TEST SUMMARY REPORT

Project Name

AutomationExercise – E-commerce Application Testing

Testing Type

Manual Testing + Basic Automation Testing

Test Environment

- Browser: Google Chrome
- OS: Windows 11
- Automation Tool: Selenium WebDriver
- Language: Java

1. Test Case Execution Summary

Total Test Cases	Executed	Passed	Failed	Not Executed
44	44	33	11	0

Summary:
All planned test cases were executed. Most core functionalities worked as expected. Some test cases failed due to functional and UI-related issues.

2. Defect Density

Formula:
Defect Density = Total Defects / Total Test Cases

Calculation:
- Total Test Cases: 44
- Total Defects: 11

Defect Density = 11 / 44 = 0.25

Interpretation:
The defect density indicates a moderate number of defects, suggesting improvements are required before final release.

3. Pass / Fail Percentage

Pass Percentage:
(33 / 44) × 100 = **75%**

Fail Percentage:
(11 / 44) × 100 = **25%**

4. Final Test Closure Report

- All 44 test cases were executed successfully

- Functional areas tested:
  - Login & Logout
  - Product Search
  - Product Details
  - Add to Cart
  - Checkout & Place Order

- 11 defects were identified and reported
- Automation scripts covered major user journeys
- Application is **partially stable** and requires fixes for reported defects

Test Status: CLOSED ✅ (with defects reported)

PDCA Cycle Implementation

The PDCA (Plan–Do–Check–Act) cycle was followed throughout the testing lifecycle to ensure continuous quality improvement of the application.

PLAN (Test Planning & Strategy)

In the planning phase, the testing approach and strategy were clearly defined before starting execution.

- Understood the application requirements and core business flows such as login, product search, cart, checkout, and logout
- Identified the scope of testing, focusing on functional and UI validation
- Prepared a detailed **Test Plan** outlining:
  - Test objectives
  - Test scope and features to be tested
  - Test environment (browser, OS, tools)
  - Entry and exit criteria
- Designed **44 manual test cases** covering positive and negative scenarios
- Selected **Selenium WebDriver with Java** for basic automation testing
- Identified key user journeys for automation, such as login, search, add to cart, checkout, and logout

Outcome:
A clear roadmap was created to ensure systematic and effective testing.

DO (Test Execution)

In this phase, the planned test cases were executed as per the test strategy.

- Executed all **44 manual test cases** in the defined test environment
- Performed functional testing on major modules:
  - Authentication
  - Product listing and search
  - Cart management
  - Checkout and order placement
- Logged **11 defects** with proper details including steps to reproduce, expected result, actual result, severity, and priority
- Developed and executed automation scripts for critical flows using Selenium and Java

- Captured execution screenshots and console output for reference

Outcome:
Test execution was completed successfully, and defects were identified and documented for improvement.

CHECK

In the check phase, the results of test execution were analyzed to assess application quality.

- Reviewed test execution results and categorized test cases as pass or fail
- Calculated key test metrics:
    - Pass percentage: 75%
    - Fail percentage: 25%
    - Defect density: 0.25
- Analyzed defect patterns to identify common issues such as:
    - UI inconsistencies
    - Validation gaps
    - Pop-up and ad-related interruptions
- Assessed the impact of defects on user experience and business flow

Outcome:

Clear visibility into application stability and areas that need improvement.

ACT (Improvements for Next Release)

Based on the analysis, corrective and preventive actions were proposed for the next release.

- Recommended fixing critical and high-impact defects before production release
- Suggested improving input validations, especially in checkout and payment-related flows
- Proposed handling dynamic pop-ups and ads more effectively to improve automation stability
- Recommended increasing automation coverage for regression testing
- Suggested improving UI consistency and error messaging for better user experience

Outcome:

Actionable improvements were identified to enhance application quality in future releases.

Overall Conclusion

By following the PDCA cycle, the testing process ensured structured planning, effective execution, proper analysis, and continuous improvement. This approach helped in identifying defects early and provided clear recommendations to improve the application quality in the next release.