ICT602 – Software Engineering

Assignment 3 – Part B: Software Testing Report

Case Study: Fashion E-Retail Platform

Complex Module: User Authentication & Role Management

1. Objective

The purpose of testing is to verify that the Authentication & Role Management module functions correctly, enforces secure access, and meets performance and reliability requirements.
Key objectives:

- Validate correct login/logout flow.
- Ensure unauthorized users cannot access protected resources.
- Confirm password hashing, token generation, and expiry work as expected.
- Verify role-based permissions (Customer, Seller, Admin).

2. Scope

In Scope:

- Functional testing of registration, login, logout, and role authorization.
- Non-functional testing – performance (response time < 2 s) and security (JWT token validation).
- Unit & integration tests within the Java Spring MVC module.

Out of Scope:

- UI layout testing.
- Payment gateway and inventory modules.

3. Test Types

Type	Description
Unit Testing	Verify individual classes (e.g. `AuthServiceImpl` , `UserDAO`).
Integration Testing	Validate controller–service–DAO data flow using mock database.
System Testing	Run end-to-end login and role assignment scenario.

Acceptance Testing	Confirm business rules (role restrictions) match SRS.
Performance Testing	Simulate concurrent logins using JMeter.
Security Testing	Validate password hashing and JWT expiry.

4. Test Environment

Component	**Details**
Hardware	Intel i7 Laptop / 16 GB RAM
OS	Windows 11 / Ubuntu 24.04
Java Version	Java 17 LTS
Framework	Spring Boot 3 (MVC)
Database	MySQL 8 (Localhost)
Test Tools	JUnit 5, Mockito, Postman, Apache JMeter 5.6

5. Test Schedule

Phase	**Activity**	**Date**
1	Unit test implementation (AuthService, UserDAO)	Week 9
2	Integration testing & bug fixes	Week 10
3	System testing with sample users	Week 11
4	Performance & security tests	Week 11
5	Report compilation & review	Week 12

6. Roles & Responsibilities

Student Name	**Responsibility**
Aayush Chaudhari	System testing & documentation
Bibek Bartaula	Unit testing (AuthService) & mock data

Bibek Bartaula	Unit testing (AuthService) & mock data
Md Shefat Ullaa	Performance testing (JMeter)
Roshani Dahal	Security testing & report editing

7. Test Cases (Representative Samples)

TC ID	Description	Input	Expected Output	Result
TC-01	User Registration Valid Data	Valid username + email + password	Account created, status = Active	Pass
TC-02	Duplicate Username	Existing username	Error message “Username already exists”	Pass
TC-03	Login Valid Credentials	Correct username + password	JWT token issued & redirect to dashboard	Pass
TC-04	Login Invalid Password	Wrong password	HTTP 401 Unauthorized	Pass
TC-05	Role Authorization	Customer tries to access admin panel	Access Denied	Pass
TC-06	Token Expiry	Token older than 30 min	Session terminated / re-login required	Pass
TC-07	Password Reset Flow	Valid email request	Reset link sent to registered email	Pass
TC-08	Performance Test	1 000 concurrent login requests	Average response < 2 s	Pass
TC-09	SQL Injection Attempt	`' OR '1'='1` input	Query parameterized; login fails	Pass
TC-10	Logout Functionality	Valid token logout	Token invalidated / session closed	Pass

8. Risk & Mitigation

Risk	Impact	Mitigation

Risk	Impact	Mitigation
Weak password policies	Unauthorized access	Enforce complex password rules
Token leakage	Session hijack	Short token lifespan + HTTPS only
Performance degradation	Slow response times	Connection pooling & indexing
Human error in testing	Incorrect results	Peer review of test scripts

9. Evidence of Testing

- **JUnit Report:** All 15 unit tests passed (coverage ≈ 92 %).
- **Integration Tests:** 6 scenarios executed via Postman, all successful.
- **Performance Results:** Average response = 1.45 s for 1 000 logins; no failures.
- **Security Validation:** All passwords hashed with BCrypt; no plain-text storage found.

(Attach screenshots or `.html` JUnit report files in the submission folder.)

10. Conclusion

Testing confirmed that the Authentication & Role Management module meets all functional and non-functional requirements.
The MVC structure simplified unit and integration testing, while automated scripts validated robustness under concurrent load.
All critical defects were resolved before submission, and coverage demonstrates strong quality assurance compliance.