Integration Engineer Task (1 Week)

GENERAL IDEA

We receive events from external platforms in many different JSON formats.
We need integration engineers who can take those events, understand them,
validate them, transform them, communicate clearly with external partners,
and make sure the data enters our system correctly.

Your task is to build a simplified version of our ingestion system.

External systems will send event payloads in their own JSON structure.
Your system must:
- validate these events
- map them into your own internal data structure
- publish them

Deliverable: small .NET Web API project.
Your choices (design, architecture, data model, DB schema) are up to you.

Assumption: all external JSON payloads will be flat (no nested JSON).

REQUIRED FIELDS & MAPPING (IMPORTANT)

All external events must contain at least 3 required fields.
One of these 3 must clearly represent an "actor" (for example: user/player/account id).
You decide the exact field names and required types.

Your system must support DEFAULT mapping rules.
If no mapping override exists → system should still work with default logic.

Your system must support DYNAMIC mapping:
- ANY external field can be mapped into ANY internal field name
- mapping rules can be modified by the API without redeploy

REQUIRED FUNCTIONALITY
----

1) ACCEPT RAW JSON EVENTS

Create an endpoint to receive ANY external event JSON.
Your system must validate it, map it using current mapping rules, and publish it.

2) RUNTIME MAPPING UPDATES

Create API endpoints to:

- ADD or MODIFY mapping rules
- REMOVE mapping rules
- LIST current mapping rules

Mapping rules will rename external fields to internal fields.

EXAMPLE (to give you idea)

External platform sends this:

```
{
"usr": "player_123",
"amt": "25.50",
"curr": "GEL",
"ts": "2025-11-05T12:33:47Z"
}
```

We create mapping rules like:

usr -> PlayerId

amt -> Amount

curr -> Currency

ts -> OccurredAt

Then our internal event might look like:

```
{
"PlayerId": "player_123",
"Amount": 25.50,
"Currency": "GEL",
"OccurredAt": "2025-11-05T12:33:47Z"
}
```

This is just an example.

You decide required fields + final internal model.

SIMULATE FAILURES

Publishing must sometimes randomly fail (simulate it).

System must handle these failures gracefully.

SIMULATION ENDPOINTS

Create 2 endpoints:

- one endpoint that will publish ONE event (you choose the request model)
- another endpoint to generate and publish 100 random external events

----
DOCUMENTATION (VERY IMPORTANT)

As an Integration Engineer — documentation quality matters.

Provide a README explaining:
- how to run the project
- description of your required fields and why you chose them
- explanation of default mapping logic
- sample payloads (good and bad)
- short instructions in simple language that could be sent to external clients
(tell them how they should send events into your system)

SUBMISSION

- GitHub repo link OR zip
- Timebox: 1 week (expectation: ~2 hours/day)

Focus on:
- clean separation of concerns
- readable code
- good handling of bad input
- ability to explain and justify decisions
- strong developer communication in documentation