

# Playwright E2E Testing Project Summary

Generated on: 2025-08-05 08:59:22

## Overview

This document outlines the structure, purpose, and functionality of the current Playwright-based end-to-end testing project. It also highlights important implementation details, current shortcomings, and safety/reliability considerations. Additionally, it provides a brief comparison between TestCafe and Playwright based on testing Syncfusion-based web UIs.

### 1. test\_main.spec.js

- Main orchestrator script.
- Executes all steps in order: login, addShipment, uploadFile, captureDocument, complyShipment, checkDiscrepancies.
- Uses strong logging (with timestamps), screenshot capture on failure, structured JSON export of failures, and error logs.
- Step wrapper (`runStep`) ensures each step is isolated and failure-tolerant.
- Strong error handling and assertion logic to verify execution status.

### 2. login.js

- Logs in to the application using credentials from `.env`.
- Uses strong assertions to ensure successful login before proceeding.

### 3. addShipment.js


- Adds a shipment using Syncfusion dropdowns and input fields.
- Has basic error checks and uses `.fill()` and `.click()` methods on specific selectors.
- Should be revisited to ensure synchronization with updated UI components.

### 4. uploadFile.js

- Automates uploading a PDF file to a shipment.
- Verifies success modal text via hard assertions (expected header & message).
- Uses `.setInputFiles()`, `.click()`, and `.waitForSelector()`.

- Ensures test doesn't continue if upload fails or if modal doesn't close.

## 5. captureDocument.js

- Automates document capture via modal and dropdown selection.
- Includes multi-step validation, modal capture, and post-action confirmation.
- Does **not** currently import or use `dropdownHandler`.
-  Needs rework — error occurs if multiple elements match the same text (strict mode violation).
- Uses strong assertions and logging to verify that the user stays on the expected page and that UI changes happen correctly.
- Implemented fallback logging in case of redirection or UI glitches.

## 6. complyShipment.js

- Automates 'Comply' button action for a shipment document.
- Waits for specific DOM elements, then clicks to trigger a backend action.
- Currently in progress or non-functional; error handling needs final validation.

## 7. checkDiscrepancies.js

- Verifies that the list of discrepancies (notes) on the page matches a reference file.
- Loads `discrepancy_notes_reference.txt`, compares to the UI-captured notes.
- Logs any missing or unexpected discrepancies.
- Uses `expect`, conditional filters, and `.innerText()` to sanitize the output.
- Strong assertion-driven design; fails test on mismatch.

## 8. config3.0.js

- Centralized config for URLs, selectors, test constants, and timeouts.
- Modular structure supporting all test steps.
- Must be updated if selectors or UI paths change.

## 9. Comparison: TestCafe vs Playwright (by Assistant)

- Playwright offers **better handling for flaky elements**, supports **parallelism**, and **headless/headed mode switching**.
- Syncfusion components (dropdowns, popups, modals) are better managed in Playwright via Playwright-specific methods like `.locator()`, `.first()`, `.waitForSelector()`, and strict mode.

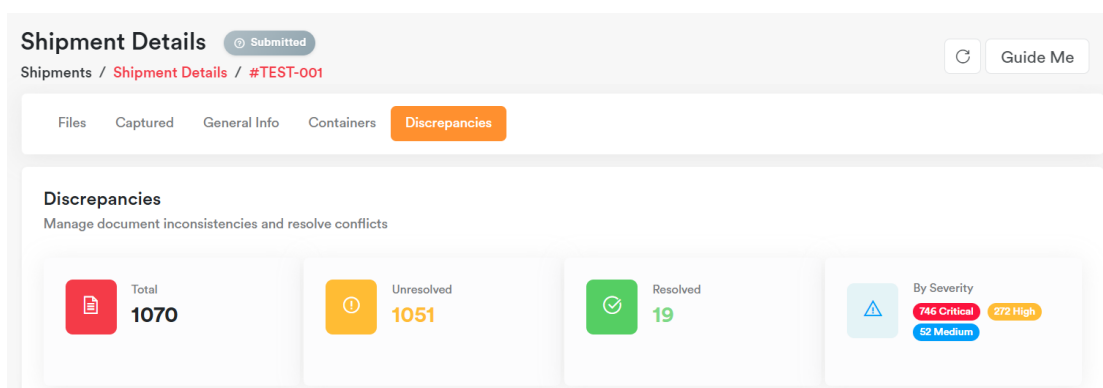
- TestCafe had trouble reliably selecting deeply nested or re-rendered dropdowns, especially within modals.
- Playwright supports robust selector chaining, has screenshot/test state export on failure, and full CI/CD integrations.
- In summary: Playwright is more CI-safe, deterministic, and expressive than TestCafe for testing Syncfusion UIs.

## 10. Final Notes

- Strong assertions (`expect`, `count`, `isVisible`, etc.) are used in **all scripts** that passed testing.
- captureDocument.js currently lacks `dropdownHandler`, which is used in other dropdown-intensive flows.
- Final flow sometimes breaks due to backend not cleaning up old states (e.g., lingering files, stale data).
- This project is a strong demonstration of modular E2E scripting with full failure reporting, but **some flows still need manual cleanup or final refactors** before being CI/CD ready.

### \*\*\*DEVELOPER NOTES\*\*\*

- While not perfect It should help whoever taking over this project with the utils and such
- There is a problem with the backend that causes deleted shipments to still be accesible by just typing their ID as link. example: even if Test-001 is deleted you can still access Test-001 by just going to this link. <https://demo.tradingdocs.ai/shipment/TEST-001> and even if you delete the shipment uploaded/compared/captured files and the discrepancies will persist. causing this issue:



The playwright-e2e file that will be uploaded to github will have more scripts than only steps3.0 which is just about testing the AI's functionality. steps2.0 focused on general website functionality control. steps file itself was just the first version of scripting.