FINAL DOCUMENTATION

Comprehensive Final Report for AutomationExercise Website

Project Group: G1_DEPI
Project Domain: Software Testing
Submitted To: [DEPI/AMIT]

TEAM MEMBERS:

1. Belal Mohammed Taher - Tester
2. Mohammed Abedel Hamed - Tester
3. Mohammed abo El Yazeed - Tester
4. Ahmed Salah Qatar - Tester
5. Marian Ghaly - Tester

Submission Date: 11/13/2025

==
==

TABLE OF CONTENTS

1.0 TEAM ROLES & RESPONSIBILITIES

MARIAN GHALY - Tester
• Primary Responsibility: Login API Testing
• API Focus: Valid/invalid credentials status codes & messages
• Tools: Postman, Manual Testing

AHMED SALAH QATER - Tester
• Primary Responsibility: User Account API Testing
• API Focus: Create, Delete, Update, Get user account
• Tools: Postman, Manual Testing

MOHAMMED ABEDEL HAMED - Tester
• Primary Responsibility: Product Listing API Testing
• API Focus: GET all Products, POST all Products List
• Tools: Postman, Manual Testing

BELAL MOHAMMED TAHER - Tester
• Primary Responsibility: Brand Listing API Testing
• API Focus: GET all Brands, PUT all Brand List
• Tools: Postman, Manual Testing

MOHAMMED ABO EL YAZEED - Tester
• Primary Responsibility: Authentication & User Management UI Testing
• Focus: User registration, login, logout, account management
• Tools: Manual Testing, Automation Support

2.0 EXECUTIVE SUMMARY

This document represents the final comprehensive report for the graduation project in Software Testing. The project involved creating and executing a complete test strategy for AutomationExercise.com. Our team systematically validated the application across UI, API, and Database layers, ensuring functional reliability and data integrity. We identified 27 defects, implemented 26 automated scenarios, and established a robust testing framework.

Key Achievements:
• 140+ test cases executed with 95% requirements coverage
• 26+ automated regression scenarios implemented
• 27 defects identified and documented with severity analysis
• Multi-layered testing strategy (UI, API, Database)
• Comprehensive tool stack implementation (Jira, Postman, Selenium, MySQL)

3.0 PROJECT OVERVIEW

The AutomationExercise website is a full-featured e-commerce platform. Project objective was to deliver a robust testing framework validating front-end user experience and back-end system integrity while establishing automated regression tests.

In-Scope Features:

- UI Layer: User registration/signup, login/logout, product listing & detail, search, categories/brands, cart operations, checkout, subscription, contact forms
- API/Service Layer: REST endpoints supporting the UI flows (signup API, login API, product list API, cart API, order submission API)
- Database Layer: Verifying data integrity for users, orders, cart items, subscriptions
- Automation Testing: Building automated scripts covering key flows for regression testing

Out-of-Scope:

- Performance testing under heavy load
- Integration with third-party payment gateways
- Administrative/back-office interfaces
- Mobile app testing
- Specialized accessibility testing

4.0 TEST STRATEGY & PLAN

TEST OBJECTIVES:

• Verify UI flows operate correctly for positive and negative scenarios
• Validate API endpoints with correct status codes and response payloads
• Ensure database integrity and data constraints enforcement
• Develop automated tests to reduce manual regression effort
• Enable regression coverage across UI, API and DB layers

TESTING LEVELS PERFORMED:

• Unit Testing: Basic component/logic level tests (handled by developers)
• API/Service Testing: REST endpoints testing for correctness, status codes, payloads, error handling
• Database Testing: Validate correct storage and retrieval of data, schema integrity, constraints
• Functional/UI Testing: Verification of end-to-end user flows through the UI
• Automation Testing: Automated execution of UI + API + DB tests for regression
• User Acceptance Testing: Business stakeholder validation from UI perspective

DEFECT MANAGEMENT:

Severity levels: Critical, High, Medium, Low with defined impact criteria
• Critical: Functionality blocked, no testing can proceed
• High: Functionality not usable, no workaround available
• Medium: Functionality issues with workaround available
• Low: Cosmetic errors with minimum impact on product use

5.0 TEAM CONTRIBUTIONS & TEST ARTIFACTS

5.1 API Testing - Login & Authentication - Marian Ghaly

- Tested: Login API with valid/invalid credentials
- Focus: Status codes and error message validation
- Verified: API response accuracy for authentication scenarios
- Key Findings: Proper status codes returned for valid/invalid login attempts

5.2 API Testing - User Account Management - Ahmed Salah Qater

- Tested: User Account API (Create, Delete, Update, Get operations)
- Focus: CRUD operations functionality and data integrity
- Verified: User management workflows and API endpoints
- Key Findings: API properly handles user account lifecycle management

5.3 API Testing - Product Catalog - Mohammed Abedel Hamed

- Tested: Product listing API (GET all Products, POST operations)
- Focus: Product data retrieval and management
- Verified: Product catalog integrity and API endpoints
- Key Findings: API successfully handles product listing and management operations

5.4 API Testing - Brand Management - Belal Mohammed Taher

- Tested: Brand Listing API (GET all Brands, PUT operations)
- Focus: Brand data management and updates
- Verified: Brand catalog functionality and API operations
- Key Findings: API properly manages brand information and updates

5.5 UI Testing - Authentication & User Management - Mohammed abo El Yazeed

- Tested: User registration, login, logout, account management (UI)
- Found: 5 defects including password validation issues
- Automated: Login/logout scenarios
- Key Findings: System allows weak passwords, accepts invalid email formats

6.0 DEFECT ANALYSIS & QUALITY METRICS

DEFECT SUMMARY:
Total Defects Identified: 27

SEVERITY DISTRIBUTION:
• Critical: 4 defects (15%) - Payment security vulnerabilities
• High: 10 defects (37%) - Major functional and validation issues
• Medium: 8 defects (30%) - Feature gaps and usability problems
• Low: 5 defects (18%) - Cosmetic and minor issues

MODULE-WISE DEFECT DISTRIBUTION:
• Checkout & Payment: 8 defects
• Authentication: 5 defects
• Product Listing: 3 defects
• Cart Operations: 2 defects
• API Layer: 1 defect
• Auxiliary Features: 2 defects

TEST COVERAGE METRICS:
• Requirements Coverage: 95%
• UI Test Cases Executed: 140+
• API Test Cases Executed: 15+
• Automation Coverage: 26 core scenarios

KEY QUALITY RISKS IDENTIFIED:
1. Payment Security: Complete lack of validation in payment system
2. Data Integrity: Missing input validation on forms leading to data corruption
3. Feature Reliability: Broken filters misleading users
4. Security: Sensitive data exposure and weak authentication

7.0 TOOLS & TECHNOLOGIES STACK

7.1 Test Management & Defect Tracking (Jira)

Our team utilized Jira for comprehensive test management throughout the project lifecycle. We created custom workflows for defect tracking, test case management by modules, and real-time dashboards for progress monitoring. The tool enabled efficient sprint planning and team collaboration with detailed defect reporting.

7.2 API Testing Framework (Postman)

We developed organized Postman collections for systematic API testing with multiple environments (dev, test, staging). Collections included Authentication, User Management, Products, and Cart & Orders with comprehensive positive and negative test scenarios using environment variables and data-driven testing approaches.

7.3 Test Automation Framework (Java + Selenium + TestNG)

Our automation framework was built using Java with Selenium WebDriver and TestNG, implementing Page Object Model design pattern for maintainable code. The framework supported cross-browser testing, comprehensive reporting, and was designed for future CI/CD integration with parallel execution capabilities.

7.4 Database Testing Strategy (MySQL Workbench + XAMPP)

For comprehensive data validation, we implemented database testing using XAMPP local server with MySQL and MySQL Workbench. We created SQL scripts for data integrity verification, schema validation, and test data management, ensuring consistent test execution across UI, API, and database layers.

7.5 Test Environment Setup

Our testing was conducted across integrated environments including the live AutomationExercise website for UI testing, dedicated API endpoints, and local OpenCart installation on XAMPP for database testing. We supported multiple browsers (Chrome, Firefox, Edge) with isolated test data strategy.

8.0 TESTING FRAMEWORK ARCHITECTURE

8.1 Automation Framework Design

Our automation framework followed a hierarchical structure with Configuration Layer for environment settings, Page Object Layer for reusable page classes, Test Logic Layer for scenario implementation, Utilities Layer for helper functions, and Execution Layer for test suite organization and parallel execution using TestNG.

8.2 API Testing Methodology

Our API testing approach included functional testing with valid and invalid inputs, data-driven testing for comprehensive coverage, integration testing with UI and database layers, security testing for authentication and authorization, and basic performance testing for critical API response time validation.

8.3 Database Testing Approach

Database testing was integrated throughout the testing lifecycle with direct SQL validation queries, data mapping verification between UI fields and database columns, constraint testing for unique and foreign key validations, and transaction testing to verify ACID properties for critical operations like order processing.

8.4 Integration Between Testing Layers

We established comprehensive traceability between different testing types including UI-API integration to correlate actions with API calls, API-DB integration to verify operations result in correct database changes, end-to-end validation for complete workflows, and systematic defect triaging to identify whether issues originated in UI, API, or database layers.

9.0 CONCLUSION & LESSONS LEARNED

PROJECT SUCCESS:
Successfully validated AutomationExercise website with comprehensive testing approach. Identified critical security issues and functional gaps while delivering robust automation framework and detailed quality assessment.

KEY FINDINGS:
• Payment system lacks validation (Critical)
• Input validation missing on forms (High)
• Filtering logic inconsistent (Medium)
• Security vulnerabilities in data handling (Critical)

RECOMMENDATIONS:
1. Fix payment security vulnerabilities immediately
2. Implement robust input validation on all forms
3. Complete missing features (Forgot Password, User Settings)
4. Enhance filtering and search functionality

FINAL ASSESSMENT:
The AutomationExercise website demonstrates solid core functionality for basic e-commerce operations but requires significant improvements in security, input validation, and error handling to be production-ready. The testing process successfully identified critical gaps and provides a clear roadmap for quality enhancement.

LESSONS LEARNED:
• Security testing should be integrated from project inception
• Automation framework pays off in regression testing efficiency
• Comprehensive tool stack enables thorough testing coverage
• Team collaboration and clear role definition are crucial for success

10.0 APPENDICES

Appendix A: Test Plan Document
Location: Project Repository/TestPlan_G1_DEPI.docx
Description: Complete test strategy with objectives, scope, approach, and detailed testing methodology

Appendix B: Test Cases Documentation
Location: Project Repository/E-Commerce_AutomationExercise_Manual_Testing.xlsx
Description: 140+ manual test cases across all modules with detailed steps, expected results, and actual results

Appendix C: Defect Reports
Location: Jira Project Board
Description: 27 defect reports with severity classification, status tracking, resolution details, and team assignments

Appendix D: Automation Framework
Location: Java Project/IntelliJ
Description: Selenium WebDriver framework with Page Object Model design, TestNG integration, and comprehensive test suites

Appendix E: API Testing Collections
Location: Postman Workspace
Description: Organized collections for all API endpoints with environment configurations and test scripts

Appendix F: Database Testing Scripts
Location: MySQL Workbench
Description: SQL queries for data validation, integrity checks, and test data management

***** END OF DOCUMENT *****