DLITHE PROJECT REPORT

PROJECT ID : CP033

PROJECT TITLE : Real-estate Advisor

TEAM MEMBERS : Adhya K S (4MT21CS007)

Advith Pai B (4MT21CS012)

Amrutha G Bangera (4MTCS21022)

Asha U Nayak (4MT19CS029)

Jeevan Naik (4MT21CS059)

REPORT

Abstract :

The Medical Store Management System is a software application designed to manage the inventory of medicines, suppliers, and customers in a medical store. The system allows users to add, display, modify, and delete medicines, suppliers, and customers' records. It also provides features such as adding and managing suppliers and customers. This system helps streamline the operations of a medical store, making it more efficient and organized.

Introduction :

Background :

Medical stores play a crucial role in the healthcare industry by providing essential medicines to patients. Managing the inventory of medicines and keeping track of suppliers and customers' information can be a challenging task. The Medical Store Management System aims to simplify and automate these processes, ensuring smooth operations and customer satisfaction.

Objectives :

- The main objectives of the Medical Store Management System are as follows:
- To maintain an organized inventory of medicines.
- To manage supplier information and contact details.
- To manage customer information and contact details.
- To provide the ability to add, display, modify, and delete records.
- To streamline the operations of a medical store.

Technologies Used :

The Medical Store Management System is implemented using the following technologies:

- C programming language
- File handling for data storage

System Architecture :

Front-End :

The front-end of the system is implemented using the console or command-line interface, where users interact with the system by entering commands and providing input.

Back-End :

The back-end of the system consists of C programming logic that performs various operations such as adding, displaying, modifying, and deleting records. It also handles file handling to store and retrieve data.

Database :

The system uses files to act as databases for storing information about medicines, suppliers, and customers. Each entity has a separate file for data storage.

File Handling in this Program:

- The program uses file handling to store and retrieve data for medicines, suppliers, and customers.
- Each entity (medicines, suppliers, customers) has its own separate file for data storage.
- File operations such as opening, reading, writing, modifying, and deleting records are performed using file pointers and functions.

Project Modules :

The project consists of the following modules:

- Module 1: Medicine Management
  - Add Medicine
  - Display Medicines
  - Modify Medicine
  - Delete Medicine
- Module 2: Supplier Management
  - Add Supplier
  - Display Suppliers
  - Modify Supplier
  - Delete Supplier
- Module 3: Customer Management
  - Add Customer
  - Display Customers
  - Modify Customer
  - Delete Customer

Design and Implementation :

Front-End Design :

- The front-end design is based on a command-line interface, where users can select options by entering numbers corresponding to their desired actions.

Back-End Design :

- The back-end logic is implemented in the C programming language.
- Each module has its functions for adding, displaying, modifying, and deleting records.
- File handling functions are used to perform operations on data files.

Database Design :

- Data for medicines, suppliers, and customers are stored in separate files.
- Each file contains records in a structured format.

Features and Functionality :

- Feature 1: Medicine Management

  - Add new medicines to the inventory.

  -Display a list of all medicines with details.

  -Modify existing medicine records.

  -Delete medicine records.

- Feature 2: Supplier Management

  -Add new suppliers to the database.

  -Display a list of all suppliers with contact details.

  -Modify supplier information.

  - Delete supplier records.

- Feature 3: Customer Management

  -Add new customers to the database.

  -Display a list of all customers with address details.

  -Modify customer information.

  -Delete customer records.

Testing :

Unit Testing :

- Each module is tested individually to ensure that it performs its functions correctly.
- Test cases are created to cover various scenarios.

Integration Testing :

- Modules are integrated to test the overall functionality of the system.
- Data flow between modules is tested.

User Acceptance Testing :

- The system is tested by end-users to ensure it meets their requirements and expectations.
- Any feedback or issues raised by users are addressed and resolved.

Challenges Faced :

- Implementing file handling for data storage and retrieval.
- Ensuring data consistency and accuracy.
- Handling errors and exceptions gracefully.
- Future Enhancements
- Implementing a graphical user interface (GUI) for a more user-friendly experience.
- Adding features for generating reports and statistics.
- Implementing security measures to protect data.

Conclusion :

The Medical Store Management System is a valuable tool for medical stores to efficiently manage their inventory, suppliers, and customers' information. It simplifies record keeping and ensures smooth operations. With future enhancements, it can further improve its functionality and usability.

References :

https://www.studytonight.com/c-projects/medical-store-management-using-c-language

https://www.codewithc.com/mini-project-in-c-medical-store-management-system/

Appendices:

Appendix A : Sample Data Files

Sample data files for medicines, suppliers, and customers used for testing the program.

medicines.dat: Contains sample medicine records.

suppliers.dat: Contains sample supplier records.

customers.dat: Contains sample customer records.

Appendix B : User Manual

A user manual providing instructions on how to use the Medical Store Management System, including detailed explanations of each menu option and functionality.

Appendix C : Test Cases

A list of test cases and their expected results used for unit testing and integration testing of the program.

Appendix D : Flowcharts

Flowcharts illustrating the flow of control and data in the program for each module (Medicine Management, Supplier Management, Customer Management).

Appendix E : Error Handling

A document detailing the error handling mechanisms implemented in the program, including how errors are detected and how they are handled.

Appendix F : Future Enhancements

A document outlining potential future enhancements and features that can be added to the program to improve its functionality and usability.

Appendix G : Sample Reports

Sample reports generated by the program, showcasing the potential reporting capabilities of the system.

Appendix H : Data Dictionary

A data dictionary providing descriptions of the data structures used in the program, including the fields and their data types.

Appendix I : Glossary

A glossary of terms and abbreviations used in the program, helping users understand the terminology used in the system.

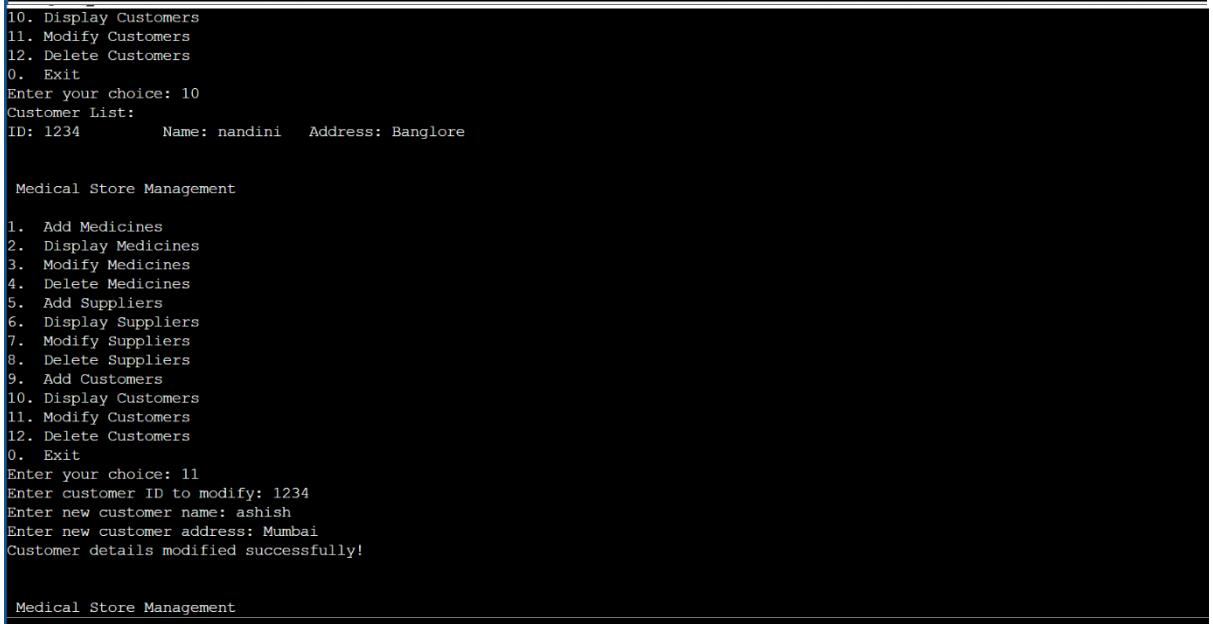
Appendix J : References

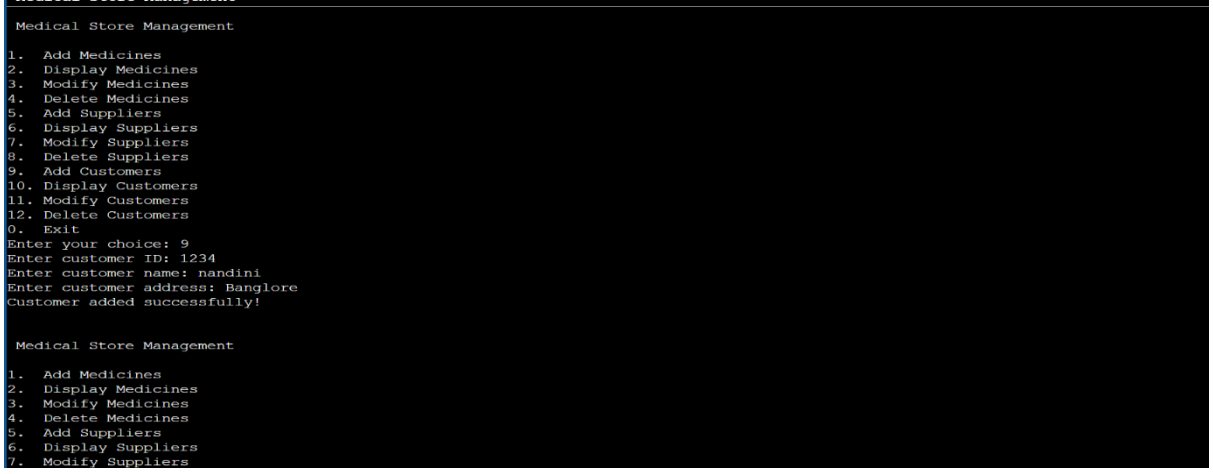
A list of references and sources of information used in the development of the program.

Note :

Appendices A and B provide placeholders and you should include the actual content of the respective sections in your documentation. Additionally, consider adding any other relevant appendices or documentation as needed for your specific project.

Screenshots :





```
10. Display Customers
11. Modify Customers
12. Delete Customers
0.  Exit
Enter your choice: 7
Enter supplier ID to modify: 123
Enter new supplier name: karan
Enter new supplier contact: 1234567899
Supplier details modified successfully!


 Medical Store Management

1.  Add Medicines
2.  Display Medicines
3.  Modify Medicines
4.  Delete Medicines
5.  Add Suppliers
6.  Display Suppliers
7.  Modify Suppliers
8.  Delete Suppliers
9.  Add Customers
10. Display Customers
11. Modify Customers
12. Delete Customers
0.  Exit
Enter your choice: 8
Enter supplier ID to delete: 123
Supplier found and deleted.


 Medical Store Management
```

```
1.  Add Medicines
2.  Display Medicines
3.  Modify Medicines
4.  Delete Medicines
5.  Add Suppliers
6.  Display Suppliers
7.  Modify Suppliers
8.  Delete Suppliers
9.  Add Customers
10. Display Customers
11. Modify Customers
12. Delete Customers
0.  Exit
Enter your choice: 6
Supplier List:
ID: 123  Name: kiran     Contact: 9876543211


 Medical Store Management

1.  Add Medicines
2.  Display Medicines
3.  Modify Medicines
4.  Delete Medicines
5.  Add Suppliers
6.  Display Suppliers
7.  Modify Suppliers
8.  Delete Suppliers
9.  Add Customers
10. Display Customers
11. Modify Customers
```

```
10. Display Customers
11. Modify Customers
12. Delete Customers
0.  Exit
Enter your choice: 4
Enter medicine ID to delete: 12
Medicine found and deleted.


 Medical Store Management

1.  Add Medicines
2.  Display Medicines
3.  Modify Medicines
4.  Delete Medicines
5.  Add Suppliers
6.  Display Suppliers
7.  Modify Suppliers
8.  Delete Suppliers
9.  Add Customers
10. Display Customers
11. Modify Customers
12. Delete Customers
0.  Exit
Enter your choice: 5
Enter supplier ID: 123
Enter supplier name: kiran
Enter supplier contact: 9876543211
Supplier added successfully!


 Medical Store Management
```

```
2.  Display Medicines
3.  Modify Medicines
4.  Delete Medicines
5.  Add Suppliers
6.  Display Suppliers
7.  Modify Suppliers
8.  Delete Suppliers
9.  Add Customers
10. Display Customers
11. Modify Customers
12. Delete Customers
0.  Exit
Enter your choice: 3
Enter medicine id: 12
Enter new medicine name: paara
Enter new medicine price: 150
Enter new quantity: 150
Medicine modified successfully!


 Medical Store Management

1.  Add Medicines
2.  Display Medicines
3.  Modify Medicines
4.  Delete Medicines
5.  Add Suppliers
6.  Display Suppliers
7.  Modify Suppliers
8.  Delete Suppliers
9.  Add Customers
10. Display Customers
```

```
1.  Add Medicines
2.  Display Medicines
3.  Modify Medicines
4.  Delete Medicines
5.  Add Suppliers
6.  Display Suppliers
7.  Modify Suppliers
8.  Delete Suppliers
9.  Add Customers
10. Display Customers
11. Modify Customers
12. Delete Customers
0.  Exit
Enter your choice: 2
Medicine List:
ID: 12   Name: paracetoomol      Price: 100.00   Quantity: 100


 Medical Store Management

1.  Add Medicines
2.  Display Medicines
3.  Modify Medicines
4.  Delete Medicines
5.  Add Suppliers
6.  Display Suppliers
7.  Modify Suppliers
8.  Delete Suppliers
9.  Add Customers
10. Display Customers
11. Modify Customers

 Medical Store Management

1.  Add Medicines
2.  Display Medicines
3.  Modify Medicines
4.  Delete Medicines
5.  Add Suppliers
6.  Display Suppliers
7.  Modify Suppliers
8.  Delete Suppliers
9.  Add Customers
10. Display Customers
11. Modify Customers
12. Delete Customers
0.  Exit
Enter your choice: 1
Enter medicine ID: 12
Enter medicine name: paracetoomol
Enter medicine price: 100
Enter quantity: 100
Medicine added successfully!


 Medical Store Management

1.  Add Medicines
2.  Display Medicines
3.  Modify Medicines
4.  Delete Medicines
5.  Add Suppliers
```

Code Snippets :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Medicine
{
    int id;
    char name[100];
    float price;
    int quantity;
};
struct Customer
{
    int id;
    char name[100];
    char address[200];
};
struct Supplier
{
    int id;
    char name[100];
    char contact[20];
};
//Add Medicine
void addMedicine()
{
    FILE *file = fopen("medicines.dat", "ab+");
    struct Medicine medicine;
    printf("Enter medicine ID: ");
```

```
    scanf("%d", &medicine.id);
    printf("Enter medicine name: ");
    scanf("%s", medicine.name);
    printf("Enter medicine price: ");
    scanf("%f", &medicine.price);
    printf("Enter quantity: ");
    scanf("%d", &medicine.quantity);
    fwrite(&medicine, sizeof(struct Medicine), 1, file);
    printf("Medicine added successfully!\n");
    fclose(file);
}
//Display Medicine
void displayMedicines()
{
    struct Medicine medicine;
    FILE *file = fopen("medicines.dat", "ab+");
    printf("Medicine List:\n");

    while (fread(&medicine, sizeof(struct Medicine), 1, file) == 1)
    {
        printf("ID: %d\t Name: %s\t Price: %.2f\t Quantity: %d\n",
               medicine.id, medicine.name, medicine.price, medicine.quantity);
    }
    fclose(file);
}
//Modify Medicines
void modifyMedicine()
{
    int id;
```

```
    printf("Enter medicine id: ");
    scanf("%d", &id);
 struct Medicine medicine;
    int found = 0;
    FILE *file = fopen("medicines.dat", "rb+"); // Open the file in read/write mode

    while (fread(&medicine, sizeof(struct Medicine), 1, file) == 1)
    {
        if (medicine.id == id)
        {
            found = 1;
             printf("Enter new medicine name: ");
            scanf("%s", medicine.name);
            printf("Enter new medicine price: ");
            scanf("%f", &medicine.price);
            printf("Enter new quantity: ");
            scanf("%d", &medicine.quantity);
            fseek(file, -sizeof(struct Medicine), SEEK_CUR);
            fwrite(&medicine, sizeof(struct Medicine), 1, file);
            printf("Medicine modified successfully!\n");
            break;
        }
    }
if (!found)
    {
        printf("Medicine with ID %d not found.\n", id);
    }
 fclose(file);
}
```

```
//Delete Medicines
void deleteMedicine()
{
    int id;
    printf("Enter medicine ID to delete: ");
    scanf("%d", &id);
struct Medicine medicine;
    int found = 0;
    FILE *file = fopen("medicines.dat", "rb");
    FILE *tempFile = fopen("temp.dat", "wb"); // Temporary file to hold non-deleted records

    while (fread(&medicine, sizeof(struct Medicine), 1, file) == 1)
    {
        if (medicine.id == id)
        {
            found = 1;
            printf("Medicine found and deleted.\n");
        }
        else
        {
            fwrite(&medicine, sizeof(struct Medicine), 1, tempFile);
        }
    }

    fclose(file);
    fclose(tempFile);

    if (!found)
    {
```

```
        printf("Medicine with ID %d not found.\n", id);
    }
    else
    {
        remove("medicines.dat");            // Delete the old medicines.dat file
        rename("temp.dat", "medicines.dat"); // Rename temp.dat to medicines.dat
    }
}
//Add Supplier
void addSupplier()
{
    FILE *file = fopen("suppliers.dat", "ab+");
    struct Supplier supplier;
    printf("Enter supplier ID: ");
    scanf("%d", &supplier.id);
    printf("Enter supplier name: ");
    scanf("%s", supplier.name);
    printf("Enter supplier contact: ");
    scanf("%s", supplier.contact);
    fwrite(&supplier, sizeof(struct Supplier), 1, file);
    printf("Supplier added successfully!\n");
    fclose(file);
}
// Function to display suppliers
void displaySupplier()
{
    struct Supplier supplier;
    FILE *file = fopen("suppliers.dat", "ab+");
    printf("Supplier List:\n");
```

```
    while (fread(&supplier, sizeof(struct Supplier), 1, file) == 1)
    {
        printf("ID: %d\t Name: %s\t Contact: %s\n",
               supplier.id, supplier.name, supplier.contact);
    }
    fclose(file);
}
// Function to delete a supplier by ID
void deleteSupplier()
{
    int id;
    printf("Enter supplier ID to delete: ");
    scanf("%d", &id);
struct Supplier supplier;
    int found = 0;
    FILE *file = fopen("suppliers.dat", "rb");
    FILE *tempFile = fopen("temp_suppliers.dat", "wb"); // Temporary file to hold non-deleted records
while (fread(&supplier, sizeof(struct Supplier), 1, file) == 1)
    {
        if (supplier.id == id)
        {
            found = 1;
            printf("Supplier found and deleted.\n");
        }
        else
        {
            fwrite(&supplier, sizeof(struct Supplier), 1, tempFile);
```

```
        }
    }
fclose(file);
    fclose(tempFile);
 if (!found)
    {
        printf("Supplier with ID %d not found.\n", id);
    }
    else
    {
        remove("suppliers.dat");                    // Delete the old suppliers.dat file
rename("temp_suppliers.dat", "suppliers.dat"); // Rename temp_suppliers.dat to suppliers.dat
    }
}
// Function to modify supplier details
void modifySupplier()
{
    int id;
    printf("Enter supplier ID to modify: ");
    scanf("%d", &id);

    struct Supplier supplier;
    int found = 0;
    FILE *file = fopen("suppliers.dat", "rb+"); // Open the file in read/write mode

    while (fread(&supplier, sizeof(struct Supplier), 1, file) == 1)
    {
        if (supplier.id == id)
        {
```

```
            found = 1;

            printf("Enter new supplier name: ");
            scanf("%s", supplier.name);
            printf("Enter new supplier contact: ");
            scanf("%s", supplier.contact);

            fseek(file, -sizeof(struct Supplier), SEEK_CUR);
            fwrite(&supplier, sizeof(struct Supplier), 1, file);

            printf("Supplier details modified successfully!\n");
            break;
        }
    }
if (!found)
    {
        printf("Supplier with ID %d not found.\n", id);
    }
 fclose(file);
}
//Add Customer
void addCustomer()
{
    FILE *file = fopen("customers.dat", "ab+");
    struct Customer customer;
    printf("Enter customer ID: ");
    scanf("%d", &customer.id);
    printf("Enter customer name: ");
    scanf("%s", customer.name);
```

```
    printf("Enter customer address: ");
    scanf("%s", customer.address);
    fwrite(&customer, sizeof(struct Customer), 1, file);
    printf("Customer added successfully!\n");
    fclose(file);
}
//Display customer
void displayCustomer()
{
    struct Customer customer;
    FILE *file = fopen("customers.dat", "ab+");
    printf("Customer List:\n");
while (fread(&customer, sizeof(struct Customer), 1, file) == 1)
    {
        printf("ID: %d\t Name: %s\t Address: %s\n",
              customer.id, customer.name, customer.address);
}
    fclose(file);
}
//Modify Customer
void modifyCustomer()
{
    int id;
    printf("Enter customer ID to modify: ");
    scanf("%d", &id);
struct Customer customer;
    int found = 0;
    FILE *file = fopen("customers.dat", "rb+"); // Open the file in read/write mode
while (fread(&customer, sizeof(struct Customer), 1, file) == 1)
```

```
    {
        if (customer.id == id)
        {
            found = 1;
            printf("Enter new customer name: ");
            scanf("%s", customer.name);
            printf("Enter new customer address: ");
            scanf("%s", customer.address);
            fseek(file, -sizeof(struct Customer), SEEK_CUR);
            fwrite(&customer, sizeof(struct Customer), 1, file);
printf("Customer details modified successfully!\n");
            break;
        }
    }
if (!found)
    {
        printf("Customer with ID %d not found.\n", id);
    }
    fclose(file);
}
//Delete Customer
void deleteCustomer()
{
    int id;
    printf("Enter customer ID to delete: ");
    scanf("%d", &id);
    struct Customer customer;
    int found = 0;
    FILE *file = fopen("customers.dat", "rb");
```

```
    FILE *tempFile = fopen("temp_customers.dat", "wb"); // Temporary file to hold non-deleted records
    while (fread(&customer, sizeof(struct Customer), 1, file) == 1)
    {
        if (customer.id == id)
        {
            found = 1;
            printf("Customer found and deleted.\n");
        }
        else
        {
            fwrite(&customer, sizeof(struct Customer), 1, tempFile);
        }
    }
    fclose(file);
    fclose(tempFile);

    if (!found)
    {
        printf("Customer with ID %d not found.\n", id);
    }
    else
    {
        remove("customers.dat");                    // Delete the old customers.dat file
        rename("temp_customers.dat", "customers.dat"); // Rename temp_customers.dat to customers.dat
    }
}

int main()
```

```
{
    int choice;
    do
    {
        printf("\n\n Medical Store Management \n\n");
        printf("1.  Add Medicines\n");
        printf("2.  Display Medicines\n");
        printf("3.  Modify Medicines\n");
        printf("4.  Delete Medicines\n");
        printf("5.  Add Suppliers\n");
        printf("6.  Display Suppliers\n");
        printf("7.  Modify Suppliers\n");
        printf("8.  Delete Suppliers\n");
        printf("9.  Add Customers\n");
        printf("10. Display Customers\n");
        printf("11. Modify Customers\n");
        printf("12. Delete Customers\n");
        printf("0.  Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            addMedicine();
            break;
        case 2:
            displayMedicines();
            break;
        case 3:
```

```
        modifyMedicine();
        break;
    case 4:
        deleteMedicine();
        break;
    case 5:
        addSupplier();
        break;
    case 6:
        displaySupplier();
        break;
    case 7:
        modifySupplier();
        break;
    case 8:
        deleteSupplier();
        break;
    case 9:
        addCustomer();
        break;
    case 10:
        displayCustomer();
        break;
    case 11:
        modifyCustomer();
        break;
    case 12:
        deleteCustomer();
        break;
```

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