PROJECT REPORT

Implementation of Hospital Management System in C

Submitted By: Satyam Singh

SAP ID: 590028838

AIM - To design and implement a Hospital Management System in C that stores and manages patient and hospital details using arrays, pointers, structures, and functions.

OBJECTIVES

Store multiple patient and hospital records using arrays of structures.

Use functions for modular operations like add, display, and sort.

Use pointers to pass arrays and structures efficiently to functions.

Implement basic validation for user inputs (IDs, beds, rating).

Demonstrate sorting of hospital records based on number of beds or rating.

ALGORITHM

1. Start the program.
2. Define two structures: one for Patient and one for Hospital with appropriate fields.
3. Declare arrays of structures to store multiple patient and hospital records.
4. Initialize counters for number of patients and hospitals to zero.

Display a menu with options

Add Patient

Display Patients

Add Hospital

Display Hospitals

Sort Hospitals by Beds Exit

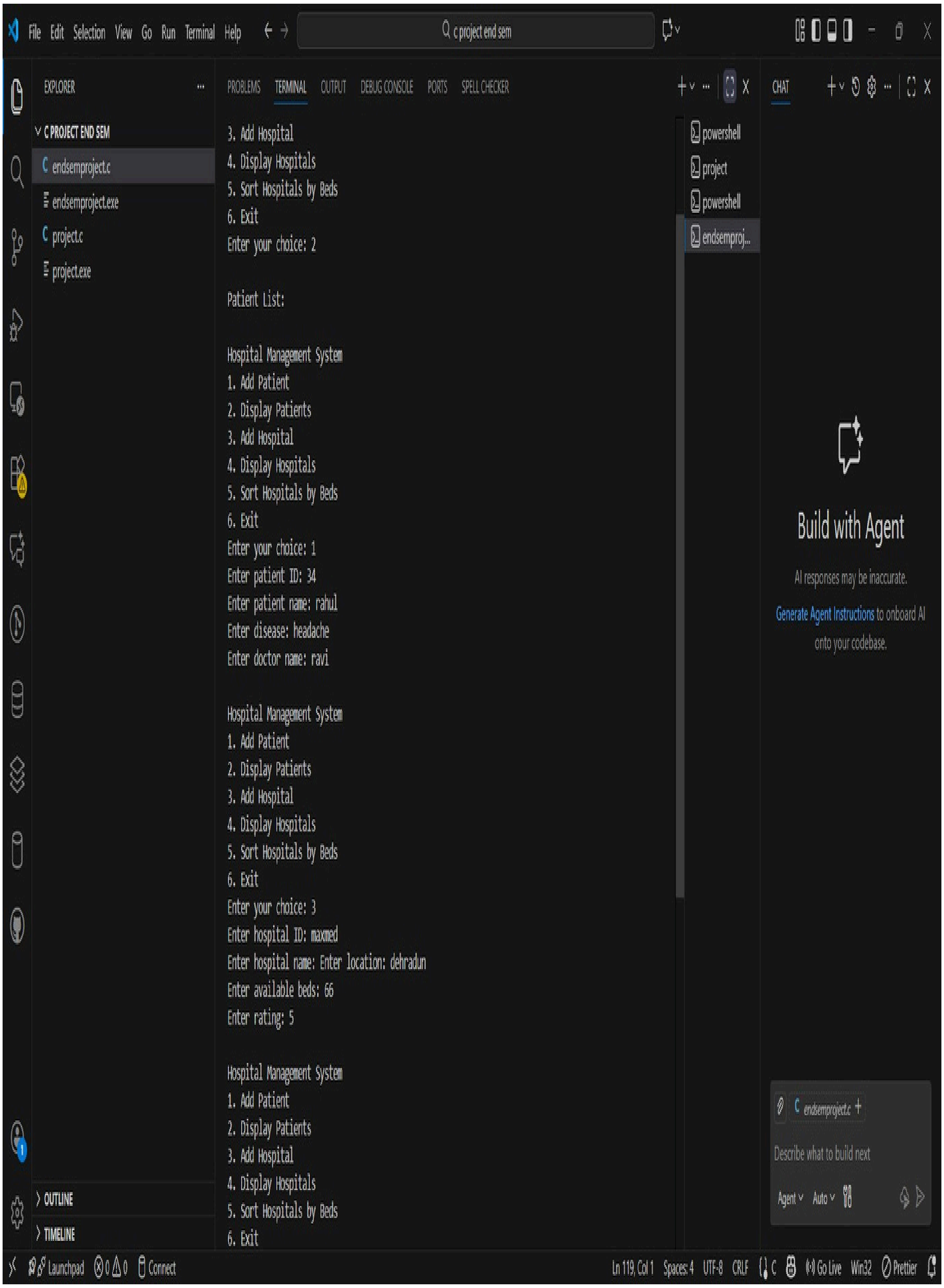
1. Read user choice.
2. If “Add Patient”: Enter details, store in array, increment count.
3. If “Display Patients”: Print all patient records.
4. If “Add Hospital”: Enter details, store in array, increment count.
5. If “Display Hospitals”: Print all hospital records.
6. If “Sort Hospitals by Beds”: Sort using bubble sort and display.
7. If “Exit”: Stop program.
8. Repeat until exit is chosen

code

```
#include <stdio.h>

#include <string.h>

// Structure for patient struct Patient {

    int id;

    char name[50];     char disease[50];     char doctor[50];

};

// Structure for hospital struct Hospital {

    int id;

    char name[50];     char location[50];     int beds;     float rating;

};

// Function to add patient void addPatient(struct Patient *p, int index) {     printf("Enter patient ID: ");
scanf("%d", &p[index].id);     printf("Enter patient name: ");     scanf("%s", p[index].name);     printf("Enter
disease: ");     scanf("%s", p[index].disease);

    printf("Enter doctor name: ");     scanf("%s", p[index].doctor);

}

// Function to display patients void displayPatients(struct Patient *p, int count) {     printf("\nPatient List:\n");
for (int i = 0; i < count; i++) {         printf("ID: %d, Name: %s, Disease: %s, Doctor: %s\n",               p[i].id,
p[i].name, p[i].disease, p[i].doctor);

    }

}

// Function to add hospital void addHospital(struct Hospital *h, int index) {     printf("Enter hospital ID: ");
scanf("%d", &h[index].id);     printf("Enter hospital name: ");     scanf("%s", h[index].name);     printf("Enter
location: ");     scanf("%s", h[index].location);     printf("Enter available beds: ");     scanf("%d", &h[index].beds);
printf("Enter rating: ");     scanf("%f", &h[index].rating);

}

// Function to display hospitals void displayHospitals(struct Hospital *h, int count) {     printf("\nHospital
List:\n");
```

```
   for (int i = 0; i < count; i++) {         printf("ID: %d, Name: %s, Location: %s, Beds: %d, Rating: %.1f\n",
h[i].id, h[i].name, h[i].location, h[i].beds, h[i].rating);

   }

}

// Function to sort hospitals by beds void sortHospitalsByBeds(struct Hospital *h, int count) {     struct Hospital
temp;     for (int i = 0; i < count - 1; i++) {         for (int j = i + 1; j < count; j++) {             if (h[i].beds < h[j].beds) {
temp = h[i];                 h[i] = h[j];                 h[j] = temp;

         }

      }

   }

   printf("\nHospitals sorted by beds:\n");     displayHospitals(h, count);

}

int main() {     struct Patient patients[100];     struct Hospital hospitals[100];     int patientCount = 0,
hospitalCount = 0;     int choice;

   while (1) {         printf("\nHospital Management System\n");         printf("1. Add Patient\n2. Display
Patients\n3. Add Hospital\n4. Display Hospitals\n5. Sort Hospitals by Beds\n6. Exit\n");         printf("Enter your
choice: ");         scanf("%d", &choice);

      switch (choice) {             case 1:

            addPatient(patients, patientCount);                 patientCount++;                 break;             case 2:

            displayPatients(patients, patientCount);

            break;             case 3:

            addHospital(hospitals, hospitalCount);                 hospitalCount++;                 break;             case 4:

            displayHospitals(hospitals, hospitalCount);

            break;             case 5:

            sortHospitalsByBeds(hospitals, hospitalCount);

            break;             case 6:

            return 0;             default:

            printf("Invalid choice!\n");

      }

   }

   return 0;

}
```

Output



```
3. Add Hospital
4. Display Hospitals
5. Sort Hospitals by Beds
6. Exit
Enter your choice: 2

Patient List:

Hospital Management System
1. Add Patient
2. Display Patients
3. Add Hospital
4. Display Hospitals
5. Sort Hospitals by Beds
6. Exit
Enter your choice: 1
Enter patient ID: 34
Enter patient name: rahul
Enter disease: headache
Enter doctor name: ravi

Hospital Management System
1. Add Patient
2. Display Patients
3. Add Hospital
4. Display Hospitals
5. Sort Hospitals by Beds
6. Exit
Enter your choice: 3
Enter hospital ID: maxmed
Enter hospital name: Enter location: dehradun
Enter available beds: 66
Enter rating: 5

Hospital Management System
1. Add Patient
2. Display Patients
3. Add Hospital
4. Display Hospitals
5. Sort Hospitals by Beds
6. Exit
```

File Edit Selection View Go Run Terminal Help
c project end sem
EXPLORER
C PROJECT END SEM
endsemproject.c
endsemproject.exe
project.c
project.exe
PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS SPELL CHECKER
powershell
project
powershell
endsemproj...
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
endsemproject.c
Describe what to build next
Agent Auto
OUTLINE
TIMELINE
Launchpad 0 0 Connect
Ln 119, Col 1 Spaces: 4 UTF-8 CRLF C Go Live Win32 Prettier

RESULT

The program successfully implements a basic Hospital Management System using arrays of structures, pointers, and functions.

CONCLUSION

This experiment strengthened understanding of arrays of structures, pointer-based passing, modular programming, and data handling in C.