# Detailed Test Cases for Order Management APIs

## User Management

### Test Case ID: TC001

Description: Verify user registration with valid data.

Preconditions: No user with the same email exists in the database.

Test Steps:

* - Prepare a payload with valid user details including name, email, phone, and password.
* - Call the /api/User/register endpoint with the prepared payload using a POST request.
* - Verify the HTTP status code is 200 (OK).
* - Query the database to ensure the user record is created with the correct details.

Expected Result: User is registered successfully and stored in the database with correct details.

### Test Case ID: TC002

Description: Verify login functionality with correct credentials.

Preconditions: A registered user exists in the database with valid email and password.

Test Steps:

* - Call the /api/User/Login endpoint with the correct email and password using a GET request.
* - Verify the HTTP status code is 200 (OK).
* - Validate the response contains a valid JWT token.
* - Decode the JWT token and confirm it includes the correct user details.

Expected Result: User is authenticated successfully, and a valid JWT token is returned.

## Product Management

### Test Case ID: TC003

Description: Verify adding a new product as an admin.

Preconditions: Admin user must be logged in.

Test Steps:

* - Prepare a payload with valid product details including name, description, price, and stock.
* - Call the /api/Product endpoint with the payload using a POST request and include the admin's JWT token in the Authorization header.
* - Verify the HTTP status code is 201 (Created).
* - Query the database to ensure the product record is created with the correct details.

Expected Result: Product is added successfully to the database with correct details.

### Test Case ID: TC004

Description: Verify fetching product details by ID.

Preconditions: At least one product exists in the database.

Test Steps:

* - Call the /api/Product/{id} endpoint with a valid product ID using a GET request.
* - Verify the HTTP status code is 200 (OK).
* - Validate the response contains the correct product details including name, description, price, and stock.

Expected Result: The correct product details are retrieved successfully.

## Cart Management

### Test Case ID: TC005

Description: Verify adding items to the cart.

Preconditions: User must be logged in.

Test Steps:

* - Prepare a payload with valid product ID and quantity.
* - Call the /api/Cart/{userId}/add endpoint with the payload using a POST request and include the user's JWT token in the Authorization header.
* - Verify the HTTP status code is 200 (OK).
* - Query the database to ensure the cart is updated with the new item and correct quantity.

Expected Result: The item is added to the cart successfully, and the database is updated.

### Test Case ID: TC006

Description: Verify clearing the cart.

Preconditions: Cart has items for the logged-in user.

Test Steps:

* - Call the /api/Cart/{userId}/clear endpoint using a DELETE request and include the user's JWT token in the Authorization header.
* - Verify the HTTP status code is 200 (OK).
* - Query the database to ensure the cart is empty for the user.

Expected Result: Cart is cleared successfully, and the database reflects the changes.

## Review Management

### Test Case ID: TC007

Description: Verify adding a review for a purchased product.

Preconditions: User must have purchased the product.

Test Steps:

* - Prepare a payload with valid review details including product ID, rating, and comment.
* - Call the /api/Review endpoint with the payload using a POST request and include the user's JWT token in the Authorization header.
* - Verify the HTTP status code is 201 (Created).
* - Query the database to ensure the review is added and the product rating is recalculated.

Expected Result: The review is added successfully, and the product rating is updated.

### Test Case ID: TC008

Description: Verify adding a review for a product not purchased.

Preconditions: User has not purchased the product.

Test Steps:

* - Prepare a payload with valid review details including product ID, rating, and comment.
* - Call the /api/Review endpoint with the payload using a POST request and include the user's JWT token in the Authorization header.
* - Verify the HTTP status code is 400 (Bad Request).
* - Validate the error message indicates the user has not purchased the product.

Expected Result: An error is returned indicating the user has not purchased the product.