Elevator System Project

Architecture and Main Algorithm Overview

System Architecture

This project models a realistic multi-elevator building using an object-oriented and modular architecture. The system is designed for extensibility and clarity, using encapsulated components that represent core parts of the simulation.

Models

- 🏢 **Building:** Serves as the top-level container managing all elevators and floors. It coordinates their interactions and maintains the overall simulation state.

- 🛗 **Floor:** Represents an individual level in the building. Each floor tracks whether an elevator has been called and stores its estimated wait time.

- 🚪 **Elevator:** Encapsulates the behavior of a single elevator, including its current location, travel direction, operational state (idle, moving, etc.), and its queue of destination floors.

Services

- 🧠 **ElevatorScheduler:** Implements the core control logic for the system. It assigns elevator requests to appropriate elevators, periodically updates their positions, and ensures smooth coordination of all moving parts.

Factories

- 🏭 **BuildingFactory:** Uses the Factory design pattern to create and initialize instances of Building, Floor, Elevator, and ElevatorScheduler. It also supports reconfiguration of system parameters (like changing the number of floors or elevators) without restarting the simulation, maintaining runtime continuity.

Frontend Components

- 💻 The user interface is built in React and provides an interactive visualization of the building. It includes controls for calling elevators from specific floors and displays elevator positions in real time, synchronized with the backend simulation logic.

Main Scheduling Algorithm

At the heart of the simulation lies a scheduling algorithm that assigns elevator calls efficiently, minimizing wait times and optimizing elevator traffic. The algorithm follows this sequence:

1. Elevator Call Handling

- 📞 When a call is initiated from a floor, the ElevatorScheduler evaluates all elevators to determine the best candidate to handle the request.

- 📊 Selection criteria include proximity to the calling floor, current movement direction, and elevator status (idle or active).

2. Elevator Movement

- 🗂 Each elevator maintains an ordered list of destination floors in a queue.

- 🔁 The scheduler iteratively updates elevator positions, moving them one floor at a time toward their next target.

- 🛑 When an elevator arrives at a target floor, it changes state (e.g., doors opening), removes the floor from the queue, and proceeds to the next destination if available.

3. Wait Time Estimation

- ⏳ The system dynamically calculates the estimated wait time for each floor, factoring in the selected elevator's current position, direction of travel, and the number of stops it must make before reaching the caller.

4. Dynamic Updates

- 🧱 The BuildingFactory supports live modifications to the building's configuration.

- ➕ Additional floors or elevators can be added on the fly, ensuring the simulation continues running without requiring a reset.