MEKELLE UNIVERSITY

School of Computing EIT-M
Department of Software Engineering

Software Testing and Quality Assurance (SENG5441)

FILIMON HAFTOM
UGR/177981/12
1. Test Planning

1.1 Overview

Objective: Ensure the platform meets quality, reliability, and user satisfaction by identifying and fixing defects while validating functionality, performance, and security.

1.2 Scope

Features to be tested:

i. User authentication and role-based access (students, teachers, admins).
ii. Student functionalities (course enrollment, payment processing, grade viewing, and receipt generation).
iii. Admin functionalities (teacher/course management, revenue tracking, and fee adjustments).
iv. Payment gateway integration (Stripe).
v. Performance requirements under concurrent user load (500 users).
vi. Security features (password hashing, secure payment processing).

Features not to be tested:

i. Mobile compatibility and extension.
ii. Large-scale database scalability beyond SQLite.

1.3 Test Objectives

a) Verify implementation of all functional and non-functional requirements.
b) Identify and fix usability issues in the UI.
c) Validate integration with Stripe and ensure compliance with security standards.
d) Ensure the platform is accessible and responsive.

1.4 Test Strategy

Testing Levels: Unit, integration, system, acceptance.
Testing Types: Functional, regression, performance, load, security, and usability testing.
Environment: Use Laravel, SQLite, Stripe sandbox API, and Tailwind CSS.
Tools: Pest, PHPUnit for unit tests, Postman for API testing

1.5 Deliverables

- ✓ Test cases/scripts.
- ✓ Test data sets for various scenarios.
- ✓ Test execution reports.
- ✓ Defect logs and tracking reports.
- ✓ Final summary report with metrics, findings, and recommendations.

1.6 Entry and Exit Criteria

Entry Criteria:

- ✓ Approved SRS and design documents.
- ✓ Development is complete for individual modules.
- ✓ Test environment setup is complete.

Exit Criteria:

- ✓ All test cases executed with results logged.
- ✓ All critical and high-priority defects fixed and re-tested.
- ✓ Acceptance criteria met.

1.7 Resources

Tools: Pest, PHPUnit, Postman, and TablePlus.

1.8 Risk Management

Identified Risks:

- ✓ Stripe API downtime.
- ✓ Ambiguities in feature requirements.

Mitigation Strategies:

- ✓ Use sandbox testing for Stripe.
- ✓ Establish continuous communication with stakeholders.

2. Test Analysis and Design

2.1 Test Analysis

Objectives:

- Derive comprehensive test conditions.
- Ensure traceability between requirements and test cases.

Activities:

- Requirement Analysis: Analyze functional and non-functional requirements from the SRS.
- Test Basis Identification: Use SRS, use cases, and design documents.
- Derive Test Conditions: Identify conditions for each feature, including edge cases.
- Prioritization: Rank test conditions based on importance and risk.

2.2 Test Design

Objectives:

- Create detailed test cases with clear steps and expected results.
- Prepare reusable test data for diverse scenarios.
- Design test environments replicating production setups.

Activities:

1. Test Case Design:

- ➢ ***Login Functionality***

ID: TC001
Description: Validate user login with valid credentials.
Preconditions: User account exists.
Steps:

1. Open the login page.
2. Enter valid email and password.
3. Click "Login".

Expected Result: Redirect to the appropriate dashboard.
Priority: High

- ➢ ***Stripe Payment Processing***

ID: TC010

Description: Verify successful payment processing.
Preconditions: User enrolled in a course, valid card details available.
Steps:

1. Go to the payment page.
2. Enter valid card details.
3. Submit payment.

Expected Result: Payment succeeds; receipt generated.
Priority: High

2. Test Data Design:

- Valid and invalid email/password combinations for login.
- Simulated Stripe API responses (success, failure, timeout).

3. Test Environment Setup:

- PHP 8.1+, Laravel framework, SQLite database.
- Enable Stripe sandbox mode.

4. Test Automation Design:

Use Postman for Stripe API automation.
Deliverables:

- Detailed test cases for all features and edge cases.
- Test execution logs and defect tracking reports.

3. Final Report

Draft a comprehensive testing summary report including pass/fail statistics, defect trends, and recommendations for improvements.