**QA Test Plan for QWID**

**Version 0.01**

# Contents

[Contents 2](#_Toc140092030)

[1. Test Strategy Information 4](#_Toc140092031)

[2. Definitions 5](#_Toc140092032)

[3. Reference Documents 5](#_Toc140092033)

[4. Introduction 5](#_Toc140092034)

[4.1 Background 5](#_Toc140092035)

[5. Integration & Intersystem Interfaces 5](#_Toc140092036)

[6. Test Environments 6](#_Toc140092037)

[6.1 Introduction 6](#_Toc140092038)

[6.2 Hardware Configuration 6](#_Toc140092039)

[6.3 Test Data 6](#_Toc140092040)

[6.4 Test Users 6](#_Toc140092041)

[7. Testing Responsibilities 7](#_Toc140092042)

[7.1 Roles & Responsibilities for Major Test Events 7](#_Toc140092043)

[8. Test Types 8](#_Toc140092044)

[8.1 Unit Testing - Devs 8](#_Toc140092045)

[8.2 Functional Integration Test – QA 9](#_Toc140092046)

[8.3 System Test 10](#_Toc140092047)

[8.4 User Acceptance Testing (UAT) 11](#_Toc140092048)

[9. Performance Testing 12](#_Toc140092049)

[9.1 Introduction 12](#_Toc140092050)

[9.2 Business Volume Metrics 12](#_Toc140092051)

[9.3 Performance Testing Methodology 12](#_Toc140092052)

[9.4 Test Scripting Criteria 12](#_Toc140092053)

[9.5 Test Execution Criteria 12](#_Toc140092054)

[9.6 Performance Test Runs 12](#_Toc140092055)

[9.7 Workload Mix Distribution 12](#_Toc140092056)

[9.8 Test Data Definitions 12](#_Toc140092057)

[9.9 Client-side Metrics 12](#_Toc140092058)

[9.10 Server-side Metrics 13](#_Toc140092059)

[9.11 Server-side Applications 13](#_Toc140092060)

[9.12 Server-side Monitoring Counters 13](#_Toc140092061)

[10. Test Deliverables 13](#_Toc140092062)

[10.1 Introduction 13](#_Toc140092063)

[10.2 Project Testing Related Tools 13](#_Toc140092064)

[11. Defect Management 13](#_Toc140092065)

[11.1 Introduction 13](#_Toc140092066)

[11.2 Objectives 14](#_Toc140092067)

[11.3 Defect Reporting & Resolution Process 14](#_Toc140092068)

[11.4 Defect Escalation Procedure 15](#_Toc140092069)

[11.5 Defect Severity Definitions 15](#_Toc140092070)

[11.6 Defect Lifecycle 15](#_Toc140092071)

[12. Results & Metrics Reporting 17](#_Toc140092072)

[12.1 Introduction 17](#_Toc140092073)

[12.2 Training Requirements 17](#_Toc140092074)

# Test Strategy Information

|  |  |
| --- | --- |
| Project Information | |
| **Project:** | QWID |
| **Version Number:** | 0.01 |
| **Testing Phase:** | Functional |
| **Date:** | 11 July 2023 |

|  |  |
| --- | --- |
| Review & Approval | |
| **PO Approval** | Tolu Okesanya |
| **PO Approval Signature** | <Signature of Project Owner> |
| **Lead Dev Approval** | Daniel Makinde |
| **Lead Dev Approval Signature** | <Signature Lead Developer representative > |
| **Lead QA Approval** | Andre Dada |
| **Lead QA Approval Signature** | <Signature Lead QA representative > |

The undersigned acknowledge that they have reviewed the Test Strategy plan and agree with the information presented within this document. Changes to this plan will be coordinated with, and approved by, the undersigned, or their designated representatives. The Project Sponsor will be notified when approvals occur.

# Definitions

|  |  |
| --- | --- |
| **Term** | **Meaning** |
| UAT | User Acceptance Testing |
| FED | Front End Developer |
| BED | Back End Developer |
|  |  |
|  |  |
|  |  |
|  |  |

# Reference Documents

|  |  |
| --- | --- |
| **Document** | **Repository Path** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Introduction

## Background

This document provides the strategy for the testing activities required for the overall testing activities for the QWID app.

QWID offers a simple and versatile solution for cross-border payments, catering to both individuals and businesses. With its easy-to-use digital payment system, multiple currency options and referral programs, QWID makes sending and receiving money from over 100+ countries, a seamless process.

The current version at the time of this document is version <what version is this?>

* *Any other relevant supporting information.*

# Integration & Intersystem Interfaces

The contents list below, shows the various interfaces/applications involved in the integration testing of the QWID app, as well as the individual points of contact/s that will be required for coordinating any integration testing.

|  |  |  |
| --- | --- | --- |
| **System ID** | **Application/Functional Area** | **Testing Responsibility/SME** |
|  |  |  |
|  |  |  |
|  |  |  |

# Test Environments

## Introduction

This chapter identifies the environments to be used for testing. The assumption being made here is that all processes will adhere to AGILE principles.

## Hardware Configuration

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Dev** | **QA** | **Pre-Prod** | **Production** |
| Application Server |  |  |  |  |
| Database Server |  |  |  |  |

## Test Data

The Test Lead will define high level data requirements as needed for the testing of the key areas, such as User IDs and passwords. Attention to detail, such as dropdown boxes and user entered information will need to be passed along to the Performance Tester*.* Also, a determination needs to be made if the DB needs to be seeded with additional data to replicate the Production environment more closely.

For example, if the Production DB has 15 million rows of data, the same amount should be used in the Test Environment:

* Describe data needed by environment
* Test data refresh requirements
* Data seeding requirements
* Data requirements for end-of-end business processes
* Means by which data can be utilized for test data creation during development execution

|  |  |  |
| --- | --- | --- |
| **Application/Service** | **Assigned Resource** | **Data Requirements** |
|  |  |  |
|  |  |  |
|  |  |  |

## Test Users

Each test case will require one or more test users. Test users must be created to replicate real business users allowing defects related to authorisation profiles and delegation of duties to be identified. Using unrealistic role assignment for test users will invalidate all functional tests.

A catalogue of test users should be maintained. Automatic provisioning of test users’ needs to be established as part of the setup of the test environment.

# Testing Responsibilities

## Roles & Responsibilities for Major Test Events

The following information serves as a guide on how the testing responsibilities will be shared across the stakeholders:

R – Responsibility

A – Accountability

C – Consulted

I – Informed

**Responsible**: Those who do the work to achieve the task. There is typically one role with a participation type o responsibility, although others can be delegated to assist in the work required.

**Accountable**: (Also Approver or final Approving Authority) those who are ultimately accountable for the correct and thorough completion of the deliverable or task, and the one to whom Responsible is accountable. In other words, An Accountable must sign off (Approve) on work that Responsible provides. There must be only one Accountable specified for each task or deliverable.

**Consulted**: Those whose opinions are sought, and with whom there is two-way communication

**Informed**: Those who are kept up-to-date on progress, often only on completion of the task or deliverable, and with who there is just one-way communication.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Phase** | **Activity** | **Test Lead** | **Dev Lead** | **DBA** | **PM** | **BA** | **Comments** |
| Requirements Analysis | Providing a detailed list of requirements in scope for the release | I | I | I | A&R | R | Functional Lead is also responsible for any functional requirements if developed for release |
| Test Case Development | Requirements review and test case development | A&R | C | C | C | C | During requirement understanding and test case development, the Test Lead will require help from other stakeholders to finalize the test approach |
| Test Case Sign-off | Test requirements review and sign off | A | R | I | I | I | While the Test Lead has the final accountability on finalizing the Test Cases, other stakeholders have a responsibility to review and provide sign off |
| QA Test Execution | Perform FT, SIT, PT and report results | A&R | R | C | C | C | During test execution, if clarifications are required, the BA/PO will be consulted. The Dev team takes equal responsibility to ensure that test execution is completed |
| Bug Reporting | Bug fixes and retest | A&R | R | C | C | C | During bug fixes, if any clarifications are required, the BA/PO will be consulted. The Dev team takes equal responsibility in resolving and ensuring retest is completed |
| QA Test Execution | Test results, review and sign off | A | A | I | A | R | The QA Lead is responsible for providing this information to the stakeholders |
| Acceptance Test Execution | UAT execution | A&R | I | I | A | R | The QA Lead is responsible for managing this process with the support of the BA to ensure that |

# Test Types

## Unit Testing - Devs

|  |  |
| --- | --- |
| **Purpose** | This preliminary test is conducted by the dev team for testing of individual configuration, custom programs and/or technical services, to ensure that they function according to the detailed technical specs.  Unit test is a white box test and should test all possible flows. Both positive and negative conditions should be tested. |
| **Development Phase** | Development and Testing |
| **Test Scope** | All configurations, code validation, memory testing, integration, code complexity, etc |
| **Test Environment** | Development environment |
| **Test Data** | Manually created by devs |
| **Interface Requirements** | NA |
| **Role** | Developer |
| **Entry Criteria** | * Formal reviews for process models, functional spes and technical specifications have been completed * All Inspection related defects have been corrected * All documentation and design of the architecture must be made available * Development of the component is complete and compiles without error * All Unit test cases are documented |
| **Exit Criteria** | * All unit test cases completed successfully * All source code is unit tested * No outstanding critical defects * All outstanding defects are entered into the defect tracker * All test results have been documented |

## Functional Integration Test – QA

|  |  |
| --- | --- |
| **Purpose** | Functional test validates the full operability of the interconnected functions, methods, or objects within a functional area. This includes a set of logically related activities or business processes to achieve a defined business process.  Functional test cases will typically consist of a series of business processes or stories joined together to achieve a business process. The smaller size of test cases will enable the testing of multiple data sets and permutations.  It happens after or in parallel with the development phase as and when all components for a specific flow are complete. Functional test will be done by independent testing team in QA environment.  During subsequent integration testing activities these business process (functional) tests are combined to build end-to-end integration test scenarios. |
| **Development Phase** | Development and Testing |
| **Test Scope** | All functional tests, requirement/story coverage using test design techniques like Orthogonal Analysis, Decision Tables, Equivalence Partitioning, etc. |
| **Test Environment** | Test Environment |
| **Test Data** | Manual data created by Test team |
| **Interface Requirements** | Interface connectivity required for impacted systems |
| **Role** | QA team |
| **Entry Criteria** | * All specs are frozen and the requirements change control process has begun * Proper test data is available * Test plans and test cases are reviewed and signed off * Unit Testing has been completed * Specifications for the product have been completed and approved * All test hardware platforms must have been successfully installed, configured and functioning properly. * All standard software tools including testing tools must have been successfully installed and functioning properly. * All personnel involved in the system test effort must be trained in tools to be used during testing process. * All personnel involved in the system test effort must be trained the usage of the application and new features. * All functional test cases are documented |
| **Exit Criteria** | * Test case execution completed with 90% passed * All defects are recorded in JIRA * No outstanding “showstopper or severe” defects * All test results have been documented * All code has been migrated into the QA environment * Coverage of code/functionality/**requirements** is 100% of functional requirements. |

## System Test

|  |  |
| --- | --- |
| **Purpose** | SIT validates a set of business processes that define a business scenario in a comprehensive and self-contained manner on a macro level.  This is an end-to-end test of the business process. Typically, a business scenario will involve testing multiple modules test cases together. The primary objective of this testing is to discover errors in the integration between different modules and to verify that the modules work together correctly as one function.  Security role-based authorization test is performed to ensure that all the security profiles and roles are being implemented as designed. Security profile is designed and built based on the job role (i.e., positions) of the end users. Security roles are assigned at the business transaction level. The objectives of security testing are:   * Ensure that user has access to the required transactions to perform their job * Ensure that the user does not have access to transactions other than what is required for the role * Ensure that accesses to critical system administration transactions are controlled. * Ensure that only authorized person has the right to view the information on Screens and Reports. |
| **Development Phase** | Development and Testing |
| **Test Scope** | * Full End to end business process * Performance Testing * Regression * Interface testing with interfacing systems * Security role-based authorization testing * End-to-End scenarios executed with user ID mapped to actual security roles * Batch jobs execution using scheduled runs |
| **Test Environment** | QA Environment or Pre-Prod |
| **Test Data** | Data injection/mocking |
| **Interface Requirements** | Interface connectivity required for all interfacing systems |
| **Role** | QA team |
| **Entry Criteria** | * All specs are frozen and the requirements change control process has begun * Proper test data is available * Test plans and test cases are reviewed and signed off * SIT ver ‘X’ has been completed * All functional test cases are documented |
| **Exit Criteria** | * Test case execution completed with 100% passed * All defects are recorded in a robust Test Management tool * No outstanding “showstopper or severe” defects * All test results have been documented * All code has been migrated into the Pre-Prod environment * No new defects have been discovered for a week prior to System Testing. * Coverage of code/functionality/**requirements** is 100% of functional requirements. |

## User Acceptance Testing (UAT)

|  |  |
| --- | --- |
| **Purpose** | User acceptance test is performed by business users. The users test the complete, end-to-end business processes to verify that the implemented solution performs the intended functions and satisfies the business requirements. |
| **Development Phase** | Final Prep or Implementation |
| **Test Scope** | * UAT * Full Regression |
| **Test Environment** | Pre-Prod/Staging |
| **Test Data** | Data injection/mocking |
| **Interface Requirements** | Interface connectivity required for all interfacing systems |
| **Role** | Process Team & Business Users |
| **Entry Criteria** | * The application works functionally as defined in the specifications * No outstanding “showstopper or severe” defects * All areas have had testing started on them unless pre agreed by UAT stakeholder/Test and Project Managers * Entire system functioning and all new components available unless previously agreed between UAT stakeholder/Test manager and project managers * All test cases are documented and reviewed prior to the commencement of UAT |
| **Exit Criteria** | * The Acceptance Tests must be completed, with a pass rate of not less than 95%. * No outstanding “showstopper or severe” defects * Less than 5 significant defects outstanding * All Test cases have been complete * No new defects have been discovered for a week prior to Production Implementation. * All test results recorded and approved * UAT test summary report documented and approved * UAT close-off meeting held. |

# Performance Testing

## Introduction

This chapter documents the non-functional testing techniques to be carried out to determine the system effectiveness with regards to stability and responsiveness under predetermined workloads.

## Business Volume Metrics

## Performance Testing Methodology

## Test Scripting Criteria

## Test Execution Criteria

## Performance Test Runs

## Workload Mix Distribution

## Test Data Definitions

## Client-side Metrics

## Server-side Metrics

## Server-side Applications

## Server-side Monitoring Counters

# Test Deliverables

## Introduction

This chapter details the deliverables from the QA Team. It also delves into the tools that will be used to support this process.

## Project Testing Related Tools

|  |  |
| --- | --- |
| **Phase/Activity** | **Test Tool Requirement** |
| Test case documentation (Manual & Automation) | JIRA |
| Requirement Management | Confluence/Google Docs |
| Test case automation development & execution | Cypress, Gitlab |
| Test execution & results reporting | JIRA |
| Defect reporting & tracking | JIRA |
| Document storage | Confluence/Google Docs |
| Business process flow | Confluence/Google Docs |
| Test data management | Confluence/Google Docs |
| Service virtualization | Browserstack, Virtualbox |
| Code analysis | Static, Manual |
| Code coverage | Cypress |
| System monitoring (during performance testing) |  |

# Defect Management

## Introduction

Defect Review Meetings will be held daily with SME Leads, Test Managers, Business Analysts, and the Product Owner. The goal of this meeting is to ensure that defects are resolved in a timely fashion and that any issues or questions are resolved. It is at these meetings that progress tracking of defect resolution and closure is communicated.

## Objectives

* To help prioritize defect fixes for implementation, legacy support, and upcoming releases.
* To discuss and assign priority and severity to defects, discuss the expected turnaround time and planned turnaround time.
* To monitor and review the progress of defect fixes that is due/overdue as of current date.
* To determine the extent of retesting required due to a fix/enhancement.
* To escalate defects/issues to the PO when a quick resolution is required, or in case of a deadlock on ownership of defects/issues.
* To identify whether a defect has been assigned to the right team/individual.
* To identify defects that need to be transferred to subsequent releases.

## Defect Reporting & Resolution Process

**Prerequisite:** Development team & Business Team should have access to defects section of Jira and are able to update the defect details.

Below diagram helps to understand the defect life cycle process quickly and easily.

Diagram

Description automatically generated

## Defect Escalation Procedure

**First level of notification**: As soon as the defect is logged in to JIRA, an auto generated email would be sent to the assigned person. Since the defect will be assigned to development team alias, all the team who are subscribed to the alias would get the email. QA will also copy and send the link for the ticket to the appropriate channel on Slack for the attention of the assigned person.

**Daily status review meeting**: Along with the test execution status discussions, all the outstanding defects would be discussed in the meeting. Development team, business team, QA and other stakeholders as appropriate would join the meeting. Defect details and estimated time of fix would be documented in JIRA accordingly.

## Defect Severity Definitions

|  |  |  |
| --- | --- | --- |
| **Severity** | **Definition** | **Expected Time for Closure** |
| Critical | A complete software system, or a subsystem, or software unit (program or Module) within the system lost its ability to perform its required function (=Failure) and no workaround available  OR  Testing of a significant number of tests cannot continue without closure  OR  Potential showstopper for Go/No-Go decision deploy a major feature to Production | 2hr |
| Major | The software system, or subsystem, or software unit (program or module) within the system produces Incorrect, Incomplete, or Inconsistent results  OR  Defect impairs the usability (capability of software to be understood, learned, used and attractive to the user when used under specified conditions [ISO 9126] | 1 Business Day |
| Minor | Everything not Major or Critical | 2 Business Days |

## Defect Lifecycle

As part of the Defect Life Cycle definition and Defect Management process, various Defect stages will be identified as mentioned below

|  |  |  |  |
| --- | --- | --- | --- |
| **Defect Status** | **Description** | **Required Previous Status** | **Next Possible Status** |
| **New** | Defect identified and raised by a Team  Defect is not reviewed by the Assigned Team | NA | Open  Assigned |
| **Open** | Assigned team acknowledges the defect by moving the defect to open status  No one has been assigned to analyse the defect | New | Assigned  Rejected  Deferred  Duplicate |
| **Assigned** | Defect is assigned to a user for analysis | New  Open | Need more info  Deferred  Duplicate  Fixed  Retest |
| **Rejected** | An invalid defect has been logged. The defect can be rejected by the Assigned Team for various reasons:   * Invalid data used by the tester * Invalid test case executed by the tester * Test steps followed by the tester were incorrect   Note: If the defect is rejected because requirements were changed but the QA team were not notified, then the defect should not be rejected, rather, it should be closed. | Open  Assigned | Assigned |
| **Fixed** | Assigned Team moves the defect to fixed when the defect is fixed and ready to be deployed. | Assigned | Retest |
| **Retest** | Assigned Team moves the defect to Re-test when the defect has been deployed for testing on the required environment | Fixed | Closed  Re-open |
| **Re-open** | In the retest fails, then the defect is reopened and assigned back to the previous team which worked on the defect.  Note: if the retest of the defect fails because of a reason different to what the defect was logged for, then a new defect should be opened for the new issue.  The current defect shouldn’t be reopened in such cases. | Retest | Assigned  Fixed  Retest |
| **Closed** | Defect passes retest and can be closed | Fixed  Retest |  |
| **Deferred** | Defect is acknowledged by the Assigned Team but cannot be fixed within the Release timeline due to some constraints. The defect is then deployed to Production as a known risk.  Note: This scenario does not apply to Critical defects.  To be able to move a defect to Deferred status, approval from all key stakeholders is required.  A Change Request (CR) needs to be initiated to cover the work required in the future to address the defect. | Assigned  Re-open |  |
| **Duplicate** | The defect is a duplicate of an existing open defect and is the same as the previous one.  Previous defect ID needs to be updated in this case | Open  Re-open | Assigned |

# Results & Metrics Reporting

## Introduction

The metrics listed below will be published to provide the stakeholders with an update and status of any version release.

To reduce the complexities around generating these reports, as well as best utilizing the efforts of the QA team in testing the application, it’s advisable to employ a test management tool which can produce said reports with minimal effort and better accuracy.

|  |  |  |
| --- | --- | --- |
| **Report Name** | **Details** | **Frequency** |
| Weekly status report | Status report to be sent by Friday every week by 3PM | Weekly |
| Daily Status Report during test execution phase | Test execution status of all the features under test. This will contain:   * Test execution planned vs completed * Test case pass/fail numbers * Defects open, close and severity | Every working day |
| QA Velocity Rate | This report will provide a view on how many defects are being worked on by the QA team over a given period | Weekly |
| Closure Report | Summary of a just concluded text execution phase for all stakeholders to sign off on | End of each test phase |

## Training Requirements

This section documents any training needs for the staff involved in the testing.

1. Automation Testing – Cypress (ongoing)

2. Performance Testing – Gatling/JMeter

3. Test scripting – BDD (ongoing)

4. Bug reporting – JIRA (ongoing)