E-commerce and Note API Testing with Playwright

This project is a comprehensive test automation suite for both UI and API testing of an e-commerce platform and a note-taking API. It leverages **Playwright with typescript** to provide reliable, modular, and efficient automated testing.

Project Overview

- **UI Testing**: Covers functionality like adding, editing, deleting products, and counter increment functionality on a web UI.
- **API Testing**: Covers operations like user registration, login, profile updates, password changes, and CRUD operations for notes.

Features

- **Playwright with typescript** for UI and API automation.
- Modular design using Page Object Model (POM) for UI elements
- Automatic test reporting and screenshot capture for failed tests.

Prerequisites

1. **Java Development Kit (JDK)**: Ensure JDK 8 or later is installed.
2. **Node.j**s : Ensure **Node.j**s last version is installed
3. **Internet Connection**: Required to download dependencies and execute API tests.

Setup

1.Clone the Repository

Write this on terminal

git clone [**https://github.com/MhmdMoustafa77/QATask.git**]

2.Install Playwright

Write this on terminal

npm install -D playwright typescript ts-node @types/node
npx playwright install

3.Playwright Browser Setup

Playwright will automatically download the necessary browsers when the tests are run. No additional setup is required.

Configuration

Playwright Configuration: playwright.config in this config file I write all utilization of project

How to Run the Tests

Running All Tests

Write this on terminal

(npx playwright test)

Running UI Tests Only

u can add only on test suit of product and counter

Running Api Tests Only

u can add only on test suit of health api and note api and user api

Test Reports

1.Location: playwright-report /index.html

2.Contents: Displays test results, including pass/fail status, execution times, and stack traces for failed tests**.**

Updating Test Data

1. **UI Tests**: Modify test data in UI test classes (e.g., product names, prices in ProductTests.spec.ts).
2. **API Tests**: Update payload data within API (UserApiTests.spec.ts, NoteApiTests.specs.ts)

Adding New Tests

1. **UI Tests**: Create new methods in existing Page Object classes (ProductPage.ts, CounterPage.ts) and call them from new test classes.
2. **API Tests**: Add new methods in the API (UserApTestsi.spec.ts, NoteApiTests. spec.ts) and create corresponding test case

Troubleshooting

1. **Playwright Browser Issues**: Ensure Playwright browsers are downloaded and accessible.

2. **API Authentication**: For API tests that require a token, ensure login credentials are valid and tokens are handled correctly.

End of document