Hospital Management System

**1)Research**

*The Current hospital management systems, as illustrated by this code here, have played a very important role in the current healthcare scenario. Their efficiency in managing data concerning patients and doctors enhances better operational efficiency, enhanced patient care, and improves adherence to regulatory requirements. Capabilities of HMS (Hospital Management System) will continue to grow with continuing technological improvements toward a future that shapes healthcare management.*

*Source(*

**2)Analysis**

*The code of the hospital management system analysis gives various strengths and areas for improvement. For example, the clear, well-designed, and structured code makes use of clear data structures for patients and doctors that generally leads to organized handling and readability of data. It makes use of input validation mechanisms, in place to ensure the correctness of data input by users, which will lead to a low probability of errors and guaranteeing data integrity. The console-based interface provides a simple UI so that it can be easily used by the staff even if their technical expertise varies. However, the system proposed could be enhanced regarding its error handling functionalities based on many input scenarios to handle such inputs and might include dynamic memory for better scalability along with data persistence features for saving and retrieving information across sessions. The system can be extended for editing and deleting records functionality in order to increase its usability further. It is therefore a good foundation for managing the operation of a medical institution, and after considering these measures of improvement, it would make it even more robust and comprehensive in healthcare management.*

**3)Ideate**

*The major objective was to improve on the functionality, usability, and adaptability of the hospital management system in line with modern healthcare requirements. The integration of these features meant that the system could significantly enhance efficiencies in operation and patient care and make it more valuable for health service providers.*

**4)Build**

*This code creates a hospital management system that works efficaciously for managing information related to patients and doctors. It holds two primary data structures: Doctor, which constitutes fields such as ID, name, specialization, and fee, and Patient, which consists of id, name, age, disease, insurance, mobile number, identity proof, billing amount, and assigned doctor. The system adds and displays patients and doctors, searches for patients based on the ID, and validates input fields such as IDs and insurance details. Since the program has an easy user-friendly console interface, it can easily interact with the users. Overall, the system aims at streamlining the administrative activities within a hospital environment by keeping all records very well and accessing the critical information.*

**5)Test**

1. **Manual Testing**: *Execute the test cases. Enter all different values and execute the program. Then analyze its outcomes.*  
2. **Boundary testing**: *The testing of boundary conditions such as for the maximum number of characters for names, IDs, or fees is conducted and it is tested how the system responds in such a scenario.*  
3. **Negative Testing**: *They will input with wrong data formats, negative numbers, and so on so that the system gives appropriate responses with error messages.*  
4. **Performance Testing**: *Although it is not described in detail, the following could be proposed as testing for maximum patients and doctors as a system performance test to find out how stable and efficient the system will be.*

*Testing these cases systematically would thus ensure overall processes, so the hospital management system works and handles a lot of real-world scenarios effectively.*

**6)Implement**

**User Interaction:**

*Menu-Driven Interface: The program uses a console menu to allow users to select various operations, enhancing usability.*

*Input Validation: Input for age and billing amounts includes validation to ensure only valid data is accepted.*

**Memory Management:**

*Dynamic Allocation: Use of malloc for appointment storage, with corresponding free operations to prevent memory leaks.*

**Freeing Memory:**

*A dedicated function (free\_memory) ensures that all dynamically allocated memory is released before the program exits.*

**Display Statistics:**

*A function to calculate and display total registered patients, doctors, and average billing amount provides insights into the system's usage.*

**Appointment Management:**

*Allow patients to view their scheduled appointments.*

*Implement appointment conflict checks (e.g., ensuring that a doctor is not double-booked)*

<https://github.com/Omkar23748ghb/Hospital-Management-System.git>