Suprabit Junior Full-Stack Java Task

Objective:

Develop a Smart Home Monitoring System that allows users to view, add, update, and delete smart devices, as well as control their specific states (e.g., on/off for lights, state of the windows & doors, temperature setting & on/off for heating, temperature setting & on/off for the AC, recording status for security cameras).

Requirements:

Backend (Java):

1. **Project Setup:**
    - Use any Java framework to create the backend service.
2. **Entities:**
    - **Device:** id, name, type
    - **Device State:**
        - **Light:** state (on/off), brightness (0-100)
        - **Thermostat:** temperature (double), mode (heating/cooling/off)
        - **Camera:** state (active/inactive), recording (boolean)
        - **Blinds:** state(active/inactive), position (0-100)
3. **Repositories:**
    - Create repositories to manage `Device` data.
4. **Services:**
    - Implement services for user management and device management.
5. **Controllers:**
    - **Device:**
        - Add a new device.
        - Update a device.
        - Delete a device.
        - Update the specific state of a device.

Database

- Use any SQL database (SQLite, PostgreSQL, etc.)

Frontend (Angular):

1. **Project Setup:**
    - Use Angular CLI to scaffold the frontend application.

2. **Primary Features:**
    - Device management: Allow users to view, add, update, and delete smart devices.
    - Device control: Provide interfaces to control the specific states of different devices (e.g., turning a light on/off, adjusting thermostat temperature, activating/deactivating a camera).
3. **Routing:**
    - Implement routing to navigate between different views such as device list, and device control.
4. **State Management:**
    - Utilize Angular services or a state management library (e.g., NgRx) to manage application state.

Optional Requirements:

1. **Dockerization:**
    - Dockerize both the backend and the frontend applications.
    - Use Docker Compose to define and run multi-container Docker applications, ensuring the backend, frontend and database can communicate and operate together seamlessly.

    Docker Setup:

    - Create a `Dockerfile` for the backend service.
    - Create a `Dockerfile` for the frontend service.
    - Setup a database service if needed.
    - Create a `docker-compose.yml` file to define services, networks, and volumes needed to run the backend and frontend together.

Deliverables:

1. Full source code for the project, ideally via a GitHub link.
2. A readme file with clear instructions on how to set up, run, and access the application.

Evaluation Criteria:

1. Correctness and completeness of the implementation.
2. Quality of code (readability, modularity, and documentation).
3. Proper use of design patterns.
4. Effectiveness of state management and user experience.
5. Completeness and clarity of setup and run instructions.