**TEST PLAN**

**MỤC LỤC**

[**1.INTRODUCTION** 1](#_Toc200098881)

[1.1. Purpose 1](#_Toc200098882)

[1.2. Scope 1](#_Toc200098883)

[1.3. Definitions and Abbreviations 2](#_Toc200098884)

[1.4. Functional Requirements 2](#_Toc200098885)

[**1.4.1. Registration and Login (Core Functionality)** 2](#_Toc200098886)

[**1.4.2. Personal Account Management (Supplementary Functionality)** 2](#_Toc200098887)

[**1.4.3. Product Search and Management (Core Functionality)** 2](#_Toc200098888)

[**1.4.4. Cart and Checkout (Core Functionality)** 3](#_Toc200098889)

[**1.4.5. Reviews and Feedback (Supplementary Functionality)** 3](#_Toc200098890)

[**1.4.6. Shipping Address Management** 3](#_Toc200098891)

[**1.4.7. Wishlist** 3](#_Toc200098892)

[**1.4.8. Admin Dashboard (Analytics)** 3](#_Toc200098893)

[**1.4.9. Notification System** 4](#_Toc200098894)

[**1.4.10. Review Management (Admin)** 4](#_Toc200098895)

[**1.4.11. External Payment Integration** 4](#_Toc200098896)

[**1.4.12. Admin Features** 4](#_Toc200098897)

[1.5. Non-functional Requirements 4](#_Toc200098898)

[**2. TEST OBJECTIVES** 5](#_Toc200098899)

[**3. TEST STRATEGY** 6](#_Toc200098900)

[**4. TESTING TOOLS AND ENVIRONMENT** 7](#_Toc200098901)

[4.1. Testing Environment 7](#_Toc200098902)

[4.2. Testing Tools 8](#_Toc200098903)

[**5. PRODUCT ACCEPTANCE CRITERIA** 8](#_Toc200098904)

[5.1. Exit Criteria 8](#_Toc200098905)

[5.2. Completion Criteria 9](#_Toc200098906)

[**6. RESOURCE ALLOCATION** 9](#_Toc200098907)

[**7. TESTING SCHEDULE** 10](#_Toc200098908)

[7.1. Test Milestones 10](#_Toc200098909)

[7.2. Detailed Test Plan 10](#_Toc200098910)

[**8. RISKS AND ASSUMPTIONS** 11](#_Toc200098911)

[**9. DELIVERABLES** 12](#_Toc200098912)

**PROJECT:** Computer Sales Management Website (CuaHangMayTinh2)  
**Draft Date:** 23/04/2025  
**Prepared by:** Group 3 – Software Testing  
**Version:** 1.0

# **1.INTRODUCTION**

## 1.1. Purpose

This document serves as a comprehensive plan for testing the Computer and RAM/SSD Sales Website to ensure that the system operates according to specifications. Making sure this system performs efficiently and is secure before being deployed in a real-world environment.

## 1.2. Scope

The test plan focuses on core system functionalities, including both Users and Admin:

* User:
* Account registration and login
* User information management
* Product search and display
* Adding to cart and checkout
* Order management and product reviews
* Admin:
* Acount registration and login
* Users management
* Product management
* Order management
* Comment management
* Account management
* Promotion management

## 1.3. Definitions and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Meaning** |
| **FR** | Functional Requirements – Core features of the system |
| **NFR** | Non-functional Requirements – Performance, security, etc. |
| **UI/UX** | User Interface / User Experience |
| **Test Case** | A testing scenario to verify a specific function |
| **Regression** | Regression Testing – Re-testing to confirm bug fixes |
| **Smoke Test** | Quick test of critical features |
| **SRS** | Software Requirements Specification |
| **SDS** | Software Design Specification |

## 1.4. Functional Requirements

### **1.4.1. Registration and Login (Core Functionality)**

* Users can register an account using email, phone number, or quick login via Google/Apple.
* Users can log in using username/emai and password.
* Support for quick login via Google or Apple. ( not operate)
* Password recovery and reset functionality via email/phone. (not operate)
* Temporary account lockout after multiple failed login attempts.

### **1.4.2. Personal Account Management (Supplementary Functionality)**

* Users can update their profile picture, name, address, phone number, and email.
* Ability to change password.
* Clear display of order history and order status.
* Users can see their wishlist items.

### **1.4.3. Product Search and Management (Core Functionality)**

* Users can search by keywords and filter by category, price, or brand.
* Product details include name, images, specifications, price, description, and availability.
* Option to add products to a wishlist.

### **1.4.4. Cart and Checkout (Core Functionality)**

* Users can add products to the cart, modify quantities, or remove items.
* Select delivery method and enter shipping address.
* Choose payment method (COD or integrated VNPay, if available).

### **1.4.5. Reviews and Feedback (Supplementary Functionality)**

* Users can rate products with stars and leave comments after purchase.
* Ability to view other users’ reviews for each product.

### **1.4.6. Shipping Address Management**

* Users can add multiple delivery addresses with details (recipient name, phone number, specific address, ward/commune, district, city/province).
* Option to set a default address for quick checkout.
* Edit or delete saved addresses.
* Display addresses in a list with quick selection support.

### **1.4.7. Wishlist**

* Users can add products to a wishlist from the product details or listing page.
* Wishlist is saved to the user’s account and accessible from their profile.
* Option to remove individual products from the wishlist.
* Ability to move products from the wishlist to the cart for purchase**.**

### **1.4.8. Admin Dashboard (Analytics)**

* Displays system overview: number of users, orders, available inventory, and revenue.
* Shows bar, line, or pie charts for orders by day, week, or month.
* Lists the top 5 best-selling products for the month.
* Alerts for low inventory levels.

### **1.4.9. Notification System**

* Notifications appear as pop-ups or icons in the navigation bar.
* When users add an item into wishlist.
* When user add an item into cart.
* When users modify item quatitis in cart.
* When user remove items from cart.
* Users can read or confirm received notifications.

### **1.4.10. Review Management (Admin)**

* Admins can view all customer reviews for products.
* Filter reviews by time, product, or star rating.
* Delete or hide reviews that violate policies, are spam, or misleading.
* Option for admins to respond to reviews (if supported).

### **1.4.11. External Payment Integration**

* Supports integration with payment gateways like VNPay, Momo, or PayPal. (not operate)
* Users can select a payment method during checkout.

### **1.4.12. Admin Features**

* Admins can add, edit, or delete products (including name, description, price, images, and stock quantity).
* Approve or update order statuses: pending, shipping, delivered, or canceled.
* Manage customer accounts: lock/unlock accounts and view user details.
* Manage reviews: delete inappropriate reviews, filter by product, rating, or date.
* View system analytics, including daily/monthly revenue, top-selling products, and system traffic.

## 1.5. Non-functional Requirements

* System response time under 3 seconds for key actions like product search, adding to cart, and checkout.
* All user passwords are encrypted using a secure algorithm (e.g., bcrypt), with HTTPS protocol and 2FA support.
* User interface functions reliably and displays correctly on major browsers (Chrome, Firefox, Edge, Safari).
* Responsive design compatible with screen resolutions from 320px upward on desktop, tablet, and mobile devices.
* All user interactions and actions are logged for auditing, testing, and error analysis.
* System architecture is flexible, supporting future module or feature expansions (e.g., additional payment gateways or promotion management).
* Interface and codebase adhere to standards for automated testing with tools like Selenium/WebDriver.

# **2. TEST OBJECTIVES**

* **Verify all functional requirements (FR):** Ensure that each functional requirement from the SRS document is fully tested through specific test cases.
* **Ensure system performance and scalability:** Measure the system’s ability to handle concurrent users, response speed, and stable operation under real-world conditions.
* **Ensure stability and reliability:** The system must not encounter crashes or critical errors during the testing of both primary and secondary usage flows.
* **Detect and address issues early:** Save time and reduce costs in later stages by identifying and fixing logical, security, or integration errors as early as possible.
* **Test usability:** Ensure the user interface is friendly, easy to understand, simple to operate, and consistent across various devices and browsers.
* **Test security:** Verify that the system is free from common vulnerabilities such as SQL injection, XSS, and CSRF. Ensure the safety of user login and transaction data.
* **Ensure cross-platform compatibility:** The system should run stably and display correctly on major browsers (Chrome, Firefox, Edge, Safari) and common mobile devices.
* **Ensure integration and maintainability:** Test the integration between the frontend and backend to ensure data consistency across modules. The system structure should support future expansion and maintenance.

# **3. TEST STRATEGY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Objective** | **Testing Techniques** | **Applicable Standards** | **Responsible Roles** | **Testing Approach** | **Exception Handling** |
| **Functional Testing** | Black-box testing, Integration testing | ISO/IEC 25010 | Tester, QA | Write test cases, execute manually or automatically | Log bugs, fix code, retest |
| **Security Testing** | Penetration testing, Privilege testing | OWASP Top 10 | Security Expert, Ethical Hacker | Simulate attacks, test data encryption | Patch vulnerabilities, update security policies |
| **Performance Testing** | Load, stress, and scalability testing | IEEE 829, ISO/IEC 9126 | Tester, DevOps | Simulate users, run JMeter to assess performance | Optimise algorithms, upgrade hardware if necessary |
| **Regression Testing** | Re-run test cases, automation testing | ISO/IEC 29119 | Tester, QA | Execute using Selenium, JUnit, TestNG | Log new issues, retest |
| **UI/UX Testing** | Usability testing, A/B testing | WCAG, ISO 9241 | UX Designer, Tester | Collect user feedback, measure interaction time | Improve interface layout, structure, colours |
| **API Testing** | API integration testing | RESTful, SOAP, OpenAPI | Developer, Tester | Use Postman, Swagger to test request/response | Fix API errors, enhance error messages |
| **Compatibility Testing** | Browser and mobile device compatibility | W3C, HTML5, CSS3 | Tester, Developer | Test on Chrome, Firefox, Safari, and mobile devices | Fix display issues, ensure cross-browser compatibility |
| **Data Testing** | Data integrity, large volume testing | ACID, ISO/IEC 27001 | DBA, Tester | Execute SQL scripts, check data tables | Optimise queries, handle data processing issues |

# **4. TESTING TOOLS AND ENVIRONMENT**

## 4.1. Testing Environment

|  |  |  |
| --- | --- | --- |
| **Category** | **Requirement / Tool** | **Description / Notes** |
| **Hardware** | **Server** | High-performance server capable of handling multiple requests per second for performance testing. |
| **PC/Laptop** | Main device used for writing test cases, executing tests, and accessing the test environment. |
| **Mobile Devices (Android/iOS)** | Test responsive features and compatibility on real devices. |
| **Software** | **Operating Systems (Windows, Linux, macOS)** | Set up a cross-platform testing environment. |
| **Browsers (Chrome, Firefox, Edge, Safari)** | Test user interface and browser compatibility. |
| **Automated Testing Tool (Selenium)** | Automate UI testing and support regression testing. |
| **Performance Testing Tool (JMeter)** | Simulate concurrent users and evaluate system performance. |
| **Security Testing Tool (OWASP ZAP)** | Scan and detect vulnerabilities such as SQL injection, XSS, etc. |
| **Database (SQL Server 2019)** | Testing data and environment restored from clean .bacpac or .sql backup files. |
| **Environment** | **Development Environment** | Where developers write code and fix bugs directly. |
| **Testing/Staging Environment** | Simulated environment used for full-scale system testing. |

## 4.2. Testing Tools

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity** | **Most Common Tool** | **Provider** | **Version (Reference)** |
| **Functional Testing** | Selenium | Selenium Project | 4.x |
| **Performance Testing** | JMeter | Apache | 5.x |
| **Security Testing** | OWASP ZAP | OWASP | 2.x |
| **Load & Scalability Testing** | K6 | Grafana Labs | 0.x |
| **UI/UX Testing** | Hotjar | Hotjar Ltd. | Cloud-based |
| **API Testing** | Postman | Postman Inc. | 10.x |
| **Regression Testing** | TestNG | Open-source | 7.x |

# **5. PRODUCT ACCEPTANCE CRITERIA**

## 5.1. Exit Criteria

|  |  |
| --- | --- |
| **Exit Criteria** | **Detailed Description** |
| **Critical Errors** | The system encounters blocker errors such as crashes or complete inaccessibility of key functionalities. |
| **Environment Not Ready** | Server is down, SQL errors, or testing devices are not operational. |
| **Unresolved Blocker Bugs** | Development team has not fixed bugs affecting core functionalities. |
| **Change in Test Scope** | Requirements from clients/instructors cause changes to core features. |

## 5.2. Completion Criteria

|  |  |
| --- | --- |
| **Evaluation Criteria** | **Detailed Description** |
| **Test Case Pass Rate** | ≥ 95% of total test cases are executed successfully. |
| **No Critical Bugs** | No remaining Severity 1 or Severity 2 bugs at the time of delivery. |
| **Performance** | Pages respond in less than 3 seconds for core actions. |
| **Functional Testing** | 100% of core functionalities are covered by both happy and negative path test cases. |
| **Security** | No critical security vulnerabilities detected by OWASP ZAP. |
| **User Testing** | A user can complete one shopping transaction without encountering any errors. |

# **6. RESOURCE ALLOCATION**

|  |  |  |
| --- | --- | --- |
| **Role** | **Responsibilities** | **Required Skills** |
| **Test Manager** | Create the test plan, manage testing resources | Leadership, project management, test planning |
| **QA Lead** | Coordinate the testing team, ensure testing progress | Testing experience, team management, defect reporting |
| **Tester / QA** | Write test cases, perform testing, log defects | Software testing, test case writing, analytical skills |
| **Automation Tester** | Write automated test scripts, run test automation | Selenium, TestNG, JUnit, Python/Java, CI/CD |
| **Performance Tester** | Conduct performance and load testing | JMeter, LoadRunner, Gatling, load testing |
| **Security Tester** | Identify security vulnerabilities, conduct penetration testing | OWASP, Ethical Hacking, Burp Suite, Kali Linux |
| **UI/UX Tester** | Evaluate user experience and usability | Interface testing, A/B testing, user behaviour analysis |
| **Database Tester** | Check data integrity, perform SQL query testing | SQL, NoSQL, ACID principles, large-scale data testing (Big Data) |

# **7. TESTING SCHEDULE**

## 7.1. Test Milestones

|  |  |
| --- | --- |
| **Phase** | **Milestone Date** |
| **Requirement analysis and test plan writing** | 23/04 |
| **Test case writing and review** | 24–25/04 |
| **Functional testing and initial bug logging** | 26/04 |
| **UI, security, and performance testing** | 27–28/04 |
| **Bug review meeting and regression testing** | 29/04 |
| **Final test report writing** | 02/05 |

## 7.2. Detailed Test Plan

|  |  |  |
| --- | --- | --- |
| **Phase** | **Date** | **Testing Activity** |
| **Requirement Analysis** | 23/04 | Review SRS/SDS, define scope and testing strategy |
| **Environment Setup** | 23–24/04 | Install Visual Studio, SQL Server, restore database, configure IIS/Selenium |
| **Test Case Writing** | 24/04 | Write detailed test cases (happy and negative paths) for FR/NFR |
| **Functional Testing** | 25–26/04 | Test registration, login, product search, add to cart, checkout |
| **Role-Based Testing** | 27/04 | Log in with various roles, verify role-specific access and interfaces |
| **UI Testing** | 28/04 | Responsive UI testing: desktop, mobile, multiple browsers |
| **Performance Testing** | 29/04 | JMeter test with >100 users, evaluate key API response time |
| **Security Testing** | 29/04 | Use OWASP ZAP: test for SQLi, XSS, CSRF, timeout, session hijacking |
| **Integration Testing** | 30/04 | Test cart → payment → invoice → revenue report flow |
| **Regression Testing** | 01/05 | Retest fixed bugs and affected flows |
| **Summary & Reporting** | 02/05 | Compile test results, bug logs, and provide final report and handover recommendations |

# **8. RISKS AND ASSUMPTIONS**

|  |  |  |
| --- | --- | --- |
| **Risk** | **Level** | **Mitigation Strategy** |
| **Inconsistent data between UI and database** | High | Periodic cross-check between frontend and backend; log all transactions |
| **UI display issues on small devices or outdated browsers** | Medium | Perform responsive design testing; support multi-browser with appropriate breakpoints |
| **Errors during database backup restoration** | High | Verify file format (.bacpac), perform daily backups before each testing cycle |
| **Unavailable or changing personnel during the project** | Medium | Clear role assignment, include detailed instructions in test cases, apply cross-training |
| **Insufficient time for regression testing** | High | Prioritise testing core functionalities; use automation to assist regression testing |
| **Test cases not covering all scenarios** | Medium | Regularly review and update test cases based on actual testing feedback |

# **9. DELIVERABLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Deliverable** | **Timeline / Week** | **Prepared By** | **Delivered To** |
| **Test Plan** | Week 1 | Test Lead | Project Manager (PM) |
| **Test Cases** | Week 2 – 3 | QA Team | Test Lead |
| **Test Scenarios** | Week 2 – 3 | QA Team | Test Lead |
| **Test Report** | Week 4 | Test Lead | PM |
| **Bug Report** | Week 4 | QA Team | Dev Team / PM |
| **Test Scripts** | Week 3 – 4 | Automation Tester | QA Lead |
| **Test Data** | Week 3 | QA Team | Dev / Test Lead |
| **User Acceptance Test Report (UAT)** | Week 4 – 5 | Test Lead | Client / PM |
| **Test Closure Report** | Week 5 | Test Lead | PM |