Hospital Management System

Overview:

The **Hospital Management System** is a mini C++ project designed for managing patient records and diagnosis information in hospitals. It ensures secure access through a login system and provides features for adding and viewing patient details and hospital information. This system relies on file handling for data storage and retrieval, making it a simple yet functional application.

Features:

1. **Secure Login System**: Ensures only authorized users can access the program.
2. **Patient Records**: Add, store, and retrieve personal and medical details of patients.
3. **Diagnosis Information**: Record symptoms, diagnosis, prescribed medicines, and ward information.
4. **Hospital Details**: Displays general information about the hospital from a predefined file.
5. **Basic File Handling**: Uses external text files for managing patient and hospital data.

Instructions for Running:

1. **Compilation**:
    - Use any C++ compiler (e.g., Dev-C++, Turbo C++, GCC) to compile the code.
    - Ensure all files, including `hos.txt` (hospital information file), are in the same directory as the executable.
2. **Execution**:
    - Run the compiled program in **Full-Screen Mode** to avoid layout distortion.
    - Enter the correct password (**"pass"**) to access the system.
3. **Menu Options**:
    - **Add New Patient Record**: Input personal and medical details for a new patient.

- **Add Diagnosis Information**: Update existing patient records with additional medical details.
- **Full History of the Patient**: View all stored information about a specific patient.
- **Information About the Hospital**: Displays hospital details from `hos.txt`.
- **Exit**: Terminates the application.

4. **File Management**:
   - Patient records are stored as text files in the project folder. Deletion requires manual file removal.
   - Keep `hos.txt` updated for accurate hospital details.

Developer Guide

Program Logic and Structure:

1. **Login System**:
   - Implements a simple password-based authentication mechanism to restrict access.
   - Password: "pass" (hardcoded in the program).
2. **Patient Records**:
   - Uses file handling to store patient data in text files.
   - Each record includes:
     - Name, address, contact number, age, sex, blood group, disease, and patient ID.
3. **Diagnosis Information**:
   - Updates patient files with additional details such as symptoms, diagnosis, prescribed medicines, and ward type.
4. **Hospital Details**:
   - Displays predefined hospital information stored in `hos.txt`.
5. **File Handling**:
   - Patient data is written to and retrieved from individual text files.
   - Records are named uniquely to prevent overwriting.

    - Manual deletion of files is required for record removal.
6. **User Interface**:
    - Displays a menu-driven interface for navigation.
    - Prompts guide users through each operation.

Code Structure:

- **main() Function**:
    - Controls the flow of the program, starting with the login process.
    - Displays the main menu and calls corresponding functions based on user input.
- **Functions**:
    - **`login()`:** Handles user authentication.
    - **`addPatientRecord()`:** Collects and stores patient details.
    - **`addDiagnosisInfo()`:** Updates diagnosis details for existing patients.
    - **`viewPatientHistory()`:** Displays all stored information about a specific patient.
    - **`viewHospitalInfo()`:** Reads and displays the hospital information file.

User Manual

Getting Started:

1. Compile the program using a C++ compiler.
2. Ensure all required files (e.g., `hos.txt`) are present in the program's directory.
3. Run the executable file in **Full-Screen Mode** for the best experience.

Login:

- Upon launching the program, you will be prompted to enter the password.
- Enter the correct password (**"pass"**) to access the system.

Using the System:

1. **Add New Patient Record**:
    - Select this option from the menu.
    - Enter the patient's details when prompted.
    - A new file will be created to store this data.
2. **Add Diagnosis Information**:
    - Choose an existing patient file from the directory.
    - Input diagnosis details such as symptoms, diagnosis, medicines, admission status, and ward type.
3. **View Patient History**:
    - Select this option to display all stored information about a patient.
4. **View Hospital Information**:
    - Displays predefined hospital details stored in `hos.txt`.
5. **Exit**:
    - Use this option to close the program.

Managing Files:

- Patient records are stored as text files in the project folder.
- To delete a record, manually locate and remove the corresponding file.
- Ensure `hos.txt` is always present and updated for accurate hospital details.

Notes:

- Always run the program in Full-Screen Mode to ensure proper layout.
- Use a text editor to manually edit or delete files when necessary.