Take-Home Task Instructions

Objective: Develop a Laravel application that demonstrates advanced proficiency in modeling, API REST calls, and PHP.

Task Requirements:

1. **Create a Laravel Project:**
    - Set up a new Laravel project.
2. **Modeling:**
    - Create a Product model with the following attributes:
        - id: Auto-increment integer, primary key
        - name: String, required
        - description: Text, optional
        - price: Decimal, required
        - stock: Integer, required
    - Create a Category model with the following attributes:
        - id: Auto-increment integer, primary key
        - name: String, required
    - Establish a many-to-many relationship between Product and Category.
3. **API Endpoints:**
    - Decide and implement the necessary RESTful API endpoints for the Product and Category models to fulfill the following operations:
        - CRUD operations for Product
        - CRUD operations for Category
        - Assigning categories to products
        - Filtering products by category
4. **Validation:**
    - Implement proper validation for both models when creating and updating. Ensure all necessary fields are validated correctly.
5. **Advanced Features:**
    - Use return type declarations for all methods.
    - Implement proper variable scoping and data encapsulation.
    - Use resource classes for API responses to ensure consistent and clean output.
    - Implement pagination for the GET /api/products endpoint.
6. **Testing:**
    - Write unit tests and feature tests to ensure all API endpoints are working correctly.
    - Ensure the tests cover both successful operations and failure scenarios (e.g., validation errors, not found).
    - Include tests for the relationship between Product and Category.

Additional Requirements:

- **Code Quality:**
    - Follow Laravel best practices and coding standards.
    - Ensure the code is clean, well-organized, and properly commented.
    - Use type hinting and return type declarations where applicable.
    - Implement proper error handling and use custom exceptions where necessary.
- **Version Control:**
    - Push the code to a public GitHub repository.
    - Do not send the code as a zip file.

Submission:

1. Create a public repository on GitHub.
2. Push your code to the repository.
3. Share the repository link.

Tips:

- Pay attention to the structure and organization of your code.
- Ensure your API responses are consistent and properly formatted (e.g., using JSON).
- Demonstrate your ability to write effective and meaningful tests.
- Showcase your understanding of advanced PHP features like type declarations and proper scoping.

Deliverables:

- A link to the GitHub repository