**Functional Test Report for Inji Wallet (Web) v0.11.1**

**17-01-2025**

# 

[Testing Scope 3](#_2et92p0)

[Test Approach 3](#_tyjcwt)

[Verified configuration 4](#_3dy6vkm)

[Test execution statistics 4](#_4d34og8)

[Functional test results 4](#_2s8eyo1)

[UI Automation test results: 4](#_17dp8vu)

[Detailed Test metrics 5](#_3rdcrjn)

# 

# Testing Scope

The scope of testing is to verify fitment to the specification from the perspective of

* Functionality
* Deployability
* Configurability
* Customizability

Verification is performed with a UI test rig.

Testing scope has been focused on the below features:

* Automation Functional UI.

# Test Approach

Persona based approach has been adopted to perform the IV&V, by simulating test scenarios that resemble a real-time implementation.

A Persona is a fictional character/user profile created to represent a user type that might use a product/or a service in a similar way. Persona based testing is a software testing technique that puts software testers in the customer's shoes, assesses their needs from the software and thereby determines use cases/scenarios that the customers will execute. The persona needs may be addressed through any of the following.

* Functionality
* Deployability
* Configurability
* Customizability

The verification methods may differ based on how the need was addressed.

# Verified configuration

Verification is performed on various configurations as mentioned below

* Default configuration - with 6 Lang
  + English
  + French
  + Arabic
  + Hindi
  + Tamil
  + Kannada

# Test execution statistics

## Functional test results

Below are the test metrics by performing Automation UI functional testing. The process followed was black box testing which based its test cases on the specifications of the software component under test. Functional test was performed in combination of individual module testing as well as integration testing. Test data were prepared in line with the user stories. Expected results were monitored by examining the user interface. The coverage includes GUI testing, System testing, End-To-End flows across multiple configurations.

## UI Automation test results:

Below are the test metrics by performing UI Automation:

| **Total** | **Passed** | **Failed** | **Skipped/Ignored** |
| --- | --- | --- | --- |
| 19 | 19 | 0 | 0 |
| Test Rate: 100%, With Pass Rate: 100% | | | |

Functional and test rig code base branch which is used for the above metrics is:

Hash: 9bf9e9f60744c372670fd9816e65cf81987b0d7e

## Detailed Test metrics

Below are the detailed test metrics by performing automation testing. The project metrics are derived from Defect density, Test coverage, Test execution coverage, test tracking and efficiency.

The various metrics that assist in test tracking and efficiency are as follows:

* Passed Test Cases Coverage: It measures the percentage of passed test cases. (Number of passed tests / Total number of tests executed) x 100
* Failed Test Case Coverage: It measures the percentage of all the failed test cases. (Number of failed tests / Total number of test cases executed) x 100

Git hub link for the xls file:

<TBD>