Department of Artificial Intelligence

22AIE101: Problem Solving & C Programming, Fall 2023

Project Report

Canteen Management System

Team Members:

Shreya Arun: CB.SC.U4AIE23253
Siri Sanjana S: CB.SC.U4AIE23249
Varshitha Thilak kumar: CB.SC.U4AIE23258

Supervised By:

Dr. Vidhya Kamakshi
Assistant Professor
Department of Artificial Intelligence
Amrita Vishwa Vidyapeetham

Date of submission: <26/12/2023>
Signature of the Project Supervisor:

TABLE OF CONTENTS

Abstract

This report documents the design and implementation of the Canteen Management System using C-programming language. The design is implemented to streamline various aspects of the Canteen Management System like management efficiency and accuracy in menu items management, order processing, and inventory control units. This is an automated interface with key functionalities such as Displaying the menu, adding new items, refilling item quantities, searching for items, placing and deleting orders, feedback and exiting the system. This highlights the significance of the system in improving management working efficiency, providing a reliable platform, and reducing manual errors. the Canteen Management System is a practical solution that automates and simplifies canteen tasks, offering a user-friendly interface for both administrators and customers.

1 Introduction

In the changing world of corporate environments managing canteen facilities effectively is crucial, for providing a seamless experience to administrators and customers alike. This report presents the creation and implementation of a Canteen Management System that has been developed using the C programming language. The main goal of this system is to streamline and optimize day to day canteen operations. Recognizing the need for a solution our Canteen Management System has been designed with modularity in mind offering flexibility and adaptability in operational scenarios. Users can access the system through a to use console interface, which caters to both menu administrators and customers placing orders.

The key features of this system include displaying the canteen menu adding and restocking items, efficient search functionality, smooth order placement and monitoring as removing outdated items. These features collectively contribute to enhancing menu management, inventory control and overall operational efficiency. This report will provide an exploration of the development process, design considerations and user centered attributes integrated into our Canteen Management System. By adopting a user approach with features our aim is to simplify tasks reduce errors and provide a versatile platform, for efficient canteen management.

2 Contributions:

- **Shreya Arun:** Implementing function for displaying menu, adding item, searching item and main function of txt and csv file.

- **Siri Sanjana :** Implementing function for placing order, display the placed order and txt file.

- **Varshitha Thilak kumar:** Implementing function for refill of an item, delete food item,main function using switch cases and csv file.

3 Methodology

The program allows the user to display the menu, add food items, search for food items, place orders, display the placed order, refill items in the menu, delete any order, and finally exit the program. The menu is initialized as serial number, food item name, price per quantity, and quantity available in the canteen. when the user chooses the option “Display menu”, it Enables users to view the available items on the canteen menu, including details such as item name, price, and quantity. When the user chooses the option “Add Item”, It Allows administrators to add new items to the canteen menu, specifying the item name, price, and initial quantity.When the administrator wants to replenish the quantity of a particular item on the menu, ensuring that the canteen is well-stocked he/she should choose the option “Refill Item”.“Search item” option When the user wants to search for a particular food item from the menu to get quick access to its availability and price.“Place Order” option allows customers to place orders by selecting items from the menu, specifying the quantity of each item, and calculating the total order amount. To display the details of current orders, including item names, quantities, and total amounts, assisting in order management the user should choose the option called “Display Order”. The “Delete Item” option Permits administrators to remove items from the menu, ensuring that outdated or unavailable items are appropriately managed. The “feedback” option allows the user to promt feedback to the management system and allows the management to improve the efficiency.

Finally “Exit” option Allows users to exit the system gracefully, ensuring a smooth and controlled termination of the application.

3.1.PSEUDOCODES:

1.DISPLAYING MENU:

```
Function displayMenu(menu[], itemCount):
    print "Menu:"
    for i = 0 to itemCount:
        print item details
        if menu[i].quantity < 5:
            print "Warning: Quantity of menu[i].name is low."
```

2.ADD ITEM:

```
Function addItem(menu[], itemCount):
    input itemName
    input itemPrice
    input itemQuantity
    menu[itemCount].name = itemName
    menu[itemCount].price = itemPrice
    menu[itemCount].quantity = itemQuantity
    itemCount++
```

3.SEARCH ITEM:

```
Function searchItem(menu[], itemCount):
  input searchItem from user

  for i = 0 to itemCount:
     if menu[i].name equals searchItem:
        print "menu[i].name is in the menu with a price of Rs menu[i].price"
        return

  print "searchItem is not in the menu."
```

4.REFILL ITEM:

```
Function refillItem(menu[], itemCount):
  displayMenu(menu,  itemCount)
  input choice from user

  if choice is invalid:
     print "Invalid choice. Please try again."
  else:
     input quantity from user
```

```
    menu[choice - 1].quantity += quantity
```

5. PLACING ORDER:

```
Function placeOrder(menu[], itemCount, orders[], orderQuantities[], subtotals[], orderCount):
    displayMenu(menu, itemCount)

    input choice from user

    while choice is not 0:
        if choice is invalid:
            print "Invalid choice. Please try again."
        else:
            input quantity from user

            if quantity > menu[choice - 1].quantity:
                print "Sorry, we only have menu[choice - 1].quantity of menu[choice - 1].name left in stock."
            else:
                add order details to orders, orderQuantities, and subtotals
                reduce menu[choice - 1].quantity by quantity

        input choice from user
```

6. DISPLAY ORDER:

```
Function displayOrder(orders[], orderQuantities[], subtotals[], orderCount):
    total = 0
    tax = 0

    print "Order:"
    print formatted order details

    for i = 0 to orderCount:
        calculate subtotal and add to total

    calculate tax
    add tax to total

    print tax and total
```

7. DELETE ORDER:

```
Function deleteItem(menu[], itemCount):
    displayMenu(menu, itemCount)
    input choice from user

    if choice is invalid:
        print "Invalid choice. Please try again."
    else:
```

```
        shift menu items to remove the chosen item
        itemCount—
```

8. FEEDBACK:

```
function getFeedback(feedback: Feedback):
    display("Enter your name: ")
    feedback.customerName = input()  // User inputs their name

    display("Rate the quantity (1-5): ")
    feedback.quantityRating = parseInt(input()) // User rates quantity on a scale of 1-5

    display("Rate the quality (1-5): ")
    feedback.qualityRating = parseInt(input()) // User rates quality on a scale of 1-5

    display("Rate the service (1-5): ")
    feedback.serviceRating = parseInt(input()) // User rates service on a scale of 1-5

    display("Any additional comments: ")
    feedback.comments = input()  // User provides additional comments, reads until newline is found
function saveFeedbackToFile(feedback: Feedback):
    file = openFile("feedback.txt", "a")  // Open the file in append mode
    if file is null:
        display("Error opening file for writing.")
        return
```

FILE HANDLING:

CSV FILE

1. Function loadMenuFromFile(menu[], itemCount):

```
    file = openFile("menu.csv", "r")
    if file is NULL:
        print "Error opening file for reading. Creating a new file."
        return
    itemCount = 0
    while read line from file:
        parse line and assign values to menu[itemCount]
        itemCount++

    closeFile(file)
```

2. Function loadMenuFromFile(menu[], itemCount):

```
file = openFile("menu.csv", "r")
if file is NULL:
    print "Error opening file for reading. Creating a new file."
    return
itemCount = 0
while read line from file:
    parse line and assign values to menu[itemCount]
    itemCount++

closeFile(file)
```

Txt files:

1. function loadMenuFromFile(menu, itemCount):

```
file = openFile("cms.txt", "r")

if file is null:
    print("Error opening file for reading. Creating a new file.")
    return
itemCount = 0
while readLineFromFile(file):
    readResult = fscanf(file, "%s %f %d", menu[itemCount].name, &menu[itemCount].price,
&menu[itemCount].quantity)

    if readResult equals 3:
        itemCount++
    else:
        break

closeFile(file)
```

2. function saveMenuToFile(menu, itemCount):

```
file = openFile("menu.txt", "w")

if file is null:
```

```
    print("Error opening file for writing.")
    return

 for each item in menu:
    writeToFile(file, "%s %.2f %d\n", item.name, item.price, item.quantity)

 closeFile(file)
```

4 Experiments

4.1 Header Files

Purpose: Includes necessary header files that provides standard functionality and access to core system operations.

Explanation

- #include<stdio.h>:

  This header file is an abbreviation for "standard input-output header" and contains routines for input and output operations. This header file declares functions such as printf, scanf, fopen, fclose, and so on. The printf and scanf functions are used in the code to display output and take user input, respectively.

- #include<strcpy>:

  This header file contains string manipulation routines. This header file declares functions such as strcpy (copy strings), strcmp (compare strings), and so on. String manipulations are performed in the code using the strcpy and strcmp routines.The very existence of these header files enables the code to make use of the functions and features defined in them. It's important to remember that these header files are part of the standard C library and are frequently used in C programs.

4.2.Structures

Explanation

A user-defined data type called a structure in C programming is used to combine various variable types under one name. A structure consists of contiguous memory addresses where each member can be of a different data type. Data can be more meaningfully arranged and represented thanks to structures.

The use of structures in this code allows for a more organized and modular representation of menu items, making it easier to manage and manipulate data related to the canteen management system.

A structure called MenuItem is defined in the code that has been provided. There are three people in this structure:

- **name:** The name of a menu item is represented as an array of characters (char), up to a maximum size of 50.
- **price:** A floating-point (float) value that indicates a menu item's cost.
- **quantity:** An integer (int) that indicates how much of a menu item there is.
- **Feedback**

4.3.Global Variables

Global variables are variables declared outside of any function in the code provided; they are often declared at the start of the file and before the main function. Because global variables are accessible to all functions inside the same file, their scope spans the entire program.

Justification

EXPLANATION

- The first is {struct MenuItem menu[100]}: The menu items are shown as an array of {MenuItem} objects. Up to 100 menu items' worth of information can be stored in it.
- {int itemCount = 0}:The number of items in the menu is tracked by this variable.When items are loaded from the file or added during the program's execution, it is updated from its starting value of 0.

- char orders[100][50]: The names of the objects that have been ordered are stored in this 2D array.
- int orderQuantities[100]: The quantities that go with the ordered goods are kept in this array.
- float subtotals[100]: Each ordered item's subtotal (price * quantity) is kept in this array.
- int orderCount = 0: This variable counts the quantity of orders that have been placed. It starts at 0 and increases as orders are placed while the software is running.

These global variables hold data that must be accessed and changed by several functions as well as the program's current state. Nonetheless, it's usually accepted as best practice to avoid utilizing as many global variables as possible and instead to utilize parameters and return values to transfer data between functions.

4.4.Functions

Explanation

The provided code contains several functions that perform specific tasks related to a Canteen Management System. Here's an explanation of each function:

- 1. loadMenuFromFile

  Purpose: Reads menu information from a file ("cms.txt") and populates the `menu` array.

  Parameters: `struct MenuItem menu[]` - an array of `MenuItem` structures, and `int *itemCount` - a pointer to the variable tracking the number of items.

  Behavior: Opens the file, reads each line containing menu item information, and updates the menu array. If the file doesn't exist, it prints an error message.

- 2. SaveMenuToFile:

Purpose: Writes menu information to a file ("menu.txt").

Parameters: struct MenuItem menu[] - an array of MenuItem structures, and int itemCount - the number of items in the menu.

Behavior:Opens the file and writes each menu item's information to a new line in the file.

- 3. displayMenu:

  Purpose: Displays the menu along with quantity information and warns if the quantity is low.

  Parameters: `struct MenuItem menu[]` - an array of `MenuItem` structures, and `int itemCount` - the number of items in the menu.

  Behavior: Iterates through the menu array and prints each item's name, price, and quantity. If the quantity is less than 5, a warning is displayed.

- 4. addItem:

  Purpose: Adds a new item to the menu.

  Parameters: `struct MenuItem menu[]` - an array of `MenuItem` structures, and `int *itemCount` - a pointer to the variable tracking the number of items.

  Behavior:Prompts the user to enter information for the new item (name, price, quantity) and adds it to the menu.

- 5. searchItem:

  Purpose:Searches for an item in the menu by name.

  Parameters: `struct MenuItem menu[]` - an array of `MenuItem` structures, and `int itemCount` - the number of items in the menu.

  Behavior: Prompts the user to enter the name of an item, searches the menu, and prints whether the item is in the menu along with its price.

- 6. refillItem:

  Purpose:Refills the quantity of a menu item.

  Parameters: `struct MenuItem menu[]` - an array of `MenuItem` structures, and `int itemCount`

- the number of items in the menu.

Behavior: Displays the menu, prompts the user to choose an item to refill, and increases its quantity based on user input.

- 7. placeOrder:

  Purpose: Takes orders from the user and updates order-related arrays.

  Parameters: `struct MenuItem menu[]` - an array of `MenuItem` structures, and several arrays related to orders.

  Behavior: Displays the menu, prompts the user to choose items and quantities for an order until the user enters 0. Updates order-related arrays and reduces item quantities in the menu.

- 8. displayOrder

  Purpose:Displays details of an order, including item names, prices, quantities, subtotals, tax, and total.

  Parameters: Several arrays related to orders and their counts.

  Behavior: Prints a detailed summary of the order, including calculated tax and total cost.

- 9.deleteItem:

  Purpose: Deletes an item from the menu.

  Parameters:`struct MenuItem menu[]` - an array of `MenuItem` structures, and `int *itemCount` - a pointer to the variable tracking the number of items.

  Behavior: Displays the menu, prompts the user to choose an item to delete, and removes it from the menu.

- 10. feedback:

Purpose:adds feedback to the management from the user

Parameter: name,quantity,quality and service to rate (1 to 5) and extra comments if any

- 11.main:

Purpose: The main function where the program execution begins. It presents a menu to the user and calls the corresponding functions based on user input.

Variables: Declares several global arrays and variables to store menu information and order details.

- These functions collectively implement a simple canteen management system with features like displaying the menu, adding items, refilling items, searching for items, placing orders, displaying orders, and deleting items.

4.5.Main Function

Purpose: The initiating point of the program where execution starts. It initializes statistics, furnishes a user interface, and calls functions supported by user selection. Decisions are made based on a sequence of options provided to the user.

- Statements with Variables:

1. struct MenuItem menu[100]: An array used to hold menu items, with a structure giving the name, price, and quantity for each item.
2. Tracks the quantity of items in the menu using int itemCount = 0.
3. orders for chars[100][50]: An array to hold item names in a list order.
4. not in chronological orderamounts[100]: An array used to hold item amounts in a hierarchy.
5. float subtotals[100]: An array where each item in an order's subtotal (price * quantity) is stored.
6. Tracks how many items are in an order using int orderCount = 0.

- Menu Loading from File:

The menu array is populated by reading menu items from a file called "cms.txt" using the loadMenuFromFile function.

- Menu Presentation and Loop of User Interaction:

The user is shown with a menu of alternatives until they select option 0, which causes the software to begin a do-while loop.The user is asked to enter their selection.

- Statement of Switch:

When a user selects a choice, a switch statement responds by either calling the appropriate routines or giving an error message.

- Function Invocations for Menu Items:

Different functions are invoked to carry out distinct actions based on the user's selection:

1. showMenu: Show the active menu.
2. addItem: Expand the menu with a new item.
3. refillItem: Raise a menu item's serving size.
4. searchItem: Use the name of a menu item to find it.
5. placeOrder: Choose items from the menu to create a new order.
6. displayOrder: Shows the current order's items and total cost.
7. deleteItem: Takes a menu item off.
8. Feedback:gives feedback to the management from the user
9. Leaving the Program:The program outputs a message and ends the loop if the user selects to exit (enters 0).
10. Menu Saving to File:The function saveMenuToFile is called to save the current menu to a file called "menu.txt" before the program ends.
11. Statement of Return: When the main function returns 0, it means that the program has run successfully.

    In general, the primary role acts as the major hub for organizing several menu and order-related tasks, communicating with users, and maintaining the Canteen Management System.

Explanation

- This program's primary purpose is to act as a gateway and coordinate the operation of a canteen management system.

4.6.Data Initialization

- Runtime user input is the main method used to initialize data. Nevertheless, for testing reasons or as a starting point, you can also initialize some default data directly in the menu array within the main method.

Explanation

- In the function we initialize the menu information by using arrays that item names price and quantity.

File Handling

This step involves managing file operations such as reading from and writing to a file the details persistently.

Explanation

For the Canteen Management System to manage persistent data, like menu items and order details, effective file management is essential. File operations are made easier by two essential functions: loadMenuFromFile and saveMenuToFile.

- saveMenuToFile: Saving Menu to File
- The saveMenuToFile method is in charge of sending the menu's current state to a text file called "menu.txt." It formats and stores the names, prices, and quantities of the menu items as iteratively navigates through them. By doing this, you may be sure that the menu data will be kept for use in subsequent program sessions.

- Loading Menu from File (loadMenuFromFile): During program startup, the equivalent function, loadMenuFromFile, pulls information from the "menu.txt" file. The details of the stored items are added to the menu array if the file is present. A new, blank menu is created by the program when the file is missing.

1. File Opening: The "menu.txt" file is attempted to be opened in read mode ("r") by the function. The program starts reading the file's contents if it is there. In the event that the file is absent, the function tells the user and creates a fresh, blank menu.

2. Reading Menu Data: The function reads data from the file using a loop and the fscanf function. The name, price, and quantity of each menu item are extracted. To maintain track of the number of menu items, the loop iterates through the file until it reaches the end, incrementing the itemCount.

3. File Closure: To save up system resources, the function closes the file after reading

CODE USING TXT FILE:

```
#include <stdio.h>
#include <string.h>

struct MenuItem {
   char name[50];
   float price;
   int quantity;
};
struct Feedback {
   char customerName[50];
   int quantityRating; // Rating from 1 to 5
   int qualityRating; // Rating from 1 to 5
   int serviceRating; // Rating from 1 to 5
   char comments[200];
};

void getFeedback(struct Feedback *feedback) {

   printf("Enter your name: ");
   scanf("%s", feedback->customerName);
   printf("Rate the quantity (1-5): ");
```

```
    scanf("%d", &feedback->quantityRating);
    printf("Rate the quality (1-5): ");
    scanf("%d", &feedback->qualityRating);
    printf("Rate the service (1-5): ");
    scanf("%d", &feedback->serviceRating);
    printf("Any additional comments: ");
    scanf(" %[^\n]s", feedback->comments); // Read until newline is found
}

void saveFeedbackToFile(struct Feedback *feedback) {
    FILE *file = fopen("feedback.txt", "a");
    if (file == NULL) {
        printf("Error opening file for writing.\n");
        return;
    }
    fprintf(file,"name quantity(1-5) quality(1-5) service(1-5) comments \n", feedback-
>customerName, feedback->quantityRating, feedback->qualityRating, feedback->serviceRating,
feedback->comments);

    fprintf(file," %s %d %d %d %s\n \t", feedback->customerName, feedback-
>quantityRating, feedback->qualityRating, feedback->serviceRating, feedback->comments);

    fclose(file);
}
void loadMenuFromFile(struct MenuItem menu[], int *itemCount) {
    FILE *file = fopen("cms.txt", "r");
    if (file == NULL) {
        printf("Error opening file for reading. Creating a new file.\n");
        return;
    }

    *itemCount = 0;

    while (fscanf(file, "%s %f %d", menu[*itemCount].name, &menu[*itemCount].price,
&menu[*itemCount].quantity) == 3) {
        (*itemCount)++;
    }

    fclose(file);
}

void saveMenuToFile(struct MenuItem menu[], int itemCount) {
    FILE *file = fopen("menu.txt", "w");
    if (file == NULL) {
        printf("Error opening file for writing.\n");
        return;
    }
```

```
int i;
    for (i = 0; i < itemCount; i++) {
        fprintf(file, "%s %.2f %d\n", menu[i].name, menu[i].price, menu[i].quantity);
    }

    fclose(file);
}

void displayMenu(struct MenuItem menu[], int itemCount) {
    printf("\nMenu:\n");
    int i;
    for (i = 0; i < itemCount; i++) {
            if (menu[i].quantity!=0)
            {


        printf("%d. %s - Rs %.5f - Quantity: %d\n", i + 1, menu[i].name, menu[i].price, menu[i].quantity);

        if (menu[i].quantity < 5) {
            printf("Warning: Quantity of %s is low.\n", menu[i].name);
        }
        }
    }
}

void addItem(struct MenuItem menu[], int *itemCount) {
    printf("Enter the name of the new item: \n");
    scanf("%s", menu[*itemCount].name);
    printf("Enter the price of the new item: \n");
    scanf("%f", &menu[*itemCount].price);
    printf("Enter the quantity of the new item: \n");
    scanf("%d", &menu[*itemCount].quantity);

    (*itemCount)++;
}

void searchItem(struct MenuItem menu[], int itemCount) {
    char searchItem[50];
    int found = 0;

    printf("Enter the name of the item to search: ");
    scanf("%s", searchItem);
int i;
    for (i = 0; i < itemCount; i++) {
        if (strcmp(menu[i].name, searchItem) == 0) {
            printf("%s is in the menu with a price of Rs %.5f\n", menu[i].name, menu[i].price);
            found = 1;
```

```
            break;
        }
    }

    if (!found) {
        printf("%s is not in the menu.\n", searchItem);
    }
}

void refillItem(struct MenuItem menu[], int itemCount) {
    int choice, quantity;

    displayMenu(menu, itemCount);

    printf("Enter the item number to refill: ");
    scanf("%d", &choice);

    if (choice < 1 || choice > itemCount) {
        printf("Invalid choice. Please try again.\n");
    } else {
        printf("Enter the quantity to add: ");
        scanf("%d", &quantity);

        menu[choice - 1].quantity += quantity;
    }
}

void placeOrder(struct MenuItem menu[], int itemCount, char orders[][50], int orderQuantities[], float
subtotals[], int *orderCount) {
    int choice, quantity;

    displayMenu(menu, itemCount);

    printf("Enter the item number to order (0 to finish): ");
    scanf("%d", &choice);

    while (choice != 0) {
        if (choice < 1 || choice > itemCount) {
            printf("Invalid choice. Please try again.\n");
        } else {
            printf("Enter the quantity: ");
            scanf("%d", &quantity);

            if (quantity > menu[choice - 1].quantity) {
                printf("Sorry, we only have %d of %s left in stock.\n", menu[choice - 1].quantity, menu[choice -
1].name);
            } else {
```

```
            strcpy(orders[*orderCount], menu[choice - 1].name);
            orderQuantities[*orderCount] = quantity;
            subtotals[*orderCount] = quantity * menu[choice - 1].price;

            menu[choice - 1].quantity -= quantity;

            (*orderCount)++;
        }
    }

    printf("Enter the item number to order (0 to finish): ");
    scanf("%d", &choice);
  }
}

void displayOrder(char orders[][50], int orderQuantities[], float subtotals[], int orderCount) {
  float total = 0;
  float tax;

  printf("\nOrder:\n");
  printf("%-20s|%-20s|%-20s|%-20s\n", "NAME", "PRICE FOR 1", "QUANTITY", "PRICE");
  printf("--------------------------------------------------------------------------------\n");
  int i;
       for ( i = 0; i < orderCount; i++) {
    printf("%-20s|%-20.2f|%-20d|%-20.2f\n", orders[i], subtotals[i] / orderQuantities[i],
orderQuantities[i], subtotals[i]);
    total += subtotals[i];
  }
  tax = total * 0.12; // Calculate 12% tax
  total += tax;      // Add tax to total

  printf("--------------------------------------------------------------------------------\n");
  printf("Tax: Rs%.2f\n", tax);
  printf("Total: Rs%.2f\n", total);
}

void deleteItem(struct MenuItem menu[], int *itemCount) {
  int choice, i;

  displayMenu(menu, *itemCount);

  printf("Enter the item number to delete: ");
  scanf("%d", &choice);

  if (choice < 1 || choice > *itemCount) {
    printf("Invalid choice. Please try again.\n");
  } else {
```

```
        for (i = choice - 1; i < *itemCount - 1; i++) {
            strcpy(menu[i].name, menu[i + 1].name);
            menu[i].price = menu[i + 1].price;
            menu[i].quantity = menu[i + 1].quantity;
        }
        (*itemCount)--;
    }
}
void giveFeedback() {
    struct Feedback feedback;
    getFeedback(&feedback);
    saveFeedbackToFile(&feedback);
    printf("Thank you for your feedback!\n");
}


int main() {
    struct MenuItem menu[100];
    int itemCount = 0;
    char orders[100][50];
    int orderQuantities[100];
    float subtotals[100];
    int orderCount = 0;

    loadMenuFromFile(menu, &itemCount);


    int choice;
    do {
        printf("\nCanteen Management System\n");
        printf("1. Display Menu\n");
        printf("2. Add Item\n");
        printf("3. Refill Item\n");
        printf("4. Search Item\n");
        printf("5. Place Order\n");
        printf("6. Display Order\n");
        printf("7. Delete Item\n");
        printf("8. Feedback\n");
        printf("0. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                displayMenu(menu, itemCount);
                break;
            case 2:
```

```
            addItem(menu, &itemCount);
            break;
        case 3:
            refillItem(menu, itemCount);
            break;
        case 4:
            searchItem(menu, itemCount);
            break;
        case 5:
            placeOrder(menu, itemCount, orders, orderQuantities, subtotals, &orderCount);
            break;
        case 6:
            displayOrder(orders, orderQuantities, subtotals, orderCount);
            break;
        case 7:
            deleteItem(menu, &itemCount);
            break;
        case 8:
             giveFeedback();
            break;
        case 0:
            printf("Exiting the program. Thank you!\n");
            break;
        default:
            printf("Invalid choice. Please try again.\n");
      }
   } while (choice != 0);

   // Before exiting the program, save the menu to the file
   saveMenuToFileFile(menu, itemCount);



   return 0;
}
```

CODE USING CSV FILE:

```
#include <stdio.h>
#include <string.h>

struct MenuItem {
   char name[50];
   float price;
   int quantity;
};
```

```
struct Feedback {
    char customerName[50];
    int quantityRating; // Rating from 1 to 5
    int qualityRating; // Rating from 1 to 5
    int serviceRating; // Rating from 1 to 5
    char comments[200];
};

void getFeedback(struct Feedback *feedback) {
    printf("Enter your name: ");
    scanf("%s", feedback->customerName);
    printf("Rate the quantity (1-5): ");
    scanf("%d", &feedback->quantityRating);
    printf("Rate the quality (1-5): ");
    scanf("%d", &feedback->qualityRating);
    printf("Rate the service (1-5): ");
    scanf("%d", &feedback->serviceRating);
    printf("Any additional comments: ");
    scanf(" %[^\n]s", feedback->comments); // Read until newline is found
}

void saveFeedbackToFile(struct Feedback *feedback) {
    FILE *file = fopen("feedback.csv", "a");
    if (file == NULL) {
        printf("Error opening file for writing.\n");
        return;
    }

     fprintf(file,"name,quantity(1-5),quality(1-5),service(1-5), comments \n", feedback->customerName,
feedback->quantityRating, feedback->qualityRating, feedback->serviceRating, feedback->comments);
    fprintf(file, "%s,%d,%d,%d,%s\n", feedback->customerName, feedback->quantityRating, feedback-
>qualityRating, feedback->serviceRating, feedback->comments);

    fclose(file);
}

void loadMenuFromFile(struct MenuItem menu[], int *itemCount) {
    FILE *file = fopen("menu.csv", "r");
    if (file == NULL) {
        printf("Error opening file for reading. Creating a new file.\n");
        return;
    }

    *itemCount = 0;

    while (fscanf(file, "%49[^,],%f,%d\n", menu[*itemCount].name, &menu[*itemCount].price,
&menu[*itemCount].quantity)==3) {
```

```
        (*itemCount)++;
    }

    fclose(file);
}

void saveMenuToFile(struct MenuItem menu[], int itemCount) {
    FILE *file = fopen("menu.csv", "w");
    if (file == NULL) {
        printf("Error opening file for writing.\n");
        return;
    }
int i;
    for (i = 0; i < itemCount; i++) {
        fprintf(file, "%s,%.2f,%d\n", menu[i].name, menu[i].price, menu[i].quantity);
    }

    fclose(file);
}

void displayMenu(struct MenuItem menu[], int itemCount) {
    printf("\nMenu:\n");
    int i;
    for (i = 0; i < itemCount; i++) {
          if (menu[i].quantity!=0)
          {


        printf("%d. %s - Rs %.5f - Quantity: %d\n", i + 1, menu[i].name, menu[i].price, menu[i].quantity);

        if (menu[i].quantity < 5) {
            printf("Warning: Quantity of %s is low.\n", menu[i].name);
        }
        }
    }
}

void addItem(struct MenuItem menu[], int *itemCount) {
    printf("Enter the name of the new item: ");
    scanf("%s", menu[*itemCount].name);
    printf("Enter the price of the new item: ");
    scanf("%f", &menu[*itemCount].price);
    printf("Enter the quantity of the new item: ");
    scanf("%d", &menu[*itemCount].quantity);

    (*itemCount)++;
}
```

```
void searchItem(struct MenuItem menu[], int itemCount) {
    char searchItem[50];
    int found = 0;

    printf("Enter the name of the item to search: ");
    scanf("%s", searchItem);
int i;
    for (i = 0; i < itemCount; i++) {
        if (strcmp(menu[i].name, searchItem) == 0) {
            printf("%s is in the menu with a price of Rs %.5f\n", menu[i].name, menu[i].price);
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("%s is not in the menu.\n", searchItem);
    }
}

void refillItem(struct MenuItem menu[], int itemCount) {
    int choice, quantity;

    displayMenu(menu, itemCount);

    printf("Enter the item number to refill: ");
    scanf("%d", &choice);

    if (choice < 1 || choice > itemCount) {
        printf("Invalid choice. Please try again.\n");
    } else {
        printf("Enter the quantity to add: ");
        scanf("%d", &quantity);

        menu[choice - 1].quantity += quantity;
    }
}

void placeOrder(struct MenuItem menu[], int itemCount, char orders[][50], int orderQuantities[], float
subtotals[], int *orderCount) {
    int choice, quantity;

    displayMenu(menu, itemCount);

    printf("Enter the item number to order (0 to finish): ");
    scanf("%d", &choice);
```

```
    while (choice != 0) {
        if (choice < 1 || choice > itemCount) {
            printf("Invalid choice. Please try again.\n");
        } else {
            printf("Enter the quantity: ");
            scanf("%d", &quantity);

            if (quantity > menu[choice - 1].quantity) {
                printf("Sorry, we only have %d of %s left in stock.\n", menu[choice - 1].quantity, menu[choice -
1].name);
            } else {
                strcpy(orders[*orderCount], menu[choice - 1].name);
                orderQuantities[*orderCount] = quantity;
                subtotals[*orderCount] = quantity * menu[choice - 1].price;

                menu[choice - 1].quantity -= quantity;

                (*orderCount)++;
            }
        }

        printf("Enter the item number to order (0 to finish): ");
        scanf("%d", &choice);
    }
}

void displayOrder(char orders[][50], int orderQuantities[], float subtotals[], int orderCount) {
    float total = 0;
    float tax;

    printf("\nOrder:\n");
    printf("%-20s|%-20s|%-20s|%-20s\n", "NAME", "PRICE FOR 1", "QUANTITY", "PRICE");
    printf("--------------------------------------------------------------------------------\n");
    int i;
            for ( i = 0; i < orderCount; i++) {
        printf("%-20s|%-20.2f|%-20d|%-20.2f\n", orders[i], subtotals[i] / orderQuantities[i],
orderQuantities[i], subtotals[i]);
        total += subtotals[i];
    }
    tax = total * 0.12;
    total += tax;

    printf("--------------------------------------------------------------------------------\n");
    printf("Tax: Rs%.2f\n", tax);
    printf("Total: Rs%.2f\n", total);
}
```

```
void deleteItem(struct MenuItem menu[], int *itemCount) {
    int choice, i;

    displayMenu(menu, *itemCount);

    printf("Enter the item number to delete: ");
    scanf("%d", &choice);

    if (choice < 1 || choice > *itemCount) {
        printf("Invalid choice. Please try again.\n");
    } else {
        for (i = choice - 1; i < *itemCount - 1; i++) {
            strcpy(menu[i].name, menu[i + 1].name);
            menu[i].price = menu[i + 1].price;
            menu[i].quantity = menu[i + 1].quantity;
        }
        (*itemCount)--;
    }
}
void giveFeedback() {
    struct Feedback feedback;
    getFeedback(&feedback);
    saveFeedbackToFile(&feedback);
    printf("Thank you for your feedback!\n");
}
int main() {
    struct MenuItem menu[100];
    int itemCount = 0;
    char orders[100][50];
    int orderQuantities[100];
    float subtotals[100];
    int orderCount = 0;

    loadMenuFromFile(menu, &itemCount);

    int choice;
    do {
        printf("\nCanteen Management System\n");
        printf("1. Display Menu\n");
        printf("2. Add Item\n");
        printf("3. Refill Item\n");
        printf("4. Search Item\n");
        printf("5. Place Order\n");
        printf("6. Display Order\n");
        printf("7. Delete Item\n");
        printf("8. feedback \n");
```

```
        printf("0. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                displayMenu(menu, itemCount);
                break;
            case 2:
                addItem(menu, &itemCount);
                break;
            case 3:
                refillItem(menu, itemCount);
                break;
            case 4:
                searchItem(menu, itemCount);
                break;
            case 5:
                placeOrder(menu, itemCount, orders, orderQuantities, subtotals, &orderCount);
                break;
            case 6:
                displayOrder(orders, orderQuantities, subtotals, orderCount);
                break;
            case 7:
                deleteItem(menu, &itemCount);
                break;
                case 8:
                        giveFeedback();
                break;

            case 0:
                printf("Exiting the program. Thank you!\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 0);

    saveMenuToFile(menu, itemCount);

    return 0;
}
```

5 Literature Review

- Laxmi and Soulakshme [1] had created a report on smart canteen system with wireless menu by using several Iot devices for capturing data.
- Dalal, Zaid, Rizvi, Ashfaq [2] had made a report on fast service of food using e-services with the help of mobile phone using android web application
- Tejaswini, Swati, Shubhi, Vishakha, Shipra [3] concluded on their report that crowded canteen issue is a problem during pandemic situations and using El-Gamal public key encryption is one of a solution.
- Akash, Kalpesh, Nithin and Smita [4] made a report on one of the common problems, huge crowd and delay in time in the canteen and proposed using ElGamal Public key encryption with a working solution.
- Lalitha, Magesh, Selvanarayanan and Keertheshwaran [5] made on report on E-canteen management system with reduced human errors, crowd in canteen mainly in educational sectors.
- Syawani and Ariffin [6] published a report on Android web applications by using agile-based methodology for buying food while facing crowd during covid pandemic.
- Rameshwari, Gaurav, Pranjal and Tejas [7] made a report on significance of E-Wallet in canteen management using web application providing a solution of crowds in canteens.
- Pranav, Sanchay and Vivek [8] made a report on the significance of feedback using micro controller-based feedback monitoring system.
- Jason, Joshua and Megan [9] made a report on poor diet management in canteen.
- Ibrahim [10] concluded in the report that to improve traditional or current methods used in canteen management we can develop an application promoting cashless payment and improving technology.

6 Results & Discussions

1. Menu Display and Management: - The item names, prices, and quantities are successfully shown on the menu by the Canteen Management System.

- The system precisely captures the name, price, and quantity of each item, and users are able to add new products to the menu.

2. File Handling: - During initialization, the system loads the menu by means of the "menu.txt" file.

- When a file is missing, the system notifies the user and generates a new, blank menu.

Placing and Handling Orders:

By choosing things from the menu and entering the desired quantity, users can place orders.

When a request for more than the amount of stock that is available, the system detects this and takes appropriate action.

Orders are processed precisely, with the purchased quantity taken out of the menu.

storing files:

The menu is successfully saved by the system to the "menu.txt" file or csv file, guaranteeing long-term preservation.

Names, prices, and quantities are formatted correctly in the stored file.

User Communication and Experience:

A favorable user experience is enhanced by the user interface, which makes it possible to interact with menu items in an intuitive manner.

Enhancements to user assistance and feedback systems may improve usability as a whole.

Data Dependability and Persistence:

Data persistence is ensured by the use of file handling, which enables the system to retain menu details between program sessions.

The method of making a new file when there isn't one already protects against possible data loss.

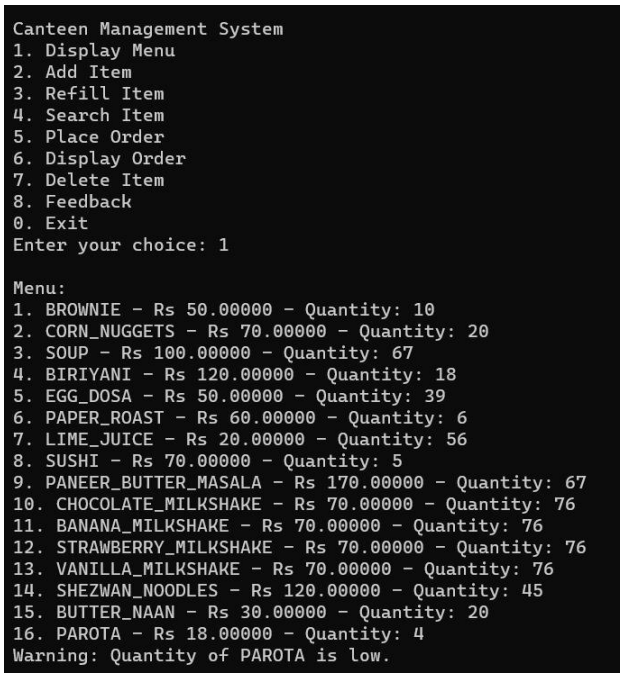
Error control and robustness of the system:

When an error occurs, such as a file not opening or an erroneous user input, the system robustly

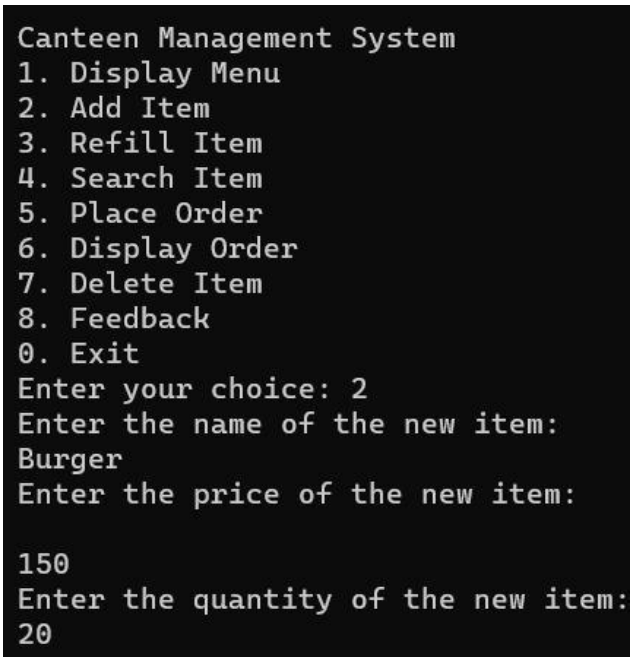
manages it and notifies users.

Output

1.Display menu
2.Add food item
3.Refill Item
4.Search item in menu
5.Place Order
6.Display Order
7.TXT file
8.CSV file



Display menu



Add item

```
Canteen Management System
1. Display Menu
2. Add Item
3. Refill Item
4. Search Item
5. Place Order
6. Display Order
7. Delete Item
8. Feedback
0. Exit
Enter your choice: 3

Menu:
1. BROWNIE - Rs 50.00000 - Quantity: 10
2. CORN_NUGGETS - Rs 70.00000 - Quantity: 20
3. SOUP - Rs 100.00000 - Quantity: 67
4. BIRIYANI - Rs 120.00000 - Quantity: 18
5. EGG_DOSA - Rs 50.00000 - Quantity: 39
6. PAPER_ROAST - Rs 60.00000 - Quantity: 6
7. LIME_JUICE - Rs 20.00000 - Quantity: 56
8. SUSHI - Rs 70.00000 - Quantity: 5
9. PANEER_BUTTER_MASALA - Rs 170.00000 - Quantity: 67
10. CHOCOLATE_MILKSHAKE - Rs 70.00000 - Quantity: 76
11. BANANA_MILKSHAKE - Rs 70.00000 - Quantity: 76
12. STRAWBERRY_MILKSHAKE - Rs 70.00000 - Quantity: 76
13. VANILLA_MILKSHAKE - Rs 70.00000 - Quantity: 76
14. SHEZWAN_NOODLES - Rs 120.00000 - Quantity: 45
15. BUTTER_NAAN - Rs 30.00000 - Quantity: 20
16. PAROTA - Rs 18.00000 - Quantity: 4
Warning: Quantity of PAROTA is low.
17. Burger - Rs 150.00000 - Quantity: 20
Enter the item number to refill: 16
Enter the quantity to add: 25
```

Refill item

```
Canteen Management System
1. Display Menu
2. Add Item
3. Refill Item
4. Search Item
5. Place Order
6. Display Order
7. Delete Item
8. Feedback
0. Exit
Enter your choice: 4
Enter the name of the item to search: SOUP
SOUP is in the menu with a price of Rs 100.00000
```

Search food item

```
Canteen Management System
1. Display Menu
2. Add Item
3. Refill Item
4. Search Item
5. Place Order
6. Display Order
7. Delete Item
8. Feedback
0. Exit
Enter your choice: 5

Menu:
1. BROWNIE - Rs 50.00000 - Quantity: 10
2. CORN_NUGGETS - Rs 70.00000 - Quantity: 20
3. SOUP - Rs 100.00000 - Quantity: 67
4. BIRIYANI - Rs 120.00000 - Quantity: 18
5. EGG_DOSA - Rs 50.00000 - Quantity: 39
6. PAPER_ROAST - Rs 60.00000 - Quantity: 6
7. LIME_JUICE - Rs 20.00000 - Quantity: 56
8. SUSHI - Rs 70.00000 - Quantity: 5
9. PANEER_BUTTER_MASALA - Rs 170.00000 - Quantity: 67
10. CHOCOLATE_MILKSHAKE - Rs 70.00000 - Quantity: 76
11. BANANA_MILKSHAKE - Rs 70.00000 - Quantity: 76
12. STRAWBERRY_MILKSHAKE - Rs 70.00000 - Quantity: 76
13. VANILLA_MILKSHAKE - Rs 70.00000 - Quantity: 76
14. SHEZWAN_NOODLES - Rs 120.00000 - Quantity: 45
15. BUTTER_NAAN - Rs 30.00000 - Quantity: 20
16. PAROTA - Rs 18.00000 - Quantity: 29
17. Burger - Rs 150.00000 - Quantity: 20
Enter the item number to order (0 to finish): 4
Enter the quantity: 4
Enter the item number to order (0 to finish): 5
Enter the quantity: 2
Enter the item number to order (0 to finish): 0
```

Place order

```
Canteen Management System
1. Display Menu
2. Add Item
3. Refill Item
4. Search Item
5. Place Order
6. Display Order
7. Delete Item
8. Feedback
0. Exit
Enter your choice: 6

Order:
NAME                |PRICE FOR 1          |QUANTITY             |PRICE
-------------------------------------------------------------------------
BIRIYANI            |120.00               |4                    |480.00
EGG_DOSA            |50.00                |2                    |100.00
-------------------------------------------------------------------------
Tax: Rs69.60
Total: Rs649.60
```

Display order

```
Canteen Management System
1. Display Menu
2. Add Item
3. Refill Item
4. Search Item
5. Place Order
6. Display Order
7. Delete Item
8. Feedback
0. Exit
Enter your choice: 7

Menu:
1. BROWNIE - Rs 50.00000 - Quantity: 10
2. CORN_NUGGETS - Rs 70.00000 - Quantity: 20
3. SOUP - Rs 100.00000 - Quantity: 67
4. BIRIYANI - Rs 120.00000 - Quantity: 14
5. EGG_DOSA - Rs 50.00000 - Quantity: 37
6. PAPER_ROAST - Rs 60.00000 - Quantity: 6
7. LIME_JUICE - Rs 20.00000 - Quantity: 56
8. SUSHI - Rs 70.00000 - Quantity: 5
9. PANEER_BUTTER_MASALA - Rs 170.00000 - Quantity: 67
10. CHOCOLATE_MILKSHAKE - Rs 70.00000 - Quantity: 76
11. BANANA_MILKSHAKE - Rs 70.00000 - Quantity: 76
12. STRAWBERRY_MILKSHAKE - Rs 70.00000 - Quantity: 76
13. VANILLA_MILKSHAKE - Rs 70.00000 - Quantity: 76
14. SHEZWAN_NOODLES - Rs 120.00000 - Quantity: 45
15. BUTTER_NAAN - Rs 30.00000 - Quantity: 20
16. PAROTA - Rs 18.00000 - Quantity: 29
17. Burger - Rs 150.00000 - Quantity: 20
Enter the item number to delete: 5
```

```
Menu:
1. BROWNIE - Rs 50.00000 - Quantity: 10
2. CORN_NUGGETS - Rs 70.00000 - Quantity: 20
3. SOUP - Rs 100.00000 - Quantity: 67
4. BIRIYANI - Rs 120.00000 - Quantity: 14
5. PAPER_ROAST - Rs 60.00000 - Quantity: 6
6. LIME_JUICE - Rs 20.00000 - Quantity: 56
7. SUSHI - Rs 70.00000 - Quantity: 5
8. PANEER_BUTTER_MASALA - Rs 170.00000 - Quantity: 67
9. CHOCOLATE_MILKSHAKE - Rs 70.00000 - Quantity: 76
10. BANANA_MILKSHAKE - Rs 70.00000 - Quantity: 76
11. STRAWBERRY_MILKSHAKE - Rs 70.00000 - Quantity: 76
12. VANILLA_MILKSHAKE - Rs 70.00000 - Quantity: 76
13. SHEZWAN_NOODLES - Rs 120.00000 - Quantity: 45
14. BUTTER_NAAN - Rs 30.00000 - Quantity: 20
15. PAROTA - Rs 18.00000 - Quantity: 29
16. Burger - Rs 150.00000 - Quantity: 20
```

Delete order

	A	B	C
1	BROWNIE	50	10
2	CORN_NUGGETS	30	20
3	SOUP	110	67
4	BIRIYANI	120	18
5	EGG_DOSA	50	39
6	PAPER_ROAST	60	6
7	LIME_JUICE	70	56
8	SUSHI	70	5
9	PANEER_BUTTER_MASALA	180	67
10	CHOCOLATE_MILKSHAKE	80	76
11	BANANA_MILKSHAKE	80	76
12	STRAWBERRY_MILKSHAKE	80	76
13	VANILLA_MILKSHAKE	80	45
14	SHEZWAN_NOODLES	120	30
15	BUTTER_NAAN	30	23
16	PAROTA	45	4

Menu written in csv file

File Edit View

```
BROWNIE 12.00 10
CORN_NUGGETS 20.00 20
SOUP 110.00 67
BIRIYANI 120.00 18
EGG_DOSA 50.00 39
PAPER_ROAST 60.00 6
LIME_JUICE 70.00 56
SUSHI 70.00 5
PANEER_BUTTER_MASALA 180.00 67
CHOCOLATE_MILKSHAKE 80.00 76
BANANA_MILKSHAKE 80.00 76
STRAWBERRY_MILKSHAKE 80.00 76
VANILLA_MILKSHAKE 80.00 76
SHEZWAN_NOODLES 120.00 45
BUTTER_NAAN 30.00 20
PAROTA 45.00 4
```

Menu written in txt file

```
Canteen Management System
1. Display Menu
2. Add Item
3. Refill Item
4. Search Item
5. Place Order
6. Display Order
7. Delete Item
8. Feedback
0. Exit
Enter your choice: 8
Enter your name: Sanjana
Rate the quantity (1-5): 4
Rate the quality (1-5): 5
Rate the service (1-5): 4
Any additional comments: Food could have been served hot
Thank you for your feedback!
```

```
Canteen Management System
1. Display Menu
2. Add Item
3. Refill Item
4. Search Item
5. Place Order
6. Display Order
7. Delete Item
8. feedback
0. Exit
Enter your choice: 8
Enter your name: diyaa
Rate the quantity (1-5): 4
Rate the quality (1-5): 5
Rate the service (1-5): 4
Any additional comments: service is slow.
Thank you for your feedback!
```

Feedback

name	quantity(1-5)	quality(1-5)	service(1-5)	comments
diyaa	4	5	4	service is slow.

Csv feedback

```
      name  quantity(1-5)   quality(1-5)   service(1-5)     comments
  Sanjana         4               5              4       Food could have been served hot
```

Txt feedback

7. Conclusion & Future Work

The Canteen Management System, implemented in the C programming language, has successfully delivered a robust and user-centric solution for the efficient management of canteen operations. This comprehensive system incorporates various features, each contributing to an enhanced user experience, streamlined menu management, and seamless order processing.

Operational Efficiency and Transparency:

The system's ability to display the menu, presenting users with detailed information about each food item, including prices and available quantities, significantly contributes to operational transparency. Users, both administrators and customers, can easily navigate through the menu, fostering an environment where choices can be made with clarity and confidence.

The "Add Item" functionality further bolsters the system's operational adaptability. The simplicity with which new food items can be added ensures that the canteen's offerings remain dynamic, accommodating changes in the menu seamlessly. This adaptability is crucial in catering to the ever-evolving preferences and demands of canteen patrons.

The inclusion of a robust "Search Food Item" feature adds a layer of convenience for users. This functionality expedites the process of locating specific items on the menu, saving time and effort. In a bustling canteen environment, efficiency in accessing information is paramount, and the system delivers on this front.

Efficient Order Processing and Flexibility:

Order placement is a critical aspect of any canteen management system, and the implemented solution excels in this regard. The "Place Order" function streamlines the process, ensuring that users can select items from the menu, specify quantities, and have their orders accurately processed. The system's validation of stock availability adds a layer of reliability, preventing potential issues related to inventory discrepancies. The "Display Order" function adds a level of transparency to the order processing stage. Users can review detailed summaries of their current orders, providing them with an opportunity to verify their selections before finalizing the order. This feature not only enhances user confidence but also reduces the likelihood of order errors.

Flexibility is a key theme throughout the system, and the "Delete Order" functionality exemplifies this attribute. Allowing users to modify their orders by removing selected items caters to the dynamic nature of customer preferences. This adaptability ensures a positive and user-friendly experience, where changes in orders can be accommodated effortlessly.

System Integrity and User Guidance:

The exit function of the system ensures a graceful termination, prioritizing data integrity and providing users with a seamless transition between program sessions. This attention to system integrity is crucial for maintaining the consistency of data across multiple interactions. The system's ability to handle errors effectively and offer clear user guidance in case of invalid inputs or potential issues contributes significantly to its reliability.

In conclusion, the Canteen Management System stands as a successful implementation that not only simplifies but also optimizes day-to-day canteen operations. The user-centric design, coupled with operational efficiency, results in a positive and seamless experience for both administrators and customers.Looking ahead, the system demonstrates potential avenues for enhancement. The exploration of scalability features to accommodate larger datasets could be a valuable addition, especially in canteens with extensive menu offerings. Additionally, the incorporation of advanced security measures and integration with external systems for financial transactions and inventory management could further elevate the system's capabilities.

The success of the Canteen Management System goes beyond its current functionality; it serves as a foundation for future improvements and innovations. As technology continues to evolve, this implementation positions itself as a reliable and adaptable solution for the ever-changing

landscape of food service management. The journey of the Canteen Management System exemplifies the impact of well-designed and implemented software solutions in optimizing operational efficiency and enhancing user experiences.

8.References

[1] Jaya Laxmi R,Soulakshmee Devi.N,2021.Smart Canteen System for a University. In 2021 International Conference on Communication, Computing and Internet of Things IEEE.

[2] Dalal, M., Barmare, Z., Rizvi, Z. and Shaikh, A., 2015. Android Based Canteen Management System. Department of Information Technology, MH SabooSiddik College of Engineering, Maharashtra, India.

[3] Sharma, T., Jha, S., Gupta, S., Singh, V. and Gautam, S., CASHLESS & ONLINE QR-CODE BASED CANTEEN MANAGEMENT SYSTEM.

[4] Katkar, A., Juvekar, K., Rohira, N. and Jangale, S., 2018. Canteen management system using E-wallet. International journal of advance research, idea and innovation.

[5] Lalitha, V., Magesh, K., Selvanarayanan, A. and Keertheshwaran, G., 2022, March. E-Canteen Management System based on Web Application. In 2022 International Conference on Communication, Computing and Internet of Things (IC3IoT) (pp. 1-4). IEEE.

[6] Zul, N.S. and Zin, N.A.M., 2022. Canteen Food Ordering and Management Application. Applied Information Technology and Computer Science, 3(2), pp.1094-1112.

[7] Fegade, R., Nandge, G., Patil, P., Gaikwad, T. and Bastawade, P.P., 2019. Canteen Management Android Application Using E-Wallet. International Research Journal of Engineering and Technology (IRJET), 6(3), pp.6624-6628.

[8] Madupu, P.K., Sharma, S. and Patel, V., 2018, April. Implementation of Feedback Monitoring System for Canteen Using IoT. In 2018 3rd International Conference for Convergence in Technology (I2CT) (pp. 1-5). IEEE.

[9] Wu, J.H., Berg, J. and Neeson, M., 2016. Overview of development and implementation of school canteen nutrition guidelines in Australia. Journal of the Home Economics Institute of Australia, 23(1), pp.2-10.

[10] Ibrahim, I.B., 2017. Cashless Meal Application for School Canteen: Meal-Go Application. CARNIVAL ON e-LEARNING (IUCEL) 2017, p.233.