# OrderHistoryServiceTest Documentation

## Overview

This document details the unit tests implemented for the OrderHistoryService class in our e-commerce application backend. The OrderHistoryService manages the recording and retrieval of order history for users, tracking completed orders and providing access to previously purchased products.

## Test Class Structure

### Dependencies

The test class uses:

- **JUnit 5**: For test execution and assertions
- **Mockito**: To mock dependencies and simulate interactions
- **Spring Boot Test**: For integration with the Spring testing framework

### Mocked Components

- **OrderHistoryRepository**: Database access for order history operations
- **CartRepository**: Database access for cart operations
- **OrderRepository**: Database access for order operations
- **ProductRepository**: Database access for product operations

### Test Setup

Before each test, the following setup is performed:

1. Initialize mocks using MockitoAnnotations
2. Generate random UUIDs for userId and orderId
3. Create test objects:
    - A test cart associated with the user (empty items list)
    - A test order history record for the user (empty orderIds list)

# Test Cases

## 1. testRecordOrderAndClearCart_ExistingOrderHistory

**Purpose**: Verify that an order can be successfully recorded in a user's order history and their cart cleared.

**Test Scenario**:

- **Arrange**:
    - Configure cartRepository mock to return the test cart for the user
    - Configure orderHistoryRepository mock to return the test order history
    - Configure orderHistoryRepository save method to return the updated order history
- **Act**:
    - Call orderHistoryService.recordOrderAndClearCart with userId and orderId
- **Assert**:
    - Verify the cart was saved after being modified
    - Verify the order history was saved after being modified
    - Verify the orderId was added to the order history's orderIds list
    - Verify the cart's items list was emptied

**Business Logic Verified**:

- Orders are properly recorded in a user's order history
- User's cart is cleared after an order is recorded
- The system maintains the relationship between users and their orders

## 2. testViewPreviousOrdersByUser

**Purpose**: Verify that a user can retrieve a list of their previous shipped orders.

**Test Scenario**:

- **Arrange**:
    - Create two test orders with the user's ID and shipped status set to true
    - Configure orderRepository mock to return these orders when queried
    - Prepare the expected list of order IDs
- **Act**:
    - Call orderHistoryService.viewPreviousOrdersByUser with the userId
- **Assert**:
    - Verify the returned response body matches the expected order IDs
    - Verify the response has 200 status code

**Business Logic Verified**:

- Users can retrieve their order history
- Only shipped orders are included in the response
- The system returns order IDs in the expected format

## 3. testGetProductsFromPreviousOrders

**Purpose**: Verify that a user can retrieve the products from their previous orders.

**Test Scenario**:

- **Arrange**:
    - Create two test products with unique IDs
    - Create two test orders that contain these products
    - Configure orderRepository mock to return the orders when queried
    - Configure productRepository mock to return the products when queried by IDs
- **Act**:
    - Call orderHistoryService.getProductsFromPreviousOrders with the userId
- **Assert**:
    - Verify the returned response body matches the expected products list
    - Verify the response has 200 status code

**Business Logic Verified**:

- Users can retrieve the products from their previous orders
- The system correctly fetches and compiles product information
- The correct product details are returned to the user

# Mocking Strategy

The tests use a consistent mocking strategy to isolate the OrderHistoryService from its dependencies:

1. **Mock Responses**: Return prepared test objects when repository methods are called
2. **Behavior Verification**: Verify that the service calls repository methods as expected
3. **State Verification**: Check that objects are modified correctly before being saved

# Test Coverage

These tests cover the core functionality of the OrderHistoryService:

- Recording new orders in a user's order history

- Clearing the cart after order processing
- Retrieving a user's previous orders
- Obtaining products from previous orders

# Conclusion

The OrderHistoryServiceTest thoroughly verifies the core functionality of the order history system. The tests ensure that orders are properly recorded, carts are cleared after checkout, and users can access their order history and previously purchased products.

These tests help maintain the integrity of the order history system as the application evolves, ensuring that users have reliable access to their purchase history and that order processing works correctly.