# 

# Testing Strategy formation Plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Feb 17–23 | Feb 24 – Mar 2 | Mar 3–9 | Mar 10–16 | Mar 17–23 | Mar 24–30 |
| Core Framework (Why, What, How) |  |  |  |  |  |  |
| Testing Methodology & Governance |  |  |  |  |  |  |
| Test Automation & Tooling Strategy |  |  |  |  |  |  |
| Test Environments & Data Management |  |  |  |  |  |  |
| Risk Management & Testing Roadmap |  |  |  |  |  |  |
| Final Review & Sign-Off |  |  |  |  |  |  |

# Define the Core Framework

## Purpose & Value Proposition

### Identifies challenges, solutions, and success metrics.

To develop an effective testing strategy, we must identify team challenges and propose solutions that ensure efficiency, accuracy, and reliability. By analyzing each issue and suggesting specific interventions, we can improve our testing processes' quality and consistency.

Table 1 outlines the problems, proposed solutions, and success metrics to measure effectiveness. This approach links each solution to measurable outcomes, promoting continuous improvement and accountability across teams.

|  |  |  |  |
| --- | --- | --- | --- |
| **Reference** | **Problem Statement** | **Proposed Actions** | **Measurables** |
|  | Duplicate test cases by various teams | Centralized test case repository | % reduction in duplicate test cases, Time saved in test execution |
|  | Inconsistent test coverage leads to defects | Automated regression testing | % of automated test coverage, Defect Leakage Rate |
|  | It takes too long to identify reconciliation gaps | Automated reconciliation checks | Average time to resolve reconciliation issues, % of automated reconciliations |
|  | End users are unable to allocate enough time for testing | Shift-left approach with early user involvement | % of test participation by end users, Number of defects found before UAT |

Table 1: Identifying Problems & Solutions

### Connects solutions to principles, purpose, and values

This table connects problems with solutions in testing and reconciliation processes, aligning them with organizational principles. It promotes efficiency through centralized test case repositories and automated regression testing, reducing redundancy and improving reliability. Early user involvement fosters collaboration and ownership, aiding the adoption of new processes. The table supports quality assurance and proactive monitoring, ensuring smooth implementation of solutions.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proposed Actions** | **Purpose** | **Values** | **Strategy** | **Principles** |
| Centralized test case repository | Standardization | Efficiency, Consistency | Consolidate test cases centrally | Single Source of Truth |
| Automated regression testing | Quality Assurance | Reliability, Accuracy | Automate regression cycles | Shift-Left Testing |
| Automated reconciliation checks | Efficiency | Speed, Accuracy | Automate reconciliation processes | Proactive Monitoring |
| Shift-left with early user involvement | Adoption & Collaboration | Engagement, Ownership | Involve users early in UAT | Collaborative Testing |

Table 2: Extending Proposed Solutions to Strategy & Principles

### Purpose

The table consolidates the purposes of suggested actions for identified problems. It ensures each proposed solution aligns with the organization's objectives and values, emphasizing efficiency, reliability, accuracy, user engagement, and collaboration. Ultimately, it serves as a strategic tool for a structured and quality-centric testing environment.

|  |  |
| --- | --- |
| Aspect | Objective |
| Standardization | Reduce redundancy |
| Quality Assurance | Improve reliability |
| Efficiency | Speed up testing |
| Adoption & Collaboration | Engage users early |

### Values

Aligning core values to a common purpose is vital. It ensures all efforts in standardization, reducing redundancy, quality assurance, and reliability are coherent and goal-directed. This fosters efficiency, collaboration, early user engagement, and shared commitment.

|  |  |
| --- | --- |
| **Aspect** | **Benefit** |
| Efficiency | Save time and effort |
| Reliability | Ensure quality |
| Accuracy | Reduce errors |
| Consistency | Use structured methods |
| Engagement | Involve stakeholders |

### Principles

The table summarizes essential principles for achieving efficiency, reliability, accuracy, and engagement in standardization and quality assurance. Key practices include keeping a common test repository, adopting a shift-left approach to identify issues early, proactive monitoring to resolve gaps swiftly, and collaborative testing to improve adoption and engagement. These principles aim to reduce redundancy and enhance coherence in all efforts.

|  |  |
| --- | --- |
| **Principles** | **Description** |
| Single Source of Truth | Maintain a common test repository |
| Shift-Left Approach | Detect and fix issues earlier |
| Proactive Monitoring | Identify and resolve reconciliation gaps quickly |
| Collaborative Testing | Work with users for better adoption |

### Strategy and Actions

The table below outlines the strategy and corresponding actions designed to enhance quality assurance and standardization efforts. Each strategy aims to address specific challenges and improve overall efficiency, reliability, and user engagement. By adopting these practices, organizations can ensure a consistent and structured approach to quality management.

|  |  |
| --- | --- |
| **Strategy** | **Proposed Actions** |
| Consolidate test cases centrally | Centralized test case repository |
| Automate regression cycles | Automated regression testing |
| Automate reconciliation processes | Automated reconciliation checks |
| Involve users early in UAT | Shift-left with early user involvement |

## Scope & Key Stakeholders

### Type of Requirements

|  |  |
| --- | --- |
| **Category** | **Description** |
| Defects | Bug fixes, patches |
| Enhancements | Feature improvements, optimizations |
| New Requirements | Greenfield projects, new functionality |

### Acceptance criteria classification

|  |
| --- |
| Comparison results available from the current processing |
| Solution is incremental and hence the final user acceptance criteria not applicable |
| acceptance criteria based on reconciliation from previous batch run |
| acceptance criteria based on technology unit testing only |

Type of testing required

|  |
| --- |
|  |
|  |
|  |
|  |

### Key Stakeholders

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Regression Testing | Unit Testing | Functional Testing | UAT |
| FCD – Design Team |  |  |  |  |
| Technology |  |  |  |  |
|  |  |  |  |  |
| FCD - FT |  |  |  |  |
| Value Stream / FinOps |  |  |  |  |

* Regression and unit testing do not maintain test cases.
* Functional and user acceptance testing overlap significantly.
* Functional testing and user acceptance testing are fundamentally similar.
* Maintain all test cases centrally in Jira to avoid duplication.
* The release or Sprint Owner must approve all test cases.
* Indicate whether each test can be automated.

## Execution Approach

# Testing Methodology and Governance

## Testing Methodology

* Test cases should be classified in four types -
  + unit test cases
  + regression test cases
  + functional test cases
  + user acceptance test cases
* there should be clear demarcation for this type of testing.
* any test case related to the new requirements and existing process should be part of functional testing
* UAT leads should either accept or reject functional testing outputs but will not repeat those test cases
* UAT test cases should be based on exploration complex scenarios and pre-approved by end users
* All test cases should be maintained in Jira for execution

## Governance

* Technology teams must own automated test scripts.
* All defects should be accompanied with failed test cases
* Functional Test Cases should be run within sprint in the development environment
* Functional Test cases can be divided as basic Functionality Testing and complex Usera Acceptance Testing
* FT should take care for all

# Test Automation and Tooling Strategy

* Automation should use dashboards, stored procedures, or Python tools where possible.

# Tooling Strategy

* JIRA should be used for all test cases maintenance

# Test Environments, Data Management, and Reporting

# Risk Management and Testing Roadmap

# Final Review and Sign-Off

# Appendix

## Overall information flow

We have option to look up at the requirements end to end flow. There are opportunities to look for moving things left. I think there should be a way for creating the test cases before hand and be ready and to be done in the development environment.

Some of the discrepancy in the information that Design team provided a lot well defined information that can be easily consumed by Tech teams to perform the build.

My suggestions:

* The Design team should suggest test cases for the acceptance criteria, as they have verified requirements with the business.
* Capture most test cases at this stage. Provide these to the tech team as input and acceptance criteria.
* The tech team must execute these steps in the system.
* Business should review test cases before marking the process "Ready to build".
* Most test cases should be completed with the build.
* Unit testing and high-level Functional Testing will be done.

The process begins with the business requirements team, which ties each requirement to initiatives and projects. These requirements are then presented in a triage forum for first-level approval. Once no concerns are raised, the requirements are forwarded to the value stream lead for second-level approval and work estimation.

Following approval, the design team proposes the necessary changes. These designs are discussed in detail during compliance or quality control review sessions. The technology team then builds the solution and hands it off to the functional testing team. Once functional testing is completed, user acceptance testing (UAT) is conducted.

Opportunities for improvement include having the design team define test scripts for both unit and functional testing as part of their acceptance criteria. The technology team should execute these test cases and maintain them in Jira. Functional and user acceptance testing teams should attend triage and quality assurance calls to gain early visibility into upcoming testing requirements.

Additionally, the functional testing team will write UAT test cases. By taking on this responsibility, they can enhance their subject-matter expertise, reduce the burden on end users, and focus on more complex testing scenarios.