QA Automation — User Module
Client Guide: Scripts Execution and Explanation
Playwright Automation — One-button run setup

Prepared for: Client
Prepared by: Yahya M. Mirza

Contents

1 Overview

This document explains how to run and understand the automated test suite for the **User Module** of your web application. It focuses on client-relevant information: running tests, expected input data, reading results, and key pre-checks before execution.

The automation uses Playwright and is configured for a one-button run that produces a clear HTML report.

2 Why This Matters

Automated tests provide fast, repeatable checks of the main user journeys in the User Module (daily symptom tracking, cycle management, and user profile updates). Instead of manually checking these flows each release, you run the suite and immediately see whether critical functionality has regressed.

Benefits:

- **Confidence:** Key flows are verified automatically.
- **Speed:** One-button run completes in minutes with a human-friendly report.
- **Clarity:** Report highlights failed steps and reasons.

3 Test Data — The Input of the Tests

All client-facing tests rely on a single source of truth: `testdata.ts` located in `tests/fixtures/`. This file contains the records and names the tests expect in the application.

3.1 Naming Conventions

Fields in `testdata.ts` use clear, descriptive names so clients know what to check. Example:

```
USER.SYMPTOMS.FLOWLEVEL = "Moderate"
```

Update `testdata.ts` if live data changes.

3.2 Common Fields for the User Module

Typical entries include:

- Flow level options
- Symptom names (cramping, mood changes, etc.)
- Days to feel normal options
- Date selections for check-ins

3.3 File Path

```
project-root/tests/fixtures/testdata.ts
```

3.4 Your Responsibility (Data Ownership)

Ensure records defined in `testdata.ts` exist in the target environment (staging/testing). Missing or renamed records can cause failures or skips.

4 Prerequisites (one-time setup)

1. Node.js LTS (v18+) — verify with `node -v`
2. Install project dependencies: `npm ci` or `npm install`
3. Install Playwright browsers: `npx playwright install`

5 How to Run Everything (One Button)

5.1 Windows (most common)

1. Open the project folder.
2. Double-click `run-tests.bat`.
3. All User Module tests run (typical runtime: 6–8 min).
4. HTML report opens automatically.

5.2 macOS / Linux

Open terminal in project root and run:

```
npm run test:all
```

Behaves the same as the one-button script.

5.3 Report

Shows `PASSED` / `FAILED` / `SKIPPED` and details for failures.

6 Skipping a Test (Optional)

Temporarily disable a test by marking it skipped:

```
test.skip("This test will be skipped", async () => { ... })
```

Skipped tests appear as SKIPPED instead of failing.

7 Fixtures & Test Files

All inputs (images, CSVs, etc.) are stored in `tests/fixtures/`.

7.1 Purpose of This Folder

Inputs for tests like file uploads or sample CSVs.

7.2 How to Change Fixtures

- **Images:** Replace files or update paths in `testdata.ts`.
- **CSV Files:** Edit or replace and update paths in `testdata.ts`.

8 Test Data Structure Explanation

The User Module uses a consistent pattern to show what is created or updated during tests.

8.1 Example — Daily Symptom Tracking

```
dailyCheckIn: {
  new: {
    flowLevel: "Moderate",
    cramping: "Severe",
    moodChanges: "Mild",
    daysToFeelNormal: "2-3"
  },
  edit: {
    oldFlowLevel: "Moderate",
    newFlowLevel: "Heavy",
    successMsg: "Check-in updated successfully"
  }
}
```

8.2 Breaking This Down

- **new:** Fields for creating a new record.
- **edit:** Fields for updating an existing record. `oldFlowLevel` is searched; `newFlowLevel` is applied.

8.3 Why This Matters

Ensures creation and modification flows are verified. Missing old records may cause failures or skips.

9 Troubleshooting & Reading Reports

9.1 Common Cases and Fixes

1. **Element not found:** UI changed. Notify automation owner.

2. **Data missing:** Required record not found. Update `testdata.ts` or app data.
3. **Slow network:** Retry on stable connection or adjust timeouts temporarily.

9.2 Reading a Failed Test

Click a failed test in the HTML report to see:

- Error message (expected vs. actual)
- Step-by-step trace and screenshots

Determine if failure is due to missing data or UI changes.

9.3 If the Report Doesn't Open Automatically

Run:

```
npx playwright show-report
```

10 Responsibilities (Important)

- Ensure all records exist in the environment before running tests.
- If multiple users run tests, use the same environment to avoid false failures.

11 Support & Change Requests

- Minor tweaks (selector updates, small test additions) — quick turnaround.
- New modules or major changes — estimated separately.
- Re-activating skipped tests — remove `.skip`.

12 Final Notes

- One double-click runs all User Module tests and opens the HTML report.
- Keep `testdata.ts` updated to match your environment.
- Typical runtime: 6–8 minutes.
- Report provides guidance on failures for faster triage.