Hospital Information And Appointment Management System

Yashika Bansal

December 01, 2025

Abstract

—---

The *Hospital Information & Appointment Management System* is a console-based software developed in C programming language to simplify and digitalize basic hospital operations. The system focuses on efficient management of patient records and appointment scheduling using file handling techniques.

It allows users to add new patients, store their medical details, search for specific patient information, and maintain permanent records through binary files. The system also provides an appointment-booking feature where each appointment is assigned a unique ID along with details of the doctor, date, and time. Auto-ID generation ensures accuracy and prevents duplication of records.

Overall, the project highlights the practical application of C programming concepts in developing real-time management systems.

CONTENTS

1. Problem Definition

—---

1.1 Overview

The *Hospital Information & Appointment Management System* is a simple C-based program that helps store patient details and manage doctor appointments. It allows adding patients, searching records, booking appointments, and viewing all stored data. The system uses binary files to keep information safe and permanent, and auto-generates unique IDs for each patient and appointment. It provides a quick, organized, and efficient way to handle basic hospital operations.

1.2 Objectives

- To maintain patient information in an organized and systematic way.
- To provide an efficient method for booking and managing appointments.
- To generate unique IDs automatically for patients and appointments.
- To enable quick searching and retrieval of patient records.
- To store data permanently using binary files for reliability.
- To reduce manual paperwork and simplify hospital record management.

1. <u>System Design</u>

—--

2.1 Algorithm

1. Start the program

2. Display main menu:
   - ➔ Add Patient
   - ➔ View Patients
   - ➔ Search Patient
   - ➔ Book Appointment
   - ➔ View Appointments
   - ➔ Exit

3. Input user choice.

4. Perform action based on choice:

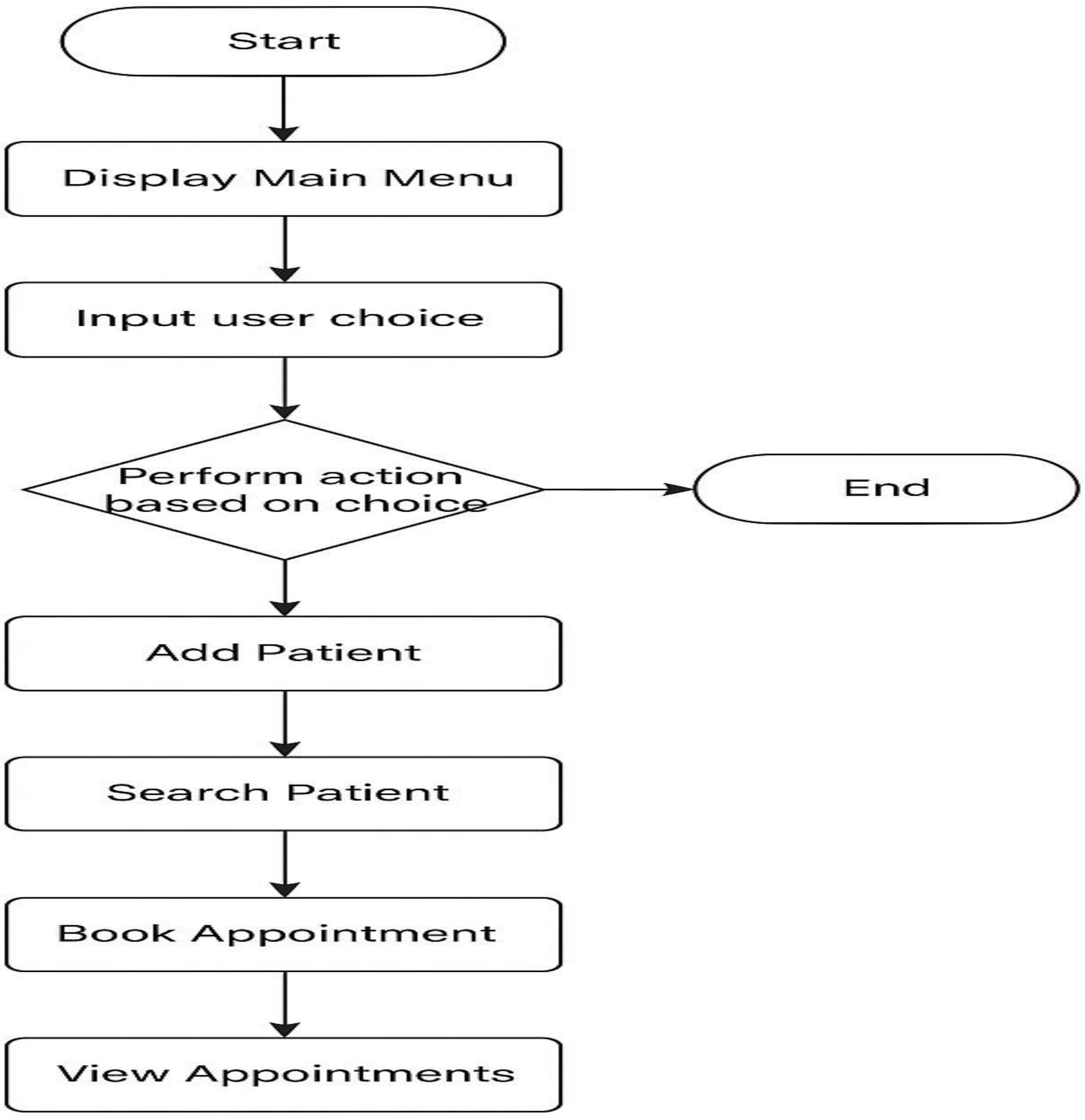
   - ➔ Add Patient:
     1. Open patient file in append mode.
     2. Generate unique patient ID.
     3. Input patient details (Name, Age, Gender, Disease).
     4. Write record to file.
     5. Close file and display success message.
   - ➔ View Patient:
     1. Open patient file in read mode.
     2. If empty, display "No patients found."
     3. Otherwise, display all patient records.
     4. Close fille
   - ➔ Search Patient:
     1. Open patient file in read mode.
     2. Input patient ID to search.
     3. Read records and compare IDs.
     4. If found, display details; else display "Not found."
     5. Close file.
   - ➔ Book Appointment:
     1. Open appointment file in append mode.
     2. Generate unique appointment ID.
     3. Input details (Patient ID, Doctor, Date, Time).
     4. Write record to file.
     5. Close file and display success message.

- ➔ View Appointments:
    1. Open appointment file in read mode.
    2. If empty, display “No appointments found.”
    3. Otherwise, display all appointment records.
    4. Close file.
- ➔ Exit: Terminate the program.

5. Repeat steps 2–4 until the user chooses to exit.

6. End.

2.2 Flowchart



2.3 Data Structure

```
typedef struct {
       int id;
        char name[50];
        int age;
         char gender[10];
         char disease[50];
  } Patient;
```

3. Implementation Details

—--

3.1 Key features

- Stores patient records using binary files.
- Schedules appointments with doctor, date, time.
- Creates unique IDs for patients and appointments.
- Finds patient details using their ID.
- Displays all patients and all appointments.
- Handles invalid input and missing files.
- Each feature implemented as a separate function.

3.2 Code Snippets

- Patient Structure

```
typedef struct {
    int id;
    char name[50];
    int age;
    char gender[10];
    char disease[50];
} Patient;
```

- Unique ID Generation

```
int generatePatientID() {
    FILE *fp = fopen("patient.dat", "rb");
    if (!fp) return 1;
    Patient p;
    int maxID = 0;
    while (fread(&p, sizeof(Patient), 1, fp)) {
        if (p.id > maxID) maxID = p.id;
    }
    fclose(fp);
    return maxID + 1;
}
```

- Add Patient

```
void addPatient() {
     FILE *fp = fopen("patient.dat", "ab")
     if (!fp) return;
    Patient p;
    p.id = generatePatientID();
    getchar();
    printf("Enter Name: ");
    fgets(p.name, sizeof(p.name), stdin);
    printf("Enter Age: ");
    scanf("%d", &p.age);
    getchar();
    printf("Enter Gender: ");
    fgets(p.gender, sizeof(p.gender), stdin);
    printf("Enter Disease: ");
    fgets(p.disease, sizeof(p.disease), stdin);
    fwrite(&p, sizeof(Patient), 1, fp);
    fclose(fp); }
```

- Search Patient by ID

```
void searchPatient() {
   FILE *fp = fopen("patient.dat", "rb");
   int id, found = 0;
printf("Enter ID: ");
 scanf("%d", &id);
  Patient p;
 while (fread(&p, sizeof(Patient), 1, fp)) {
     if (p.id == id) {
         printf("Name: %s\nAge: %d\n", p.name, p.age);
         found = 1;
         break;
     }
 }
 fclose(fp);
   if (!found);
       printf("Patient not found.\n");
}
```

- Book Appointment

```
void bookAppointment() {
        Appointment a;
        a.appointID = generateAppointmentID();
        printf("Enter Patient ID: ");
        scanf("%d", &a.patientID);
        FILE *fp = fopen("appoint.dat", "ab");
        fwrite(&a, sizeof(Appointment), 1, fp);
        fclose(fp); }
```

- Main Menu

```
Int main() {
 int choice;
 while (1) {
     printf("\n===== Hospital Management System =====\n");
    printf("1. Add Patient\n");
    printf("2. Search Patient\n");
    printf("3. View All Patients\n");
    printf("4. Book Appointment\n");
    printf("5. View All Appointments\n");
    printf("6. Exit\n);
    printf("Enter Choice\n);
    scanf("%d", &choice);
    switch (choice) {
           case 1: addPatient(); break;
           case 2: searchPatient(); break;
           case 3: viewPatients(); break;
           case 4: bookAppointment(); break;
           case 5: viewAppointments(); break;
           case 6: exit(0);
           default: printf("Invalid choice.\n");
                           } } }
```

4. Testing & Results

—---

Test Case 1: Add Patient

- Input: Valid Patient details.
- Output: “Patient Added Successfully! Assigned ID: X”
- Result: 1. Patient data stored successfully in *patient.dat*
  2. Unique ID generated correctly.

Test Case 2: View All Patients

- Action: Select “View All Patients”.
- Output:List of all stored patient records.
- Result: Complete list displayed with ID, Name, Age, Gender, Disease.

Test Case 3: Search Patient by ID

- Input: Patient ID = 1
- Output: Patient details of ID 1.
- Result: 1. Correct patient details displayed

  2. If ID not found → “No patient with ID found”

Test Case 4: Book Appointment

- Input: Patient ID: 1
  Doctor: Dr. XYZ
  Date: XX/XX/XXXX
  Time: XX
- Output: “Appointment Booked Successfully! Appointment ID: XXXX”
- Result: 1.Appointment saved to *appoint.dat*
  2. Auto-generated appointment ID working correctly

Test Case 5: View Appointments

- Action: Select “View All Appointments”
- Output: List of all stored appointments.
- Result: 1. All appointment details displayed properly
  2. Correct mapping of Patient ID and Appointment ID

Test Case 6: Invalid Choice Handling

- Input: Menu choice = 9
- Output:“Invalid choice! Try again.”
- Result: System handled invalid input correctly

5. Conclusion & Future Work

—---

5.1 Conclusion

The Hospital Information and Appointment Management System successfully provides an efficient way to store patient data, search records, and manage appointments using file handling in C. The system is simple, fast, and reliable, allowing easy data retrieval and smooth management of hospital operations. Overall, the project meets all requirements and demonstrates effective use of modular programming and structured data management.

5.2 Future Work

- Implement a graphical user interface (GUI) to improve user interaction and usability.
- Add user authentication for secure access by admin, doctors, and reception staff.
- Support for multiple data export formats (CSV, Excel) for reporting and record analysis.

6. References

- UPES C Major Project Guidelines
- C Standard Library Documentation.
- Online resources