

**VIETNAM NATIONAL UNIVERSITY, HANOI**  
**UNIVERSITY OF ENGINEERING AND TECHNOLOGY**



**TESTING DOCUMENT FOR**  
**ONLINE ECOMMERCE WEBSITE**

**SEMESTER 232**

**GROUP 10**

<b>Group members:</b>	<b>Phan Anh Tú</b>	<b>Student ID: 22028238</b>
	<b>Mai Anh Tuấn</b>	<b>Student ID: 22028144</b>
	<b>Nguyễn Ngọc Hưng</b>	<b>Student ID: 22028142</b>
	<b>Nguyễn Đức Phát</b>	<b>Student ID: 22028298</b>
	<b>Nguyễn Nhật Phong</b>	<b>Student ID: 22028272</b>

<b>Course:</b>	<b>Software Engineering</b>
<b>Course ID:</b>	<b>INT2208E 23</b>
<b>Instructor:</b>	<b>Assoc. Prof. Đặng Đức Hạnh</b>
	<b>Bsc. Kiều Văn Tuyên</b>

**HÀ NỘI – 2024**

---

# **Testing Document**

**for**

# **Online Ecommerce Website**

**Version 2.0 approved**

**Prepared by Group 10**

**University of Engineering and Technology**

**April 30<sup>th</sup>, 2024**

Date	Document version	Description	Author
13 <sup>th</sup> April, 2024	0.1	Construct the skeleton of the Content table (What we will write)	Phan Anh Tú
17 <sup>th</sup> May, 2024	1.0	<b>Subsections details authors:</b>  <b>1. Introduction</b>  <b>2. Test Strategy and Approach</b>  <b>3. Test Cases</b>  <b>4. Test Data</b>  <b>5. Test Execution</b>  <b>6. Test Reporting</b>	Mai Anh Tuấn  Nguyễn Đức Phát  Nguyễn Ngọc Hưng, Nguyễn Đức Phát, Mai Anh Tuấn  Nguyễn Ngọc Hưng, Nguyễn Đức Phát, Mai Anh Tuấn  Nguyễn Ngọc Hưng, Nguyễn Đức Phát, Mai Anh Tuấn  Nguyễn Nhật Phong
20 <sup>th</sup> , May, 2024	2.0	Format Document	Phan Anh Tú

## Table of Contents

1.	Introduction.....	8
1.1.	Purpose .....	8
1.2.	Intended Audience and Reading Suggestion .....	8
1.3.	Product Scope .....	9
1.4.	References: .....	10
2.	Test Strategy and Approach.....	10
2.1.	Test Types .....	10
2.1.1.	Unit Tests .....	10
2.1.2.	Functional Tests .....	10
2.2.	Test Techniques.....	10
2.2.1.	Black-box Testing .....	10
2.2.2.	White-box Testing.....	11
2.3.	Test Enviroments.....	11
2.3.1.	Development Environment: Local environment for unit test and integration test. ....	11
2.3.2.	Integration Environment: Test the integration of various modules developed by different teams. ....	11
2.3.3.	User Acceptance Testing (UAT) Environment: Validate the system against business requirements with end-user.....	11
3.	Test Cases .....	11
3.1.	Account Registration .....	11
3.2.	Account Login .....	12
3.3.	Edit Account Information.....	13
3.4.	Search Items .....	13
3.5.	Add Item to Cart.....	14
3.6.	View Cart.....	14
3.7.	Update Cart.....	15
3.8.	Proceed to Buy .....	15
3.9.	Upload Listings .....	16
3.10.	Product Filtering and Sorting .....	16
3.11.	Order Cancellation.....	17
3.12.	Order Confirmation .....	17
3.13.	View Order History .....	17
3.14.	Order Tracking .....	18
4.	Test Data .....	18
4.1.	Data Sources .....	18
4.1.1.	Real-World Data: .....	18
4.1.2.	Generated Data:.....	18

4.2.	Data Preparation .....	18
4.2.1.	Anonymization .....	19
4.2.2.	Data Cleansing .....	19
4.2.3.	Normalization.....	19
4.2.4.	Data Enrichment: .....	19
4.2.5.	Boundary Values:.....	19
4.2.6.	Security Testing: .....	19
5.	Test Execution .....	19
5.1.	Test Schedule.....	20
5.2.	Test Tools .....	20
5.2.1.	<b>PostMan</b> .....	20
5.2.2.	<b>TestComplete</b> .....	21
5.3.	Defect Tracking .....	22
5.3.1.	Bug in login test .....	22
6.	Test Reporting.....	23
6.1.	Test Metrics .....	23
6.1.1.	Test cases pass rate.....	23
6.1.2.	Overall .....	24
6.1.2.	Defect Density.....	24
	Defect density = (Defect Detected/Total Size) = (12/4.909) = 2.44 (Defects/KLOC) .....	25
6.2.	Test Summary Report.....	25
6.2.1.	Using PostMan Test Tool.....	25
6.2.1.1.	Create an account.....	25
6.2.1.2.	View products .....	26
6.2.1.3.	Add a product to cart .....	27
6.2.1.4.	Create an order.....	28
6.2.1.5.	List Orders of product for seller.....	30
6.2.2.	Using TestComplete.....	31

## Table of Figures

Figure 1. Create an account Test Result .....	26
Figure 2. View Product Test Result.....	27
Figure 3. Add a product to cart Test Result .....	28
Figure 4. Create an order Test Result (1).....	29
Figure 5. Create an order Test Result (2).....	30
Figure 6. List Orders of product for seller Test Result .....	31

## Table of Tables

Table 1. Account Registration Test Cases .....	12
Table 2. Account Login Test Cases .....	12
Table 3. Edit Account Information Test Cases .....	13
Table 4. Search Items Test Cases.....	14
Table 5. Add Item to Cart Test Cases .....	14
Table 6. View Cart Test Cases.....	14
Table 7. Update Cart Test Cases .....	15
Table 8. Proceed to Buy Test Cases.....	15
Table 9. Upload Listing Test Cases .....	16
Table 10. Product Filtering and Sorting Test Cases.....	16
Table 11. Order Cancellation Test Cases.....	17
Table 12. Order Confirmation Test Cases .....	17
Table 13. View Order History Test Cases .....	17
Table 14. Order Tracking Test Cases.....	18
Table 15. Test Schedule .....	20
Table 16. Postman Test Tool Report .....	23
Table 17. TestComplete Test Tool Report.....	24
Table 18. Defect Density .....	24
Table 19. Create an account Summary Report.....	25
Table 20. View products Summary Report.....	26
Table 21. Add a product to cart Summary Report .....	27
Table 22. Create an order Summary Report .....	28
Table 23. List Orders of product for seller Summary Report .....	30
Table 24. Account Registration Summary Report .....	32
Table 25. Account Login Summary Report .....	32
Table 26. Add items to Cart Summary Report.....	33
Table 27. Proceed to Buy Summary Report .....	34
Table 28. Product Sorting and Filter Summary Report.....	35

# 1. Introduction

In today's digital age, online commerce has revolutionized the way businesses operate and how consumers purchase goods and services. The rapid advancement of technology has facilitated the development of sophisticated ecommerce platforms that provide seamless and efficient shopping experiences. However, to ensure these platforms deliver optimal performance, reliability, and user satisfaction, rigorous testing is paramount. This document outlines the comprehensive testing strategy for an online ecommerce website developed by Group 10 at the University of Engineering and Technology, Vietnam National University Hanoi.

## *1.1.Purpose*

Testing is a critical component of this project, ensuring that the online ecommerce platform meets the highest standards of quality, performance, and user satisfaction. The primary goals of our testing strategy include identifying and addressing potential issues early in the development cycle, validating that all functionalities work as intended, and ensuring a seamless user experience. Effective testing minimizes the risk of software defects, enhances the reliability and security of the platform, and ultimately contributes to higher customer satisfaction and retention.

## *1.2.Intended Audience and Reading Suggestion*

This document is intended for various stakeholders involved in the project, providing relevant information to each group based on their roles:

- **Developers:** Developers should focus on the technical details, architectural designs, coding standards, and API integration guidelines to understand the testing requirements and ensure proper implementation and maintenance. This will help them identify areas that require rigorous testing and areas where code optimization can be implemented.
- **Project Managers:** Project managers need to review the project roadmap, resource allocation, and risk management strategies. They should focus on milestones and deliverables to keep the project on track. Understanding the testing phases and outcomes will help them make informed decisions about project timelines and resource allocation.
- **Testers:** Testers must concentrate on the detailed test cases, methodologies, and scenarios to perform comprehensive functional, usability, and performance testing. They need to ensure all aspects of the platform are tested thoroughly to meet quality standards and to identify any defects or areas for improvement.
- **Marketing Staff:** Marketing staff should understand the testing outcomes to better plan and execute marketing strategies based on the platform's capabilities and user feedback. Knowledge of the platform's performance and features will help them create effective marketing campaigns and customer engagement strategies.
- **Users:** Users can gain insights into the platform's testing processes, helping them understand the reliability and performance assurances provided. This transparency can build trust and confidence in the platform's quality and security.
- **Documentation Writers:** Documentation writers need to ensure clarity, coherence, and accessibility of information by following the structured layout and content style guidelines provided. They should document the features,



functionalities, and usage guidelines accurately to assist users and stakeholders in navigating and utilizing the platform effectively.

### **1.3. Product Scope**

The scope of this project includes the comprehensive testing of all major components and functionalities of the online e-commerce platform. This detailed testing effort aims to ensure that each core feature functions correctly and provides a seamless user experience. The core features included in this scope are as follows:

1. **User Registration and Login:** Ensuring new users can create accounts with valid credentials and registered users can securely log in to their accounts. This includes validation of input fields, error handling for incorrect inputs, and security measures such as password strength requirements and protection against common vulnerabilities like SQL injection and brute force attacks.
2. **Profile Creation and Management:** Testing will cover the full lifecycle of user profile management, ensuring users can create, update, and manage their profiles securely. This involves verifying personal information management, and the ability to adjust privacy settings. The focus will be on ensuring data integrity, security, and user-friendly interfaces for profile management tasks.
3. **Product Catalog:** This feature involves testing the functionalities for uploading, viewing, and browsing products listed on the platform. The testing will ensure products are displayed accurately with all relevant details such as descriptions, prices, and images. It will also cover the search and filtering functionalities, verifying that users can efficiently search for products using various criteria and that the filters return accurate and relevant results.
4. **Shopping Cart and Checkout Process:** The testing will ensure that users can add, update, and remove items in their shopping cart, and proceed smoothly through the checkout process. This includes validating the accuracy of item quantities, prices, and total amounts. The checkout process will be tested for handling different payment methods securely and ensuring that the user experience is smooth and error-free. Security aspects, such as the protection of payment information and secure transactions, will be thoroughly tested.
5. **Order Management:** This feature covers the entire order lifecycle from placement to tracking and cancellation. Testing will ensure that users can place orders, view order details, and track the status of their orders accurately. It also includes testing the functionality for order updates and cancellations. The system should provide users with appropriate notifications for order updates, such as dispatch and delivery statuses, ensuring that the information is timely and accurate.
6. **Notifications:** The notification system will be tested to ensure users receive timely and accurate notifications for various events, such as order updates, new messages, and promotional offers. The notifications should be relevant, actionable, and delivered through appropriate channels (e.g., email, in-app notifications).

Out of scope for this testing document are external integrations with third-party systems not developed in-house, which includes any external APIs or services that the platform may interact with but are not under the direct control of the development team. Additionally, exploratory testing beyond documented requirements is not included. While exploratory testing can uncover unexpected issues, the focus of this testing strategy is based on predefined requirements and test cases. Furthermore, testing on platforms or devices not specified in the project requirements is excluded. Testing will

be conducted only on the platforms and devices that have been specified in the project requirements. Any other platforms or devices will not be included in the scope of this testing effort. The focus will remain strictly on the components and functionalities developed and maintained by the project team, ensuring a thorough and targeted testing approach.

### ***1.4. References:***

- [1] Ian Sommerville, *Software Engineering 10<sup>th</sup> Edition*, Pearson, 2016.
- [2] Software Engineering course slides provided by lecturers.

## **2. Test Strategy and Approach**

### ***2.1. Test Types***

#### ***2.1.1. Unit Tests***

Unit tests will focus on verifying the smallest parts of the application, typically individual functions or methods within the code. For an eCommerce website, unit tests will include:

Cart Operations: Ensure functions for adding, removing, and updating items in the cart work as expected.

User Authentication: Test the login, registration, and password reset functionalities.

Order Processing: Validate functions responsible for creating and managing orders.

#### ***2.1.2. Functional Tests***

Functional tests will ensure that the end-to-end functionality of the eCommerce website works as intended. These tests will cover:

User Registration and Login: Verify that users can register, login, and logout successfully.

Product Search and Filter: Test the search functionality and filters to ensure users can find products easily.

Shopping Cart: Validate the entire workflow from adding items to the cart, updating quantities, and removing items.

Checkout Process: Ensure that users can complete the checkout process, including entering shipping information, selecting payment methods, and placing orders.

### ***2.2. Test Techniques***

#### ***2.2.1. Black-box Testing***

**Description:** Black-box testing involves testing the software without any knowledge of the internal code structure, implementation details, or internal paths. The tester focuses on inputs and expected outputs without considering how the application achieves those outputs. This method tests the software from the user's perspective and is primarily concerned with validating functional requirements and ensuring the software behaves as expected.

### **Applications:**

**Functional Testing:** Validates the functionality of the software against the requirements. For a hotel booking website, this includes checking if users can search for hotels, make bookings, cancel reservations, etc.

**Regression Testing:** Ensures that new code changes do not adversely affect the existing functionality. This involves re-running previously passed test cases on new builds.

**Acceptance Testing:** Conducted to determine if the system satisfies the business requirements and is ready for deployment. It often involves scenarios that mimic real-world usage.

**Security Testing:** Identifies vulnerabilities and ensures the application is secure from potential threats. This includes testing for SQL injection, cross-site scripting (XSS), etc.

#### **2.2.2. *White-box Testing***

**Description:** White-box testing involves testing the internal structures, logic, and code of the software. The tester needs to have knowledge of the internal workings of the application. This method is used to verify that each path, decision, and condition in the code works as intended and helps identify hidden errors within the code.

### **Applications:**

- *Unit Testing:* Focuses on testing individual units or components of the software to ensure they function correctly. For a hotel booking website, this could involve testing individual functions or methods like login, search, booking, etc.
- *Integration Testing:* Tests the interfaces and interactions between different modules or services to ensure they work together correctly.
- *Code Coverage Analysis:* Ensures that all parts of the code are executed and tested, including statement coverage, branch coverage, and path coverage.

#### **2.3. *Test Environments***

2.3.1. *Development Environment:* Local environment for unit test and integration test.

2.3.2. *Integration Environment:* Test the integration of various modules developed by different teams.

2.3.3. *User Acceptance Testing (UAT) Environment:* Validate the system against business requirements with end-user

## **3. Test Cases**

### **3.1. *Account Registration***

Table 1. Account Registration Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-001	Verify login functionality	Valid username, password	Successful login, user dashboard loads	Login page redirects and shows dashboard	High	None
TC-002	Registration with Existing Email	Email: existinguser@example.com Password: 123456	Error message indicating the email is already registered	Error message displays correctly	High	None
TC-003	Registration with short password	Email: newuser@example.com Password: 123	Error message indicating password too short	Error message displays correctly	Medium	None
TC-004	Registration with Invalid Email or Password	Email: userexample.com, Password: 1234567	Error message indicating invalid email format or password	Error message displays correctly	Medium	None
TC-005	Registration with Missing Information	Email: , Password: 123456	Error message indicating required fields are missing	Error message displays correctly	High	None

### 3.2. Account Login

Table 2. Account Login Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-006	Verify login functionality	Valid username, password	Successful login, user dashboard loads	Login page redirects and shows dashboard	High	None
TC-007	Login with invalid Email Or	Email: userexample.com,	Error message indicating	Error message	Medium	None

	short password	Password: 123	invalid email format or short password	displays correctly		
TC-008	Login with Missing Information	Email: , Password: 123456	Error message indicating required fields are missing	Error message displays correctly	High	None
TC-009	Login with wrong email or password	Email: <a href="mailto:h@gmail.com">h@gmail.com</a> Password: 1234567	Error message indicating that email or password can be wrong and require user re-enter	Error message displays correctly	High	None

### 3.3. *Edit Account Information*

Table 3. Edit Account Information Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-010	Successfully Edit Account Information	Name: new user Email: <a href="mailto:newuser@example.com">newuser@example.com</a> Password: NewPass@123	Account information is successfully updated	Account page shows updated information	Medium	None
TC-011	Edit with Invalid Email	Email: newuserexample.com Password: 123	Error message indicating invalid email format or weak password	Error message displays correctly	Medium	None

### 3.4. *Search Items*

Table 4. Search Items Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-012	Search for Existing Items	Search Query: "laptop"	Display list of laptops	Search results display laptops	Medium	None
TC-013	Search with Blank Query	Search Query: ""	Display list of products do not change	Nothing is change	Medium	None

### 3.5. Add Item to Cart

Table 5. Add Item to Cart Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-014	Add Available Item to Cart	Item ID: 1	Item is successfully added to the cart	Cart shows the added item	High	None
TC-015	Add Item with a quantity higher than its quantity in stock	Quantity: 100 Quantity in stock: 50	Limit the quantity that user can enter is the quantity in stock of products	The quantity user want to buy cannot excess the quantity in stock	Medium	None
TC-016	Add Duplicate Item to Cart	Item ID: 1	Item quantity is updated in the cart	Cart shows updated item quantity	High	None

### 3.6. View Cart

Table 6. View Cart Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
--------------	------------------	--------	-----------------	--------------------	----------	--------------

TC-017	View items in user cart	ID user: 5	A display list for items added to user cart	Cart shows the added item	High	None
--------	-------------------------	------------	---	---------------------------	------	------

### 3.7. *Update Cart*

Table 7. Update Cart Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-018	Remove an Item from the Cart	Item ID: 1	Item is successfully removed from the cart	Cart no longer shows the removed item	Medium	TC-014

### 3.8. *Proceed to Buy*

Table 8. Proceed to Buy Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-019	Checkout with Valid Payment Information	User ID: 1, Payment Details: Valid Payment Information	Purchase is completed and order confirmation is displayed	Order confirmation page shows correctly	High	TC-014
TC-020	Proceed to Buy with Empty Cart	User ID: 2	None	None	Medium	None
TC-021	Proceed to buy successful from cart	User ID: 1 List of Products want to buy	Purchase is completed and users are redirected to a successful page	Redirect to a successful page	High	TC – 019
TC-022	Proceed to buy directly from the product page	User ID: 1 Product ID: 5	Preparation for purchase, and users are redirected to confirm page	Display a confirm page for users, includes the product they want to buy	High	None

### 3.9. Upload Listings

Table 9. Upload Listing Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-023	Upload New Product Listing	Product Details: Valid Product Information	Product is successfully uploaded and listed	Product appears in the product catalog	Medium	None
TC-024	Upload Listing with Missing Information	Product Details: Missing Information	Error message indicating required fields are missing	Error message displays correctly	Medium	None
TC-025	Upload some information for products that are sold such as name, description, quantity, ...	Details for product: name, quantity, price, ...	Information of product is update	Successful message is display to sellers	Medium	None

### 3.10. Product Filtering and Sorting

Table 10. Product Filtering and Sorting Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-026	Filter Products by Category	Category: "Quần áo"	Products filtered to show only "Quần áo" category	Products list displays only items from the selected category	High	None
TC-027	Sort Products by Price	Sort Order: "Low to High"	Products sorted in ascending order of price	Products list displays items sorted by price	High	None



TC-028	Sort Products by Best Seller	Sort Order: "Best Seller"	Products sorted based on best-selling status	Products list displays items sorted by best-selling rank	High	None
--------	------------------------------	---------------------------	--	--	------	------

### 3.11. Order Cancellation

Table 11. Order Cancellation Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-029	Cancel an Order for user	Order ID: 1 User ID: 2	Order is successfully canceled	Order status updated to "Canceled"	Medium	None
TC-030	Cancel an Order for seller	Order ID: 1 Seller ID: 5	Order is successfully canceled	Order status updated to "Canceled"	Medium	None

### 3.12. Order Confirmation

Table 12. Order Confirmation Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC - 031	Confirm an Order for Sellers	Order ID: 7 Seller ID: 6	Order is successfully confirm	Order status updated to "Confirm"	Medium	None

### 3.13. View Order History

Table 13. View Order History Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC - 032	View Order History for users	User ID: 8	A list of Orders is displayed to user	Display a list of orders	Medium	None
TC - 033	View Order History	Product ID: 8	A list of orders for	Display a list of	Medium	None

	following products for sellers	Seller ID: 10	product seller chose is displayed	orders for product		
--	--------------------------------	---------------	-----------------------------------	--------------------	--	--

### 3.14. Order Tracking

Table 14. Order Tracking Test Cases

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies
TC-034	Track Order Status	Order ID: 1	Display current status of the order	Order tracking page shows current order status	Medium	None

## 4. Test Data

### 4.1.Data Sources

Test data will be sourced from a combination of generated and real-world data to provide comprehensive coverage and realistic testing scenarios:

#### 4.1.1.Real-World Data:

Where feasible and permissible, anonymized real-world data will be utilized, offering a more accurate representation of system performance under actual usage conditions. This approach is particularly valuable for validating performance and load testing scenarios. Actual user data obtained from the live environment of the Market Swift. This includes historical user activity such as browsing and viewing products, modifying profiles, payment processing, and transaction histories. Additionally, it will incorporate data from sellers related to adding new products, updating inventory, and modifying product details. This comprehensive approach ensures a realistic evaluation of system performance and user experience.

#### 4.1.2. Generated Data:

A significant portion of the test data will be synthetically generated to encompass a wide range of scenarios. This includes the creation of fictitious yet realistic user profiles, product listings, orders, and order details. For instance, user profiles will contain varied demographic information, diverse purchase histories, and different levels of interaction with the platform. Products will be created with various attributes such as different categories, price ranges, stock levels, and descriptions. Orders and order details will reflect different combinations of product selections, quantities, and customer preferences. The advantage of generated data is the ability to control and replicate specific conditions, which is essential for thorough and consistent testing.

### 4.2.Data Preparation

Special processing and formatting of test data are required to align it with the testing requirements and scenarios, ensuring it is both valid and reliable

#### *4.2.1. Anonymization*

For real-world data, anonymization techniques will be applied to protect personal identifiers and sensitive information. This includes masking or encrypting names, addresses, phone numbers, email addresses, and any other personal data that could lead to the identification of real users. Techniques such as data masking, tokenization, and encryption will be used to ensure that the data remains useful for testing purposes while protecting privacy. Anonymized data allows testers to work with realistic datasets without compromising user confidentiality.

#### *4.2.2. Data Cleansing*

Ensuring that the test data is free from irrelevant, duplicate, or corrupt entries is vital for maintaining the accuracy of the test results. This involves cleaning the data to remove any anomalies or inconsistencies that could skew the test outcomes. Data cleansing processes will include identifying and correcting errors in the data, removing duplicates, and ensuring that all data entries are complete and accurate.

#### *4.2.3. Normalization*

All test data must adhere to the system's expected input structures. This involves standardizing data formats such as date and time formats (e.g., YYYY-MM-DD), numerical values (e.g., using consistent decimal places), and text fields (e.g., ensuring proper case usage and avoiding special characters that are not supported). This consistency is crucial to prevent discrepancies and errors during testing. For example, dates should be formatted uniformly across all test cases to avoid issues with date parsing and comparison.

#### *4.2.4. Data Enrichment:*

In some cases, additional data may need to be added to the existing datasets to ensure comprehensive coverage of all test scenarios. This might involve enriching the data with additional attributes or creating new data points to cover specific test cases. Data enrichment ensures that all possible scenarios are accounted for and that the test data is robust and thorough.

#### *4.2.5. Boundary Values:*

Data will be created to test the boundary values of input fields. This involves identifying the minimum and maximum limits for each input and generating test cases that include values at, just below, and just above these boundaries. For example, if a field accepts numeric input between 1 and 100, test data will include values such as 0, 1, 100, and 101. Testing boundary values helps identify any issues with input validation and ensures that the system can handle edge cases correctly.

#### *4.2.6. Security Testing:*

To identify vulnerabilities, test data will include malicious inputs designed to test the system's security. This includes SQL injection strings, cross-site scripting (XSS) payloads, and other types of malicious data intended to exploit potential security weaknesses. By including these inputs, the testing process can verify that the system properly sanitizes and handles user inputs to prevent security breaches.

## **5. Test Execution**

## 5.1. Test Schedule

Table 15. Test Schedule

Task Name	Start	Finish	Responsibility	Due
Test Planning	14.05	14.05	Nguyen Duc Phat	15.05
Review Requirements documents	14.05	14.05	Nguyen Duc Phat; Nguyen Ngoc Hung	15.05
Create initial test estimates	15.05	16.05	Nguyen Duc Phat	17.05
Design Unit test	15.05	16.05	Nguyen Ngoc Hung	17.05
Design Functional test	15.05	16.05	Nguyen Duc Phat	17.05
Functional testing – Iteration 1	16.05	17.05	Nguyen Duc Phat	19.05
Unit testing - Iteration 1	17.05	17.05	Nguyen Ngoc Hung	19.05
Functional testing – Iteration 2	17.05	18.05	Nguyen Duc Phat	19.05
Unit testing – Iteration 2	18.05	18.05	Nguyen Ngoc Hung	19.05
System testing	18.05	18.05	Nguyen Duc Phat; Nguyen Ngoc Hung	19.05
Report bug and resolve	18.05	18.05	Nguyen Duc Phat; Nguyen Ngoc Hung	19.05
Test Summarize Report	19.05	19.05	Nguyen Nhat Phong	20.05

## 5.2. Test Tools

### 5.2.1. PostMan

Postman is a popular and powerful API testing tool that simplifies the process of developing, testing, and debugging APIs. It provides a user-friendly interface for creating, sending, and managing HTTP requests, making it essential for web developers and testers working with RESTful services and APIs. Postman supports a wide range of HTTP methods (GET, POST, PUT, DELETE, etc.), allowing users to thoroughly test their APIs.

#### ***Key Features of Postman:***

##### 1. API Testing:

Postman provides robust API testing capabilities, allowing users to thoroughly test the functionality, performance, and security of their APIs. Users can create and run tests to validate endpoints, check response times, and ensure the API behaves as expected under various conditions.

##### 2. Automated Testing:

Postman enables users to automate API testing using built-in scripting capabilities. Users can write tests in JavaScript to validate responses, check status codes, and ensure data integrity. Postman's Collection Runner and Newman command-line tool support running automated tests in CI/CD pipelines.

### 3. User-Friendly Interface:

Postman offers an intuitive interface that enables users to create and manage HTTP requests without extensive programming knowledge. It helps organize requests into collections and folders, making it easier to manage large test suites.

### 4. Environment Management:

Postman supports managing environments to store variables such as API endpoints and authentication tokens, useful for testing in different environments (e.g., development, staging, production) without manual configuration changes.

### 5. Unit Testing:

Postman is well-suited for unit testing individual API endpoints. By isolating and testing each endpoint independently, developers can ensure that each function of their API works correctly before integrating them into larger workflows. Postman's ability to write and run test scripts directly in the tool makes it ideal for validating individual units of code.

### 6. Integration with Other Tools:

Postman integrates with various tools and services, including GitHub, Jenkins, and Slack, streamlining workflows and enhancing the testing and development process.

## 5.2.2. TestComplete

TestComplete is a powerful test automation tool with many key features that enhance the efficiency and quality of the software testing process. Here are the key features of TestComplete:

### **Cross-Platform Support:**

Test multiple types of applications: Including web, desktop, and mobile.

Supports popular web browsers: Such as Chrome, Firefox, Edge, etc.

Supports mobile operating systems: Including Android and iOS.

### **Diverse Programming Languages:**

Test scripting: Allows writing test scripts in various languages such as JavaScript, Python, VBScript, JScript, C++Script, and C#Script.

### **Drag-and-Drop Interface:**

User-friendly interface: Enables creating test scripts using drag-and-drop, requiring no advanced programming skills.

### **Keyword-Driven Testing:**

Keyword testing: Uses keywords to build test scripts, making it easy to manage and understand test cases.

### **Object Recognition Engine:**

Powerful object recognition: Identifies and interacts with user interface (UI) components accurately, even when the interface changes.

### **Reusability of Test Scripts:**

Reuse test scripts: Allows reusing test scripts, saving time and effort in the testing process.

#### **Test Visualizer:**

Test visualization tool: Captures screenshots and videos of test steps for easy tracking and debugging.

#### **Data-Driven Testing:**

Data-driven testing: Supports using multiple data sets to test a particular function or module.

### ***5.3. Defect Tracking***

#### ***5.3.1. Bug in login test***

##### **Detection**

During the login phase, an error was identified when a user attempts to log in with non-existent information. The system incorrectly handled this scenario by returning a status code of 200, which indicates a successful request, instead of the appropriate status code of 400, which signifies a bad request due to incorrect information.

##### **Recording the Issue**

The bug was promptly documented and reported to the developer teams using Jira. This ensured that all relevant details were captured, and the development team was made aware of the issue for further analysis.

##### **Analysis and Evaluation**

Upon notification, the development and QA teams convened to discuss the specifics of the error. The discussion aimed to understand the root cause of the problem and devise a solution. It was determined that due to a developer oversight, the system was incorrectly returning a status code of 200. This incorrect response allowed the login process to continue erroneously, bypassing the frontend's mechanism to handle error codes and perform appropriate redirects.

##### **Error Explanation**

The root cause of the error was identified as a misconfiguration in the error handling mechanism within the login API. When users entered incorrect developer information, the system was designed to return a status code of 400. However, due to a bug, it returned 200 instead. This led to the frontend misinterpreting the response as a successful login, thereby allowing the user to proceed without setting the appropriate access tokens.

##### **Fix Implementation**

To rectify this issue, the status code returned by the system was corrected from 200 to 400. This change ensured that when a user entered incorrect or non-existent information, the process was appropriately halted. Additionally, it

ensured that no access tokens were set for invalid login attempts, thereby maintaining the integrity of the login process.

### Retesting and Validation

After implementing the fix, extensive retesting was conducted to validate the solution. The tests confirmed that users could no longer log in with incorrect or non-existent information. The system now correctly returns a status code of 400, triggering the frontend's error handling mechanisms to prevent the login process from continuing erroneously.

### Closure

With the successful resolution and validation of the fix, the issue was marked as resolved in Jira. The bug was closed after ensuring that the login process now functions as intended, providing a robust and error-free user authentication experience.

## 6. Test Reporting

### 6.1. Test Metrics

#### 6.1.1. Test cases pass rate

##### 6.1.1.1. Using PostMan Test Tool

Table 16. Postman Test Tool Report

Test Scenario	Test Case IDs	Test Execution Time (ms)	Number Of Test Cases	Passed Test Case IDs	Failed Test Case IDs	Pass Rate
Create an account	BE-001	339	1	BE-001	None	100%
View Products	BE-002	222	1	BE-002	None	100%
Add a Product to Cart	BE-003	1228	1	BE-003	None	100%
Create an Order	BE-004 BE-005	164 761	2	BE-004 BE-005	None	100%
List Orders of product for seller	BE-006	164	1	BE-006	None	100%

##### 1.1..1. Using TestComplete Test Tool

Table 17. TestComplete Test Tool Report

Test Scenario	Test IDs	Case Number Of Test Cases	Passed Test Case IDs	Failed Test Case IDs	Test Pass Rate
<b>Account Registration</b>	TC-001 TC-002 TC-005	3	TC-001 TC-002 TC-005	None	100%
<b>Account Login</b>	TC-006 TC-008 TC-009	3	TC-006 TC-008	TC-009	66.67%
<b>Add Items to Cart</b>	TC-014 TC-015 TC-016	3	TC-014 TC-015 TC-016	None	100%
<b>Process to Buy</b>	TC-019 TC-021 TC-022	3	TC-019 TC-021 TC-022	None	100%
<b>Product Sorting and Filter</b>	TC-026 TC-027 TC-028	3	TC-026 TC-027 TC-028	None	100%

### 1.1..2. Overall

Passed Test Cases Percentage = (Number of Passed Tests/Total Number of Tests Executed) x 100% = (33/34) x 100% = 97.06%

### 6.1.2. Defect Density

Table 18. Defect Density

Defect	Date of Detection	Date Solved	Module Size (KLOC)
<b>Cart properties</b>	02/05/2024	07/05/2024	1.863
<b>Models' name overlapping</b>	09/05/2024	09/05/2024	1.863
<b>Typo in field name in type model</b>	09/05/2024	09/05/2024	1.863
<b>Typing name for OrdersDetail</b>	09/05/2024	09/05/2024	1.863



<b>Checkout Order function</b>	30/04/2024	10/05/2024	1.863
<b>Importing database path</b>	08/05/2024	10/05/2024	1.863
<b>Login interface response for wrong login information</b>	02/05/2024	11/05/2024	3.046
<b>Handling error in get and delete from cart for user</b>	04/05/2024	12/05/2024	1.863
<b>Changing total product in Checkout Orders</b>	04/05/2024	12/05/2024	1.863
<b>UI contains unwanted components</b>	14/05/2024	18/05/2024	3.046
<b>Returned data of APIs</b>	14/05/2024	18/05/2024	1.863
<b>Deleting products after buying</b>	02/05/2024	18/05/2024	1.863

Defect density = (Defect Detected/Total Size) = (12/4.909) = 2.44 (Defects/KLOC)

## 6.2. Test Summary Report

### 6.2.1. Using PostMan Test Tool

#### 6.2.1.1. Create an account

Table 19. Create an account Summary Report

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies	Pass/Fail
BE-001	Create an account on database	Information: - Email	An account	An account	High	None	Pass

		- Password	is created	is created			
--	--	------------	------------	------------	--	--	--

Test Result:

<input checked="" type="checkbox"/>	password	123456
<input checked="" type="checkbox"/>	email	hungnbc6@gmail.com
<input type="checkbox"/>	name	C

Body

200 OK 339 ms 751 B Save as exam

Pretty Raw Preview Visualize JSON

```

1  {
2    "role": 1,
3    "status": 1,
4    "created_at": "2024-05-18T10:03:17.693Z",
5    "id": 6,
6    "password": "123456",
7    "email": "hungnbc6@gmail.com",
8    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NiwiYWFWb0IjoxNzE2MDI2NTk3LCJleHAiOjE3MTYyODU3OTd9.pKuYnCeNcDbmTRZDzYUvC_s3JpnqCqWlD4jq4Le_E8"
9  }
```

Figure 1. Create an account Test Result

#### 6.2.1.2. View products

Table 20. View products Summary Report

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies	Pass/Fail
BE-002	View products		A list of products from database	Receive a list of products from database	High	None	Pass

Test Result

GET

▼

http://localhost:8000/show

Send

▼

Params

Auth

Headers (10)

Body ●

Pre-req.

Tests

Settings

Cookies

x-www-form-urlencoded ▼

	Key	Value	Description	...	Bulk Edit
<input type="checkbox"/>	username	admin			
<input checked="" type="checkbox"/>	password	123456			
<input checked="" type="checkbox"/>	email	hungnbc6@gmail.com			
<input type="checkbox"/>	name	C			

Body ▼

200 OK

222 ms

4.04 KB

Save as example

...

Pretty

Raw

Preview

Visualize

JSON ▼

☰

📄

🔍

```

2      "products": [
3        {
4          "id": 16,
5          "name": "Tủ lạnh",
6          "price": 2000000,
7          "description": "Tủ lạnh dành cho gia đình",
8          "quantityPerUnit": 1,
9          "unitInStock": 5,
10         "unitInOrders": 0,
11         "reOrderLevel": 2,
12         "listColor": "Trắng",
13         "status": 1,
14         "createdDate": "2024-05-17T15:14:43.000Z",
15         "typeId": 2,
16         "sellerId": 1,
17         "quantitySold": 123,
18         "imageProduct": "https://down-vn.img.susercontent.com/file/
sg-11134201-7rce0-ltozai19jrshc9, https://down-vn.img.
susercontent.com/file/sđ-11134201-7rcdf-ltozaih2wmuin21"

```

Figure 2. View Product Test Result

### 6.2.1.3. Add a product to cart

Table 21. Add a product to cart Summary Report

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies	Pass/Fail
--------------	------------------	--------	-----------------	--------------------	----------	--------------	-----------

BE-003	Add a product to user cart	User ID Product ID	Product is added, Display a success message	Product is added	High	None	Pass
--------	----------------------------	-----------------------	---	------------------	------	------	------

#### Test Result

POST http://localhost:8000/user/cart/add/1

Send

Params Auth Headers (10) **Body** Pre-req. Tests Settings Cookies

x-www-form-urlencoded

	Key	Value	Description	...	Bulk Edit
<input type="checkbox"/>	username	admin			
<input checked="" type="checkbox"/>	password	123456			
<input checked="" type="checkbox"/>	email	hungnbc6@gmail.com			
<input type="checkbox"/>	name				

body 200 OK 1228 ms 281 B Save as example

Pretty Raw Preview Visualize HTML

1 Product added to cart

Figure 3. Add a product to cart Test Result

#### 6.2.1.4. Create an order

Table 22. Create an order Summary Report

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies	Pass/Fail
BE-004	Create an prepare view for orders	User Id Product Ids	Send a list including orderDetails about products that user want to buy	Send a list including orderDetails about products that user want to buy	High	None	Pass
BE-005	Checkout an order	User Id	Create an order and orderDetail	Create an order and orderDetail	High	BE-004	Pass

	and confirm	Product Ids	1, then set status to "Pending"	1, then set status to "Pending"			
--	-------------	-------------	---------------------------------	---------------------------------	--	--	--

Test Result

POST

http://localhost:8000/user/cart/checkout/1

Send

ParamsAuthHeaders (10)Body●Pre-req. Tests◆SettingsCookies

body

200 OK164 ms655 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "message": "Order placed and paid successfully",
3   "orderSummary": {
4     "totalItems": 10,
5     "totalPrice": 8000000,
6     "items": [
7       {
8         "name": "Áo khoác nam",
9         "price": 800000,
10        "quantity": 10,
11        "total": 8000000
12      }
13    ]
14  },
15  "order": {
16    "shipmentDate": "2024-05-18T10:33:15.871Z",
17    "createdDate": "2024-05-18T10:33:15.871Z",
18    "id": 10,
19    "totalPrice": 8000000,
20    "status": 1,
21    "paymentMode": 1,
22    "paymentDate": "2024-05-18T10:33:15.870Z",
23    "userId": "1"
24  }
25 }
```

Figure 4. Create an order Test Result (1)

POST http://localhost:8000/user/cart/prepare/1 Send

Params Auth Headers (10) Body Pre-req. Tests Settings Cookies

x-www-form-urlencoded

<input checked="" type="checkbox"/>	color	1	
<input checked="" type="checkbox"/>	size	2	
<input checked="" type="checkbox"/>	productIds	1 2	
	Key	Value	Description

body 200 OK 761 ms 707 B Save as example

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "productId": 1,
4     "name": "Áo thun nam",
5     "image": "https://down-vn.img.susercontent.com/file/
        vn-11134207-7r98o-lu4h4rucsv816f",
6     "quantity": 2,
7     "price": 200000,
8     "color": 1,
9     "discount": "0.00"
10  },
11  {
12    "productId": 2,
13    "name": "Quần jean nam",
14    "image": "https://down-vn.img.susercontent.com/file/
        vn-11134207-7qukw-lf5rilhgtu1j04, https://down-vn.img.susercontent.
        com/file/vn-11134207-7r98o-luip98pv6nmad8",
15    "quantity": 3,
16    "price": 500000,

```

Figure 5. Create an order Test Result (2)

#### 6.2.1.5. List Orders of product for seller

Table 23. List Orders of product for seller Summary Report

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Dependencies	Pass/Fail
BE-006	List Orders of product for seller	Seller ID	Receive a list of orders	Receive a list of orders of product	High	None	Pass

		Product ID	of product	(ProductDetail)			
--	--	------------	------------	-----------------	--	--	--

## Test Result

GET

▼

http://localhost:8000/seller/listOrders/1

Send

▼

Params

Auth

Headers (10)

Body ●

Pre-req.

Tests ♦

Settings

Cookies

x-www-form-urlencoded ▼

	Key	Value	Description	...	Bulk Edit
<input type="checkbox"/>	username	admin			
<input checked="" type="checkbox"/>	password	123456			
<input checked="" type="checkbox"/>	email	hungnbc2@gmail.com			
<input type="checkbox"/>	name				

Body ▼

200 OK

164 ms

1.38 KB

Save as example

...

Pretty

Raw

Preview

Visualize

JSON ▼

≡

📄

🔍

```

1  [
2    [
3      {
4        "id": 1,
5        "quantity": 2,
6        "unitPrice": "250000.00",
7        "color": 1,
8        "discount": null,
9        "productId": 1,
10       "orderId": 1,
11       "status": 1
12     }
13   ],
14   [
15     {
16       "id": 2,
17       "quantity": 3,
18       "unitPrice": "150000.00",
19       "color": 2,

```

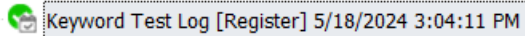
Figure 6. List Orders of product for seller Test Result

## 6.2.2. Using TestComplete

### 6.2.2.1. Account Registration

Table 24. Account Registration Summary Report

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Actual Output	Pass/Fail
TC-001	Verify login functionality	Valid username, password	Successful login, user dashboard loads	Redirects and shows dashboard	High	Redirects and shows dashboard	pass
TC-002	Registration with Existing Email	Email: <a href="mailto:existinguser@example.com">existinguser@example.com</a> Password: 123456	Error message indicating the email is already registered	Error message displays correctly	High	Error message displays correctly	pass
TC-005	Registration with Missing Information	Email: , Password: 123456	Error message indicating required fields are missing	Error message displays correctly	High	Error message displays correctly	pass

Log: 


#### 6.2.2.2. Account Login

Table 25. Account Login Summary Report

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Actual Output	Pass/Fail
TC-006	Verify login functionality	Valid username, password	Successful login, user dashboard loads	Login page redirects and shows dashboard	High	Successful login, user dashboard loads	pass



TC-008	Login with Missing Information	Email: , Password: 123456	Error message indicating required fields are missing	Error message displays correctly	High	Error message indicating required fields are missing	pass
TC-009	Login with wrong email or password	Email: <a href="mailto:h@gmail.com">h@gmail.com</a> Password: 1234567	Error message indicating that email or password can be wrong and require user re-enter	Error message displays correctly	High	Login and return a blank white page	Fail


Log:  Keyword Test Log [Login] 5/18/2024

#### 6.2.2.3. Add items to Cart

Table 26. Add items to Cart Summary Report

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Actual Output	Pass/Fail
TC-014	Add Available Item to Cart	Item ID: 1	Item is successfully added to the cart	Cart shows the added item	High	Item is successfully added to the cart	pass
TC-015	Add Item with a quantity higher than its quantity in stock	Quantity : 100 Quantity in stock: 50	Limit the quantity that user can enter is the quantity in stock of products	The quantity user want to buy can not excess the quantity in stock	Medium	Quantity was limited	pass


TC-016	Add Duplicate Item to Cart	Item ID: 1	Item quantity is updated in the cart	Cart shows updated item quantity	High	Quantity was updated in cart	pass
--------	----------------------------	------------	--------------------------------------	----------------------------------	------	------------------------------	------

Log:  Keyword Test Log [AddToCart] 5/18/2024

#### 6.2.2.4. Proceed to buy

Table 27. Proceed to Buy Summary Report

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Actual Output	Pass/Fail
TC-019	Checkout with Valid Payment Information	User ID: 1, Payment Details: Valid Payment Information	Purchase is completed and order confirmation is displayed	Order confirmation page shows correctly	High	Purchase is completed and order confirmation is displayed	pass
TC-021	Proceed to buy successful from cart	User ID: 1 List of Products want to buy	Purchase is completed and users are redirected to an successful page	Redirect to a successful page	High	Purchase is completed and users are redirected to an successful page	pass
TC-022	Proceed to buy directly from the product page	User ID: 1 Product ID: 5	Preparation for purchase, and users are redirected to confirm page	Display a confirm page for users, includes the product they want to buy	High	Display a confirm page for users, includes the product they want to buy	pass

Log :  Keyword Test Log [ProcessToBuyItem] 5/18/2024

#### 6.2.2.5. Product Sorting and Filter

Table 28. Product Sorting and Filter Summary Report

Test Case ID	Test Description	Inputs	Expected Output	Pass/Fail Criteria	Priority	Actual Output	Pass/Fail
TC-026	Filter Products by Category	Category : "Quần áo"	Products filtered to show only "Quần áo" category	Products list displays only items from the selected category	High	Products filtered to show only "Quần áo" category	Pass
TC-027	Sort Products by Price	Sort Order: "Low to High"	Products sorted in ascending order of price	Products list displays items sorted by price	High	Products sorted in ascending order of price	Pass
TC-028	Sort Products by Best Seller	Sort Order: "Best Seller"	Products sorted based on best-selling status	Products list displays items sorted by best-selling rank	High	Products sorted based on best-selling status	Pass

Log: None (This part is automated when code so there is no need to use automation tools and is thoroughly tested by developers)