**Міністерство освіти і науки України**
**Національний технічний університет**
**«Дніпровська політехніка»**

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

**Кафедра** Програмного забезпечення комп'ютерних систем
(повна назва)

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## кваліфікаційної роботи ступеня

*магістра*
(назва освітньо-кваліфікаційного рівня)

| | |
|---|---|
| **студента** | *Бадая Данила Костянтиновича* |
| | (ПІБ) |
| **академічної групи** | *121м-22-1* |
| | (шифр) |
| **спеціальності** | *121 Інженерія програмного забезпечення* |
| | (код і назва спеціальності) |
| **освітньої програми** | *«121 Інженерія програмного забезпечення»* |
| | (назва освітньої програми) |
| **на тему:** | *Розробка та дослідження ефективності впровадження автоматизованого тестування веб-додатків.* |

_____*Д.К. Бадай*

| **Керівники** | **Прізвище, ініціали** | **Оцінка за шкалою** | | **Підпис** |
|---|---|---|---|---|
| | | **рейтинговою** | **інституційною** | |
| розділів кваліфікаційної роботи | | | | |
| **спеціальний** | *Проф. Бердник М.Г.* | | | |
| | | | | |

| | | | | |
|---|---|---|---|---|
| **Рецензент** | *Доц. Шедловський І.А.* | | | |

| | | | | |
|---|---|---|---|---|
| **Нормоконтролер** | *Доц. Гуліна І.Г.* | | | |

**Дніпро**
**2023**

**Міністерство освіти і науки України**
**Національний технічний університет**
**«Дніпровська політехніка»**

**ЗАТВЕРДЖЕНО:**
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)
_____ І.Г. Гуліна
(підпис) (прізвище, ініціали)
« » _____ 20 ____ Року

**ЗАВДАННЯ**
**на виконання кваліфікаційної роботи магістра**

**спеціальності** *121 Комп'ютерні науки*
(код і назва спеціальності)

**студенту** *121м-22-1* *Бадаю Данилу Костянтиновичу*
(група) (прізвище та ініціали)

**Тема кваліфікаційної роботи** *Розробка та дослідження ефективності впровадження автоматизованого тестування веб-додатків.*

## 1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

## 2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

**Об'єкт досліджень** – процес розробки веб-додатків та впровадження автоматизованого тестування..

**Предмет досліджень** – методи тестування, які можуть бути автоматизовані, такі як тестування інтерфейсу, функціональне та негативне тестування, тестування продуктивності та безпеки.

**Мета роботи** – покращити якість тестуємого веб-додатку завдяки ефектимному використанні ресурсів і підтримці постійного вдосконалення в розробці та тестуванні.

**Методи дослідження** – Для вирішення поставлених задач використані методи: порівняльний аналіз, покриття тестуванням, аналіз вигод, аналіз надійності веб-додатка.

## 3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

**Наукова новизна** – запропоновано методи покращення і спрощення ведення тестування веб-додатків та підвищення ефективності тестування.

**Практична цінність** – полягає в тому, що результат дослідження можна використовувати для тестування будь яких веб-додатків.

## 4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безпосереднє використання запропонованих методів. В результаті роботи повинна бути розроблена тестувальна стратегия для тестування веб-додатку.

## 5 ЕТАПИ ВИКОНАННЯ РОБІТ

| **Найменування етапів робіт** | **Строки виконання робіт (початок – кінець** |
|---|---|
| Аналіз теми та постановка задачі. | 15.09.2023-30.09.2023 |
| Дослідження існуючих підходів та інтеграція найбіль оптімального рішення. | 01.10.2023-05.11.2023 |
| Створення Page Object моделі тестування для взаємодії з веб--додатком. | 06.11.2023-20.01.23 |

## 6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

**Економічний ефект** від реалізації результатів роботи очікується позитивним завдяки розробці ефективних алгоритмів тестування для веб-додатку.

**Соціальний ефект** від реалізації результатів роботи очікується позитивним, завдяки полегшенню роботи тестувальників та розробників веб-додатку.

Завдання видав _____ *Бердник М. Г.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Бадай Д.К.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *15.09.2023 р.*

Термін подання дипломного проекту до ЕК *24.12.2023*

# РЕФЕРАТ

Пояснювальна записка: 89 с., 35 рис., 3 дод., 26 джерел.

Об'єкт дослідження: процес розробки веб-додатків та впровадження автоматизованого тестування.

Предмет дослідження: методи тестування, які можуть бути автоматизовані, таких як тестування інтерфейсу, функціональне та негативне тестування, тестування продуктивності та безпеки.

Мета роботи: покращити якість продукту і ефективного використання ресурсів і підтримка постійного вдосконалення в розробці та тестуванні.

Методи дослідження: для вирішення поставлених задач використані методи: порівняльний аналіз, покриття тестуванням, аналіз вигод, аналіз надійності веб-додатка.

Новизна отриманих результатів: визначається тим, що розробляються та впроваджуватся нові методи автоматизації тестування, які можуть ефективніше та швидше виконувати тестові сценарії, а також вивчення та оцінка можливостей відновлення автоматизованих тестів після змін в коді веб-додатку.

Практична цінність результатів: полягає в тому, що тестування веб-додатків може бути важливою частиною для розробників та тестувальників та менеджерів проектів та організацій у цілому. Наприклад: підвищення якості продуку, зменшення часу на тестування та випуск веб-додатків.

Область застосування: ця галузь велика, оскільки веб-додатки є ключовим елементом інтернет-технологій нашого часу і важливі для підприємств користувачів та розробників.

Значення роботи та висновки: Впровадження автоматизованого тестування веб-додатків виправдовується на практиці через покращення якості продукту, оптимізацію використання ресурсів та підвищення продуктивності розробницьких команд.

Прогнози щодо розвитку досліджень: з урахуванням швидкого розвитку технологій та зростаючого значення автоматизованого тестування, можна очікувати, що дослідження в даній темі буде активно розвиватися та вдосконалюватися у багатьох напрямках.

Список ключових слів: програма, комп’ютер, проектування, юнит тест, вкладка, веб-додаток, автоматизація.

# ABSTRACT

Explanatory note: 89 pp., 35 figs., 3 appendices, 26 sources.

Object of research: the process of developing web add-ons and introducing automated testing.

Subject of research: when considering various testing methods that can be automated such as front-end testing, functional and negative testing, productivity and unemployment testing.

Purpose of Master's thesis: this reduces the quality of the product, ensures efficient utilization of resources and encourages continuous perfection in development and testing.

Research methods. to achieve the best objectives, we use the following methods: benchmark analysis, test-based analysis, benefit analysis, and reliability analysis of the web application.

Originality of research consists of the obtained results is determined by the fact that new test automation methods are developed and implemented, which can more efficiently and quickly execute test scenarios, as well as studying and evaluating the possibilities of restoring automated tests after changes in the web application code.

Practical value of the results consists of the findings is that web application testing can be significant for developers, testers, project managers, and organizations in general. For example: improving product quality, reducing web application release cycle time, increasing web application user trust.

Scope of application. this scope is broad because web applications are a key element of Internet technologies and are important to businesses, users, and developers.

The value of the work and conclusions. The implementation of automated testing of web applications is justified in practice by improving product quality, optimizing the use of resources and increasing the productivity of development teams.

Research forecast and development. given the rapid development of technology and the growing importance of automated testing, it can be expected that research in this topic will actively develop and improve in many directions.



Keywords: program, computer, design, unit test, tab, web app, automation.

**ЗМІСТ**

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Unit Test – модульний тест;

Smoke testing – димове тестування;

WB – White box;

GB – Grey box;

ОС – операційна система;

BB – Black box;

XPath – мова запитів для вибору вузлів з XML документів;

CI/CD Continuous Integration (CI) та Continuous Delivery (CD);

POM – Page Object Model;

BDD – Behavior-Driven Development;

TDD – Test-Driven Development;

API – Application Programming Interface.

# ВСТУП

Тема автоматизованого тестування є важливою з двох причин. Вона ставить за мету розгляд відомих методів та інструментів для автоматизованого тестування веб-додатків та прагне вивчити ефективність впровадження цих засобів на практиці. Завдяки розвитку різноманітних фреймворків, інструментів та методологій, автоматизоване тестування набуває все більшої популярності в розробницьких колах. Проте, виникає необхідність в глибокому розумінні ефективності цього підходу та розробці стратегій для оптимального впровадження його в процес розробки веб-додатків. Данна тема сприяє сучасному впровадженню Agile методологій розробки що дозволяэ прискорювати цикл розробки та випуску веб-додатку через автоматизовані тестові сценарії. Також це дозволяє вявляти дефекти на ранніх етапах розробки веб-додатку. Забезпечує швидке виконання тестів під час змін у коду веб-додатку. Виключає необхідність ручного виконання рутинних тестових тасок що дозволяє нам збільшити увагу на інших задачах. Знижує витрати на тестування. Дозволяє швидко та ефективно проводити тести на витривалість у веб-додатку.

Мета дослідження: покращити якість веб-додатку завдяки ефективного використання ресурсів і підтримка постійного вдосконалення в розробці та тестуванні.

Об’єкт дослідження: об'єктом дослідження є процес розробки веб-додатків та впровадження автоматизованого тестування.

Предмет дослідження: це методи тестування, які можуть бути автоматизовані, таких як тестування інтерфейсу, функціональне та негативне тестування, тестування продуктивності та безпеки.

Методи дослідження: аналіз літератури, проведення практичних експриментів, системний аналіз, аналіз трендів.

Новизна запропонованих рішень: визначається тим, що розробляються та впроваджуватся нові методи автоматизації тестування, які можуть ефективніше та швидше виконувати тестові сценарії, а також вивчення та оцінка

можливостей відновлення автоматизованих тестів після змін в коді веб-додатку.

Практичне значення: полягає в тому, що тестування веб-додатків може бути важливою частиною для розробників та тестувальників та менеджерів проектів та організацій у цілому. Наприклад: підвищення якості продуку, зменшення часу на тестування та випуск веб-додатків.

Особистий внесок автора:

- результати роботи отримані автором самостійно;
- вибір методів дослідження та технологій реалізації;
- імплементація POM для зручного тестування веб-додатку;
- оцінка отриманних результатів.

Робота складається з вступу, трьох розділів, висновків, списку використаних джерел і трьох додатків. Загальний обсяг роботи становить 89 сторінок, в тому числі 35 зображень, і список використаних джерел.

# РОЗДІЛ 1

# АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Загальні відомості з автоматизації тестування додатків

Тема автоматизації є важливою у зв'язку зі стрімким розвитком веб-розробки та зростанням вимог до якості програмного забезпеченнях [3]. Автоматизоване тестуваня стає визначальним етапом в розробці для забезпечення стабільності, надійності та швидкого впровадження нових функціональностей веб-додатків.

Автоматизоване тестування веб-додатків передбачає використання спеціалізованих інструментів і програмного забезпечення для автоматичного створення сценаріїв тестування та аналізу результатів. Метою є забезпечення високої якості, ефективності та надійності програмного забезпечення.

У зв’язку зі зростанням складності веб-додатків і потребою швидкого випуску нових функцій автоматизоване тестування стає ключовим елементом процесу розробки. Це дозволяє пришвидшити розробку, зменшити ймовірність помилок і покращити загальну якість вашого продукту.

У сучасному світі існують різні методи та інструменти для автоматизованого тестування, такі як Selenium, Appium, JUnit, TestNG тощо [2]. Вибір конкретних інструментів залежить від характеристик веб-додатку та вимог до тестування.

Тестування веб-додатків включає різні етапи, такі як планування тестування, розробка тестового сценарію, генерація тестових даних, виконання тесту, аналіз результатів і звітність. Кожен етап вимагає уваги та методології.

З появою нових технологій, таких як SPA (Single Page Application), PWA (Progressive Web Application) і використання різноманітних бібліотек і фреймворків JavaScript, тестування стало складнішим.

Зростання використання мобільних пристроїв призвело до необхідності автоматизованого тестування мобільних версій веб-додатків, що вимагає спеціальних підходів.

Автоматизоване тестування має включати інструменти для виявлення потенційних вразливостей і гарантувати, що ваш веб-додаток є високозахищеним.

Відтворення умов тестування в реальному часі та миттєве виявлення помилок стають ключовими в середовищах DevOps і безперервного розгортання.

Веб-розробка є ключовою галузю в інформаційних технологіях, і вона постійно розвивається. Зростання обсягів веб-додатків та їх складність вимагають високоефективного тестуваня для забезпечення надійності та якості продуктів.

Освіта та навчання досягли значного прогресу. Завдяки розвитку онлайн-курсів і навчальних платформ фахівці мають можливість вивчати сучасні техніки та методи автоматизованого тестування, що допоможе виховати нове покоління професіоналів.

На великих підприємствах із великими обсягами даних і складними корпоративними системами важливо мати ефективні засоби тестування. Впровадження автоматизованого тестування може забезпечити високу якість масштабного програмного забезпечення.

Розробка та дослідження ефективності впровадження автоматизованого тестування веб-додатків є важливою частиною циклу розробки. Оскільки складність додатків і вимоги до їх якості постійно зростають, використання засобів автоматизації стає необхідним. Змінний технологічний ландшафт вимагає постійного вдосконалення методів та інструментів для забезпечення валідності та надійності тестування.

Однією з основних цілей розробки є підвищення якості програмних продуктів. Використання автоматизованого тестування може виявити та виправити дефекти до випуску продукту, допомагаючи покращити взаємодію з користувачами та задоволеність клієнтів.

Ефективність самоперевірки безпосередньо впливає на цикл розробки. Швидке виявлення та виправлення помилок дозволяють прискорити процес розробки та забезпечити швидке впровадження нових функцій.

Ще однією важливою функцією автоматизованого тестування є економія часу розробника. Автоматизація тестування дозволяє скоротити час, витрачений на ручне тестування, і надає експертам можливість більше зосередитися на написанні якісного коду та вирішенні інших важливих завдань.

Важливим аспектом є інтеграція автоматизованого тестування в інтегровані середовища розробки та системи контролю версій. Це дозволяє автоматично виконувати тестування під час розробки, просуваючи концепцію безперервного тестування та забезпечуючи безперервний контроль якості коду.

Сфери застосування для розробки та дослідження ефективності автоматизованих тестів широкі та різноманітні. Він охоплює різні галузі, від фінансових установ і медичних установ до торгових платформ і соціальних мереж. Впровадження цих технологій є актуальним завданням для будь-якої компанії, яка прагне надавати своїм користувачам якісне програмне забезпечення.

Особливо у фінансовому секторі, де безпека та надійність високо цінуються, автоматизоване тестування відіграє життєво важливу роль у забезпеченні правильного функціонування фінансових послуг та операцій.

У медичній сфері автоматизоване тестування веб-додатків може використовуватися для перевірки правильності роботи електронної системи медичних записів, забезпечуючи тим самим точність і безпеку обробки інформації про пацієнтів.

В онлайн-комерції автоматизоване тестування може допомогти впровадити нові функції та забезпечити правильну роботу на платформі електронної комерції.

У світі соціальних мереж, де зручність і стабільність є критично важливими, автоматизоване тестування може допомогти виявити та виправити помилки, які виникають під час взаємодії користувачів із платформою.

Ось кілька прикладів того, як автоматизоване тестування використовується в різних галузях. Ця технологія універсальна і важлива для будь-якої компанії, що розробляє або використовує веб-додатки.

Таким чином, розробка та дослідження ефективності реалізацій автоматизованого тестування веб-додатків є дуже актуальною темою в сучасному інформаційному середовищі. Його цілі включають покращення якості програмного забезпечення, оптимізацію циклів розробки та ефективне використання робочого часу розробників. Області застосування охоплюють різні галузі впровадження автоматизованого тестування стало ключовим елементом стратегії розвитку компанії з надання якісного надійного програмного забезпечення.

Головна мета кваліфікаційної роботи це створити автотести та наглядно показати що автоматизація веб-додатків має корисний вплив. У вік цифрових технологій необхідне ефективне керування електронними ресурсами. Розробка спрямована на створення тестових сценаріїв, які можна ефективно автоматизувати, щоб прискорити розробку нового функціонала та підтримку вже існуючиго.

Автоматизоване тестування, на відміну від ручного, спрощує процес виявлення багів за допомогою спеціальних програм, чим скорочує витрати й час на цикл тестування.

Іншими словами, це дозволяє отримати готовий програмний продукт без багів в коротші терміни, ніж при ручному тестуванні.

### 1.2. Огляд сучасних підходів до автоматизованого тестування веб-додатків

Зараз веб-додатки вимагають дуже високої якості та бажано випуску випуску програмного забезпечення тому автоматизоване тестування стало невід'ємною складовою для досягнення цих цілей. Наразі є декілька підходів до автоматизованого тесутвання веб-додатків.

POM (рис 1.1) - шаблон проєктування, що використовується при написанні автоматизованих тестів, який дає змогу абстрагуватись від окремих елементів HTML і інкапсулювати їх у функції доступу до елементів інтерфейсу вищого рівня, як їх бачить користувач [11].

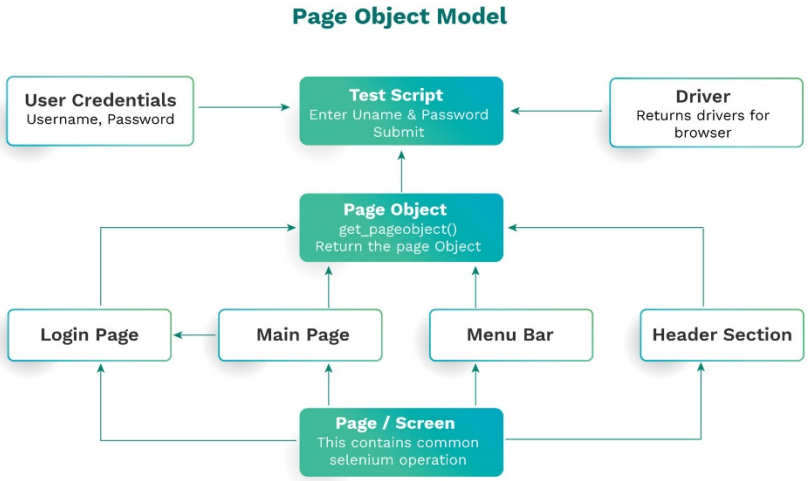



Рис 1.1. Графічне представлення POM

BDD - це такий підхід до створення програмного забезпечення, що зосереджується на тому, як програма має поводитися в конкретних ситуаціях. Це як історія, яку розповідає програма, описуючи свою поведінку (Рис 1.2).

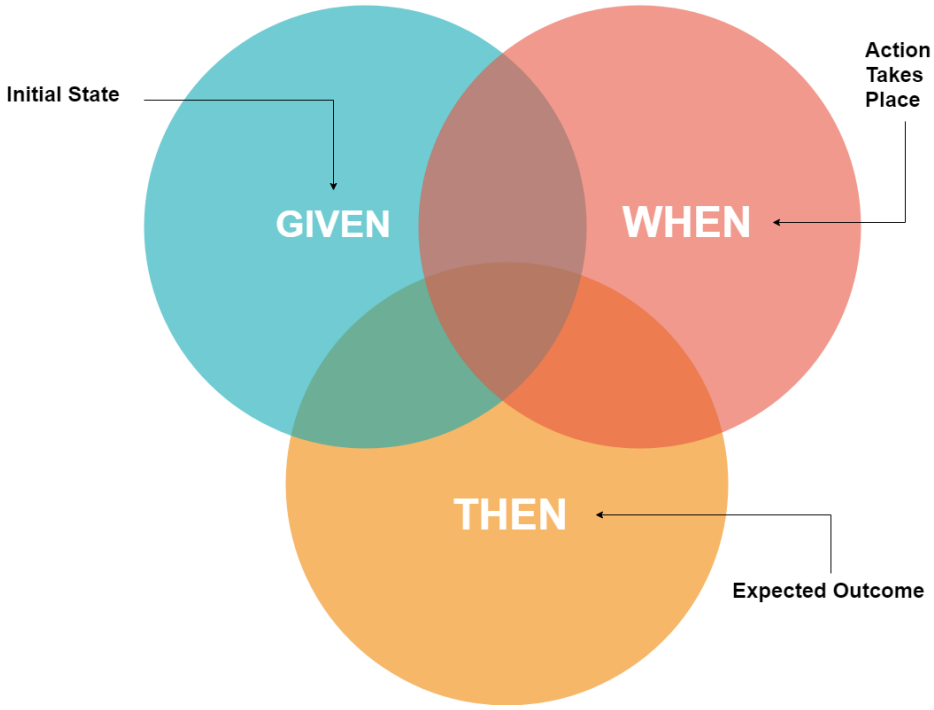



Рис 1.2 Графічне представлення BDD

TDD - це один із сучасних методів створення програм, при якому процес розробки складається з ряду циклів, що повторюються, при виконанні яких відбувається поступове налагодження і поліпшення програмного коду окремого блоку або модуля. Відмінною особливістю цього підходу від традиційних методів програмування є попередня розробка тестів до створення програмного коду програми (Рис 1.3).

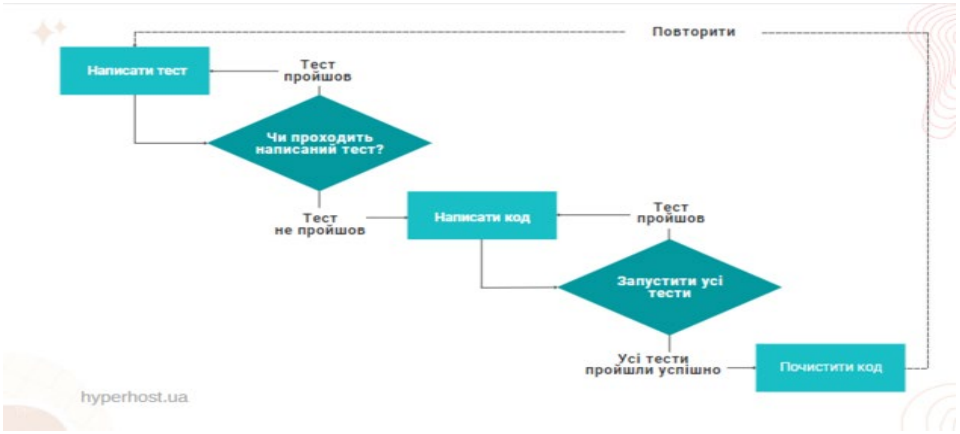



Рис 1.3 Графічне представлення TDD

Keyword-Driven testing - це метод сценаріїв, у якому файли містять ключові слова, пов'язані з даними, що тестуються, додатком. Ці ключові слова описують набір дій, необхідних для виконання визначеного кроку. Тест на основі ключових слів складається з ключових слів високого та низького рівня, включаючи аргументи ключових слів, які складені для опису дії тестового прикладу. Це також називається тестуванням на основі таблиці або тестуванням на основі дій (Рис 1.4).

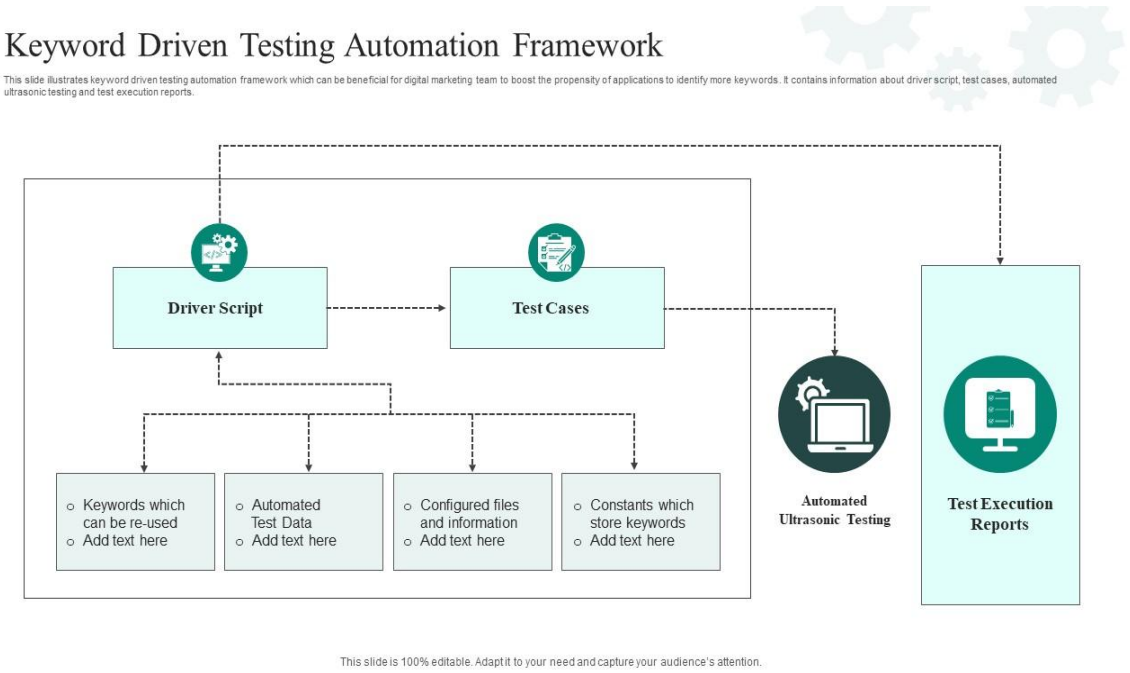



Рис 1.4 Графічне представлення Keyword Driven Test

Record and Playback - Цей підхід дозволяє розробникам записувати їх взаємодію з веб-додатком, а потім відтворювати цю взаємодію для проведення тестів. Він популярний завдяки простоті використання, але може бути менш гнучким для складних сценаріїв.

API Testing Framework - це форма тестування, яка повністю автоматизується сторонньою програмою, коли ви тестуєте API онлайн, ви можете перевірити все, від безпеки та продуктивності до функціональності та ефективності використання ресурсів. Процес працює шляхом запуску тестової програми з API і простого очікування результатів, оскільки тест встановлює якість API. Деякі програми автоматизованого тестування підтримують спеціальні тести, такі як визначення конкретних областей для тестування, високі рівні конфігурації та аналіз результатів.

## 1.3. Аналіз інструментів для автоматизованого тестування веб-додатків

Перед тим як імплементувати автоматизацію потрібно обрати інструмент за допомогою якого ми будемо тестувати веб-додаток. У

більшості випадків вони використовуються для тестування веб-додатків, але цим не обмежуються.

Стек популярних інструментів:

- selenium WebDriver;
- appium;
- cypress;
- jenkins;
- testNG;
- postman.

Особливості інструментів:

Selenium взаємодіє з такими браузерами як Chrome, Firefox, Safari, Edge та має дуже широкий спектр розришень та різноманітні інтеграції з фреймворками тестування як Junit або TestNG. Він підтримує різні мови програмування та платформи (C#, Java, Python), що робить його дуже гнучким.

Appium дуже цікавій інструмент для розробки якщо ми тестиємо мобільний додаток на IOS або Android. Він має підтримку крос-платформеного тестування. Він підтримує різні мови програмування та платформи, що робить його дуже гнучким.

Cypress був розроблений спеціально для тестування веб-додатків має дуже високу швидкість виконання та прост у використанні. Також має підтримку асинхронного коду. Він дозволяє виконувати тести в реальному часі та надає багато функцій для легкого взаємодії з елементами сторінок.

Jenkins це інструмент для CI\CD яка має автоматизовану збірку та розгортання веб-додатків. Також має велику кількість плагінів.

TestNG це фреймворк на базі JUnit з додатковими функціями. Має підтримку параметризованих тестів та групування тестів. Він підтримує анотації для конфігурації тестів та взаємодії з іншими фреймворками.

Postman має фокусіровку на тестуванні API. Може створювати авто кейси для RESTful API.

TestComplete платформа є основою для створення автоматичних веб-застосунків, інструментів тестування програмного забезпечення та автоматизації мобільних додатків. створює надійні автоматизовані тести графічних інтерфейсів для маси стаціонарних, мобільних, онлайн та стислих додатків.

Apache Jmeter це інструмент для виконання тестів продуктивності веб-додатків. Він дозволяє моделювати навантаження на сервер та оцінювати продуктивність веб-додатків під різними умовами.

Стратегії автоматизованого теустування веб-додатків: тестування на ранніх етапах: це тестування може виявити та усунути помилки на ранній стадії, таким чином зменшивши витрати на їх виправлення.

Інтеграція та впровадження CI/CD: автоматичне тестування інтегровано в процес CI/CD, що дозволяє автоматично запускати тести для кожної зміни коду та автоматично впроваджувати нові функції.

Генерація тестових даних:

- використання автоматизованих інструментів для створення тестових даних спрощує процес написання тестів і підтримки тестового середовища;
- автоматизоване тестування сучасних веб-додатків є складним і динамічним процесом, який вимагає поєднання методів, інструментів і стратегій;
- використовуючи новітні технології та методи, розробники можуть ефективно та надійно тестувати веб-додатки, забезпечуючи високу якість і задоволення користувачів.

Ці інструменти дозволяють розробникам автоматизувати умови тестування та оптимізувати процедури тестування для забезпечення високої якості програмного забезпечення. Однако вибір конкретних інструментів має бути основан на конкретних вимогах веб-додатку та на розсуд тестувальників.

## 1.4. Постановка задачі

Розробка та дослідження впровадження автоматизованого тестування веб-додатків є важливою галуззю інформаційних технологій, оскільки створює та вивчає методи та інструменти для автоматизації процесу тестування веб-додатків. Метою цієї розробки є підвищення якості програмного забезпечення, виявлення та усунення помилок на ранніх стадіях розробки, ефективне використання робочого часу розробників, забезпечення стабільності та надійності веб-додатків.

У даній кваліфікаційній роботі потрібно зробити:

- аналіз потреб;
- вибір інструментарію для тестування;
- розробка тестових сценаріїв;
- налаштування тестової структури;
- розробка автотестів;
- логування та аналіз отриманих результатів;
- оцінка ефективності.

Призначення написаних автотестів:

- вирішення повторювані завдань для економії часу;
- простота підтримки;
- підвищення ефективності;
- виключити помилки внаслідок «людського фактора»;
- забезпечити прозорий контроль над процесом тестування;
- скоротити час виходу готового додатку на ринок;
- поліпшити процес розробки.

## 1.5. Висновки

В розділі було розглянуто загальні відомості, аналіз інструмнтів а також постановка задачі. Оцінка інструментів автоматизованого тестування веб-додатків є важливим кроком у забезпеченні якості програмного забезпечення. Оскільки вимоги до швидкості розробки зростають і мережеві технології

змінюються, вибір правильних інструментів стає все більш важливим. Кожен інструмент та методологія розробки автотестів для веб-додатку має свої переваги та недоліки але вибор залежить лише від конкретних вимог які поставлени перед тестувальником додатка.

# РОЗДІЛ 2

# ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

## 2.1. Функціональне призначення створення автотестів

Результатом виконання кваліфікаційної роботи повинни бути автоматизованні тести для веб-додатку, завдяки яким ми зможемо проаналізювати галузь автоматизації та отримати висновки які покажуть яку перевагу має автоматизація над манульним тестування та навіщо тестування додатків поторібно у сучасному житті. Якщо проєкт великий, зростає, в його складі кілька підсистем і «ручні» тест-кейси вже налічують кілька сотень, автоматизація дозволить підвищити продуктивність тестувальника, який не буде витрачати тижні на перевірку тест-кейсів. Не тільки великий проєкт, але і велика команда програмістів потребує автоматизації тестування, щоб прискорити виявлення багів при взаємодії різних модулів коду та оперативно їх виправити. Тестування — необхідний етап створення продукту. Автоматизація дозволяє полегшити й прискорити цей процес. Але не всі види тестування потребують автоматизації, а тільки ті, які засновані на діях, що повторюються. Існує велика кількість типів тестування, і вибір певного варіанту залежить від поставлених завдань та очікуваного результату. Для цього розробляється стратегія автоматизації.

Вибір тієї чи іншої стратегії залежить від того, з яким проєктом стикається компанія-тестувальник. Якщо автоматичного тестування (АТ) не проводилося, точні цілі не були поставлені, тоді рекомендується почати з підготовчих етапів, уважно поставитися до вибору інструментів, і починати роботу з верхнього рівня, не сильно заглиблюючись в АТ певних модулів.

Функціональне призначення для створення автотестів веб додатка визначає кілька пунктів:

- взаємодія із елементами інтерфейсу. Будь-яка можливість ідентифікації з різноманітними елементами веб-сторінки. Наприклад: кнопки, тексти, веб-форми, таблиці та зображення;
- робота з даними. Введення або витягування даних з форм або баз даних для перевірки правильності обробки та зберігання інформації;
- создання різних типів тестів. Наприклад: функціональні, регресійні, інтераційні, смоук тести;
- підтримка роботи з різноманітними веб-браузерами. Тестування веб-додатку на різних браузерах. Таких як Google Chrome, Firefox, Opera, Safari, Edge;
- логування інформації про помилки. Логування та виведення інформації про помилки завдяки якої розробники зможуть швидше та легше виправити помилки, дефекти веб-додатку;
- графічне представлення результатів. Завдяки цьому процесу ми забезпечуємо швидкий аналіз дефектів веб-додатку.

## 2.2. Процес автоматизації тестів

Перший етап: автотест запускається на основі проведених ручних тестів з перевіреними сценаріями. Для автоматизації процесу необхідно обрати інструмент тестування в залежності від типу продукту: веб-сайт, мобільний додаток, API і т.д. При його виборі звертаємо увагу на відповідність бюджету (скільки компанія готова заплатити за нього), на технічні характеристики й підтримку технологій, що використовуються в продукті; на вимоги до кваліфікації фахівців, що працюють з даними інструментом і на якість звітів, який генерує цей інструмент.

Другий етап: визначаємо обсяг автоматизації. Обсяги залежать від того, яка вироблена стратегія продукту. Якщо це основний продукт, то краще забезпечити максимальне покриття автоматичними тестами. Якщо це прототип, то тут велику роль відіграють терміни, а не якість продукту.

Третій етап: розробляємо стратегію і план автоматизації. Для цього проєктується інфраструктура для автоматизації (готуються необхідні стенди), затверджується графік запуску сценаріїв. Перед запуском автоматичних тестів йде підготовка тестових даних. Після виконання тестів, аналізу звіту випробувань і фіксації помилок йде відстеження, виправлення і повторне тестування.

Четвертий етап: обслуговуємо тести для перевірки якості автоматизації. Це необхідно для підвищення ефективності вже наявних сценаріїв і при розробці нових.

### 2.3. Засоби та інструменти для автоматизованого тестуваня

При створенні додатку був виконан аналіз інструментів для тестування. Нижче наведено кілька інструментів для автоматизації тестування:

- Visual Studio 2022;
- Selenium WebDriver;
- C# Test Automation Framework (Nunit).

На етапі проектування автотестів був обраний Page Object Model.

POM – Page Object Model – це шаблон дизайну, який широко використовується в автоматизації тестів, що створює сховище об'єктів для елементів вебінтерфейсу.

Модель PageObject – зараз дуже популярний фреймворк для автоматизації тестування, багато компаній працюють з ним через його простоту обслуговування тестів і зниження дублювання коду.

Згідно з цією моделлю для кожної вебсторінки у програмі повинен бути відповідний клас сторінки. Цей клас Page визначає WebElements цієї вебсторінки, а також містить методи Page, які виконують операції з такими WebElements. Назви цих методів слід вказувати відповідно до завдання, яке вони виконують. Вони повинні відповідати діям, які вони виконують, наприклад, метод очікування, поки елемент з'явиться – waitForPaymentScreenDisplay()

Цей клас Page буде шукати всі WebElements на сторінці й також міститиме методи для роботи з ними.

Такий шаблон вперше застосували в проекті WebDriver, який пізніше об'єднався з Selenium.

Головною перевагою моделі є те, що у випадку зміни структури будь-якої сторінки, не потрібно заново складати тести, досить буде змінити код для об'єкта сторінки.

Отож, модель Page Object надає наступні переваги:

- чіткий поділ між кодом тестів і кодом специфічним для сторінки (таким як локатори);
- POM дозволяє при змінах в дизайні сторінки - змінювати відповідний код тільки в одному місці;
- Page Object Patten повідомляє елементи окремо від реалізації тесту. Ця концепція робить код є більш зрозумілим;
- код стає меншим і він більш оптимізований. Його можна повторно використати;
- методи отримують більш реальні імена та відображають виконану дію на UI, наприклад, gotoHomePage().

### 2.4. Технології та мови програмування

C# – проста, потужна, статично типізована, об’єктно орієнтована мова програмування від компанії Microsoft. C# входить до сімейства мов програмування C, синтаксис мови буде знайомим програмістам, що працювали з C, C++, Java та JavaScript.

Перша версія мови C# була створена в 1998-2001 роках, групою інженерів Microsoft під керівництвом Андреса Гейлсберга та Скотта Вільтаумота, як основна мова програмування платформи Microsoft .Net [7].

C# увібрав в себе найкращі властивості попередників – мов C, C++, Modula, Object Pascal, спираючись на практичний досвід їх використання. Деякі проблематичні моделі, що до цього використовувались у мовах програмування,

зокрема множинне спадкування класів(яке використовується у мові C++), були свідомо виключені.

В багатьох відношеннях мова C# дуже схожа на Java, це відображено в синтаксисах та основних поняттях цих мов програмування.

Назва мови C# трактується як наступне покоління розвитку C++, а символ # – символізує “++++”. Спочатку в назві фігурував дієз - “♯”(англійською sharp), однак через відсутність цього символу на клавіатурі, використовується знак для позначення номеру “#”. Загалом назву мови можна позначати обома символами [21].

Розробка мови C# розпочалася в грудні 1998 року, та готувався до випуску разом з продуктами групи Millenium. Проект мав назву COOL(C-style Object Oriented Language), та розроблявся як аналог Java від компанії Oracle. C# був анонсований, широкому загалу, в 2000 році, як основна мова платформи Microsoft .Net Framework. В цьому ж році з’явилася перша загальнодоступна бета-версія.

Перша фінальна версія мови програмування C# була випущена в 2002 році разом з середовищем інтегрованої розробки програмного забезпечення Visual Studio .Net.

HTML – На початку свого шляху, майже, кожен думає, що HTML - це мова програмування. Але це не так. Зараз я розповім тобі, що таке HTML.

Для початку уяви, що HTML це мова, якою спілкується твій браузер. Тобто, це все таки мова, але не мова програмування.

HTML - скорочення від "HyperText Mark-up Language" - перекладається як "Мова розмітка гіпертексту" (Гіпертекст - це текст, що не послідовно зв'язаний з іншими документами, тобто у вас є змога з першої сторінки документу перейти на останню). Іншими словами HTML - це мова розмітки, або ще один спосіб зберігання інформації. За допомогою HTML ти позначаєш текст, вказуючи своєму веб-переглядачу, як він має розуміти позначений текст, так само як і на жорсткому диску інформація зберігається в блоках, кластерах, секторах,

доріжках і тільки за допомогою, такої, визначеної структури твій комп'ютер розуміє, що треба, а що не треба зчитувати.

У HTML текст позначається за допомогою теґів. Кожен HTML документ буде складатися з деякої групи елементів, де кожен елемент буде визначатися (починатися та закінчуватися) певним теґом (Для деяких елементів кінцевий теґ не є обов'язковим). Теґ — це назва елементу, записана у кутових дужках (< >)

Кожен HTML теґ має свою унікальну назву з визначеним синтаксисом, яка записується латинськими літерами і не чутливий до регістру.

CSS – (абревіатура від Cascading Style Sheets, що в перекладі означає каскадні таблиці стилів) - це спеціальна мова (мова стилів), за допомогою якої описують вигляду документів (як і де відображати елементи веб-сторінки), написаних мовами розмітки даних. Найчастіше CSS використовується для документів, котрі розмічені мовою HTML, XHTML та XML.

Зараз HTML в чистому вигляді має дуже обмежений набір інструментів, що не дозволяє вирішувати ті чи інші дизайнерські та функціональні замисли веб-ремісників. Ну ось хоч би, до прикладу, взяти початкове запитання всіх веб-ремісників "Як прибрати підкреслення у посиланні?" Або "Як змінити стиль посилання, при наведенні на нього курсора?" За допомогою лише одного HTML такого зробити не вдасться!. А таких запитань безліч. Тут й приходить на допомогу CSS, який вирішує більшість завдань, що відносяться до стильового оформлення сторінки.

Одна з головних переваг використання CSS - це можливість розділити зміст сторінки від її оформлення. Таке розділення дозволило покращити сприйняття та доступність змісту, забезпечити більшу гнучкість та контроль за відображенням змісту в різних умовах, зробити зміст більш структурованим та простим, прибрати повторення та ін. Власне це ж і була основна мета створення цієї технології [20].

## 2.5. План дослідження та процедури імплементації

Різноманітність фаз і процесів можуть бути включені в стратегію дослідження та процедури впровадження для автоматизованого тестування веб-додатків, щоб переконатися в життєздатності та ефективності автоматизації в даному проекті. Нижче наведено огляд, який може слугувати основою для дослідження та практичного впровадження автоматизованого тестування веб-додатків:

- визначення мети автоматизації;
- оцінка ефективності;
- визначення обсягу автоматизації;
- вибір інструментів;
- планування ресурсів;
- налагодження тестового оточення;
- валідація та верифікація тестів;
- оцінка результатів та виявлення дефектів.

## 2.6. Обґрунтування та організація вхідних та вихідних даних

Вхідними даними додатку є інформація про функціональність та технічні особливості веб-додатка. Це допомогає нам визначити які тести потрібно автоматизовувати. Усі тести мають строгу та логічну структуру групування (фукнціональне тестування, регресійне, нагрузочне)

Вихідними даними є дані про стан пройдених тестів під час виконання.

## 2.7. Висновок

Підсумовуюче усе вище сказане. Кожна розгянута  модель має свої переваги та недоліки але PageObject Factory була обрана за наступними критеріями:

- зручна модель управління сторінками що позволяє швидко адаптуватися до змін у веб-додатку;
- читабельний код який сприяє на швидкість розробці особливо в команді що також поліпшує роботу с рефакторінгом тестів;
- використаний дизайн дозволяє забеспечувати нам швидку адаптацію до змін в интерфейсі веб-додатку;
- данна модель допомогає нам уникнути дублювання коду.

# РОЗДІЛ 3
# ІМПЛЕМЕНТАЦІЯ ТЕСТІВ ДО ВЕБ-ДОДАТКУ

## 3.1. Алгоритм роботи зробленого тестувального дизайна

Шаблон дизайну PageObject моделює області інтерфейсу користувача як об’єкти в тестовому коді, який також можна розглядати як сховище об’єктів для елементів веб-інтерфейсу користувача. Функціональні класи ( PageObjects ) у цьому дизайні представляють логічний зв’язок між сторінками програми. Кожен клас називається PageObjects і повертає інші PageObjects для полегшення потоку між сторінками. Клас Page Object відповідає за пошук WebElements цієї сторінки, а також містить методи, які виконують операції над цими WebElements.

Оскільки PageObjects повертаються, стає необхідним моделювати як успішні, так і невдалі події, які можуть статися під час взаємодії зі сторінкою.

Переваги використання шаблону об’єкта сторінки:

- легко обслуговувати;
- легка читабельність скриптів;
- зменшити або усунути дублювання;
- можливість повторного використання коду;
- надійність.

PageFactory у класі C# є розширенням шаблону дизайну об’єкта сторінки. Це вбудована концепція POM для Selenium WebDriver, але вона дуже оптимізована. Він використовується для ініціалізації елементів об’єкта сторінки або створення екземпляра самих об’єктів сторінки. Анотації для елементів також можна створювати ( і рекомендувати ), оскільки властивості опису не завжди можуть бути достатньо описовими, щоб відрізнити один об’єкт від іншого.

Він використовується для ініціалізації елементів класу Page без використання « FindElement » або « FindElements ». Анотації можна використовувати для надання описових імен цільових об’єктів для покращення читабельності коду.

Анотація Find.By - Як випливає з назви, це допомагає знаходити елементи на сторінці за допомогою стратегії «За». @FindBy може приймати TagName, PartialLinkText, Name, LinkText, Id, Css, ClassName, XPath як атрибути. Альтернативний механізм пошуку елемента або списку елементів. Це дозволяє користувачам швидко та легко створювати PageObjects.

[FindsBy(How = How.Id, Using = "username")] private IWebElement UserName { get; комплект; }

Наведений вище код створить PageObject і назве його UserName , знайшовши його за допомогою локатора ID.

InitElements - створює екземпляр даного класу. Цей метод намагатиметься створити примірник наданого йому класу, переважно використовуючи конструктор, який приймає екземпляр WebDriver як єдиний аргумент, або повертаючись до конструктора без аргументів. Виняток буде створено, якщо клас не може бути створений.

Клас Base – базовий класий застований як наслідник для автоматизованих тестів з використанням бібліотеки NUnit для тестування та ExtentReports для створення логів про виконання тестів.

ExtentReports і ExtentTest - бібліотеки AventStack.ExtentReports використовуються для створення логів про виконання тестів. (Рис. 3.1)

OneTimeSetUp – цей атрібут позначає що метод буде виконан лише один раз перед запуском усіх тестів. У цьому випадку він використовується для сетапу логів.

```
[OneTimeSetUp]
0 references
public void Setup()
{
    string workingDirectory = Environment.CurrentDirectory;
    string projectDirectory = Directory.GetParent(workingDirectory).Parent.Parent.FullName;
    string reportPath = projectDirectory + "//index.html";
    var htmlReporter = new ExtentHtmlReporter(reportPath);
    extent = new ExtentReports();
    extent.AttachReporter(htmlReporter);
    extent.AddSystemInfo("Host Name", "Localhost");
    extent.AddSystemInfo("Environment", "QA");
    extent.AddSystemInfo("Username", "Danil Baday");
}
```

Рис. 3.1. Setup функция класу Base.cs

ThreadLocal<IWebDriver> - завдяки ньому ми маємо унікальний екземпляр елементу IWebDriver для кожного з потоків. Це дозволяє нам використати паралельне тестування (Рис. 3.2, Рис. 3.3).

SetUp:

- метод [SetUp] позначає код що буде виконан перед кожним тестом;
- об’єкт ‘test’ буде використан для додавання результатів у лог файл;
- змінна ‘browserName’ завантажує ім’я браузера з конфігурацією;
- initBrowser ініціалізує екземпляр браузера залежно від значення змінної.

```
public void StartBrowser()
{
  test = extent.CreateTest(TestContext.CurrentContext.Test.Name);
    //Configuration
    browserName = TestContext.Parameters["browserName"];
    if (browserName == null)
    {
        browserName = ConfigurationManager.AppSettings["browser"];
        //browserName = "Chrome";
    }
    InitBrowser(browserName);

    driver.Value.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);

    driver.Value.Manage().Window.Maximize();
    driver.Value.Url = "file:///D:/%D0%BA%D1%83%D1%80%D1%81%D0%B0%D1%87/finally/Magistr/Automation/Automation/Pages/login.html";
}
```

Рис. 3.2. StartBrowser функция класу Base.cs

```
1 reference
public IWebDriver getDriver()
{
    return driver.Value;
}

1 reference
public void InitBrowser(string browserName)
{

    switch (browserName)
    {
        case "Firefox":
            new WebDriverManager.DriverManager().SetUpDriver(new FirefoxConfig());
            driver.Value = new FirefoxDriver();
            break;

        case "Chrome":
            new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
            driver.Value = new ChromeDriver();
            break;

        case "Edge":
            driver.Value = new EdgeDriver();
            break;
    }
}
```

Рис. 3.3. InitBrowser функция класу Base.cs

TearDown (Рис. 3.4):

- tearDown виконується після кожного тесту;
- визначає статус виконання тесту;
- якщо тест упав робиться знімок екрана та додається у лог;
- збирає та зачиняє лог файл;
- браузер завершує роботу.

```
[TearDown]
0 references
public void AfterTest()
{
    var status = TestContext.CurrentContext.Result.Outcome.Status;
    var stackTrace = TestContext.CurrentContext.Result.StackTrace;



    DateTime time = DateTime.Now;
    String fileName = "Screenshot_" + time.ToString("h_mm_ss") + ".png";

    if (status == TestStatus.Failed)
    {

        test.Fail("Test failed", captureScreenShot(driver.Value, fileName));
        test.Log(Status.Fail,"test failed with logtrace" + stackTrace);

    }
    else if (status == TestStatus.Passed)
    {

    }

    extent.Flush();
    driver.Value.Quit();
}
```

Рис. 3.4. TearDown функция класу Base.cs

CaptureScreenShot – Метод завдяки якому у разі невдачного тесту робиться знімок екрану (Рис. 3.5).

```
1 reference
public MediaEntityModelProvider captureScreenShot(IWebDriver driver,String screenShotName)
{
   ITakesScreenshot ts = (ITakesScreenshot)driver;
   var screenshot= ts.GetScreenshot().AsBase64EncodedString;

   return MediaEntityBuilder.CreateScreenCaptureFromBase64String(screenshot, screenShotName).Build();
}
```

Рис. 3.5. CaptureScreenShot функция класу Base.cs

JsonReader – Цей метод створю об’єкт JsonReader для читання даних з JSON (Рис. 3.6).

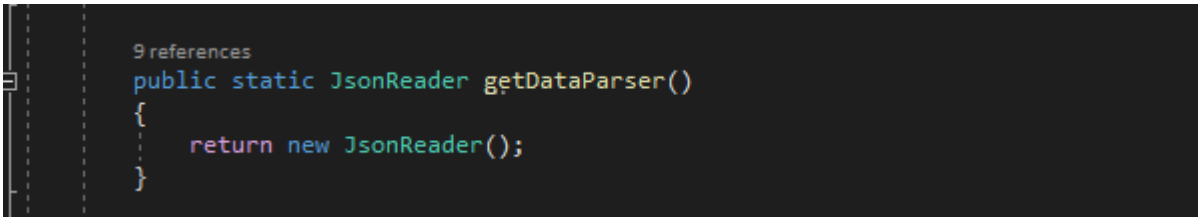
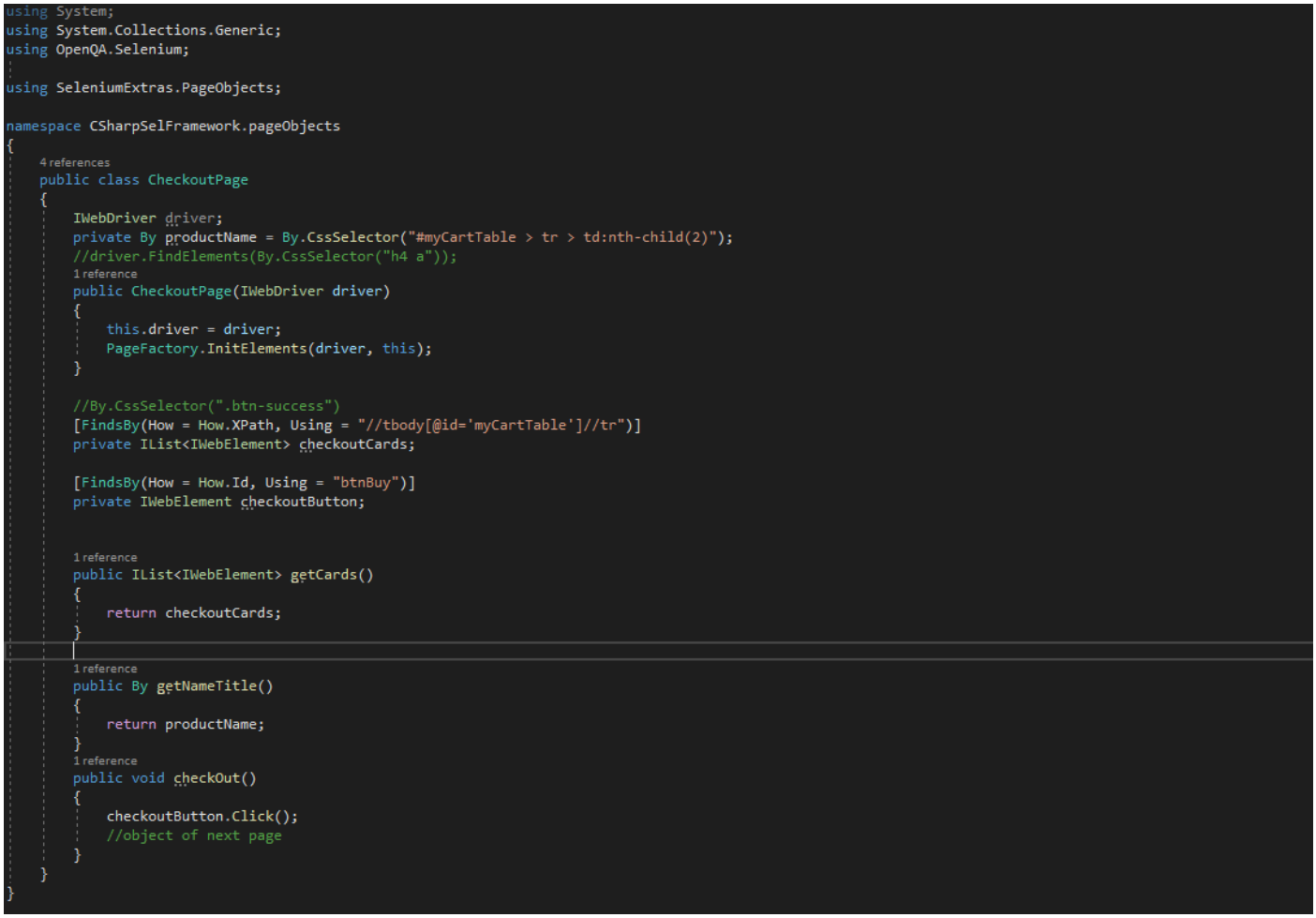



Рис. 3.6. GetDataParser функция класу Base.cs

Клас – CheckoutPage описує сторінку оформлення замовленя у веб-додатку. Основні елементи сторінки (Рис. 3.7):

- productName – Селектор для знаходження назви продуку на сторінці;
- checkoutCards – Список елементів для ініціалізациі у колекцію карточок продуктів за допомогою PageFactory;
- checkoutButton – Кнопка покупки замовлення;
- конструктор – отримує об’єкт IWebDriver та ініціалізує елементи сторінки через метод PageFactory.InitElements;
- getCards() – метод який повертає список елементів на сторінці чекаут;
- getNameTitle() – метод що повертає селектор елементів з назвою продукту;
- checkout() – метод для оформлення замовлення.



```
using System;
using System.Collections.Generic;
using OpenQA.Selenium;

using SeleniumExtras.PageObjects;

namespace CSharpSelFramework.pageObjects
{
    4 references
    public class CheckoutPage
    {
        IWebDriver driver;
        private By productName = By.CssSelector("#myCartTable > tr > td:nth-child(2)");
        //driver.FindElements(By.CssSelector("h4 a"));
        1 reference
        public CheckoutPage(IWebDriver driver)
        {
            this.driver = driver;
            PageFactory.InitElements(driver, this);
        }

        //By.CssSelector(".btn-success")
        [FindsBy(How = How.XPath, Using = "//tbody[@id='myCartTable']//tr")]
        private IList<IWebElement> checkoutCards;

        [FindsBy(How = How.Id, Using = "btnBuy")]
        private IWebElement checkoutButton;


        1 reference
        public IList<IWebElement> getCards()
        {
            return checkoutCards;
        }
        1 reference
        public By getNameTitle()
        {
            return productName;
        }
        1 reference
        public void checkOut()
        {
            checkoutButton.Click();
            //object of next page
        }
    }
}
```

Рис. 3.7. Клас CheckoutPage.cs

Клас – LoginPage описує сторінку входу у веб-додаток.

Основні елементи сторінки (Рис. 3.8, Рис. 3.9, Рис. 3.10, Рис. 3.11):

- loginPage() – конструктор класу який ініціалізує екземпляр драйвера та елементи сторінки;
- validLogin() – метод для введення логіну, паролю, вибору чекбоксу та натискання на кнопку “Sign In”;
- validModalLogin() - метод для виклику модального вікна та введення логіну та паролю у модальному вікні;
- getUserName() – метод отримання веб-елемента логіну.

```
//Pageobject factory

[FindsBy(How = How.Id, Using = "username")]
private IWebElement username;

[FindsBy(How = How.Name, Using = "password")]
private IWebElement password;

[FindsBy(How = How.Id, Using = "password")]
private IWebElement modalPassword;

[FindsBy(How = How.Id, Using = "email")]
private IWebElement modalEmail;

[FindsBy(How = How.XPath, Using = "//*[@id='exampleModal']/div/div/div[3]/button[2]")]
private IWebElement enterButton;

[FindsBy(How = How.PartialLinkText, Using = "Hello User")]
private IWebElement loginModal;

[FindsBy(How = How.XPath, Using = "//div[@class='form-group'][5]/label/span/input")]
private IWebElement checkBox;

[FindsBy(How = How.CssSelector, Using = "input[value='Sign In']")]
private IWebElement signInButton;
```

Рис. 3.8. Оголошення Web-елементів сторінки за допомогою аннотацій PageFactory

```
1 reference
public LoginPage(IWebDriver driver)
{
    this.driver = driver;
    PageFactory.InitElements(driver, this);
}
```

Рис. 3.9. Конструктор класу LoginPage.cs

```
1 reference
public ProductsPage validLogin(string user, string pass)
{
    username.SendKeys(user);
    password.SendKeys(pass);
    checkBox.Click();
    signInButton.Click();
    return new ProductsPage(driver);
}
```

Рис. 3.10. Функція вода логіна і паролю та натиска кнопки увійти класу LoginPage.cs

```
public ProductsPage validModalLogin(string email, string pass)
{
    loginModal.Click();
    modalEmail.SendKeys(email);
    modalPassword.SendKeys(pass);
    enterButton.Click();
    Thread.Sleep(2500);
    return new ProductsPage(driver);
}
```

Рис. 3.11. Функція ввода логіна і паролю у модальному вікні класу LoginPage.cs

Клас – ProductsPage описує сторінку для взаємодії з елементами сторінки продуктів у веб-додатку.

Основні елементи сторінки (Рис. 3.12, Рис. 3.13, Рис. 3.14, Рис. 3.15, Рис. 3.16, Рис. 3.17, Рис. 3.18, Рис. 3.19):

- waitForPageDisplay() – метод вижидач який очікує, поки сторінка буде видимою;
- getCards() – метод який повертає список веб-елементів, які представляють картки продуктів;
- getCardTitle() – метод який повертає локатор заголовка продукту;
- addToCartButton() – метод який повертає локатор кнопки “Add to Cart”.

```
IWebDriver driver;
//By cardTitle = By.CssSelector(".card-title a");
By cardTitle = By.CssSelector("#product > div > a > p:nth-child(2)");
By addToCart = By.CssSelector("#product > div > button");
```

Рис. 3.12. Оголошення локаторів (елементів сторінки) за допомогою засобів PageFactory класу ProductsPage.cs

```
2 references
public ProductsPage(IWebDriver driver)
{
    this.driver = driver;
    PageFactory.InitElements(driver, this);

}
```

Рис. 3.13. Конструктор класу, який ініціалізує екземпляр драйвера та елементи сторінки класу ProductsPage.cs

```
[FindsBy(How = How.Id, Using = "product")]
private IList<IWebElement> cards;

[FindsBy(How = How.Id, Using = "Checkout")]
private IWebElement checkoutButton;
```

Рис. 3.14. Оголошення веб-елементів сторінки за допомогою PageFactory класу ProductsPage.cs

```
1 reference
public void waitForPageDisplay()
{
    WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(8));
    wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions.ElementIsVisible(By.XPath("//h1[text()='Welcome to our shop']")));
}
```

Рис. 3.15. Метод для очікування відображення сторінки класу ProductsPage.cs

```
1 reference
public IList<IWebElement> getCards()
{
    return cards;
}
```

Рис. 3.16. Метод для отримання списку веб-елементів карток продуктів класу ProductsPage.cs

```
1 reference
public By getCardTitle()
{

    return cardTitle;
}
```

Рис. 3.17. Метод для отримання локатора заголовка картки продукту класу ProductsPage.cs

```
1 reference
public CheckoutPage checkout()
{

    checkoutButton.Click();
    return new CheckoutPage(driver);
}
```

Рис. 3.18. Метод для виклику сторінки оформлення замовлення класу ProductsPage.cs

```
1 reference
public By addToCartButton()
{

    return addToCart;
}
```

Рис. 3.19. Метод для отримання локатора кнопки "Add to Cart" класу ProductsPage.cs

Клас SortWebTables – використовується для сортування та порівняння даних у таблиці на веб-сторінці додатку (Рис. 3.20). Нижче наведен опис роботи

Старт:

- запускає хром браузер;
- встановлюється таймаут для очікування на 5 секунд;
- максимізує вікно веб-браузера;
- відкривається вказана адреса яка вказує на table.html.

```
[SetUp]

0 references
public void StartBrowser()
{
    new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
    driver = new ChromeDriver();

    driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);

    driver.Manage().Window.Maximize();
    driver.Url = "file:///D:/%D0%BA%D1%83%D1%80%D1%81%D0%B0%D1%87/finally/Magistr/Automation/Automation/Pages/table.html#/offers";
}
```

Рис. 3.20. Прекондішн кейсу у класі SortWebTables.cs

Сортування таблиці (Рис. 3.21):

- вибирає зі списку вибору кількість записів на сторінці;
- збирає дані з першого стовпчика таблиці у ArrayList з іменем a;
- виводить елементи в консоль перед сортуванням;
- сортує ArrayList;
- виводить елементи в консоль після сортування;
- клікає на заголовок стовпчика "fruit name" для сортування таблиці;
- збирає дані (назви овочів) після сортування у ArrayList з іменем b;
- перевіряє, що ArrayList a і ArrayList b рівні за допомогою NUnit-асерції.

```
[Test]

0 references
public void SortTable()
{
    ArrayList a = new ArrayList();
    SelectElement dropdown = new SelectElement(driver.FindElement(By.Id("page-menu")));
    dropdown.SelectByValue("20");

    // step 1 - Get all veggie names into arraylist A
    IList <IWebElement> veggies = driver.FindElements(By.XPath("//tr/td[1]"));

    foreach(IWebElement veggie in veggies)
    {
        a.Add(veggie.Text);
    }

    //step 2- Sort this arraylist  -A

    foreach (String element in a)
    {
        TestContext.Progress.WriteLine(element);
    }

    TestContext.Progress.WriteLine("After sorting");
    a.Sort();
    foreach( String element in a)
    {
        TestContext.Progress.WriteLine(element);
    }

    //step 3 - go and click column
    driver.FindElement(By.CssSelector("th[aria-label *= 'fruit name']")).Click();

    //step 4- Get all veggie names into arraylist B

    ArrayList b = new ArrayList();

    IList<IWebElement> sortedVeggies = driver.FindElements(By.XPath("//tr/td[1]"));

    foreach (IWebElement veggie in sortedVeggies)
    {
        b.Add(veggie.Text);
    }

    // arraylist A to B = equal
    Assert.AreEqual(a, b);
    driver.Quit();


}
```

Рис. 3.21. Автоматизований тест на сортування та порівняння таблиць

Клас E2ETest містить два методи для тестування різних аспектів веб-додатка (Рис. 3.22):

Метод EndToEndFlow:

- виконує весь енд ту енд потік тестування;
- використовує методи сторінок (validLogin, validModalLogin, getCards, addToCartButton, тощо) та очікує, що сторінка відобразиться після входу (waitForPageDisplay);
- проходить через всі продукти на сторінці та додає їх у кошик, якщо вони збігаються з переданим масивом очікуваних продуктів;
- перевіряє, чи відобразилися обрані продукти на сторінці оформлення замовлення;
- перевіряє, чи очікувані та фактичні продукти співпадають, використовуючи NUnit-асерцію;
- завершує оформлення замовлення.

```
[Parallelizable(ParallelScope.Self)]
0 references
public class E2ETest : Base
{

    [Test, TestCaseSource("AddTestDataConfig"), Category("Regression")]

    [Parallelizable(ParallelScope.All)]
    0 references
    public void EndToEndFlow(string username, string password, string[] expectedProducts)
    {
        string[] actualProducts = new string[2];
        LoginPage loginPage = new LoginPage(getDriver());
        ProductsPage productPage = loginPage.validLogin(username, password);
        productPage.waitForPageDisplay();
        productPage = loginPage.validModalLogin("sa@gmail.com", "Pa$$w0rd5");
        IList<IWebElement> products = productPage.getCards();

        foreach (IWebElement product in products)
        {
            if (expectedProducts.Contains(product.FindElement(productPage.getCardTitle()).Text))
            {
                product.FindElement(productPage.addToCartButton()).Click();
            }
        }
        CheckoutPage checkoutPage = productPage.checkout();

        IList<IWebElement> checkoutCards = checkoutPage.getCards();

        for (int i = 0; i < checkoutCards.Count; i++)
        {
            //actualProducts[i] = checkoutCards[i].Text;
            actualProducts[i] = checkoutCards[i].FindElement(checkoutPage.getNameTitle()).Text;
        }
        Assert.AreEqual(expectedProducts, actualProducts);
        checkoutPage.checkOut();
    }
```

Рис. 3.22. Автоматизований тест на перевірку очікуваних та фактичних продуктів

Метод LocatorsIdentification (Рис. 3.23):

- демонструє використання різних методів локації елементів на веб-сторінці;
- використовує методи By.Id, By.Name, By.XPath, тощо для знаходження елементів на сторінці та виконання з ними різних дій;
- використовує очікування (WebDriverWait) для чекання певного стану елемента перед виконанням дій;
- виводить текст помилки в консоль за допомогою TestContext.Progress.

```
[Test, Category("Smoke")]
0references
public void LocatorsIdentification()
{
    driver.Value.FindElement(By.Id("username")).SendKeys("Baday");
    driver.Value.FindElement(By.Name("password")).SendKeys("wrong");
    //css selector & xpath
    //  tagname[attribute ='value']
    //    #id  #terms  - class name -> css .classname
    //    driver.FindElement(By.CssSelector("input[value='Sign In']")).Click();

    //    //tagName[@attribute = 'value']

    // CSS - .text-info span:nth-child(1) input
    //xpath - //label[@class='text-info']/span/input

    driver.Value.FindElement(By.XPath("//div[@class='form-group'][5]/label/span/input")).Click();

    driver.Value.FindElement(By.XPath("//input[@value='Sign In']")).Click();

    WebDriverWait wait = new WebDriverWait(driver.Value, TimeSpan.FromSeconds(8));
    wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions
   .TextToBePresentInElementValue(driver.Value.FindElement(By.Id("signInBtn")), "Sign In"));

    String errorMessage = driver.Value.FindElement(By.ClassName("alert-danger")).Text;
    TestContext.Progress.WriteLine(errorMessage);
}
0references
```

Рис. 3.23. Автоматизований тест на перевірку правильного відображення елементів

Метод AddTestDataConfig:

- постачає тести з даними за допомогою методу extractData для username та password та extractDataArray для products;
- метод використовується як TestCaseSource для передачі даних в метод EndToEndFlow.

Клас WindowHandlers – містить тест який взаємодіє з веб-браузером та перевіряє елементи на разних вікнах. Кроки виконання тесту:

Метод StartBrowser (Рис. 3.24):

- ініціалізація хромдрайвера та його налаштування;
- макксимізація вікна браузера;
- перехід до сторінки тестування через URL.

```
[SetUp]

0 references
public void StartBrowser()
{
    new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
    driver = new ChromeDriver();

    driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);

    driver.Manage().Window.Maximize();
    driver.Url = "file:///D:/%D0%BA%D1%83%D1%80%D1%81%D0%B0%D1%87/finally/Magistr/Automation/Automation/Pages/login.html";
}
```

Рис. 3.24. Реалізація методу StartBrowser() у класі WindowHandlers.cs

Метод WindowHandle (Рис. 3.25):

- клікає на елемент з класом "blinkingText" на вихідній сторінці;
- перевіряє, чи кількість вікон дорівнює 2 (перше вікно та нове вікно, яке з'являється після кліку);
- отримує ідентифікатор вікна, яке з'явилося (друге вікно);
- перемикається на нове вікно;
- отримує текст з елементу на новому вікні та обробляє його, розділяючи та вирізаючи необхідну інформацію;
- перемикається назад на вихідне вікно та вводить знайдену інформацію у поле з ідентифікатором "username";
- завершує роботу веб-браузера.

```
[Test]

0 references
public void WindowHandle()
{
    string email = "rector@nmu.org.ua";
    string parentWindowId = driver.CurrentWindowHandle;
    driver.FindElement(By.ClassName("blinkingText")).Click();

    Assert.AreEqual(2, driver.WindowHandles.Count);//1

    String childWindowName = driver.WindowHandles[1];

    driver.SwitchTo().Window(childWindowName);

    TestContext.Progress.WriteLine(driver.FindElement(By.CssSelector("#contacts > p")).Text);
    string text = driver.FindElement(By.CssSelector("#contacts > p")).Text;

    string[] splittedText = text.Split("e-mail: ");

    string[] trimmedString = splittedText[1].Trim().Split(" ");

    Assert.AreEqual(email, trimmedString[0]);

    driver.SwitchTo().Window(parentWindowId);

    driver.FindElement(By.Id("username")).SendKeys(trimmedString[0]);
    driver.Quit();
}
```

Рис. 3.25. Перевірка знайденого тексту з очікуваємим

### 3.2. Використані програмні засоби та бібліотеки

Під час розробки даного застосунку були використані такі програмні засоби:

- Visual Studio 2022 - це найкраще інтегроване середовище розробки для створення багатофункціональних, привабливих кроссплатформенних додатків для Windows, Mac, Linux, iOS та Android;
- DotNetSeleniumExtras.PageObjects - Цей пакет забезпечує реалізацію PageFactory для .NET, замінюючи реалізацію, спочатку надану проектом Selenium;
- DotNetSeleniumExtras.PageObjects.Core - Форк з увімкненою підтримкою ядра dotnet та іншими функціями;
- DotNetSeleniumExtras.WaitHeleprs - Цей пакет забезпечує реалізацію класу ExpectedConditions для використання з WebDriverWait у .NET, замінюючи реалізацію, спочатку надану проектом Selenium;
- ExtentReports - Extent Reporting Framework (.Net Standard), версія спільноти;
- Microsoft.NET.Test.Sdk - Цілі та властивості Msbuild для побудови. НЕ тестові проекти;
- NUnit - має плавний синтаксис assert, параметризовані, загальні та теоретичні тести та розширюється користувачем;
- NUnit3TestAdapter - Пакет, що включає NUnit 3 TestAdapter для Visual Studio 2012 і новіших версій. З цим пакетом вам не потрібно встановлювати пакет адаптера VSIX і вам не потрібно завантажувати адаптер на ваш сервер TFS;
- Selenium.WebDriver - це набір різних програмних інструментів, кожен з яких має свій підхід для підтримки автоматизації браузера. Ці інструменти відрізняються високою гнучкістю, що дозволяє багато варіантів пошуку та

керування елементами в браузері та один однією з його ключових особливостей є підтримка автоматизації кількох платформ браузерів;

System.Configuration.ConfigurationManager;

− WebDriverManager - Автоматичне керування бінарними файлами Selenium WebDriver для .Net.

Принцип роботи «Фабрики об’єктів сторінки» (POF) базується на ідеї використання шаблону дизайну «Об’єкт сторінки» для організації та керування веб-елементами на сторіках веб-додатків. Цей підхід став популярним у світі автоматизованого тестування,

особливо для тестування веб-додатків.

Основні компоненти фабрики об’єктів сторінки:

Об’єкт сторінки – це клас, який представляє сторінку веб-програми та містить взаємодії з елементами цієї сторінки. Він має методи для виконання операцій на сторінці та отримання інформації з елементів.

Фабрика сторінок — це інструмент або бібліотека, яка автоматизує процес ініціалізації елементів об’єкта сторінки. Як правило, він надає анотації для

позначення елементів на сторінці та вбудовані методи для їх автоматичної ініціалізації [13].

Основні принципи роботи:

Створити об'єкт сторінки: для кожної сторінки веб-додатку створюється окремий клас об’єктів сторінки, який відповідає за взаємодію з цією сторінкою.

Назва елемента: за допомогою коментарів або іншої розмітки визначаються елементи на сторінці, з якими буде взаємодіяти тест.

Ініціалізація елементів: фабрика сторінок автоматично ініціалізує елементи сторінки, визначені в об’єкті сторінки, роблячи їх доступними для тестування.

Спосіб взаємодії: у класі об’єктів сторінки створюються методи для реалізації різноманітних операцій на сторінці, таких як введення тексту, натискання кнопок, отримання значень тощо.

Повторне використання коду: фабрики об’єктів сторінки сприяють повторному використанню коду, оскільки той самий об’єкт сторінки можна використовувати в різних тестах [16].

Забезпечити стабільність тесту: виокремлюючи логіку взаємодії з елементами від тестового сценарію, Page Object Factory забезпечує стабільність тесту під час зміни структури сторінки.

Користувачеві  потрібно мати систему, яка може запускатися та працювати з сучасними браузерами. Відповідні характеристики:

- цп – 64-розрядний процесор із тактовою частотою не нижче 1,8 ГГц. рекомендується двоядерний або з великою кількістю ядер. Підтримуються процесори Intel (x64) та Apple Silicon (arm64);
- відеопам’ять – 4 ГБ ОЗП; рекомендується 8 ГБ ОЗП (мінімум 4 ГБ при виконанні на віртуальній машині);
- місце на жорсткому диску: від 2.2 ГБ до 13 ГБ вільного місця, залежно від встановлених компонентів;
- ос Windows 7/8/10/11;
- оперативна пам’ять – 8Гб.

### 3.3. Виклик та завантаження програми

Розроблений тести завантажується на будь-який хостинг у нашому випадку це локальна машина. Далі користувачу потрібно открити солюшн фреймворку за допомогою Visual Studio 2022.

### 3.4. Опис інтерфейсу користувача

Після запуску Visual Studio 2022 користувач має обрати проект з автотестами (Рис.3.26).

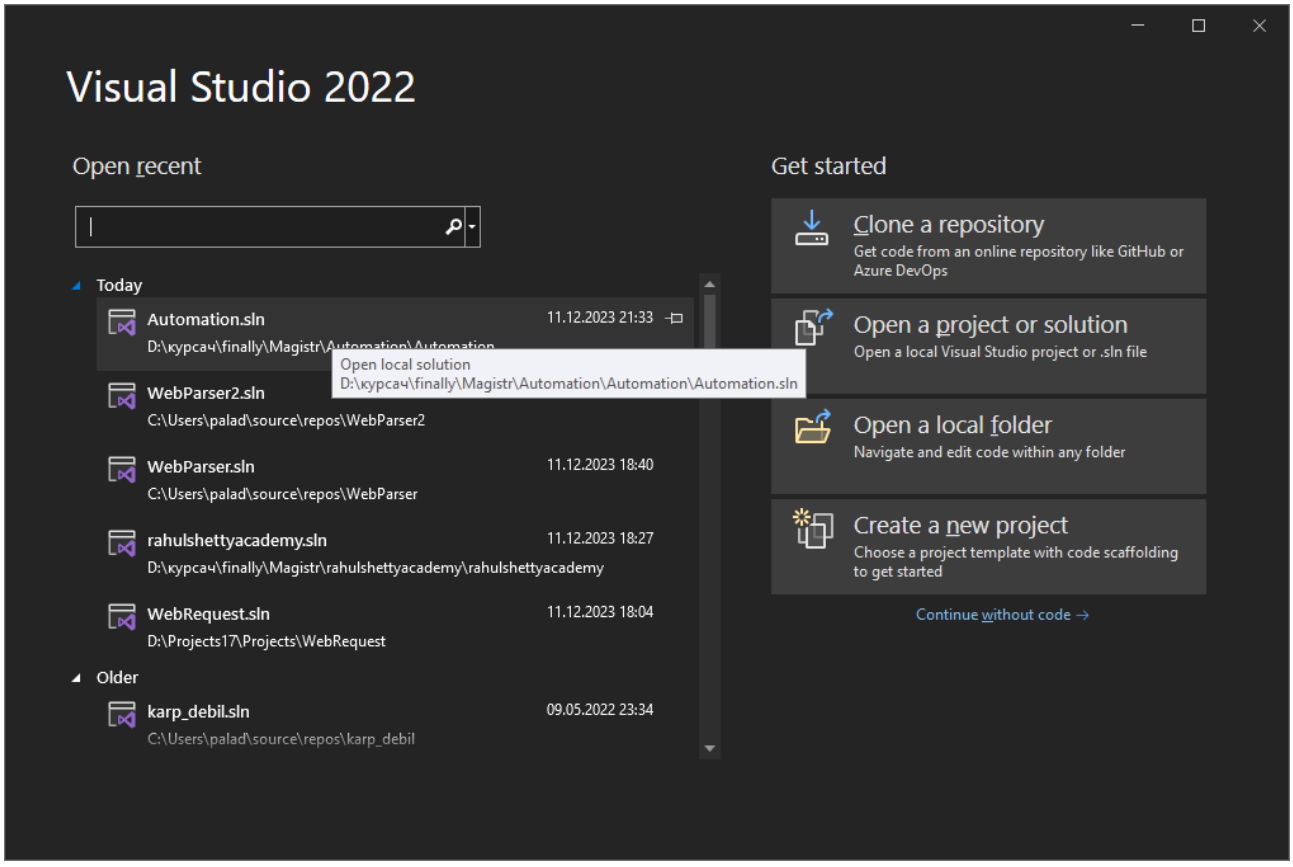



Рис. 3.26. Головне вікно Visual Studio 2022

Після цього відкривається головне вікно проекту (рис. 3.27).

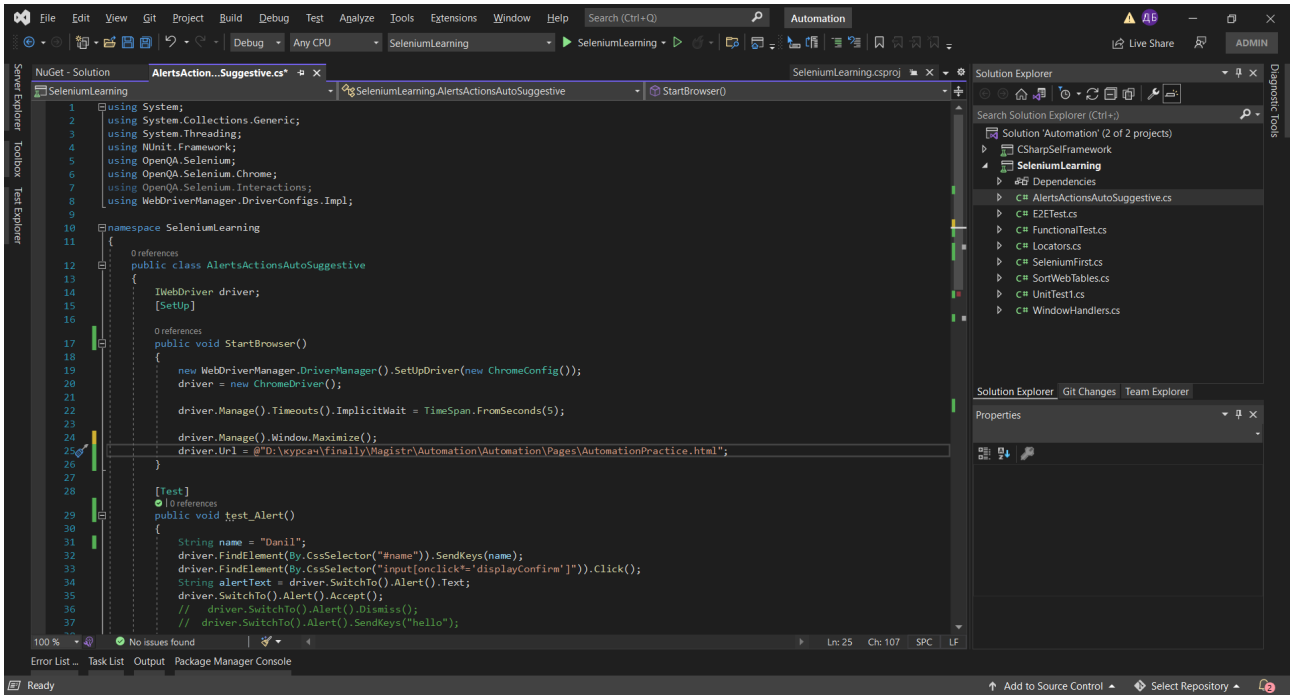



Рис. 3.27. Вікно проекту с кодом

Далі потрібно увімкнути вікно Test Explorer (Рис. 3.28)

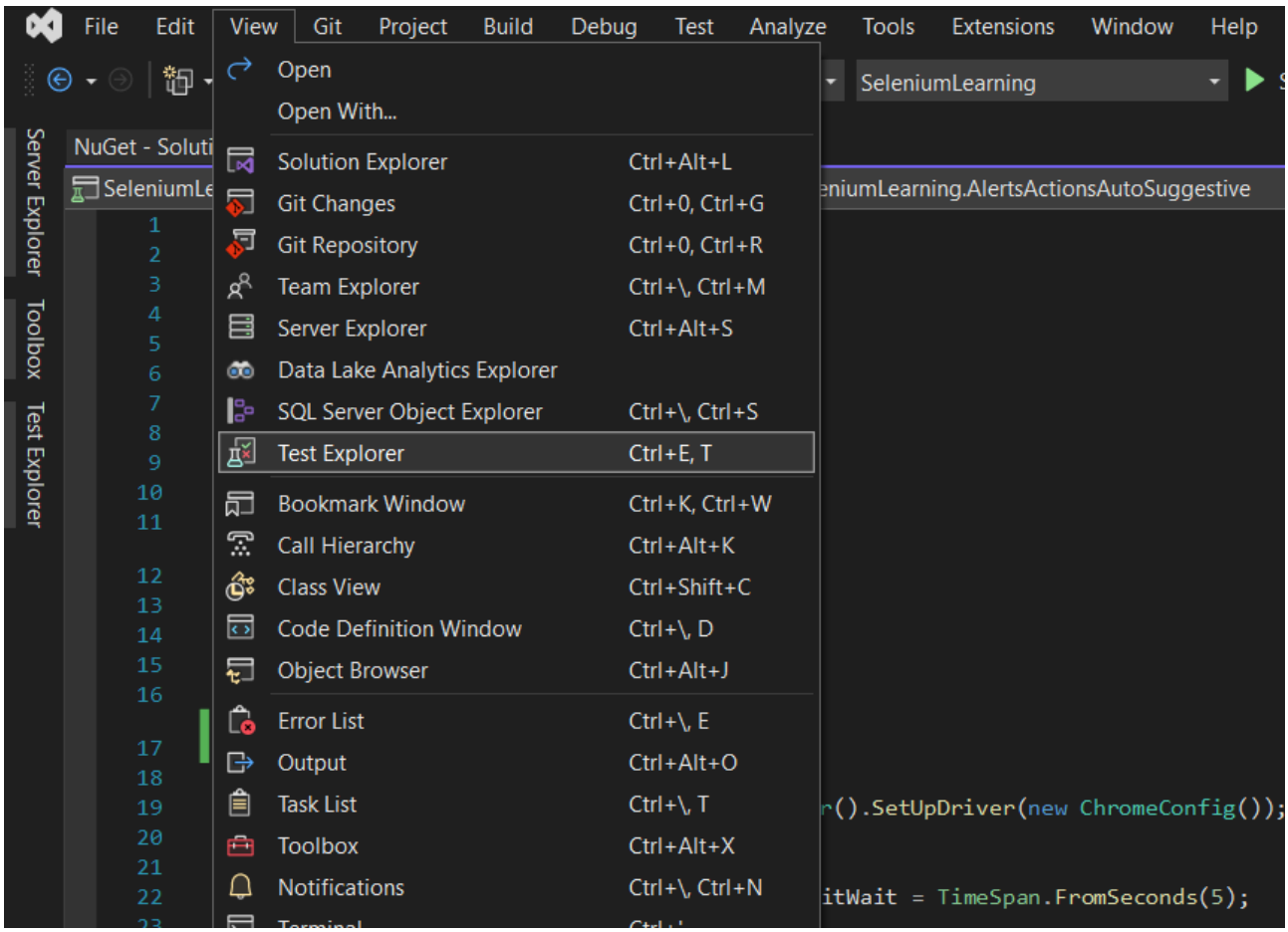


Рис. 3.28. Вкладка з Test Explorer

У цьому меню ми можемо побачити усі наші кейсі та їх статус (Рис. 3.29, Рис. 3.30, Рис. 3.31, Рис. 3.32, Рис. 3.33).

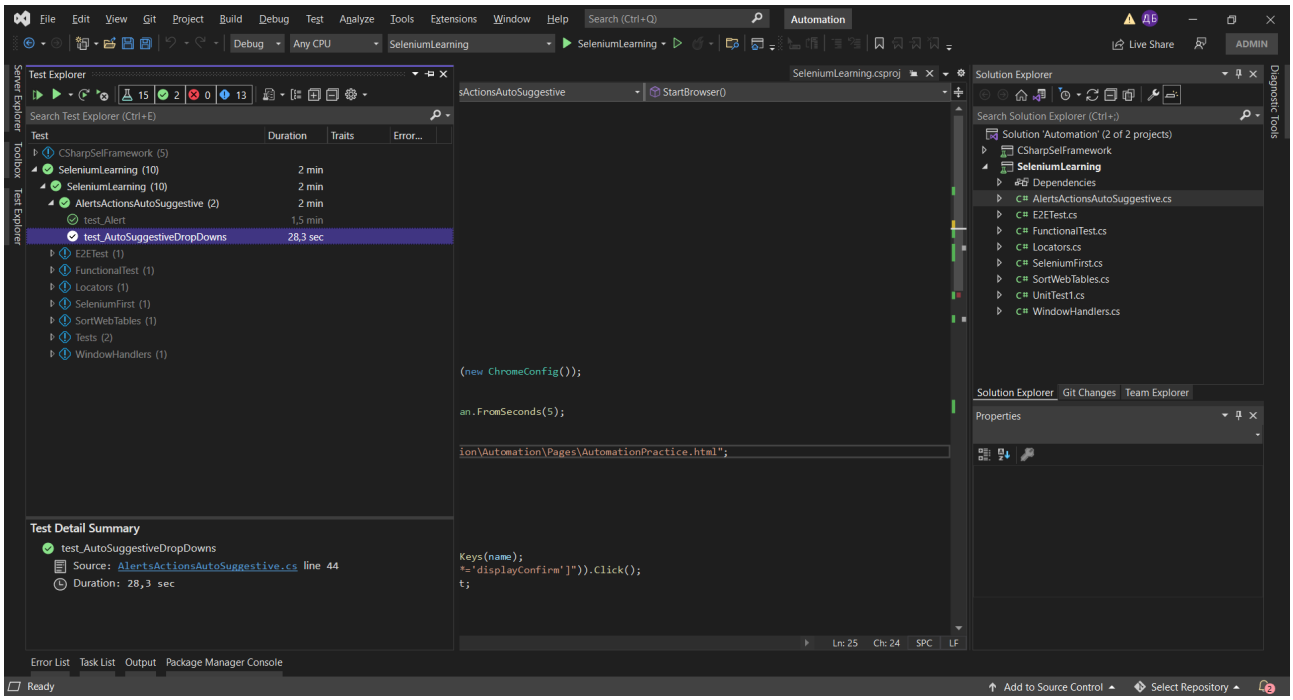



Рис. 3.29. Вікно статусу та запуску кейсів

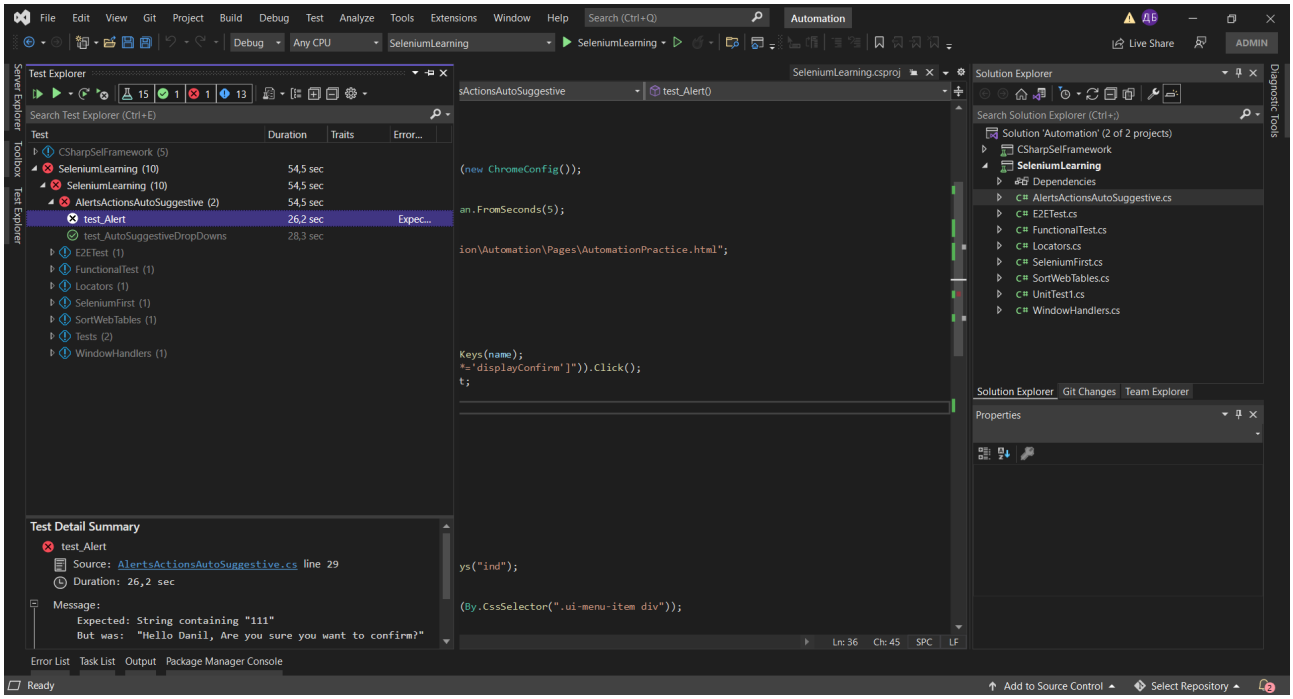



Рис. 3.30. Вікно статусу та запуску кейсів

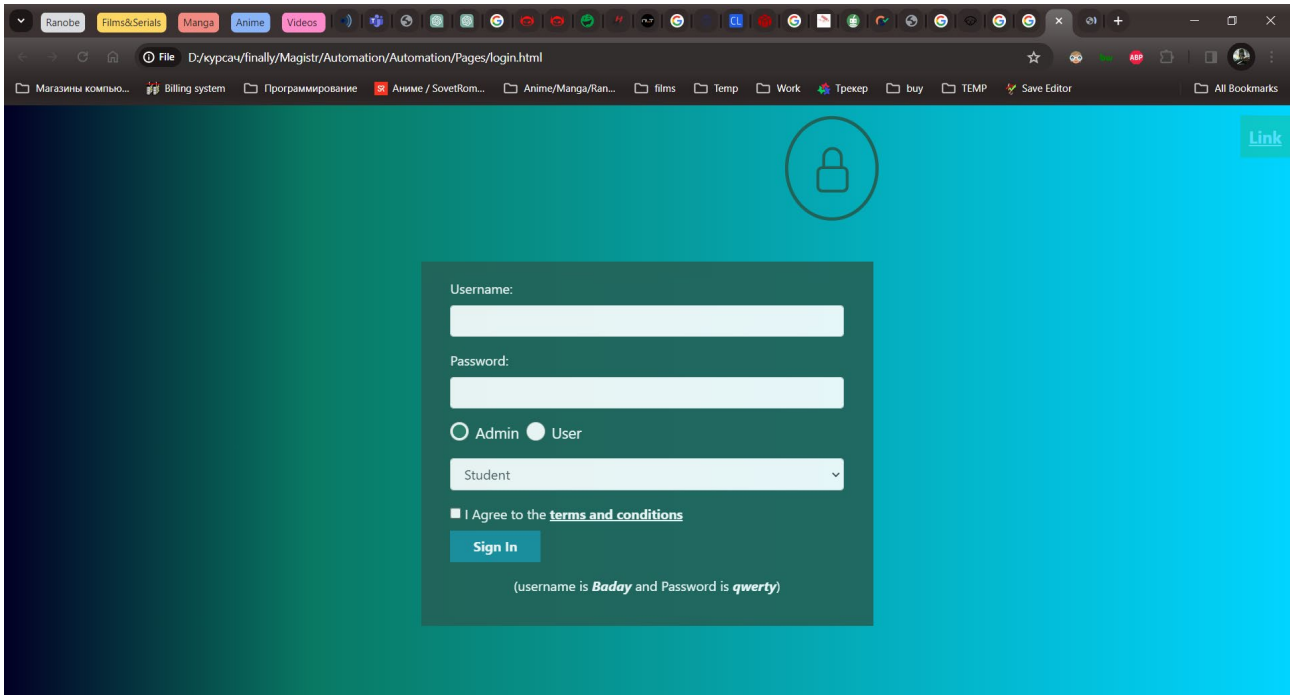



Рис. 3.31. Приклад тестуємого веб-додатку

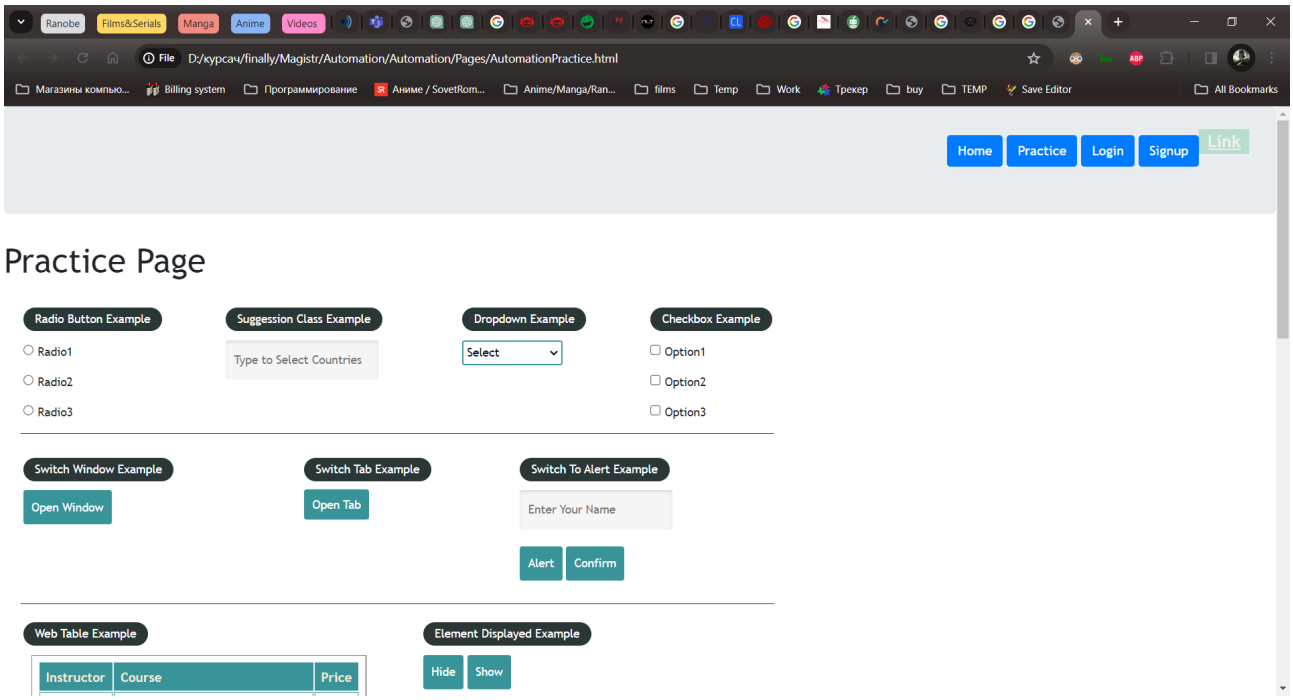



Рис. 3.32. Приклад тестуємого веб-додатку

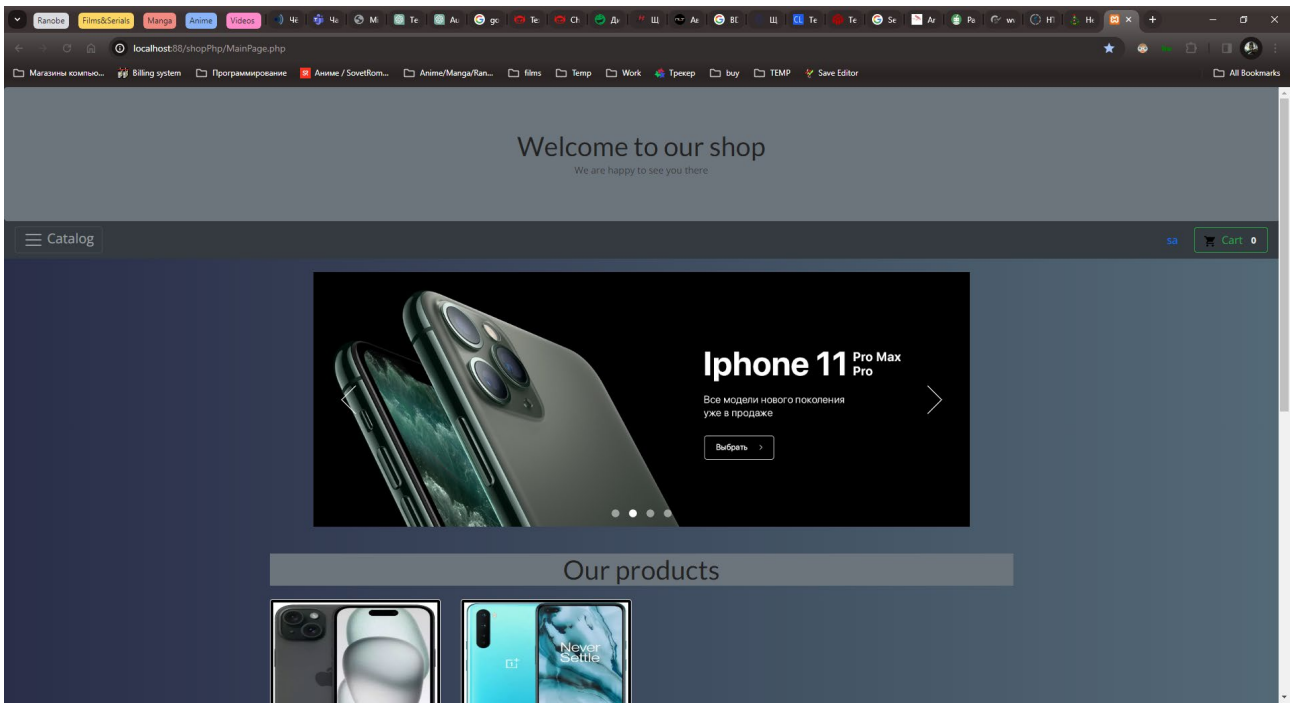



Рис. 3.33. Приклад тестуємого веб-додатку

### 3.5. Збір та аналіз даних під час тестування

Під час виконання автоматизованих тестів були зібрані такі дані як фіксація їх стану, логи, час виконання (Рис 3.34, Рис. 3.35).

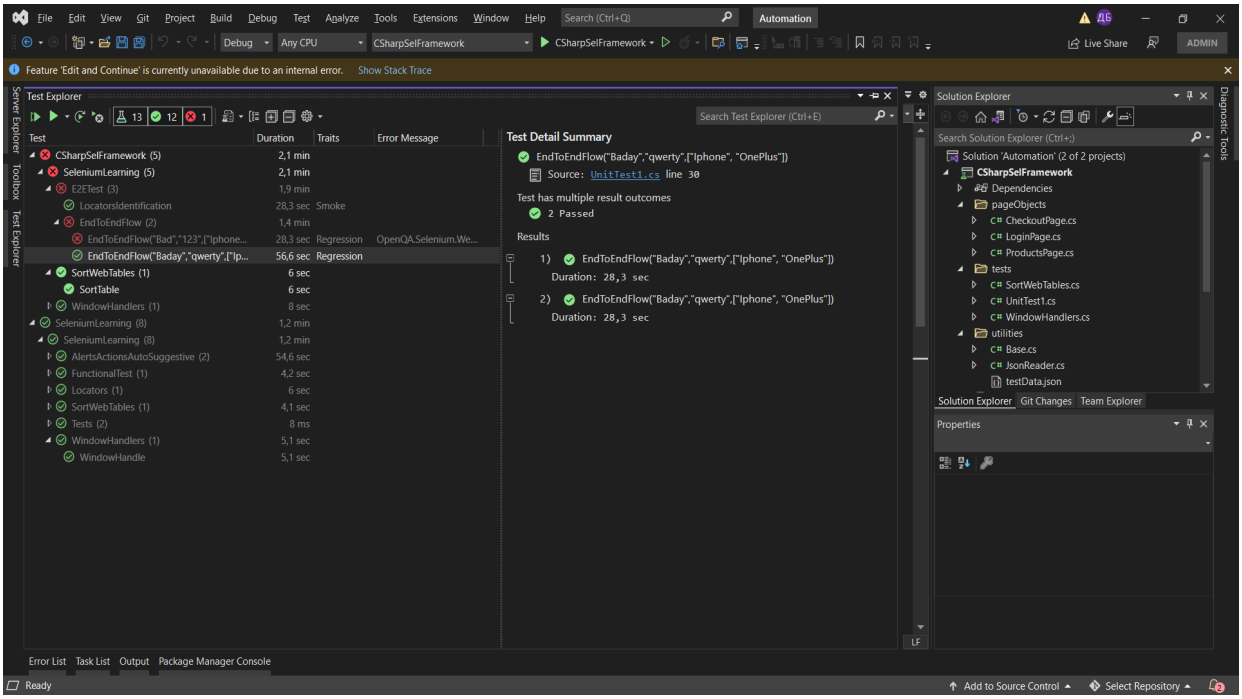



Рис. 3.34. Стан виконаних тестів та час виконання

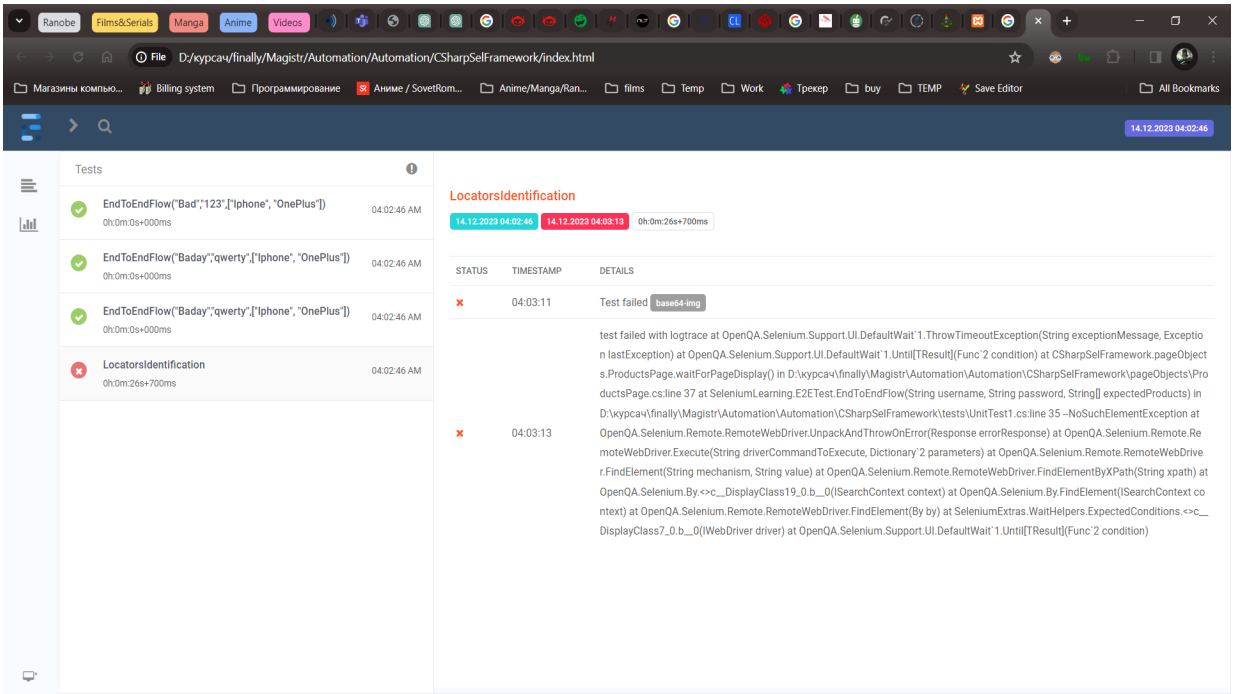



Рис. 3.35. Логування помилок

### 3.6. Реалізація веб-додатку

Для написання тестів був створен та використан веб-додаток інтернет магазин.

Стек технологій веб-додатку:

- веб-сервер APACHE;
- mySql;
- php;
- html;
- css;
- javaScript;
- redBeanPHP.

Проект HTTP-сервера Apache — це проект спільної розробки програмного забезпечення, спрямований на створення надійних, функціональних і загальнодоступних реалізацій вихідного коду HTTP комерційного рівня (веб-сервер). Проектом керує команда волонтерів з усього світу, які використовують Інтернет, веб-сервери та відповідну документацію для спілкування, планування та розробки. Цей проект є частиною Apache Software Foundation. Крім того, сотні користувачів додали ідеї, код і документацію до проекту. Метою цього документу є короткий опис історії HTTP-сервера Apache і відзначення багатьох його учасників.

Веб-сервер Apache має кілька ключових відмінностей:

Модульні та розширювані. Немає необхідності завантажувати багатомегабайтні дистрибутиви окремих компонентів. Основна версія запропонованого пакета Apache+PHP+Perl+MySQL має розмір лише близько 3,1 МБ і повністю функціональна.

MySQL (International Phonetic Alphabet) — це безкоштовна система керування реляційною базою даних. MySQL розроблено та підтримується корпорацією Oracle, яка придбала права на торговельну марку шляхом придбання Sun Microsystems, яка раніше придбала шведську компанію MySQL AB. Продукт ліцензовано згідно з Ліцензія GNU General Public License, має свою

комерційну ліцензію, крім того, розробники створюють функції для авторизованих користувачів, і саме завдяки такій послідовності механізм реплікації з'явився практично в самих ранніх версіях.

Широкий діапазон типів таблиць підтримує гнучкість СУБД MySQL: користувачі можуть вибирати між таблицями MyISAM, які підтримують повнотекстовий пошук, і таблицями InnoDB, які підтримують транзакції на рівні одного запису. Також, MySQL постачається зі спеціальним зразком типу таблиці, який демонструє, як створювати нові типи таблиць. Завдяки відкритій архітектурі та ліцензії GPL до бази даних MySQL постійно додаються нові типи таблиць.

PHP ("PHP: Hypertext Preprocessor") — є широко використовуваною мовою скриптів загального призначення з відкритим кодом, створеною для Інтернету та вбудованою в HTML.

Це дуже відрізняється від сценарію, написаного на Perl або C - замість того, щоб писати програму з великою кількістю команд для виведення HTML, ви пишете HTML-скрипт із деяким вбудованим кодом, який щось робить. Код PHP загорнутий у спеціальні початкові та кінцеві теги (<?php...?>), які дозволяють увійти та вийти з режиму PHP.

Що відрізняє PHP від інших подібних мов, таких як JavaScript на стороні клієнта, так це те, що код виконується на сервері. Ви можете налаштувати свій веб-сервер для обслуговування всіх файлів HTML за допомогою PHP, і користувач дійсно не зможе знати, що відбувається.

PHP доступний для всіх основних операційних систем (ОС), включаючи Linux, Microsoft Windows і, можливо, інші. PHP підтримує більшість існуючих веб-серверів: Apache, Microsoft Internet Information Server, Personal Web Server тощо. Для більшості таких серверів PHP має модулі. В інших випадках, коли підтримується стандарт CGI, PHP може діяти як процесор CGI.

У PHP у вас немає обмежень на виведення HTML. PHP може виводити динамічно згенеровані зображення, PDF-файли і навіть Flash-кліпи. Ви також можете легко виводити будь-який текст, включаючи XHTML та будь-який інший

файл XML. PHP може автоматично генерувати ці файли та зберігати їх у файловій системі замість того, щоб їх друкувати, створюючи кеш на стороні сервера для вашого динамічного вмісту. Однією з найбільш потужних і привабливих функцій PHP є підтримка великої кількості баз даних (БД).

HTML (від англ. HyperText Markup Language — "мова гіпертекстової розмітки") - стандартизована мова розмітки документів для перегляду веб-сторінок у браузері. Веб-браузери отримують HTML документ від сервера за протоколами HTTP/HTTPS або відкривають з локального диска, далі інтерпретують код в інтерфейс, який відображатиметься на екрані монітора.

Елементи HTML є будівельними блоками сторінок HTML. За допомогою HTML різні конструкції, зображення та інші об'єкти, такі як інтерактивна веб-форма, можуть бути вбудовані в сторінку, що відображається. HTML надає засоби для створення заголовків, абзаців, списків, посилань, цитат та інших елементів. Елементи HTML виділяються тегами, записаними з використанням кутових дужок. Такі теги, як <img/> та <input/>, безпосередньо вводять контент на сторінку. Інші теги, такі як <p>, оточують і оформляють текст у собі і можуть включати інші теги як поделементів. Браузери не відображають HTML-теги, але використовують їх для інтерпретації вмісту сторінки.

У HTML можна вбудувати програмний код мовою програмування JavaScript, керувати поведінкою та змістом веб-сторінок. Також включення CSS в HTML описує зовнішній вигляд та макет сторінки.

CSS — використовується творцями веб-сторінок для завдання кольорів, шрифтів, стилів, розташування окремих блоків та інших аспектів представлення цих веб-сторінок. Основною метою розробки CSS є огородження та відокремлення опису логічної структури веб-сторінки (яке виробляється за допомогою HTML або інших мов розмітки) від опису зовнішнього вигляду цієї веб-сторінки (яке тепер виробляється за допомогою формальної мови CSS). Такий поділ може збільшити доступність документа, надати більшу гнучкість та можливість управління його поданням, а також зменшити складність та повторюваність у структурному вмісті.

RedBeanPHP — ще одна ORM-бібліотека. Основна її відмінність від колег, типу Propel або Doctrine, без необхідності ручного конфігурування об'єктів. Тобто. ніяких xml, yml чи ini-файлів. RedBenPHP на льоту створює таблиці, поля та індекси. Будь-який об'єкт можна пов'язати з іншим. З БД підтримуються MySQL, SQLite та Postgres.

### 3.7. Визначення покращень в розробці веб-додатку завдяки автоматизації

Мануальне тестування у більшості випадків займає більше часу особливо при тестах з пербіркою змінних або після внесення змін. Автоматизоване тестування вирішує цю проблему та дозволяє нам збільшити швидкість розробки веб-додатку.

Людский фактор може негативно впливати на роботу тестів. Автоматизація дозволяє нам виникнути цієї проблеми так як автоматизовані тести виконуються однаково декілька раз, що дозволяє нам зменшити ризик помилок.

Ручне тестування може інколи буває обмеженим у покритті. Наприклад дуже великий веб-додаток. Наперевага цьому автоматизовані кейси можуть легко покрити велики частини коду. Під час тривалої розробки веб-додатку

збільшується обсяг коду через це мануальне тестування вимагає більших зусиль та людино ресурсів. Автоматизація дуже легко вирішує це питання.

Мануальне регресійне тестування витрачує дуже багато часу. Автотести можна легко імплементувати у CI/CD пайплайни.

Мануальне виявлення помилок займає більше часу ніж автоматизований лог.

### 3.8. Висновки

Автоматизоване як і ручне тестування має як переваги так і недоліки

Переваги автоматизованого тестування:

- швидкість;
- високий рівень покриття;
- всиока ефективність у великих проектах.

Недоліки автоматизованого тестування:

- початкови витрати на написання тестів;
- постійна підтримка існуючих тестів;
- зміна функціонального коду програми та випливаючий з цього рефакторінг.

Переваги мануального тестування:

- високий реалізації на початкових етапах;
- легка адаптація до змін у проекті.

Недоліки маунального тестування:

- багато часу витручається під час виконання повторюваних тестів;
- обмежене покриття.

# ВИСНОВКИ

Тестування веб-додатків — це процес, який вимагає відданості та інтеграції спеціалізованих інструментів для забезпечення високоякісного програмного продукту. У наш час, коли швидкість розробки та змін програмного забезпечення є ключовим конкурентним фактором, автоматизоване тестування є необхідністю. У квалификаційної роботі ключові аспекти, пов’язані з розробкою та дослідженням ефективності реалізацій автоматизованого тестування веб-додатків.

Автоматизоване тестування було зроблено використовуючи:

- мову програмування c#;
- selenium WebDriver;
- бібліотека NUnit.

За допомогою цього тестувального додатку можна здійснювати ефективне тестування автоматизованих веб-додатків:

- дозволяє автоматизувати веб-інтерфейс будь якого веб-додатку завдяки selenium який взаємодіє з елементами веб-сторінок;
- паттерн проектування Page Object Model дозволяє відокремити логіку що полегшує підтримку коду тестів;
- зручне управління елементами веб-сторінок завдяки Page Factory;
- можливость підтримки роботи з декількома браузерами;
- можливость використовувати параметризовани тести;
- генерація логів у веб-інтерфейс завдяки ExtendReports.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Офіційна документація Selenium. https://www.selenium.dev/documentation/
2. Ліза Кріспін. Гнучке тестування: практичний посібник для тестувальників і гнучких команд.
3. Джейсон Арбон. Як Google тестує програмне забезпечення.
4. Бред Петтихорд Уроки тестування програмного забезпечення, 2001. 320 с.
5. Лора Томсон і Люк Веллінг, Розробка Web-додатків на PHP та MySQL, 2020. 768 с.
6. Гленфорд Маєрс. Мистецтво тестування програмного забезпечення. 2011. 178 с.
7. Ніксон Робін, Створюємо динамічні веб-сайти за допомогою PHP, MySQL, JavaScript, CSS та HTML5. 5-те вид. 2020. 816 с.
8. Пол Йоргенсон. Тестування програмного забезпечення: майстерний підхід. 2013. 494 с.
9. Сем Канер. Тестування комп'ютерного програмного забезпечення, 1999. 480 с.
10. Мартін Фаулер. Рефакторинг коду на JavaScript: покращення проекту існуючого коду. ,2020. 464 с.
11. Навніш Гарг. Автоматизація тестування за допомогою Selenium WebDriver, 2014. 404 с.
12. Унмеш Гундеча. Кулінарна книга інструментів тестування селену, 2012. 326 с.
13. Чжиминь Жан. Рецепти Selenium WebDriver у C#, 2015. 188 с.
14. Дороті Грейам. Основи тестування програмного забезпечення, 2006. 242 с.
15. Джеймс Уітакер Як зламати програмне забезпечення, 2002. 208 с.
16. Улізабет Хендріксон. Дослідіть це! Зменште ризик і підвищте впевненість за допомогою дослідницького тестування, 2013, 186 с.
17. Джеральд Мисарош. Тестові шаблони xUnit, 2007. 833 с.

18. Лі Копленд. Практичний посібник із розробки тестів програмного забезпечення, 2004. 312 с.
19. Сем Канер. Робоча книга з тестування домену, 2013. 488 с.
20. Ліза Кріспін. Більш спритне тестування: навчальні подорожі для всієї команди, 2014. 544 с.
21. Алан Пейдж. Як ми тестуємо програмне забезпечення в Microsoft, 2008. 405 с
22. Алі Мілі. Тестування програмного забезпечення: поняття та операції, 2015. 400 с.
23. Гаятрі Мохан. Тестування повного стека, 2022. 405 с.
24. Діана Ларсен. Agile Retrospectives: Робимо хороші команди чудовими, 2006. 178 с.
25. Байо Ерінле. Тестування продуктивності за допомогою JMeter 2.9, 2013. 148 с.
26. Ваверу Мваура. Learning PHP: Наскрізне веб-тестування за допомогою Cypress, 2021. 240 с.

**ДОДАТОК А.**

# ЛІСТІНГ ПРОГРАМИ

AlertsActuionsAutoSuggestive.cs:

```
using System;
using System.Collections.Generic;
using System.Threading;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Interactions;
using WebDriverManager.DriverConfigs.Impl;

namespace SeleniumLearning
{
   public class AlertsActionsAutoSuggestive
   {
      IWebDriver driver;
      [SetUp]

      public void StartBrowser()
      {
         new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
         driver = new ChromeDriver();
         driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);
         driver.Manage().Window.Maximize();
         driver.Url =
@"D:\курсач\finally\Magistr\Automation\Automation\Pages\AutomationPractice.html";
      }

      [Test]
      public void test_Alert()
      {
         string name = "Danil";
         driver.FindElement(By.CssSelector("#name")).SendKeys(name);
         driver.FindElement(By.CssSelector("input[onclick*='displayConfirm']")).Click();
         string alertText = driver.SwitchTo().Alert().Text;
         driver.SwitchTo().Alert().Accept();
         //   driver.SwitchTo().Alert().Dismiss();
         //  driver.SwitchTo().Alert().SendKeys("hello");

         StringAssert.Contains(name, alertText);
      }


      [Test]
      public void test_AutoSuggestiveDropDowns()
      {

         driver.FindElement(By.Id("autocomplete")).SendKeys("ind");
```

```
            Thread.Sleep(3000);

            IList<IWebElement> options = driver.FindElements(By.CssSelector(".ui-menu-item div"));

            foreach (IWebElement option in options)
            {
               if (option.Text.Equals("India"))

               {
                  option.Click();
               }
            }


TestContext.Progress.WriteLine(driver.FindElement(By.Id("autocomplete")).GetAttribute("value"))
;
         }

      }

}
```

FunctionalTest.cs:

```
using System;
using System.Collections.Generic;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using WebDriverManager.DriverConfigs.Impl;

namespace SeleniumLearning
{
   public class FunctionalTest
   {
      IWebDriver driver;

      [SetUp]
      public void StartBrowser()
      {
         new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
         driver = new ChromeDriver();

         driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);

         driver.Manage().Window.Maximize();
         driver.Url =
"file:///D:/%D0%BA%D1%83%D1%80%D1%81%D0%B0%D1%87/finally/Magistr/Automation/
Automation/Pages/login.html";
      }

      [Test]
```

```
        public void dropdown()
        {
            IWebElement dropdown = driver.FindElement(By.CssSelector("select.form-control"));

            SelectElement s = new SelectElement(dropdown);
            s.SelectByText("Teacher");
            s.SelectByValue("consult");
            s.SelectByIndex(1);

            IList <IWebElement> rdos = driver.FindElements(By.CssSelector("input[type='radio']"));

            foreach(IWebElement radioButton in rdos)
            {
              if(radioButton.GetAttribute("value").Equals("user"))
              {
                  radioButton.Click();
              }

            }
            WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(8));

wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions.ElementToBeClickable(By.Id("okayB
tn")));

            driver.FindElement(By.Id("okayBtn")).Click();
            bool result = driver.FindElements(By.Id("usertype"))[1].Selected;
            Assert.That(result, Is.True);

        }

    }
}
```

Locators.cs:

```
using System;
using System.Threading;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using WebDriverManager.DriverConfigs.Impl;

namespace SeleniumLearning
{
    public class Locators
    {
        IWebDriver driver;

        [SetUp]

        public void StartBrowser()
        {
```

```
            new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
            driver = new ChromeDriver();
            //Implicit wait 5seconds can be decalred globally
            //3 seconds
           // driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);

            driver.Manage().Window.Maximize();
            driver.Url =
"file:///D:/%D0%BA%D1%83%D1%80%D1%81%D0%B0%D1%87/finally/Magistr/Automation/
Automation/Pages/login.html";
        }

        [Test]
        public void LocatorsIdentification()
        {
            driver.FindElement(By.Id("username")).SendKeys("Bad");
            driver.FindElement(By.Id("username")).Clear();
            driver.FindElement(By.Id("username")).SendKeys("Baday");
            driver.FindElement(By.Name("password")).SendKeys("123456");
            //css selector & xpath
            //  tagname[attribute ='value']
            //    #id  #terms  - class name -> css .classname
            //    driver.FindElement(By.CssSelector("input[value='Sign In']")).Click();

            //    //tagName[@attribute = 'value']

            // CSS - .text-info span:nth-child(1) input
            //xpath - //label[@class='text-info']/span/input

            driver.FindElement(By.XPath("//div[@class='form-group'][5]/label/span/input")).Click();

            driver.FindElement(By.XPath("//input[@value='Sign In']")).Click();

            WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(8));
            wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions
           .TextToBePresentInElementValue(driver.FindElement(By.Id("signInBtn")), "Sign In"));

            string errorMessage = driver.FindElement(By.ClassName("alert-danger")).Text;
            TestContext.Progress.WriteLine(errorMessage);

            IWebElement link = driver.FindElement(By.LinkText("NMU"));
            string hrefAttr = link.GetAttribute("href");
            string expectedUrl = "https://www.nmu.org.ua/ua/";
            Assert.AreEqual(expectedUrl, hrefAttr);

            //validate url of the link text


        }

    }
}
```

SortWebTables.php:

```
using System;
using System.Collections;
using System.Collections.Generic;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using WebDriverManager.DriverConfigs.Impl;

namespace SeleniumLearning
{
   public class SortWebTables
   {
      IWebDriver driver;

      [SetUp]

      public void StartBrowser()
      {
         new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
         driver = new ChromeDriver();
         driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);
         driver.Manage().Window.Maximize();
         driver.Url =
"file:///D:/%D0%BA%D1%83%D1%80%D1%81%D0%B0%D1%87/finally/Magistr/Automation/
Automation/Pages/table.html#/offers";
      }

      [Test]

      public void SortTable()
      {
         ArrayList a = new ArrayList();
         SelectElement dropdown = new SelectElement(driver.FindElement(By.Id("page-menu")));
         dropdown.SelectByValue("20");

         // step 1 - Get all veggie names into arraylist A
        IList <IWebElement> veggies = driver.FindElements(By.XPath("//tr/td[1]"));

         foreach(IWebElement veggie in veggies)
         {
            a.Add(veggie.Text);
         }

         //step 2- Sort this arraylist  -A

         foreach (String element in a)
         {
            TestContext.Progress.WriteLine(element);
         }
```

```
            TestContext.Progress.WriteLine("After sorting");
            a.Sort();
            foreach( String element in a )
            {
               TestContext.Progress.WriteLine(element);
            }

            //step 3 - go and click column
            driver.FindElement(By.CssSelector("th[aria-label *= 'fruit name']")).Click();

            //step 4- Get all veggie names into arraylist B

            ArrayList b = new ArrayList();

            IList<IWebElement> sortedVeggies = driver.FindElements(By.XPath("//tr/td[1]"));

            foreach (IWebElement veggie in sortedVeggies)
            {
               b.Add(veggie.Text);
            }
            // arraylist A to B = equal
            Assert.AreEqual(a, b);
        }
    }
}
```

WindowHandlers.cs:

```
 using System;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using WebDriverManager.DriverConfigs.Impl;

namespace SeleniumLearning
{
   public class WindowHandlers
   {
      IWebDriver driver;

      [SetUp]

      public void StartBrowser()
      {
         new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
         driver = new ChromeDriver();
         driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);
         driver.Manage().Window.Maximize();
         driver.Url =
"file:///D:/%D0%BA%D1%83%D1%80%D1%81%D0%B0%D1%87/finally/Magistr/Automation/
Automation/Pages/login.html";
      }
```

```
        [Test]
        public void WindowHandle()
        {
            string email = "rector@nmu.org.ua";
            string parentWindowId = driver.CurrentWindowHandle;
            driver.FindElement(By.ClassName("blinkingText")).Click();

            Assert.AreEqual(2, driver.WindowHandles.Count); //1

            string childWindowName = driver.WindowHandles[1];

            driver.SwitchTo().Window(childWindowName);

            TestContext.Progress.WriteLine(driver.FindElement(By.CssSelector("#contacts >
p")).Text);
            string text = driver.FindElement(By.CssSelector("#contacts > p")).Text;

            string[] splittedText = text.Split("e-mail: ");

            string[] trimmedString = splittedText[1].Trim().Split(" ");

            Assert.AreEqual(email, trimmedString[0]);

            driver.SwitchTo().Window(parentWindowId);

            driver.FindElement(By.Id("username")).SendKeys(trimmedString[0]);
        }
    }
}
```

## CSarpSelFramework

### CheckoutPage.cs:

```
using System;
using System.Collections.Generic;
using OpenQA.Selenium;

using SeleniumExtras.PageObjects;

namespace CSharpSelFramework.pageObjects
{
    public class CheckoutPage
    {
        IWebDriver driver;
        private By productName = By.CssSelector("#myCartTable > tr > td:nth-child(2)");
        //driver.FindElements(By.CssSelector("h4 a"));
        public CheckoutPage(IWebDriver driver)
        {
            this.driver = driver;
            PageFactory.InitElements(driver, this);
```

```
        }

        //By.CssSelector(".btn-success")
        [FindsBy(How = How.XPath, Using = "//tbody[@id='myCartTable']//tr")]
        private IList<IWebElement> checkoutCards;

        [FindsBy(How = How.Id, Using = "btnBuy")]
        private IWebElement checkoutButton;


        public IList<IWebElement> getCards()
        {
            return checkoutCards;
        }

        public By getNameTitle()
        {
            return productName;
        }
        public void checkOut()
        {
            checkoutButton.Click();
            //object of next page
        }
    }
}
```

## LoginPage.cs

```
using System;
using System.Threading;
using OpenQA.Selenium;
using SeleniumExtras.PageObjects;

namespace CSharpSelFramework.pageObjects
{
    public class LoginPage
    {
        //driver.FindElement(By.Id("username"))
        //By.Id("username")

        //driver.FindElement(By.Name("password")).SendKeys("learning");
        //driver.FindElement(By.XPath("//div[@class='form-group'][5]/label/span/input")).Click();
        //driver.FindElement(By.XPath("//input[@value='Sign In']")).Click();
        private IWebDriver driver;

        public LoginPage(IWebDriver driver)
        {
            this.driver = driver;
            PageFactory.InitElements(driver, this);
        }
```

```
        //Pageobject factory

        [FindsBy(How = How.Id, Using = "username")]
        private IWebElement username;

        [FindsBy(How = How.Name, Using = "password")]
        private IWebElement password;

        [FindsBy(How = How.Id, Using = "password")]
        private IWebElement modalPassword;

        [FindsBy(How = How.Id, Using = "email")]
        private IWebElement modalEmail;

        [FindsBy(How = How.XPath, Using = "//*[@id='exampleModal']/div/div/div[3]/button[2]")]
        private IWebElement enterButton;

        [FindsBy(How = How.PartialLinkText, Using = "Hello User")]
        private IWebElement loginModal;

        [FindsBy(How = How.XPath, Using = "//div[@class='form-group'][5]/label/span/input")]
        private IWebElement checkBox;

        [FindsBy(How = How.CssSelector, Using = "input[value='Sign In']")]
        private IWebElement signInButton;

        public ProductsPage validLogin(string user, string pass)
        {
            username.SendKeys(user);
            password.SendKeys(pass);
            checkBox.Click();
            signInButton.Click();
            return new ProductsPage(driver);
        }
        public ProductsPage validModalLogin(string email, string pass)
        {
            loginModal.Click();
            modalEmail.SendKeys(email);
            modalPassword.SendKeys(pass);
            enterButton.Click();
            Thread.Sleep(2500);
            return new ProductsPage(driver);
        }

        public IWebElement getUserName()
        {
            return username;
        }
    }
}
```

ProductsPage.cs

```
using System;
using System.Collections.Generic;
using OpenQA.Selenium;
using OpenQA.Selenium.Support.UI;
using SeleniumExtras.PageObjects;

namespace CSharpSelFramework.pageObjects
{
   public class ProductsPage
   {

      IWebDriver driver;
      //By cardTitle = By.CssSelector(".card-title a");
      By cardTitle = By.CssSelector("#product > div > a > p:nth-child(2)");
      By addToCart = By.CssSelector("#product > div > button");

      public ProductsPage(IWebDriver driver)
      {
         this.driver = driver;
         PageFactory.InitElements(driver, this);

      }

      //[FindsBy(How = How.TagName, Using = "app-card")]
      [FindsBy(How = How.Id, Using = "product")]
      private IList<IWebElement> cards;

      [FindsBy(How = How.Id, Using = "Checkout")]
      private IWebElement checkoutButton;


      public void waitForPageDisplay()

      {

         WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(8));

wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions.ElementIsVisible(By.XPath("//h1[text
()='Welcome to our shop']")));
      }

      public IList<IWebElement> getCards()
      {

         return cards;
      }

      public By getCardTitle()
      {

         return cardTitle;
```

```
        }

        public CheckoutPage checkout()
        {

            checkoutButton.Click();
            return new CheckoutPage(driver);
        }

        public By addToCartButton()
        {

            return addToCart;
        }


    }
}
```

SortWebTables.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using WebDriverManager.DriverConfigs.Impl;

namespace SeleniumLearning
{
    [Parallelizable(ParallelScope.Self)]
    public class SortWebTables
    {
        IWebDriver driver;

        [SetUp]

        public void StartBrowser()

        {
            new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
            driver = new ChromeDriver();

            driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);

            driver.Manage().Window.Maximize();
            driver.Url =
"file:///D:/%D0%BA%D1%83%D1%80%D1%81%D0%B0%D1%87/finally/Magistr/Automation/
Automation/Pages/table.html#/offers";
        }
```

```
[Test]


public void SortTable()
{
   ArrayList a = new ArrayList();
   SelectElement dropdown = new SelectElement(driver.FindElement(By.Id("page-menu")));
   dropdown.SelectByValue("20");

   // step 1 - Get all veggie names into arraylist A
   IList <IWebElement> veggies = driver.FindElements(By.XPath("//tr/td[1]"));

   foreach(IWebElement veggie in veggies)
   {
      a.Add(veggie.Text);
   }

   //step 2- Sort this arraylist  -A

   foreach (String element in a)
   {
      TestContext.Progress.WriteLine(element);
   }

   TestContext.Progress.WriteLine("After sorting");
   a.Sort();
   foreach( String element in a)
   {
      TestContext.Progress.WriteLine(element);
   }

   //step 3 - go and click column
   driver.FindElement(By.CssSelector("th[aria-label *= 'fruit name']")).Click();

   //step 4- Get all veggie names into arraylist B

   ArrayList b = new ArrayList();

   IList<IWebElement> sortedVeggies = driver.FindElements(By.XPath("//tr/td[1]"));

   foreach (IWebElement veggie in sortedVeggies)
   {
      b.Add(veggie.Text);
   }

   // arraylist A to B = equal
   Assert.AreEqual(a, b);
   driver.Quit();
```

```
        }
    }
}
```

UnitTest.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using CSharpSelFramework.pageObjects;
using CSharpSelFramework.utilities;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using WebDriverManager.DriverConfigs.Impl;

namespace SeleniumLearning
{
    [Parallelizable(ParallelScope.Self)]
    public class E2ETest : Base
    {

        [Test, TestCaseSource("AddTestDataConfig"), Category("Regression")]


        [Parallelizable(ParallelScope.All)]
        public void EndToEndFlow(string username, string password, string[] expectedProducts)
        {
            string[] actualProducts = new string[2];
            LoginPage loginPage = new LoginPage(getDriver());
            ProductsPage productPage = loginPage.validLogin(username, password);
            productPage.waitForPageDisplay();
            productPage = loginPage.validModalLogin("sa@gmail.com", "Pa$$w0rd5");
            IList<IWebElement> products = productPage.getCards();

            foreach (IWebElement product in products)
            {
                if (expectedProducts.Contains(product.FindElement(productPage.getCardTitle()).Text))
                {
                    product.FindElement(productPage.addToCartButton()).Click();
                }
            }
            CheckoutPage checkoutPage = productPage.checkout();

            IList<IWebElement> checkoutCards = checkoutPage.getCards();

            for (int i = 0; i < checkoutCards.Count; i++)
            {
```

```
            //actualProducts[i] = checkoutCards[i].Text;
            actualProducts[i] = checkoutCards[i].FindElement(checkoutPage.getNameTitle()).Text;
        }
        Assert.AreEqual(expectedProducts, actualProducts);
        checkoutPage.checkOut();
    }



    [Test, Category("Smoke")]
    public void LocatorsIdentification()
    {
        driver.Value.FindElement(By.Id("username")).SendKeys("Baday");
        driver.Value.FindElement(By.Name("password")).SendKeys("wrong");
        //css selector & xpath
        //  tagname[attribute ='value']
        //    #id  #terms  - class name -> css .classname
        //    driver.FindElement(By.CssSelector("input[value='Sign In']")).Click();

        //    //tagName[@attribute = 'value']

        // CSS - .text-info span:nth-child(1) input
        //xpath - //label[@class='text-info']/span/input

        driver.Value.FindElement(By.XPath("//div[@class='form-
group'][5]/label/span/input")).Click();

        driver.Value.FindElement(By.XPath("//input[@value='Sign In']")).Click();

        WebDriverWait wait = new WebDriverWait(driver.Value, TimeSpan.FromSeconds(8));
        wait.Until(SeleniumExtras.WaitHelpers.ExpectedConditions
       .TextToBePresentInElementValue(driver.Value.FindElement(By.Id("signInBtn")), "Sign
In"));

        String errorMessage = driver.Value.FindElement(By.ClassName("alert-danger")).Text;
        TestContext.Progress.WriteLine(errorMessage);
    }
    public static IEnumerable<TestCaseData> AddTestDataConfig()
    {
        yield return new TestCaseData(getDataParser().extractData("username"),
getDataParser().extractData("password"), getDataParser().extractDataArray("products"));
        yield return new TestCaseData(getDataParser().extractData("username"),
getDataParser().extractData("password"), getDataParser().extractDataArray("products"));
        yield return new TestCaseData(getDataParser().extractData("username_wrong"),
getDataParser().extractData("password_wrong"), getDataParser().extractDataArray("products"));
    }
  }
}
```

## WindowHandlers.cs

```
using System;
```

```
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using WebDriverManager.DriverConfigs.Impl;

namespace SeleniumLearning
{
   [Parallelizable(ParallelScope.Self)]
   public class WindowHandlers
   {
      IWebDriver driver;

      [SetUp]

      public void StartBrowser()
      {
         new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
         driver = new ChromeDriver();

         driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);

         driver.Manage().Window.Maximize();
         driver.Url =
"file:///D:/%D0%BA%D1%83%D1%80%D1%81%D0%B0%D1%87/finally/Magistr/Automation/
Automation/Pages/login.html";
      }

      [Test]

      public void WindowHandle()
      {
         string email = "rector@nmu.org.ua";
         string parentWindowId = driver.CurrentWindowHandle;
         driver.FindElement(By.ClassName("blinkingText")).Click();

         Assert.AreEqual(2, driver.WindowHandles.Count);//1

         String childWindowName = driver.WindowHandles[1];

         driver.SwitchTo().Window(childWindowName);

         TestContext.Progress.WriteLine(driver.FindElement(By.CssSelector("#contacts >
p")).Text);
         string text = driver.FindElement(By.CssSelector("#contacts > p")).Text;

         string[] splittedText = text.Split("e-mail: ");

         string[] trimmedString = splittedText[1].Trim().Split(" ");

         Assert.AreEqual(email, trimmedString[0]);

         driver.SwitchTo().Window(parentWindowId);
```

```
            driver.FindElement(By.Id("username")).SendKeys(trimmedString[0]);
            driver.Quit();
        }
    }
}
```

## Base.cs

```
using System;
using System.Configuration;
using System.IO;
using System.Threading;
using AventStack.ExtentReports;
using AventStack.ExtentReports.Reporter;
using CSharpSelFramework.utilities;
using NUnit.Framework;
using NUnit.Framework.Interfaces;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Edge;
using OpenQA.Selenium.Firefox;
using WebDriverManager.DriverConfigs.Impl;

namespace CSharpSelFramework.utilities
{
    public class Base
    {
        public ExtentReports extent;
        public ExtentTest test;
        string browserName;
        //report file
        [OneTimeSetUp]
        public void Setup()
        {
            string workingDirectory = Environment.CurrentDirectory;
            string projectDirectory = Directory.GetParent(workingDirectory).Parent.Parent.FullName;
            string reportPath = projectDirectory + "//index.html";
            var htmlReporter = new ExtentHtmlReporter(reportPath);
            extent = new ExtentReports();
            extent.AttachReporter(htmlReporter);
            extent.AddSystemInfo("Host Name", "Localhost");
            extent.AddSystemInfo("Environment", "QA");
            extent.AddSystemInfo("Username", "Danil Baday");
        }

        // public IWebDriver driver;
        public ThreadLocal<IWebDriver> driver = new ThreadLocal<IWebDriver>();
        [SetUp]

        public void StartBrowser()
        {
```

```
        test = extent.CreateTest(TestContext.CurrentContext.Test.Name);
          //Configuration
          browserName = TestContext.Parameters["browserName"];
          if (browserName == null)
          {
             browserName = ConfigurationManager.AppSettings["browser"];
             //browserName = "Chrome";
          }
          InitBrowser(browserName);

          driver.Value.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(5);

          driver.Value.Manage().Window.Maximize();
          driver.Value.Url =
"file:///D:/%D0%BA%D1%83%D1%80%D1%81%D0%B0%D1%87/finally/Magistr/Automation/
Automation/Pages/login.html";
      }

      public IWebDriver getDriver()
      {
         return driver.Value;
      }

      public void InitBrowser(string browserName)
      {

         switch (browserName)
         {
            case "Firefox":
               new WebDriverManager.DriverManager().SetUpDriver(new FirefoxConfig());
               driver.Value = new FirefoxDriver();
               break;

            case "Chrome":
               new WebDriverManager.DriverManager().SetUpDriver(new ChromeConfig());
               driver.Value = new ChromeDriver();
               break;

            case "Edge":
               driver.Value = new EdgeDriver();
               break;
         }
      }

      public static JsonReader getDataParser()
      {
         return new JsonReader();
      }

      [TearDown]
      public void AfterTest()
      {
```

```
            var status = TestContext.CurrentContext.Result.Outcome.Status;
            var stackTrace = TestContext.CurrentContext.Result.StackTrace;



            DateTime time = DateTime.Now;
            String fileName = "Screenshot_" + time.ToString("h_mm_ss") + ".png";

            if (status == TestStatus.Failed)
            {

               test.Fail("Test failed", captureScreenShot(driver.Value, fileName));
               test.Log(Status.Fail,"test failed with logtrace" + stackTrace);

            }
            else if (status == TestStatus.Passed)
            {

            }

            extent.Flush();
            driver.Value.Quit();
        }

        public MediaEntityModelProvider captureScreenShot(IWebDriver driver,String
screenShotName)
        {
          ITakesScreenshot ts = (ITakesScreenshot)driver;
          var screenshot= ts.GetScreenshot().AsBase64EncodedString;

          return MediaEntityBuilder.CreateScreenCaptureFromBase64String(screenshot,
screenShotName).Build();
        }
    }
}
```

## MainPage.php

```
<?php
if (!isset($_SESSION))
   session_start();

require 'phpScripts/ProductInfo.php';
require 'phpScripts/UserInfo.php';
require 'phpScripts/ServerInfo.php';


if(isset($_SESSION['key']))
{
   $userInfo = new UserInfo($_SESSION['key'], $_SESSION['email'], $_SESSION['name']);
   $btn = $userInfo->IsAdmin();
}
```

```
else
{
   $userInfo = new UserInfo(",",");
}
$productInfo = new ProductInfo;
$serverInfo = new ServerInfo;

$info = $productInfo->info;
$goods = $productInfo->goods;
$goodsArr = $productInfo->goodsArr;
$modal = $userInfo->ModalWindowType();
$cartModal = $userInfo->ModalWindowCartType();
$myCart = null;


$text = "Catalog";
if($serverInfo->IsMobile())
{
   $text = "";
}


if(isset($_SESSION['savingCart'])) {
   $myCart = $_SESSION['savingCart'];
}


?>


<!DOCTYPE html>
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">

   <link rel="stylesheet" href="css/myStyles.css">

   <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css">
   <!-- Custom css -->
   <link rel="stylesheet" href="css/default.css">
   <!-- Main css -->
   <link rel="stylesheet" href="css/style.css">
   <!-- Responsive css -->
   <link rel="stylesheet" href="css/responsive.css">
   <!-- Fontawesome css -->
   <link rel="stylesheet" href="css/font-awesome.min.css">
   <!-- Ionicons css -->
   <link rel="stylesheet" href="css/ionicons.min.css">
   <!-- linearicons css -->
   <link rel="stylesheet" href="css/linearicons.css">
```

```
    <!--link rel="stylesheet" href="css/search.css"-->
    <!-- AJAX and Bootstrap scripts -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js"></script>
    <!-- Sliders scripts -->
    <script src="js/jsslider.min.js" type="text/javascript"></script>
    <script src="js/jsslider.js" type="text/javascript"></script>
    <!-- Buying script -->
    <script src="js/__Buy.js"></script>
    <!-- Search script -->
    <!--script src="js/search.js"></script-->

        <style>
            hr {
                border:1px solid rgb(211, 211, 211);
                border-bottom-width: 0;
            }

            .jssorl-009-spin img {
                animation-name: jssorl-009-spin;
                animation-duration: 1.6s;
                animation-iteration-count: infinite;
                animation-timing-function: linear;
            }

            @keyframes jssorl-009-spin {
                from { transform: rotate(0deg); }
                to { transform: rotate(360deg); }
            }


            .jssorb051 .i {position:absolute;cursor:pointer;}
            .jssorb051 .i .b {fill:#fff;fill-opacity:0.5;}
            .jssorb051 .i:hover .b {fill-opacity:.7;}
            .jssorb051 .iav .b {fill-opacity: 1;}
            .jssorb051 .i.idn {opacity:.3;}


            .jssora051 {display:block;position:absolute;cursor:pointer;}
            .jssora051 .a {fill:none;stroke:#fff;stroke-width:360;stroke-miterlimit:10;}
            .jssora051:hover {opacity:.8;}
            .jssora051.jssora051dn {opacity:.5;}
            .jssora051.jssora051ds {opacity:.3;pointer-events:none;}
        </style>


    <title>Main Page</title>
<body class="background">

<div class="jumbotron text-center bg-secondary" style="margin-bottom:0">
```

```
   <h1>Welcome to our shop</h1>
   <p>We are happy to see you there</p>
</div>


<!-- Start Navbar -->
<div class="pos-f-t">
   <nav id="anchor" class="navbar navbar-dark bg-dark">
      <div class="form-inline">
         <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarToggleExternalContent" aria-controls="navbarToggleExternalContent" aria-
expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span> <?php echo $text?>
         </button>

      </div>


      <div class="form-inline">
         <a href="#" class="mr-4" data-toggle="modal" data-target="#exampleModal"><?php
if(!isset($_SESSION['key'])): echo 'Hello User'; else: echo $_SESSION['name']; endif;?></a>
         <!--button class="btn btn-outline-success mr-sm-2">Compare</button-->
         <button id="Checkout" data-toggle="modal" data-target="#cartModal" class="btn btn-
outline-success my-2 my-sm-0"><span><img src="fonts/icon_cart.svg" width="20px"
height="20px"></span>
             Cart  <span class="badge badge-dark" id="cartCount"><?php
if(isset($_SESSION['count'])): echo $_SESSION['count']; endif;?></span></button>
      </div>




      <!-- Cart Modal -->
      <?php if(empty($cartModal)): echo "<div class=\"modal fade\" id=\"cartModal\" tabindex=\"-
1\" role=\"dialog\" aria-labelledby=\"exampleModalLongTitle\" aria-hidden=\"true\">
  <div class=\"modal-dialog\" role=\"document\">
   <div class=\"modal-content\">
    <div class=\"modal-header\">
     <h5 class=\"modal-title\" id=\"cartModal\">Add to cart</h5>
     <button type=\"button\" class=\"close\" data-dismiss=\"modal\" aria-label=\"Close\">
      <span aria-hidden=\"true\">&times;</span>
     </button>
    </div>
    <div class=\"modal-body\">
     <table class=\"table table-striped\" style=\"cursor: pointer\">
        <thead>
        <tr>
           <th>Id</th>
           <th>Product Name</th>
           <th>Firm Name</th>
           <th>Type</th>
           <th>Price</th>
```

```
          </tr>
          </thead>
          <tbody id=\"myCartTable\">
             $myCart
          </tbody>
        </table>
      </div>
      <div class=\"modal-footer\">
      <form method='post' action='phpScripts/clearCart.php'>
        <button onclick=\"clear()\" type=\"submit\" class=\"btn btn-danger\">Clear</button>
        </form>
        <button type=\"button\" class=\"btn btn-secondary\" data-dismiss=\"modal\">Close</button>
        <form method='post' action='phpScripts/buy.php'>
        <button onclick=\"buy()\" type=\"submit\" class=\"btn btn-primary\"
id=\"btnBuy\">Buy</button>
        </form>
      </div>
    </div>
  </div>
</div>"; endif;?>
       <!--End Cart Modal -->


       <!-- Cabinet Modal -->
      <?php echo $modal?>
       <!--End Cabinet Modal -->

    </nav>
    <!-- Categories -->
    <div class="collapse" id="navbarToggleExternalContent">
       <div class="bg-dark p-4">
          <h4 class="text-white">Products</h4>
          <hr><br>
          <ul>
             <li><a href="Smartphones.php">Smartphones</a></li>
             <li><a href="Laptops.php">Laptops</a></li>
             <li><a href="Tvs.php">TVs</a></li>
          </ul>
          <br><hr><br>
          <!-- Categories -->
          <!-- Searcher -->
          <!--form class="form-inline my-2 my-lg-0">
             <input name="search" class="form-control mr-sm-2" type="search"
placeholder="Search" aria-label="Search">
             <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>
          </form-->
          <form method="GET" action="searchPage.php">
             <div class="row form-group">
                <div class="col-sm-10">
                   <input name="search" class="form-control mr-sm-2" type="search"
autocomplete="off" placeholder="I am looking for..." aria-label="Search">
                   <ul class="search_result"></ul>
```

```
            </div>
            <div class="col-sm-2">
               <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Searching</button>
            </div>
         </div>
      </form>

      <!-- Searcher -->
      <hr><br>

      <!-- First checkbox row -->
      <form method="post" action="SortedPage.php">
      <div class="row mb-5">
         <div class="col-sm-3">
            <div class="custom-control custom-checkbox">
               <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="Apple" id="checkbox1">
               <label class="custom-control-label text-light" for="checkbox1">
                  Apple
               </label>
            </div>
         </div>
         <div class="col-sm-3">
            <div class="custom-control custom-checkbox">
               <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="Google" id="checkbox2">
               <label class="custom-control-label text-light" for="checkbox2">
                  Google
               </label>
            </div>
         </div>
         <div class="col-sm-3">
            <div class="custom-control custom-checkbox">
               <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="Xiaomi" id="checkbox3">
               <label class="custom-control-label text-light" for="checkbox3">
                  Xiaomi
               </label>
            </div>
         </div>
         <div class="col-sm-3">
            <div class="custom-control custom-checkbox">
               <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="Meizu" id="checkbox4">
               <label class="custom-control-label text-light" for="checkbox4">
                  Meizu
               </label>
            </div>
         </div>
      </div>
      <!-- First checkbox row -->
```

```
        <!-- Second checkbox row -->
        <div class="row mb-5">
          <div class="col-sm-3">
            <div class="custom-control custom-checkbox">
              <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="HP" id="checkbox5">
              <label class="custom-control-label text-light" for="checkbox5">
                HP
              </label>
            </div>
          </div>
          <div class="col-sm-3">
            <div class="custom-control custom-checkbox">
              <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="Lenovo" id="checkbox6">
              <label class="custom-control-label text-light" for="checkbox6">
                Lenovo
              </label>
            </div>
          </div>
          <div class="col-sm-3">
            <div class="custom-control custom-checkbox">
              <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="Huawei" id="checkbox7">
              <label class="custom-control-label text-light" for="checkbox7">
                Huawei
              </label>
            </div>
          </div>
          <div class="col-sm-3">
            <div class="custom-control custom-checkbox">
              <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="Samsung" id="checkbox8">
              <label class="custom-control-label text-light" for="checkbox8">
                Samsung
              </label>
            </div>
          </div>
        </div>
        <!-- Second checkbox row -->


        <!-- Third checkbox row -->
        <div class="row mb-5">
          <div class="col-sm-3">
            <div class="custom-control custom-checkbox">
              <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="Asus" id="checkbox9">
              <label class="custom-control-label text-light" for="checkbox9">
                Asus
              </label>
```

```
                </div>
            </div>
            <div class="col-sm-3">
                <div class="custom-control custom-checkbox">
                    <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="Acer" id="checkbox10">
                    <label class="custom-control-label text-light" for="checkbox10">
                        Acer
                    </label>
                </div>
            </div>
            <div class="col-sm-3">
                <div class="custom-control custom-checkbox">
                    <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="MSI" id="checkbox11">
                    <label class="custom-control-label text-light" for="checkbox11">
                        MSI
                    </label>
                </div>
            </div>
            <div class="col-sm-3">
                <div class="custom-control custom-checkbox">
                    <input class="custom-control-input" type="checkbox" name="filterCheckBox[]"
value="Razer" id="checkbox12">
                    <label class="custom-control-label text-light" for="checkbox12">
                        Razer
                    </label>
                </div>
            </div>
        </div>
        <!-- Third checkbox row -->

        <div class="row">
            <div class="col-sm">
                <button type="submit" class="btn btn-outline-success">Sorted</button>
            </div>
        </div>
        </form>

      </div>
   </div>
</div>

<!-- End Navbar-->

<br>


<!-- Start Slider -->
<div id="jssor_1" style="position:relative;margin:0
auto;top:0px;left:0px;width:980px;height:380px;overflow:hidden;visibility:hidden;">
   <!-- Loading Screen -->
```

```
    <div data-u="loading" class="jssorl-009-spin"
style="position:absolute;top:0px;left:0px;width:100%;height:100%;text-align:center;background-
color:rgba(0,0,0,0.7);">
        <img style="margin-top:-19px;position:relative;top:50%;width:38px;height:38px;"
src="js/img/spin.svg" />
    </div>
    <div data-u="slides"
style="cursor:default;position:relative;top:0px;left:0px;width:980px;height:380px;overflow:hidden;
">
        <div>
            <img data-u="image" src="img/adverisment/back_to_school_desk.png" />
        </div>
        <div>
            <img data-u="image" src="img/adverisment/desktop_in_stock.png" />
        </div>
        <div>
            <img data-u="image" src="img/adverisment/mac_desk.png" />
        </div>
        <div>
            <img data-u="image" src="img/adverisment/1300x500_shop-partner_2_.jpg" />
        </div>

    </div>
    <!-- Bullet Navigator -->
    <div data-u="navigator" class="jssorb051" style="position:absolute;bottom:12px;right:12px;"
data-autocenter="1" data-scale="0.5" data-scale-bottom="0.75">
        <div data-u="prototype" class="i" style="width:16px;height:16px;">
            <svg viewbox="0 0 16000 16000"
style="position:absolute;top:0;left:0;width:100%;height:100%;">
                <circle class="b" cx="8000" cy="8000" r="5800"></circle>
            </svg>
        </div>
    </div>
    <!-- Arrow Navigator -->
    <div data-u="arrowleft" class="jssora051" style="width:55px;height:55px;top:0px;left:25px;"
data-autocenter="2" data-scale="0.75" data-scale-left="0.75">
        <svg viewbox="0 0 16000 16000"
style="position:absolute;top:0;left:0;width:100%;height:100%;">
            <polyline class="a" points="11040,1920 4960,8000 11040,14080 "></polyline>
        </svg>
    </div>
    <div data-u="arrowright" class="jssora051" style="width:55px;height:55px;top:0px;right:25px;"
data-autocenter="2" data-scale="0.75" data-scale-right="0.75">
        <svg viewbox="0 0 16000 16000"
style="position:absolute;top:0;left:0;width:100%;height:100%;">
            <polyline class="a" points="4960,1920 11040,8000 4960,14080 "></polyline>
        </svg>
    </div>
</div>
<!-- End Slider -->
```

```
<br><br>

<div class="container" id="maindiv">
   <center><h1 class="bg-secondary">Our products</h1></center>
<br>

   <?php echo $info?>
      <div class="row" id="catalog">
         <?php foreach($goodsArr['smartphones'] as $item): echo $item; endforeach; ?>
      </div>
   <?php if(!empty($goodsArr['laptops'])): echo "<hr><br>"; endif;?>
   <div class="row">
      <?php if(!empty($goodsArr['laptops'])): foreach ($goodsArr['laptops'] as $item): echo $item;
endforeach; endif;?>
   </div>
   <?php if(!empty($goodsArr['tvs'])): echo "<hr><br>"; endif;?>
   <div class="row">
      <?php if(!empty($goodsArr['tvs'])): foreach ($goodsArr['tvs'] as $item): echo $item;
endforeach; endif;?>
   </div>
</div>




<!-- Footer Area Start Here -->
<footer class="off-white-bg2 pt-95 bdr-top pt-sm-55">
   <!-- Footer Top Start -->
   <div class="footer-top">
      <div class="container">
         <!-- Signup Newsletter Start -->
         <div class="row mb-60">
            <div class="col-xl-7 col-lg-7 ml-auto mr-auto col-md-8">
               <div class="news-desc text-center mb-30">
                  <h3>Subscribe!</h3>
                  <p>Be nice guys and subscribe to get unuseful distribution</p>
               </div>
               <div class="newsletter-box">
                  <form action="#">
                     <input class="subscribe" placeholder="your email address" name="email"
id="subscribe" type="text">
                     <button type="submit" class="submit">Subscribe!</button>
                  </form>
               </div>
            </div>
         </div>
         <!-- Signup-Newsletter End -->
         <div class="row">
            <!-- Single Footer Start -->
            <div class="col-lg-2 col-md-4 col-sm-6">
               <div class="single-footer mb-sm-40">
                  <h3 class="footer-title">Information</h3>
```

```
                    <div class="footer-content">
                        <ul class="footer-list">
                            <li><a href="#">About Us</a></li>
                            <li><a href="#">Delivery info</a></li>
                            <li><a href="#">Private Policy</a></li>
                            <li><a href="#">FAQ</a></li>
                            <li><a href="#">Return Policy</a></li>
                        </ul>
                    </div>
                </div>
            </div>

            <div class="col-lg-4 col-md-6 col-sm-6">
                <div class="single-footer mb-sm-40">
                    <h3 class="footer-title">Contacts</h3>
                    <div class="footer-content">
                        <ul class="footer-list address-content">
                            <li><i class="lnr lnr-map-marker"></i> Address: Fog Street 99.</li>
                            <li><i class="lnr lnr-envelope"></i><a href="https://mail.google.com">Mail:
bidshop@gmail.com</a></li>
                            <li>
                                <i class="lnr lnr-phone-handset"></i> Telephone: (+380) 96 666 6666)
                            </li>
                        </ul>

                    </div>
                </div>
            </div>
            <!-- Single Footer Start -->
        </div>
        <!-- Row End -->
    </div>
    <!-- Container End -->
</div>
<!-- Footer Top End -->

<!-- Footer Bottom Start -->
<div class="footer-bottom pb-30">
    <div class="container">

        <div class="copyright-text text-center">
            <p>Copyright © 2019 <a target="_blank" href="#">Badayindustries</a> All Rights
Reserved.</p>
        </div>
    </div>
    <!-- Container End -->
</div>
<!-- Footer Bottom End -->
</footer>
<!-- Footer Area End Here -->
```

```
    <script type="text/javascript">
        $(document).ready(function(){
            $("#menu").on("click","a", function (event) {
                event.preventDefault();
                var id  = $(this).attr('href'),
                    top = $(id).offset().top;
                $('body,html').animate({scrollTop: top}, 1500);
            });
        });
    </script>
<script type="text/javascript">jssor_1_slider_init();</script>
</body>
```

Всі інші файли можна буде переглянути на на cd диску, на якому буде архів з готовим проектом.

**ДОДАТОК В**

# ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

| Ім’я файлу | Опис |
|---|---|
| Пояснювальні документи | |
| Кваліфікаційна робота Бадай.docx | Пояснювальна записка до кваліфікаційної роботи. Документ Word. |
| Кваліфікаційна робота Бадай.pdf | Пояснювальна записка до кваліфікаційної роботи в форматі PDF. |
| Програма | |
| Бадай.rar | Архів. Містить коди програми і скомпільовану програму. |
| Презентація | |
| Бадай.ppt | Презентація кваліфікаційної роботи. |