Scala Engineer - Coding Challenge

Please complete the following task within a week. Ideally, you should not spend more than 4-5 hours on the task. It's better to present a simple, but well-tested and documented solution, than an over-sized and only half working one.

Task:

Design and implement a control system for multiple elevators (say up to 16) in Scala 2. For the sake of simplicity, your system should be implemented as a manually time-stepped simulation. Building a simulation that progresses in real-time is out of scope for this task.

Your system should provide the capabilities to:

1. query the state of the elevators (what floor are they on and where are they going)
2. request to pick up a passenger on a given floor
3. request to drop off a passenger on a given floor
4. time-step the simulation
5. set the initial state of the simulation (on which floor the elevators start and their initial target floors)

For example, we could imagine an interface like this after the initial setup phase:

```
trait ElevatorControlSystem {
 def status(): Seq[(Int, Int, Int)]
 def pickUp(Int, Int)
 def dropOff(Int, Int)
 def step()
}
```

Here we have chosen to represent the status of each elevator as three integers: Elevator ID, Floor Number, Goal Floor Number

A pickup request is two integers: Pickup Floor, Direction (negative for down, positive for up). The drop off request is also modeled as two integers: elevator ID, target floor.

This is not a particularly well designed interface, and it also leaves some open conceptual questions. For example, the elevator status has only one goal floor; but an elevator could have multiple goal floors enqueued. Please feel encouraged to improve this simple design!

The most important part of this challenge is the scheduling problem: Which elevator should serve a pickup request and how should it be enqueued in the forthcoming stops of that elevator? The simplest implementation would be to serve requests in FCFS (first-come, first-served) order. This is clearly bad — imagine waiting for such an elevator! Please discuss how your algorithm improves on FCFS and explain the assumptions you made in your write-up.

Please also provide build and run instructions.

Submission:

Submit your solution as a .zip archive containing the code and documentation.

Good luck!