### **Assignment: Data Transformation and API Integration**

#### **Objective**

Develop a backend application that integrates with two different APIs to fetch job offer data in distinct structures, transforms the data into a unified structure, stores it in a database, and provides an API to retrieve and filter the transformed data. The goal is to evaluate your skills in data transformation, API design, database structuring, and handling common development challenges such as error handling, duplicate management, and scalability.

### **Requirements**

#### **1. Data Transformation**

* Fetch data from **two APIs** (provided in the assignment).
  + API 1: Returns job offer data in [Structure A](https://assignment.devotel.io/api/provider1/jobs).
  + API 2: Returns job offer data in [Structure B](https://assignment.devotel.io/api/provider2/jobs).
* Transform the data from both APIs into a **unified structure**.
* Ensure the transformation process is designed to:
  + Be **scalable** and **maintainable** to handle additional APIs with minimal code changes.
  + Prevent duplication of data during transformation or storage.
  + Demonstrate your ability to write clean, reusable, and modular code.

#### **2. Scheduled Data Fetching**

* Use a **cron job** or an equivalent scheduler to periodically fetch data from the APIs.
  + Bonus: Configurable scheduling (e.g., frequency defined in a configuration file or environment variable).

#### **3. Database Storage**

* Store the transformed data in a **relational database** of your choice (e.g., PostgreSQL, MySQL).
* Ensure the database schema is well-structured and optimized for queries.
* Avoid storing duplicate data by implementing proper checks or logic.

#### **4. API Development**

* Develop an API endpoint to retrieve the transformed data from the database.
  + Endpoint: /api/job-offers
  + Features:
    - Support basic filters (e.g., filter by job title, location, or salary range).
    - Paginate results.
    - Include error handling for invalid queries or server errors.
* Implement the backend using **TypeScript**.
* **Bonus**: Use **NestJS** to build the backend.

#### **5. Error Handling**

* Implement robust error handling throughout the application:
  + Handle API call failures (e.g., retries, logging errors).
  + Gracefully handle database errors.
  + Provide clear error messages in the API responses.

#### **6. Testing**

* Include test cases for:
  + Data transformation logic.
  + Scheduler (cron job) functionality.
  + API endpoint functionality.
  + **Bonus**: Write integration tests for end-to-end functionality.

### **Expected Deliverables**

1. **Codebase**:
   * A well-organized and readable codebase.
   * Clear instructions on how to run the application.
2. **Database Schema**:
   * A normalized database structure supporting unified job offer data.
3. **API Documentation**:
   * Document the /api/job-offers endpoint with:
     + Supported query parameters for filtering.
     + Example requests and responses.
4. **Test Cases**:
   * Unit tests for key functionalities.
   * Bonus: Integration tests for end-to-end workflows.
5. **Error Logs**:
   * Provide a mechanism to log errors (e.g., API failures, database issues).

### **Evaluation Criteria**

1. **Code Quality**:
   * Clean, modular, and maintainable code.
   * Adherence to TypeScript best practices.
2. **Scalability**:
   * Ease of adding new APIs with different structures.
3. **API Design**:
   * Usability of the /api/job-offers endpoint (e.g., filter options, pagination).
4. **Database Design**:
   * Effective and normalized schema design.
   * Ability to handle duplicate records.
5. **Error Handling**:
   * Robust handling of edge cases and clear error messages.
6. **Testing**:
   * Comprehensive test coverage.
7. **Bonus Points**:
   * Usage of **NestJS**.
   * Scheduled data fetching via a cron job.
   * Comprehensive API documentation.
   * Handling and avoiding duplicate data.

### **Setup**

1. APIs  
   1- <https://assignment.devotel.io/api/provider1/jobs> Structure A.

2- <https://assignment.devotel.io/api/provider2/jobs> Structure B.

1. Submit your solution as a public Git repository.

#### **Timeline**

You have **5 days** to complete the assignment. If you need any clarification, please reach out during the first 48 hours.