ERP Incident Triage Portal – Take-Home Assignment

Background

Our team is building an AI-assisted platform to help enterprise customers triage and resolve ERP incidents faster. These incidents typically originate from Oracle ERP environments and contain unstructured descriptions of operational problems (e.g., invoices stuck, integrations failing, access issues, etc.).

This assignment asks you to build a small but realistic MVP of an **ERP Incident Triage Portal**. The goal is not to build a fully polished product, but to demonstrate how you think about system design, trade-offs, and execution across frontend, backend, and cloud infrastructure.

This exercise is intentionally open-ended and contains some ambiguity. We expect you to make reasonable assumptions and ask clarifying questions where needed.

Scope & Time Expectations

- Expected effort: ~1 week of part-time work
- Focus on clarity, correctness, and engineering judgment
- Over-engineering is discouraged

Problem Statement

Build a web-based system that allows users to submit ERP incidents, automatically enrich those incidents, and view them through a simple UI.

The system should consist of:

- A frontend web application
- A backend API
- Integration with at least one AWS service (free tier only)

Functional Requirements

1. Incident Submission

Users should be able to submit a new ERP incident with the following fields:

- Title
- Description (free-text)
- ERP Module (e.g., AP, AR, GL, Inventory, HR, Payroll)
- Environment (Prod / Test)

- Business Unit

2. Incident Enrichment

When an incident is submitted, the backend should automatically enrich it with derived metadata.

At minimum, include:

Severity

Assign a severity level (e.g., P1 / P2 / P3) based on the incident data.

- The logic is up to you (rules, heuristics, or light AI are all acceptable).

- Document your approach and assumptions.

Category

Classify the incident into one of several categories, such as:

- Configuration Issue

- Data Issue

- Integration Failure

- Security / Access

- Unknown

Optional (nice to have):

- Auto-generated short summary

- Suggested next step or action

3. Incident Viewing

The frontend should allow users to:

- View a list of submitted incidents

- Click into an incident to view its details, including derived fields

- See status and timestamps

Optional enhancements:

- Filtering by severity or module

- Status indicators or badges

4. Backend API

Expose a small REST API that supports:

- Creating incidents
- Listing incidents
- Retrieving a single incident
- Updating incident status

Language and framework are up to you.

5. AWS Integration

You must integrate with at least one AWS service using the free tier.

Examples include (not exhaustive):

- DynamoDB for persistence
- S3 for storing incident payloads or logs
- Lambda for backend execution
- API Gateway
- CloudWatch for logging

Choose what makes sense for your design and explain your reasoning.

Technical Constraints

- Use AWS free tier only
- The system must be runnable locally
- The frontend and backend must be cleanly separated
- Authentication is optional unless you believe it is required for your design

Deliverables

Please provide:

1. A GitHub repository containing your solution
2. A README that includes:
   - High-level architecture diagram
   - Technology choices and rationale
   - Instructions to run the project locally

- Any assumptions made
- What you would improve or extend with more time

3. A deployed URL (optional, but a plus)

What We Are Evaluating

We are primarily looking at:

- System design and structure
- Code quality and readability
- API design and data modeling
- Practical use of AWS services
- Ability to reason about ambiguity and trade-offs
- Communication through documentation

Notes

- You are encouraged to ask clarifying questions before or during implementation.
- There is no single “correct” solution.
- Simplicity with clear reasoning is preferred over completeness.

We look forward to reviewing your approach and design decisions. Please feel free to create an architectural design to present this in the last round of interview.