Міністерство освіти і науки України

Івано-Франківський національний технічний

університет нафти і газу

кафедра ІПЗ

**Лабораторна робота №8**

Тема: «Огляд сучасного популярного автоматизованого інструментарію»

Варіант №1

Виконав:

студент групи ІП 21-3

Амброз О.С.

Перевірив:

Храбатин Р.І.

м. Івано-Франківськ

2025 р.

**Мета:** дослідити та порівняти існуючі сучасні інструменти для створення автоматизованих рішень на основі JavaScript.

**Хід роботи**

1. Тести на playwright:

Playwright — це сучасна бібліотека для автоматизації веббраузерів, створена компанією Microsoft. Вона дозволяє писати тести, які імітують дії користувача: відкриття сторінки, натискання кнопок, введення тексту у форми, перевірку наявності елементів на сторінці тощо. Playwright підтримує популярні браузери, такі як Chromium (Google Chrome), Firefox та WebKit (Safari), і дозволяє запускати їх як у звичайному, так і в headless (без вікна) режимі.

Робота з Playwright починається зі створення браузерного контексту і нової сторінки, на якій відбувається вся взаємодія. Скрипт переходить за заданою URL-адресою, після чого тест виконує послідовність дій: вводить дані у поля, натискає кнопки, обирає елементи тощо. Кожен крок можна супроводжувати перевірками, щоб переконатися, що сторінка поводиться очікувано.

Однією з сильних сторін Playwright є можливість очікування — бібліотека автоматично чекає, поки елементи з’являться, зникнуть або зміняться. Це знижує кількість помилок, які часто виникають у тестах через повільне завантаження сторінок чи елементів.

Playwright також дозволяє робити скріншоти, знімати відео виконання тестів, збирати лог-файли — це зручно для аналізу помилок. Підтримується тестування на мобільних пристроях, а також можливість емуляції різних розмірів екранів і геолокації.

Бібліотека підтримує різні мови програмування, включаючи JavaScript, TypeScript, Python та C#, і добре інтегрується з системами CI/CD. Завдяки простому API та гнучким можливостям, Playwright є потужним інструментом для перевірки якості вебзастосунків на всіх етапах розробки.



Рисунок 1.1 – Результат виконнання

2. Запуск тестів

Запускаємо тести за допомогою команди:

npm run test:e2e

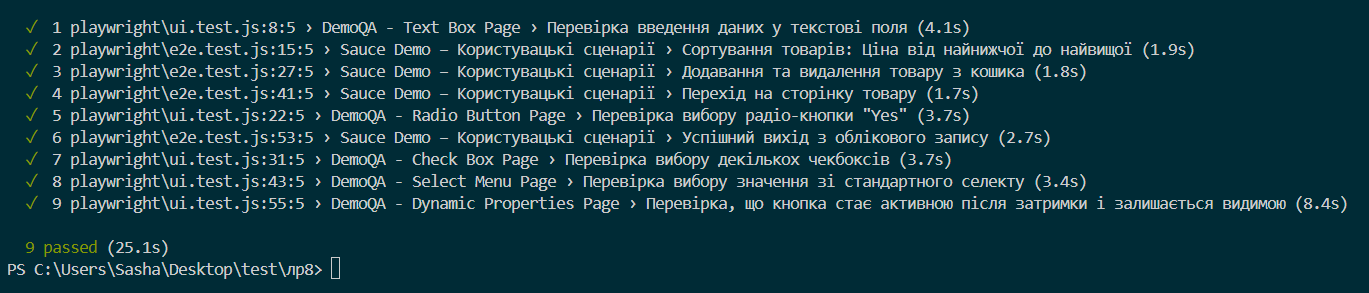


Рисунок 1.2 – Результат виконнання

3. Puppeteer

Тестування за допомогою puppeteer - це процес автоматизації взаємодії з веб-сайтом або веб-додатком у середовищі браузера. У моєму випадку, тестування використовується для перевірки коректності роботи форми додавання нерухомості через веб-інтерфейс.

Тест починається з запуску браузера за допомогою puppeteer. Відбувається відкриття нової сторінки та перехід на URL, де розташована форма додавання нерухомості. У моєму випадку це сторінка "http://localhost:3000/sale.html".

Puppeteer автоматично вводить дані в різні поля форми, такі як назва, місце знаходження, ціна, тип нерухомості та вибір файлу для фотографії. Це відображається у фрагменті коду з використанням методу page.type для введення текстових даних та page.select для вибору опцій. Приклад коду зображено на рисунку 1.3.

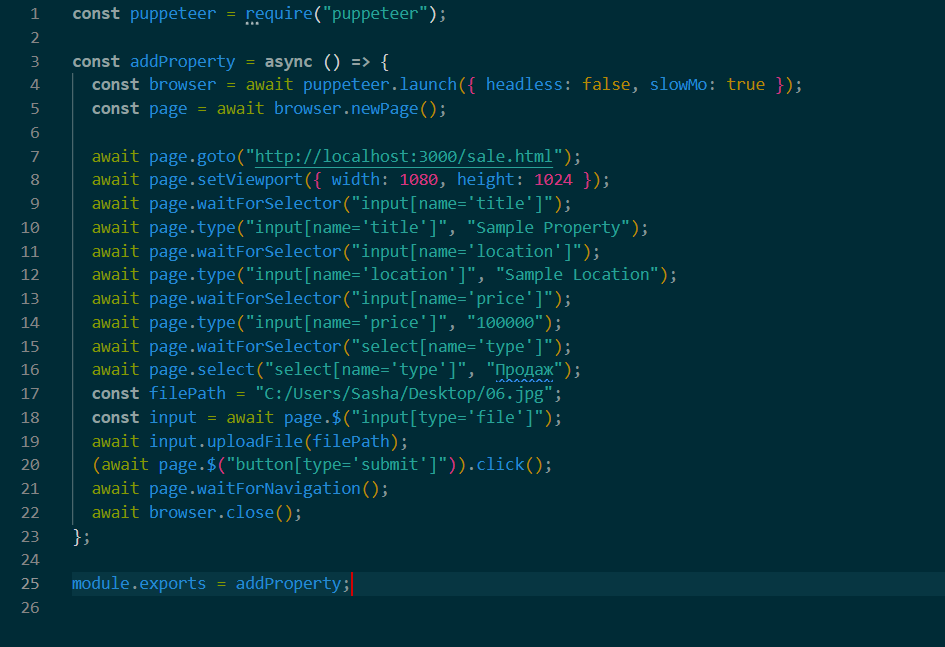


Рисунок 1.3 – Приклад коду для тестування

Після заповнення всіх полів форми, puppeteer взаємодіє з елементом кнопки "Відправити" і викликає подію click. Це імітує дію користувача, який натискає на кнопку для відправлення заповненої форми.

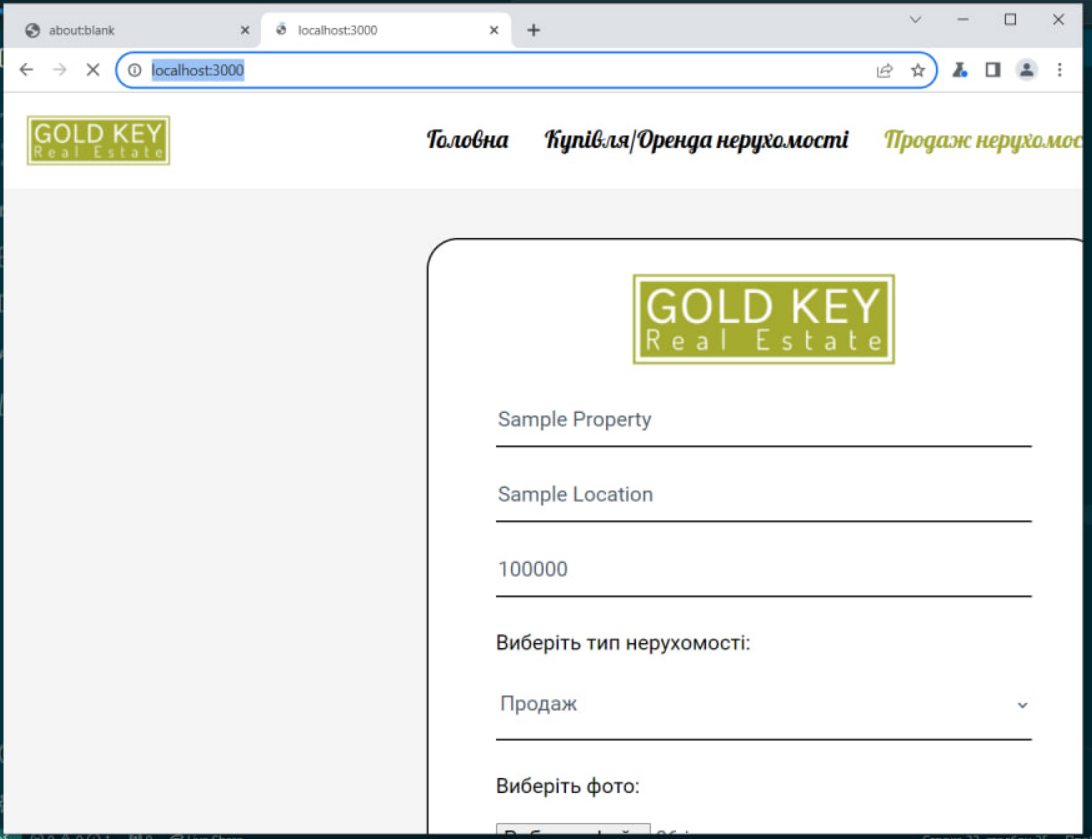


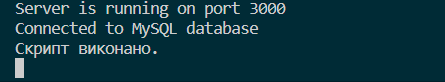
Рисунок 1.4 – Вигляд тестування puppeteer

Puppeteer чекає, доки відбудеться перехід на нову сторінку або викличеться callback-функція, що підтверджує успішне відправлення форми. Це забезпечує коректність переходу до наступної частини взаємодії з сайтом.

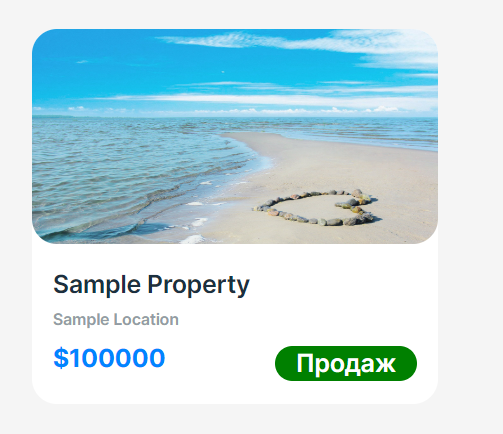
Після відправлення форми, тест проводить перевірки, щоб підтвердити, що дані були додані правильно в базу даних або що користувач отримав відповідний підтвердження на екрані. Це може включати перевірку елементів або використання інших методів для забезпечення коректності роботи.



Рисунок 1.5 – Результат виконання тесту в БД



*а) – у консолі*



*б) – у веб-застосунку*

Рисунок 1.6 (а-б) – Результат виконання тесту

У кінці тестового сценарію puppeteer закриває браузер, завершуючи взаємодію з сайтом. Це може бути важливим для звільнення ресурсів та підтримки чистоти середовища тестування.

|  |  |  |  |
| --- | --- | --- | --- |
| Інструмент | Сильна сторона | Слабка сторона | Типові задачі |
| Puppeteer | |  | | --- | | Високий контроль над Chromium, простий у налаштуванні, стабільний API |  |  | | --- | |  | | |  | | --- | | Працює тільки з Chrome/Chromium (офіційно), менша кросбраузерна підтримка |  |  | | --- | |  | | |  | | --- | | UI тести, PDF/скріншоти, веб-скрейпінг, автоматизація браузера |  |  | | --- | |  | |
| Playwright | |  | | --- | | Підтримка кількох браузерів (Chrome, Firefox, WebKit), автоочікування, багатомовність |  |  | | --- | |  | | |  | | --- | | Вищий поріг входу порівняно з Puppeteer, більший розмір бібліотеки |  |  | | --- | |  | | UI тести, кросбраузерне тестування, інтеграція в CI/CD, складна веб-автоматизація |

**Висновок:** під час виконання даної лабораторної роботи я дослідив та порівняти існуючі сучасні інструменти для створення автоматизованих рішень на основі JavaScript.

Посилання на репозиторій: <https://github.com/free-bit-dot/tests>