**Elevator Project Overview**

The project features an elevator control system implemented in TypeScript, utilizing a modular and object-oriented architecture. The architecture comprises several key components, each serving a specific role in managing the elevator system.

**1. Elevator Class:**

- The `Elevator` class represents individual elevator units within the system.

- Each elevator object encapsulates properties such as `id`, `currentFloor`, `destinations`, `inMotion`, and methods for elevator movement control.

- Elevators are responsible for handling movement requests, managing destinations, and animating their movement between floors.

**2. ElevatorController Class:**

- The `ElevatorController` class acts as the central controller for managing multiple elevators.

- It initializes and maintains an array of elevator instances and orchestrates their operations.

- The controller dispatches elevator requests, assigns the closest available elevator to handle the request, and coordinates elevator movements.

**3. BuildingElementFactory:**

- The `BuildingElementFactory` class facilitates the creation of building elements such as floors and buttons.

- It provides a factory method to create instances of building elements, which are then used to construct elevators.

**4. ElevatorFactory:**

- The `ElevatorFactory` class is responsible for creating instances of elevators.

- It utilizes the `BuildingElementFactory` to create building element instances and initializes elevator objects with the necessary parameters.

**Main Algorithm:**

The primary algorithm implemented in the project revolves around efficiently dispatching elevators to handle floor requests. Here's an overview of the main algorithm developed for the project:

**1. Call Elevator Functionality:**

- When a floor button is pressed, the system determines the closest available elevator to handle the request.

- The `ElevatorController` class calculates the distance between each elevator's current position and the target floor.

- The elevator with the shortest distance to the target floor is assigned to handle the request.

**2. Elevator Movement and Destination Management:**

- Upon receiving a floor request, the assigned elevator adds the target floor to its destination queue.

- The elevator processes its destination queue and moves to the next floor in the queue if not already in motion.

- Elevators handle animations, sound effects, and floor updates as they move between floors.

- Once an elevator reaches its destination floor, it notifies the system and remains idle until the next request.

**Unimplemented Requirements:**

**1.** Changing the background color of the button from the moment the elevator is summoned until its arrival.

**2.** Flexible encoding of elements and the structure of `index.html` through user input.

**Issues in the Code:**

**Algorithm:**

The algorithm currently implemented is not efficient enough, because it calculates the nearest elevator in terms of physical distance. A more efficient algorithm would compute the closest elevator in terms of time distance. At this stage, it has not been implemented.

**Code Structure:**

The code structure is not sufficiently modular and does not fully adhere to the Single Responsibility Principle.