**Hospital Management System – Final Project Report**

**Deliverable 1:**

**1. Project Objectives and Scope**

The primary objective of this project is to develop a foundational Hospital Management System that efficiently manages patient records and doctor schedules. The system leverages modular programming and arrays to ensure simplicity, efficiency, and scalability. The key functionalities implemented include:

* Storing and managing patient data for up to 50 patients.
* Basic operations such as adding, searching, viewing, and discharging patient records.
* Managing doctor schedules using a 2D array.
* Implementing input validation to ensure data integrity.
* Providing a menu-driven interface for ease of use by hospital administrators.

This project is designed to be a functional prototype, demonstrating the core functionalities of a hospital management system that can later be extended with databases and more advanced features.

**2. Design and Implementation Details**

The system is designed using C programming language with a structured approach. Key components include:

* **Data Structures:**
  + Arrays are used to store patient records and doctor schedules.
  + A 2D array is used to manage doctor shifts across different days of the week.
* **Functions:**
  + addPatientRecord(): Allows users to enter and store patient details.
  + viewPatientRecords(): Displays all stored patient records in a structured format.
  + searchPatientRecord(): Searches for a patient by ID or name.
  + dischargePatientRecord(): Removes a discharged patient from the system.
  + manageDoctorSchedule(): Assigns doctors to specific shifts in a week.
  + viewSchedule(): Displays the full doctor schedule.
  + menu(): Provides a text-based menu for user interaction.
  + scanInt(): Accepts integers only.
  + clearInputBuffer(): Clears the input buffer.
  + returnToMenu(): Make user hit the enter key before going back to the menu.
* **Input Validation:**
  + Ensuring unique patient IDs.
  + Validating age entries as positive integers.
  + Preventing over-assignment of hospital rooms.
  + Ensures only the data type the program wants is retrieved.
* **User Interface:**
  + A command-line interface (CLI) is implemented for simplicity and ease of use.
  + A menu-driven system allows users to navigate different functionalities efficiently.

**3. Challenges and Solutions**

During the development of the project, several challenges were encountered:

* **Ensuring Unique Patient IDs:**
  + Solution: Implemented a function idExists() to check for duplicate IDs before adding a new patient.
* **Managing Limited Room Assignments:**
  + Solution: Introduced roomNumExists() function to ensure no more than two patients are assigned to a single room.
* **Handling User Input Errors:**
  + Solution: Implemented input validation for numeric fields to prevent incorrect entries and added error messages for invalid inputs.
* **Managing Doctor Schedules:**
  + Solution: Used a 2D array to store and display doctor assignments effectively.
* **Making sure only integers are entered:**
  + Solution: Scan for both integer and character. Checks if 2 items were entered or If the character is not ‘\n’ then return a negative one indicating a character was entered.

**4. Testing Procedures and Results**

To ensure the functionality of the system, the following testing methods were employed:

* **Unit Testing:** Each function was tested individually to verify expected behavior.
* **Integration Testing:** All functions were tested together to confirm seamless interaction.
* **Boundary Testing:** Input limits were tested, such as maximum patients and valid age ranges.
* **User Testing:** Simulated hospital administrator interactions to verify usability.

**Test Results:**

* Successfully added and retrieved patient records.
* Search functionality accurately located patient details.
* Discharge operation correctly removed patient records.
* Doctor schedule management allowed assignments and displayed schedules as expected.
* Input validation prevented incorrect or duplicate data entry.

**5. Conclusion**

The project successfully met its objectives by delivering a functional Hospital Management System with patient record management and doctor scheduling. The modular design ensures easy maintenance and scalability. Future improvements can include:

* Implementing a database for persistent storage.
* Enhancing the interface with a graphical user interface (GUI).
* Introducing additional features such as billing and appointment scheduling.

Overall, the project provided valuable experience in modular programming, array manipulation, input validation, and system design, laying the groundwork for further development into a comprehensive hospital management solution.