Project Setup Instructions

1. Assumptions Made

1.1 User Authentication: All staff members added by the administrator are assigned a non-changeable User ID and a default password, which can be changed after their initial login.
1.2 Doctor Availability: Doctors are assumed to be available for consultations every day, with their appointment slots automatically initialized daily. Doctors have the option to mark specific slots as unavailable on particular dates if needed.
1.3 Lunch Break Scheduling: All doctors have a scheduled lunch break at 1:00 PM, during which appointments cannot be booked. This time slot is automatically marked as unavailable.
1.4 Inventory Replenishment: When the administrator approves a replenishment request and the stock level is found to be below the set threshold, the inventory is automatically updated to twice the threshold value to ensure sufficient stock availability.
1.5 Single Administrator: The system assumes that there is only one administrator account with full control over functionalities of managing staff, inventory, and appointments. There is no functionality for multiple administrators or delegation of administrative tasks.
1.6 Appointment Slot Duration: Each appointment slot is assumed to have a fixed duration of 30 minutes. This duration is consistent for all doctors and cannot be adjusted individually.
1.7 Default Appointment Status: All appointments that are scheduled by the patients are to start with a default status of "scheduled”. An appointment is confirmed only when the doctor changes the status to “confirmed”. Only then the slot becomes unavailable to other patients.
1.8 Inventory Quantity in Integer Units: Inventory quantities are assumed to be in whole numbers, with no support for decimals. This is relevant for dosage management in prescriptions.
1.9 Fixed Threshold Values for Replenishment: The threshold values for inventory replenishment are assumed to be set by the administrator and are consistent across different medicines. There is no functionality for adjusting threshold values per item.
1.10 Prescribing Medicine during Appointment: It is assumed that a Doctor always prescribes 1 or more medicines in every appointment. Doctor always sets the prescribed medicine status to = “Prescribed”. We assume there will be no typing errors in this.

🛠️ Setup Instructions

Follow these steps to clone, compile, and run the project through the terminal:

###1️⃣ **Clone the Project Repository from GitHub**

`git clone https://github.com/QianYUHA/SC2002-project-HMS.git`

###2️⃣ **Navigate to the Repository:**

`cd SC2002-project-HMS`

###3️⃣ **Extract the src Folder on your Desktop or Downloads**

`cp -r src ~/Desktop/`

###4️⃣ **Navigate to the src Folder:**

`cd ~/Desktop/src`

###5️⃣ **Compile the project:**

`javac MainApp/Main.java`

###6️⃣ **Run the project:**

`java MainApp.Main`

NOTE:

If not using GitHub can follow the below instructions to download the implementation code from the submission folder:

1. **Unpack the Project Files:**
    - Download and unzip the project package. Make sure all necessary files are available in the project directory, including staff.txt, patients.txt, and inventory.txt.
2. **Pre-populated Data:**
    - **Staff Information** (staff.txt): The project comes with a pre-populated list of staff members for easier implementation. This includes various doctors, an administrator, a receptionist, and a pharmacist. The administrator has the option to add new staff, but this step has been streamlined by the existing data.
        - **Administrator Login Details**:
            - User ID: 1
            - Password: admin123
        - **Note**: Only one administrator exists in the system with the credentials mentioned above. This administrator has full control over adding new staff, medicines, and managing inventory.
    - **Inventory Information** (inventory.txt): Some medicines have been pre-added to the inventory to simplify implementation. This inventory file includes fields like medicine ID, name, quantity, etc.
        - **Adding Medicines**: When a doctor prescribes a medicine that is not available in the inventory, the pharmacist is notified of its absence while attempting to dispense it. The pharmacist then sends a replenishment request to the administrator for the specific medicine. Upon reviewing the request, the administrator is provided with the option to add the requested medicine to the inventory before approving the replenishment.
    - **Patient Information** (patients.txt): The project includes pre-existing patient records to facilitate initial testing. New patients can be registered as needed.
3. **Starting the Application: (Please first read the assumptions made in the project)**

    Note:
    You may need to press the <Enter> key twice **sometimes** to go back to a menu whenever prompted.

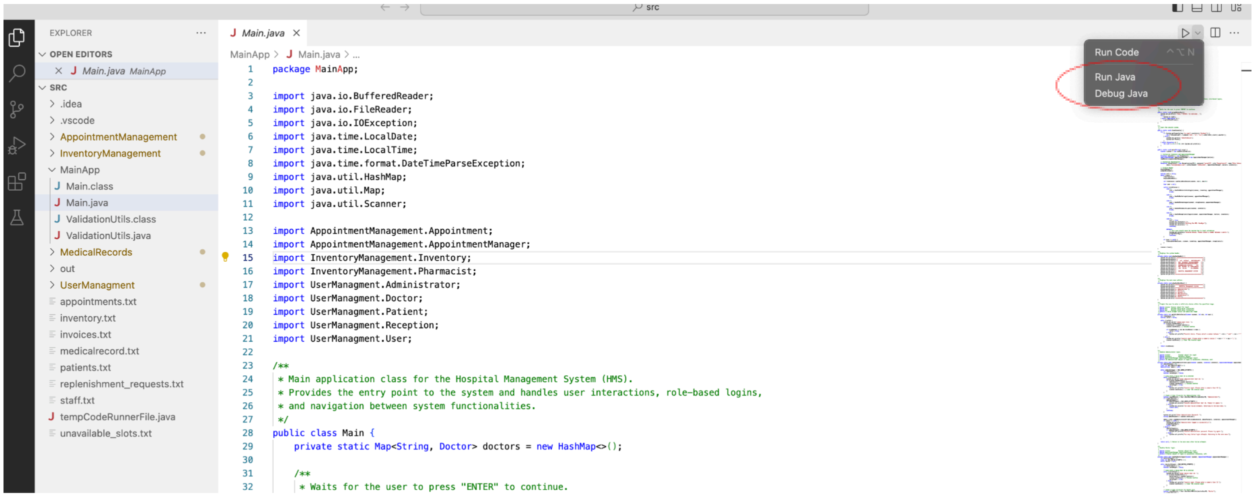
    - Launch the main program file (e.g., Main.java or equivalent), which should initialize the system and load the pre-existing data from the text files.

- **Login as Different Roles**: Test the system by logging in with various roles:

1. **Administrator**: Use the administrator credentials (User ID: 1, Password: admin123) to access full system functionality for managing staff and inventory.
2. **Patient:** Can use pre-existing credentials for patients to test appointment scheduling or can register new patients.
3. **Other Roles** (Doctor, Pharmacist, Receptionist): You can use the credentials listed in staff.txt to log in as a doctor, pharmacist, or receptionist and test role-specific functionalities. Alternatively, log in as the administrator to add new staff members and then test their login functionality.

NOTE:

If you are running on VS Code, ensure that you run the Main by using "Run Java".



```java
package MainApp;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.format.DateTimeParseException;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

import AppointmentManagement.Appointment;
import AppointmentManagement.AppointmentManager;
import InventoryManagement.Inventory;
import InventoryManagement.Pharmacist;
import UserManagment.Administrator;
import UserManagment.Doctor;
import UserManagment.Patient;
import UserManagment.Reception;
import UserManagment.User;

/**
 * Main application class for the Hospital Management System (HMS).
 * Provides the entry point to the system and handles user interactions, role-based logins,
 * and navigation between system functionalities.
 */
public class Main {
    private static Map<String, Doctor> doctors = new HashMap<>();

    /**
     * Waits for the user to press "ENTER" to continue.
```