СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВОЛОДИМИРА ДАЛЯ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ПРОГРАМУВАННЯ

Пояснювальна записка

до бакалаврської дипломної роботи

(освітньо-кваліфікаційний рівень)

на тему: Утиліта для автоматизації виконання завдань QA-тестувальника

Виконав: студент 4 курсу, групи ІПЗ -19д

спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Лобанов Тимофій Володимирович
(прізвище та ініціали)

Керівник доц., к.т.н. Іванов В.Г
(прізвище та ініціали)

Рецензент доц., к.т.н. Ратов Д. В.
(прізвище та ініціали)

Київ - 2023

ЗМІСТ

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи бакалавру складається з: 43 сторінок основного тексту, 18 рисунків, 1 додатку, 10 використаних джерел.

Метою даної бакалаврської дипломної роботи є розробка програмного забезпечення дня тестування сайту Східноукраїнського національного університету імені Володимира Даля, з метою перевірки його працездатності та взаємодії з користувачем.

Об’єктом дослідження є розробка програмного забезпечення для тестування сайту Східноукраїнського національного університету імені Володимира Даля за допомогою мови програмування Python, фрейморку Selenium та IDE PyCharm.

Головні завдання роботи:

1. Аналіз предметної області.
2. Огляд предметної галузі.
3. Огляд існуючих рішень.
4. Розробка моделі сайту та архітектури алгоритму ПЗ
5. Реалізація моделі сайту та архітектури алгоритму ПЗ
6. Апробація програмного забезпечення.

Методи дослідження – аналіз предметної області буде виконаний з акцентом на ієрархічне представлення об'єктів та сформованими логічними зв'язки між процесами.

Ключові слова: ПРОЄКТ, QA, ТЕСТУВАЛЬНИК, PYCHARM, PYTHON**,** IDE, SELENIUM, ФРЕЙМВОРК

ВСТУП

Автоматизоване тестування якості (QA) стало невід'ємною частиною розробки веб-додатків та сайтів. Воно дозволяє забезпечити високий рівень надійності, функціональності та продуктивності веб-платформи, а також скоротити час та ресурси, що витрачаються на тестування.

У сучасному інформаційному суспільстві вебсайти є одним із ключових каналів комунікації між організацією та її клієнтами. Задоволення потреб користувача та забезпечення бездоганної працездатності сайту стають важливими пріоритетами для компаній, включаючи університети.

У рамках даної дипломної роботи було розроблено проєкт, спрямований на автоматичне тестування сайту Східноукраїнського національного університету імені Володимира Даля, з метою перевірки його працездатності та взаємодії з користувачем.

Метою даного проєкту є розробка програмного забезпечення, здатного автоматизувати процес тестування сайту університету на різні аспекти, одними з них є взаємодія з користувачем і коректність роботи інтерактивних елементів сайту. Автоматизоване тестування дозволяє значно скоротити час та ресурси, що витрачаються на ручне тестування, а також забезпечити підвищену точність та надійність результатів.

Для реалізації цього проекту використовувалася мова програмування Python у розробці PyCharm. Python був обраний завдяки своїй простоті, гнучкості та багатому екосистемі інструментів для автоматизації тестування. Як інструмент для взаємодії з веб-сайтом був використаний веб-драйвер Selenium, який надає можливості для керування браузером та виконання дій на веб-сторінці.

Основними завданнями проекту було створення набору тестових сценаріїв, що взаємодіють із різними елементами сайту, а також розробка механізму перевірки працездатності цих елементів. В рамках проєкту було здійснено тестування функціональності сайту, перевірку коректності роботи слайд-меню та аналіз роботи поля пошуку. Усі результати тестування були

записані до звіту, що дозволяє швидко оцінити працездатність сайту та виявити потенційні проблеми.

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Автоматизоване тестування веб-сайтів є важливим етапом у розробці програмного забезпечення. Забезпечення якісної функціональності та стабільності веб-додатків є критичним для забезпечення задоволення користувачів та впевненості у роботі сайту.

Аналіз предметної області передбачає детальне вивчення вимог та функціональності сайту університету для побудови ефективних тестових сценаріїв та розробки автоматизованих тестів. При проведенні аналізу, слід враховувати наступні аспекти:

1.1 Функціональні вимоги сайту:

Перед початком роботи над автоматизованими тестами, необхідно з'ясувати функціональні вимоги до веб-додатку. Це означає визначення основних функцій, які має виконувати сайт, таких як реєстрація користувачів, авторизація, навігація, робота з формами, обробка даних тощо.

1.2 Навігація та структура сайту:

Дослідження навігації та структури сайту дозволяє зрозуміти логіку переходів між сторінками, особливості внутрішнього побудови та взаємозв'язків між елементами. Це допомагає визначити ключові сторінки та елементи, які потрібно враховувати при тестуванні.

1.3 Особливості введення та виведення даних:

Аналіз введення та виведення даних на сайті включає вивчення форм, таблиць, списків, повідомлень та інших елементів, які взаємодіють з користувачем. Це допомагає виявити важливі поля для перевірки, валідацію даних, обробку помилок та повідомлення про стан системи.

1.4 Сценарії тестування:

На основі отриманих вимог та досліджень предметної області, необхідно створити сценарії тестування, які включають в себе послідовність

дій користувача та очікувані результати. Сценарії тестування допоможуть визначити обсяг роботи та необхідність автоматизації певних функцій сайту.

1.5 Постановка задачі

Метою кваліфікаційної роботи є розробка системи автоматизованого тестування веб-сайту Східноукраїнського національного університету імені Володимира Даля з метою перевірки його функціональності, стабільності та надійності. З урахуванням результатів аналізу предметної області, були поставлені наступні задачі:

1. Розробити набір тестових сценаріїв, які охоплюють основні функції та сценарії взаємодії користувачів з веб-додатком.

2. Визначити найоптимальніший метод пошуку елементів для забезпечення стабільності тестів та швидкості їх виконання.

3. Розробити алгоритм пошуку та локалізації елементів на сторінці веб-сайту.

4. Перевірити коректність введення та виведення даних на сайті, включаючи форми, таблиці, повідомлення та інші елементи взаємодії з користувачем.

5. Забезпечити стабільність, надійність та швидкість виконання автоматизованих тестів на веб-сайті.

Виконання цих задач дозволить забезпечити високу якість програмного забезпечення та покращити користувацький досвід сайту університету.

1.6 Методологія тестування

Методологія тестування [8] є систематичним підходом до планування, виконання та оцінки процесу тестування програмного забезпечення. Вона визначає набір кроків, правил та процедур, які допомагають забезпечити ефективність та якість тестування. Основна мета методології тестування полягає в виявленні помилок [9], визначенні якості продукту та забезпеченні відповідності вимогам користувачів.

Основні принципи, які можуть бути використані в методології тестування, включають наступні [9]:

1. Планування тестування: Цей крок передбачає визначення цілей, обсягу та стратегії тестування. Він включає складання плану тестування, визначення ресурсів та визначення тестових вимог.

2. Аналіз вимог: Важливо ретельно вивчити вимоги до програмного забезпечення, щоб зрозуміти очікувану функціональність та вимоги до продукту. Це допомагає побудувати відповідні тестові сценарії та тестові набори.

3. Розробка тестових сценаріїв: Тестові сценарії визначають послідовність дій, які необхідно виконати для перевірки функціональності продукту. Вони описують очікувані результати та умови, за яких тест вважається успішним або невдалим.

4. Виконання тестів: У цьому етапі тестові сценарії виконуються, а результати тестування реєструються. Помилки та проблеми фіксуються і повідомляються розробникам для виправлення.

5. Аналіз результатів: Після виконання тестів аналізуються результати для виявлення помилок, трендів та шаблонів. Це допомагає виправити проблеми та вдосконалити якість продукту.

6. Звітність: Після завершення тестування складається звіт, який містить інформацію про виконані тести, виявлені помилки та рекомендації щодо покращення продукту.

Популярні методологій тестування [8]:

1. Водопадна модель (Waterfall Model): Ця методологія передбачає послідовне виконання етапів розробки та тестування від початку до кінця. Кожен етап має чітко визначені вхідні та вихідні дані, і перехід до

наступного етапу відбувається після завершення попереднього. Цей підхід підходить для проектів з передбачуваними та сталими вимогами.

2. Ітераційна модель (Iterative Model): Ця методологія передбачає ітераційний цикл розробки та тестування. Процес поділяється на короткі ітерації, кожна з яких включає в себе фази розробки, тестування та зворотного зв'язку. Кожна ітерація додає новий функціонал до продукту, а тестування проводиться на кожному етапі. Цей підхід підходить для проектів зі змінними вимогами та можливістю поетапної розробки.

3. Agile-методології: Agile включає в себе низку ітераційних методологій розробки, таких як Scrum, Kanban та Extreme Programming (XP). Ці методології покладають акцент на гнучкість, співпрацю та швидкі ітерації. Вони покликані забезпечити високу реагувальність на змінюючіся вимоги та впроваджувати зміни на кожному етапі розробки та тестування.

4. Тестування згідно специфікацій (Specification-based Testing): Цей підхід ґрунтується на аналізі вимог і створенні тестів на основі специфікацій програмного забезпечення. Використовуються техніки, такі як тестування еквівалентних класів, граничних значень та таблиць припущень. Метою є переконатися, що програмне забезпечення виконує визначені специфікації.

5. Тестування на основі скриптів (Scripted Testing): Цей підхід включає створення скриптів тестування, які визначають послідовність дій та очікувані результати. Скрипти виконуються автоматично, що дозволяє ефективно виконувати повторювані тести. Цей метод добре підходить для автоматизованого тестування.

РОЗДІЛ 2 ОГЛЯД ПРЕДМЕТНОЇ ГАЛУЗІ

У цьому розділі розглянемо інструменти та методи, що використовуються для автоматичного тестування веб-сайту університету. Застосування автоматизації тестування веб-додатків у сфері освіти є особливо важливим, оскільки забезпечує високу якість та ефективність тестового процесу.

Один із інструментів, який буде використано, є Selenium - потужний фреймворк для автоматизованого тестування веб-додатків [1]. Він дозволяє створювати скрипти для автоматизації взаємодії з веб-елементами, заповнення форм, навігації по сторінкам та перевірки очікуваного результату. Selenium надає можливість ефективно взаємодіяти з різними браузерами, такими як Chrome, Firefox, а також виконувати тести на різних операційних системах.

Крім того, буде розглянуто методи роботи з різними типами веб-елементів, які можна знайти на сайті університету. Зокрема, будуть розглядатися методи пошуку елементів за тегом, класом, CSS селектором та іншими атрибутами. Це дозволить точно вказувати, з якими елементами буде взаємодія з допомогою тестів.

Важливо також звернути увагу на структуру та логіку сайту університету. Для полегшення орієнтації та зручного написання тестів буде розроблена загальна модель сайту, яка буде відображати основні сторінки, елементи та шаблони, що використовуються. Це допоможе організувати тестовий код та забезпечити стабільність тестів при зміні структури сайту.

Останнім кроком буде створення алгоритму пошуку елементів на сайті університету та запис цього алгоритму у звіт. Буде ретельно проаналізована структура HTML-сторінок та визначимо, які атрибути елементів можна використовувати для їхнього ідентифікування. Запис алгоритму у звіт дозволить зрозуміти, як саме взаємодіють конкретні елементи сайту з користувачами з конкретними елементами сайту.

Таким чином, у даному розділі проведемо аналіз предметної області,

описавши основні інструменти та методи роботи з автоматичним тестуванням веб-сайту університету. Постановка задачі буде полягати у розробці ефективного тестового середовища, яке дозволить забезпечити якість, стабільність та швидкість виконання тестів.

2.1 Python

Python [2] - це високорівнева, інтерпретована мова програмування з простим синтаксисом і читабельним кодом. Вона здатна працювати на різних платформах і має широкий спектр бібліотек та фреймворків, що робить її популярним вибором для автоматизованого тестування.

Одним з головних переваг Python у контексті тестування є його простота в освоєнні. Мова має лаконічний синтаксис, що дозволяє швидко розробляти тестові скрипти без зайвих зусиль..

Python має потужну та розширювану стандартну бібліотеку, яка включає модулі для роботи з мережею, регулярними виразами, файловою системою та багатьма іншими аспектами, що зустрічаються в автоматизованому тестуванні. Крім того, існують спеціалізовані бібліотеки, такі як Selenium, PyTest та Robot Framework, які полегшують написання тестів і прискорюють розробку.

Python є зручним інструментом для автоматизованого тестування. Він дозволяє розробляти стабільні, читабельні та підтримувані тести зі швидкістю і ефективністю.

2.2 IDE PyCharm

PyCharm [3] - це інтегроване середовище розробки (IDE) для мови програмування Python. PyCharm надає розширений набір інструментів та функціональностей, які полегшують процес тестування та забезпечують зручне середовище для розробки автоматизованих тестів веб-додатків.

Основні переваги PyCharm для тестування веб-сайтів:

1. Підтримка Selenium: PyCharm надає вбудовану підтримку для Selenium - популярного інструменту для автоматизації веб-тестування. Це

означає, що можна легко налаштувати інтеграцію з Selenium, використовуючи PyCharm, і зручно писати та виконувати веб-тести прямо з IDE.

2. Підтримка різних фреймворків: PyCharm дозволяє використовувати різні фреймворки для тестування веб-сайтів, такі як PyTest, Robot Framework, Behave та інші.

3. Інтегроване відлагодження: PyCharm надає потужний відлагоджувач, який допомагає виявляти та виправляти помилки в тестовому коді. У IDE можна крокувати по коду, перевіряти значення змінних, встановлювати точки зупину та аналізувати виконання тестів для швидкого виявлення проблем.

4. Управління залежностями: PyCharm має вбудовану систему для управління залежностями Python, таку як pip та віртуальне оточення. Це дозволяє легко встановлювати необхідні пакети для тестування веб-сайтів і забезпечувати чистоту середовища для кожного проекту.

5. Аналіз коду та автодоповнення: PyCharm забезпечує потужний аналіз коду та автодоповнення, що допомагає прискорити процес написання тестів. PyCharm дозволяє отримувати підказки про доступні методи та атрибути веб-елементів, перевіряти синтаксис, виявляти потенційні проблеми та отримувати рекомендації щодо поліпшення коду.

Загалом, PyCharm [3] є потужним інструментом для розробки та тестування веб-сайтів. Він надає широкий набір функціональностей, що полегшують процес написання тестів, виконання автоматизованого тестування та відлагодження тестового коду. Зручне середовище розробки та підтримка різних фреймворків роблять PyCharm відмінним вибором для тестування веб-сайтів з використанням Python.

2.3 Selenium

Selenium [1] – це популярний інструмент для автоматизації веб-браузера. Він надає різні можливості для автоматизації дій, які зазвичай

виконуються веб-користувачем, таких як відкриття веб-сторінок, заповнення форм, виконання кліків та вилучення інформації.

Основними компонентами Selenium є:

1. Selenium WebDriver Це основна частина Selenium, яка надає програмний інтерфейс для взаємодії з браузером. WebDriver підтримує різні браузери, такі як Chrome, Firefox, Safari, Edge та інші. Він надає методи для пошуку елементів на веб-сторінці, взаємодії з елементами, керування вікнами та вкладками браузера, виконання JavaScript коду та багато іншого.

2. Selenium IDE: Це інтегроване середовище розробки, яке дозволяє записувати та відтворювати дії користувача на веб-сторінці. За допомогою Selenium IDE можна швидко створювати автоматизовані тести без написання коду. Однак, Selenium IDE має обмежені можливості в порівнянні з WebDriver і зазвичай використовується для створення простих тестових сценаріїв.

3. Selenium Grid: Це інструмент, який дозволяє розподілити виконання автоматизованих тестів на кілька комп'ютерів чи віртуальних машин. За допомогою Selenium Grid можна одночасно запускати тести на різних браузерах та платформах, що підвищує продуктивність та прискорює процес тестування.

Selenium [1] підтримує різні мови програмування, включаючи Python, Java, C#, Ruby та інші. За допомогою Selenium та мови програмування Python можна створювати потужні та гнучкі автоматизовані тести для веб-додатків. Selenium надає безліч методів та функцій, які полегшують взаємодію з браузером та виконання перевірок. Він також дозволяє інтегруватися з іншими інструментами та фреймворками, такими як PyTest та Robot Framework, розширюючи можливості автоматизації тестування.

Selenium є надійним та широко використовуваним інструментом у співтоваристві тестувальників та розробників, який допомагає автоматизувати тестування веб-додатків, покращуючи ефективність та якість процесу розробки.

2.4 Методи пошуку елементів

В сфері автоматизованого тестування веб-додатків, ефективний пошук елементів є одним з найважливіших аспектів [4]. Завдяки методам пошуку елементів у Selenium, розробники та тестувальники можуть легко знаходити та взаємодіяти з різноманітними елементами на веб-сторінках.

Кожен метод має свої унікальні характеристики та використовується в різних сценаріях тестування. Наприклад, пошук за ID надає швидкий та точний спосіб знаходження елемента, тоді як CSS селектори та XPath дозволяють виконувати складніші запити для знаходження елементів згідно з їхніми атрибутами чи ієрархією.

```
ID = "id"
NAME = "name"
XPATH = "xpath"
LINK_TEXT = "link text"
PARTIAL_LINK_TEXT = "partial link text"
TAG_NAME = "tag name"
CLASS_NAME = "class name"
CSS_SELECTOR = "css selector"
```

```
find_element(By.ID, "id")
find_element(By.NAME, "name")
find_element(By.XPATH, "xpath")
find_element(By.LINK_TEXT, "link text")
find_element(By.PARTIAL_LINK_TEXT, "partial link text")
find_element(By.TAG_NAME, "tag name")
find_element(By.CLASS_NAME, "class name")
find_element(By.CSS_SELECTOR, "css selector")
```

Рис. 2.1 – Методи пошуку елементів

2.4.1 Locating by Id

Одним з найпоширеніших способів знаходження елементів на сторінці є пошук за ідентифікатором (ID) [4].

ID (ідентифікатор) - це унікальний атрибут, який присвоюється елементу HTML на сторінці. Кожен елемент повинен мати унікальний ID, який можна використовувати для точного його визначення.

Для пошуку елементів за ID використовується метод find_element(By.ID, "ідентифікатор"). Ось приклад коду на мові Python, який демонструє пошук елемента за ID:

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
  </form>
 </body>
</html>
```

```
login_form = driver.find_element(By.ID, 'loginForm')
```

Рис. 2.2 – Пошук елементу за ID

У цьому прикладі спочатку створюємо екземпляр веб-драйвера, відкриваємо веб-сторінку та знаходимо елемент за його ID, вказаним у методі find_element. Потім можемо виконати різні дії зі знайденим елементом, наприклад, натиснути на нього.

Важливо відзначити, що якщо на сторінці знаходиться декілька елементів з однаковим ID, метод find_element поверне перший знайдений елемент. Якщо немає елементів з вказаним ID, виникне виключення NoSuchElementException. У цьому разі потрібно бути впевненим, що ID елемента є унікальним на сторінці перед його використанням для пошуку.

2.4.2 Locating by Name

Атрибут "name" використовується для надання унікальної назви елементу на веб-сторінці. Завдяки цьому атрибуту, розробники можуть ідентифікувати елементи та здійснювати з ними взаємодію.

Для пошуку елементів за атрибутом "name" [4] у Selenium можна використовувати метод find_element_by_name() або find_elements_by_name(). Перший метод повертає перший знайдений елемент зі специфікованим атрибутом "name", тоді як другий метод повертає список усіх знайдених елементів зі співпадаючим атрибутом.

Наприклад, для знаходження інпут-поля з атрибутом "name" рівним "email", можна використовувати наступний код:

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
</body>
</html>
```

```
username = driver.find_element(By.NAME, 'username')
password = driver.find_element(By.NAME, 'password')
```

Рис. 2.3 – Пошук елементу за Name

Якщо елемент знайдений, його можна використовувати для подальшої взаємодії, наприклад, для введення тексту в поле або натискання кнопки. Якщо ж елемент не знайдений, виникне виняток NoSuchElementException, який можна обробити для подальшого контролю потоку виконання.

Важливо мати на увазі, що атрибут "name" повинен бути унікальним для кожного елемента на сторінці, щоб уникнути помилок у пошуку. Додатковою перевагою пошуку за атрибутом "name" є швидкість виконання, оскільки цей метод використовує оптимізований пошук внутрішньо у Selenium.

Загалом, пошук елементів за атрибутом "name" у Selenium [1] є простим та ефективним способом ідентифікації елементів на веб-сторінці. Використання цього методу дозволяє забезпечити стабільність та точність автоматизованих тестів, що спрощує розробку та підтримку тестових сценаріїв.

2.4.3 Locating by XPath

Пошук елементів за допомогою XPath є потужним і гнучким методом веб-скрапінгу та автоматизації тестування в Selenium [4]. XPath (XML Path

Language) - це мова запитів для навігації та вибору елементів у XML-документах, включаючи HTML.

XPath дозволяє точно локалізувати елементи на веб-сторінці шляхом використання шаблонів шляху до елементів. Шлях до елемента в XPath може бути вказаний за його розташуванням у дереві DOM, за атрибутами, текстовим змістом тощо. XPath надає різні можливості фільтрації та пошуку, що дозволяє знаходити елементи за різними критеріями.

У Selenium можна використовувати методи find_element_by_xpath() та find_elements_by_xpath() для пошуку елементів за допомогою XPath. Перший метод повертає перший знайдений елемент, який відповідає заданому XPath, тоді як другий метод повертає список усіх знайдених елементів.

Наприклад, для знаходження кнопки з текстом "Вхід", можна використати наступний XPath-шлях:

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
</body>
</html>
```

```
login_form = driver.find_element(By.XPATH, "/html/body/form[1]")
login_form = driver.find_element(By.XPATH, "//form[1]")
login_form = driver.find_element(By.XPATH, "//form[@id='loginForm']")
```

Рис. 2.4 – Пошук елементу за XPath

2.4.4 Locating Hyperlinks by Link Text

Цей метод дозволяє знаходити елементи на веб-сторінці за текстом посилання, який вони містять [4].

У Selenium для пошуку елементів по Hyperlinks by Link Text використовується метод find_element_by_link_text або find_elements_by_link_text, залежно від того, чи потрібно знайти один

елемент чи декілька елементів, відповідно. При використанні цих методів вказується текст посилання, який потрібно знайти.

Наприклад, для пошуку єдиного елемента з гіперпосиланням за текстом "Детальніше" використовується наступний код:

```
<html>
 <body>
  <p>Are you sure you want to do this?</p>
  <a href="continue.html">Continue</a>
  <a href="cancel.html">Cancel</a>
</body>
</html>
```

```
continue_link = driver.find_element(By.LINK_TEXT, 'Continue')
continue_link = driver.find_element(By.PARTIAL_LINK_TEXT, 'Conti')
```

Рис. 2.5 – Пошук елементу за Hyperlinks by Link Text

2.4.5 Locating Elements by Tag Name

Метод `find_elements_by_tag_name()` є одним із способів пошуку елементів на веб-сторінці за тегом за допомогою Selenium. Цей метод дозволяє знайти всі елементи, які відповідають певному тегу HTML [4].

Синтаксис використання методу `find_elements_by_tag_name()` досить простий. Потрібно створити екземпляр об'єкту WebDriver, а потім викликати метод `find_elements_by_tag_name()` на цьому об'єкті, передаючи йому назву тегу, який шукається. Метод повертає список знайдених елементів, які відповідають вказаному тегу.

Однією з переваг використання методу `find_elements_by_tag_name()` є його простота. Можна шукати елементи, базуючись на їх тегу, що є основною характеристикою елементів на сторінці. Це дозволяє легко і швидко здійснювати пошук потрібних елементів.

Однак, варто враховувати, що пошук елементів за тегом може повернути багато результатів, особливо якщо веб-сторінка містить багато елементів з однаковим тегом. В такому випадку важливо точно визначити, які саме елементи потрібні для подальшого використання в тестах.

Використання методу `find_elements_by_tag_name()` у поєднанні з іншими методами пошуку дозволяє більш гнучко і точно локалізувати та взаємодіяти з елементами на веб-сторінці, що допомагає забезпечити надійність тестів та швидкість їх виконання.

```
<html>
 <body>
  <p class="content">Site content goes here.</p>
</body>
</html>
```

```
content = driver.find_element(By.CLASS_NAME, 'content')
```

Рис. 2.6 – Пошук елементу за Elements by Tag Name

2.4.6 Locating Elements by Class Name

Метод `find_elements_by_class_name()` є одним зі способів локалізації елементів на веб-сторінці за допомогою Selenium. Цей метод дозволяє знайти всі елементи, які мають певний клас CSS [4].

Для використання методу `find_elements_by_class_name()` потрібно створити екземпляр об'єкту WebDriver і викликати метод на цьому об'єкті, передаючи йому назву класу, за яким шукаються елементи. Метод повертає список знайдених елементів, які мають вказаний клас.

Один з головних переваг використання методу `find_elements_by_class_name()` полягає в його простоті. Можна шукати елементи, використовуючи їх клас CSS, що є одним з основних способів визначення стилів елементів на веб-сторінці. Це дозволяє легко і швидко знаходити потрібні елементи для подальшого взаємодії з ними.

Однак, важливо пам'ятати, що якщо на сторінці є кілька елементів з однаковим класом, метод `find_elements_by_class_name()` поверне список всіх знайдених елементів. В такому випадку, може знадобитися додаткова логіка для точного вибору того елемента, який шукається.

Використання методу `find_elements_by_class_name()` в поєднанні з іншими методами пошуку дозволяє зручно та ефективно знаходити та

взаємодіяти з елементами на веб-сторінці, що сприяє створенню надійних та швидких тестів.

```
<html>
 <body>
  <p class="content">Site content goes here.</p>
</body>
</html>
```

```
content = driver.find_element(By.CLASS_NAME, 'content')
```

Рис. 2.7 – Пошук елементу за Elements by Class Name

2.4.7 Locating Elements by CSS Selectors

Метод find_elements_by_css_selector() є одним зі способів локалізації елементів на веб-сторінці за допомогою Selenium. Він дозволяє знайти елементи за допомогою CSS-селекторів [4], які є потужним і гнучким засобом ідентифікації елементів на веб-сторінці.

Для використання методу find_elements_by_css_selector() потрібно створити екземпляр об'єкту WebDriver і викликати метод на цьому об'єкті, передаючи йому CSS-селектор, за яким знаходяться елементи. Метод повертає список знайдених елементів, що відповідають CSS-селектору. CSS-селектори - це механізм, що дозволяє вибирати елементи на сторінці за їх структурою, класами, ідентифікаторами, атрибутами та іншими характеристиками. Вони мають сильний синтаксис, який дозволяє точно вибирати потрібні елементи зі сторінки.

Одина з головних переваг використання методу find_elements_by_css_selector() полягає в його гнучкості. Можна точно вибирати елементи, використовуючи CSS-селектори, що дозволяє знаходити бажані елементи за їх унікальними характеристиками. Це дозволяє забезпечити стабільність тестів та швидкість їх виконання.

Ще одна перевага використання CSS-селекторів полягає в тому, що вони є стандартом і широко використовуються веб-розробниками для стилізації та ідентифікації елементів. Це означає, що можна легко

адаптуватися до змін на веб-сторінці, використовуючи CSS-селектори для пошуку елементів.

Однак, важливо пам'ятати, що використання неправильного CSS-селектора може призвести до невдалих пошукових результатів або навіть невірної ідентифікації елементів. Тому потрібно ретельно перевіряти CSS-селектори перед їх використанням і переконатися, що вони точно вибирають потрібні елементи на веб-сторінці.

```
<html>
 <body>
  <p class="content">Site content goes here.</p>
</body>
</html>
```

```
content = driver.find_element(By.CSS_SELECTOR, 'p.content')
```

Рис. 2.8 – Пошук елементу за Elements by CSS Selectors

2.5 Chrome WebDriver

Веб-драйвер Chrome [5] - це програмне забезпечення, яке використовується для автоматизації взаємодії між вашою програмою на мові програмування і браузером Google Chrome. Веб-драйвер Chrome надає можливість керувати веб-браузером, виконувати дії, такі як відкриття сторінок, заповнення форм, натискання кнопок, перевірка вмісту тощо.

Ось кілька важливих рис веб-драйвера Chrome:

1. Підтримка мов програмування: Веб-драйвер Chrome підтримує кілька мов програмування, таких як Python [2], Java, C#, Ruby і багато інших.

2. Контроль браузера: Веб-драйвер Chrome дозволяє керувати різними аспектами браузера, такими як вікна, вкладки, URL-адреси, cookies, налаштування і т.д. Веб-драйвер може відкривати нові сторінки, перемикатися між вкладками, зчитувати та змінювати значення елементів на сторінках.

3. Інспектор елементів: Веб-драйвер Chrome надає доступ до інспектора елементів, що дозволяє отримувати інформацію про структуру та атрибути елементів на сторінках веб-додатка. Це дозволяє знаходити,

взаємодіяти та перевіряти елементи на сторінках під час виконання автоматизованих тестів.

4. Збір результатів: Веб-драйвер Chrome може збирати різні дані про сторінки, такі як HTML-код, сніппети JavaScript, скріншоти тощо. Це дозволяє аналізувати стан сторінки та перевіряти правильність відображення вмісту.

5. Конфігурація: Веб-драйвер Chrome надає можливості конфігурації, такі як налаштування проксі-сервера, управління розміром вікна, режим безпеки тощо. Веб-драйвер можна налаштувати залежно від потреб і вимог тестування.

Щоб використовувати веб-драйвер Chrome у програмі, потрібно встановити веб-драйвер та забезпечити його доступність для програми. Зазвичай це включає встановлення веб-драйвера, вказання шляху до нього у коді та створення екземпляра веб-драйвера для взаємодії з браузером Chrome.

Хром веб-драйвер [5] відіграє важливу роль у тестуванні веб-сайтів. Ось декілька ключових ролей, які він виконує:

1. Автоматизація взаємодії з браузером: Хром веб-драйвер дозволяє автоматизувати взаємодію з браузером Google Chrome. Веб-драйвер може відкривати сторінки, орієнтуватися по сайту, заповнювати форми, натискати кнопки і виконувати різні дії, які зазвичай виконує користувач.

2. Перевірка функціональності та стабільності: Хром веб-драйвер дозволяє автоматизовано перевіряти функціональність веб-сайту. У ньому можна створювати тестові сценарії, що включають послідовність дій користувача та очікувані результати. Це допомагає виявляти помилки, перевіряти правильність відображення вмісту, перевіряти реакцію сайту на різні дії користувача.

3. Перехоплення та аналіз елементів сторінки: Хром веб-драйвер дозволяє отримувати доступ до структури сторінки, її елементів та атрибутів. За допомогою нього можна зчитувати текст, значення полів введення, отримувати інформацію про розміри елементів, виконувати пошук елементів

за ідентифікаторами, класами, CSS-селекторами тощо. Це дозволяє здійснювати перевірки та взаємодію з конкретними елементами сторінки.

4. Збір результатів тестування: Хром веб-драйвер може збирати різні дані про сторінки під час тестування. Наприклад, можна отримувати HTML-код сторінки, скріншоти, логи процесу взаємодії з браузером. Це допомагає аналізувати результати тестування, виявляти проблеми та вдосконалювати якість веб-додатка.

У загальному, Хром веб-драйвер є потужним інструментом для автоматизації тестування веб-сайтів з використанням браузера Google Chrome. Він надає зручний інтерфейс для керування браузером та отримання доступу до елементів сторінки, що дозволяє ефективно тестувати та перевіряти функціональність та стабільність веб-додатків.

РОЗДІЛ 3 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

3.1 PyTest

PyTest [6] є потужним фреймворком для автоматизованого тестування виконання коду на мові Python. Він надає простий і зрозумілий спосіб написання тестів, а також має багато корисних функцій і можливостей, що роблять процес тестування ефективним та продуктивним.

Ось кілька ключових особливостей PyTest:

1. Простий синтаксис: PyTest пропонує простий і зрозумілий синтаксис для написання тестів. У ньому можна використовувати звичайні функції Python зі спеціальними префіксами test_ для позначення тестових випадків.

2. Автоматичне виявлення тестів: PyTest автоматично виявляє тестові випадки, які були написані, без необхідності в явному описі тестових наборів чи використанні спеціальних класів.

3. Розширювання функціональності: PyTest має широкий спектр плагінів, які дозволяють розширити його функціональність. У ньому можна використовувати плагіни для збору звітів, інтеграції з іншими інструментами, генерації звітів покриття коду, налаштування фікстур та багато іншого.

4. Підтримка параметризованих тестів: PyTest надає можливість параметризувати тести, дозволяючи запускати один і той самий тестовий випадок з різними вхідними даними або параметрами.

5. Інтеграція з іншими інструментами: PyTest легко інтегрується з іншими популярними інструментами тестування і розробки, такими як Selenium для автоматизації веб-тестування, Coverage для аналізу покриття коду тестами, Mock для створення замінників та іншими.

6. Підтримка асинхронного тестування: PyTest надає підтримку асинхронного тестування для коду, який використовує асинхронні фреймворки, такі як asyncio, Tornado, або Twisted.

PyTest [6] робить процес написання тестів більш простим, швидким і зручним. Він дозволяє вам швидко створювати розширені тестові набори,

перевіряти очікувані результати, забезпечувати покриття коду тестами і здійснювати інтеграцію з іншими інструментами для створення повноцінного процесу автоматизованого тестування.

3.2 Robot Framework

Robot Framework [7] є гнучким інструментом для автоматизованого тестування і автоматизації процесів розробки. Цей фреймворк надає простий і зрозумілий синтаксис, який дозволяє навчитися використовувати його швидко навіть новим користувачам.

Ось кілька ключових особливостей Robot Framework:

1. Легкість використання: Robot Framework використовує зрозумілий синтаксис, що базується на ключових словах та таблицях. Це дозволяє навіть людям без попереднього досвіду з автоматизованим тестуванням швидко відчути комфорт при використанні фреймворку.

2. Підтримка множини платформ: Robot Framework можна використовувати для тестування різних типів додатків, включаючи веб-додатки, настільні програми, мобільні додатки та API. Це робить його універсальним інструментом для автоматизації тестування.

3. Розширюваність: Фреймворк Robot Framework дозволяє розширювати його функціональність за допомогою власних бібліотек і плагінів. За його допомогою можна створювати свої власні ключові слова та бібліотеки для використання у тестових сценаріях.

4. Підтримка даних в форматі таблиць: Robot Framework використовує таблиці для організації тестових даних. Це дозволяє створювати структуровані тестові сценарії та легко змінювати дані без потреби внесення змін у коді.

5. Підтримка звітності: Robot Framework надає детальні звіти про виконання тестів, включаючи інформацію про успішність, невдачі та покриття коду тестами. Це допомагає зрозуміти результати тестування та знайти проблеми в коді.

Robot Framework [7] є цікавим і гнучким інструментом, який допомагає автоматизувати тестування додатків та забезпечує зручність управління тестовими сценаріями та даними. Він підтримує співпрацю з різними інструментами та має активну спільноту, що робить його одним із найпопулярніших фреймворків для автоматизованого тестування.

РОЗДІЛ 4 АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вибір інструментів для роботи

Після огляду існуючих рішень і розгляду вимог проекту було прийнято рішення не використовувати готові фреймворки для автоматизації тестування, а провести дослідження щодо ефективності різних методів пошуку елементів за допомогою Selenium. Це рішення було прийняте з метою забезпечення більшої гнучкості та контролю над процесом тестування.

Один з ключових аспектів дослідження була ефективність методів пошуку елементів. Використання неправильного методу може призвести до нестабільних тестів, що може затримувати виконання тестового набору або навіть спричинити його невдачу. Тому було важливо визначити оптимальний метод пошуку, який би забезпечив стабільність тестів і мінімізував час виконання.

В процесі дослідження були використані різні методи пошуку елементів за допомогою Selenium, такі як пошук за ідентифікатором, назвою класу, CSS селекторами та іншими атрибутами. Кожен метод був докладно вивчений і проаналізований з точки зору його ефективності і стабільності. Після порівняння результатів було визначено найоптимальніший метод пошуку, який забезпечує найкращу комбінацію стабільності та швидкості виконання тестів.

Враховуючи результати дослідження, був зроблений висновок, що правильний вибір методу пошуку елементів є важливим кроком для успішного автоматизованого тестування. Використання Selenium разом з оптимальним методом пошуку дозволить нам створити стабільні та ефективні тести, що покриють широкий спектр функціональності веб-сайту університету.

При розгляді вибору мови програмування для написання тестів, було вирішено використовувати мову Python. Python має багато переваг, які роблять його ідеальним вибором для написання автоматизованих тестів. Він

має простий і зрозумілий синтаксис, широкий набір бібліотек, включаючи Selenium, що дозволяють легко розробляти і підтримувати тестовий фреймворк.

4.2 Аналіз структури сайту

Після детального аналізу структури та функціональності веб-сайту університету, була створена загальна модель сайту (рис. 4.1), яка допомагає уявити основні компоненти та взаємозв'язки між ними. Ця модель створена з метою полегшити розробку тестових сценаріїв та поліпшити орієнтацію тестування у структурі веб-додатку.

У загальну модель сайту включено основні сторінки, елементи і дії, які є об'єктом автоматизованого тестування. Це пошукове поле, кнопки навігації, таблиці з даними тощо. За допомогою цієї моделі в процесі тестування буде зрозуміло, які елементи сайту є важливими для тестування і як вони взаємодіють між собою. Загальна модель сайту є важливим інструментом при розробці тестових сценаріїв, оскільки вона допомагає уникнути пропусків та забезпечує покриття основних функцій та можливостей веб-додатку. Вона також дозволяє зрозуміти, які частини сайту можуть бути вразливими до змін і вимагають більшої уваги під час регресійного тестування.

Завдяки загальній моделі сайту тестувальники можуть працювати більш систематично, орієнтуючись на визначені компоненти та їх взаємозв'язки. Це сприяє ефективнішому та структурованому процесу автоматизованого тестування, допомагає виявляти проблеми та забезпечує більшу надійність тестових сценаріїв.

Рис. 4.1 – Загальна модель сайту університету

4.3 Алгоритм ПЗ

На основі проведеного аналізу структури веб-сторінки та її компонентів був розроблений ефективний алгоритм пошуку елементів (рис. 4.2). Цей алгоритм дозволяє знаходити та взаємодіяти з потрібними елементами веб-сторінки з використанням Selenium. Він враховує різні методи локаторів, такі як шлях до елемента, ідентифікатор, клас, атрибути або CSS-селектори.

Розроблений алгоритм був успішно інтегрований у тестовий фреймворк, що базується на Selenium. Це дозволяє автоматизовано виконувати дії на веб-сторінці, такі як заповнення полів, клікання на кнопки, навігація тощо. Крім того, фреймворк дозволяє перевіряти правильність виконання цих дій шляхом порівняння фактичних результатів з очікуваними.

Результати пошуку елементів та виконання тестів записуються у звіт, що є важливим аспектом автоматизованого тестування. Звіт допомагає тестувальникам аналізувати результати тестування, виявляти проблеми та помилки, а також здійснювати звітність про виконані дії та стан тестової системи. Завдяки звіту тестувальники можуть швидко знайти та виправити проблеми, що допомагає забезпечити якість тестових сценаріїв та ефективність процесу тестування в цілому.

Застосування розробленого алгоритму пошуку елементів у тестовому фреймворку з використанням Selenium значно полегшує розробку, підтримку та виконання автоматизованих тестів. Це забезпечує стабільність тестів, швидкість їх виконання та полегшує аналіз результатів тестування.

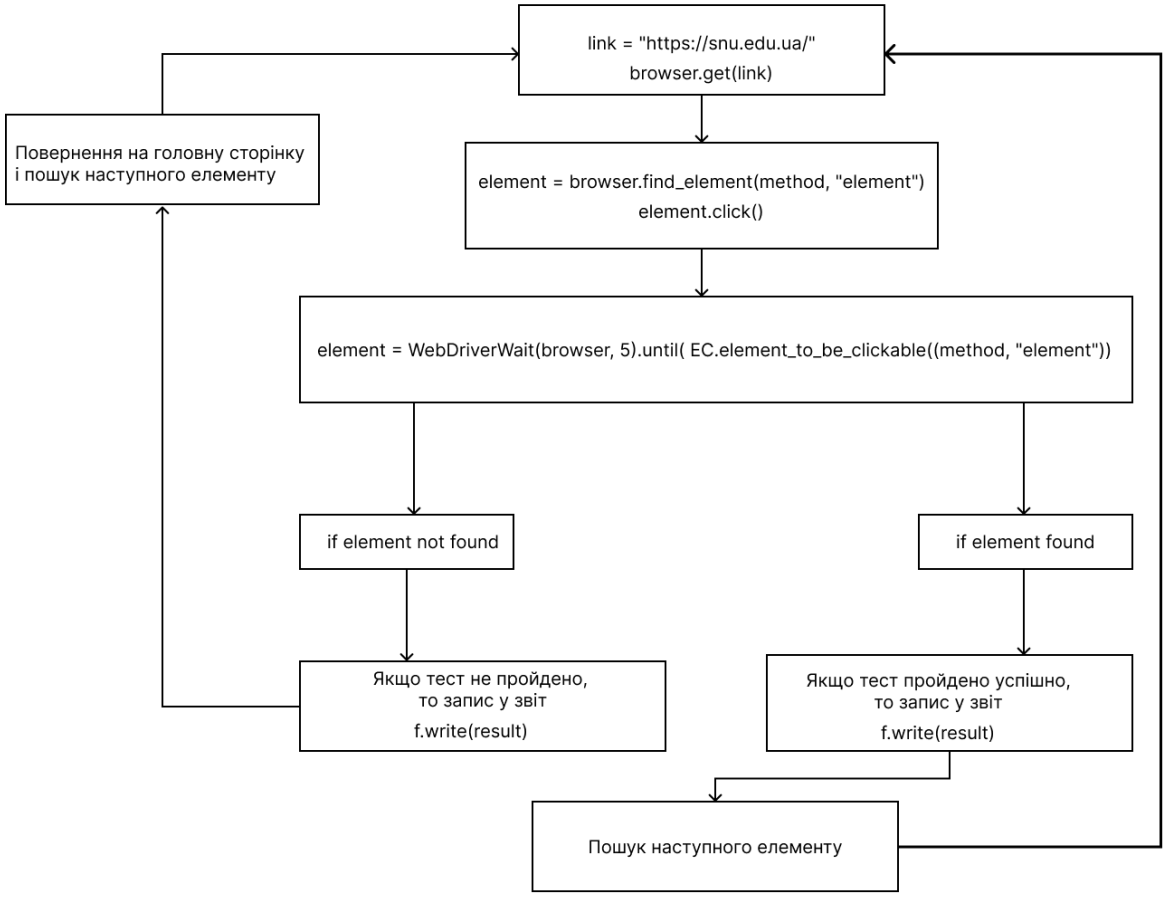

Рис. 4.2 – Алгоритм роботи ПО

Пошук елементів на сайті відбувається за допомогою різних методів та підходів, але працює за одним принципом - знайти та взаємодіяти з потрібними елементами веб-сторінки. Це основна мета автоматизованого тестування, яка дозволяє перевірити функціональність та правильність роботи сайту.

1. Перший етап тестування полягає у відкриванні вікна браузера з сайтом університету в тестовому режимі. Це створює ізольоване середовище для виконання тестів та запобігає впливу зовнішніх факторів на їх результати.

2. Другий етап включає пошук потрібного елемента на веб-сторінці. Для цього можуть використовуватись різні методи, такі як пошук за ідентифікатором, класом, назвою тегу або за допомогою CSS-селекторів.

Важливо знайти унікальні атрибути елемента, які не змінюються при зміні структури сайту.

3. Третій етап виконує дії зі знайденим елементом, наприклад, натискання на кнопку або заповнення поля вводу. Це дозволяє перевірити, як правильно реагує веб-сторінка на взаємодію з користувачем.

4. Четвертий етап передбачає очікування на зворотний зв'язок від елемента. Зазвичай встановлюється обмежений час очікування, у даному випадку 5 секунд. Якщо елемент не відповідає протягом цього часу, виникає виключення, яке записує у звіт про невдалу взаємодію з елементом.

5. П'ятий етап включає перевірку результатів взаємодії з елементом. Якщо елемент відповідає та потрібна сторінка успішно відкривається, результат тесту вважається успішним і записується в звіт. Після цього алгоритм повертає веб-сторінку на головний екран для продовження тестування.

У випадку, коли елемент не відповідає або відкривається зовсім інша сторінка, відповідна помилка записується в звіт, що тест елементу не пройдено. Після цього алгоритм повертає веб-сторінку на головний екран і починає пошук наступного елемента для тестування.

6. Шостий етап включає завершення тестування. Після виконання всіх тестів алгоритм записує в звіт, що тестування завершено, і закриває тестове вікно браузера. Це для того, щоб підсумувати результати тестування та завершити процес автоматизованого тестування. Такий підхід дозволяє зручно аналізувати результати та виявляти проблеми, що допомагає вдосконалювати якість та стабільність веб-сайту університету.

Ці етапи розробки архітектури програмного забезпечення є ключовими для забезпечення ефективної та стабільної роботи програми автоматизації тестування. Детальне дослідження різних методів пошуку елементів та вибір мови програмування відіграють важливу роль у розробці потужного тестового фреймворку та успішному виконанні автоматизованих тестів.

Аналіз різних методів пошуку елементів за допомогою Selenium

дозволяє визначити найефективніші підходи, які забезпечують стабільність тестів та швидкість їх виконання. Це дозволяє зрозуміти, яким чином взаємодіяти з елементами веб-сторінки та забезпечити надійну автоматизацію різноманітних дій.

Вибір мови програмування Python [10] для розробки тестового фреймворку є розумним рішенням з багатьма перевагами. Python має простий та зрозумілий синтаксис, що полегшує розробку та підтримку коду. Він має велику кількість бібліотек, включаючи Selenium, що сприяє швидкому та ефективному написанню тестів. Крім того, Python є популярною мовою програмування в галузі тестування, що сприяє доступності ресурсів, підтримці спільноти та наявності різноманітних інструментів.

У результаті, розробка архітектури програмного забезпечення для автоматизованого тестування сайту університету заснована на глибокому аналізі та дослідженні різних аспектів, таких як методи пошуку елементів та вибір мови програмування [10]. Ці етапи формують основу для успішного розроблення потужного тестового фреймворку, який забезпечує надійне та ефективне виконання автоматизованих тестів.

РОЗДІЛ 5 АПРОБАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Звіт тестування

У процесі розробки алгоритму пошуку елементів для тестування сайту Східноукраїнського національного університету імені Володимира Даля, було розроблено спеціалізоване програмне забезпечення для автоматизованого тестування. Це ПЗ створено з метою спростити та прискорити процес виконання тестових сценаріїв та забезпечити ефективну взаємодію з веб-сторінкою університету.

Розроблене програмне забезпечення дозволяє автоматично відкривати браузер та завантажувати тестовий веб-сайт університету в спеціальному тестовому режимі. Воно використовує розроблений алгоритм пошуку елементів для знаходження та взаємодії з потрібними елементами веб-сторінки. ПЗ забезпечує виконання дій, передбачених у тестових сценаріях, таких як клік на елемент, заповнення форм, навігація між сторінками та перевірка відповідних результатів.

Основною перевагою розробленого програмного забезпечення є його здатність автоматизувати виконання повторюваних тестових сценаріїв, що дозволяє економити час та зусилля тестувальників. Крім того, ПЗ забезпечує стабільність тестів, оскільки воно використовує розроблений алгоритм пошуку елементів, який дозволяє точно локалізувати та взаємодіяти з елементами навіть під час зміни структури веб-сторінки.

TestSNU.py | test_results.txt | test_results.csv

```
Автоматизоване тестування сайту Східноукраїнського національного університету імені Володимира Даля.

1. Відображення сайту: Тест пройдено успішно!

2. Вкладка HELP VDEUNU: Тест пройдено успішно!

3. Кнопка Домівка: Тест пройдено успішно!

4. Тест меню E-університет:
  1) Портал електронного навчання: Тест пройдено успішно!
  2) Портал приймальної комісії: Тест пройдено успішно!
  3) Наукові видання: Тест пройдено успішно!

5. Тест меню Розклад занять:
  1) Семестровий розклад: Тест пройдено успішно!
  2) Розклад занять аспірантів: Тест пройдено успішно!

6. Тест меню СНУ - Wiki: Тест пройдено успішно!

7. Тест меню МОН України: Тест пройдено успішно!

8. Тест меню мови:
 1) English: Тест пройдено успішно!
 2) Українська: Тест пройдено успішно!

9. Тест меню Університет:
  1) Розділ Про Університет: Тест пройдено успішно!
  2) Розділ Інформаційний пакет СНУ ім. В. Даля: Тест пройдено успішно!
  3) Розділ Стратегія розвитку СНУ ім. В. Даля: Тест пройдено успішно!
  4) Розділ Прозорість та інформаційна відкритість університету: Тест пройдено успішно!
  5) Розділ Нормативно-правова документація: Тест пройдено успішно!
  6) Розділ Якість освіти: Тест пройдено успішно!
```

Рис. 5.1 – Звіт тестування веб-сайту університету

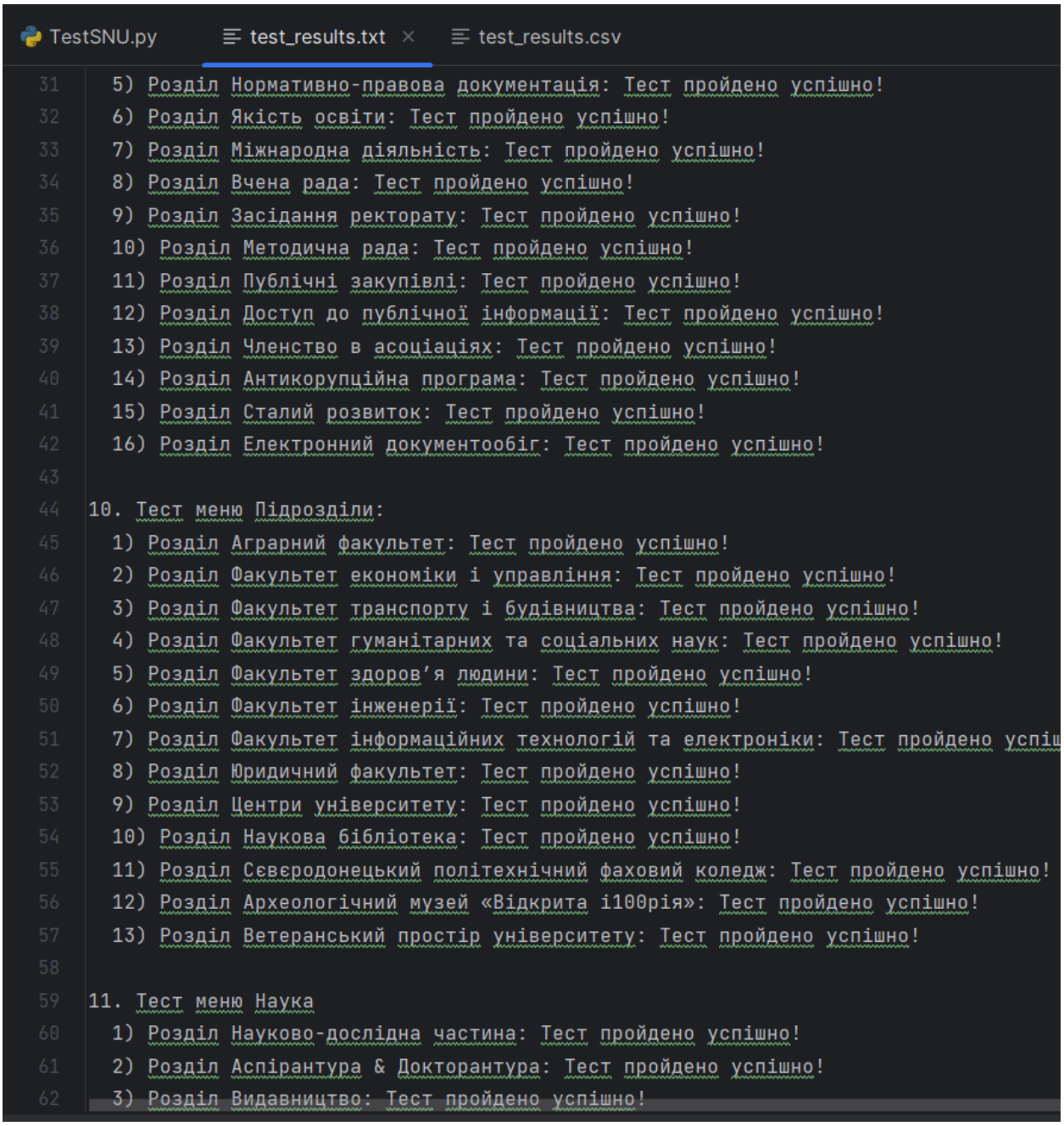

Рис. 5.2 – Звіт тестування веб-сайту університету

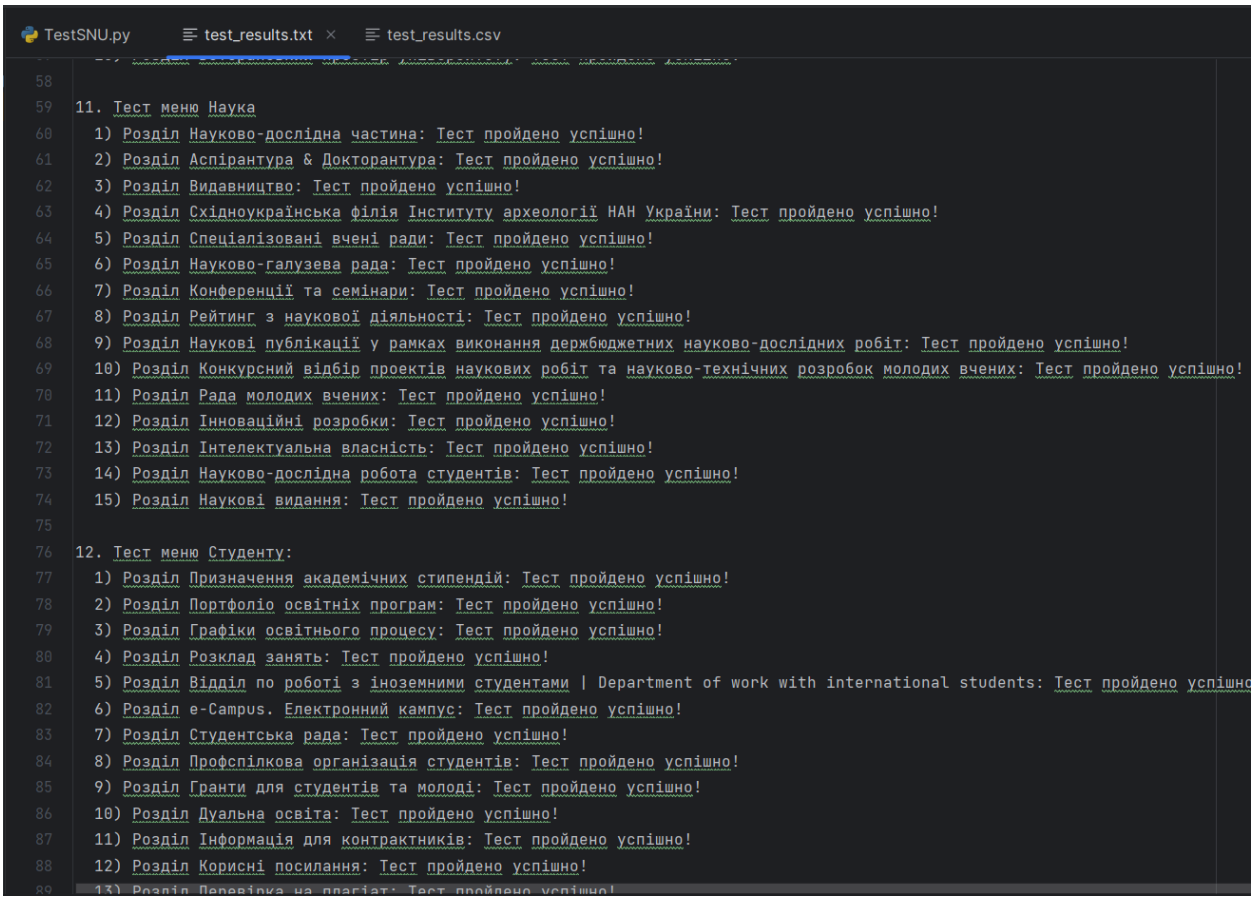

Рис. 5.3 – Звіт тестування веб-сайту університету

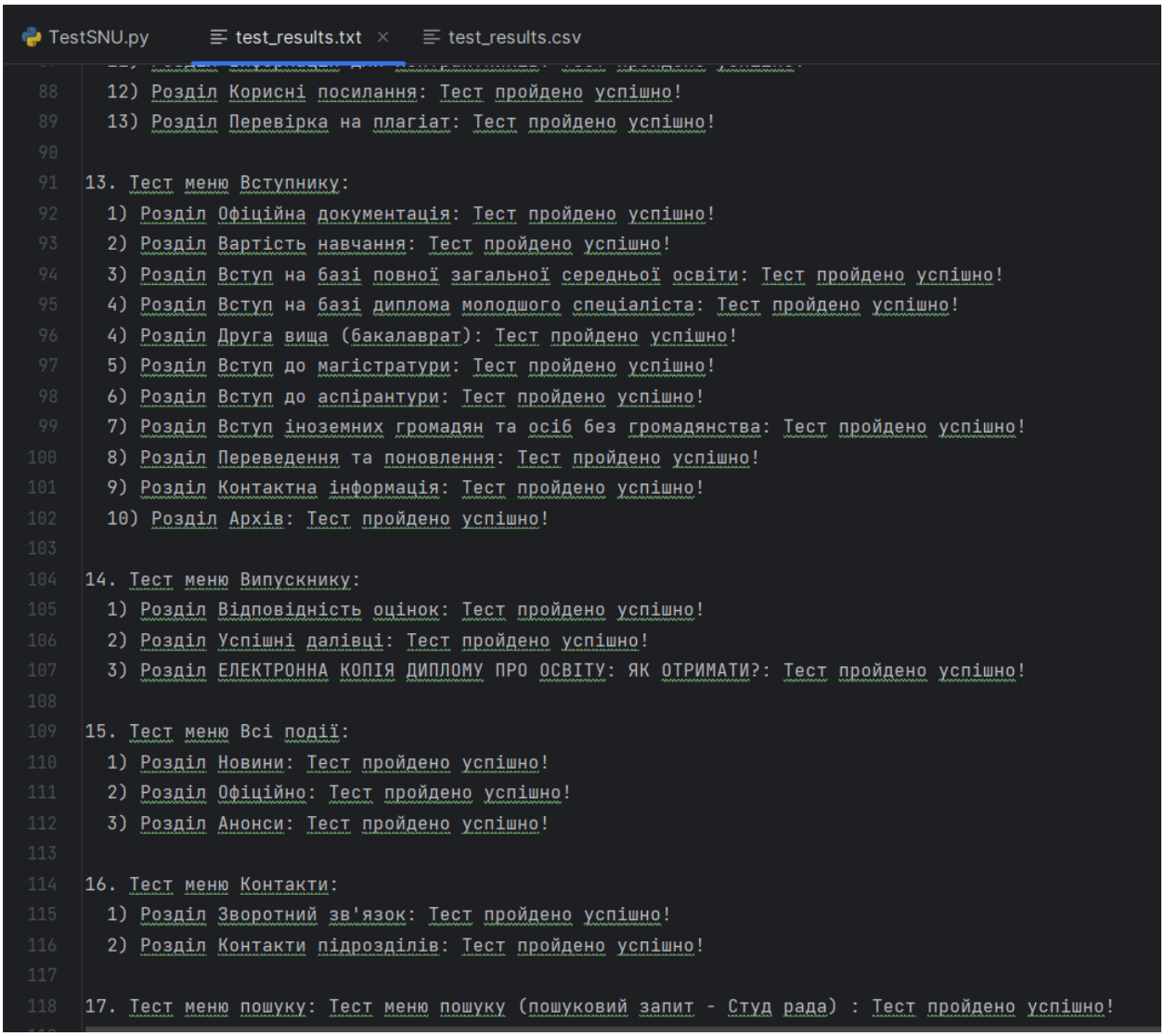

Рис. 5.4 – Звіт тестування веб-сайту університету

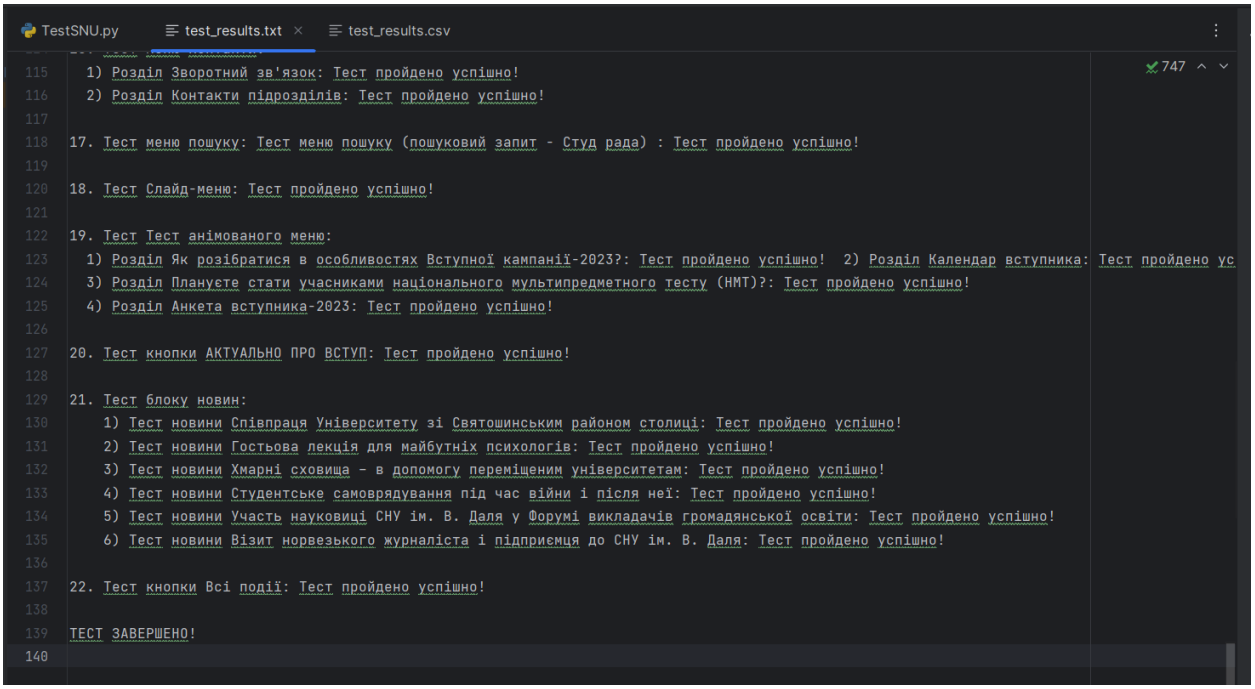

Рис. 5.5 – Звіт тестування веб-сайту університету

Результати виконання тестових сценаріїв зберігаються у зручному для аналізу форматі, наприклад, у текстовому файлі(рис. 5.1, рис. 5.2, рис. 5.3, рис. 5.4, рис. 5.5), який може бути легко перетворений у інші формати, такі як csv, xlsx або xml. Це спрощує процес аналізу результатів тестування та сприяє виявленню потенційних проблем у функціональності веб-сторінки.

Враховуючи вищезазначені переваги, розроблене програмне забезпечення для тестування становить важливий крок у забезпеченні якості та надійності сайту Східноукраїнського національного університету імені Володимира Даля. Воно дозволяє ефективно виконувати автоматизовані тести, полегшує процес розробки та підтримки тестового фреймворку і сприяє впровадженню надійних програмних рішень.

5.2 Робота ПЗ для тестування

На рисунках (рис. 5.6, рис. 5.7, рис. 5.8) ілюструється процес роботи програмного забезпечення для тестування сайту університету. Ці рисунки детально показують послідовність дій та взаємодію ПЗ з веб-сторінкою університету.

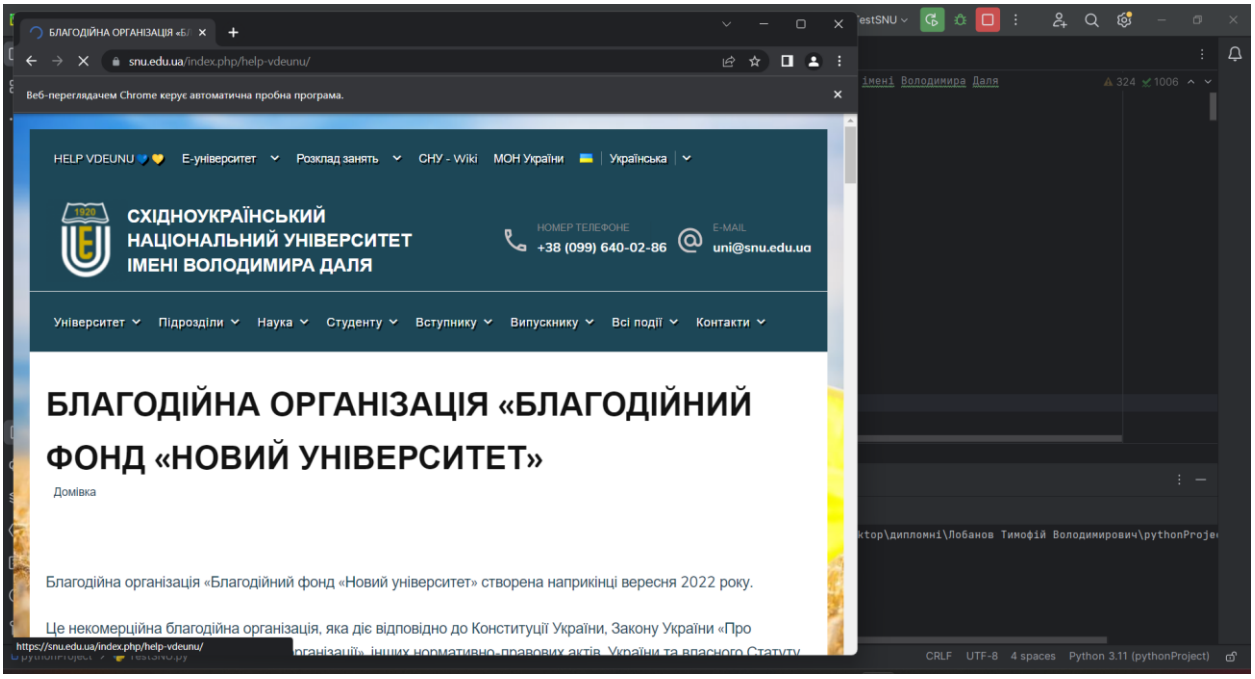

Рис. 5.6 – Робота програмного забезпечення

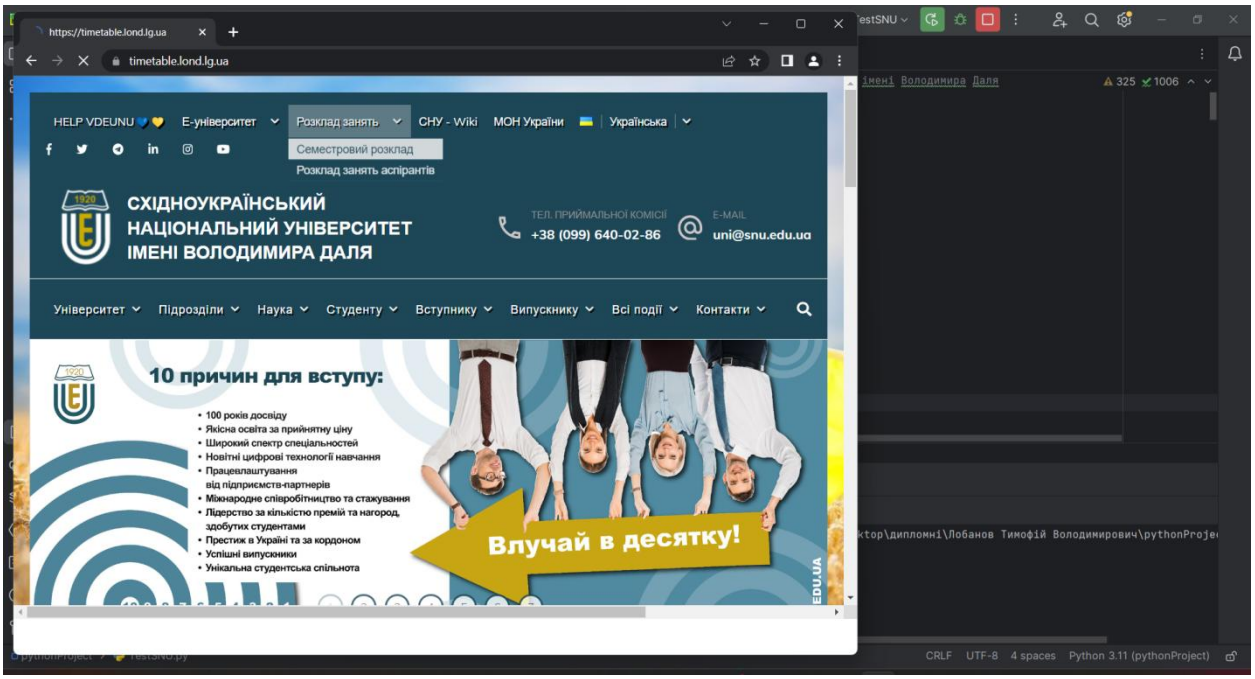

Рис. 5.7 – Робота програмного забезпечення

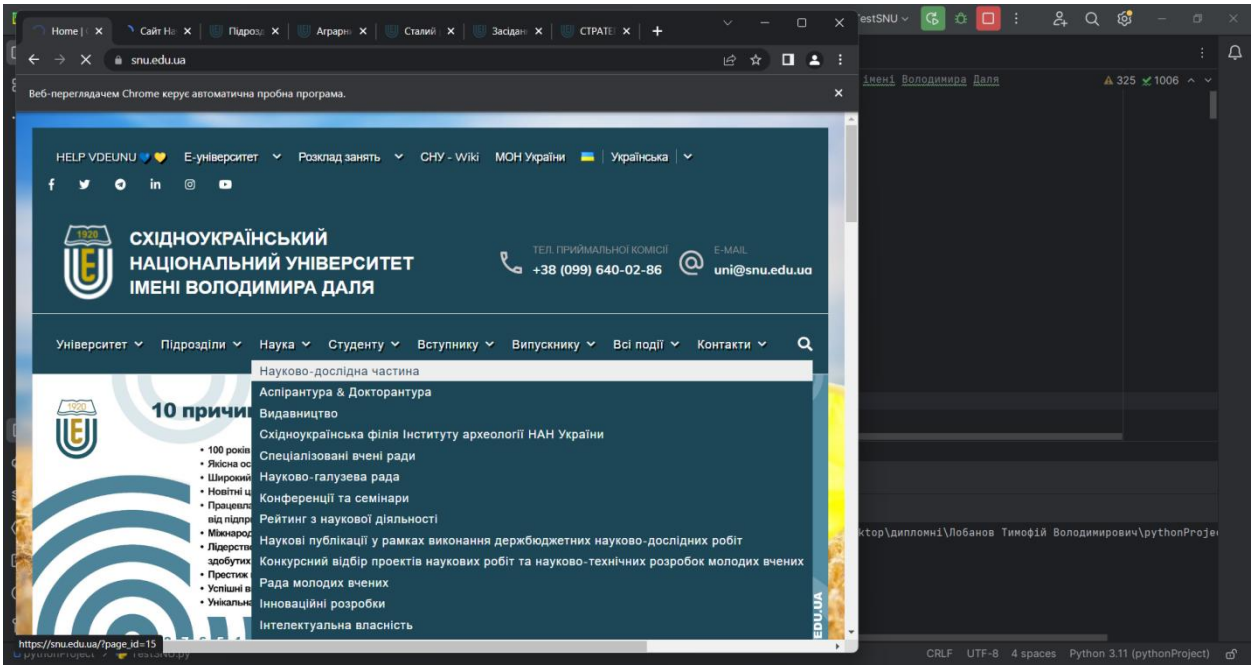

Рис. 5.8 – Робота програмного забезпечення

Запуск програмного забезпечення тестування сайту університету здійснюється безпосередньо у розробничому середовищі (IDE). Це рішення прийняте для забезпечення зручності та гнучкості у випадку, коли на сайті університету регулярно відбуваються оновлення та зміни. Завдяки використанню IDE, можна легко додавати нові елементи до алгоритму пошуку та взаємодії з веб-сторінкою, що дозволяє підтримувати тестове ПЗ актуальним і сумісним з оновленим сайтом університету.

Алгоритм роботи програмного забезпечення тестування сайту університету виконується крок за кроком, розпочинаючи з верхніх елементів і просуваючись до нижніх. Кожен крок алгоритму включає в себе послідовність дій, необхідних для знаходження, взаємодії та перевірки кожного елементу на веб-сторінці. Такий підхід забезпечує систематичне тестування всіх необхідних елементів та забезпечує повноту охоплення функціональності сайту університету під час виконання тестових сценаріїв.

Результатом такої структурованої та послідовної роботи програмного забезпечення тестування є ефективне та надійне виконання автоматизованих тестів на сайті університету.

ВИСНОВОК

Розробка системи автоматизованого тестування веб-сайту університету виявилась важливим кроком у забезпеченні якості та стабільності програмного забезпечення. Кваліфікаційна робота бакалавру була спрямований на перевірку функціональності, стабільності та надійності сайту, що має велике значення для задоволення користувачів та впевненості в його безперебійній роботі.

Під час проєкту було проведено аналіз предметної області, визначено функціональні вимоги сайту, досліджено навігацію та структуру, аналізовано особливості введення та виведення даних. На основі цього аналізу були розроблені тестові сценарії, реалізовано алгоритм пошуку та локалізації елементів на сторінці сайту.

В процесі розробки системи було досягнуто поставлені цілі, такі як автоматизація виконання тестових сценаріїв, забезпечення стабільності тестів та швидкості їх виконання, перевірка коректності введення даних на сайті. Результатом роботи є система, яка забезпечує високу якість програмного забезпечення сайту університету та покращений користувацький досвід.

Проєкт з тестування веб-сайту університету є успішним, оскільки він виконує свою основну мету - перевірку функціональності, стабільності та надійності сайту. Розроблена система автоматизованого тестування дозволяє ефективно виявляти можливі проблеми та дефекти в роботі сайту, що сприяє підвищенню якості та надійності його функціонування.

Завершений проєкт становить важливий крок у процесі розробки та підтримки веб-додатку університету. Він надає розробникам та тестувальникам ефективний інструмент для виявлення та усунення помилок,

а також забезпечує впевненість у стабільній та надійній роботі сайту. Завдяки цьому проєкту університет може продовжувати свою діяльність з впевненістю у функціональності та якості свого веб-сайту, задовольняючи потреби користувачів та забезпечуючи їм зручний досвід взаємодії з ним.

СПИСОК ЛІТЕРАТУРИ

1. Package Search Official Website Selenium — 2023 — URL: https://www.selenium.dev/
2. Package Search Official Website PYTHON — 2023 — URL: https://www.python.org/
3. Package Search Official Website PyCharm — 2023 — URL: https://www.jetbrains.com/ru-ru/pycharm/
4. Selenium guide— 2018 — URL: https://selenium-python.readthedocs.io/locating-elements.html#locating-by-name
5. Package Search Official Website Chrome WebDriver — 2020 — URL: https://chromedriver.chromium.org/downloads
6. Pytest: helps you write better programs — 2022 — URL: https://docs.pytest.org/en/7.3.x/
7. Package Search Official Website Robot Framework — 2022 — URL: https://robotframework.org/
8. The Art of Software Testing. Glenford J. Myers, Corey Sandler, and Tom Bad, 2011.c.54 — 175.
9. Testing Computer Software. Cem Kaner, Jack Falk, and Hung Q. Nguyen.gett.,1999.c.73 — 105.
10. Python Testing Cookbook. Greg L. Turnquist.2011.c.114— 187.

ДОДАТОК А

```
# Автоматизоване тестування сайту Східноукраїнського національного університету імені Володимира Даля
# Автор: Лобанов Тимофій Володимирович. Група ІПЗ-19д
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time

# Відкриття сайту СНУ ім. Володимира Даля
link = "https://snu.edu.ua/"

options = webdriver.ChromeOptions()
options.add_argument("--start-maximized")
browser = webdriver.Chrome(options=options)

browser.get(link)

# Очікуємо, поки сторінка завантажиться
browser.implicitly_wait(5)

with open("test_results.txt", "w") as f:
    f.write("Автоматизоване тестування сайту Східноукраїнського національного університету імені Володимира Даля.\n")

if browser.current_url == link:
    with open("test_results.txt", "a") as f:
        f.write("\n1. Відображення сайту: Тест пройдено успішно!\n")
```

```
else:
    with open("test_results.txt", "a") as f:
        f.write("\n1. Відображення сайту: Тест не пройдено!\n")

try:
    search_help_link = WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.LINK_TEXT, 'HELP VDEUNU□□'))
    )
    search_help_link.click()

    # Очікуємо, поки сторінка завантажиться
    browser.implicitly_wait(5)

    with open("test_results.txt", "a") as f:
        f.write("\n2. Вкладка HELP VDEUNU: Тест пройдено успішно!\n")
except NoSuchElementException:
    with open("test_results.txt", "a") as f:
        f.write("\n2. Вкладка HELP VDEUNU: Тест не пройдено!\n")
    browser.get(link)

dia_home_button = None
try:
    dia_home_button = WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.CSS_SELECTOR,
"a[class*=home_crumb]"))
    )
    dia_home_button.click()
    with open("test_results.txt", "a") as f:
        f.write("\n3. Кнопка Домівка: Тест пройдено успішно!\n")
except NoSuchElementException:
```

```
    with open("test_results.txt", "a") as f:
        f.write("\n3. Кнопка Домівка: Тест не пройдено!\n")

# Відкриття меню "Е-університет"
# Перевірка підвкладки "Портал електронного навчання" у вкладці "Е-університет"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

with open("test_results.txt", "a") as f:
    f.write("\n4. Тест меню Е-університет:\n")

try:
    euniver_link = WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.LINK_TEXT, "Е-університет"))
    )
    euniver_link.click()

    port_link = WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.LINK_TEXT, "Портал електронного навчання"))
    )
    port_link.click()

    write_result("  1) Портал електронного навчання: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  1) Портал електронного навчання: Тест не пройдено!")

browser.get(link)
```

```
# Перевірка підвкладки "Портал приймальної комісії" у вкладці "Е-університет"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

euniver_link = WebDriverWait(browser, 5).until(
    EC.presence_of_element_located((By.LINK_TEXT, "Е-університет"))
)
euniver_link.click()

try:
    portcom_link = WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.LINK_TEXT, "Портал приймальної комісії"))
    )
    portcom_link.click()
    write_result("  2) Портал приймальної комісії: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  2) Портал приймальної комісії: Тест не пройдено!")

browser.get(link)

# Перевірка підвкладки "Наукові видання" у вкладці "Е-університет"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

euniver_link = browser.find_element(By.LINK_TEXT, "Е-університет")
```

```
euniver_link.click()

try:
    science_link = WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.LINK_TEXT, "Наукові видання"))
    )
    science_link.click()
    write_result("  3) Наукові видання: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  3) Наукові видання: Тест не пройдено!")

browser.get(link)

# Перевірка вкладки Розклад занять
# Перехід до е-розкладу
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Тест меню "Розклад занять"
rozklad_link = browser.find_element(By.LINK_TEXT, "Розклад занять")
rozklad_link.click()
try:
    srozklad_link = WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.LINK_TEXT, "Семестровий
розклад"))
    )
    srozklad_link.click()
    write_result("\n5. Тест меню Розклад занять:\n  1) Семестровий розклад:
Тест пройдено успішно!")
```

```
except NoSuchElementException:
    write_result("\n5. Тест меню Розклад занять:\n  1) Семестровий розклад: Тест не пройдено!")

browser.get(link)

# Тест меню "Розклад занять аспірантів"
rozklad_link = browser.find_element(By.LINK_TEXT, "Розклад занять")
rozklad_link.click()
try:
    asproz_link = WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.LINK_TEXT, "Розклад занять аспірантів"))
    )
    asproz_link.click()
    write_result("  2) Розклад занять аспірантів: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  2) Розклад занять аспірантів: Тест не пройдено!")

browser.get(link)


# SNU-WIKI
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result)

snuwiki_link = browser.find_element(By.LINK_TEXT, "СНУ - Wiki")
snuwiki_link.click()
try:
```

```
    WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.TAG_NAME, "h1"))
    )
    write_result("\n6. Тест меню СНУ - Wiki: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("\n6. Тест меню СНУ - Wiki: Тест не пройдено!")

browser.get(link)

# MON Ukraine
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write("\n" + result)

mon_link = browser.find_element(By.LINK_TEXT, "МОН України")
mon_link.click()
try:
    WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.TAG_NAME, "h1"))
    )
    write_result("\n7. Тест меню МОН України: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("\n7. Тест меню МОН України: Тест не пройдено!")

browser.get(link)

# Languages
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result)
```

```
# Українська
lang_link = browser.find_element(By.LINK_TEXT, "Українська")
lang_link.click()
WebDriverWait(browser, 10).until(
    EC.presence_of_element_located((By.LINK_TEXT, "English"))
)
# English
lang_link = browser.find_element(By.LINK_TEXT, "English")
lang_link.click()
WebDriverWait(browser, 10).until(
    EC.presence_of_element_located((By.LINK_TEXT, "Українська"))
)
try:
    write_result("\n \n8. Тест меню мови:\n 1) English: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("\n \n8. Тест меню мови:\n 1) English: Тест не пройдено!")

# English
lang_link = browser.find_element(By.LINK_TEXT, "English")
lang_link.click()
WebDriverWait(browser, 10).until(
    EC.presence_of_element_located((By.LINK_TEXT, "Українська"))
)
# Українська
lang_link = browser.find_element(By.LINK_TEXT, "Українська")
lang_link.click()
try:
    write_result("\n 2) Українська: Тест пройдено успішно!")
except NoSuchElementException:
```

```
    write_result("\n 2) Українська: Тест не пройдено!")

##########################################################################
###########
# Меню "Університет"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()
WebDriverWait(browser, 5).until(
    EC.presence_of_element_located((By.LINK_TEXT, "Про університет"))
)

with open("test_results.txt", "a") as f:
    f.write("\n \n9. Тест меню Університет: \n")

pro_uni_link = browser.find_element(By.LINK_TEXT, "Про університет")
pro_uni_link.click()

try:
    write_result("  1) Розділ Про Університет: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  1) Розділ Про Університет: Тест не пройдено!")

browser.get(link)

def write_result(result):
    with open("test_results.txt", "a") as f:
```

```
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

info_pack_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Інформаційний пакет СНУ ім. В. Даля"))
)
info_pack_link.click()

try:
    write_result("  2) Розділ Інформаційний пакет СНУ ім. В. Даля: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  2) Розділ Інформаційний пакет СНУ ім. В. Даля: Тест не пройдено!")

browser.get(link)

# Розділ "Стратегія розвитку СНУ ім. В. Даля"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

strategy_link = WebDriverWait(browser, 5).until(
```

```
    EC.element_to_be_clickable((By.LINK_TEXT, "Стратегія розвитку СНУ ім. В. Даля"))
)
strategy_link.click()

try:
    write_result("  3) Розділ Стратегія розвитку СНУ ім. В. Даля: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  3) Розділ Стратегія розвитку СНУ ім. В. Даля: Тест не пройдено!")

browser.get(link)
browser.execute_script("window.close();")
browser.switch_to.window(browser.window_handles[0])

# Розділ "Прозорість та інформаційна відкритість університету"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

prozorist_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Прозорість та інформаційна відкритість університету"))
)
prozorist_link.click()
```

```
try:
    write_result("  4) Розділ Прозорість та інформаційна відкритість
університету: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  4) Розділ Прозорість та інформаційна відкритість
університету: Тест не пройдено!")

browser.get(link)
# Розділ "Нормативно-правова документація"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

normat_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Нормативно-правова
документація"))
)
normat_link.click()

try:
    write_result("  5) Розділ Нормативно-правова документація: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  5) Розділ Нормативно-правова документація: Тест не
пройдено!")

browser.get(link)
```

```
# Розділ "Якість освіти"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

jakist_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Якість освіти"))
)
jakist_link.click()

try:
    write_result("  6) Розділ Якість освіти: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  6) Розділ Якість освіти: Тест не пройдено!")

browser.get(link)

# Розділ "Міжнародна діяльність"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

mij_dial_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Міжнародна діяльність"))
```

```
)
mij_dial_link.click()

try:
    write_result("  7) Розділ Міжнародна діяльність: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  7) Розділ Міжнародна діяльність: Тест не пройдено!")

browser.get(link)
# Розділ "Вчена рада"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

rada_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Вчена рада"))
)
rada_link.click()

try:
    write_result("  8) Розділ Вчена рада: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  8) Розділ Вчена рада: Тест не пройдено!")

browser.get(link)
# Розділ "Засідання ректорату"
def write_result(result):
```

```
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

rektorat_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Засідання ректорату"))
)
rektorat_link.click()

try:
    write_result("  9) Розділ Засідання ректорату: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  9) Розділ Засідання ректорату: Тест не пройдено!")

browser.get(link)
browser.execute_script("window.close();")
browser.switch_to.window(browser.window_handles[0])

# Розділ "Методична рада"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

metod_rada_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Методична рада"))
```

```
)
metod_rada_link.click()

try:
    write_result("  10) Розділ Методична рада: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  10) Розділ Методична рада: Тест не пройдено!")

browser.get(link)
# Розділ "Публічні закупівлі"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

public_buy_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Публічні закупівлі"))
)
public_buy_link.click()

try:
    write_result("  11) Розділ Публічні закупівлі: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  11) Розділ Публічні закупівлі: Тест не пройдено!")

browser.get(link)
# Розділ "Доступ до публічної інформації"
def write_result(result):
```

```
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

public_info_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Доступ до публічної інформації"))
)
public_info_link.click()

try:
    write_result("  12) Розділ Доступ до публічної інформації: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  12) Розділ Доступ до публічної інформації: Тест не пройдено!")

browser.get(link)
# Розділ "Членство в асоціаціях"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

join_association_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Членство в асоціаціях"))
```

```
)
join_association_link.click()

try:
    write_result("  13) Розділ Членство в асоціаціях: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  13) Розділ Членство в асоціаціях: Тест не пройдено!")

browser.get(link)
# Розділ "Антикорупційна програма"

def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

anti_corruption_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Антикорупційна програма"))
)
anti_corruption_link.click()

try:
    write_result("  14) Розділ Антикорупційна програма: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  14) Розділ Антикорупційна програма: Тест не пройдено!")

browser.get(link)
```

```
# Розділ "Сталий розвиток"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

upgrade_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Сталий розвиток"))
)
upgrade_link.click()

try:
    write_result("  15) Розділ Сталий розвиток: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  15) Розділ Сталий розвиток: Тест не пройдено!")

browser.get(link)
# Розділ "Електронний документообіг"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_uni_link = browser.find_element(By.LINK_TEXT, "Університет")
men_uni_link.click()

el_doc_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Електронний
документообіг"))
```

```
)
el_doc_link.click()

try:
    write_result("  16) Розділ Електронний документообіг: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  16) Розділ Електронний документообіг: Тест не пройдено!")

browser.get(link)

# Меню "Підрозділи"
men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()
with open("test_results.txt", "a") as f:
    f.write("\n10. Тест меню Підрозділи:\n")
    # Розділ "Аграрний факультет"

def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

agrfakult_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Аграрний факультет"))
)
agrfakult_link.click()
```

```
try:
    write_result("  1) Розділ Аграрний факультет: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  1) Розділ Аграрний факультет: Тест не пройдено!")

browser.get(link)
# Розділ "Факультет економіки і управління"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

economfakult_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Факультет економіки і
управління"))
)
economfakult_link.click()

try:
    write_result("  2) Розділ Факультет економіки і управління: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  2) Розділ Факультет економіки і управління: Тест не
пройдено!")

browser.get(link)
# Розділ "Факультет транспорту і будівництва"
def write_result(result):
```

```
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

transportfakult_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Факультет транспорту і будівництва"))
)
transportfakult_link.click()

try:
    write_result("  3) Розділ Факультет транспорту і будівництва: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  3) Розділ Факультет транспорту і будівництва: Тест не пройдено!")

browser.get(link)

# Розділ "Факультет гуманітарних та соціальних наук"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

gumanfakult_link = WebDriverWait(browser, 5).until(
```

```
    EC.element_to_be_clickable((By.LINK_TEXT, "Факультет гуманітарних та соціальних наук"))
)
gumanfakult_link.click()

try:
    write_result("  4) Розділ Факультет гуманітарних та соціальних наук: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  4) Розділ Факультет гуманітарних та соціальних наук: Тест не пройдено!")

browser.get(link)

# Розділ "Факультет здоров’я людини"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

zdorovfakult_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Факультет здоров’я людини"))
)
zdorovfakult_link.click()

try:
```

```
    write_result("  5) Розділ Факультет здоров’я людини: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  5) Розділ Факультет здоров’я людини: Тест не пройдено!")

browser.get(link)

# Розділ "Факультет інженерії"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

ingenerfakult_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Факультет інженерії"))
)
ingenerfakult_link.click()

try:
    write_result("  6) Розділ Факультет інженерії: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  6) Розділ Факультет інженерії: Тест не пройдено!")

browser.get(link)

# Розділ "Факультет інформаційних технологій та електроніки"
def write_result(result):
    with open("test_results.txt", "a") as f:
```

```
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

informationfakult_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Факультет інформаційних
технологій та електроніки"))
)
informationfakult_link.click()

try:
    write_result("  7) Розділ Факультет інформаційних технологій та
електроніки: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  7) Розділ Факультет інформаційних технологій та
електроніки: Тест не пройдено!")

browser.get(link)

# Розділ "Юридичний факультет"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

yridfakult_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Юридичний факультет"))
```

```
)
yridfakult_link.click()

try:
    write_result("  8) Розділ Юридичний факультет: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  8) Розділ Юридичний факультет: Тест не пройдено!")

browser.get(link)

# Розділ "Центри університету"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

center_uni_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Центри університету"))
)
center_uni_link.click()

try:
    write_result("  9) Розділ Центри університету: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  9) Розділ Центри університету: Тест не пройдено!")

browser.get(link)
```

```
# Розділ "Наукова бібліотека"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

science_biblio_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Наукова бібліотека"))
)
science_biblio_link.click()

try:
    write_result("  10) Розділ Наукова бібліотека: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  10) Розділ Наукова бібліотека: Тест не пройдено!")

browser.get(link)

# Розділ "Сєвєродонецький політехнічний фаховий коледж"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

poli_coledge_link = WebDriverWait(browser, 5).until(
```

```
    EC.element_to_be_clickable((By.LINK_TEXT, "Сєвєродонецький
політехнічний фаховий коледж"))
)
poli_coledge_link.click()

try:
    write_result("  11) Розділ Сєвєродонецький політехнічний фаховий коледж:
Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  11) Розділ Сєвєродонецький політехнічний фаховий коледж:
Тест не пройдено!")

browser.get(link)

# Розділ "Археологічний музей «Відкрита і100рія»"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

museul_i100ria_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Археологічний музей
«Відкрита і100рія»"))
)
museul_i100ria_link.click()

try:
```

```
    write_result("  12) Розділ Археологічний музей «Відкрита і100рія»: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  12) Розділ Археологічний музей «Відкрита і100рія»: Тест не пройдено!")

browser.get(link)

# Розділ "Ветеранський простір університету"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_pidrozdil_link = browser.find_element(By.LINK_TEXT, "Підрозділи")
men_pidrozdil_link.click()

veterans_area_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Ветеранський простір університету"))
)
veterans_area_link.click()

try:
    write_result("  13) Розділ Ветеранський простір університету: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  13) Розділ Ветеранський простір університету: Тест не пройдено!")

browser.get(link)
```

```
# Меню "Наука"
men_science_link = browser.find_element(By.LINK_TEXT, "Наука")
men_science_link.click()
with open("test_results.txt", "a") as f:
    f.write("\n11. Тест меню Наука")

# Розділ "Науково-дослідна частина"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука")
men_science_link.click()

science_explore_link = browser.find_element(By.LINK_TEXT, "Науково-
дослідна частина")
science_explore_link.click()

try:
    write_result("\n  1) Розділ Науково-дослідна частина: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("\n  1) Розділ Науково-дослідна частина: Тест не пройдено!")

browser.get(link)

# Розділ "Аспірантура & Докторантура"
def write_result(result):
    with open("test_results.txt", "a") as f:
```

```
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

science_explore_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Аспірантура &
Докторантура"))
)
science_explore_link.click()

try:
    write_result("  2) Розділ Аспірантура & Докторантура: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  2) Розділ Аспірантура & Докторантура: Тест не пройдено!")

browser.get(link)

# Розділ "Видавництво"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

publishing_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Видавництво"))
)
publishing_link.click()
```

```
try:
    write_result("  3) Розділ Видавництво: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  3) Розділ Видавництво: Тест не пройдено!")

browser.get(link)

# Розділ "Східноукраїнська філія Інституту археології НАН України"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

inst_archaeology = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Східноукраїнська філія
Інституту археології НАН України"))
)
inst_archaeology.click()

try:
    write_result("  4) Розділ Східноукраїнська філія Інституту археології НАН
України: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  4) Розділ Східноукраїнська філія Інституту археології НАН
України: Тест не пройдено!")

browser.get(link)

# Розділ "Спеціалізовані вчені ради"
```

```
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

special_vche_rada = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Спеціалізовані вчені ради"))
)
special_vche_rada.click()

try:
    write_result("  5) Розділ Спеціалізовані вчені ради: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  5) Розділ Спеціалізовані вчені ради: Тест не пройдено!")

browser.get(link)

# Розділ "Науково-галузева рада"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

science_rada = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Науково-галузева рада"))
)
science_rada.click()
```

```
try:
    write_result("  6) Розділ Науково-галузева рада: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  6) Розділ Науково-галузева рада: Тест не пройдено!")

browser.get(link)

# Розділ "Конференції та семінари"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

conference_seminar = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Конференції та семінари"))
)
conference_seminar.click()

try:
    write_result("  7) Розділ Конференції та семінари: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  7) Розділ Конференції та семінари: Тест не пройдено!")

browser.get(link)

# Розділ "Рейтинг з наукової діяльності"
def write_result(result):
    with open("test_results.txt", "a") as f:
```

```
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

score_science_dial = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Рейтинг з наукової
діяльності"))
)
score_science_dial.click()

try:
    write_result("  8) Розділ Рейтинг з наукової діяльності: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  8) Розділ Рейтинг з наукової діяльності: Тест не пройдено!")

browser.get(link)

# Розділ "Наукові публікації у рамках виконання держбюджетних науково-
дослідних робіт"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

budget_science_publish = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Наукові публікації у рамках
виконання держбюджетних науково-дослідних робіт"))
)
```

```
budget_science_publish.click()

try:
    write_result("  9) Розділ Наукові публікації у рамках виконання держбюджетних науково-дослідних робіт: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  9) Розділ Наукові публікації у рамках виконання держбюджетних науково-дослідних робіт: Тест не пройдено!")

browser.get(link)

# Розділ "Конкурсний відбір проектів наукових робіт та науково-технічних розробок молодих вчених"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

contest_science_work = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Конкурсний відбір проектів наукових робіт та науково-технічних розробок молодих вчених"))
)
contest_science_work.click()

try:
    write_result("  10) Розділ Конкурсний відбір проектів наукових робіт та науково-технічних розробок молодих вчених: Тест пройдено успішно!")
except NoSuchElementException:
```

```
    write_result("  10) Розділ Конкурсний відбір проектів наукових робіт та науково-технічних розробок молодих вчених: Тест не пройдено!")

browser.get(link)

# Розділ "Рада молодих вчених"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

rada_young_sci = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Рада молодих вчених"))
)
rada_young_sci.click()

try:
    write_result("  11) Розділ Рада молодих вчених: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  11) Розділ Рада молодих вчених: Тест не пройдено!")

browser.get(link)

# Розділ "Інноваційні розробки"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()
```

```
inovation_dev = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Інноваційні розробки"))
)
inovation_dev.click()

try:
    write_result("  12) Розділ Інноваційні розробки: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  12) Розділ Інноваційні розробки: Тест не пройдено!")

browser.get(link)

# Розділ "Інтелектуальна власність"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

intellectual_property = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Інтелектуальна власність"))
)
intellectual_property.click()

try:
    write_result("  13) Розділ Інтелектуальна власність: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  13) Розділ Інтелектуальна власність: Тест не пройдено!")
```

```
browser.get(link)

# Розділ "Науково-дослідна робота студентів"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_science_link = browser.find_element(By.LINK_TEXT, "Наука").click()

science_explore = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Науково-дослідна робота студентів"))
)
science_explore.click()

try:
    write_result("  14) Розділ Науково-дослідна робота студентів: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  14) Розділ Науково-дослідна робота студентів: Тест не пройдено!")

browser.get(link)

# Розділ "Наукові видання"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")
```

```
# Меню "Наука"
men_science_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Наука"))
)
men_science_link.click()

# Розділ "Наукові видання"
science_publish = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Наукові видання"))
)
science_publish.click()

try:
    write_result("  15) Розділ Наукові видання: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  15) Розділ Наукові видання: Тест не пройдено!")

browser.get(link)
# Меню "Студенту"
men_student_link = browser.find_element(By.LINK_TEXT, "Студенту")
men_student_link.click()
with open("test_results.txt", "a") as f:
    f.write("\n12. Тест меню Студенту:\n")

# Розділ "Призначення академічних стипендій"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Студенту"
```

```
men_student_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Студенту"))
)
men_student_link.click()

# Розділ "Призначення академічних стипендій"
academ_pay = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Призначення академічних стипендій"))
)
academ_pay.click()

try:
    write_result("  1) Розділ Призначення академічних стипендій: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  1) Розділ Призначення академічних стипендій: Тест не пройдено!")

browser.get(link)

 # Розділ "Портфоліо освітніх програм"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Студенту"
men_student_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Студенту"))
)
```

```
men_student_link.click()

# Розділ "Портфоліо освітніх програм"
portf_edu_program = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Портфоліо освітніх
програм"))
)
portf_edu_program.click()

try:
    write_result("  2) Розділ Портфоліо освітніх програм: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  2) Розділ Портфоліо освітніх програм: Тест не пройдено!")

browser.get(link)

# Розділ "Графіки освітнього процесу"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Студенту"
men_student_link = browser.find_element(By.LINK_TEXT, "Студенту").click()

# Розділ "Графіки освітнього процесу"
graph_edu = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Графіки освітнього
процесу"))
)
```

```
graph_edu.click()

try:
    write_result("  3) Розділ Графіки освітнього процесу: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  3) Розділ Графіки освітнього процесу: Тест не пройдено!")

browser.get(link)

# Розділ "Розклад занять"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Студенту"
men_student_link = browser.find_element(By.LINK_TEXT, "Студенту").click()

# Розділ "Розклад занять"
time_lesson = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Розклад занять"))
)
time_lesson.click()

try:
    write_result("  4) Розділ Розклад занять: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  4) Розділ Розклад занять: Тест не пройдено!")

browser.get(link)
```

```
# Розділ "Відділ по роботі з іноземними студентами | Department of work with international students"

with open("test_results.txt", "a") as f:
    f.write("  5) Розділ Відділ по роботі з іноземними студентами | Department of work with international students: Тест пройдено успішно!\n")

# Меню "Студенту"
men_student_link = browser.find_element(By.LINK_TEXT, "Студенту")
men_student_link.click()

# Розділ "e-Campus. Електронний кампус"
e_Campus = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "e-Campus. Електронний кампус"))
)
e_Campus.click()

try:
    write_result("  6) Розділ e-Campus. Електронний кампус: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  6) Розділ e-Campus. Електронний кампус: Тест не пройдено!")

browser.get(link)
    # Розділ "Студентська рада"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")
```

```
# Меню "Студенту"
men_student_link = browser.find_element(By.LINK_TEXT, "Студенту").click()

# Розділ "Студентська рада"
stud_rada = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Студентська рада"))
)
stud_rada.click()

try:
    write_result("  7) Розділ Студентська рада: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  7) Розділ Студентська рада: Тест не пройдено!")

browser.get(link)

# Розділ "Профспілкова організація студентів"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Студенту"
men_student_link = browser.find_element(By.LINK_TEXT, "Студенту").click()

# Розділ "Профспілкова організація студентів"
profkom = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Профспілкова організація
студентів"))
)
```

```
profkom.click()

try:
    write_result("  8) Розділ Профспілкова організація студентів: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  8) Розділ Профспілкова організація студентів: Тест не пройдено!")

browser.get(link)

# Розділ "Гранти для студентів  та молоді"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Студенту"
men_student_link = browser.find_element(By.LINK_TEXT, "Студенту")
men_student_link.click()

wait = WebDriverWait(browser, 5)  # Ініціалізація об'єкту очікування

# Розділ "Гранти для студентів та молоді"
grants_stud = wait.until(EC.element_to_be_clickable((By.LINK_TEXT, "Гранти для студентів та молоді")))
grants_stud.click()

try:
    write_result("  9) Розділ Гранти для студентів та молоді: Тест пройдено успішно!")
```

```
except NoSuchElementException:
    write_result("  9) Розділ Гранти для студентів та молоді: Тест не
пройдено!")

browser.get(link)

# Розділ "Дуальна освіта"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Студенту"
men_student_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Студенту"))
)
men_student_link.click()

# Розділ "Дуальна освіта"
dual_edu = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Дуальна освіта"))
)
dual_edu.click()

try:
    write_result("  10) Розділ Дуальна освіта: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  10) Розділ Дуальна освіта: Тест не пройдено!")

browser.get(link)
```

```
# Розділ "Інформація для контрактників"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Студенту"
men_student_link = browser.find_element(By.LINK_TEXT, "Студенту")
men_student_link.click()

# Розділ "Інформація для контрактників"
info_contr = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Інформація для
контрактників"))
)
info_contr.click()

try:
    write_result("  11) Розділ Інформація для контрактників: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  11) Розділ Інформація для контрактників: Тест не
пройдено!")

browser.get(link)
# Розділ "Корисні посилання"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Студенту"
```

```
men_student_link = browser.find_element(By.LINK_TEXT, "Студенту")
men_student_link.click()

# Розділ "Корисні посилання"
useful_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Корисні посилання"))
)
useful_link.click()

try:
    write_result("  12) Розділ Корисні посилання: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  12) Розділ Корисні посилання: Тест не пройдено!")

browser.get(link)

# Розділ "Перевірка на плагіат"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Студенту"
men_student_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Студенту"))
)
men_student_link.click()

# Розділ "Перевірка на плагіат"
check_plagiarism = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Перевірка на плагіат"))
```

```
)
check_plagiarism.click()

try:
    write_result("  13) Розділ Перевірка на плагіат: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  13) Розділ Перевірка на плагіат: Тест не пройдено!")

browser.get(link)

# Меню "Вступнику"
men_vstupniky_link = browser.find_element(By.LINK_TEXT, "Вступнику")
men_vstupniky_link.click()
with open("test_results.txt", "a") as f:
    f.write("\n13. Тест меню Вступнику:\n")

# Розділ "Офіційна документація"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Вступнику"
men_vstupniky_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Вступнику"))
)
men_vstupniky_link.click()

# Розділ "Офіційна документація"
offic_doc = browser.find_element(By.LINK_TEXT, "Офіційна документація")
offic_doc.click()
```

```
try:
    write_result("  1) Розділ Офіційна документація: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  1) Розділ Офіційна документація: Тест не пройдено!")

browser.get(link)

# Розділ "Вартість навчання"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")
men_vstupniky_link = browser.find_element(By.LINK_TEXT,
"Вступнику").click()
cost_edu = browser.find_element(By.LINK_TEXT, "Вартість навчання")
cost_edu.click()
try:
    write_result("  2) Розділ Вартість навчання: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  2) Розділ Вартість навчання: Тест не пройдено!")
time.sleep(5)
browser.get(link)
# Розділ "Вступ на базі повної загальної середньої освіти"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")
men_vstupniky_link = browser.find_element(By.LINK_TEXT,
"Вступнику").click()
vstup_second_edu = browser.find_element(By.LINK_TEXT, "Вступ на базі
повної загальної середньої освіти")
```

```
vstup_second_edu.click()
try:
    write_result("  3) Розділ Вступ на базі повної загальної середньої освіти: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  3) Розділ Вступ на базі повної загальної середньої освіти: Тест не пройдено!")
time.sleep(5)
browser.get(link)

# Розділ "Вступ на базі диплома молодшого спеціаліста"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Вступнику"
men_vstupniky_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Вступнику"))
)
men_vstupniky_link.click()

# Розділ "Вступ на базі диплома молодшого спеціаліста"
vstup_young_spec = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Вступ на базі диплома молодшого спеціаліста"))
)
vstup_young_spec.click()

try:
```

```
    write_result("  4) Розділ Вступ на базі диплома молодшого спеціаліста: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  4) Розділ Вступ на базі диплома молодшого спеціаліста: Тест не пройдено!")

browser.get(link)

# Розділ "Друга вища (бакалаврат)"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Вступнику"
men_vstupniky_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Вступнику"))
)
men_vstupniky_link.click()

# Розділ "Друга вища (бакалаврат)"
second_high = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Друга вища (бакалаврат)"))
)
second_high.click()

try:
    write_result("  4) Розділ Друга вища (бакалаврат): Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  4) Розділ Друга вища (бакалаврат): Тест не пройдено!")
```

```
browser.get(link)
# Розділ "Вступ до магістратури"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Вступнику"
men_vstupniky_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Вступнику"))
)
men_vstupniky_link.click()

# Розділ "Вступ до магістратури"
vstup_magistr = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Вступ до магістратури"))
)
vstup_magistr.click()

try:
    write_result("  5) Розділ Вступ до магістратури: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  5) Розділ Вступ до магістратури: Тест не пройдено!")

browser.get(link)

# Розділ "Вступ до аспірантури"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")
```

```
# Меню "Вступнику"
men_vstupniky_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Вступнику"))
)
men_vstupniky_link.click()

# Розділ "Вступ до аспірантури"
vstup_aspirant = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Вступ до аспірантури"))
)
vstup_aspirant.click()

try:
    write_result("  6) Розділ Вступ до аспірантури: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  6) Розділ Вступ до аспірантури: Тест не пройдено!")

browser.get(link)

# Розділ "Вступ іноземних громадян та осіб без громадянства"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Вступнику"
men_vstupniky_link = browser.find_element(By.LINK_TEXT, "Вступнику")
men_vstupniky_link.click()

# Розділ "Вступ іноземних громадян та осіб без громадянства"
vstup_inozem_without_citizenship = WebDriverWait(browser, 5).until(
```

```
    EC.element_to_be_clickable((By.LINK_TEXT, "Вступ іноземних громадян
та осіб без громадянства"))
)
vstup_inozem_without_citizenship.click()

try:
    write_result("  7) Розділ Вступ іноземних громадян та осіб без
громадянства: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  7) Розділ Вступ іноземних громадян та осіб без
громадянства: Тест не пройдено!")

browser.get(link)

# Розділ "Переведення та поновлення"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Вступнику"
men_vstupniky_link = browser.find_element(By.LINK_TEXT, "Вступнику")
men_vstupniky_link.click()

# Розділ "Переведення та поновлення"
transfers_renewals = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Переведення та
поновлення"))
)
transfers_renewals.click()
```

```
try:
    write_result("  8) Розділ Переведення та поновлення: Тест пройдено
успішно!")
except NoSuchElementException:
    write_result("  8) Розділ Переведення та поновлення: Тест не пройдено!")

browser.get(link)

# Розділ "Контактна інформація"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

# Меню "Вступнику"
men_vstupniky_link = browser.find_element(By.LINK_TEXT, "Вступнику")
men_vstupniky_link.click()

# Розділ "Контактна інформація"
contact_info = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Контактна інформація"))
)
contact_info.click()

try:
    write_result("  9) Розділ Контактна інформація: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  9) Розділ Контактна інформація: Тест не пройдено!")

browser.get(link)
```

```
# Розділ "Архів"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_vstupniky_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Вступнику"))
)
men_vstupniky_link.click()

archive = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Архів"))
)
archive.click()

try:
    write_result("  10) Розділ Архів: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  10) Розділ Архів: Тест не пройдено!")

browser.get(link)

# Меню "Випускнику"
men_vipusk_link = browser.find_element(By.LINK_TEXT, "Випускнику")
men_vipusk_link.click()
with open("test_results.txt", "a") as f:
    f.write("\n14. Тест меню Випускнику:\n")

# Розділ "Відповідність оцінок"
def write_result(result):
```

```
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_vipusk_link = WebDriverWait(browser, 10).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Випускнику"))
)
men_vipusk_link.click()

correct_grades = WebDriverWait(browser, 10).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Відповідність оцінок"))
)
correct_grades.click()

try:
    write_result("  1) Розділ Відповідність оцінок: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  1) Розділ Відповідність оцінок: Тест не пройдено!")

browser.get(link)

# Розділ "Успішні далівці"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_vipusk_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Випускнику"))
)
men_vipusk_link.click()
```

```
best_stud = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Успішні далівці"))
)
best_stud.click()

try:
    write_result("  2) Розділ Успішні далівці: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  2) Розділ Успішні далівці: Тест не пройдено!")

browser.get(link)

# Розділ "ЕЛЕКТРОННА КОПІЯ ДИПЛОМУ ПРО ОСВІТУ: ЯК ОТРИМАТИ?"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_vipusk_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Випускнику"))
)
men_vipusk_link.click()

copy_dip = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "ЕЛЕКТРОННА КОПІЯ ДИПЛОМУ ПРО ОСВІТУ: ЯК ОТРИМАТИ?"))
)
copy_dip.click()

try:
```

```
    write_result("  3) Розділ ЕЛЕКТРОННА КОПІЯ ДИПЛОМУ ПРО ОСВІТУ:
ЯК ОТРИМАТИ?: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  3) Розділ ЕЛЕКТРОННА КОПІЯ ДИПЛОМУ ПРО ОСВІТУ:
ЯК ОТРИМАТИ?: Тест не пройдено!")

browser.get(link)

# Меню "Всі події"
men_podii_link = browser.find_element(By.LINK_TEXT, "Всі події")
men_podii_link.click()
with open("test_results.txt", "a") as f:
    f.write("\n15. Тест меню Всі події:\n")
# Розділ "Новини"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_podii_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Всі події"))
)
men_podii_link.click()

news = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Новини"))
)
news.click()

try:
    write_result("  1) Розділ Новини: Тест пройдено успішно!")
```

```
except NoSuchElementException:
    write_result("  1) Розділ Новини: Тест не пройдено!")

browser.get(link)

# Розділ "Офіційно"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_podii_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Всі події"))
)
men_podii_link.click()

official = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Офіційно"))
)
official.click()

try:
    write_result("  2) Розділ Офіційно: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  2) Розділ Офіційно: Тест не пройдено!")

browser.get(link)

# Розділ "Анонси"
def write_result(result):
    with open("test_results.txt", "a") as f:
```

```
        f.write(result + "\n")

men_podii_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Всі події"))
)
men_podii_link.click()

anons = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Анонси"))
)
anons.click()

try:
    write_result("  3) Розділ Анонси: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  3) Розділ Анонси: Тест не пройдено!")

browser.get(link)

# Меню "Контакти"
men_contact_link = browser.find_element(By.LINK_TEXT, "Контакти")
men_contact_link.click()
with open("test_results.txt", "a") as f:
    f.write("\n16. Тест меню Контакти:\n")
# Розділ "Зворотний зв'язок"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_contact_link = WebDriverWait(browser, 5).until(
```

```
    EC.element_to_be_clickable((By.LINK_TEXT, "Контакти"))
)
men_contact_link.click()

call_back = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Зворотний зв'язок"))
)
call_back.click()

try:
    write_result("  1) Розділ Зворотний зв'язок: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  1) Розділ Зворотний зв'язок: Тест не пройдено!")

browser.get(link)

# Розділ "Контакти підрозділів"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result + "\n")

men_contact_link = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Контакти"))
)
men_contact_link.click()

pidrozdil_contacts = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Контакти підрозділів"))
)
pidrozdil_contacts.click()
```

```
try:
    write_result("  2) Розділ Контакти підрозділів: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  2) Розділ Контакти підрозділів: Тест не пройдено!")

browser.get(link)

# Меню пошуку
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write("\n17. Тест меню пошуку: " + result)

search_button = browser.find_element(By.ID, "search-btn")
search_button.click()

search_input = WebDriverWait(browser, 5).until(
    EC.visibility_of_element_located((By.XPATH,
"/html/body/div/header/div[2]/div/div/div/form/label/input"))
)
search_input.send_keys("Студ рада")

submit_button = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.XPATH,
"/html/body/div/header/div[2]/div/div/div/form/input"))
)
submit_button.click()

if submit_button:
```

```
    write_result("Тест меню пошуку (пошуковий запит - Студ рада) : Тест пройдено успішно!\n")
else:
    write_result("Тест меню пошуку (пошуковий запит - Студ рада) : Тест не пройдено!\n")

browser.get(link)


# Тест Слайдера
count = 0
with open('test_results.txt', 'a') as f:
    for i in range(10):
        search_string = browser.find_element(By.CSS_SELECTOR, "a[class*=flex-next]")
        time.sleep(0.5)
        dia = search_string.click()
        count += 1
    if count == 10:
        f.write("\n18. Тест Слайд-меню: Тест пройдено успішно!\n")
    else:
        f.write("\n18. Тест Слайд-меню: Тест не пройдено!\n")
browser.get(link)
# Тест анімованого меню
# 1
def write_result(result):
    with open('test_results.txt','a') as f:
        f.write("\n19. Тест Тест анімованого меню:\n")
        f.write(result)
```

```
anim_links_text1 = WebDriverWait(browser, 5).until(
   EC.element_to_be_clickable((By.LINK_TEXT, "Як розібратися в
особливостях Вступної кампанії-2023?"))
)
anim_links_text1.click()

try:
   write_result("  1) Розділ Як розібратися в особливостях Вступної кампанії-
2023?: Тест пройдено успішно!")
except NoSuchElementException:
   write_result("  1) Розділ Як розібратися в особливостях Вступної кампанії-
2023?: Тест не пройдено!")
browser.get(link)
# 2
def write_result(result):
   with open('test_results.txt','a') as f:
      f.write(result + "\n")
anim_links_text2 = WebDriverWait(browser, 5).until(
   EC.element_to_be_clickable((By.LINK_TEXT, "Календар вступника"))
)
anim_links_text2.click()

try:
   write_result("\n  2) Розділ Календар вступника: Тест пройдено успішно!")
except NoSuchElementException:
   write_result("\n  2) Розділ Календар вступника: Тест не пройдено!")
browser.get(link)

# 3
def write_result(result):
```

```
    with open('test_results.txt','a') as f:
        f.write(result + "\n")
anim_links_text3 = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Плануєте стати учасниками національного мультипредметного тесту (НМТ)?"))
)
anim_links_text3.click()

try:
    write_result("  3) Розділ Плануєте стати учасниками національного мультипредметного тесту (НМТ)?: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  3) Розділ Плануєте стати учасниками національного мультипредметного тесту (НМТ)?: Тест не пройдено!")
browser.get(link)
# 4
def write_result(result):
    with open('test_results.txt','a') as f:
        f.write(result + "\n")
anim_links_text4 = WebDriverWait(browser, 5).until(
    EC.element_to_be_clickable((By.LINK_TEXT, "Анкета вступника-2023"))
)
anim_links_text4.click()

try:
    write_result("  4) Розділ Анкета вступника-2023: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("  4) Розділ Анкета вступника-2023: Тест не пройдено!")
browser.get(link)
```

```
# Тест кнопки "АКТУАЛЬНО ПРО ВСТУП"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result)

actual_vstup = browser.find_element(By.LINK_TEXT, "АКТУАЛЬНО ПРО
ВСТУП")
actual_vstup.click()
try:
    WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.LINK_TEXT, "АКТУАЛЬНО ПРО
ВСТУП"))
    )
    write_result("\n20. Тест кнопки АКТУАЛЬНО ПРО ВСТУП: Тест пройдено
успішно!\n")
except NoSuchElementException:
    write_result("\n20. Тест кнопки АКТУАЛЬНО ПРО ВСТУП: Тест не
пройдено!\n")

browser.get(link)

# Тест новини "Співпраця Університету зі Святошинським районом столиці"
# news1
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write("\n21. Тест блоку новин:\n")
        f.write(result)
```

```
news1 = browser.find_element(By.XPATH,
"//*[@id='courses']/div/ul/li[1]/div/div/h2/a")

try:
    news1.click(

    )
    write_result("    1) Тест новини Співпраця Університету зі Святошинським районом столиці: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("    1) Тест новини Співпраця Університету зі Святошинським районом столиці: Тест не пройдено!")

browser.get(link)

#news2
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result)

news2 = browser.find_element(By.XPATH,
"//*[@id='courses']/div/ul/li[2]/div/div/h2/a")

try:
    news2.click(

    )
    write_result("\n    2) Тест новини Гостьова лекція для майбутніх психологів: Тест пройдено успішно!")
except NoSuchElementException:
```

```
    write_result("\n    2)Тест новини Гостьова лекція для майбутніх психологів:
Тест не пройдено!")

browser.get(link)

#news3
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result)

news3 = browser.find_element(By.XPATH,
"//*[@id='courses']/div/ul/li[3]/div/div/h2/a")

try:
    news3.click(

    )
    write_result("\n    3) Тест новини Хмарні сховища – в допомогу
переміщеним університетам: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("\n    3)Тест новини Хмарні сховища – в допомогу
переміщеним університетам: Тест не пройдено!")

browser.get(link)

#news4
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result)
```

```
news4 = browser.find_element(By.XPATH,
"//*[@id='courses']/div/ul/li[4]/div/div/h2/a")

try:
    news4.click(

    )
    write_result("\n    4) Тест новини Студентське самоврядування під час війни і після неї: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("\n    4) Тест новини Студентське самоврядування під час війни і після неї: Тест не пройдено!")

browser.get(link)

#news5
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result)

news5 = browser.find_element(By.XPATH,
"//*[@id='courses']/div/ul/li[5]/div/div/h2/a")

try:
    news5.click(

    )
    write_result("\n    5) Тест новини Участь науковиці СНУ ім. В. Даля у Форумі викладачів громадянської освіти: Тест пройдено успішно!")
except NoSuchElementException:
```

```
    write_result("\n    5) Тест новини Участь науковиці СНУ ім. В. Даля у Форумі викладачів громадянської освіти: Тест не пройдено!")

browser.get(link)

#news6
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result)

news6 = browser.find_element(By.XPATH, "//*[@id='courses']/div/ul/li[6]/div/div/h2/a")
try:
    news6.click(

    )
    write_result("\n    6) Тест новини Візит норвезького журналіста і підприємця до СНУ ім. В. Даля: Тест пройдено успішно!")
except NoSuchElementException:
    write_result("\n    6) Тест новини Візит норвезького журналіста і підприємця до СНУ ім. В. Даля: Тест не пройдено!")

browser.get(link)

# Тест кнопки "Всі події"
def write_result(result):
    with open("test_results.txt", "a") as f:
        f.write(result)

actual_vstup = browser.find_element(By.LINK_TEXT, "Всі події")
```

```
actual_vstup.click()
try:
    WebDriverWait(browser, 5).until(
        EC.presence_of_element_located((By.CLASS_NAME, "btn-learn-more"))
    )
    write_result("\n\n22. Тест кнопки Всі події: Тест пройдено успішно!\n")
except NoSuchElementException:
    write_result("\n\n22. Тест кнопки Всі події: Тест не пройдено!\n")

browser.get(link)

with open('test_results.txt','a') as f:
    f.write("\nТЕСТ ЗАВЕРШЕНО!\n")
```