Task: Build an Extendable Order and Payment Management API

Objective:

Develop a Laravel-based API for managing orders and payments, with a focus on clean code principles and extensibility. The system should allow adding new payment gateways with minimal effort.

Requirements:

Core API Features:

- Order Management:
- Create Order: Accept user details, purchased items (product name, quantity, price), and calculate the total.
- Update Order: Modify existing order details.
- Delete Order: Delete an order (only if no payments are associated).
- View Orders: Retrieve all orders or filter by status (pending, confirmed, cancelled).
- Payment Management:
- Process Payment: Simulate payment processing for an order. Payment fields: Payment ID, associated Order ID, payment status (pending, successful, failed), payment method (credit_card, paypal, etc.). Payment should trigger the appropriate gateway logic.
- Add New Payment Gateway: Implement the system using a strategy pattern or similar design to allow adding new payment gateways with minimal changes to the codebase.
- View Payments: Retrieve payment details for a specific order or all payments.
- Business Rules: Payments can only be processed for orders in the confirmed status. Orders cannot be deleted if they have associated payments.

API Design:

- Follow RESTful API principles.
- Use appropriate HTTP methods and status codes.
- Provide pagination for list endpoints.

Authentication:

- Secure the APIs using JWT authentication.
- Include endpoints for user registration and login.

Validation:

- Ensure all API inputs are validated.
- Provide meaningful error messages.

Extensibility:

- Use a design pattern (e.g., strategy pattern) to ensure adding a new payment gateway involves minimal code changes.

- Allow configuration of gateways (e.g., API keys, secrets) through .env or a database.

Documentation:

- Use Postman or similar tools to create API documentation:
- Include detailed examples of requests and responses (success and error cases).
- Organize endpoints by functionality (e.g., Orders, Payments, Authentication).
- Provide collection export for easy import.

Testing:

- Write unit and feature tests for the API, including tests for payment gateway logic.

Evaluation Criteria:

1. API Implementation: Proper design of endpoints (naming conventions, HTTP methods), adherence to RESTful principles, and secure handling of authentication and validation.
2. Clean Code: Follow PSR-12 standards, use of design patterns (e.g., strategy pattern for payment gateways), and modular and DRY code.
3. Extensibility: Ease of adding new payment gateways and documentation of how to add a new gateway.
4. Documentation: Completeness and clarity of the API documentation and quality of examples provided in the documentation.
5. Testing: Coverage and quality of test cases, especially for payment processing.

Instructions:

1. Delivery:

- Submit a Git repository link with:
- Laravel project code.
- Postman collection or API documentation file.
- README file containing:
- Setup instructions.
- Explanation of the payment gateway extensibility.
- Any additional notes or assumptions.

2. Timeline: Provide a maximum of 3–5 days for completion.