Backend Engineer Assignment

Problem Statement

We want to design a simplified version of a **Communication Aggregator System** — a backend that receives messages from multiple sources and routes them intelligently to the right communication channel (Email, SMS, WhatsApp, etc.).

You'll design and implement a **3-microservice system** that performs this routing flow end-to-end.

PS - You may choose any tech stack – **NodeJS GraphQL with async microservices communication (RabbitMQ, Redis Streams, Kafka) would be plus.**

System Overview

You will build **multiple independent microservices** that communicate with each other:

---

1. Task Router Service

- Accepts incoming message requests via REST /GraphQL API.
- Validates the payload.
- Applies routing logic based on the `channel` field.
- Logging
- Decides which delivery service to forward the message to.
- Example logic:
    - `"email"` → Email Service
    - `"sms"` → SMS Service
    - `"whatsapp"` → WhatsApp Service
- Should handle retry if delivery fails (simulate retry logic).
- Duplicate body handling/ should not send duplicate message

---

2. Delivery Service

- Simulates sending messages for **Email**, **SMS**, and **WhatsApp**.
- You don't need to integrate real APIs — just log or store the message in a local DB

---

3. Logging service

- Produce logs for observability POV from service1 and service2 with complete journey of API with trace and subtraces.
- Save these logs in **Elasticsearch**, such that it can easily use it as on Kibana for troubleshooting.

Requirements

- Each service should run independently (separate ports or containers).
- Include a **README** explaining:
    - Architecture overview
    - Communication method and reasoning
    - How to start each service
    - **Postman Collection**.

Deliverables

1. **High-Level Design (HLD)**

    - Architectural Diagram
    - Show data flow between all 3 services
    - Mention chosen communication pattern and justification

2. **Working Prototype**

    - Source code of all 3 services
    - Basic setup/run instructions
    - Functional flow visible in logs and simple DB table

3. **Short README**

    - Overview of services and their purpose
    - Setup + run steps
    - Example payloads and expected output

In the tech interview:

- You'll walk us through your **HLD** and reasoning.
- Show a **live demo, and we will also test via ngrok** (local services interacting end-to-end).
- Discuss trade-offs and design decisions.