Technical Assessment

Signal Decoder — Backend Software Engineer

Background

A communications tower receives signals from nearby devices. Each device has a unique signal pattern — a fixed sequence of numbers representing signal strength at each time slot.

When multiple devices transmit at the same time, the tower receives a single combined signal. The combined signal is the sum of each transmitting device's pattern, position by position.

Example:

```
Device A pattern:  [3, 1, 4]
Device B pattern:  [1, 5, 2]

If both transmit, the tower receives:
[3+1, 1+5, 4+2] = [4, 6, 6]
```

The tower knows all registered device patterns. It needs to figure out which devices were transmitting based on the combined signal it received.

Provided API

The following endpoints are already built and available for you to use. You can call these to generate test data and experiment with the problem before building your solution.

Endpoint 1 — Generate Devices

`GET /api/devices/generate?count={n}&signalLength={len}`

Generates a set of random registered devices with signal patterns.

```
// GET /api/devices/generate?count=5&signalLength=4

{
  "devices": {
    "D01": [3, 7, 2, 5],
    "D02": [8, 1, 4, 0],
    "D03": [1, 3, 6, 2],
    "D04": [5, 0, 1, 8],
    "D05": [2, 4, 3, 1]
  }
}
```

Endpoint 2 — Simulate Transmission

`POST /api/signal/simulate`

Takes a set of devices and picks a random subset to transmit. Returns the combined signal but NOT which devices transmitted.

```
// Request
{
  "devices": {
    "D01": [3, 7, 2, 5],
    "D02": [8, 1, 4, 0],
    "D03": [1, 3, 6, 2],
    "D04": [5, 0, 1, 8],
    "D05": [2, 4, 3, 1]
  }
}

// Response
{
  "receivedSignal": [4, 10, 8, 7],
  "activeDeviceCount": 2,
  "signalLength": 4,
  "totalDevices": 5
}
```

The response tells you the combined signal and how many devices transmitted, but not which ones. That is your job.

Your Task — Build the Decoder

Create a REST API endpoint that accepts a set of registered devices, a received signal, and a tolerance value. Your endpoint must determine which combination of devices produced the received signal.

`POST /api/signal/decode`

```
// Request
{
  "devices": {
    "D01": [3, 7, 2, 5],
    "D02": [8, 1, 4, 0],
    "D03": [1, 3, 6, 2],
    "D04": [5, 0, 1, 8],
    "D05": [2, 4, 3, 1]
  },
  "receivedSignal": [4, 10, 8, 7],
  "tolerance": 0
}
```

```
// Response
{
  "transmittingDevices": ["D01", "D03"],
  "decodedSignals": {
    "D01": [3, 7, 2, 5],
    "D03": [1, 3, 6, 2]
  },
  "computedSum": [4, 10, 8, 7],
  "matchesReceived": true,
  "solveTimeMs": 2
}
```

Requirements

1. When tolerance is 0, the computed sum must match the received signal exactly at every position.

2. When tolerance is greater than 0, each position can differ by up to that amount. For example, tolerance 1 means |computed – received| ≤ 1 at every position.

3. If no valid combination exists, return an appropriate error.

4. If multiple valid combinations exist, return all of them.

5. Signal values are always non-negative integers.

6. Include the solve time in your response — we will test with larger inputs.

Understanding Tolerance

Real-world signals have noise. The tower's reading may not be perfectly accurate. The tolerance parameter controls how much error is acceptable.

```
Tolerance = 0 (exact match):
  Device A + B sum:  [4, 6, 6]
  Received signal:   [4, 6, 6]
  Match? Yes — every position is exact.
```

```
Tolerance = 1 (fuzzy match):
  Device A + B sum:  [4, 6, 6]
  Received signal:   [4, 7, 6]
  Position 0: |4 - 4| = 0  <= 1  OK
  Position 1: |6 - 7| = 1  <= 1  OK
  Position 2: |6 - 6| = 0  <= 1  OK
  Match? Yes — every position is within tolerance.
```

Test Scenarios

Your decoder must handle the following scenarios. We will test with these and others.

Scenario	Devices	Signal Length	Tolerance	What It Tests
Simple	5	4	0	Basic correctness
Medium	15	5	0	Performance under load
Hard	25	6	0	Algorithm efficiency
Noisy	10	4	1	Fuzzy matching
No match	5	3	0	Error handling
Multiple	5	2	0	Returning all solutions

Technical Guidelines

Language & Framework: ASP.NET Core 8+ (C#)

Architecture: Clean architecture — Controllers, Services, Domain, Tests

Dependency Injection: Use DI for services where appropriate

Validation: Input validation with meaningful error messages

Testing: Unit tests using xUnit or NUnit covering core logic and edge cases

Documentation: Swagger / OpenAPI for endpoint documentation

README: Setup, build, and run instructions

Evaluation Criteria

Area	What We're Looking For
Algorithm	Does it handle 25+ devices without timing out?
Architecture	Clean separation of concerns, testable components
Error Handling	Invalid inputs, no solution found, mismatched signal lengths
Testing	Edge cases covered, not just the happy path
Performance	Does the candidate measure and discuss solve time?
Documentation	Clear README, code comments explaining approach and trade-offs

Deliverables

Item	Description
Source Code Repository	Complete Git repository with commit history
README.md	Setup, build, and run instructions
API Implementation	Functional decoder endpoint with validation and error handling
Tests	Unit tests for core solver logic and edge cases
Optional: Deployment	Public endpoint if deployed to a cloud host

Questions Welcome

You are encouraged to ask for clarification at any point. The goal is not only to test your coding skills but also your approach to problem-solving and communication.

You may use AI tools or assistants, as long as you fully understand and can explain every part of your solution during the follow-up discussion.

Good luck, and happy coding!