Technical Task
For Back End Developer

Section 1

Routing:

1. Create a Laravel route that directs to a controller method.
2. Implement a route parameter and retrieve its value in the controller.

Middleware:

1. Create a custom middleware that logs the incoming request.
2. Apply the middleware to a specific route.

Section 2:

Eloquent ORM:

1. Create a model for a "Product" with fields: name, price, and quantity.
2. Implement a query to fetch all products with a price greater than a specified amount.

Migrations and Seeders:

1. Create a migration to add a new column "category_id" to the "products" table.
2. Create a seeder to populate the "category_id" with random values.

Section 3:

Authentication:

1. Implement user authentication using Laravel's built-in authentication system.
2. Create a middleware to ensure only authenticated users can access certain routes.

Authorization:

Use Laravel's policy to define authorization rules for updating and deleting products.

Section 4:

API Routes:

1. Create a RESTful API endpoint to retrieve a list of products.
2. Implement pagination for the API response.

API Authentication:

1. Secure the API endpoint with token-based authentication.
2. Provide instructions on how to generate and use the API token.

Section 5:

Unit Testing:

1. Write a unit test for a controller method that adds a new product.
2. Use a test database for isolation.

Feature Testing:

1. Write a feature test for the authentication process.
2. Test both successful and unsuccessful login attempts.

Section 6:

Blade Templates:

1. Create a Blade template to display a list of products.
2. Include a form to add a new product.

AJAX Integration:

Implement AJAX functionality to add a new product without refreshing the page.

Section 5:

Unit Testing:

1. Write a unit test for a controller method that adds a new product.
2. Use a test database for isolation.

Feature Testing:

1. Write a feature test for the authentication process.
2. Test both successful and unsuccessful login attempts.

Section 6: Frontend Integration

Blade Templates:

1. Create a Blade template to display a list of products.
2. Include a form to add a new product.

AJAX Integration:

Implement AJAX functionality to add a new product without refreshing the page.

Section 7: Stripe Integration

Stripe Setup:

1. Sign up for a Stripe account if not already done.
2. Integrate the Stripe PHP SDK into the Laravel project.

Payment Form:

1. Create a Blade view with a form to collect payment details (card number, expiration date, and CVC).
2. Use Stripe Elements or Checkout for a secure payment form.

Controller Action:

1. Implement a controller action to handle the form submission.
2. Use the Stripe PHP SDK to create a charge for the provided payment details.

Payment Confirmation:

1. After a successful payment, display a confirmation message to the user.
2. Store the payment details and confirmation in the database.

Webhooks:

1. Implement a webhook to handle Stripe events (e.g., payment success or failure).
2. Update the order status in the database based on the webhook events.

Instructions:

1. You should create a new Laravel project for this test.
2. Provide a README file with instructions on how to set up and run the project.
3. You should follow Laravel conventions, and write clean, well-documented code.
4. You should obtain and use your own Stripe API keys (test mode) for this task.
5. You should Ensure follow security best practices, such as using environment variables for sensitive information.
6. Include proper error handling for failed payment attempts.
7. You should document the Stripe integration process in the README file.