DAY 5 – Testing and Backend Refinement – [MORENT]

Marketplace Testing, Error Handling, and Backend Integration Documentation

Prepared by: Iqra Abdul Qadir

Marketplace Builder Hackathon

MORENT- car rental

1. Introduction

This document outlines the testing procedures and results, performance optimization efforts, security measures implemented, and challenges faced during the marketplace project. The goal was to ensure the application is functional, secure, performant, and provides a seamless experience across browsers and devices. Testing covered functional, non-functional, user acceptance, and security aspects. Robust error handling and graceful API error fallbacks were also integrated to enhance user experience and system reliability.

2. Test Cases Executed and Their Results

2.1 Functional Testing

- **User Registration and Login**
  Test Case: Verify new user registration and login functionality.
  Result: *Passed* – Users were able to register, receive verification emails, and log in without issues.
- **Product Listing and Search**
  Test Case: Verify that products are correctly listed and searchable by keywords and filters.
  Result: *Passed* – Search functionality returned relevant results with appropriate filtering.
- **Checkout Process**
  Test Case: Ensure that the checkout process, including cart management, payment processing, and order confirmation, functions correctly.
  Result: *Passed* – The checkout flow worked as expected; error messages were displayed when invalid payment details were entered.

2.2 Non-Functional Testing

- **Performance Testing**
  Test Case: Measure response times under varying load conditions.
  Result: *Passed* – Response times remained within acceptable limits (sub-2 second latency) after optimization steps.
- **Cross-Browser Compatibility**
  Test Case: Verify that the application functions across multiple browsers (Chrome, Firefox, Edge, Safari).
  Result: *Passed* – Minor CSS adjustments were made to ensure consistency; no critical issues found.
- **Device Responsiveness**
  Test Case: Ensure that the marketplace layout adapts well to mobile devices, tablets, and desktops.
  Result: *Passed* – Responsive design principles were successfully applied; UI elements scaled appropriately across devices.

2.3 User Acceptance Testing (UAT)

- **User Feedback on Usability**
  Test Case: Collect feedback from a group of target users regarding the ease of navigation, clarity of UI messages, and overall experience.
  Result: *Passed* – Users reported a positive experience with clear instructions and responsive design.

2.4 Security Testing

- **Vulnerability Scanning**
  Test Case: Run automated tools to identify common vulnerabilities (SQL injection, XSS, CSRF).
  Result: *Passed* – No critical vulnerabilities were detected; additional measures are in place to monitor potential threats.
- **Error Handling & Fallbacks**
  Test Case: Simulate API errors and test the UI’s fallback messages.
  Result: *Passed* – Clear, user-friendly error messages were displayed, and logs were appropriately captured.

2.5 CSV-Based Test Report

Below is a sample excerpt of the CSV-based test report generated during testing. The full report as been included in the same folder as this document.

```
TestCaseID,TestDescription,ExpectedResult,ActualResult,Status,Comments
TC001,User Registration and Login,Successful registration and login,User
registered and logged in,Pass,Verification email sent successfully.
TC002,Product Search,Display matching products,List of products displayed
correctly,Pass,Minor latency observed during peak load.
```

```
TC003,Checkout Process,Successful checkout with valid inputs,Order processed
with valid payment,Pass,Error displayed for invalid payment details.
TC004,Cross-Browser Compatibility,Consistent UI across browsers,Minor CSS
differences observed,Pass,Resolved with browser-specific fixes.
TC005,API Error Handling,Display clear error message on API failure,Fallback
UI displayed,Pass,Error logs captured.
```

---

3. Performance Optimization Steps

- **Code Refactoring:**
  Optimized backend queries and front-end scripts to reduce load times. Lazy loading techniques were implemented for images and other media.
- **Caching Strategies:**
  Implemented server-side and client-side caching to reduce repetitive API calls and improve response time.
- **Minification & Compression:**
  All CSS, JavaScript, and HTML files were minified. Gzip compression was enabled on the server to reduce the size of data transmitted.
- **Database Optimization:**
  Indexed frequently queried fields in the database and optimized SQL queries to reduce response times under load.
- **Load Testing and Benchmarking:**
  Used performance testing tools to simulate high traffic and adjusted server resources accordingly to ensure the application remains responsive.

---

4. Security Measures Implemented

- **Input Validation & Sanitization:**
  Ensured all user inputs were validated and sanitized to prevent SQL injection and cross-site scripting (XSS) attacks.
- **Robust Error Handling:**
  Implemented try-catch blocks around API calls and critical operations. Fallback UI elements were provided to inform users gracefully in case of failures, while detailed error logs were recorded for backend monitoring.
- **HTTPS Enforcement:**
  Configured SSL/TLS to secure data in transit between the client and server.
- **Authentication and Authorization:**
  Utilized token-based authentication (JWT) to manage user sessions securely. Access control measures were strictly enforced on sensitive endpoints.
- **Regular Security Scans:**
  Integrated automated vulnerability scanning tools into the CI/CD pipeline to catch and address potential security issues early.

---

5. Challenges Faced and Resolutions Applied

- **Challenge:** *Cross-Browser Inconsistencies*
  Issue: Minor UI inconsistencies were observed between browsers, affecting the overall look and feel of the marketplace.
  Resolution: Adjusted CSS rules and implemented browser-specific prefixes to ensure consistent rendering across all supported browsers.
- **Challenge:** *API Downtime and Fallback Handling*
  Issue: Simulated API failures exposed gaps in the initial error handling, resulting in unclear user feedback.
  Resolution: Developed comprehensive fallback UI components that display clear, user-friendly messages. Detailed error logs were implemented to track the issues for further analysis.
- **Challenge:** *Performance Under Load*
  Issue: Initial load testing revealed performance bottlenecks during high traffic periods.
  Resolution: Introduced caching strategies, optimized database queries, and refactored inefficient code. Subsequent load testing confirmed significant performance improvements.
- **Challenge:** *User Acceptance Feedback*
  Issue: Some users reported difficulties in navigating the checkout process.
  Resolution: Simplified the checkout flow by reducing the number of steps and enhancing UI clarity based on user feedback. Additional tooltips and inline error messages were added.

---

6. Conclusion

The comprehensive testing and integration phase have ensured that the marketplace application meets functional and non-functional requirements, is secure, performs well under load, and provides a seamless user experience across different devices and browsers. Continuous monitoring, regular performance reviews, and iterative improvements are planned to maintain the application's high standards.

---

7. Appendix

CSV Test Report Overview:
For detailed logs and test results, please refer to the attached CSV report. The CSV file includes all test cases, expected outcomes, actual results, status (Pass/Fail), and additional comments for each scenario. Also, functional deliverables have been included in a corresponding folder.
