Laravel Developer Tasks: API Integration and Design Patterns

These two tasks are designed to assess the Laravel developer's abilities in third-party API integration, using design patterns, and effectively working with Eloquent relationships in Laravel models.

# Task 1: Third-Party API Integration and Use of Design Patterns in Laravel Application

Objective: Create a Laravel application that integrates with a third-party API (such as a payment gateway, weather service, or social media API) and employs Repository and Service Layer patterns.  
  
Details:  
- Build a new Laravel application that interacts with a third-party API, retrieves data, and stores it in the local database.  
- Use the Repository Pattern and Service Layer Pattern to manage API requests and responses, ensuring clean and maintainable code.  
- Handle API failures with proper logging and retry mechanisms.  
- Display the retrieved data on the front end using Vue.js or Blade templates.  
  
Required Features:  
- Third-party API integration (retrieving data).  
- Use of Repository and Service Design Patterns.  
- Displaying data on the frontend interface.  
- Error handling and logging.  
  
Challenge:  
This task will assess the developer’s capability to implement design patterns in Laravel while integrating with a third-party API.

# Task 2: Complex Eloquent Relationships and API Integration in Laravel Application

Objective: Create a Laravel application that utilizes complex Eloquent relationships and integrates with a third-party API.  
  
Details:  
- Develop a system involving users, orders, and products with relationships such as:  
 - `One-to-Many` (e.g., a user can have many orders),  
 - `Many-to-Many` (e.g., a product can be associated with many orders).  
- Sync user orders and product data with an external API (such as an inventory or e-commerce platform).  
- Use proper Eloquent relationships to easily fetch order history and product details for a user.  
- Use the Laravel Factory Pattern to generate mock data for users, orders, and products.  
- Implement a background process using Job Queues to sync orders with the third-party API.  
  
Required Features:  
- Complex Eloquent relationships (One-to-Many, Many-to-Many).  
- Syncing order and product data with a third-party API.  
- Using the Factory Pattern to generate sample data.  
- Implementing a background job queue.  
  
Challenge:  
This task will evaluate the developer’s skill in managing relationships in Laravel and syncing data with a third-party API through a background process.