PROJECT REPORT

Medical Store Management System

Course: Programming in C (B.Tech 1st Sem)

Submitted To: Mohsin F. Dar

Submitted By: Raghav Bhardwaj

SAP ID: 590025137

Date: December 1, 2025

TABLE OF CONTENTS

---

1. ABSTRACT

The **Medical Store Management System** is a functional C program designed to digitize the inventory processes of a pharmacy. Traditional manual record-keeping is error-prone and time-consuming. This project automates the tracking of medicine stock, prices, and sales data.

The system allows users to **Add, View, Search, Update, and Delete** medicine records. A key feature is **File Persistence**, ensuring that data is saved to inventory.txt automatically upon exit and reloaded when the program starts. This project demonstrates core C concepts including Structures (struct), file handling (fscanf, fprintf), and pointers.

Keywords: Inventory Management, File Handling, CRUD Operations, C Programming, Structures.

---

2. PROBLEM DEFINITION

Background

Managing a medical store involves tracking hundreds of different medicines, their quantities, and prices. Doing this manually in paper ledgers often leads to calculation errors, stock mismatches, and difficulty in locating specific items quickly.

Problem Statement

Pharmacists need a simple, offline digital tool to:

- Maintain a database of available medicines.
- Quickly search for a medicine by its ID.
- Update stock levels (add stock or reduce after sales).
- Ensure data is not lost when the computer is turned off.

Objectives

- Create a menu-driven **Inventory System** in C.
- Implement **CRUD** (Create, Read, Update, Delete) functionalities.
- Use **File I/O** to permanently store medicine records.
- Utilize **Structures** to group related data (ID, Name, Quantity, Price).

---

3. SYSTEM DESIGN

How the Program Works

The program operates via a while(1) loop containing a switch-case menu with 6 main options:

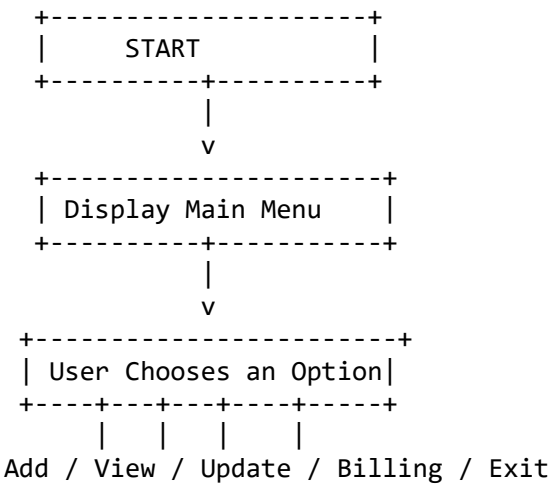
1. **Add:** Input new medicine details (ID, Name, Qty, Price).
2. **View:** Display the full list of medicines currently in stock.
3. **Search:** Find a specific medicine using its unique ID.
4. **Update:** Modify the quantity of an existing medicine (useful for restocking).
5. **Delete:** Remove a medicine from the database permanently.
6. **Exit:** Saves all data to inventory.txt and terminates the program.

Data Storage & Handling

- **Memory:** The program uses an array of structures Medicine m[MAX] to hold up to 100 records in RAM for fast processing.
- **Storage:** Data is written to inventory.txt in a formatted text structure (%d %s %d %.2f) to ensure readability and easy retrieval.

---

4. FLOWCHART



System Flow:

1. **Start** -> **Load Data** from inventory.txt.
2. **Display Menu** (1. Add, 2. View, 3. Search, 4. Update, 5. Delete, 6. Exit).
3. **User Input** -> Read Choice.
4. **Decision Diamond:**
    - If 1: Call add() -> Input Details -> Return to Menu.
    - If 2: Call view() -> Loop through Array -> Print -> Return to Menu.
    - If 3: Call search() -> Compare IDs -> Print Result -> Return to Menu.
    - If 4: Call update() -> Find ID -> Math Operation on Qty -> Return to Menu.
    - If 5: Call del() -> Shift Array Elements -> Decrement Count -> Return to Menu.
    - If 6: Call SaveToFile() -> Write Array to Disk -> **Stop**.

---

5. ALGORITHM

Main Control Loop

1. **Initialize** array Medicine m[MAX] and counter n = 0.

2. **Call** load(m, &n) to populate array from file.
3. **Loop** (Infinite):
    - Print Menu Options.
    - Read User Choice ch.
    - **Switch(ch)**:
        - Case 1: add()
        - Case 2: view()
        - Case 3: search()
        - Case 4: update()
        - Case 5: del()
        - Case 6: SaveToFile(), then break loop.
4. **End**.

Delete Function Algorithm

1. **Input** ID to delete.
2. **Search** for the index pos where m[i].id == ID.
3. **If** not found, print error and return.
4. **Loop** from i = pos to n - 1:
    - Shift element: m[i] = m[i + 1] (Overwrites the deleted item with the next one).
5. **Decrement** n (Total count).
6. **Print** "Deleted".

---

6. IMPLEMENTATION DETAILS

Variables Used

- Medicine m[MAX]: An array of structures representing the inventory.
- int n: A counter variable passed by reference (*n) to track the total number of medicines.
- FILE *fp: File pointer used for reading and writing inventory.txt.

Structure Definition

The project uses a custom header file medical.h to define the data structure:

C

```
typedef struct medical {
  int id;        // Unique ID for medicine
```

```
    char name[50];    // Name of the medicine
    int qty;          // Current stock quantity
    float price;      // Price per unit
} Medicine;
```

Functions Created

- load(Medicine m[], int *n): Reads formatted data from the text file into the structure array.
- SaveToFile(Medicine m[], int n): Writes the current state of the array back to the text file upon exit.
- update(Medicine m[], int n): Locates a medicine by ID and modifies its quantity using += operator.
- del(Medicine m[], int *n): Handles logical deletion by shifting array elements.

---

7. TESTING & RESULTS

Test Cases

Test 1: Adding a New Medicine

- **Input:** ID: 101, Name: Paracetamol, Qty: 50, Price: 10.5
- **Expected:** "Added." message displayed.
- **Result:** PASS

Test 2: Saving to File

- **Action:** Select Option 6 (Exit).
- **Expected:** A file named inventory.txt is created/updated with the data "101 Paracetamol 50 10.50".
- **Result:** PASS

Test 3: Searching (Valid)

- **Input:** Search ID 101.
- **Expected:** Display details: "101 Paracetamol 50 10.50".
- **Result:** PASS

Test 4: Deleting an Item

- **Input:** Delete ID 101.
- **Expected:** Item removed, total count decreases by 1. Search for 101 subsequently returns "Not Found".
- **Result:** PASS

8. CONCLUSION & FUTURE WORK

What I Achieved

I successfully developed a **Medical Store Management System** that solves the problem of manual inventory tracking. The project successfully implements:

- **Persistent Storage:** Data survives program restarts.
- **Modular Code:** Logic is separated into functions for better readability.
- **Data Integrity:** Users can safely add, remove, and update records.

What I Learned

- How to use **Structures** (struct) to model real-world objects like medicines.
- The logic behind **File Handling** in C (fopen, fclose, modes "r" and "w").
- How to manipulate arrays (shifting elements) to simulate deletion.

Future Improvements

- **Billing System:** Add a function to calculate total bill based on quantity sold.
- **GUI:** Upgrade the console interface to a graphical one using C# or Python.

---

9. REFERENCES

1. UPES Course Material: Programming in C (B.Tech 1st Year).
2. Pharmacy Inventory Management Articles.