtofile(struct item rec);

int writeSuppliertoFile(struct supplier rec);

int writeCustomertoFile(struct customer rec);

int writePurchaseTransactionToFile(struct transaction rec);

int writeSaleTransactionToFile(struct transaction rec);

int updateItemQuantity(const char *itemCode, long quantityChange);

void askDisplayOption(char option);

int deleteItem();

int deleteSupplier();

int deleteCustomer();

int deletePurchaseTransaction();

int deleteSaleTransaction();

void updateInventory(const char *itemCode, long qty, char transType) ;

void writeDeletedItemToFile(const char *deletedItem);

void writeDeletedCustomerToFile(const struct customer *deletedCustomer);

void writeDeletedSupplierToFile(const struct supplier *deletedSupplier);
```

This set of functions encapsulates the core functionalities of the "Electro Shop" inventory management system. The functions cover a spectrum of operations, including creating and customizing items, suppliers, and customers, as well as recording purchase and sale transactions. Display functions allow users to view various aspects of the inventory.

Additionally, the system generates detailed reports for items, suppliers, customers, purchase transactions, and sale transactions. File-handling functions, such as writing and updating binary files, are implemented for persistent data storage. Notably, the code incorporates error-handling mechanisms and quantity updates for items.

The deletion functions remove records selectively, ensuring a streamlined data management process. Overall, these functions collectively form a comprehensive and modular framework, facilitating efficient inventory control, data integrity, and a user-friendly experience in our "Electro Shop" system.

6.5 Comprehensive Insights into Report File Creation Functions for Enhanced Inventory Management in Electro Shop”

6.5.1. Item Master Report (item_master.txt):

The function create_report_item generates a detailed report capturing essential item details, including codes, descriptions, quantities, sale prices, and reorder quantities. This report, stored in "item_master.txt," acts as a fundamental reference for effective inventory management, offering a comprehensive snapshot of the current item status.

```
int create_report_item(struct item rec) {

   FILE *rp;

   struct item prec;

   int stat;

   int itemcounter;
```

```
    FILE *file = fopen("item_master.txt", "w");

    rp = fopen("item_master.dat", "rb");

    fprintf(file, "_________ELCETRO SHOCK PRO ELECTRONIC SHOP________\n\n");

    fprintf(file, "\n %-15s %-30s %-10s %-10s %-15s \n", "Item Code", "Item Description",
"Quantity", "Sale Price", "Reorder Qty");

    itemcounter = 0;

    stat = fread(&prec, sizeof(rec), 1, rp);

    while (stat == 1) {

        fprintf(file, " %-15s %-30s %-10ld %-10.2f %-15ld \n", prec.code, prec.desc, prec.qty,
prec.price, prec.reordqty);

        itemcounter += 1;

        stat = fread(&prec, sizeof(rec), 1, rp);

    }

    printf("\n\n Report Created with %d Records \n", itemcounter);

    fclose(rp);

    fclose(file);

}
```

6.5.2. Supplier Report (supplier.txt):

The create_report_supplier function compiles a report containing supplier codes and names, providing valuable insights into the supplier database. The resulting "supplier.txt" file serves as a critical resource for managing supplier relationships and maintaining an organized and up-to-date supplier directory.

6.5.3. Customer Report (customer.txt):

Utilizing the create_report_customer function, this report encapsulates customer codes and names, offering a consolidated view of the customer database. The "customer.txt" file supports the establishment of personalized customer relationships and facilitates efficient customer information management within the system.

6.6. Transaction report:

6.6.1. Purchase Transaction Report (PurchaseTransaction.txt):

The create_report_PurchaseTransaction function generates a detailed report on purchase transactions, encompassing item codes, names, quantities, and total prices. This report, stored in "PurchaseTransaction.txt," aids in monitoring procurement activities and contributes crucial data for maintaining accurate financial records.

6.6.2. Sale Transaction Report (SaleTransaction.txt):

Using the create_report SaleTransaction function, this report provides a comprehensive overview of sale transactions, including item codes, names, quantities, and total prices. The resulting "SaleTransaction.txt" file serves as a vital tool for monitoring sales activities, ensuring up-to-date financial records, and facilitating strategic decision-making.

6.7. Customization: Crafting Unique Inventory Profiles

6.7.1. Item Customization (customizeitem()):

void customizeitem();

The **customizeitem()** function in the Electro Shop enables users to personalize specific items by entering item codes. This function acts as a creative workshop, allowing users to adjust descriptions, quantities, prices, and reorder quantities. With a few simple inputs, users can sculpt a unique inventory that aligns precisely with the dynamic needs of the business.

From refining product details to meeting specific requirements, the **customizeitem()** function provides a streamlined process for tailoring the inventory to the Electro Shop's distinct character and demands.

```
void customizeitem() {
    FILE *fp;
    struct item updaterec;
    char itemName[31];
    char itemCode[11];
    int found = 0;
    int customizeOption;
    printf("Enter the Item Name to customize: ");
    scanf("%s", itemName);
    printf("Enter the Item Code to customize: ");
    scanf("%s", itemCode);
    fp = fopen("item_master.dat", "r+b");
    if (fp == NULL) {
        printf("Error opening item_master.dat file.\n");
        return;
    }
    // Searching for the item by name and code (case-insensitive)
    while (fread(&updaterec, sizeof(updaterec), 1, fp) == 1) {
        if (strcasecmp(updaterec.desc, itemName) == 0 && strcmp(updaterec.code, itemCode) == 0) {
```

```
found = 1;
// Asking for customization option
printf("Select customization option:\n");
printf("1. Customize Price\n");
printf("2. Customize Reorder Quantity\n");
printf("3. Customize Both Price and Reorder Quantity\n");
printf("4. Customize Name of the item\n");
printf("Option: ");
scanf("%d", &customizeOption);
// Updating price, reorder quantity, or both based on the chosen option
switch (customizeOption) {
    case 1:
        printf("Enter the new price: ");
        scanf("%lf", &updaterec.price);
        break;
    case 2:
        printf("Enter the new reorder quantity: ");
        scanf("%ld", &updaterec.reordqty);
        break;
    case 3:
        printf("Enter the new price: ");
        scanf("%lf", &updaterec.price);
        printf("Enter the new reorder quantity: ");
```

```
                    scanf("%ld", &updaterec.reordqty);
                    break;
                case 4:
                    printf("Enter new name :");
                    scanf("%s",&updaterec.desc);
                    break;
                default:
                    printf("Invalid option.\n");
                    break;
            }
            // Moving the file cursor to the beginning of the record
            fseek(fp, -sizeof(updaterec), SEEK_CUR);
            // Writing the updated record back to the file
            fwrite(&updaterec, sizeof(updaterec), 1, fp);
            printf("Item customization successful.\n");
            break;
        }
    }
    if (!found) {
        printf("Item not found.\n");
    }
    fclose(fp);
}
```

6.7.2. Supplier Customization (customizeSupplierDetails()):

void customizeSupplierDetails();

Forging partnerships with precision, the customizeSupplierDetails() function empowers users to shape the details of suppliers. From modifying contact information to refining the supplier's address and contact numbers, this function ensures that the network of suppliers aligns seamlessly with the evolving needs of the electronic haven. Each customization becomes a stroke in the canvas of supplier relationships.

6.7.3. Customer Customization (customizeCustomerDetails()):

void customizeCustomerDetails();

In the realm of customer relationships, the customizeCustomerDetails() function takes center stage. Users navigate through a tailored interface to edit and enhance customer details, including names, addresses, contact numbers, and email IDs. This customization not only fosters personalized interactions but also strengthens the bonds between the Electro Shop and its clients.

6.8. Inventory Update: Fine-Tuning Data Dynamics

6.8.1. File Access and Error Handling:

The function begins by attempting to open the "item_master.dat" file in read and write mode ("rb+"). This mode is chosen to allow both reading and writing operations on the file concurrently. The conditional check ensures that if the file opening operation encounters any issues, such as the file not existing or permissions being insufficient, an error message is displayed, and the function gracefully exits, preventing potential data corruption.

```
file = fopen("item_master.dat", "rb+");
if (file == NULL) {
    printf("Error opening item_master.dat file for updating inventory.\n");
    return;
    }
```

6.8.2. Iterative Item Search:

The function enters a loop to sequentially read each record from the binary file, representing individual items. The loop's purpose is to locate the specific item that matches the provided itemCode. The fread function reads the next item from the file, and the strcmp function compares the item codes.

```
while (fread(&currentItem, sizeof(struct item), 1, file) == 1) {
    if (strcmp(currentItem.desc, itemCode) == 0) {
        // Item found, proceed with inventory update...
        break;
        }
```

6.8.3. Inventory Update Logic:

Upon finding the target item, the function updates its inventory quantity based on the transaction type. For a purchase transaction (transType == 'P'), the quantity is increased; for a sale transaction (transType == 'S'), the quantity is decreased.

```
if (transType == 'P') {
   currentItem.qty += qty; // Increasing quantity for purchase
} else if (transType == 'S') {
   currentItem.qty -= qty;} // Decreasing quantity for sal
```

6.9. File Pointer Manipulation

6.9.1. File Pointer Manipulation and Write Operation:

After the inventory update, the function manipulates the file pointer to move back to the beginning of the current item's record. This repositioning is necessary before writing the updated item back to the file to avoid overwriting other data.

```
fseek(file, -sizeof(struct item), SEEK_CUR);
if (fwrite(&currentItem, sizeof(struct item), 1, file) != 1) {
   printf("Error updating item inventory.\n");
```

6.9.2. Efficient Exit from Loop:

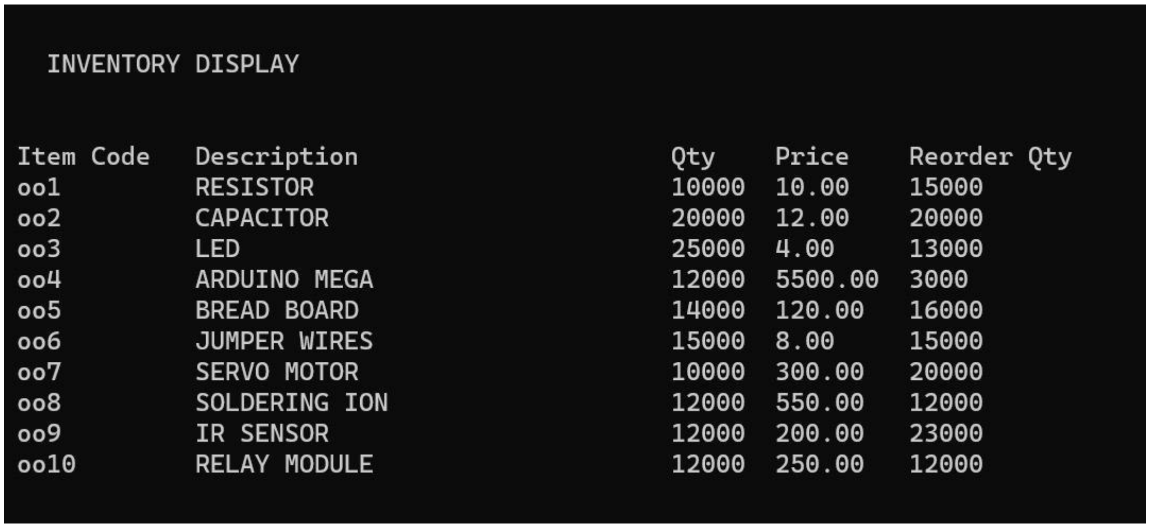
Once the target item is found and updated, the loop is terminated using the break statement. This efficient exit ensures that the function doesn't unnecessarily continue iterating through the rest of the file.

6.9.3. File Closure:

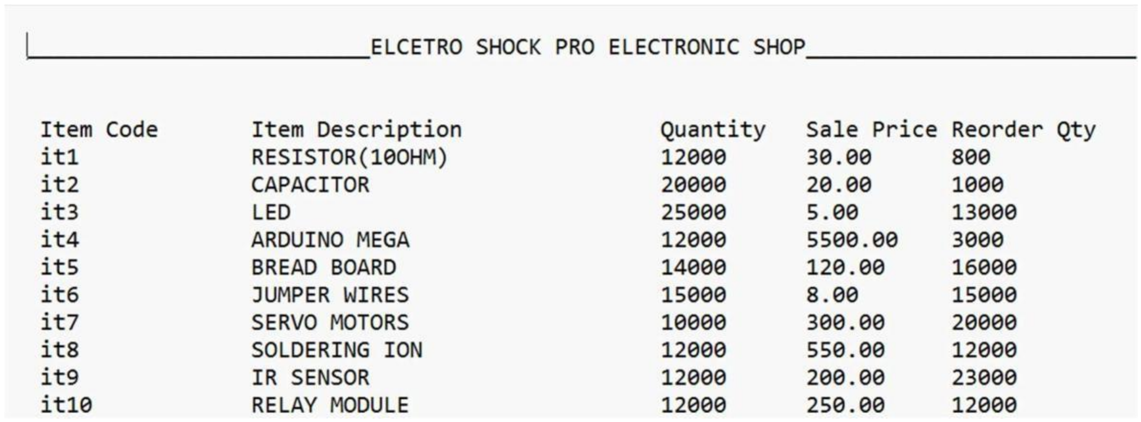
The function closes the "item_master.dat" file after the update operation. Proper file closure is crucial for releasing system resources and maintaining data consistency.

7. Results

7.1 Inventory display:

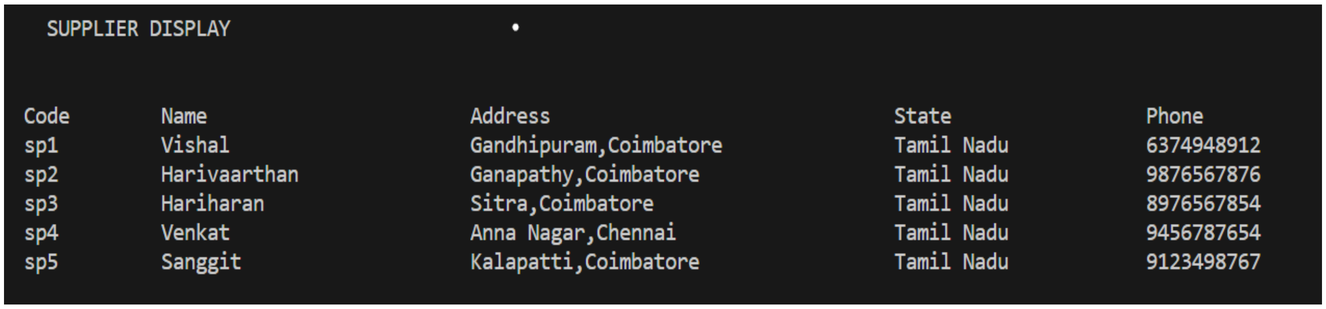


```
  INVENTORY DISPLAY


Item Code    Description                     Qty    Price     Reorder Qty
oo1          RESISTOR                        10000  10.00     15000
oo2          CAPACITOR                       20000  12.00     20000
oo3          LED                             25000  4.00      13000
oo4          ARDUINO MEGA                    12000  5500.00   3000
oo5          BREAD BOARD                     14000  120.00    16000
oo6          JUMPER WIRES                    15000  8.00      15000
oo7          SERVO MOTOR                     10000  300.00    20000
oo8          SOLDERING ION                   12000  550.00    12000
oo9          IR SENSOR                       12000  200.00    23000
oo10         RELAY MODULE                    12000  250.00    12000
```



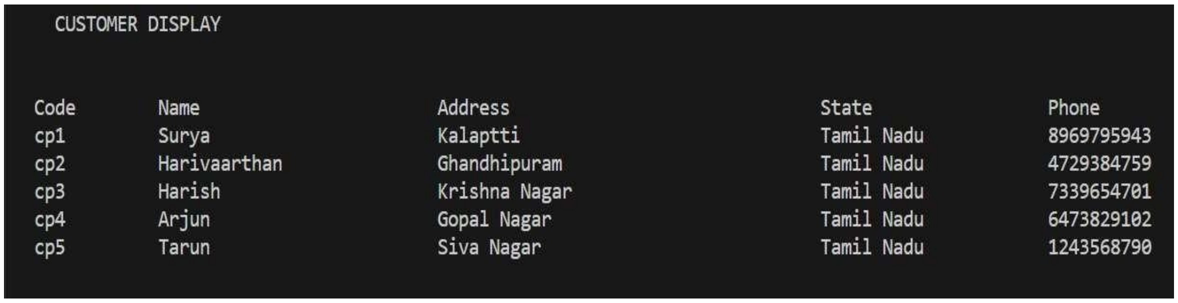
```
|__________________________ELCETRO SHOCK PRO ELECTRONIC SHOP_________________________


 Item Code       Item Description               Quantity   Sale Price Reorder Qty
 it1             RESISTOR(10OHM)                12000      30.00      800
 it2             CAPACITOR                      20000      20.00      1000
 it3             LED                            25000      5.00       13000
 it4             ARDUINO MEGA                   12000      5500.00    3000
 it5             BREAD BOARD                    14000      120.00     16000
 it6             JUMPER WIRES                   15000      8.00       15000
 it7             SERVO MOTORS                   10000      300.00     20000
 it8             SOLDERING ION                  12000      550.00     12000
 it9             IR SENSOR                      12000      200.00     23000
 it10            RELAY MODULE                   12000      250.00     12000
```

7.2 Supplier display:



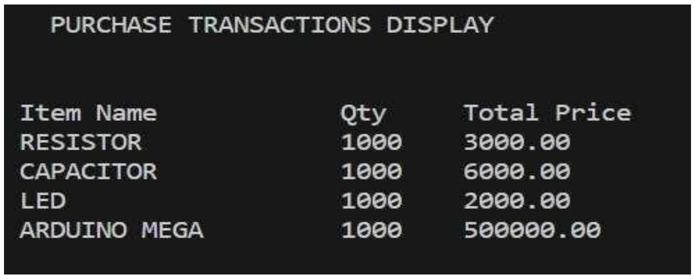
```
  SUPPLIER DISPLAY


Code       Name                Address                        State           Phone
sp1        Vishal              Gandhipuram,Coimbatore         Tamil Nadu      6374948912
sp2        Harivaarthan        Ganapathy,Coimbatore           Tamil Nadu      9876567876
sp3        Hariharan           Sitra,Coimbatore               Tamil Nadu      8976567854
sp4        Venkat              Anna Nagar,Chennai             Tamil Nadu      9456787654
sp5        Sanggit             Kalapatti,Coimbatore           Tamil Nadu      9123498767
```

7.3 Customer display:



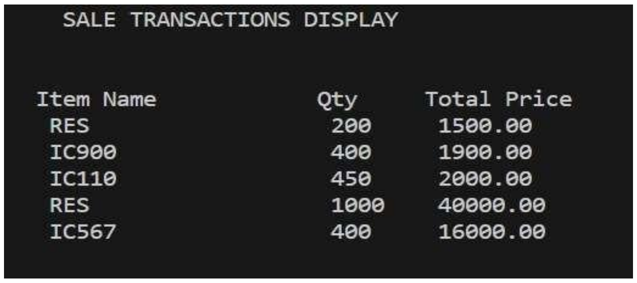
```
  CUSTOMER DISPLAY


Code      Name              Address                    State         Phone
cp1       Surya             Kalaptti                   Tamil Nadu    8969795943
cp2       Harivaarthan      Ghandhipuram               Tamil Nadu    4729384759
cp3       Harish            Krishna Nagar              Tamil Nadu    7339654701
cp4       Arjun             Gopal Nagar                Tamil Nadu    6473829102
cp5       Tarun             Siva Nagar                 Tamil Nadu    1243568790
```

7.4 Purchase transaction display:



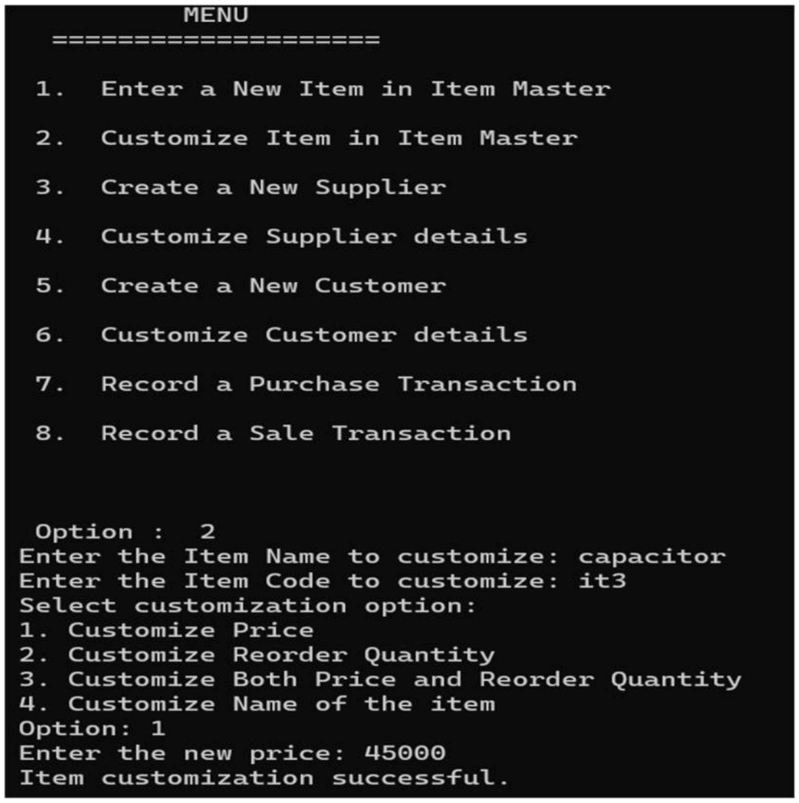
```
  PURCHASE TRANSACTIONS DISPLAY


Item Name            Qty     Total Price
RESISTOR             1000    3000.00
CAPACITOR            1000    6000.00
LED                  1000    2000.00
ARDUINO MEGA         1000    500000.00
```

7.5 Sale transaction display:



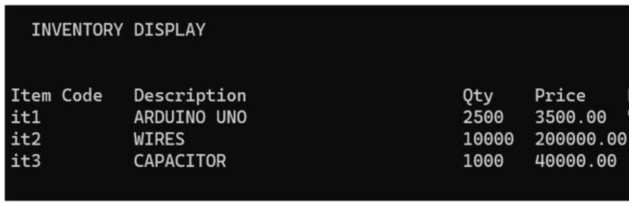
```
   SALE TRANSACTIONS DISPLAY


 Item Name            Qty      Total Price
  RES                  200      1500.00
  IC900                400      1900.00
  IC110                450      2000.00
  RES                  1000     40000.00
  IC567                400      16000.00
```

7.6.Customizing records



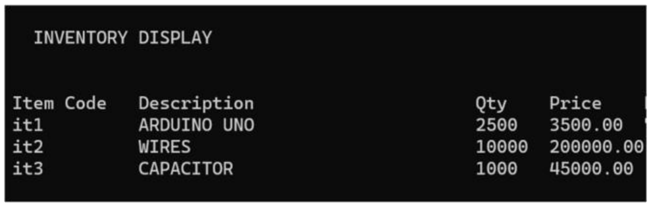
```
          MENU
  ====================

 1.  Enter a New Item in Item Master

 2.  Customize Item in Item Master

 3.  Create a New Supplier

 4.  Customize Supplier details

 5.  Create a New Customer

 6.  Customize Customer details

 7.  Record a Purchase Transaction

 8.  Record a Sale Transaction



 Option :  2
Enter the Item Name to customize: capacitor
Enter the Item Code to customize: it3
Select customization option:
1. Customize Price
2. Customize Reorder Quantity
3. Customize Both Price and Reorder Quantity
4. Customize Name of the item
Option: 1
Enter the new price: 45000
Item customization successful.
```

Initial:



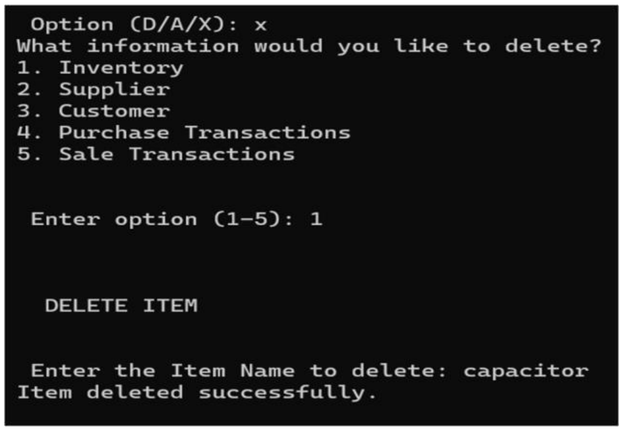
```
 INVENTORY DISPLAY


Item Code   Description                       Qty    Price
it1         ARDUINO UNO                       2500   3500.00
it2         WIRES                             10000  200000.00
it3         CAPACITOR                         1000   40000.00
```

Final:



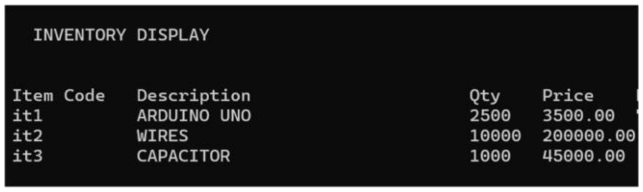
```
 INVENTORY DISPLAY


Item Code   Description                       Qty    Price
it1         ARDUINO UNO                       2500   3500.00
it2         WIRES                             10000  200000.00
it3         CAPACITOR                         1000   45000.00
```

7.7.Deleting records



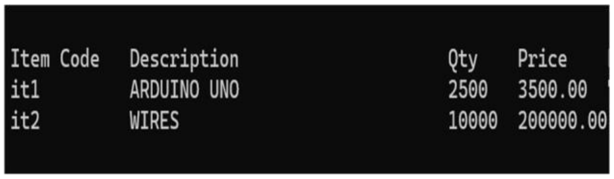
```
 Option (D/A/X): x
What information would you like to delete?
1. Inventory
2. Supplier
3. Customer
4. Purchase Transactions
5. Sale Transactions


 Enter option (1-5): 1



  DELETE ITEM


 Enter the Item Name to delete: capacitor
Item deleted successfully.
```

Initial:



```
  INVENTORY DISPLAY


Item Code   Description                     Qty    Price
it1         ARDUINO UNO                     2500   3500.00
it2         WIRES                           10000  200000.00
it3         CAPACITOR                       1000   45000.00
```

Final:



```
Item Code   Description                     Qty    Price
it1         ARDUINO UNO                     2500   3500.00
it2         WIRES                           10000  200000.00
```

8. Conclusion and Future Work

Our team has gained valuable experience in developing an Inventory Management System for an electronic shop. This project has been an educational platform that allowed us to master the C programming language. As we worked on the project, we gained valuable insights into file handling in C, which is essential for managing and organizing data efficiently. Apart from programming skills, we also learned about the meticulous record-keeping practices of electronic shops. This knowledge not only broadened our perspectives on inventory management but also allowed us to appreciate the real-world applications of the programming skills we were acquiring. We analysed research papers to understand how similar inventory management systems have been implemented in various programming languages. This comparative analysis enriched our understanding and gave us a deeper appreciation for the versatility and adaptability of programming languages across different domains. Our primary objective was to create an effective Inventory Management System, and we are proud to say that we have achieved this goal. This project stands as a testament to our progression in programming, file handling, and a more comprehensive understanding of inventory management principles in the realm of an electronic shop. The knowledge gained from this experience will undoubtedly form a robust foundation for our future pursuits in both programming and business management arenas, showcasing our continuous growth and adaptability in these fields.

To enhance the efficiency of the management system, two key strategies can be employed. Firstly, expanding the range of options will contribute to user-friendliness. This entails incorporating additional features that cater to diverse user needs. Secondly, implementing a graphical user interface (GUI) facilitates seamless navigation through the system's functions. The GUI serves as an accessible platform, offering comprehensive information and easy access to vital functions via intuitive buttons. In Python, Tkinter can be utilized for GUI development, while in C, the GTK application proves instrumental in creating user-friendly windows. Employing these methods ensures a more efficient and user-centric management system.

9. References

[1] Tejesh, B.S.S. and Neeraja, S.J.A.E.J., 2018. Warehouse inventory management system using IoT and open source framework. *Alexandria engineering journal*, *57*(4), pp.3817-3823.

[2] Ahmadi, E., Masel, D.T., Metcalf, A.Y. and Schuller, K., 2019. Inventory management of surgical supplies and sterile instruments in hospitals: a literature review. Health Systems, 8(2), pp.134-151.

[3] Mao, J., Xing, H. and Zhang, X., 2018. Design of intelligent warehouse management system. *Wireless Personal Communications*, *102*, pp.1355-1367.

[4] Tanamal, R., Nurdiansyah, Y. and Firdaus, F., 2020. Inventory Support System for Retail Shop. In *E3S Web of Conferences* (Vol. 188, p. 00020). EDP Sciences.

[5] Deshmukh, R.R., Pawar, A., Katore, V., Dabhade, N., Kasture, V. and Bagul, S., 2021. Inventory Management System.

[6] Saillaja, V., Menaka, M., Kumaravel, V. and Machap, K., 2023, June. Development of an IoT-based Inventory Management System for Retail Stores. In 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS) (pp. 954-958). IEEE.

[7] Macas, C.V.M., Aguirre, J.A.E., Arcentales-Carrión, R. and Peña, M., 2021, March. Inventory management for retail companies: A literature review and current trends. In 2021 Second International Conference on Information Systems and Software Technologies (ICI2ST) (pp. 71-78). IEEE.

[8] Zhao, B. and Tu, C., 2021. Research and development of inventory management and human resource management in ERP. *Wireless Communications and Mobile Computing*, *2021*, pp.1-12.

[9] Koralage, S.I., 2017. *Web based Bookstore Management System for Wisdom* (Doctoral dissertation).

[10] Chebet, E. and Kitheka, S., 2019. Effects of inventory management system on firm performance–an empirical study. International Journal of Innovative Science and Research Technology, 4(9), pp.34-242.