Smartbuy: A product order management system

Course No: EE 1222

Course Title: Computer Fundamentals and Programming

Course Instructors:

Md. Anis Ahmed

Md. Sajjad Hossain

Presented by:

Name:Sanjid Hasan

Roll:2203090

Session:2022-2023

Year:2nd

Date:10 September,2024

Department of Electrical And Electronic Engineering

Khulna University Of Engineering And Technology, Khulna

Objectives:

1. To build a system for easy business management.
2. To create a simple interface for customers to browse and buy products.
3. To keep track of inventory with real-time updates.
4. To add a secure payment option for safe transactions.
5. To improve the customer experience with product suggestions.
6. To make sure the system can handle more users as it grows.
7. To use efficient data structures for quick product searches.

Introduction:

The **SmartBuy** project represents an advanced solution for managing retail operations with a focus on user experience and system efficiency. This project aims to create a seamless interface for customers and a powerful management tool for administrators.

When the application starts, users are greeted with three key options: placing an order, accessing the admin panel, or exiting the system. The ordering process is designed to be intuitive, allowing users to easily select from a dynamically updated price list. Using arrays, the system efficiently handles product data, enabling users to specify quantities and types of items they wish to purchase. After completing an order, the system generates a detailed bill, which is saved in a text file and can be converted into a PDF for convenience.

For administrators, **SmartBuy** offers a secure panel with features for managing product information and viewing transaction logs. This panel allows for real-time updates to the products.txt file, ensuring that the system remains accurate and up-to-date with minimal manual effort.

By integrating file handling, dynamic memory allocation, and array management, **SmartBuy** not only enhances the shopping experience but also showcases a comprehensive understanding of the programming concepts covered in the EE 1222 course. This project highlights practical applications of these techniques, providing a robust and scalable solution for retail management.

Source Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>
#include <string.h>
#define NUM_ITEMS 7
#define FILENAME "products.txt"
#define ADMIN_PASSWORD "1222"

// Global variables
int inStock[NUM_ITEMS];
char foodName[NUM_ITEMS][50];
double price[NUM_ITEMS];

// Function prototypes
void loadProducts();
void saveProducts();
void printFoodTable();
void displayMenu();
void displayAdminMenu();
void printReceipt(int numItems, int f[], int q[], double total);
void logTransaction(int numItems, int f[], int q[], double total);
void viewTransactions();
void editProduct();

int main() {
    loadProducts();
    int choice;

    while (1) {
        displayMenu();
        if (scanf("%d", &choice) != 1) {
            fprintf(stderr, "Error: invalid input.\n");
            while (getchar() != '\n'); // Clear the input buffer
            continue;
        }
```

```
        switch (choice) {
            case 1: {
                printf("You have selected Order Food.\n");
                printFoodTable();
                printf("Enter the number of different items you want to
order: ");
                int numItems;
                if (scanf("%d", &numItems) != 1 || numItems <= 0) {
                    fprintf(stderr, "Error: invalid number of items.\n");
                    while (getchar() != '\n'); // Clear the input buffer
                    continue;
                }

                int *f = (int*)malloc(numItems * sizeof(int));
                int *q = (int*)malloc(numItems * sizeof(int));
                if (f == NULL || q == NULL) {
                    fprintf(stderr, "Error: memory allocation failed.\n");
                    return 1;
                }

                double total = 0.0;

                for (int i = 0; i < numItems; ++i) {
                    printf("Enter the food no. for item %d: ", i + 1);
                    if (scanf("%d", &f[i]) != 1 || f[i] < 1 || f[i] >
NUM_ITEMS) {
                        fprintf(stderr, "Error: invalid food number.\n");
                        while (getchar() != '\n'); // Clear the input
buffer
                        --i;
                        continue;
                    }
                    printf("Enter the quantity for item %d: ", i + 1);
                    if (scanf("%d", &q[i]) != 1 || q[i] < 1) {
                        fprintf(stderr, "Error: invalid quantity.\n");
                        while (getchar() != '\n'); // Clear the input
buffer
                        --i;
                        continue;
                    }
                    if (q[i] > inStock[f[i] - 1]) {
                        fprintf(stderr, "Error: insufficient stock for
```

```
%s.\n", foodName[f[i] - 1]);
                        --i;
                        continue;
                    }
                    total += price[f[i] - 1] * q[i];
                    inStock[f[i] - 1] -= q[i];
                    printf("You have ordered %d %s. Total cost so far:
%.2lf\n", q[i], foodName[f[i] - 1], total);
                }
                printReceipt(numItems, f, q, total);
                logTransaction(numItems, f, q, total);
                saveProducts(); // Save changes to the products file

                free(f);
                free(q);
                break;
            }
            case 2:
                displayAdminMenu();
                break;
            case 3:
                printf("Thanks for visiting.\n");
                return 0;
            default:
                fprintf(stderr, "Error: invalid choice.\n");
                break;
        }
    }

    return 0;
}

// Function to load products from the file
void loadProducts() {
    FILE *file = fopen(FILENAME, "r");
    if (file == NULL) {
        fprintf(stderr, "Error: could not open %s.\n", FILENAME);
        exit(1);
    }

    for (int i = 0; i < NUM_ITEMS; ++i) {
        fscanf(file, "%[^,],%lf,%d\n", foodName[i], &price[i],
```

```
&inStock[i]);
    }

    fclose(file);
}

// Function to save products to the file
void saveProducts() {
    FILE *file = fopen(FILENAME, "w");
    if (file == NULL) {
        fprintf(stderr, "Error: could not open %s.\n", FILENAME);
        exit(1);
    }

    for (int i = 0; i < NUM_ITEMS; ++i) {
        fprintf(file, "%s,%.2lf,%d\n", foodName[i], price[i], inStock[i]);
    }

    fclose(file);
}

// Function to print the food table
void printFoodTable() {
    printf("_____________________________________________________\n");
    printf("|  Food No.  |  Food Name   |  Price    |  In Stock   |\n");
    printf("-----------------------------------------------------\n");

    for (int i = 0; i < NUM_ITEMS; ++i) {
        printf("|    %2d      | %-12s | %6.2lf    |     %3d     |\n", i +
1, foodName[i], price[i], inStock[i]);
        if (i != NUM_ITEMS - 1) {
            printf("+-----------------------------------------------------
+\n");
        }
    }
    printf("+-----------------------------------------------------+\n");
}

// Function to display the main menu
void displayMenu() {
    printf("\nWelcome to our shop. How may I help you?\n");
    printf("1. Order Food\n");
```

```
    printf("2. Admin Panel\n");
    printf("3. Exit\n");
}

// Function to display the admin menu
void displayAdminMenu() {
    char password[50];
    printf("\nEnter admin password: ");
    scanf("%s", password);
    if (strcmp(password, ADMIN_PASSWORD) == 0) {
        int choice;
        printf("\nAdmin Panel:\n");
        printf("1. View Transactions\n");
        printf("2. Edit Product\n");
        printf("3. Return to Main Menu\n");

        if (scanf("%d", &choice) != 1) {
            fprintf(stderr, "Error: invalid input.\n");
            while (getchar() != '\n'); // Clear the input buffer
            return;
        }

        switch (choice) {
            case 1:
                viewTransactions();
                break;
            case 2:
                editProduct();
                saveProducts(); // Save changes to the products file
                break;
            case 3:
                return;
            default:
                fprintf(stderr, "Error: invalid choice.\n");
                break;
        }
    } else {
        printf("Error: incorrect password.\n");
    }
}

void printReceipt(int numItems, int f[], int q[], double total) {
```

```
    FILE *billFile = fopen("bill.txt", "w");
    if (billFile != NULL) {
        time_t t;
        time(&t);
        fprintf(billFile, "----------------------------------------\n");
        fprintf(billFile, "SmartBuy: A Product order Management System\n");
        fprintf(billFile, "----------------------------------------\n");
        fprintf(billFile, "Date: %s", ctime(&t));
        fprintf(billFile, "----------------------------------------\n");
        fprintf(billFile, "Item      Quantity    Price    Total\n");
        fprintf(billFile, "----------------------------------------\n");

        for (int i = 0; i < numItems; ++i) {
            fprintf(billFile, "%-12s %3d  %8.2lf  %8.2lf\n", foodName[f[i]
- 1], q[i], price[f[i] - 1], price[f[i] - 1] * q[i]);
        }

        fprintf(billFile, "----------------------------------------\n");
        fprintf(billFile, "TOTAL                        %8.2lf\n", total);
        fprintf(billFile, "----------------------------------------\n");
        fprintf(billFile, " Thanks for choosing us!           \n");
        fprintf(billFile, "----------------------------------------\n");
        fclose(billFile);
        printf("Bill has been saved to bill.txt\n");
    } else {
        printf("Error creating bill file.\n");
    }
}

// Function to log the transaction
void logTransaction(int numItems, int f[], int q[], double total) {
    FILE *logFile = fopen("transactions.log", "a");
    if (logFile != NULL) {
        time_t t;
        time(&t);
        fprintf(logFile, "----------------------------------------\n");
        fprintf(logFile, "Date: %s", ctime(&t));
        fprintf(logFile, "----------------------------------------\n");
        fprintf(logFile, "Item          Qty    Price    Total\n");
        fprintf(logFile, "----------------------------------------\n");

        for (int i = 0; i < numItems; ++i) {
```

```
        fprintf(logFile, "%-12s %3d   %8.2lf %8.2lf\n", foodName[f[i] - 1],
q[i], price[f[i] - 1], price[f[i] - 1] * q[i]);
        }

        fprintf(logFile, "----------------------------------------\n");
        fprintf(logFile, "TOTAL                        %8.2lf\n", total);
        fprintf(logFile, "----------------------------------------\n");
        fclose(logFile);
    } else {
        printf("Error creating transaction log file.\n");
    }
}

// Function to view transactions
void viewTransactions() {
    FILE *logFile = fopen("transactions.log", "r");
    if (logFile != NULL) {
        char line[256];
        while (fgets(line, sizeof(line), logFile)) {
            printf("%s", line);
        }
        fclose(logFile);
    } else {
        printf("Error opening transaction log file.\n");
    }
}

// Function to edit a product
void editProduct() {
    int choice, newPrice, newStock;
    printFoodTable();
    printf("Enter the food no. to edit: ");
    if (scanf("%d", &choice) != 1 || choice < 1 || choice > NUM_ITEMS) {
        fprintf(stderr, "Error: invalid food number.\n");
        while (getchar() != '\n'); // Clear the input buffer
        return;
    }

    printf("Enter the new price: ");
    if (scanf("%d", &newPrice) != 1 || newPrice < 0) {
        fprintf(stderr, "Error: invalid price.\n");
        while (getchar() != '\n'); // Clear the input buffer
```

```
        return;
    }

    printf("Enter the new stock quantity: ");
    if (scanf("%d", &newStock) != 1 || newStock < 0) {
        fprintf(stderr, "Error: invalid stock quantity.\n");
        while (getchar() != '\n'); // Clear the input buffer
        return;
    }

    price[choice - 1] = newPrice;
    inStock[choice - 1] = newStock;
    printf("Product updated successfully.\n");
}
```

Products.txt : Required text for running the code

```
Pen,226.00,10
Ball,100.00,19
bat,475.00,19
boat,350.00,20
file,125.00,15
Sandwich,150.00,25
Cake,200.00,30
```

The products.txt file is a key part of the **SmartBuy** system. It stores all the information about the products in a simple text format, making it easy to manage and update. Each line in the file represents a product and contains three pieces of information:

1. **Product Name**: The name of the item (e.g., Pen, Ball, Bat).
2. **Price**: How much the item costs (e.g., 226.00 for a Pen).
3. **Quantity**: How many of the item are available (e.g., 10 Pens).

Here's how products.txt helps the **SmartBuy** system:

1. **Easy Updates**: Administrators can quickly change prices, quantities, or add new products by editing this file. This makes it simple to keep the product info current without changing the actual program code.

2. **Current Info**: The system uses products.txt to show up-to-date product details to customers. When someone places an order, the system checks this file to get the latest prices and stock levels.

User-Friendly: Since the file is in a plain text format, anyone can open and edit it with a basic text editor. This means you don't need special software or technical skills to update the product information.

3. **Scalable**: As more products are added or if product details change, you can just update the products.txt file. This makes it easy for the system to grow and adapt to new products or changes in inventory.
4. **Accurate Billing**: The file helps the system generate accurate bills and manage inventory. It ensures that when an order is processed, the system uses the correct prices and updates the stock levels accordingly.

In short, products.txt makes managing product information easy and efficient, which is crucial for running the **SmartBuy** system smoothly.

Output of Code:

```
Welcome to our shop. How may I help you?
1. Order Food
2. Admin Panel
3. Exit
```

After selecting 1,

```
You have selected Order Food.
_____________________________________________________
|  Food No.  |  Food Name   |  Price    |  In Stock   |
-----------------------------------------------------
|     1      | Pen          | 226.00    |      10     |
+----------------------------------------------------+
|     2      | Ball         | 100.00    |      19     |
+----------------------------------------------------+
|     3      | bat          | 475.00    |      19     |
+----------------------------------------------------+
|     4      | boat         | 350.00    |      20     |
+----------------------------------------------------+
|     5      | file         | 125.00    |      15     |
+----------------------------------------------------+
|     6      | Sandwich     | 150.00    |      25     |
+----------------------------------------------------+
|     7      | Cake         | 200.00    |      30     |
+----------------------------------------------------+
Enter the number of different items you want to order:
```

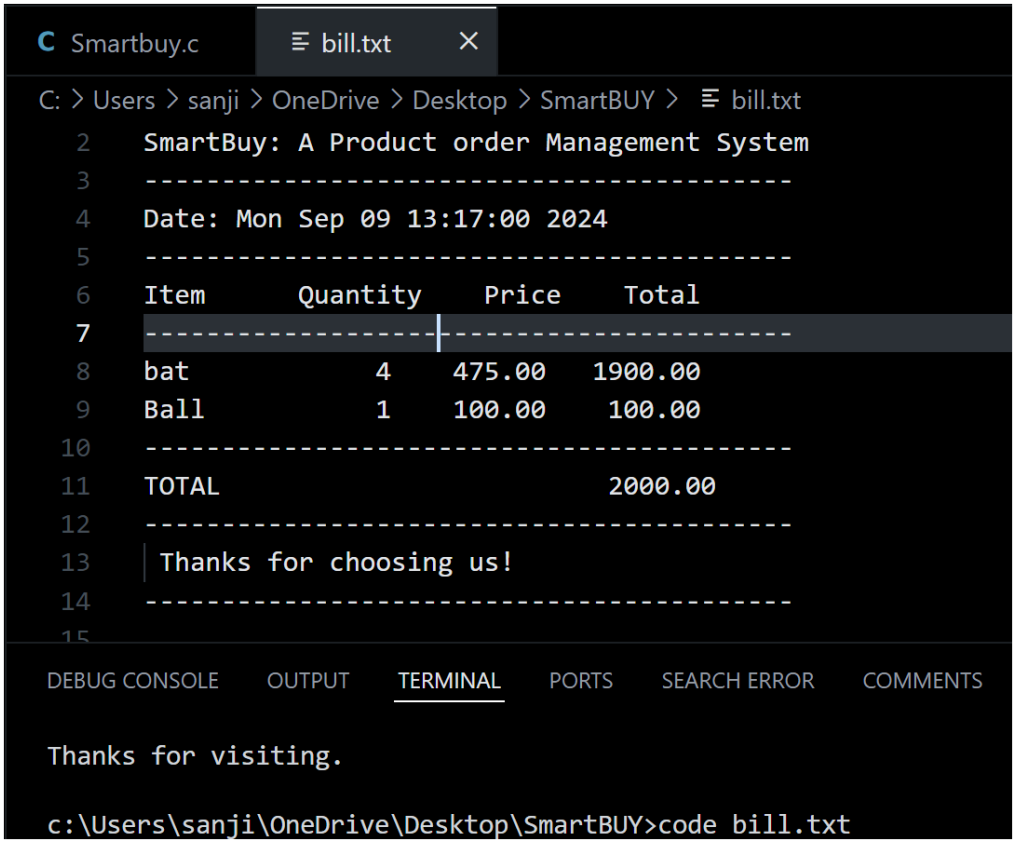
Then selecting products to buy,

```
Enter the number of different items you want to order: 2
Enter the food no. for item 1: 3
Enter the quantity for item 1: 4
You have ordered 4 bat. Total cost so far: 1900.00
Enter the food no. for item 2: 2
Enter the quantity for item 2: 1
You have ordered 1 Ball. Total cost so far: 2000.00
Bill has been saved to bill.txt

Welcome to our shop. How may I help you?
1. Order Food
2. Admin Panel
3. Exit
```

Pressing 3 will lead to exit button.In the meantime, a receipt was printed in bill.txt

Bill.txt



There are more!

You open the admin panel with the provided password '1222'

```
Welcome to our shop. How may I help you?
1. Order Food
2. Admin Panel
3. Exit
2

Enter admin password: 1222

Admin Panel:
1. View Transactions
2. Edit Product
3. Return to Main Menu
```

Selecting 1,will show the logs of transaction,

```
--------------------------------------------------------------------------------
---------
Date: Thu Aug 01 00:46:29 2024
----------------------------------------
```

```
Item          Qty    Price    Total
-----------------------------------------
Sandwich       2     150.00   300.00
-----------------------------------------
TOTAL                         300.00
-----------------------------------------
-----------------------------------------
Date: Thu Aug 01 17:07:56 2024
-----------------------------------------
Item          Qty    Price    Total

candy          2     220.00   440.00
-----------------------------------------
TOTAL                         440.00
-----------------------------------------
-----------------------------------------
Date: Thu Aug 01 22:35:05 2024
-----------------------------------------
Item          Qty    Price    Total
-----------------------------------------
Pen            2     224.00   448.00
-----------------------------------------
TOTAL                         448.00
-----------------------------------------
-----------------------------------------
Date: Thu Aug 01 22:37:37 2024
-----------------------------------------
Item          Qty    Price    Total
-----------------------------------------
candy         22     220.00  4840.00
-----------------------------------------
TOTAL                        4840.00
-----------------------------------------
-----------------------------------------
Date: Wed Aug 21 19:34:14 2024
-----------------------------------------
Item          Qty    Price    Total
-----------------------------------------
Pen            2     224.00   448.00
Ball           1     100.00   100.00
-----------------------------------------
TOTAL                         548.00
-----------------------------------------
```

```
----------------------------------------
Date: Tue Aug 27 00:08:42 2024
----------------------------------------
Item          Qty    Price    Total
----------------------------------------
Pen            2     224.00   448.00
bat            3     475.00  1425.00
Candy          2     880.00  1760.00
----------------------------------------
TOTAL                        3633.00
----------------------------------------
----------------------------------------
Date: Fri Sep 06 21:13:34 2024
----------------------------------------
Item          Qty    Price    Total
----------------------------------------
bat            2     475.00   950.00
Pen            6     224.00  1344.00
----------------------------------------
TOTAL                        2294.00
----------------------------------------
----------------------------------------
Date: Mon Sep 09 10:52:18 2024
----------------------------------------
Item          Qty    Price    Total
----------------------------------------
Pen            2     226.00   452.00
bat            1     475.00   475.00
----------------------------------------
TOTAL                         927.00
----------------------------------------
----------------------------------------
Date: Mon Sep 09 10:54:58 2024
----------------------------------------
Item          Qty    Price    Total
----------------------------------------
Ball           5     100.00   500.00
----------------------------------------
TOTAL                         500.00
----------------------------------------
----------------------------------------
Date: Mon Sep 09 12:51:23 2024
----------------------------------------
```

```
Item          Qty    Price    Total
----------------------------------------
Pen            1     226.00   226.00
Pen            1     226.00   226.00
----------------------------------------
TOTAL                         452.00
----------------------------------------
----------------------------------------
Date: Mon Sep 09 13:17:00 2024
----------------------------------------
Item          Qty    Price    Total
----------------------------------------
bat            4     475.00  1900.00
Ball           1     100.00   100.00
----------------------------------------
TOTAL                        2000.00
----------------------------------------
```

Selecting 2,will give a way to change products price and stock info.

```
Enter the product no. to edit: 1
Enter the new price: 150
Enter the new stock quantity: 25
Product updated successfully.
```

Pressing 3 will help to opt out.

Discussion:

In this project, we developed SmartBuy.c, a smart product management system, to streamline and automate retail operations. The objective was to design a program that could handle common tasks such as managing a price list, handling customer orders, generating bills, and recording transaction logs. The program was implemented using C language and incorporated several key concepts, including arrays, file handling, pointers, loops, dynamic memory allocations, and switch cases.

Key Features and Design Choices:

1. Order and Price Management:

The system uses arrays to store product information and customer orders. Arrays were chosen due to their straightforward implementation and fixed-size allocation, which allows quick access and retrieval of data. Each product is assigned a unique ID, with corresponding prices stored in parallel arrays. This facilitates efficient searching and modification of price and quantity data.

2. File Handling:

To ensure the program could save and retrieve transaction history, we implemented file handling techniques. This feature allows the system to maintain a log of customer transactions, even after the program is closed. We used file input/output operations to store the details of each transaction, such as the items purchased, the total cost, and the date of the transaction. This ensures data persistence and provides the ability to review transaction history at any time.

3. Dynamic Memory Allocation:

In the bill generation process, dynamic memory allocation was used to manage customer orders. Instead of using a fixed array size for storing orders, we utilized pointers and dynamic memory functions such as `malloc` and `free` to allocate memory based on the number of items in a customer's cart. This improves memory efficiency and avoids wastage of resources when fewer products are ordered.

4. Modular Approach with Switch Case:

We structured the program using a modular approach, with functions dedicated to specific tasks like adding products, processing orders, generating bills, and logging transactions. This makes the code more readable and easier to maintain. A switch case was used to handle the main menu of the program, allowing users to choose between different operations such as placing orders, generating bills, and viewing transaction logs. This enhances the user experience and makes the interface intuitive.

5. Error Handling:

Several error-handling mechanisms were implemented to manage invalid inputs, such as incorrect product IDs or order quantities exceeding stock. These checks are essential to maintain the robustness of the program and prevent unexpected crashes or incorrect billing.

Challenges Faced:

One of the major challenges in this project was memory management, especially while dealing with dynamic arrays for handling orders. We encountered issues with memory leaks initially, which we solved by carefully placing `free()` calls after memory was no longer needed. Additionally, designing an efficient structure to manage product IDs and prices required careful thought to ensure the program could handle edge cases like invalid inputs.

Another challenge was file handling, particularly ensuring that the transaction log was updated correctly after each operation and that the file was opened and closed appropriately to prevent data loss.

Conclusion:

The **SmartBUY** project helped us learn how to use basic programming concepts like arrays, file handling, and loops to create a simple product management system. We were able to build a program that processes orders, generates bills, and keeps track of transactions. This project gave us a better understanding of C programming and how it can be used to solve real-life problems in managing products. In the future, we could improve the program by adding more features or making it easier to use.