Hospital-Record-Management-System-using-C-programming-Python

1. Overview of the System The Hospital Record Management System is designed to manage health records, appointments, and user roles in a structured and accessible way. The project has two main implementations:

C Program: A console-based implementation with file handling and basic functionalities. Python Program with GUI: An enhanced system with a user-friendly graphical interface, implemented using Python's tkinter module. 2. Key Functionalities and Logic Implementation A. User Login Logic:

Reads from a user credentials file (users.txt) for admins, doctors, and receptionists. Checks if the entered username and password match any existing credentials. Role-based access control (admin, doctor, receptionist) is implemented by associating a role field (admin, doctor, receptionist) in the file. File Implementation:

Credentials stored in users.txt in the format: Copy code username password role During login, the program parses the file line by line to validate the user's credentials and retrieve their role. B. Creating a New Client Login Logic: Clients are handled separately from system roles. During registration, the client enters their desired username and password. The data is appended to client.txt in the format: Copy code username password Purpose: Separates clients from system operators for clarity and security. Allows clients to view or cancel their own health records or appointments. 3. Role-Based Accessibility Each role (admin, doctor, receptionist, client) has specific accessible features:

Admin:

Create, update, delete, search, and display health records. Add, update, cancel, and display appointments. Export health records to a CSV file. Full control over hospital data. Doctor:

View and update health records. Search health records. View appointments. Receptionist:

Add and display appointments. Search health records. Client:

View their own health records. Cancel their appointments. 4. Text and CSV File Implementation Text Files:

Data for health records, appointments, and user credentials is stored in .txt files. Each file is read line-by-line to retrieve data and updated line-by-line for modifications. Format examples: health_records.txt: Copy code record_id,patient_name,diagnosis,treatment_plan,doctor_assigned appointments.txt: lua Copy code appointment_id,patient_name,doctor_name,date,time CSV File for Health Records:

Purpose: Exports health records to a CSV file for easier sharing, reporting, and data analysis. Implementation: Health records from health_records.txt are read and written into a structured CSV file using Python's csv module. The exported file can be opened in spreadsheet applications like Excel for visualization and editing. 5. Functional Inputs and Outputs Create Health Record:

Input: Record ID, patient name, diagnosis, treatment plan, doctor assigned. Output: Record added to health_records.txt. Add Appointment:

Input: Appointment ID, patient name, doctor name, date, and time. Output: Appointment added to appointments.txt. Search Health Record:

Input: Record ID or patient name. Output: Matching record details. Export Health Records to CSV:

Input: None. Output: health_records.csv file created from existing health records. Cancel Appointment:

Input: Appointment ID. Output: Appointment removed from appointments.txt. 6. Why We Used Python Instead of GUI in C Programming C Programming Limitations:

GUI development in C is cumbersome, requiring libraries like GTK or WinAPI, which are platform-dependent and have steep learning curves. The integration of GUI libraries in a portable and maintainable manner adds significant complexity to the project. Python Advantages:

Ease of GUI Development: Python's tkinter library simplifies GUI creation with minimal boilerplate code. Cross-Platform: Python GUIs work seamlessly across operating systems without additional dependencies. Faster Development: Python is higher-level, making it faster to implement complex features compared to C. 7. Purpose of visualize.py Objective:

Adds a visual representation of the data. Allows graphical plotting (e.g., appointments by date, patient records by doctor) using libraries like matplotlib or seaborn. Benefits:

Helps analyze trends (e.g., busiest doctors, most common diagnoses). Provides insights for administrative decisions. Conclusion The Hospital Record Management System leverages text and CSV files for structured data storage, Python for ease of GUI development, and clear role-based access for security and efficiency. The modular design allows for scalability, while the CSV export ensures interoperability with other tools like Excel for detailed analysis and reporting.

This system effectively meets the needs of hospital staff and clients, balancing functionality, usability, and maintainability.