SOFTWARE ENGINEERING, BACKEND

TECHNICAL CODING EXAM
SOFTWARE ENGINEERING, TECHNOLOGY AND ENGINEERING DEPARTMENT

GENERAL

This is a coding interview customized for different programming languages to assess candidates on general coding and language-specific criteria.

REQUIREMENTS

1. Backend
    a. Create an API endpoint for each of the following functionalities
        i. Registration
        ii. Login
        iii. Products
        iv. Orders
    b. Technical Requirement
        i. It must be database driven
        ii. It must be built in Laravel Framework
        iii. It should be test driven

Note: Create a github repository for these activities (make it separate repo and make an installation guide) and make it private, kindly add the following email address, a.zamora@integratrcorp.com, g.yabut@integratrcorp.com

1. Backend

1.1. Registration API

Features:

a. Guest user can register
b. As a guest user, in order to register I need an API endpoint to capture its information

Scenario: Successful Registration
Given the request body is
{
“email”: backend@integratrcorp.com,
“password”: “test123”
}
When I request “/register” using “POST” method
Then the response header http code is 201
And the response body is
{
“message”: “User successfully registered”
}

Scenario Outline: Unsuccessful registration due to email is already taken
Given the request body is
{
“email”: “johndoe@integratrcorp.com”,

"password: "test123"
}
When I request "/register" using "POST" method
Then the response header http code is 400
And the response body is
{
"message": "Email already taken"
}

Additional Requirements (plus points if it will be implemented)
1. Email sending should be asynchronous (using jobs and queues)
2. Unit testing for the 2 scenarios

1.2. Authentication/Login

Feature: Registered user can login, As a registered user I need an API endpoint to verify the credential **Scenario Outline**: Successful login
Given the request body is
{
"email": "johndoe@integratrcorp.com",
"password: "test123"
}
When I request "/login" using "POST" method
Then the response header http status code is 201
And the response body is
{
"access_token": <Json Web Token>
}

Scenario Outline: Unsuccessful login due to invalid credentials
Given the request body is
{
"email": "johndoe@integratrcorp.com",
"password: "test121231233"
}
When I request "/login" using "POST" method
Then the response header http status code is 401
And the response body is
{
"message": "Invalid credentials
}

Additional Requirements (plus points if it will be implemented)
1. Account locking for 5 minutes (after 5 failed attempts)
2. Unit Testing for the 2 scenarios

1.3. Orders

Feature: Order a specific product, as a registered user I need an endpoint to order a product

Scenario Outline: Successfully created an order

Given the request body is
{
“product_id”: “1”,
“quantity: “2”
}
And the request header is
{
Authorization: Bearer <access_token>
}
When I request “/orders” using “POST” method
Then the response header http status code is 201.
And the response body is
{
“message”: “You have successfully ordered this product.”
}

Scenario Outline: Unsuccessful order due to insufficient stock of a product
Given the request body is
{
“product_id”: “2”,
“quantity: “9999”
}
And the request header is
{
Authorization: Bearer <access_token>
}
When I request “/orders” using “POST”
Then the response header http status code is 400.
And the response body is
{
“message”: “Failed to order this product due to unavailability of the stock”
}

Scenario Outline: List all orders
Given the request header is
{
Authorization: Bearer <access_token>
}
When I request “/orders” using “GET” method
Then the response header http status code is 200.
And the response body is
[
{
“product_id”: “1”,
“quantity”: “24”,
“created_at”: “2021-07-26 09:18:00”
},
{
“product_id”: “2”,
“quantity”: “23”,
“created_at”: “2021-07-26 09:18:00”

```
        }
    ]
```

Scenario Outline: List a single order
Given the request header is

```
    {
        Authorization: Bearer <access_token>
    }
```

When I request "/orders/1" using "GET" method
Then the response header http status code is 200.
And the response body is

```
    {
        "product_id": "1",
        "quantity": "24"
    }
```

Scenario Outline: Edit a single order
Given the request header is

```
    {
        Authorization: Bearer <access_token>
    }
```

When I request "/orders/1/edit" using "GET" method
Then the response header http status code is 200.
And the response body is

```
    {
        "product_id": "1",
        "quantity": "24",
        "created_at": "2021-07-26 09:18:00"
    }
```

Scenario Outline: Update a single order
Given the request header is

```
    {
        Authorization: Bearer <access_token>
    }
```

When I request "/orders/1" using "PUT" method
Then the response header http status code is 201.
And the response body is

```
    {
        "product_id": "1",
        "quantity": "25"
    }
```

Scenario Outline: Unsuccessful update of order due to insufficient stock of a product
Given the request body is

```
    {
        "product_id": "2",
        "quantity: "9999"
    }
```

And the request header is
{
Authorization: Bearer <access_token>
}
When I request "/orders/1" using "PUT" method
Then the response header http status code is 400.
And the response body is
{
"message": "Failed to order this product due to unavailability of the stock"
}

Scenario Outline: Delete a single order
Given the request header is
{
Authorization: Bearer <access_token>
}
When I request "/orders/1" using "DELETE" method
Then the response header http status code is 204.
And the response body is
{
"message": "Order successfully deleted"
}

Additional Requirements
1. Create a seeder for "Products" with the following fields
a. Id
b. Name
c. Available Stock
2. Create "Orders" table with the following fields
a. product_id *Note: Kindly create a relationship to products table*
b. quantity
c. created_at
3. Once an order has been successfully placed, deduct the given quantity on the available stock on the specified product.
4. Unit Testing for 2 scenarios
5. Provide API endpoint for listing of products
Scenario Outline: List all Products
Given the request header is
{
Authorization: Bearer <access_token>
}
When I request "/products" using "GET" method
Then the response header http status code is 200.
And the response body is
[
{
"id": "1",
"name": "Product 1",
"available_stock": "24"
},
{
"id": "2",

```
            "name": "Product 2",
            "available_stock": "100"
        }
    ]
```

1.4. Dashboard API

Scenario Outline: List all Products by Top 5 Summary
Given the request header is

```
    {
        Authorization: Bearer <access_token>
    }
```

When I request "/reports/dashboard/summary" using "GET" method
Then the response header http status code is 200.
And the response body is

```
    [
        {
            "product_id": "1",
            "name": "Product 1",
            "total_quantity": "199"
        },
        {
            "product_id": "2",
            "name": "Product 2",
            "total_quantity": "188"
        },
        {
            "product_id": "3",
            "name": "Product 3",
            "total_quantity": "166"
        },
        {
            "product_id": "4",
            "name": "Product 4",
            "total_quantity": "120"
        },
        {
            "product_id": "5",
            "name": "Product 5",
            "total_quantity": "111"
        }

]
```

Scenario Outline: List all Products Total Quantity by Day Date Filter, it can be filtered by date range
Given the request body or params is

```
    {
        "from": "2021-07-26",
        "to": "2021-07-31"
```

}
And the request header is
{
Authorization: Bearer <access_token>
}
When I request "/reports/dashboard/date_filter" using "GET" method
Then the response header http status code is 200.
And the response body is

```
[
    {
        "product_id": "1",
        "name": "Product 1",
        "total_quantity": "199",
        "created_at": "2021-07-26"
    },
    {
        "product_id": "2",
        "name": "Product 2",
        "total_quantity": "188",
        "created_at": "2021-07-27"
    },
    {
        "product_id": "3",
        "name": "Product 3",
        "total_quantity": "166",
        "created_at": "2021-07-28"
    },
    {
        "product_id": "4",
        "name": "Product 4",
        "total_quantity": "120",
        "created_at": "2021-07-27"
    },
    {
        "product_id": "5",
        "name": "Product 5",
        "total_quantity": "111",
        "created_at": "2021-07-26"
    }

]
```

Additional Requirements

1. It should get the top 5 sold product from descending to ascending order
2. Create a custom column total_quantity
3. You must use aggregates to get the total quantity