**HO CHI MINH CITY UNIVERSITY OF INDUSTRY AND TRADE HO CHI MINH CITY**

**FACULTY OF INFORMATION TECHNOLOGY**

**----o0o----**

**FINAL PROJECT REPORT**

**SOFTWARE VALIDATION**

**TOPIC: TEST PLAN DOCUMENT FOR COMPUTER STORE MANAGEMENT WEBSITE PROJECT**

**Lecturer**: Ngo Minh Anh Thu

**Students undertake:**

1. 2001222539 – Nguyen Thanh Ly Ly

2. 2001222435 – Huynh Hoang Hai Long

3. 2001224956 – Lai Phuoc Thinh

4. 2001224204 – Le Tan Tai

*Ho Chi Minh City, April 2025*

**TABLE OF CONTENTS**

[CHAPTER 1. TEST STRATEGY 1](#_Toc200106353)

[1.1. Introduction 1](#_Toc200106354)

[1.2. Test scope 1](#_Toc200106355)

[1.2.1. Characteristics to be tested 2](#_Toc200106356)

[1.2.2. Non-test characteristics 2](#_Toc200106357)

[1.3. Subjects of use 2](#_Toc200106358)

[1.4. Arising problems and risks 3](#_Toc200106359)

[1.5. Test logic 3](#_Toc200106360)

[1.5.1. The person who will perform the test 3](#_Toc200106361)

[1.5.2. The right time to test 3](#_Toc200106362)

[1.5.3. Supporting test tools 4](#_Toc200106363)

[1.6. Objectives of the test 4](#_Toc200106364)

[1.7. Definitions and acronyms 4](#_Toc200106365)

[1.8. References 5](#_Toc200106366)

[CHAPTER 2. TEST STANDARDS 6](#_Toc200106367)

[2.1. Standards for conducting tests 6](#_Toc200106368)

[2.1.1. Stop test criteria 6](#_Toc200106369)

[2.1.2. Output standards 7](#_Toc200106370)

[2.2. Personnel planning 7](#_Toc200106371)

[2.3. Test environment 8](#_Toc200106372)

[2.4. Requirements for functional testing 9](#_Toc200106373)

[2.5. Non-functional requirements 10](#_Toc200106374)

[2.6. Testing strategy 11](#_Toc200106375)

[2.6.1. Types of tests 11](#_Toc200106376)

[2.6.2. Test methods 11](#_Toc200106377)

[2.6.3. Testing process 12](#_Toc200106378)

[2.6.4. Test details 13](#_Toc200106379)

[2.7. Systems and Environment 15](#_Toc200106380)

[CHAPTER 3. PLANS AND PROJECTIONS 17](#_Toc200106381)

[3.1. Evaluation of the entire project 17](#_Toc200106382)

[3.2. Plan for completing tasks 17](#_Toc200106383)

[3.3. Before the test 17](#_Toc200106384)

[3.4. During the test 18](#_Toc200106385)

[3.5. After the test round 18](#_Toc200106386)

[3.6. Risks and assumptions 18](#_Toc200106387)

[3.7. Handover products 19](#_Toc200106388)

[CHAPTER 4. CONCLUSIONS AND RECOMMENDATIONS 21](#_Toc200106389)

[4.1. Summary of test results 21](#_Toc200106390)

[4.2. Recommendations 21](#_Toc200106391)

**PREFACE**

This Test Plan is designed to ensure that the Computer Store e-commerce system operates stably, meeting the functional and non-functional requirements according to the SRS/SDS document. The goal is to verify correctness, performance, security, and user experience through comprehensive testing, including functions such as registration, cart, ordering, evaluation, and system administration.

This document provides:

* Detailed Test Plan: Types of tests, methods, and schedules.
* Specific Test Case: Covers key and additional functions (address management, favorites list, external payments).
* Acceptance criteria: Ensure the system is ready to deploy.
* Risk management: Minimize problems during testing

# CHAPTER 1. TEST STRATEGY

## ****1.1. Introduction****

This Test Plan is built to ensure that the CuaHangMayTinh e-commerce system operates stably, meeting the functional and non-functional requirements according to the SRS/SDS document. The goal is to verify correctness, performance, security, and user experience through comprehensive testing, including functions such as registration, cart, ordering, evaluation, and system administration.

This document provides:

Detailed Test Plan:

* Types of tests, methods, and schedules.
* Specific Test Case: Covers key and additional functions (address management, favorites list, external payments).
* Acceptance criteria: Ensure the system is ready to deploy.
* Risk management: Minimize problems during testing.

## ****1.2. Test scope****

The scope of the test includes:

**Customer Functions:**

* Register, login, forgot password
* View, search, filter products.
* Product details, reviews.
* Shopping cart: add, delete, update.
* Order, confirm, track orders.
* Manage personal information, delivery address.
* Contact.

**Admin Functions:**

* Product management.
* Order management, reviews, feedback.
* Statistical dashboard.

**Non-functional requirements:**

* < 3-second page response with 100 concurrent users.
* Responsive interface on desktop, tablet, mobile.
* Security: Password encryption (bcrypt), anti-SQL Injection, XSS, CSRF.

**Out of scope:**

* Deep security testing (DDoS) penetration testing.
* Performance testing with >1000 users.
* OpenAI API integration (not yet implemented).

### 1.2.1. Characteristics to be tested

* User Registration/Login.
* Add/Remove products to cart.
* Order and process payment.
* Show product details.
* Product search and filtering function.
* Admin page (CRUD products, order management).
* Security testing (SQL Injection, XSS).

### 1.2.2. Non-test characteristics

* Server infrastructure and hosting services.
* Third-party payment systems (integration testing only).

## 1.3. Subjects of use

* Group 6: Support debugging and backend testing.
* Group 6: Conducting tests and reporting errors.
* Project Management: Track progress and quality.
* Customer/Administrator: Confirm the system meets the requirements.

## 1.4. Arising problems and risks

1. Difficulty in simulating boundary cases.
2. The change in requirements in the middle of the development process.
3. Lack of resources to test the actual environment.
4. Undetected security risk.

## 1.5. Test logic

### 1.5.1. The person who will perform the test

Internal team 06 and third-party testers.

### 1.5.2. The right time to test

Choosing the right time to perform the test is extremely important, in order to ensure that the testing process takes place efficiently, save time and resources, and detect potential errors in the system in a timely manner. In this project, the team determined that the appropriate time to conduct testing was after each development sprint, especially when the main functions had been completed and the development team had confirmed stable operation in the development environment. In addition, critical testing rounds are performed immediately before the product is deployed to production to ensure the system fully meets functional, non-functional, performance, and security requirements.

Specifically, functional testing and UI testing will be carried out after completing each module. Integration testing will be conducted after the modules have been paired and it is necessary to evaluate the compatibility and connection between components. Performance testing and security testing will be conducted when the system is stable, has enough simulation data, and is ready to measure the load under real-world scenarios. Finally, regression testing is prioritized after each source code update or after a bug fix, to ensure that old functions are not affected by new changes.

The rational arrangement of testing time helps team 6 optimize resources, minimize risks and improve the quality of output products.

### 1.5.3. Supporting test tools

|  |  |  |
| --- | --- | --- |
| **Activity** | **Tool** | **Purpose** |
| Functional Testing | Selenium | User Flow Testing |
| API Testing | Postman | Check request/response |
| Performance Testing | JMeter | Load Metering 100 Users |
| Security Testing | OWASP ZAP | SQLi, XSS Detection |
| Database Testing | SQL Server Management Studio | Query checks, stored procedure |
| Error management | Jira | Error logging and tracking |

## 1.6. Objectives of the test

* Make sure the functions are working properly according to the specification.
* Verify system performance and security.
* Ensure a smooth interface and user experience.

## 1.7. Definitions and acronyms

|  |  |
| --- | --- |
| **Terminology** | **Interpretation** |
| FR | Functional Requirements |
| NFR | Non-functional Requirements |
| UI/UX | User Interface/User Experience |
| Test Case | Testing Scenarios |
| Regression | Regression Testing |
| Smoke Test | Key Feature Quick Test |
| SRS | Software Requirements Specification |
| SDS | Software Design Specification |

## 1.8. References

* SRS/SDS: Nhom06\_SRS\_SDS.docx.
* Source file: WebsiteMuaBanMayTinh\_KDPM\_Detest1
* Standard: IEEE 829-2008, ISO/IEC 25010.

# CHAPTER 2. TEST STANDARDS

## 2.1. Standards for conducting tests

### 2.1.1. Stop test criteria

The stop test standard is defined to ensure that the testing process will be paused at the right time when conditions are not guaranteed to continue, avoid wasting resources and time, and minimize risks that may affect the progress and quality of the project. Within the scope of this project, the team sets out the following criteria for suspension of testing:

* When critical bugs are detected that disrupt or directly affect the main functions of the system, such as page crashes, data loss, and critical logic errors that prevent users from completing the purchase or payment process.
* When blocker-type errors appear, such as system crashes, loss of access (service unavailable), inability to log in, shopping cart unable to add products, or errors in modules that have a chain effect on the entire system.
* When the test environment is unavailable or unstable, including problems such as server down, database errors, unable to connect APIs, and persistent network errors, test tools cannot operate properly.
* When there is a request to change the project scope, adjust the functionality from the customer, lecturer or project manager, leading to a significant change in the test requirements, making the current test cases no longer relevant.
* When the number of outstanding errors exceeds the allowable threshold agreed upon in the test plan, affecting the reliability of subsequent test results.

The application of these test suspension standards will help the test team (Group 6) proactively propose to stop testing in a timely manner, coordinate closely with the development team (developer) to overcome outstanding problems, ensure the next testing process is most effective and minimize the occurrence of repeated errors.

### 2.1.2. Output standards

The system is satisfactory if:

* 100% of the test cases for the main function are passed.
* No critical errors (crashes, data loss, logic errors).
* The pass test case rate ≥ 95%.
* Page response < 3 seconds with 100 users.
* Responsive interface on devices and browsers.
* No underlying security vulnerabilities (SQL Injection, XSS, CSRF).

## 2.2. Personnel planning

| **Role** | **Responsibility** | **Required Skills** |
| --- | --- | --- |
| **Test Manager** | Planning, supervising testing, approving handover. | Project management, communication, understanding the Agile process. |
| **Lead** | Coordinate testers, review test cases, report bugs on Jira. | Software testing, using Jira, requirements analysis. |
| **Tester** | Write test cases, manual tests (Login.cshtml, GioHang.cshtml), report bugs. | Write test cases, test interfaces/functions. |
| **Automation Tester** | Write a Selenium script for the order flow, test the API using Postman. | Programming (Python/Java), Selenium, Postman. |
| **Performance Tester** | Performance testing using JMeter (Index.cshtml, DatHangThanhCong.cshtml). | JMeter, performance analysis, database optimization. |
| **Security Tester** | Security checks (SQLi, XSS) on Login.cshtml, XacNhanDonHang.cshtml using OWASP ZAP. | OWASP Top 10, OWASP ZAP, web security. |
| **UI/UX Tester** | Check responsiveness, falling leaf effect (DatHangThanhCong.cshtml), UX on Index.cshtml. | UX testing, WCAG 2.1, Browser Developer Tools. |
| **Database Tester** | Examine data (MT, DonHang) and stored procedures using SQL Server. | SQL, database testing, query optimization. |
| **Developer** | Fix logic errors (GioHangController), test units, support Tester. | C# programming, unit testing, debug code. |

## 2.3. Test environment

* Hardware: Laptop/PC, iOS and Android phones.
* Software: Browsers (Chrome, Firefox, Edge), Windows and macOS operating systems.
* Server: The staging environment is the same as the production environment.

## 2.4. Requirements for functional testing

Based on SRS and additional ideas:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Function** | **Describe** | **Priority** |
| FR-001 | Register an account | Register by email, phone number (Register.cshtml). | Tall |
| FR-002 | Log | Log in with your email/phone number and password (Login.cshtml). | Tall |
| FR-003 | Forgot Password | Send the new password via email (Forgot.cshtml). | Average |
| FR-004 | Product Show | Display products, search, filter by category/price (Index.cshtml). | Tall |
| FR-005 | Detailed Product Description | View details, add to favorites (Details.cshtml). | Tall |
| FR-006 | Product Reviews | Reviews (1-5 stars, reviews) after purchase (Details.cshtml). | Average |
| FR-007 | Cart Management | Add, delete, update quantity (GioHang.cshtml, ThemGioHang.cshtml). | Tall |
| FR-008 | Place an order | Enter the address, select payment, and confirm (XacNhanDonHang.cshtml). | Tall |
| FR-009 | Order Tracking | See the list, order details (DanhSachDonHang.cshtml, SeeDonHang.cshtml). | Tall |
| FR-010 | Management of Personal Information | Update information, change password (ThongTin.cshtml). | Tall |
| FR-011 | Manage shipping addresses | Add, edit, delete, select the default address (ThongTin.cshtml). | Average |
| FR-012 | Wishlist | Add, remove products, move to cart (Details.cshtml). | Average |
| FR-013 | Contact | Send feedback via the form (LienHe.cshtml). | Average |
| FR-014 | Product Manager (Admin) | Add, edit, delete products (Add.cshtml, Edit.cshtml, Delete.cshtml). | Tall |
| FR-015 | Order Management (Admin) | Browse, update, cancel orders (DSDonHang.cshtml). | Tall |
| FR-016 | Review Manager (Admin) | View and delete inappropriate reviews (DSPhanhoi.cshtml). | Average |
| FR-017 | Statistics Dashboard (Admin) | Display revenue, inventory, order number. | Average |

## 2.5. Non-functional requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Request** | **Describe** | **Priority** |
| NFR-001 | Performance | Page response < 3 seconds with 100 users. | Tall |
| NFR-002 | Interface | Responsive on desktop (1200px), tablet (768px), mobile (576px). | Tall |
| NFR-003 | Confidentiality | Password encryption (bcrypt), anti-SQL Injection, XSS, CSRF. | Tall |
| NFR-004 | Compatible | Works on Chrome, Firefox, Edge, Safari. | Tall |
| NFR-005 | Usability | Friendly interface, clear action button, white-blue-black color. | Average |

## 2.6. Testing strategy

### 2.6.1. Types of tests

* **Functional testing**: Registration verification, shopping cart, ordering, evaluation, administration.
* **Interface test**: Responsive, effects (falling leaves, hover).
* **Integration testing**: Frontend-backend interaction, inventory updates, billing.
* **Performance Test**: Measures response time under load.
* **Security testing**: Authentication, encryption, anti-attack.
* **Regression testing**: Re-run the test case after fixing the error.
* **Compatibility testing**: Testing across browsers and devices.

### 2.6.2. Test methods

In this software testing project, the team applies a flexible combination of various testing methods to ensure comprehensive test coverage, high accuracy and early detection of potential errors. The test methods are selected in accordance with each specific type of test, ensuring both functional testing and evaluation of performance, security, and user experience.

Specifically, the team uses the following methods:

* **Manual Testing:** This is a testing method that is performed directly by the tester without the use of automation tools, mainly applied to UI testing, UX testing, functional testing, and initial regression testing. This method helps detect errors related to interface, layout, user operation, usability, and problems that are difficult to recognize with automated tools.
* **Automation Testing:** Used for iterative, well-scripted tests that require fast testing on multiple runs, such as regression testing, happy path testing, or API testing. In the project, tools such as **Selenium WebDriver** are applied to automate interface testing, Postman is used to test APIs, which help speed up testing, reduce errors, and save effort for the Team 6 team.
* **Black-box Testing:** This is a software requirements-based (SRS) testing method that tests inputs and outputs without regard to the internal processing logic. This method is mainly used to evaluate the main functions of the system, including login, registration, shopping cart, ordering, payment,...
* **White-box Testing:** Applied to check in detail the source code and internal processing logic of the system. Developers will perform unit testing, logic controller testing, stored procedures, and backend components to ensure the accuracy, optimization, and security of the source code.
* **Integration Testing:** Used to test the process of communication and data exchange between modules or between frontend and backend. This approach helps ensure that individual components work smoothly when integrated into a complete system.

The team selects the appropriate testing method based on the actual requirements, the nature of each module and the available resources, thereby helping the testing process achieve the highest efficiency, timely detection of errors, risk mitigation and software quality assurance before deployment.

### 2.6.3. Testing process

1. **Planning**: Identify test cases, resources, schedules.
2. **Test case design**: Write detailed test cases based on SRS.
3. **Perform the test**:
   * Group 6 manually tests the interface and functionality.
   * Developers run automated tests of APIs and databases.
4. **Bug report**: Recorded on Jira (description, level, snapshot).
5. **Bug fixes and regressions**: Developers fix bugs, double-check.
6. **Result report**: Summary of pass/fail ratio, remaining errors.

### 2.6.4. Test details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Purpose of the test** | **Testing Techniques** | **Applicable Standards** | **Responsible roles** | **How to test** | **Exception Handling** |
| **Functional Testing** | Edge value analysis, user flow testing | ISO/IEC 25010 | Tester | Design SRS-based test cases for functions such as login (Login.cshtml), add shopping cart (ThemGioHang.cshtml), place an order (XacNhanDonHang.cshtml). Run manually and automatically using Selenium to test the flow from registration to successful order. | Log errors into Jira with screenshots, developers fix the logic, and retest the entire thread. |
| **Security Testing** | Vulnerability testing, validation testing | OWASP Top 10, ISO/IEC 27001 | Security Expert | Use Burp Suite to test SQL Injection and XSS on the login form (Login.cshtml) and order (XacNhanDonHang.cshtml). Verify password encryption (SHA-256) and CSRF tokens during the session. | Patch the hole, add an input filter, test again with Burp Suite. |
| **Performance Testing** | Load test, latency test | ISO/IEC 9126 | DevOps, Tester | Run LoadRunner to simulate 100 users accessing /MayTinh/Details and /GioHang/DatHangThanhCong. Measure response time (< 3 seconds) and CPU usage. | Optimize queries in GioHangController, add indexes to MT tables, check again under load. |
| **Regression Testing** | Repeatable testing, automation | ISO/IEC 29119 | Automation Tester | Automate test cases using Selenium WebDriver for key functions (shopping cart, ordering). Run again after each code update to make sure no new errors appear on GioHang.cshtml or DanhSachDonHang.cshtml. | Noting new bugs, developers fix and re-run the entire test suite. |
| **UI/UX Testing** | Interactive testing, responsive testing | WCAG 2.1 | UX Designer, Tester | Test the interface on Chrome, Firefox, Edge with a resolution of 1920x1080, 768px, 576px. Evaluate the falling leaf effect (DatHangThanhCong.cshtml) and product grid layout (Index.cshtml) for smoothness and intuitiveness. | Adjust CSS/JavaScript, improve effect load times, retest on devices. |
| **API Testing** | Integration testing, error testing | RESTful | Developer, Tester | Use Postman to test the APIs in KhachHangController (login, update information) and GioHangController (add/remove carts). Verify the response code and data payload. | Fixed API logic, added detailed error messages, re-checked all endpoints. |
| **Compatibility Testing** | Cross-platform testing, browser testing | W3C, CSS3 | Tester, Developer | Check the interface and functionality on Chrome, Firefox, Edge, and mobile devices (iPhone 12, iPad Air) with views such as Details.cshtml, ThongTin.cshtml. Make sure the hover and carousel effect work properly. | Fixed CSS/JS errors, double-checked on other browsers and devices. |
| **Data Testing** | Integrity testing, mass testing | ACID | DBA, Tester | Run SQL queries on SQL Server Management Studio to check the DonHang, CTDonHang, and DanhGia tables. Test stored procedure CapNhatKhoduaTrenDonHang with simulated data (100 products, 50 orders). | Optimize stored procedures, fix data errors, restore data from backups if necessary. |

## 2.7. Systems and Environment

* **Staging Medium**:
  + Servers: Windows Server 2019, 4GB RAM, Intel Core i5 CPU, 256GB SSD.
  + CSDL: SQL Server 2019.
  + Simulated data: 100 products, 50 orders, 20 accounts.
* **Testing Equipment**:
  + PC: Windows 11, 1920x1080.
  + Tablet: iPad Air (768px).
  + Mobile: iPhone 12 (576px).
* **Browser**: Chrome v120, Firefox v115, Edge v120, Safari v16.

# CHAPTER 3. PLANS AND PROJECTIONS

## 3.1. Evaluation of the entire project

Medium-sized projects, e-commerce websites need to ensure stability and good security. Estimated test duration: 4 weeks.

## 3.2. Plan for completing tasks

|  |  |  |
| --- | --- | --- |
| **Stage** | **Time** | **Content** |
| Requirements Analysis | 15/04/2025 – 05/05/2025 | Study SRS/SDS, Q&A, make a Test Plan |
| Setting up the environment | 16/04/2025 | Install Visual Studio, SQL Server, DB Recovery |
| Write a test case | 06-07/05/2025 | Write 170 test cases for FR/NFR |
| Functional Testing | 07-09/05/2025 | Test registration, shopping cart, order, admin |
| Interface Testing | 10/05/2025 | Responsive, falling leaf effect |
| Integration Testing | 11/05/2025 | Frontend-backend, VNPay payments, warehouse updates |
| Performance Testing | 12/05/2025 | Load Measurement with JMeter |
| Security Testing | 12/05/2025 | Running OWASP ZAP |
| Regression Testing | 13/05/2025 | Re-run the test case after fixing the error |
| Report Results | 14/05/2025 | Summarizing reports and handing over |

## 3.3. Before the test

* Prepare test data
* Setting up the environment
* Check out the resources
* Complete the main function code.

## 3.4. During the test

* How to report a bug
* Error management tools (Excel, Jira, TestRail...)
* Record detection errors, take screenshots.
* Assess the severity of the error.

## 3.5. After the test round

* Summarize the results.
* Evaluating the effectiveness of the test process
* Meeting with the dev team to fix the bug.
* Official test report.

## 3.6. Risks and assumptions

|  |  |  |
| --- | --- | --- |
| **Risk** | **Degree** | **Remedies** |
| Wrong logic to update inventory when ordering | Tall | Test stored procedure CapNhatKhoduaTrenDonHang with SQL Server, run the test case for XacNhanDonHang.cshtml. |
| Security vulnerability in the login form | Tall | Use Burp Suite to check SQLi, XSS on Login.cshtml, add CSRF tokens, SHA-256 encryption. |
| Slow page loading (> 3 seconds) | Tall | Measure performance using JMeter on Details.cshtml, optimizing MT table queries. |
| Unresponsive interface on mobile | Average | Check Index.cshtml, GioHang.cshtml on 576px resolution using Chrome DevTools. |
| The effect of falling leaves causes lag | Average | Test the effect in DatHangThanhCong.cshtml, optimize JavaScript with DevTools Browser. |
| Lack of simulation data | Average | Prepare 100 products, 50 orders in SQL Server before testing. |
| Server staging is unstable | Tall | Back up your data daily, test Windows Server 2019 and SQL Server before testing. |
| Test cases have not been fully covered | Short | Review test cases based on SRS, add scripts to Details.cshtml, ThongTin.cshtml. |

## 3.7. Handover products

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Products handed over** | **Time** | **Handover** | **Recipient** | **Detailed Description** |
| **Test Plan** | 05/05/2025 | Test Manager | Project Management | Test plan, scope definition, schedule, 70 test cases. |
| **Test Cases** | 07/05/2025 | Group 6 | Team Leader | 70 test cases (Excel) for Login.cshtml, GioHang.cshtml, XacNhanDonHang.cshtml. |
| **Test Scripts** | 12/05/2025 | Automation Tester | Team Leader | Selenium script, Postman for order flow, and GioHangController API. |
| **Test Data** | 06/05/2025 | Group 6 | Developer | SQL data: 100 products, 50 orders for MT board, DonHang. |
| **Bug Report** | 14/05/2025 | Group 6 | Developer | Report bugs (Jira) from Login.cshtml, XacNhanDonHang.cshtml, with level. |
| **Test Report** | 14/05/2025 | Team Leader | Project Management | Report (PDF): 95% ≥ pass rate, performance results, responsiveness. |
| **UAT Report** | 15/05/2025 | Test Lead | Customer | Confirm the system meets SRS for DatHangThanhCong.cshtml, DanhSachDonHang.cshtml. |

# CHAPTER 4. CONCLUSIONS AND RECOMMENDATIONS

## 4.1. Summary of test results

After the process of implementing the test based on the plan and the test cases that have been built, the project implementation team has conducted a total of all test cases according to the plan. The test results recorded a high pass rate of test cases, most of the main functions were operating stably, ensuring the requirements described in the SRS/SDS specification document. A number of bugs have been detected, including critical bugs and minor bugs related to the user interface and user experience (UX/UI). These errors have been recorded with illustrations (screenshots) to serve the analysis, evaluation and remediation process.

## 4.2. Recommendations

Based on the test results, the team proposed a number of recommendations to improve product quality before putting it into practice. Firstly, critical errors need to be prioritized and thoroughly fixed before the system is deployed, in order to ensure stability and safety for end users. Second, the team recommends that the development team consider optimizing system performance, especially in features that require large amounts of data retrieval, such as searching, filtering products, and shopping carts, to ensure that the response speed meets the set standards even under high load conditions. Finally, the team proposes to add and expand automation tests for regression tests in the following development rounds. This not only saves time, minimizes the risk of missing errors, but also contributes to improving professionalism, modernizing the testing process and ensuring product quality in the long term.

In general, the testing process has brought many positive results, helping to detect and fix existing problems in a timely manner, and provide useful solutions and recommendations for the product completion stage. This will be an important premise for the Computer Store Management Website project to achieve success when implemented in practice, meeting the requirements of functionality, performance, security and user experience.