🧠 Zeew+ AI-Powered Adaptive Dashboard – Technical Challenge

Welcome!
At Zeew+, we're building a system that can connect to any franchise's database, understand its structure, and generate a fully adaptive dashboard with tailored modules and AI insights.

This challenge is designed to evaluate how well you can think modularly, build scalable architecture, and simulate intelligence in system behavior.

🎯 Objective

Build a mini version of Zeew+ that can:

1. Ingest an unknown JSON schema simulating a franchise's data.
2. Analyze the schema and detect relevant modules (e.g. orders, inventory, events).
3. Dynamically generate a custom dashboard showing only modules that are detected.
4. Simulate basic AI insights or suggestions based on the data.

🗂️ Input Data (Schema)

You'll be given (or can define) a sample JSON file that simulates a franchise's data:

```
{
  "orders": [...],
  "inventory": [...],
  "staff": [...],
  "reviews": [...],
  "shifts": [...],
  "events": [...]
}
```

This structure may vary. The system must **not assume** fixed collections or fields.

✅ Requirements

1. Schema Analysis (Backend)

- Parse the input schema.
- Determine which modules exist based on collection names and fields.

Output a list of detected modules like:

```
 [
  { "module": "Orders", "fields": ["order_id", "status", "total"] },
  { "module": "Inventory", "fields": ["item", "stock"] }
]
```

-

2. Dynamic Dashboard (Frontend)

- Generate dashboard cards/modules for each detected module.
- Show simple UI for each (e.g., orders chart, inventory alerts).
- No hardcoding — frontend must adapt to the backend's module plan.

3. Simulated AI Suggestions

- Add at least 2 logic-based insights. Examples:
    - "Store A inventory is running low on 5+ items."
    - "Average customer rating dropped below 3.5 in the last 7 days."

🔧 Tech Stack

- Backend: Node.js / Laravel / Python (your choice)
- Frontend: React / Vue / Next.js (your choice)

- Optional: Simple Express or Laravel API for serving JSON

Use what you're comfortable with, but you should explain your architectural choices.

🌟 Bonus (Optional but Valuable)

- Allow re-upload of a different JSON schema to regenerate the dashboard.
- Add a “confidence score” per module based on field completeness.
- Use modular UI components for each module type (e.g. OrderModule, ReviewModule).

📦 Deliverables

- GitHub repo with source code
- Loom or short video walking through:
  - The dashboard behavior
  - How your module detection logic works
  - Your logic and architecture decisions
  - How you would scale this into a full Zeew+ system

- **Submit your answers until 14th May 2025 by email to mohamed@zeew.eu.**