QAmatic User Guide

A practical guide to game testing

QAmatic is a QA portal for testing Stake Engine games. It helps a QA team keep common test cases in one place, add game-specific cases for a single run, and produce a clear PDF report.

It works without GitHub or source-code access. The main requirement is a playable Stake Engine game link.

What QAmatic Can Do

- Set up a test run for Slot, Table, or Instant games.
- Check that the selected game type matches the information in the game link and test cases.
- Maintain shared test-case templates for each game type.
- Import multiple Excel or CSV files at once.
- Keep detailed test instructions, including preconditions, steps, expected result, BDD scenarios, type, and attachments.
- Run Lite or Deep testing.
- Show live results, filter them by status, and download a branded PDF report.
- Capture a browser screen image for failed Deep tests when one is available.

Start a Test Run

1. Open **Setup Game Testing**.
2. Choose the game type: **Slot game**, **Table game**, or **Instant game**.
3. Enter the game name. QAmatic automatically uses title case, for example `golden boot` becomes `Golden Boot`.
4. Add the playable Stake Engine beta or alpha URL.
5. Optionally upload one or more game-specific test-case files.
6. Choose Lite testing or Deep testing.
7. Select **Start testing**.

After a field has been changed, the selected game type is locked. Use **Reset** to start a new setup and choose a different type.

Game Name Checks

QAmatic blocks placeholder names such as `test`, `random words`, or `demo`.

When the playable URL contains a readable game name, the entered name must share at least 50% of its words with the URL name. This helps prevent a real game link from being tested under the wrong game name.

For example, `Golden` can match a `golden-boot` URL, while an unrelated name such as `Blue Keno` is rejected.

Game Type Checks

QAmatic looks at the URL, game name, and supplied test cases to identify the game type. If these signals point to a different type from the one selected, QAmatic blocks the run and explains the mismatch.

If QAmatic cannot confidently identify a type, it does not block the run.

Game-Specific Test Cases

Game-specific files are for cases that belong to one game or one test run. They are not added to the common template library.

Supported Files

You can select multiple files at once. Only these formats are accepted:

- Excel: `.xlsx` and `.xls`
- CSV: `.csv`

Each selected file shows a status such as **Reading**, **Ready for run**, or **Not added**. The portal also shows how many cases were read from each file.

Files are read in the browser. They are not uploaded to a cloud service and are not saved as files by QAmatic. Parsed cases are kept only for the current browser session and current test setup. Refreshing or resetting clears them.

Imported Columns

QAmatic accepts flexible column names and ignores columns it does not need. The most useful columns are:

Purpose	Recognised column names
Source case ID	`Test case ID`, `Test ID`, `ID`
Test case name	`Title`, `Test Case`, `Test Case Title`, `Description`
Category	`Category`, `Module`
Precondition	`Precondition`, `Preconditions`
Steps	`Steps`, `Test Steps`, `Step`
Expected result	`Expected result`, `Expected results`, `Expected outcome`, `Expected`
BDD scenarios	`BDD scenarios`, `BDD scenario`, `Scenario`, `Scenarios`
Type	`Type`, `Test type`
Attachments	`Attachments`, `Attachment`
Test scope	`Test depth`, `Execution scope`, `Run in`, `Depth`

`Folder` is intentionally ignored.

For Testiny exports, QAmatic keeps the original Testiny case ID as **Testiny case ID**. This makes it easy to trace a result back to Testiny.

Test Case Templates

Open **Templates Config** to manage the common library used by every game of a selected type.

Template Controls

- Choose a game type from the prominent **Game type** dropdown.
- Import one or more Excel or CSV files into that game type's common library.
- Add a new test case directly from the library section.
- Edit or delete individual cases.
- Select multiple cases, including **Select all**, and delete them together.

Template changes are saved to QAmatic's local data store at `qamatic/data/templates.json`. They remain available after a browser refresh and after the local server restarts.

QAmatic creates the template Test ID automatically. IDs are incremental and are never reused after a case is deleted. Imported source IDs are kept separately as the Testiny case ID.

Test Case Details

Each template case can store:

- Test case name and category
- Precondition

- Steps
- Expected result
- BDD scenarios
- Type
- Attachments
- Test scope

Use the execution-details expander in the template list to view imported detail. Use the edit icon to update it.

Test Scope and Test Depth

Every case has one of three scopes:

- **Lite only**: included only in Lite testing.
- **Deep only**: included only in Deep testing.
- **Lite + Deep**: included in both modes.

QAmatic does not show out-of-scope cases as skipped. It leaves them out of the run completely.

Lite Testing

Lite testing does not open a browser with Playwright. It checks the setup, URL, template selection, and test-run configuration.

Lite runs these cases:

- Lite only
- Lite + Deep

Deep Testing

Deep testing opens the playable URL in Playwright. It checks whether the game page loads, looks for game-related network activity, and records a screenshot for a failed browser check when possible.

Deep runs these cases:

- Deep only
- Lite + Deep

Deep testing currently provides a browser and network baseline. Written preconditions, steps, expected results, and BDD scenarios are retained with every case and passed into the run and report. To turn written scenarios into reliable browser actions, QAmatic also needs machine-readable instructions such as UI selectors, explicit actions, or executable BDD bindings. For math-event assertions, it needs the game math-event names and payload contract.

Run Results and Reports

The results table shows the test case, category, status, result message, and Testiny case ID where available.

Available result statuses are:

- **Pass**: the check completed successfully.
- **Fail**: the check failed. A screen-capture link appears when Deep testing was able to capture one.
- **Manual**: the current runner needs more game-specific automation information to make a reliable automatic assertion.
- **Skipped**: shown only for exceptional cases; normal scope selection removes ineligible cases before the run starts.

Use the **Status** filter to focus on a subset of results. The PDF report uses the same active filter, so a report downloaded after selecting **Fail** contains only failed cases.

The PDF includes:

- QAmatic branding and logo
- Game and run information
- A pass/fail/manual/skipped summary
- Color-coded status tags
- Filtered result rows
- Stored test-case context
- Failure screenshots when available

When results are visible, QAmatic asks for confirmation before Reset and before refreshing, closing, or navigating away from the page. This helps avoid losing the visible run results accidentally.

What You Need for a Full Deep Test

For the strongest automated coverage, provide:

- A playable Stake Engine beta or alpha link from an authenticated test account.
- The correct game name and game type.
- Game-specific test cases in Excel or CSV.
- Stable UI selectors or accessibility IDs for game controls and result states.
- A list of game math events and example event payloads.
- Expected RTP, max-win, special feature rules, and controlled or replayable test scenarios where relevant.

No GitHub link or local source-code path is required for the standard portal workflow. Code access becomes useful only when the team wants to build deeper, game-specific browser automation or inspect game events directly.