Project Structure Document

Project Overview

This project manages the elevator system of a building, handling the interactions between elevators, floors, and various timing mechanisms. The project is structured to modularize .different components for clarity and maintainability

File Descriptions

building class

Purpose: Manages the building's overall structure and coordinates interactions .between floors and elevators

:**Key Components**

`Building` class: Initializes the building with a certain number of floors and .elevators

.Methods to add floors and elevators, and to start the elevator system

elevator class

Purpose: Handles the operations of an elevator, including moving between floors, .stopping, and responding to user requests

:**Key Components**

.`Elevator` class: Manages the state and behavior of an elevator

.Methods to move the elevator, handle requests, and manage doors

floor class

Purpose: Represents a floor in the building and includes functionalities for handling .floor-specific events and interactions with the elevator

:**Key Components**

.`Floor` class: Initializes a floor with specific attributes

.Methods to request an elevator and interact with it

main_elevator.py

Purpose: Entry point for the elevator system, initializing and starting the elevator .operations

:**Key Components**

.Main execution script to initialize the building, floors, and elevators

.Starts the elevator system and handles user inputs or automated scenarios

settings.py

Purpose: Contains configuration settings for the building and elevator system, such .as the number of floors, timing intervals, and other adjustable parameters

:**Key Components**

Configuration variables: Settings like the number of floors, number of .elevators, and operational parameters

stopwatch class

Purpose: Provides timing functionalities to track the duration of various operations .within the system

:**Key Components**

.`Stopwatch` class: Methods to start, stop, and measure elapsed time

timer class

.**Purpose**: Offers timing capabilities, such as countdowns or delays

:**Key Components**

`Timer` class: Methods to start a timer, check remaining time, and handle .timeouts

Directory Structure

python
Copy code

```
project_root/
|
building class          # Main logic for the building structure --|
elevator class          # Elevator operations and user request --|
handling
floor class             # Floor representation and events --|
handling
main_elevator.py        # Entry point for the elevator system --|
settings.py             # Configuration settings for the system --|
stopwatch class         # Stopwatch functionalities for tracking --|
durations
```

```
timer class             # Timer functionalities for countdowns --`
and delays
```

How to Use

.**Setup**: Ensure all necessary dependencies are installed

.**Configuration**: Adjust settings in `settings.py` as needed for your specific use case

.**Execution**: Run `main_elevator.py` to start the elevator system