SelfActualize.ai Take-Home Assignment Instructions

Founding Backend Engineer (10/30)

Objective:
This assignment is designed to assess your technical skills and problem-solving ability by creating a basic but functional backend service for a communication router. The goal is to see how you approach backend design, API creation, and deployment setup.

Time Management:
Please aim to spend no more than **8 hours** on this assignment. You're welcome to use any resources at your disposal, such as documentation, tutorials, ChatGPT, Google, and more. The goal is to see what you can build within a limited timeframe while highlighting your strengths.

Task

Build a basic **communication microservice** that allows sending messages through multiple channels. The service should have one endpoint to accept message details and log each message's delivery status in a database.

Features:

1. **Message Sending Endpoint**:
    - Implement a single API endpoint `/sendMessage`.
    - The endpoint should accept a JSON payload with:
        - `type` (e.g., "email" or "SMS"),
        - `recipient` (e.g., phone number or email),
        - `content` (message text).
    - Use a mock function to simulate sending a message for one type (choose either email or SMS). This mock can simply return a successful response.
2. **Message Logging**:
    - Store each message's delivery status and timestamp in a **PostgreSQL** (or SQLite) database.
3. **CI/CD Pipeline Setup**:
    - Set up a basic CI/CD pipeline (using **GitHub Actions** or a similar tool) that:
        - Runs tests on the microservice.
        - Builds and deploys the service to a local Docker container, simulating production.

Requirements

- **Backend**:
    - Use **Python** or **Node.js** for backend development.
    - Implement the microservice with a lightweight framework, such as **Flask** for Python or **Express** for Node.js.
    - Ensure the database interactions for message logging are efficient.
- **CI/CD Pipeline**:
    - Configure a CI/CD pipeline that runs automated tests and builds and deploys the service to a local Docker environment.
    - Include at least one test for the `/sendMessage` endpoint to verify payload validation and response handling.

Bonus Points

1. **Expanded Channel Support**:
   Add flexibility to the microservice, allowing it to support future channels (like Slack or WhatsApp) by adjusting the API or routing structure.
2. **Deployment**:
   Deploy the service to a cloud service such as AWS.
3. **Documentation**:
   Include comments or a README describing the structure and key parts of the codebase.

Focus on Your Strengths

We encourage you to showcase your strongest skills. Whether it's writing clean, efficient code, setting up a robust CI/CD pipeline, or structuring the database effectively, let your strengths shine in this assignment.

Submission

- Please submit your code via a **GitHub repository link**.
- Ensure the repository is public or accessible to us.

Deadline

- Please submit your completed assignment within **48 hours** of receiving this email.

Evaluation Criteria

- **API Design & Code Quality**: Structure, clarity, and adherence to best practices.
- **Functionality**: Correctness and efficiency of core features, including message sending and logging.
- **CI/CD Implementation**: Ability to automate tests and deploy the service.
- **Documentation**: Completeness and clarity of instructions and comments.
- **Bonus**: Expanded features, cloud deployment, and documentation quality.

If you have any questions, feel free to reach out. We're excited to see how you approach this assignment and leverage your strengths!