Backend Developer Coding Challenge

Objective:
The goal of this task is to evaluate your ability to handle common Laravel development challenges, including performance optimization, handling large datasets, implementing pagination, request validation, writing tests, and API documentation.

Task Description

You are tasked with creating a paginated API endpoint to list products with their vendors and ratings. The endpoint should support filtering and include comprehensive testing and documentation.

Requirements

1. Database Seeding

Write a database seeder to generate a large dataset for testing purposes:

- **Vendors:** Create 1000 vendors.
- **Products:** Each vendor should have between 100–500 products.
- **Ratings:** Each product should have between 1–50 ratings.

2. API Endpoint

Create a paginated API endpoint at `/api/products` that fulfills the following requirements:

1. **Pagination:** Return results in a paginated format, with 15 items per page by default.
2. **Filtering:** Support optional query parameters:
    - `vendor_id` (integer): Filter products by the vendor ID.
    - `name` (string): Search products by a name substring.
3. **Eager Loading:** Optimize database queries by using eager loading to prevent the N+1 problem.
4. **Request Validation:** Use a Laravel `Request` class to validate the query parameters.
5. **Response Format:** Return data in the JSON format like this (you can change this according to data structure you design):

```
{

    "data": [
        {
            "name": "Sample Product",
            "vendor": "Kiza Ltd",
            "ratings": [
                { "name": "Jane Smith", "text": "Great
product!", “rating”: 5 }
            ]
        }
    ],
    "meta": {
        …
    }
}
```

3. Tests

Write tests using **PEST** to verify the following:

1. **Pagination:** Ensure the API returns correct pagination metadata and limits results per page.
2. **Filtering:** Validate that filtering by `vendor_id` and `name` works as expected.
3. **Validation:** Test that the API returns appropriate error responses for invalid parameters.

4. API Documentation

Provide clear and concise documentation for the `/api/products` endpoint, including:

1. **Endpoint Details:** Method (`GET`), URL, and description.
2. **Query Parameters:**
    - `vendor_id` (optional, integer): Filter products by vendor ID.
    - `name` (optional, string): Search products by name.
    - `page` (optional, integer): Specify the page number (default: 1).
    - `per_page` (optional, integer): Number of items per page (default: 15).
3. **Response Format:** Include an example response with explanations of the fields.

Deliverables

Fully runnable and testable Laravel application including:

1. Laravel models and migrations
2. Laravel seeder to generate the dataset.

3. `/api/products` API endpoint code.
4. Tests written in PEST.
5. Swagger OpenAPI documentation.

Evaluation Criteria

- **Code Quality:** Clean, maintainable, and readable code adhering to Laravel best practices.
- **Performance:** Efficient database queries using eager loading to handle the N+1 problem.
- **Validation:** Proper use of Laravel's validation for incoming requests.
- **Testing:** Comprehensive test coverage with clear assertions.
- **Documentation:** Clear and professional API documentation.

Submission

- Submit your code in a GIT repository (cloud or compressed).
- Include instructions for running the project and tests.

Good luck! We're excited to see your solution.