Quathera

Reports & Analytics Guide

Test coverage matrix, bug reports, Jira integration, and analytics.

Version 1.0 | 2025

Table of Contents

1. Test Coverage Matrix

The Test Coverage Matrix is a visual dashboard showing test results across versions:

Matrix Structure:

- Rows: Test scenarios/form pages
- Columns: Application versions (newest on right)
- Cells: Pass/fail status for that test on that version

What It Shows:

- Overall test health at a glance
- Which tests are failing on current version
- Trends over time (improving or degrading)
- Areas needing attention

Accessing the Matrix:

1. Go to 'Reports' in the sidebar
2. Select 'Coverage Matrix'
3. Choose project and date range

2. Understanding Coverage Colors

The matrix uses color coding for instant insights:

■ Green (100% Pass):

All scenarios for this form passed on this version.

■ Orange (50-99% Pass):

Some scenarios passed, some failed. Needs attention.

■ Red (<50% Pass):

Majority of scenarios failed. Critical issues likely.

■ Gray (Not Run):

Tests haven't been run for this version yet.

Click any cell to drill down into specific scenario results.

3. Bug Reports

When tests fail, AI automatically generates detailed bug reports:

When Bugs Are Created:

- Validation mismatch (expected vs actual)
- Element not found and cannot self-heal
- Unexpected error message
- Visual verification failure
- Network error (4xx, 5xx)

Viewing Bug Reports:

1. Go to 'Bug Reports' in sidebar
2. See list of all bugs found
3. Filter by status, severity, form, date
4. Click any bug for full details

4. Bug Report Contents

Each bug report includes comprehensive information:

Summary:

Auto-generated title describing the issue.

Steps to Reproduce:

Exact navigation path and actions to reproduce the bug.

Expected Result:

What should have happened based on the test scenario.

Actual Result:

What actually happened (the failure).

Screenshots:

- Screenshot at moment of failure
- Screenshot of expected state (if available)
- Highlighted elements showing the issue

Technical Details:

- Browser and version
- Environment (QA/Staging/Prod)
- URL where failure occurred
- Timestamp
- User type logged in as

5. Jira Integration Setup

Connect Quathera to Jira for automatic bug creation:

Setup Steps:

1. Go to 'Settings' > 'Integrations'
2. Click 'Connect Jira'
3. Enter your Jira instance URL
4. Authenticate with Jira (OAuth or API token)
5. Select default project for bugs
6. Map Quathera fields to Jira fields
7. Save configuration

Field Mapping:

- Quathera Summary → Jira Summary
- Quathera Description → Jira Description
- Quathera Severity → Jira Priority
- Quathera Form → Jira Component (optional)
- Quathera Screenshots → Jira Attachments

6. Automatic Bug Creation

Once Jira is connected, bugs are created automatically:

How It Works:

1. Test fails and bug is detected
2. AI generates bug report with all details
3. AI checks for duplicates (see next section)
4. If not duplicate, creates Jira issue
5. Attaches screenshots to Jira issue
6. Links Quathera bug to Jira issue

Configuration Options:

• Auto-create: Create Jira issues automatically

• Manual review: Queue bugs for review before creating

• Severity threshold: Only create for High/Critical

7. Duplicate Detection

AI prevents creating duplicate bugs:

How Duplicates Are Detected:

Before creating a new bug, AI checks:

- Same form and field involved
- Similar error message or failure type
- Same navigation path
- Recent occurrence (within configured window)

What Happens with Duplicates:

- New occurrence is linked to existing bug
- Occurrence count is incremented
- Latest screenshot is added
- Comment added to existing Jira issue

This keeps your Jira clean and shows how often bugs occur.

8. Regression Identification

AI identifies when fixed bugs reappear:

How It Works:

1. A bug was previously found and fixed (Jira issue closed)
2. Same failure occurs in a new test run
3. AI matches it to the closed bug
4. Bug is flagged as a REGRESSION

Regression Handling:

- Jira issue is reopened automatically
- Comment added explaining it's a regression
- Higher priority may be assigned
- Notifications sent to assignee

Why This Matters:

Regressions indicate problems with code stability or test coverage. Tracking them helps identify problematic areas.

9. Exporting Reports

Export data for sharing and analysis:

Coverage Matrix Export:

- PDF: Visual report with color coding
- Excel: Detailed data for analysis
- CSV: Raw data for custom processing

Bug Report Export:

- PDF: Individual bug report
- PDF: Summary of all bugs in date range
- Excel: Bug list with all details

Test Run Report:

- PDF: Summary of test run results
- Includes pass/fail counts, duration, key failures

Scheduled Reports:

Configure automatic reports to be emailed:

- Daily summary of test results
- Weekly coverage trend report
- Monthly quality metrics