Test Strategy Document for Vendor Portal

# Scope And Overview

## Scope:

This Test Strategy document outlines the approach and methodology for testing the Vendor Portal of the Online Market Place (OMP) application. The Vendor Portal is a crucial component of the OMP system and is responsible for vendor onboarding, store management, product catalogue, pricing, and order processing.

## Overview:

The testing objective is to ensure the Vendor Portal functions correctly, securely, and is user-friendly. This document provides a comprehensive overview of the testing approach, test types, test environment setup, tools, release control, risk analysis, and review and approval processes.

# Test Approach

## Unit Testing:

* Conducted by developers during the development phase to ensure individual unit of testable code is tested.
* To be initiated at the start of development and continue in parallel during development.
* - Automation strategy: Developers are responsible for writing unit tests using unittest library.

Integration Testing:

* Verifies the interaction between different modules and components of the Vendor Portal.
* Begins after unit testing and continues throughout the development phase.
* Automation strategy: Integration tests will be automated using unittest.mock library.

## System Testing:

* Validates the end-to-end functionality of the Vendor Portal.
* Starts after integration testing and progresses as new features are developed.
* Automation strategy: Automation scripts will be created to execute and validate critical workflows by UI using selenium or API using requests library respectively.

Regression Testing:

* Ensures that new updates do not introduce defects into existing functionality.
* Runs continuously alongside development and system testing.
* Automation strategy: Regression test suites will be automated to ensure quick validation of existing features.

## Usability Testing:

* Assesses the user-friendliness of the Vendor Portal.
* To be conducted after system testing, involving real vendors.
* No automation; usability experts will observe and collect feedback.

## Load and Performance Testing:

* Evaluates the system's ability to handle expected load and performance benchmarks.
* To be initiated in the later stages of development once System tests validate full implementation of functionality.
* Automation strategy: Performance testing will done using PyTest-benchmark .

Security Testing:

* Identifies and addresses vulnerabilities in the Vendor Portal.
* Conducted throughout development and prior to release.
* Automation strategy: Dynamic Application Security Testing tools like OWASP ZAP will be employed alongside static code analysis tool like pylint.

## Defect Management Cycle:

* Defects will be logged in Jira.
* Jira will be configured to display fields as per defect template provided below.
* Defect status will be updated throughout the testing process.
* Defect triage meetings will be held regularly to prioritize and assign defects.
* Testers will perform re-testing after developers fix defects.
* Regression testing will include verifying fixed defects.
* Test sign-off will occur once all critical defects are resolved.

### Defect Template

Project: Online Market Place (OMP)

Module: Vendor Portal

Tester: [Tester Name]

Date: [Date]

**Defect Details**

Defect ID: [Unique Defect ID]

Severity: [Low / Medium / High / Critical]

Priority: [Low / Medium / High / Critical]

Status: [Open / In Progress / Fixed / Reopened / Closed]

Defect Type: [Functional / Usability / Performance / Security / Others]

Reported By: [Tester's Name]

Reported Date: [Date]

**Summary**

Summary: [A brief one-line summary of the defect]

**Description**

Description: [Detailed description of the defect, including steps to reproduce, actual behaviour, and expected behaviour.]

**Attachments**

Attachments: [If applicable, include screenshots, logs, or any relevant files]

**Environment Details**

Environment: [Specify the test environment, including hardware, software, browsers, and versions]

Operating System: [Specify the OS, e.g., Windows 10, macOS]

Browser: [Specify the browser used, e.g., Chrome, Firefox]

Device: [Specify the device if applicable, e.g., Desktop, Mobile]

**Steps to Reproduce**

**Actual Behaviour**

**Expected Behaviour**

**Defect Analysis**

Root Cause: [Specify the root cause of the defect, if known]

Impact Analysis: [Explain the potential impact of the defect on the system]

**Defect Resolution**

Resolution: [Describe how the defect was fixed]

Testing Status: [Provide the current testing status, e.g., retesting, verification]

Closed Date: [Date when the defect was closed, if applicable]

**Defect Review**

Reviewed By: [Name of the reviewer]

Review Date: [Date of the review]

Comments: [Any additional comments or notes from the reviewer]

# Test Environment

## Environments:

* Separate environments for Development, Testing, UAT, and Production.
* Environments should have similar configurations

## Test Data:

* Whenever possible test data on test environment will be a copy of masked production data.
* When consistent test is required or production data is unavailable, libraries like Library\_boy and Gaussian will be used.

# Testing Tools

## Test Management:

* Test management tool TestRail will be used for test case management, execution, and defect tracking with Jira Plugin.
* Every release will have a test plan with required test cases earmarked for various different test runs.
* All newly added tests for a newly created feature or modified tests will have release tag associated with them.

## Automation:

* Test Automated framework PyTest will be used to automate UI and API tests.

## Performance and Load Testing:

* Performance testing will be automated by PyTest-benchmark

## Security Testing:

* Dynamic Application Security Testing tools like OWASP ZAP will be employed alongside static code analysis tool like pylint.

# Release Control

* Docker containers will be used for deploying new builds to test environments.
* Automation code will be deployed in their own containers.

# Risk Analysis

- List of potential risks, including security vulnerabilities, performance bottlenecks, and usability issues.

- Mitigation plans for each identified risk.

- Contingency plans to address risks if they materialize.

# Review And Approvals

- All entities involved in project management, including business teams, development teams, system administrators, and quality assurance teams, will review and approve this Test Strategy document.

- Sign-off from stakeholders is required before testing commences.