Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ІНФОРМАЦІЙНО ІНТЕГРОВАНА СИСТЕМА ОЦІНЮВАННЯ ГОТЕЛЬНОГО БІЗНЕСУ

Виконав: студент 2 курсу, групи 1КІ-22м
спеціальності 123 — «Комп'ютерна інженерія»
_____ Щербань К. П.

Керівник: к.т.н., доц.каф. ОТ
_____ Тарновський М. Г.
«15» 12 2023 р.

Опонент: к.т.н., доц. каф. ПЗ
_____ Войтко В. В.
«18» 12 2023 р.

Допущено до захисту

Завідувач кафедри ОТ
_____ д.т.н., проф. Азаров О. Д.
«20» 12 2023 р

ВНТУ 2023

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ~~ОТ~~, д.т.н., проф.
_____ Азаров О. Д.
"26" 09 2023року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Щербаню Костянтину Павловичу

1 Тема роботи «Інформаційно інтегрована система оцінювання готельного бізнесу», керівник роботи Тарновський М. Г. к.т.н., доцент, затверджено наказом вищого навчального закладу від 18.09.23 року № 247.

2 Строк подання студентом роботи 18.12.2023.

3 Вихідні дані до роботи: призначення системи — тестування веб-сайтів готелів; при тестуванні використовувати інформацію з баз даних міжнародних систем бронювання; підтримка можливості змінювати тестові сценарії.

4 Зміст текстової частини (перелік питань, які потрібно розробити): вступ, аналіз предметної області, аналіз методів та засобів тестування веб-застосунків; розробка системи тестування сайтів готелів, тестування системи, економічна частина.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): архітектура системи, схема основних модулів системи, діаграма прецедентів.

6 Консультанти розділів роботи наведені в таблиці 1.

Таблиця 1 — Консультанти роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Тарновський М. Г. к.т.н., доцент	19.09.2023р	15.12.2023р
5	Небава М. І. проф., к.е.н	1.11.2023	15.12.2023

7 Дата видачі завдання «19» 09 2023 року.

8 Календарний план виконання приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломного роботи	Термін виконання		Примітка
		початок	закінчення	
1	Постановка задачі роботи	19.09.2023		виконав
2	Обгрунтування актуальності теми. Аналіз предметної області	19.09.2023	15.10.2023	виконано
3	Аналіз методів та засобів тестування веб-застосунків	16.10.2023	29.10.2023	виконано
4	Розробка системи тестування сайтів готелів	30.10.2023	19.11.2023	виконано
5	Тестування системи	20.11.2023	26.11.2023	виконано
6	Оцінка комерційного потенціалу розробки	27.11.2023	3.12.2023	виконано
7	Оформлення пояснювальної записки та ілюстративного матеріалу	4.12.2023	10.12.2023	виконано
8	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	11.02.2023	15.02.2023	виконано

Студент _____ Щербань К. П.

Керівник роботи _____ Тарновський М.Г.

АНОТАЦІЯ

Щербань К. П. Інформаційно інтегрована система оцінювання готельного бізнесу. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна Інженерія, Вінниця: ВНТУ, 2023, 83 с.

На укр. мові. Бібліогр.: 22 назв; рис.: 32; табл. 5.

Інтернет-ресурси стали ключовим елементом маркетингових стратегій готелів, дозволяючи гнучко керувати даними для взаємодії з споживачами. В сфері соціально-культурного сервісу та туризму інформаційні технології виявляються однією з найсуттєвіших складових. Результативність їх використання суттєво впливає на продуктивність туристичного бізнесу, де збір, обробка і передача інформації стають все більш актуальними завдяки надійності та оперативності.

Серед інноваційних технологій, що активно впроваджуються в готельний бізнес, можна виділити наступні: системи управління готелем на хмарних платформах, автоматизовані системи самообслуговування, розробка мобільних додатків, використання технологій штучного інтелекту, впровадження віртуальної та доповненої реальності, застосування технологій 3D-зображення, використання технології блокчейн, а також Інтернет речей (IoT) , тощо.

Ключові слова: веб-сервіс, автоматизоване тестування,.NET Core, React

ABSTRACT

Shcherban K.P.

Information integrated hotel evaluation system. Information integrated hotel business evaluation system. Master's qualification work in specialty 123 - computer engineering, educational program - computer engineering. Vinnytsia: VNTU, 2023, 83 p.

In Ukrainian. Refernces.: 22 titles, Fig.32, Tab. 5.

Internet resources have become a crucial element of hotel marketing strategies, enabling flexible data management for consumer market interaction. In the field of social-cultural services and tourism, information technologies stand out as one of the most essential components. The effectiveness of their application significantly determines the productivity of the tourism industry, where the reliability and promptness of information gathering, processing, and transmission become increasingly relevant.

Among the innovative technologies actively implemented in the hotel business, the following can be highlighted: cloud-based hotel management systems, self-service automated systems, mobile application development, the use of artificial intelligence technologies, the adoption of virtual and augmented reality, the application of 3D imaging technologies, blockchain technology utilization, as well as Internet of Things (IoT) technology, and many others.

Keywords: web service, automated testing, .NET Core, React.

ЗМІСТ

ВСТУП

Сучасне життя важко уявити життя без різноманітних гаджетів та Інтернету. Тому практично усі сучасні бізнеси та підприємства прагнуть вести свою діяльність не тільки в офлайні, а й онлайн, надаючи клієнтам можливість самостійно отримувати інформацію, замовляти та оплачувати послуги, здійснювати покупки і т.і. Усі ці дії здійснюються за допомогою Інтернет-ресурсів.

Актуальність теми дослідження полягає в тому, що готельний бізнес в останні десятиріччя зазнав значної трансформації, пов’язаної з впровадженням комп’ютерних технологій. У результаті у теперішній час споживачі мають повний контроль над усіма етапами планування поїздки: від пошуку призначення до бронювання житла. Це робить обов'язковою наявність у будь-якого готелю постійної присутності в Інтернеті, щоб привести споживача з пошукової системи безпосередньо на офіційний сайт готельного підприємства — основний канал онлайн-продажів та основну точку контакту між гостем та готелем.

Веб-сайти мають необмежені можливості для створення позитивного першого враження у потенційного гостя: можна розмістити фотографії номерів готелю, відгуки клієнтів або нагороди готелю. Вони є основним місцем, де гості шукають інформацію про готель, оскільки інформація, що представлена на сайті, доступна у будь-який час. Як результат, веб-сайт є ідеальною платформою для передачі цінностей готельного бренду та надання маркетингових повідомлень, які приваблюють цільову аудиторію. У той самий час, веб-сайти є гарним інструментом для підвищення рівня представництва готелю в Інтернеті, особливо коли ефективно виконується пошукова оптимізація [1].

Разом із цим веб-сайт готельного підприємства є не лише джерелом інформації для гостей, а й для готелю. З одного боку він, як правило, надає широкий спектр інформації, яка створює імідж та повідомляє про готель: його

історію, визначні пам'ятки, відмінні риси, можливості для організації конференцій, пропозиції ресторанів тощо. З іншого — завдяки веб-сайту готель бачить аналітику та статистику. Наприклад, на якій сторінці чи місці на сторінці затримуються споживачі, які фотографії більше подобаються або яку інформацію пропускають, і тим самим вона є зайвою [2].

З розвитком і популяризацією веб-застосувань, вони все більше стають складними та великими. Такі веб-застосування стає все важче не лише розробляти, а й підтримувати, контролюючи та забезпечуючи вірність їх функціонування. Контроль за працездатністю сайту забезпечується шляхом його тестування, під час якого саме і відбувається перевірка відповідності між реальною поведінкою програми та поведінкою, яку очікує клієнт.

Метою дослідження є вдосконалення інформаційної системи тестування сайтів міжнародних готелів для оцінювання доступності та достовірності представленої на них інформації.

Задачі дослідження:

— реалізація інформаційної мережевої системи тестування;

— аналіз інформаційних систем;

— огляд існуючих систем автоматизованого тестування;

— проведення варіантного аналізу засобів для розроби сайтів, на основі якої створено веб-додаток

Об’єкт дослідження — процеси збору та аналізу даних в мережі Інтернет.

Предмет дослідження — інформаційні системи автоматичного тестування Інтернет-ресурсів.

Новизна роботи полягає у вдосконалені системи автоматичного тестування веб-сайтів, в якій, на відміну від існуючих, реалізовано порівняння інформації, що представлена на сайті, з даними, що зберігаються у базах даних міжнародних систем бронювання.

Практичне значення роботи полягає в тому, що отримується можливість оцінювати достовірність інформації, що подається на сайтах готелів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Сучасні Інтернет-технології в готельній індустрії

Розвиток комп’ютерних систем та мережних технологій створив умови для всеосяжної комп’ютеризації та інформатизації будь-яких бізнесів. Не залишилася осторонь і сфера готельного бізнесу, в якій впровадження комп’ютерних інформаційних технологій відбулося одним із перших. Це пов’язано з тим, що готельний бізнес завжди був пов’язаний з обробкою великих обсягів інформації: бронювання, управління номерної базою тощо. З розвитком технологій готельні компанії почали використовувати спеціалізовані системи, щоб покращити якість обслуговування гостей та спростити роботу персоналу.

З точки зору організації та управління готельні комплекси є складними системами, які складаються з взаємопов'язаних служб. Кількість та завдання кожної з них визначаються категорією готелю, специфікою номерного фонду, розташуванням та низкою інших факторів. При цьому є підрозділи, присутні у структурі кожного готелю. Це відділ управління, адміністрація, підрозділ обслуговування, служба бронювання. Комп’ютерна інформаційна система готельного комплексу покликана об'єднати перелічені модулі та забезпечити їх функціональну взаємодію [3].

Комп'ютеризація готельного бізнесу сьогодні може відбуватися на трьох рівнях. Перший включає автоматизацію бізнес-процесів усередині готелю, що забезпечується впровадженням інформаційної системи, яка є сукупністю бази даних та комплексу програмно-апаратних засобів для обробки інформації, що зберігається в цій базі даних. Інформація про роботу готелю накопичується та зберігається на сервері. Сервер баз даних і пов'язані з ним локальною мережею робочі місця підрозділів із встановленим на них програмним забезпеченням, утворюють внутрішню мережну систему готелю клієнт-серверної архітектури.

Клієнт-серверна технологія забезпечує доступ до бази даних з будь-якого робочого місця відповідно до прав доступу [4], [5].

Популяризація та розвиток Інтернет технологій обумовлює перехід до наступного, другого рівня, який полягає у створенні зовнішньої інформаційної системи для підтримки бізнес-процесів готелю через мережу Інтернет. У результаті отримується можливість зв'язати внутрішню комп'ютерну інформаційну систему готелю з її зовнішніми партнерами та клієнтами. Сучасні комп'ютерні системи управління готельними підприємствами, що з'явилися завдяки Інтернету, дозволили оптимізувати внутрішні комунікаційні взаємини між готельними підрозділами, створити новий механізм управління та оформлення клієнтів тощо.

На третьому рівні внутрішня та зовнішня системи об'єднуються в єдине бізнес-середовище. Це забезпечує інтегрування усіх внутрішніх служб готелю та надає можливість формувати відгуки на будь-які запити ззовні з використанням методів електронного обміну даними.

Сучасні комп'ютерні системи готельних комплексів мають відкриту архітектуру, що надає системі велику гнучкість, легкість у використанні та широкі можливості інтеграції з системами інших готелів та зовнішніми інформаційними ресурсами через мережу Інтернет. Це дозволяє об'єднувати інформаційні та управлінські ресурси окремих готелів у єдиний інформаційний простір, що надає значні можливості для збору та аналізу даних.

Розвиток Інтернету дозволяє не лише здешевити засоби зв'язку, а й отримати реальну можливість налагодити роботу всіх учасників ринку готельних послуг як єдиного офісу. Дані про заявки гостей, тарифи, за якими гості проживають, їх вподобання та побажання стають загально доступними для аналізу, що дозволяє оперативно та гнучко адаптувати бізнес-політику готелю під індивідуальні потреби різних сегментів клієнтів та покращувати сервіс. Створюється єдина інформаційна база, яка є потужним інструментом для аналізу ринку та ефективного маркетингу готельних послуг [6].

Поряд із цим бази даних, що лежать в основі сучасних систем управління готелями, дозволяють акумулювати та зберігати докладну інформацію щодо роботи готелю та його взаємин із кожним гостем. Ефективне використання зібраних даних є ключовим фактором для досягнення готелем конкурентної переваги на ринку. Накопичені дані стають безцінним капіталом для готелю. Вони полегшують готелі прогнозування попиту послуги та проведення більш ефективної маркетингової політики. Готель отримує можливість реалізовувати програми частого гостя та заохочувати своїх постійних клієнтів. Платіжна історія кожного клієнта готелю дозволяє правильно будувати кредитну політику.

Інтернет-технології оказали значний вплив на підходи до надання готельних послуг. Сьогодні Інтернет використовується практично у всіх основних бізнес-процесах сучасного готелю, починаючи від пошуку та залучення клієнтів як комунікаційний та маркетинговий інструмент і закінчуючи формуванням асортименту послуг. Однією з найважливіших серед них є можливість онлайн бронювання [1], [5].

Системи онлайн бронювання готельних номерів — це програмні платформи, які дозволяють користувачам забронювати номер онлайн. Такі системи включають інформацію про готелі та номери, доступні дати та ціни, а також послуги, що пропонуються в готелях. Вони дозволяють гостям швидко та зручно забронювати номер, дізнатися ціни та інші умови.

Онлайн бронювання може відбуватися через веб-сайт готелю або через глобальні системи бронювання. В мережі Інтернет працюють багато самостійних розділів готелів та багаточисельні системи бронювання номерів. Вони активно використовуються у діяльності туристичних агенцій та туристичних операторів. Поряд із ними і окремі клієнти, які шукають можливість самостійно вибрати для себе номер.

Інформаційні ресурси Інтернету є важливим елементом маркетингової стратегії готелю. Вони забезпечують динамічне інформаційне управління

споживчим ринком, а також дають можливість представляти себе у глобальних GDS (Global Distribution System) та (або) альтернативних ADS (Alternative Distribution System) системах бронювання, внаслідок чого готельні підприємства отримують доступ до численних варіантів каналів продажу. Робота з глобальними системами бронювання через Інтернет дозволяє готелям не лише надавати всім учасникам ринку оперативну та достовірну інформацію ціни та кількість місць, а й мати можливість стежити за проходженням замовлення всіх етапах його здійснення.

Важливою перевагою для будь-якого готелю, представленого в міжнародних системах бронювання та в Інтернеті, безумовно, є своєчасність, повнота та доступність інформації, що передається. Отримується можливість не лише в режимі реального часу приймати заявки та передавати підтвердження бронювання, а й проводити гнучку маркетингову та цінову політику, досягаючи максимальної прибутковості від кожної отриманої заявки. Створювати високоефективну стратегію продажів номерного фонду, що базується на аналізі тенденцій та взаємодій на ринку, готель має можливість, контролюючи умови реалізації своїх номерів. Ефективність досягається за рахунок впровадження нового покоління готельних систем, що створюють своєрідну інтерактивну інформаційну базу готелів, інтегровану з електронними системами бронювання. Запит кінцевого клієнта про готельні послуги автоматично обробляється з урахуванням його індивідуальних переваг, дозволяючи миттєво скласти оптимальну пропозицію, та з великою ймовірністю забезпечення позитивної реакції клієнта та подальше здійснення бронювання.

Найповніша інтеграція офісів бронювання готельних компаній з окремими готелями досягається при використанні спеціалізованих систем централізованого бронювання CRS (Central Reservations Systems). Вони дають ширші можливості пошуку, вибору та бронювання номерів у готелях мережі. Крім того, за допомогою систем центрального бронювання окремі готелі також мають можливість здійснювати взаємне бронювання номерів в інших готелях

мережі. Таким чином, досягається тісна взаємодія готелів один з одним та з центральним офісом, відбувається об'єднання зусиль усіх готелів для залучення нових та утримання старих клієнтів.

З іншого боку, комп’ютерні системи бронювання дозволяють клієнтам отримувати інформацію з великих баз даних щодо наявності вільних номерів, пропонованих послуг, вартості, якості, часу прибуття та відправлення і т.і. Спеціальний модуль системи ART (Automated Request Tools — Автоматизовані Інструменти Бронювання), що виконаний за принципом ASP (Application Service Provider — Постачальник Послуг Застосувань) та функціонує на будь-якому комп'ютері, підключеному до Інтернет. Модуль ART дозволяє клієнтам готелю самостійно в реальному режимі часу через Інтернет здійснювати бронювання номерів, конференц-приміщень та передавати заявки на проведення заходів. Модуль включає такі системи: «Обмін даних», «Конфігурація», «Клієнти», «Туристичні агенції», «Рахунок до отримання», «Продажі», «Управління тарифами», «Центральне бронювання».

Майбутнє індустрії готельного бізнесу визначається сучасним бумом технологій Інтернету речей (IoT). Інтернет речей — це система взаємопов'язаних фізичних пристроїв, оснащених датчиками, програмним забезпеченням, мережевим підключенням, які дозволяють об'єктам збирати, обмінюватися і діяти на основі даних. За своєю суттю IoT є концепцією, у межах якої пристрої, незалежно від своїх природи чи функцій, пов'язані між собою через Інтернет. Завдяки цьому отримується можливість об'єднувати різні системи між собою, будувати мережу мереж. Це дозволяє змінити бізнес-моделі цілих галузей, формуючи нові правила економіки спільного використання, виключаючи посередників з бізнес-моделі. У діловому світі Інтернет речей зробив глибокий вплив, перетворивши галузі та відкривши нові можливості для інновацій. Можливість збирати та аналізувати величезні обсяги даних із взаємозалежних пристроїв справила революцію у процесах прийняття рішень та операційної ефективності [6].

Впровадження технології IoT в готельну індустрію кваліфікує готелі як розумні будинки, які є важливими аспектами розумних міст. Парадигма IoT відкриває нові можливості для негайних, персоналізованих та локалізованих послуг, оскільки готель може більш точно оцінювати поведінку та те, чому віддають перевагу гості.

Інтернет речей також дозволяє готелям підвищувати внутрішню ефективність різних відділів, таких як, наприклад, стійки реєстрації, хаускіпінгу (служба прибирання, прасування, догляду за готелем і його інвентарем), відділів продажу, маркетингу тощо. Крім того, засоби інтернету речей також надають можливість впроваджувати політику економії через інтелектуальне управління енергоспоживанням, а також виявляти несправності та збої в реальному час, полегшуючи оперативне технічне обслуговування [7].

У готелях, в яких широко розгорнуті системи Інтернету речей, запити на ремонт та технічне обслуговування можуть бути задоволені швидко, оскільки більшість датчиків та пристроїв можуть виявляти та самостійно діагностувати проблеми. Своєчасний ремонт та технічне обслуговування дозволяють швидко заселити готельні номери, тим самим скоротити втрати доходів. Щоб підвищити швидкість «реагування» готельних систем, вони повинні бути оснащені великою кількістю обчислювальних ресурсів та безперешкодним доступом до гостьових даних та даних систем управління, що потребує децентралізованої платформи обчислень та управління даними. Технологія IoT вже поширюється в індустрії гостинності з терміналами реєстрації, технологіями в номерах та мобільними програмами для готелів, а деякі з перспективних майбутніх технологій IoT, таких як доповнена реальність або розпізнавання осіб, безсумнівно, приведуть до успіху та відкриють нові перспективи.

1.2 Системи збору та аналізу даних з Інтернет-ресурсів готелів

Зазвичай вибір готелю розпочинається з використання загальної мережі вибору готелів, таких як наприклад booking.com, де користувач має можливість

ознайомитися з тими чи іншим готелем. Ця інформація береться із баз даних. Крім того, для успішного маркетингу потрібно проводити збір та аналіз даних із сайтів конкурентів.

Автоматизований збір інформації з будь-якого сайту, її аналіз, перетворення та видача у структурованому вигляді, найчастіше у вигляді таблиці з набором даних називається «парсингом». Парсер сайту — це будь-яка програма чи сервіс, що здійснює автоматичний збір інформації із заданого ресурсу. Парсинг інформації з сайтів конкурентів дозволяє у режимі реального часу дозволяє оперативно реагувати на дії конкурентів. При цьому вартість послуг може автоматично змінюватися в залежності від календарної дати, дня тижня, часу доби [8].

Взагалі парсинг можна розділити на два типи: технічний та парсинг з метою розвитку бізнесу. Технічний парсинг використовується для виявлення різних проблем сайту:

— пошук недійсних та некоректних посилань;

— виявлення дублів або інших проблем з мета-тегами та заголовками;

— перевірка налаштування мікророзмітки на сайті;

— виявлення небажаних сторінок, які відкриті для індексації;

— інші технічні завдання.

На основі отриманих даних спеціаліст складає технічні завдання усунення виявлених проблем.

Парсинг сайту з метою розвитку бізнесу використовується для вирішення таких завдань:

— збір інформації про пропозиції конкурентів;

— збір відгуків та коментарів;

— аналіз структури сайтів-конкурентів з метою поліпшення та розвитку власної структури.

Процес парсингу пов’язаний з вилученням великого масиву даних із веб-ресурсів та реалізується за допомогою спеціальних скриптів. Парсер ходить за

посиланнями вказаного сайту і сканує код кожної сторінки, збираючи інформацію про неї у будь-який файл. Результатом парсингу є сукупність інформації з усіх сторінок сайту.

Парсинг працює на основі XPath-запитів, за допомогою яких відбувається звернення до певної ділянки коду сторінки та витягує з нього задану критерієм інформацію. Алгоритм стандартного парсингу сайту є таким:

— пошук необхідних даних у вихідному вигляді;

— вилучення даних із відділенням від програмного коду;

— формування звіту відповідно до вимог, які були поставлені.

У теперішній час можна виділити чотири основні інструменти для парсингу сайтів:

— Google таблиці (Google Spreadsheet);

— NetPeak Spider;

— ComparseR;

— Screaming Frog SEO Spider.

Google таблиці є зручний способом для парсингу, якщо немає необхідності парсити велику кількість даних, оскільки є ліміти на кількість xml запитів на день. За допомогою Google таблиць можна парсувати метадані, заголовки, найменування товарів, ціни, пошту та багато іншого [9].

Google таблиці надають набір різних інструментів для збору зовнішніх даних. Серед них основними є функції імпорту даних з іншої таблиці Google ImportRange, функція імпорту таблиць та списків з веб-сторінки ImportHTML та функція імпорту інших даних з веб-сторінки ImportXML.

Ще одним інструментом є програма NetPeak Spider для швидкого пошуку помилок, системного аналізу та парсингу сайтів. Безкоштовний період 14 днів, є варіанти платних ліцензій на місяць та більше [10].

Програма має інтуїтивно зрозумілий інтерфейс, дозволяє знаходити та кластеризувати помилки, знайдені на сайті, позначає їх різними кольорами в залежності від ступеня критичності.

Основні можливості Netpeak Spider:

— перевіряє більше 80 ключових помилок внутрішньої оптимізації сайту;

— аналізує понад 70 базових параметрів оптимізації під пошукові системи (SEO —Search Engine Optimization);

— висока швидкість сканування;

— можливість аналізу великих сайтів;

— налаштування та парсинг кастомних html-даних.

Ще одним з інструментів є спеціалізована програма ComparseR, що призначена для глибокого вивчення індексації сайту. Демо-версія ComparseR має 2 обмеження:

— парсіт лише перші 150 сторінок сайту або перші 150 результатів видачі;

— не має механізму самооновлення та демо-дистрибутив оновлюється лише у критичних випадках.

Парсер ComparseR примітний тим, що він дозволяє порівнювати інформацію на сайті і тим, що індексується в пошукових системах. Відповідно він дозволяє знайти сторінки, які не індексуються пошуковими системами, або, навпаки, сторінки-сироти, тобто сторінки, на які немає посилань на сайті [11].

Ще одним прикладом парсера є програма Screaming Frog SEO Spider. Програма вимагає встановлення JAVA, дозволяє можливість налаштування та запуску програми відповідно до розкладу із заданими налаштуваннями парсингу зі збереженням усіх необхідних звітів та підтримує підключення різних API [12]:

— Google Analytics;

— Google Search Console;

— PageSpeed Insights;

— Majestic;

— Ahrefs;

— Moz.

Найбільш орієнтованим на збір даних з сайтів готелів є парсер Booking.com, що є налаштуванням Datacol, яке автоматично отримує інформацію про готелі з сайту booking.com. Парсер підтримує понад 15 форматів для експорту зібраних даних, а саме: експорт у базу даних, CMS, файли CSV, XML, XLS, можливість налаштувати власний формат експорту. Парсер Booking автоматично здійснює збір таких даних: назва готелю, адреса, оцінка, опис та зображення. Інформація про послуги автоматично зберігається у файлі CSV [13].

Основні переваги парсера Booking.com на базі Datacol це:

— можливість доналаштування парсера booking.com безпосередньо під потреби користувача;

— можливість перекладати, додатково обробляти зібрані дані за допомогою плагінів, а також завантажувати їх у різні формати та cms;

— можливість циклічного запуску, коли результати виконання першого завдання парсингу будуть вхідними для другого завдання збору даних.

1.3 Аналіз сучасних систем тестування Інтернет-сайтів

На сьогоднішній день існує безліч систем для автоматизованого тестування веб-сайтів. Одні з них мають доволі значну вартість при достатньо мізерному функціоналі і низької швидкості роботи. Інші є безкоштовними та відкритими системами, але більшість їх створюють тести за методом record and play.

Одним з прикладів системи тестування сайтів є програма Selenium, реалізована на Java та може аналізувати файли певного фреймворку, щоб знаходити команди для керування браузером і команди для виконання певних дій і перевірок [14].

Інтерфейс програми Selenium показаний на рис. 1.1. Selenium підтримує Microsoft Windows Explorer, Google Chrome, Mozilla Suite і Mozilla Firefox для Microsoft Windows, Linux і Apple Macintosh. Проект Selenium також створює

інструмент Selenium IDE, версію популярної бібліотеки Selenium з графічним інтерфейсом. Він реалізований як розширення браузера Firefox розміром 240 Кб, включаючи сам Selenium. Цей інструмент дозволяє записувати та відтворювати записи, які є стандартними HTML-сторінками з єдиною таблицею, що містить команди. Самоперевірки створюються в Selenium IDE відповідно до сценарію та ігрової моделі [14].

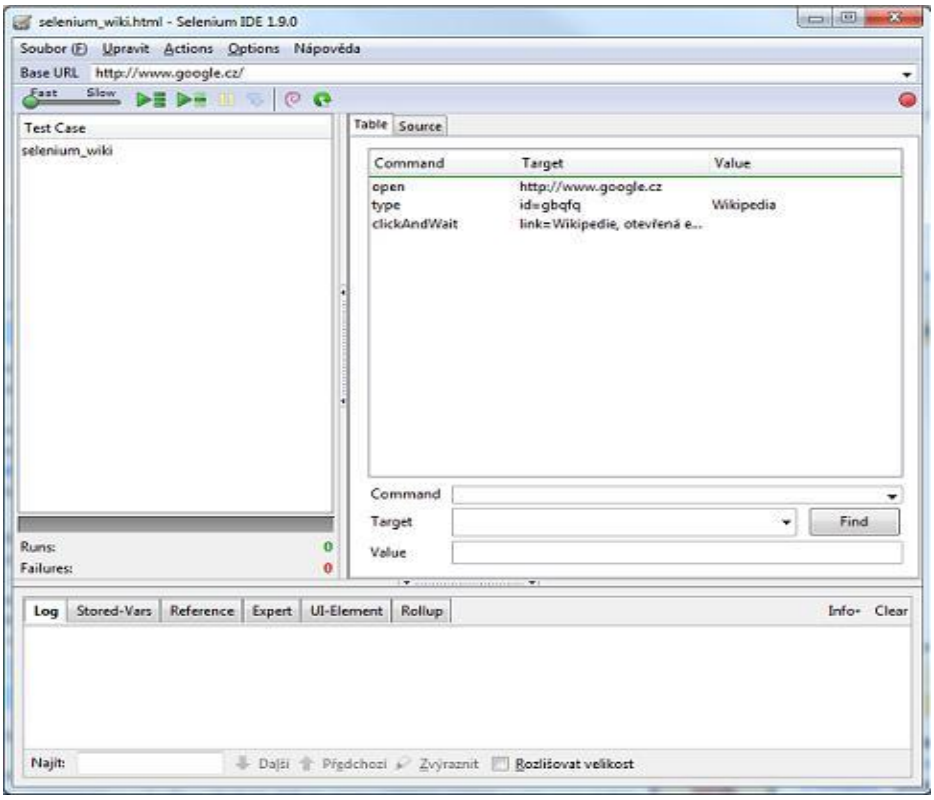


Рисунок 1.1 — ІНТЕРФЕЙС SELENIUM

Ще одним прикладом є програма HP QuickTest Professional (QTP) — один із провідних інструментів для автоматизації функціонального тестування. QTP використовує VBScript для розробки автоматизованих тестів (рис. 1.2) [15].

QTP підтримує нижченаведені технології:

—технологію Windows Presentation Foundation;

—технологію Web services;

—технологію Macromedia Flex;

—технологію Ajax;

—технологію Delphi;

—технологію .NET;

—технологію VisualBasic;

—технологію ActiveX;

—технологію Java;

—технологію Oracle;

—технологію SAPSolution;

—технологію PowerBuilder;

—технологію технологію Siebel;

—технологію PeopleSoft;

—технологію VisualAge;

—технологію Stingray.

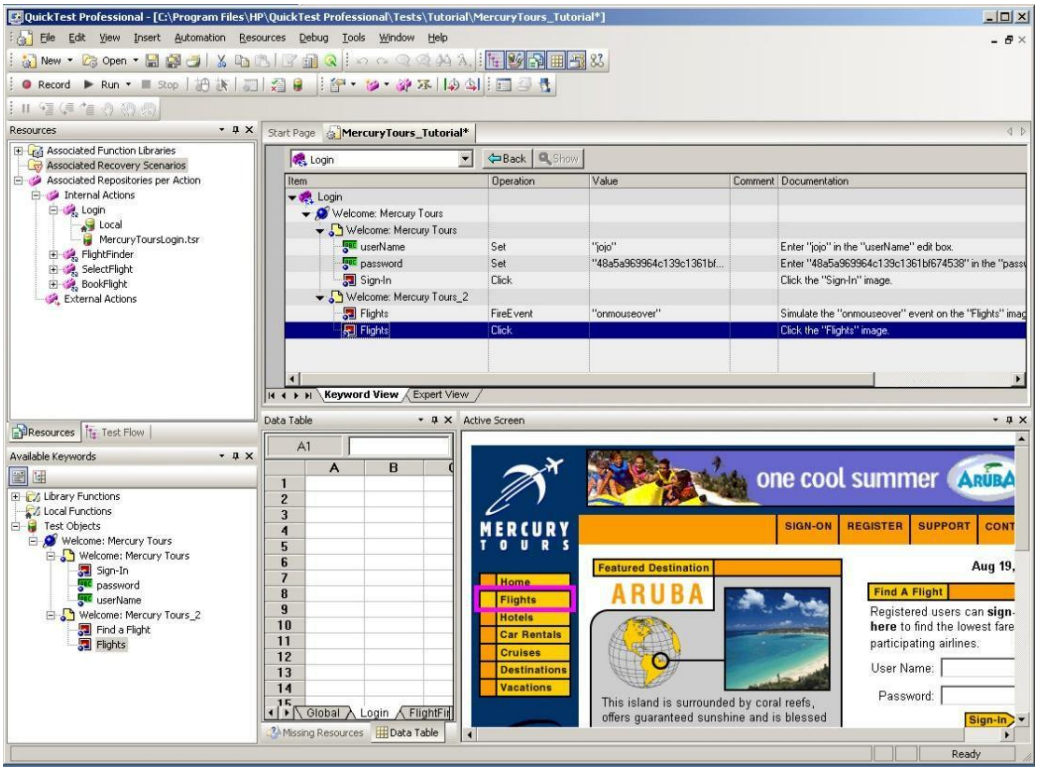


Рисунок 1.2 — Інтерфейс HP QuckTest Professional

HP рекомендує використовувати QTP через інтеграцію з HP Quality Center для зв’язування тестів і вимог, підтримки тестів, керування розгортанням і створення звітів.

На відміну від деяких інших продуктів для автоматизації

функціонального тестування, QTP дозволяє керувати текстом скрипта, який генерується в процесі запису дій користувача, що скорочує час, необхідний для розробки тесту [15].

У QTP інформація про всі об'єкти екранного інтерфейсу зберігається в спеціальному сховищі (Object Storage), яке може здатися новому користувачеві незрозумілим. Недоліки вибору важливих функцій кожного типу об’єктів екранного інтерфейсу можна налаштувати окремо, наприклад, ім’я вікна, ширину таблиці та стовпець таблиці можна вказати порядковим номером. Реалізований механізм порівняння текстових даних за допомогою регулярних виразів [15].

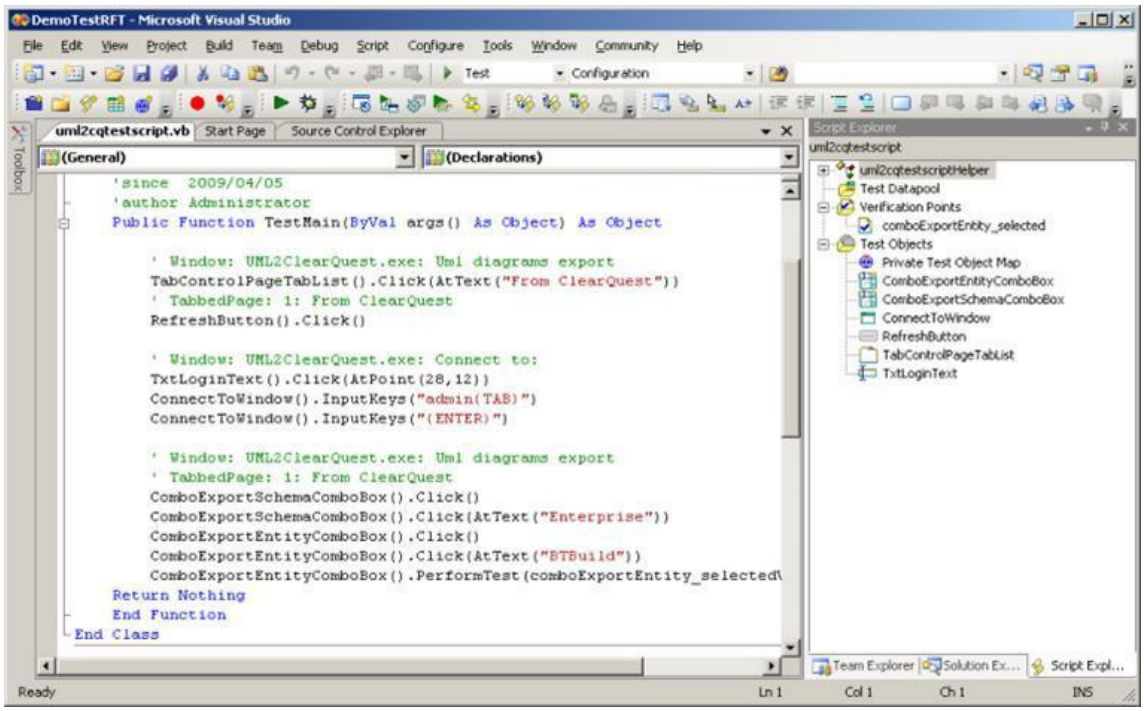


Рисунок 1.3 — Інтерфейс IBM Rational Functional Tester

Найбільш близькою до розглядуваної системи є система Rational Functional Tester (рис. 1.3), що надає засоби автоматизованого тестування, які дозволяють тестувальникам виконувати функціональне тестування, регресійне тестування, тестування інтерфейсу користувача та тестування на основі даних. Функціональне тестування Rational інтегровано з IBM Rational Quality Manager — потужним набором інструментів керування тестуванням,

виявленням дефектів, контролем версій сценаріїв тестування та керування вимогами. Інструменти, включені в цю платформу, прискорюють розробку програм, значно полегшуючи координацію та спілкування всередині команди розробників [16].

Завдяки інструментам, доступним у IBM Rational Quality Manager, уся команда розробників отримує точну інформацію про результати тестування. Зокрема, розробники можуть легко отримати доступ до помилок, знайдених тестувальниками, що допоможе їм легко відтворити знайдені помилки. Крім того, це забезпечить ланцюг управління необхідною інформацією для оцінки рівня ризику на кожному етапі проекту [16].

IBM Rational Quality Manager звільняє тестувальників від ручного відстеження змін вимог, автоматично призначаючи сценарії тестування зміненим вимогам.

Ключові особливості:

— тестування додатків Java (засоби керування J2EE, J2SE, SWT, AWT/JFC);

— тестування веб-додатків (додатки HTML, DHTML, XML, JavaScript, Java);

— написання тестових скриптів Java;

— використання технології ScriptAssure та перевірка змін даних програми;

— інтеграція з раціональними засобами масової комунікації;

— підтримує тестування програм 3270 (zSeries) і 5250 (iSeries) за допомогою розширення Functional Test для термінальних програм;

— повна інтеграція в середовище Eclipse, WebSphere Studio та Rational XDE Developer [16]

1.4 Порівняння існуючих систем

Порівняно з існуючими системами виявилося, що немає системи, яка б

повністю відповідала всім потребам організації автоматизованого процесу тестування і водночас могла б вільно та легко розширюватися.

Система QuickTest Professional від HP є одним із лідерів ринку створення автотестів, але, на жаль, ціна її ліцензії коливається від 8 000 до 10 000 доларів США, що досить високо в умовах ринку українських IT-компаній.

Rational Functional Testing System схожа на QuickTest Professional. Він також є платним, вартість ліцензії становить до 6000 доларів.

Хоча система Selenium IDE безкоштовна, вона добре працює та має відкритий вихідний код, але її потрібно очистити. За допомогою цієї системи ви можете створювати автотести на Java, але виконання цих тестів займе дуже багато часу, і це не варто за поточних темпів розробки та розробки проекту.

1.5 Формування вимог до системи

По-перше, потрібно чітко визначити ключові поняття, які визначають, як працює автоматизована система функціонального тестування. Компетентні вимоги до проекту гарантують, що він буде виконано максимально якісно та матиме всі необхідні функції.

Таким чином, до автоматизованої системи функціонального тестування, що розробляється, висуваються такі вимоги:

— простота створення та підтримки тестів навіть для того, хто раніше не працював у сфері автоматизації тестування, але володіє навичками програмування;

— система має бути оптимізованою, гнучкою та здатною до тестування на різних рівнях від інтеграції до системи;

— проведення тестів на різних платформах і пристроях для перевірки сумісності;

— наявність розвиненої системи звітності, виникає необхідність автоматичного формування різних типів звітів за станом проекту (кількість дефектів, швидкість роботи тощо);

— система повинна складатися з окремих модулів, відповідальних за конкретні питання розробки тестів;

— функціональність системи повинна легко розширюватися.

2 МЕТОДИ ТА ЗАСОБИ ТЕСТУВАННЯ ВЕБ ЗАСТОСУНКІВ

2.1 Дослідження видів тестування

На сьогоднішній день не існує загальноприйнятого визначення «типу тестування програмного забезпечення». Нерідко методи, рівні та навіть методи планування експерименту визначають як тип експерименту. Наприклад, іноді тестування білого ящика, інтеграційне тестування та навіть граничне тестування вважаються типами тестування.

Під час виконання тесту тестувальник зосереджується на певній меті тесту. Існує три види тестування програмного забезпечення залежно від їх цілей [17]:

— функціональне тестування;

— нефункціональне тестування;

— експериментуйте зі змінами;

Звичайно, існує багато типів тестування. Ви можете побачити їх на рисунку нижче:

ПІДТИПИ ВИДІВ ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

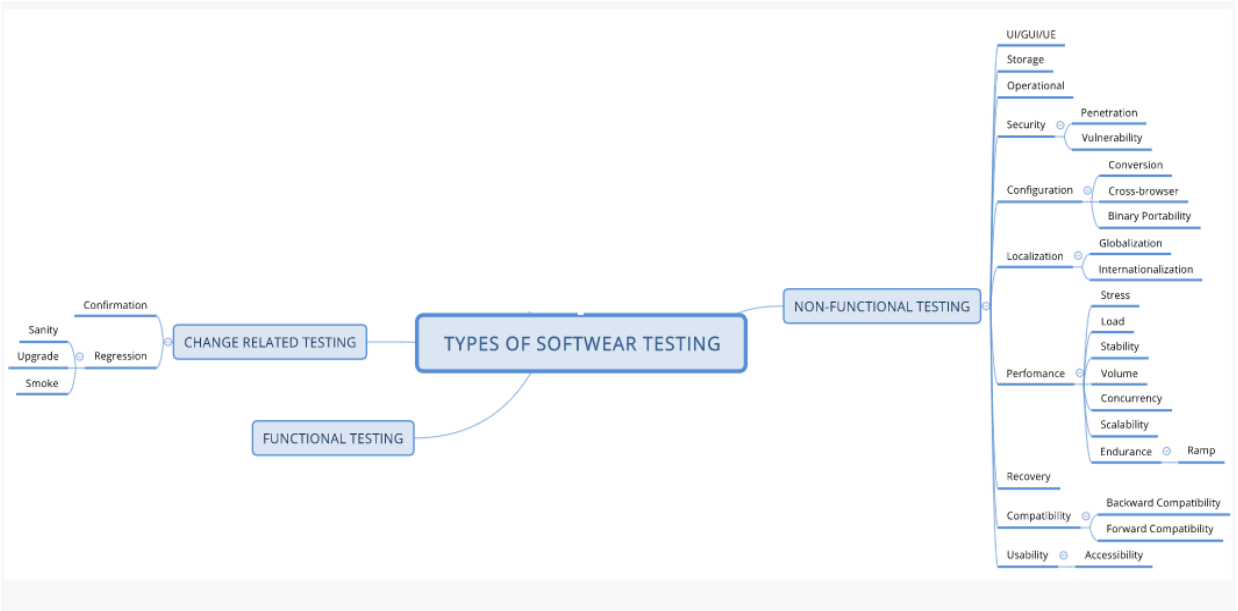


Рисунок 2.1 —

2.1.1 Функціональне тестування

Функціональне тестування перевіряє, чи кожна функція програмного забезпечення працює відповідно до специфікації вимог. Функціональне тестування показує, «що робить машина». Мета цього тесту — перевірити, чи працює система чи ні.

Функціональне тестування можна виконати двома способами: тестування на основі вимог і тестування на основі робочого процесу.

Тестування на основі вимог виконується відповідно до визначених вимог.

Бізнес-тестування виконується на основі знань, що базуються на повсякденному комерційному використанні системи.

Переваги функціонального тестування:

— функціональне тестування імітує реальне використання системи;

— виконується в умовах, наближених до умов замовника;

— функціональне тестування не робить припущень щодо архітектури системи.

— тестування легко реалізувати.

Особливості функціонального тестування:

— є реальна можливість для проведення великої кількості надмірних експериментів;

— логічні помилки в програмному забезпеченні можуть бути пропущені під час функціонального тестування.

Нижче перераховано найпоширеніші види функціонального тестування є:

Тестування безпеки — це тип тестування програмного забезпечення, який виявляє вразливості, загрози та ризики в програмному забезпеченні та запобігає зловмисним атакам хакерів. Метою тестування безпеки є виявлення всіх вразливостей у програмних системах, які можуть призвести до втрати інформації, прибутку та репутації.

Функціональне тестування є формою тестування програмного забезпечення, яка тестує програмну систему на відповідність функціональним вимогам/специфікаціям. Метою функціонального тестування є перевірка

функціональності кожного програмного забезпечення шляхом перевірки відповідного програмного забезпечення та вихідних даних на відповідність функціональним вимогам.

Функціональне тестування в основному включає тестування чорного ящика і не стосується вихідного коду програми. Цей тест перевіряє інтерфейс користувача, API, базу даних, безпеку, зв'язок клієнт-сервер та інші функції програми, що тестується. Тестування можна проводити вручну або за допомогою автоматизації.

Спільне тестування — це функціональний тест, який перевіряє здатність програмного забезпечення взаємодіяти з одним або кількома компонентами чи системами. Тестування взаємодії включає тестування на сумісність (тестування взаємності) та тестування інтеграції.

Тестування на сумісність — це тип тестування програмного забезпечення для перевірки того, що ваше програмне забезпечення може працювати на різних пристроях, операційних системах, програмах, мережевих середовищах або мобільних пристроях [17].

Інтеграційне тестування — це тип тестування, у якому модулі програми логічно об'єднуються та тестуються як група. Типовий програмний проект складається з багатьох програмних модулів, закодованих різними програмістами. Метою цього рівня тестування є виявлення дефектів у взаємодії цих програмних модулів під час інтеграції. Тестування інтеграції зосереджено на перевірці передачі даних між цими модулями.

2.1.2 Нефункціональне тестування

Типи нефункціональних тестів стосуються нефункціональних вимог. Нефункціональне тестування допомагає оцінити готовність системи за різними критеріями, які охоплює функціональне тестування. На відміну від функціонального тесту, він показує, «наскільки добре працює пристрій» [17]. Щоб краще зрозуміти цей тип тестування, нам потрібно розглянути підтипи.

Тестування продуктивності — це процес тестування програмного забезпечення, який використовується для перевірки швидкості роботи програмного забезпечення, часу відгуку, стабільності, надійності, масштабованості та використання ресурсів за певного навантаження. Основною метою тестування продуктивності є виявлення та усунення вузьких місць у продуктивності програмного забезпечення. Тестування продуктивності включає нижченаведені види тестування.

Навантажувальне тестування — це процес нефункціонального тестування програмного забезпечення, під час якого функціональність програмного забезпечення перевіряється під певним очікуваним навантаженням. Він визначає поведінку програмного забезпечення, коли до нього звертаються декілька користувачів одночасно. Метою тестового тестування є покращення вузьких місць продуктивності та забезпечення стабільності та функціональності програмного забезпечення перед розгортанням.

Стрес-тестування — це різновид тестування програмного забезпечення, яке перевіряє стабільність і надійність програмного забезпечення. Метою стрес-тестування є вимірювання надійності та можливостей обробки помилок програмного забезпечення під дуже високим навантаженням, а також переконатися, що програмне забезпечення не виходить з ладу в кризових ситуаціях. Він навіть перевіряє стандартні кінцеві точки та оцінює продуктивність програмного забезпечення в екстремальних умовах.

Тестування стабільності — це форма невпровадженого тестування програмного забезпечення для вимірювання здатності та здатності програмного забезпечення безперервно функціонувати протягом тривалого періоду часу. Метою тестування стабільності є перевірка того, чи програмне забезпечення виходить з ладу або виходить з ладу в будь-який час під час нормального використання.

Тест установки перевіряє забезпечення якості та зосереджується на тому, що клієнти повинні зробити, щоб успішно встановити та налаштувати нове

програмне забезпечення. Це може бути повний, частковий або інсталяційний процес оновлення, і зазвичай виконується інженером з тестування програмного забезпечення разом із менеджером конфігурації.

Юзабіліті-тестування — це тип тестування, який перевіряє, наскільки простим і зручним є програмний продукт. В основному, під час цього тесту перевіряється, що інтерфейс зручний, навігаційний, зрозумілий і чи немає мільйона кроків для виконання простої операції.

Тестування програмного забезпечення відновлення/помилки програмного забезпечення, помилки мережі тощо. це техніка тестування програмного забезпечення, яка перевіряє його здатність уникати таких помилок, як Мета тестування відновлення полягає в тому, щоб визначити, чи можна продовжувати роботу програмного забезпечення після збою або втрати цілісності. Тестування відновлення передбачає повернення програмного забезпечення до визначеного стану цілісності та повторні операції до невдачі.
Тест настройки, тестування конфігурації — це техніка тестування програмного забезпечення, за якої тестуються численні комбінації програмного та апаратного забезпечення для оцінки функціональних вимог програмного забезпечення та визначення оптимальних конфігурацій для роботи програмного забезпечення без будь-яких помилок чи дефектів [4].

2.1.3 Тестування, пов’язане зі змінами

Тестування виправлень виконується, щоб визначити, чи були виправлені раніше виправлені помилки, і виявити помилки, які могли випадково з’явитися в новій версії. Відповідно до цих цілей розрізняють 4 види тестування.

Димове тестування — це техніка тестування програмного забезпечення, яка виконується після встановлення програмного забезпечення, щоб переконатися, що критичне програмне забезпечення працює належним чином. Це робиться без детального функціонального або регресійного тестування. Основна мета димового тестування — відхилити дефектне програмне

забезпечення, щоб команда контролю якості не витрачала час на тестування зламаного програмного забезпечення.

Регресійне тестування визначається як тип тестування програмного забезпечення для підтвердження того, що нещодавні зміни в програмі чи коді не впливають негативно на існуючу функціональність. Це не що інше, як частковий або повний вибір повторно виконаних тестів, щоб переконатися, що існуюча операція працює належним чином.

Це тестування проводиться, щоб переконатися, що нові зміни коду не вплинуть негативно на існуючу функціональність. Це гарантує, що старий код працюватиме й після зміни останнього коду.

Функціональне тестування, також відоме як тестування дезінфекції, — це тип тестування програмного забезпечення, яке виконується після отримання програмного забезпечення, вносячи незначні зміни в код або функції, щоб переконатися, що помилки виправлено та не виникає інших проблем. це змінюється. Мета полягає в тому, щоб визначити, чи запропонована функція працює приблизно так, як очікувалося. Якщо перевірка працездатності завершується невдачею, збірка відхиляється, щоб заощадити час і кошти на ретельніше тестування.

Деякі джерела неправильно трактують, що перевірка на прцездатність та тест на дим —це одне й те саме. Ці види тестів мають «вектори руху» в різних напрямках. На відміну від димового тестування, перевірка працездатності глибоко зосереджена на функції, що перевіряється, тоді як димове тестування широко сфокусовано, щоб охопити якомога більше функцій тестами за короткий проміжок часу [18].

2.2 Методи, технології та специфікаці

2.2.1 Cucumber

Одним із великих викликів автоматизації тестування є підтримка, формалізація та розуміння тестових випадків. Наприклад, якщо в компанії

з’явилася нова людина, потрібно багато часу, щоб зрозуміти, що відбувається під час того чи іншого суду. Полегшити цю проблему допоміг фреймворк Cucumber [19].

Cucumber використовується для написання програмного коду за допомогою методології розробки, керованої поведінкою (BDD). Суть методології полягає в тому, щоб спочатку формалізувати вимоги, а потім на основі цих вимог працювати. У випадку автоматизованої системи тестування цей підхід підходить для формалізації тестових випадків природною мовою, а потім реалізації кожного кроку в програмному коді. Це дозволяє внести прозорість і ясність в сам процес тестування, прискорити роботу і розуміння того, що і як тестується [9]. Файли із задокументованими тестами мають розширення .Feature і зазвичай мають назву тесту, наприклад Login.feature (рисунок 2.2).

```
Feature: Trying To Login
  Scenario: I'm going to loginpage and trying to login
    Given i'm navigating to home page
    Then i'm trying to login as admin
```

Рисунок 2.2 — Приклад кейсу LogIn.Feature

Ключове слово Attribute використовується для формалізації назви тесту, а ключове слово Scenario використовується для швидкого опису самого алгоритму тесту або очікуваного результату.

Ключові слова використовуються для зв’язку офіційного тесту та його опису мовою програмування:Given;

—When;

—Then;

—And.

Ці ключові слова працюють як посилання в Інтернеті, кажучи, що існує метод, пов’язаний з цим кроком у тесті. Cucumber підтримує багато природних мов для створення тестів, включаючи російську, що робить його одним з

найпотужніших фреймворків для досягнення простоти та зрозумілості тестів та іспитів.

Подібним чином у Cucumber є вбудований генератор звітів, який дозволяє вказати, в якій точці виникла перешкода (рисунок 2.3).

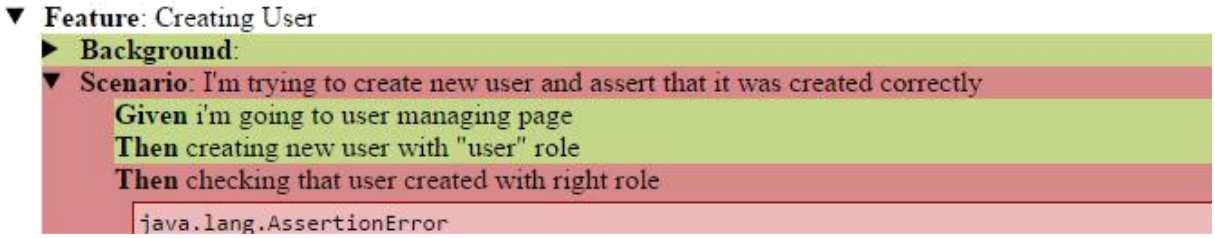


Рисунок 2.3 — Звіт згенерований фреймворком Cucumber

2.2.2 Selenium WebDriver

Для роботи з будь-яким сайтом або веб-програмою використовуються браузер. Браузер — прикладне програмне забезпечення для перегляду веб-сторінок. Перш ніж взаємодіяти з сайтом, потрібно вирішити завдання взаємодії з браузером як таким. На допомогу прийшов Selenium WebDriver [20]. Цей драйвер є ключовим елементом системи і дозволяє імітувати роботу користувача з браузером: клікати на посилання, заповнювати поля, очищати поля, натискати на кнопки, переходити за посиланням і т. д. WebDriver здатний взаємодіяти з більшістю популярних браузерів: InternetExplorer, Mozilla FireFox, Google Chrome і т.д.

2.2.3 Бібліотека тестування jUnit

Було проведено більше досліджень щодо популярних бібліотек тестування, і на їх користь була обрана бібліотека jUnit. Це дозволить вам додати точку порівняння між фактичним і очікуваним результатом у будь-якому місці тесту.

jUnit — це бібліотека для модульного тестування програмного забезпечення на Java [21]. Створений Кентом Беком і Еріком Гаммою, JUnit

належить до сімейства фреймворків xUnit для різних мов програмування, які походять від SUnit Кента Бека для Smalltalk. jUnit створив екосистему розширень – jMock, EasyMock, DbUnit, HttpUnit тощо. jUnit PHP (PHPUnit), C# (NUnit), Python (PyUnit), Fortran (fUnit), Delphi (DUnit, DUnit) Free Pascal (FPCUnit), Perl (Test::Part), C++ (CPPUnit), Flex (FlexUnit)), JavaScript (JSUnit), COS (COSUnit) [21].

jUnit дозволяє перевіряти певні твердження, виражені в тестовому параметрі або тесті на рівність рядків. Приклад перевірки рівності рядка:

— String nameOne = "Готель 1";

— String nameTwo = "Готель 2";

— підтвердити (nameOne.equalsIgnoreCase(nameTwo))

Цей код призведе до помилки твердження (asser-tationError), оскільки ім’я рядка One не дорівнює імені рядка Two.

2.2.4 Система автоматичного складання проекту Maven

В результаті з'явилися деякі програмні засоби, кожен з яких відповідає за певне завдання. Для Java існує система збірки Maven, розроблена спільнотою Apache Software Foundation для автоматичного створення проектів.

Maven використовує мову POM, яка є частиною мови XML, для запису структури проекту. Файли опису проекту містять специфікацію, залежності від інших проектів, окремі етапи процесу побудови проекту та список плагінів, які виконують розклад побудови.

2.3 Мови та автоматизоване середовище тестування

Java — мова програмування загального призначення, що належить корпорації Oracle. Java заснована на принципах об'єктно-орієнтованого програмування. Мова дотримується принципу VORA (напишіть один раз, запустіть будь-де), що приносить багато переваг на різних платформах.

Багато великих корпорацій використовують Java для підтримки своїх внутрішніх систем. Java є найпопулярнішою мовою програмування, яка використовується для автоматизації тестування. 44% клієнтів використовують Java для автоматизованого тестування. Існує багато фреймворків, плагінів і освітніх ресурсів, які підтримують Java для автоматизації тестування, доводячи, що підтримка спільноти є рушійним фактором при виборі мови програмування для автоматизації тестування. Оскільки команди інтегрують інструменти автоматизації тестування з інструментами розробки продуктів, це може пояснити популярність Java для тестування інтерфейсу користувача. Незважаючи на те, що JUnit є популярним фреймворком модульного тестування, кілька фреймворків автоматизованого тестування з відкритим кодом було розроблено з використанням Java. Автоматичне кросбраузерне тестування для веб-продукту (веб-сайт/веб-додаток) можна виконати за допомогою JUnit у поєднанні з Selenium WebDriver.

IntelliJ IDEA має найпотужніші функції завершення коду в Java [22]. Його алгоритм передбачення може точно передбачити, що кодер хоче написати та відтворити, навіть якщо він не знає точної назви класу, члена чи іншого ресурсу.

IntelliJ IDEA — це інтегроване середовище розробки (IDE) для мов JVM, але воно може надавати багато плагінів (поліглот), призначених для підвищення продуктивності розробників. Забезпечуючи інтелектуальне завершення коду, статичний аналіз і маніпулювання кодом, ви можете виконувати рутинні та повторювані завдання, дозволяючи вам зосередитися на красивій стороні розробки програмного забезпечення, яка є не тільки ефективною, але й цікавою. IntelliJ IDEA — це платформа IDE, яка забезпечує стабільну роботу в Windows, macOS і Linux.

IntelliJ IDEA використовується для створення програм на наступних мовах, які можна скомпілювати в байт-код JVM: Java, Kotlin, Scala, Groovy. IntelliJ IDEA забезпечує середовище, орієнтоване на редактор. Він відстежує

ваш контекст і автоматично пропонує інструменти, необхідні для мінімізації ризику переривання процесу розробника [22].

3 РОЗРОБКА МЕРЕЖЕВОЇ СИСТЕМИ ТЕСТУВАННЯ САЙТІВ ГОТЕЛІВ

3.1 Розробка архітектури системи

На сьогоднішній день існують дві основні архітектури побулови систем: монолітна та мікросервісна. Монолітна архітектура дозволяє виконувати швидкі ітерації. Проте вона погано масштабується, важко модифікується, особливо коли окремі компоненти сильно зв’язані між собою та важко оновлюється при виникненні необхідності застосування нових технологій.

Відповідно сьогодні більш популярними стають мікросервіси. Це пов’язано на сам перед з тим, такий підхід має суттєві забезпечує гарну масштабованість та гнучкість. Мікросервісну архітектуру використовують такі компанії як Netflix, Google, Amazon та інші.

Розподіл додатку на кілька сервісів дає змогу прискорити процес розробки та забезпечити легкість підтримання у майбутньому. Кожен мікросервіс може розгортатися окремо, незалежно від інших. Проте при цьому виникають додаткові труднощі, пов’язані з необхідністю налаштувати зв’язки між усіма модулями та базами даних в розподіленій системі.

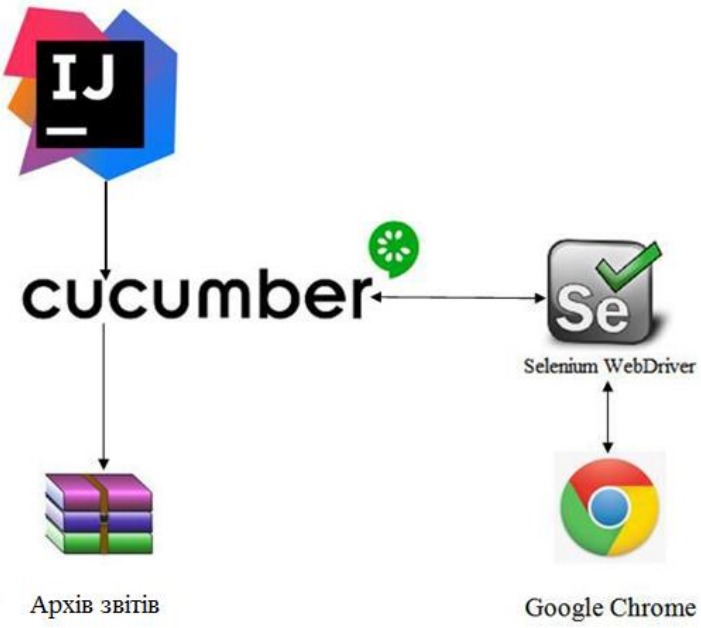


Рисунок 3.1 —АРХІТЕКТУРА СИСТЕМИ

Запропонована системна архітектура, що наведена на рис. 3.1, має імпортувати файл pom.xml для автоконструктора Maven, коли проект вперше відкривається на новій машині в IDE. Створюються зв’язки між компонентами проекту та завантажуються відсутні бібліотеки.

Система реалізована на основі клієнт-серверної архітектури. Як сховище даних вибрано СУБД PostgreSQL. Обмін даними між клієнтською та серверною частинами здійснюється за протоколом REST. Схема основних модулів системи наведена у додатку В. Система є багатопоточною, завдяки чому забезпечується можливість прийому запитів великої кількості користувачів.

Серверна частина поділена на модулі. Першим є модул, що забезпечує тестування. Другий модуль містить описів об’єктів системи, що потім будуть збережені у бази даних. Останніми є модуль, що виконує прийом та обробку запитів, та модуль обробки помилок. Основним з цих модулів є перший модуль, в якому реалізована логіка створення тестових сценаріїв, наборів даних та профілю корисувача. Клієнтська частина утворює інтерфейс системи та дозволяє відправляти запити користувача на сервер.

Діаграма прецедентів (use-case) наведена у додатку Г та описує варіанти використання або набору дій, які система може виконувати під час взаємодії з одним або декількома користувачами системи — акторами. Інженеру доступний весь функціонал системи, зокрема створення тест кейсів та тестових сценаріїв для подальшого тестування обраного проекту. Адміністратору доступні усі функції інженера, а також можливість додавати нових працівників, переглядати інформацію про кожного працівника

3.2 Основні класи проекту

Основними класами в проекті є AcceptanceTest і клас Hooks. Ці класи працюють у парах, щоб забезпечити гнучке та надійне тестування.

Коли виконується клас AcceptanceTest, він запускає структуру прийняття, яка шукає ключові слова Before, Test, перевіряє їх і викликає методи, які

відповідають цим ключовим словам.

На етапі Before нам потрібно очистити або створити папку для звітів і запустити браузер, а потім отримати назву сценарію з файлу .feature і відобразити його в консолі, де почав виконуватися наш сценарій. Якщо одночасно виконується кілька сценаріїв, виконується перевірка для кількох сценаріїв, і якщо ми вже перевірили запис, повторна перевірка не виконується. За ці дії відповідає метод setUp() класу Hooks.3.2.2 Test

На цьому кроці ініціалізується настроюваний об’єкт, який оброблятиме тест і взаємодіятиме з Selenium WebDriver, а тестовий сценарій виконується відповідно до офіційного тестового сценарію.

Після отримання позитивного чи негативного результату тестування WebDriver завершує роботу, а файли звіту генеруються в папці звітів — етап After. На консолі також відображається інформація про завершення тесту

Щоб WebDriver знав, до якого елемента застосувати певну дію, елементи на сторінці мають бути зіставлені за допомогою одного з двох методів: XPATH, CSS.

Отримавши чіткий опис елемента зі сторінки, необхідно створити клас з назвою тестованої сторінки та описати елементи всередині неї (рис. 3.2).

```
@Locate(how = XPATH, using = ".//*[@id='side-menu']/li[3]/ul/li[3]/a")
public PageObject requestNewUser;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[1]/div[2]/input")
public PageObject firstNameRequest;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[2]/div[2]/input")
public PageObject lastNameRequest;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[3]/div[2]/input")
public PageObject emailRequest;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[4]/div[2]/input")
public PageObject companyRequest;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[6]/div/button[3]")
public PageObject requestBtn;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[6]/div/button[1]")
public PageObject cancelBtn;
@Locate(how = CSS, using = ".alert")
public PageObject succsessAlert;
```

Рисунок 3.2 — Приклад опису елементів сторінки

3.3 Опис об'єктів веб-сторінки

3.3.1 XPATH

XPATH — це мова запитів для елементів документа XML. XSLT — це стандарт консорціуму W3C, призначений для організації доступу до частин документа XML у файлах перетворення. XPath розроблено для полегшення навігації в XML через DOM. XPath використовує компактний синтаксис, відмінний від синтаксису XML. У 2007 році була завершена розробка версії 2.0, частини мови XQuery1.0. У грудні 2009 року почалася розробка версії 2.1, яка використовує XQuery1.1 [19] .

Існує зручне розширення для Mozilla Firefox під назвою FirePath (рис. 3.3), щоб знайти XPATH елемента. Це розширення найзручніше у використанні, щоб знайти XPATH потрібного елемента, достатньо натиснути стрілкою миші на кнопку інтерфейсу та клацнути на потрібному елементі на сторінці. У діалоговому вікні відобразиться виділений вихідний код елемента, а в рядку пошуку відобразиться XPATH.

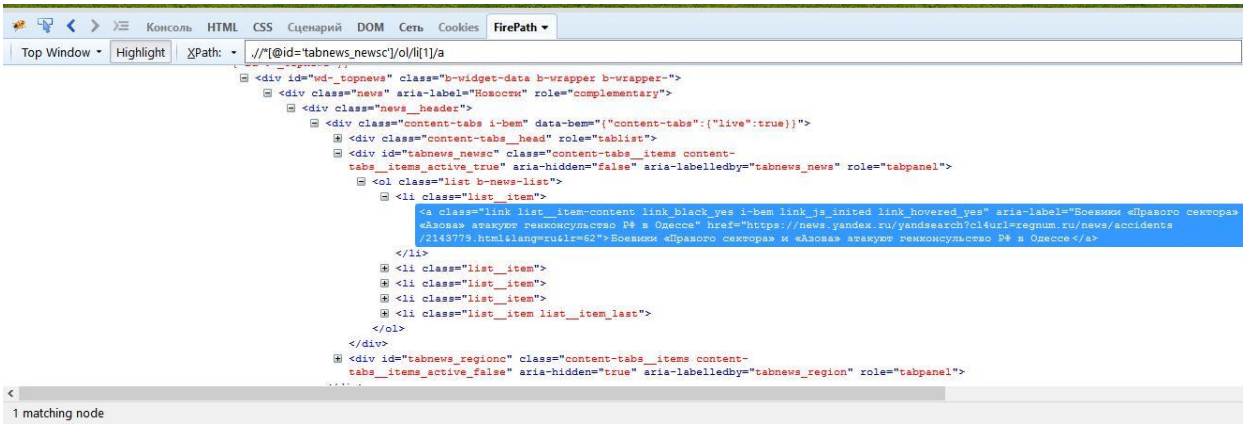


Рисунок 3.3 — Інтерфейс розширення FirePath

3.3.2 CSS

CSS — це формальна мова для опису зовнішнього вигляду документа, написаного за допомогою мови розмітки [20].

Він в основному використовується для опису зовнішнього вигляду веб-сторінок, написаних мовами розмітки HTML і XHTML, але також може бути застосований до будь-якого документа XML, наприклад SVG або XUL [20] .

Щоб вибрати елемент на сторінці за допомогою CSS, вам потрібно створити селектор CSS. Щоб отримати якісний селектор, нам потрібно відкрити інструменти розробника в браузері, навести курсор миші на цікавить нас елемент, у вікні інструментів розробника відобразиться код, який відповідає за розміщення цього елемента на сторінці . . Далі потрібно визначити основні атрибути елемента (батьківський клас, наявність дочірніх класів, наявність настроюваних атрибутів тощо) і створити селектор CSS.

Також є програма SuperSelector для вибору селектора. Ця довідка написана на JavaScript, дуже проста у використанні та дуже гнучка у конфігурації. SuperSelector можна викликати на будь-якому сайті, для цього достатньо створити закладку в браузері і активувати її на потрібній сторінці простим клацанням мишки. Утиліта запуститься, з’явиться інтерфейс, потім Ctrl + клацніть на необхідному елементі сторінки, щоб відкрити селектор CSS. На малюнку 3.4 показано приклад функції SuperSelector.

Рисунок 3.4 — ПРИКЛАД РОБОТИ SUPERSELECTOR

3.4 Додаткові класи для WebDriver

Selenium WebDriver має потужний API, який дозволяє кодувати команди для роботи з браузером.

Базові команди Selenium WebDriver API:

— get ()— метод дозволяє передати URL веб-сторінки;

— findElement() — метод, який дозволяє WebDriver знайти елемент на сторінці за допомогою певного XPATH або CSS;

— click() — метод, який дозволяє клацати будь-який об'єкт на сторінці

— sendKeys() — це метод, який дозволяє вводити будь-який текст у поле введення. Цей метод можна викликати для будь-якого елемента на сторінці, наприклад, імітуючи натискання гарячих клавіш;

— clear()— метод, який дозволяє очистити текстове поле.

Виклик WebDriver за допомогою цих команд робить тестовий код занадто великим і його важко читати, тому було вирішено написати класи та методи, які спрощують взаємодію з API.

WebDriverFactory — цей клас містить методи для ініціалізації драйвера, необхідного певному браузеру.

WebDriverHelper — клас містить методи, які спрощують виклик часто використовуваних функцій WebDriver. Наприклад, метод CurrentWebDriver() скасовує поточний драйвер і виводить таке повідомлення в журнал і консоль. Метод NavigateTo(String url) використовується для переходу до веб-сторінки за допомогою посилання, наданого в змінній String url. Існує також метод disableAskLeavePagePo-pup(), який відповідає за відключення повідомлення з проханням підтвердити, що користувач залишає сторінку, дозволяючи драйверу працювати без перерв.

PageObject () — це клас, який містить методи на основі опису об’єктів на сторінці та дозволяє чекати певного стану сторінки, наприклад, метод waitUntilIs-Clickable () дозволяє чекати, поки елемент на сторінку можна

клацнути, а waitUntilAppears () дозволяє чекати, доки певний елемент не з'явиться на сторінці, дає

BasePage — це клас-контейнер для основних об'єктів, які містяться майже на кожній веб-сторінці (верхній колонтитул, меню, нижній колонтитул тощо), а також методів signIn() і signOut() для входу та виходу з облікового запису користувача облікового запису веб-програми .

3.5 СИСТЕМА ГЕНЕРАЦІЇ ЗВІТІВ З ПРОВЕДЕНИХ ТЕСТІВ

Було додано додаткову систему реєстрації подій для виведення інформації в реальному часі на консоль середовища розробки з можливістю створення файлу, що містить звіти про тести, виконані системою.

Клас LoggerFactory має метод (), який використовується для ініціалізації системи реєстрації подій, створення каталогів, що містять звіти, а також файли журналів у форматі .log (рис. 3.5).

```
16:40:03 INFO: Instantiating Chrome driver
16:40:03 INFO: Looking up for driver driver/chromedriver.exe...
16:40:11 INFO: ->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->
16:40:11 INFO: Running scenario 'creating-user;i'm-trying-to-create-new-user-and-assert-that-it-was-created-correctly
16:40:11 INFO: Navigating to home page
16:40:54 INFO: Attempting to login as 'admin' with password 'qwerty123456'...
16:40:54 INFO: Waiting until element 'usernameField' appears..
16:40:54 INFO: Clicking on 'usernameField'
16:40:54 INFO: Looking for 'usernameField' element
16:40:54 INFO: Sending keys [admin] to 'usernameField'
16:40:54 INFO: Looking for 'usernameField' element
16:40:55 INFO: Clicking on 'passwordField'
16:40:55 INFO: Looking for 'passwordField' element
16:40:55 INFO: Sending keys [qwerty123456] to 'passwordField'
16:40:55 INFO: Looking for 'passwordField' element
16:40:55 INFO: Clicking on 'signInBtnPopup'
16:40:55 INFO: Looking for 'signInBtnPopup' element
16:40:55 INFO: Waiting until element 'usersSideMenu' appears..
16:40:57 INFO: Clicking on 'usersSideMenu'
16:40:57 INFO: Looking for 'usersSideMenu' element
16:40:57 INFO: Waiting until element 'manageUsersSubMenu' appears..
16:40:57 INFO: Clicking on 'manageUsersSubMenu'
16:40:57 INFO: Looking for 'manageUsersSubMenu' element
16:41:03 INFO: Waiting until element 'addBtn' appears..
16:41:03 INFO: Waiting until element 'addBtn' is clickable..
16:41:03 INFO: Waiting until element 'addBtn' is enabled..
16:41:03 INFO: Waiting until attribute 'disabled' of element 'addBtn' doesn't exist..
16:41:03 INFO: Clicking on 'addBtn'
16:41:03 INFO: Looking for 'addBtn' element
16:41:03 INFO: Waiting until element 'fullName' appears..
16:41:04 INFO: Sending keys [Jeremy Oyster] to 'fullName'
16:41:04 INFO: Looking for 'fullName' element
16:41:04 INFO: Clearing input of 'email'
16:41:04 INFO: Looking for 'email' element
16:41:04 INFO: Sending keys [gioxproo@yomail.info] to 'email'
16:41:04 INFO: Looking for 'email' element
```

Рисунок 3.5 — Приклад звіту про проведення тесту

Клас LoggerUtils містить метод zipTestLogs(), за допомогою якого відбувається архівація звітів про проведений тест і розміщення архіву (рисунок 3.6) в директорії містить дані звіти. Адреса директорії встановлюється в конфігураційному файлі. Також цей клас містить метод cleanUpLogsDir(), який служить для відчистки директорії зі звітами і метод closeLoggerHandlers() який служить для завершення роботи системи запису подій.

Клас RecordFormatter містить формат дати, який буде записано в Звіти.

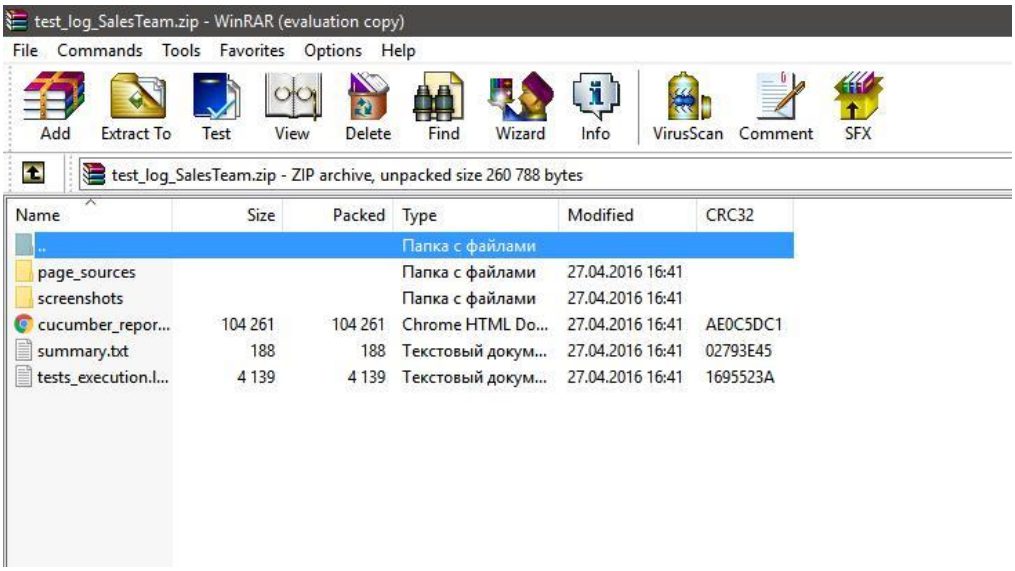


Рисунок 3.6 — Приклад архіву, отриманого за допомогою zipTestLogs()

3.6 Класи збору додаткової інформації

Реалізовано клас TestUtils, який містить такі методи для отримання додаткової інформації про веб-сторінку.

SaveBrowserScreenshot(String outputDir, String fileName) — це метод, який дозволяє зберегти знімок вікна веб-переглядача до папки, заданої змінною outputDir, ім’я якої є змінною fileName. Якщо ім’я файлу не вказано, типовим ім’ям є error.png.

SaveCurrentPageSource(String outputDir, String fileName) – метод, який зберігає вихідний код сторінки в папку зі змінною fileName, папку, адресу якої

вказує змінна outputDir.

Sleep(int seconds) — цей метод дозволяє зупинити хід тесту на необхідну кількість секунд, які проходять цілочисельну змінну. Через певний проміжок часу тест продовжиться.

3.7 Файли конфігурації

Для зручності налаштування системи було вирішено експортувати основну інформацію, таку як тип накопичувача, адреса папки для зберігання звітів, адреса папки для зберігання зображень вікна браузера та адреса папки. щоб зберегти вихідний код сайту в окремому конфігураційному файлі, що робить тестовий код більш читабельним і легко модифікованим. Наприклад, якщо вам потрібно змінити тип драйвера, достатньо змінити змінну "driver.type", якщо потрібно зберігати скріншоти в папці для зберігання скріншотів, достатньо отримати значення скріншотів. змінна.

Крім того, клас CucumberConfiguration створюється для користувацької конфігурації, яка має вказувати змінну клею, що містить шлях до кроків, кодованих за допомогою необхідного формалізованого тестового контейнера та класу Hooks. формат отриманого звіту такий самий, як указана змінна формату.

```
package masterov.aspirity.ui_tests;

import cucumber.api.CucumberOptions;

@CucumberOptions(
        features = "src/test/resources/features/TryingToGoToPageByURL.feature",
        //features = "src/test/resources/features/LogIn.feature",
        //features = "src/test/resources/features/CreatingQuote.feature",
        //features = "src/test/resources/features/CreatingUser.feature",
        //features = "src/test/resources/features/RequestingNewUser.feature",
        //features = "src/test/resources/features/CreatingQuoteAfterAcceptingNewUser.feature",
        format = {"html:target/cucumber"},
        glue = {"masterov.aspirity.ui_tests.cucumber"}
)
public class CucumberConfiguration {
}
```

Рисунок 3.7 — Приклад змісту класу CucumberConfiguration

3.8 Структура проекту

На рисунку 3.8 зображено структуру отриманого проекту.

Папка src містить усі файли класів, необхідні для належного функціонування системи.

Пакет conf містить класи, які дозволяють збирати інформацію про тестове середовище та налаштовувати тестову систему на основі отриманої інформації.

У папці Onion є класи, які містять інструкції на основі формалізованих кроків, розташовані в папці test/resources/features, а також у класі Hooks.

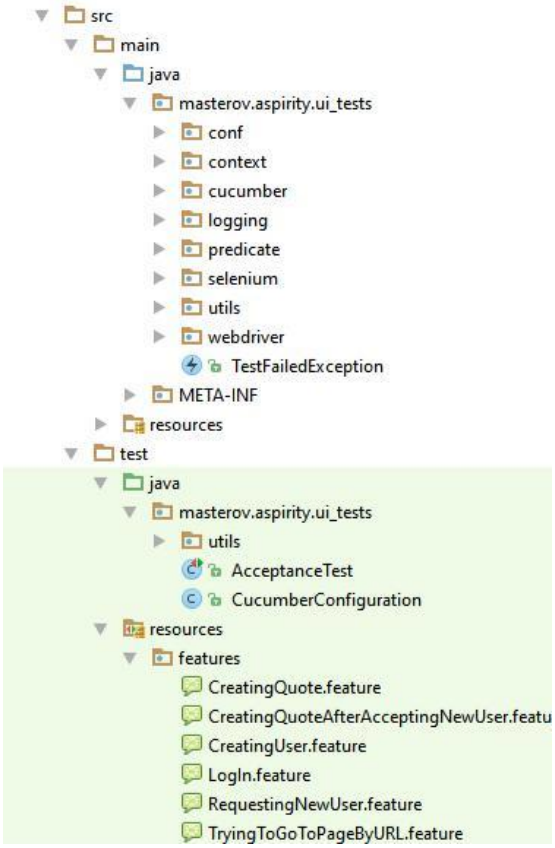


Рисунок 3.8 — СТРУКТУРА ПРОЕКТУ

Папка specifics містить класи, які дозволяють тестовій системі розуміти анотовані об’єкти тестованих веб-сторінок, а також класи, які містять методи, які дозволяють перевіряти існування певних елементів тестової веб-сторінки на основі API WebDriver. .

Пакет Selenium містить описи об’єктів із перевірених веб-сайтів, а також базові класи для опису об’єктів веб-сайту PageObject і BasePage.

Папка журналу містить класи, відповідальні за збір і виведення тестових даних.

Пакет WebDriver містить початкові класи Selenium WebDriver, а також навчальні посібники, які спрощують роботу з WebDriver.

4 ТЕСТУВАННЯ СИСТЕМИ ОЦІНЮВАННЯ ГОТЕЛЬНОГО БІЗНЕСУ

4.1 Створення тестового сценарію

Як перший крок, ми розглянемо, де використовується Selenium і як це написано в коді. На рисунку. 4.1. Ви можете побачити сценарій, який WebDriver використовує для взаємодії з елементами сторінки та описує очікування WebDriverWait.

Розглянемо клас BasePage, де оголошено веб-драйвер і сервер.

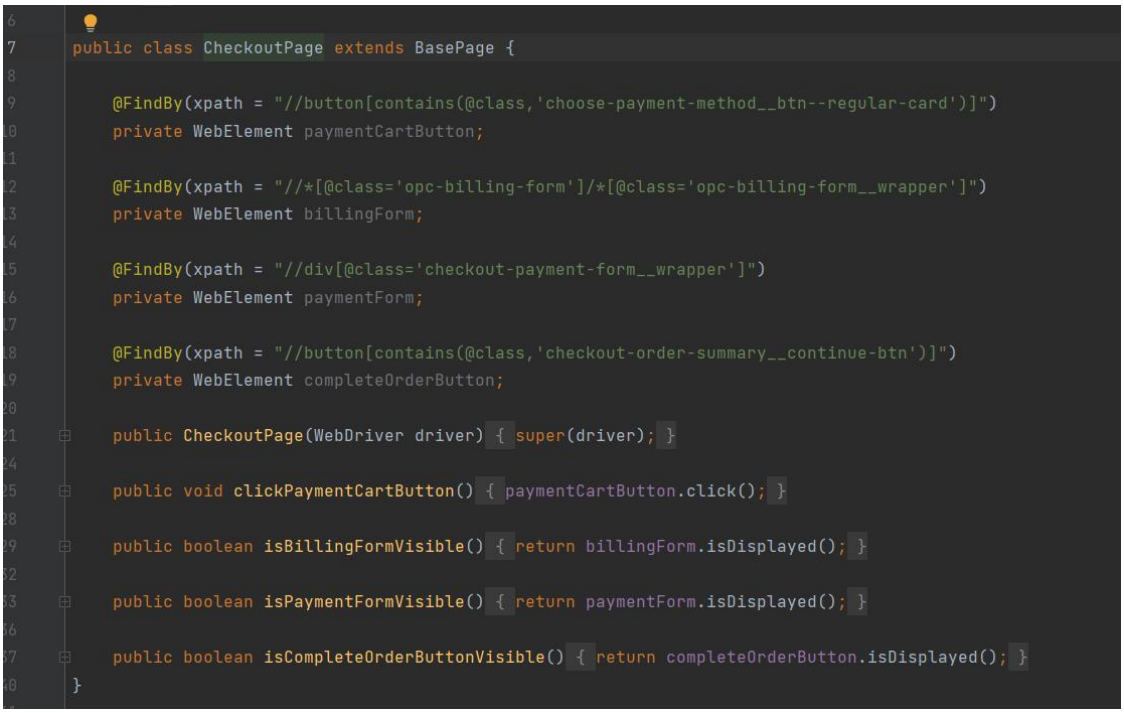
```
public class BasePage {

    protected WebDriver driver;

    public BasePage(WebDriver driver) {
        this.driver = driver;
        PageFactory.initElements(driver, page: this);
    }

    public void waitForPageLoadComplete(long timeToWait) {
        new WebDriverWait(driver, timeToWait).until(
                webDriver -> ((JavascriptExecutor) webDriver).executeScript( s: "return document.readyState").equals("complete"));
    }

    public void waitForAjaxToComplete(long timeToWait) {
        new WebDriverWait(driver, timeToWait).until(
                webDriver -> ((JavascriptExecutor) webDriver).executeScript( s: "return window.jQuery != undefined && jQuery.active == 0;"));
    }

    public void waitForAjaxToCompletePdp(long timeToWait) {
        new WebDriverWait(driver, timeToWait).until(
                webDriver -> ((JavascriptExecutor) webDriver).executeScript( s: "return window.jQuery != undefined && jQuery.active <=2;"));
    }

    public void waitVisibilityOfElement(long timeToWait, WebElement element) {
        WebDriverWait wait = new WebDriverWait(driver, timeToWait);
        wait.until(ExpectedConditions.visibilityOf(element));
    }
}
```

Рисунок 4.1 — Клас BasePage: оголошення веб драйвера та вейтерів

Щоб все це запрацювало, потрібно завантажити поточну версію драйвера, помістити його в папку проекту і зв'язати в цьому класі. Але зараз технології просунуті, і ми можемо замінити всі ці дії цими стрічками у файлі POM

```
        </dependency>
        <dependency>
            <groupId>io.github.bonigarcia</groupId>
            <artifactId>webdrivermanager</artifactId>
            <version>4.3.1</version>
        </dependency>
    </dependencies>
```

Рисунок 4.2 — Опис автоматизованого підключення Webdriver через POM файл



```
public class CheckoutPage extends BasePage {

    @FindBy(xpath = "//button[contains(@class,'choose-payment-method__btn--regular-card')]")
    private WebElement paymentCartButton;

    @FindBy(xpath = "//*[@class='opc-billing-form']/*[@class='opc-billing-form__wrapper']")
    private WebElement billingForm;

    @FindBy(xpath = "//div[@class='checkout-payment-form__wrapper']")
    private WebElement paymentForm;

    @FindBy(xpath = "//button[contains(@class,'checkout-order-summary__continue-btn')]")
    private WebElement completeOrderButton;

    public CheckoutPage(WebDriver driver) { super(driver); }

    public void clickPaymentCartButton() { paymentCartButton.click(); }

    public boolean isBillingFormVisible() { return billingForm.isDisplayed(); }

    public boolean isPaymentFormVisible() { return paymentForm.isDisplayed(); }

    public boolean isCompleteOrderButtonVisible() { return completeOrderButton.isDisplayed(); }
}
```

Рисунок 4.3. — Клас Checkout Page

Це дозволяє автоматично отримати поточну версію веб-драйвера та взаємодіяти зі сторінкою без додаткових зусиль.

Тепер, коли веб-драйвер встановлено, настав час зрозуміти, що робити далі. На даний момент популярним є PFM, тому ми залишимо його. Що він робить і чим може нам допомогти?

```
public class HomePage extends BasePage {

    @FindBy(xpath = "//header[contains(@class, 'global-header global-header--sticky') or @class='page-header']")
    private WebElement header;

    @FindBy(xpath = "//footer")
    private WebElement footer;

    @FindBy(xpath = ".//a[@class='global-header__main-bar__utility-nav__user-cart__link']")
    private WebElement cartIcon;

    @FindBy(xpath = ".//a[contains(@class, 'header-top-bar__input__language')]/span")
    private WebElement languageButton;

    @FindBy(xpath = "//button[contains(@class,'enterprise-account__button_sign-in')]")
    private WebElement signInButton;

    @FindBy(xpath = "//button[contains(@class, 'enterprise-account__button_register')]")
    private WebElement registerButton;

    @FindBy(xpath = "//div[@class='gigya-screen-dialog-main']")
    private WebElement signInPopup;

    @FindBy(xpath = ".//input[@name='username'][@placeholder='Email *']")
    private WebElement emailField;
```

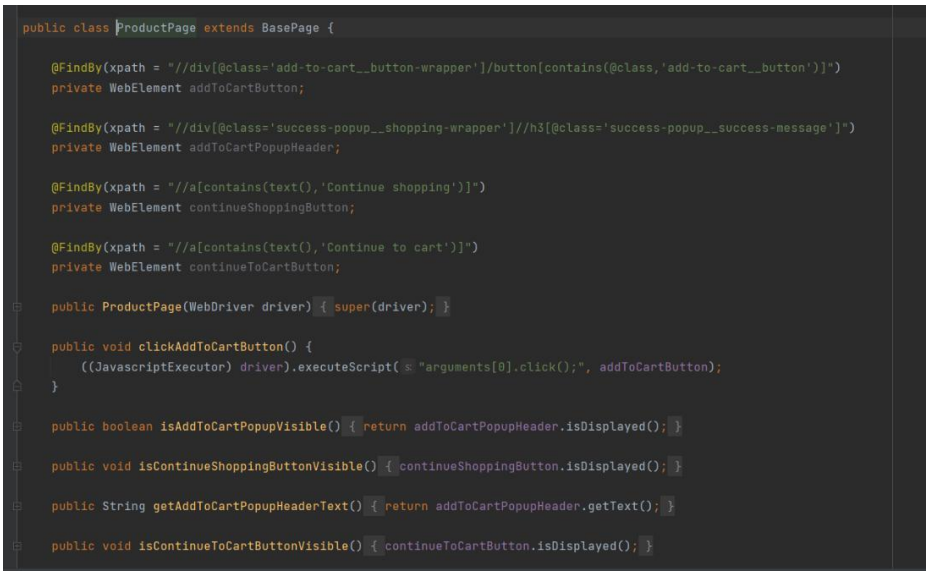
Рисунок 4.4 — Клас HomePage: опис локаторів

Завдяки цьому всі функції та всі елементи сторінки, з якими ми взаємодіємо, пов’язані зі схожими класами сторінок. Саме там відбуваються всі взаємодії (в методах) і пошук або відкриття необхідних елементів. Ось приклади уроків з нашої сторінки.

Локатор — це команда, яка повідомляє Selenium, які елементи графічного інтерфейсу (скажімо, текстове поле, кнопки, прапорці тощо) потрібні для роботи. Визначення правильних елементів графічного інтерфейсу є необхідною умовою для створення сценарію автоматизації.

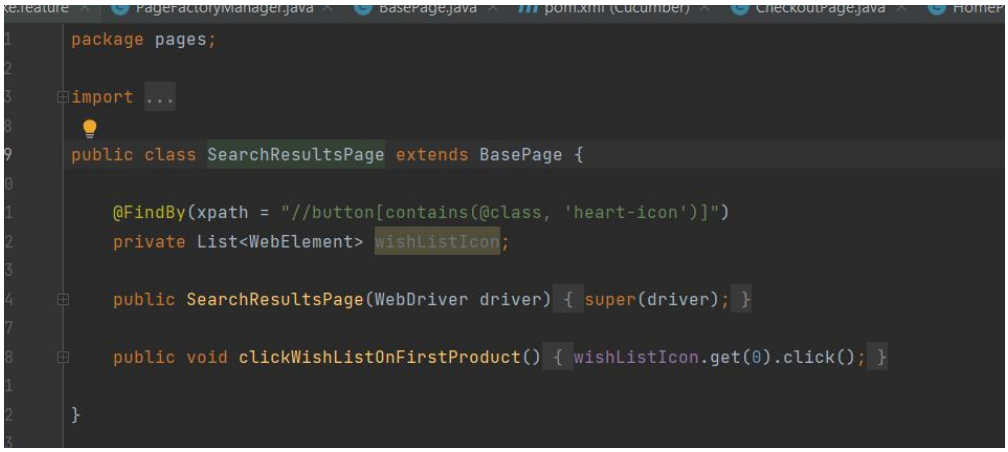
```
    public WebElement getSignInPopup() { return signInPopup; }

    public void clickSignInPopupCloseButton() {
        ((JavascriptExecutor) driver).executeScript( s: "arguments[0].click()", signInPopupCloseButton);
    }

    public void clickStoreButton() { storeButton.click(); }

    public boolean isStorePopupVisible() { return storePopup.isDisplayed(); }

    public void isSearchFieldVisible() { searchField.isDisplayed(); }

    public void clickCartButton() { cartIcon.click(); }

    public void clickLanguageButton() { languageButton.click(); }

    public void enterTextToSearchField(final String searchText) {
        searchField.clear();
        searchField.sendKeys(searchText);
    }

    public void clickSearchButton() { searchButton.click(); }

    public WebElement getWishListProductsCount() { return wishListProductsCount; }

    public String getAmountOfProductsInWishList() { return wishListProductsCount.getText(); }

}
```

Рисунок 4.5 — Клас HomePage: опис методів



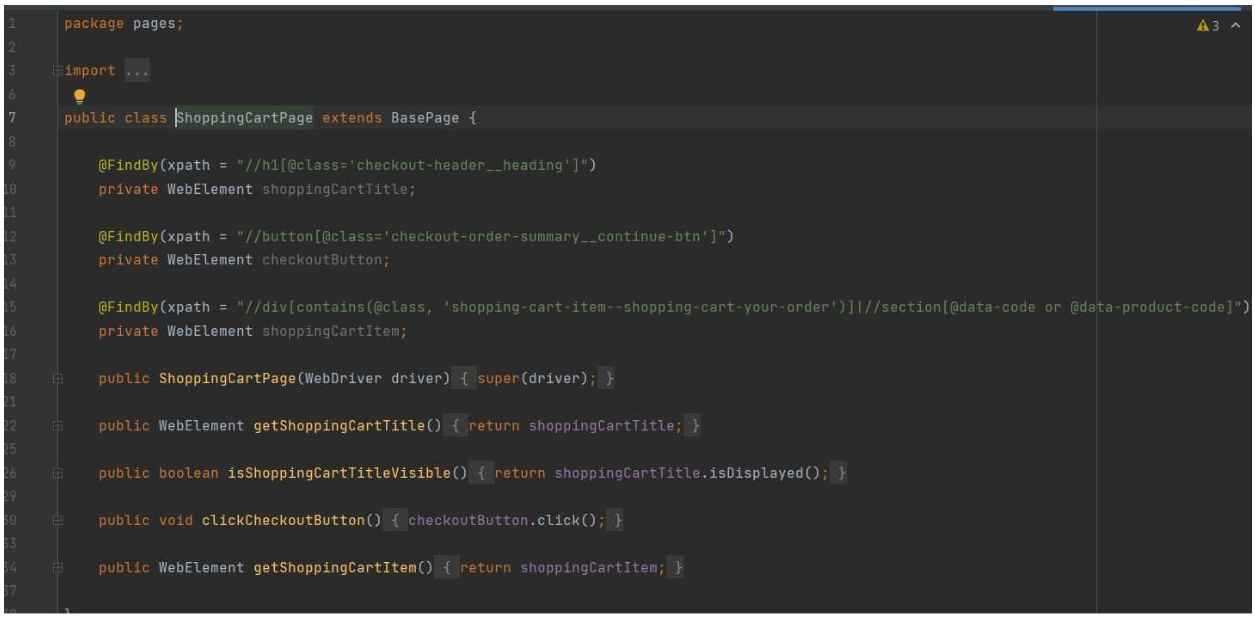
```
public class ProductPage extends BasePage {

    @FindBy(xpath = "//div[@class='add-to-cart__button-wrapper']/button[contains(@class,'add-to-cart__button')]")
    private WebElement addToCartButton;

    @FindBy(xpath = "//div[@class='success-popup__shopping-wrapper']//h3[@class='success-popup__success-message']")
    private WebElement addToCartPopupHeader;

    @FindBy(xpath = "//a[contains(text(),'Continue shopping')]")
    private WebElement continueShoppingButton;

    @FindBy(xpath = "//a[contains(text(),'Continue to cart')]")
    private WebElement continueToCartButton;

    public ProductPage(WebDriver driver) { super(driver); }

    public void clickAddToCartButton() {
        ((JavascriptExecutor) driver).executeScript( s: "arguments[0].click();", addToCartButton);
    }

    public boolean isAddToCartPopupVisible() { return addToCartPopupHeader.isDisplayed(); }

    public void isContinueShoppingButtonVisible() { continueShoppingButton.isDisplayed(); }

    public String getAddToCartPopupHeaderText() { return addToCartPopupHeader.getText(); }

    public void isContinueToCartButtonVisible() { continueToCartButton.isDisplayed(); }
```

Рисунок 4.6 — Клас ProductPage: опис методів і локаторів



```java
package pages;

import ...

public class SearchResultsPage extends BasePage {

    @FindBy(xpath = "//button[contains(@class, 'heart-icon')]")
    private List<WebElement> wishListIcon;

    public SearchResultsPage(WebDriver driver) { super(driver); }

    public void clickWishListOnFirstProduct() { wishListIcon.get(0).click(); }

}
```

Рисунок 4.7 — Клас SearchResultPage

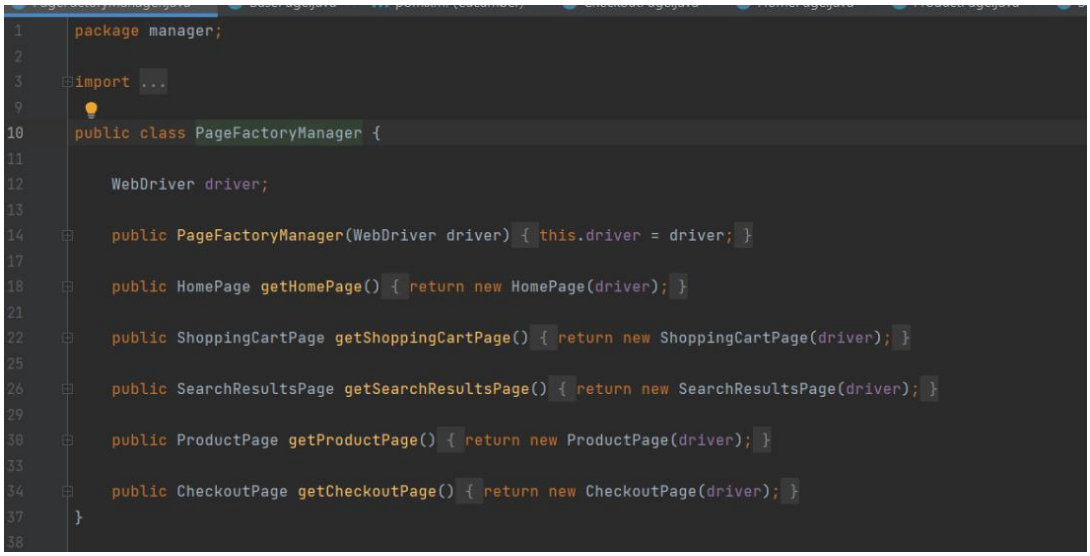


```java
package pages;

import ...

public class ShoppingCartPage extends BasePage {

    @FindBy(xpath = "//h1[@class='checkout-header__heading']")
    private WebElement shoppingCartTitle;

    @FindBy(xpath = "//button[@class='checkout-order-summary__continue-btn']")
    private WebElement checkoutButton;

    @FindBy(xpath = "//div[contains(@class, 'shopping-cart-item--shopping-cart-your-order')]|//section[@data-code or @data-product-code]")
    private WebElement shoppingCartItem;

    public ShoppingCartPage(WebDriver driver) { super(driver); }

    public WebElement getShoppingCartTitle() { return shoppingCartTitle; }

    public boolean isShoppingCartTitleVisible() { return shoppingCartTitle.isDisplayed(); }

    public void clickCheckoutButton() { checkoutButton.click(); }

    public WebElement getShoppingCartItem() { return shoppingCartItem; }
```

Рисунок 4.8 — Клас ShoppingCartPage

Видно, що Find часто використовується для анотації, що є лише чіткою ознакою нашого шаблону Page Factory.

Тепер, коли є заготовки, потрібно мати спосіб отримати їх конкретні приклади та взаємодіяти з ними. У цьому нам допомагає спеціальний клас PageFactoryManager. Цей клас використовується як конструктор для повернення нам готових копій потрібних сторінок, в яких ми вже навчаємо потрібним методам і взаємодіємо зі сторінкою. Все це можна побачити з методів нашого класу менеджера.



```
package manager;

import ...

public class PageFactoryManager {

    WebDriver driver;

    public PageFactoryManager(WebDriver driver) { this.driver = driver; }

    public HomePage getHomePage() { return new HomePage(driver); }

    public ShoppingCartPage getShoppingCartPage() { return new ShoppingCartPage(driver); }

    public SearchResultsPage getSearchResultsPage() { return new SearchResultsPage(driver); }

    public ProductPage getProductPage() { return new ProductPage(driver); }

    public CheckoutPage getCheckoutPage() { return new CheckoutPage(driver); }
}
```

Рисунок 4.9 — Клас PageFactoryManager

Даний клас використовуватиметься як білдер для отримання готових еземплярів бажаних сторінок, що дозволить перейти безпосередньо до тестування.

Після як визначені готові сторінки з необхідною функціональністю та спосіб їх повернення, ми можемо почати створювати тести за допомогою нашої технології BDD. Є два способи зробити це — перший — негайно почати писати тестову логіку: файл .feature. Далі описано кроки в спеціальному класі для специфікатора, або спочатку можна описати кроки класу, а потім створити файл .feature на їх основі. Спосіб 2 є більш зручним, оскільки, не втрачається загальний огляд зображення. Отже, розглянемо перший спосіб.

За допомогою ключових слів, що використовуються в технології BDD створимо Featiure file як це показана на рисунку 4.10. Після створення файлу функцій нам потрібно підключити наші кроки до сторінок, створивши додатковий клас, який визначає кроки (рис. 4.11).

Це розроблюваний клас, і анотації в останніх двох методах точно відповідають назвам кроків у файлі функції. Дуже хороший спосіб — клацнути правою кнопкою миші на створеному кроці у файлі функцій і вибрати «нове визначення кроку» (рисунок 4.12).

```
Feature: Smoke
  As a user
  I want to test all main site functional
  So that I can be sure that site works correctly

  Scenario Outline: Check add product to wishlist
    Given User opens '<homePage>' page
    And User checks search field visibility
    When User makes search by keyword '<keyword>'
    And User clicks search button
    And User clicks wish list on first product
    Then User checks that amount of products in wish list are '<amountOfProducts>'

    Examples:
      | homePage                          | keyword | amountOfProducts |
      | https://www.canadiantire.ca/en.html | cake    | 1                |

  Scenario Outline: Check site main functions
    Given User opens '<homePage>' page
    And User checks header visibility
    And User checks footer visibility
    And User checks search field visibility
    And User checks cart visibility
    And User checks that language switcher is '<languageSwitcher>'
    And User checks register button visibility
    And User checks sign in button visibility
    When User clicks 'Sign In' button
    And User checks email and password fields visibility on sign in popup
    And User closes sign in popup
    And User opens store popup
    And User checks that store popup visible
    And User opens shopping cart
```

Рисунок 4.10 — Feature File

```
25
26      private static final long DEFAULT_TIMEOUT = 60;
27      WebDriver driver;
28      HomePage homePage;
29      ShoppingCartPage shoppingCartPage;
30      SearchResultsPage searchResultsPage;
31      ProductPage productPage;
32      CheckoutPage checkoutPage;
33      PageFactoryManager pageFactoryManager;
34
35      @Before
36      public void testsSetUp() {
37          chromedriver().setup();
38          driver = new ChromeDriver();
39          driver.manage().window().maximize();
40          pageFactoryManager = new PageFactoryManager(driver);
41      }
42
43      @Given("User opens {string} page")
44      public void openPage(final String url) {
45          homePage = pageFactoryManager.getHomePage();
46          homePage.openHomePage(url);
47      }
48
49      @And("User checks header visibility")
50      public void checkHeaderVisibility() {
51          homePage.waitForPageLoadComplete(DEFAULT_TIMEOUT);
52          homePage.waitForAjaxToComplete(DEFAULT_TIMEOUT);
53          homePage.isHeaderVisible();
54      }
55
```

Рисунок 4.11 — Клас з реалізацією кроків

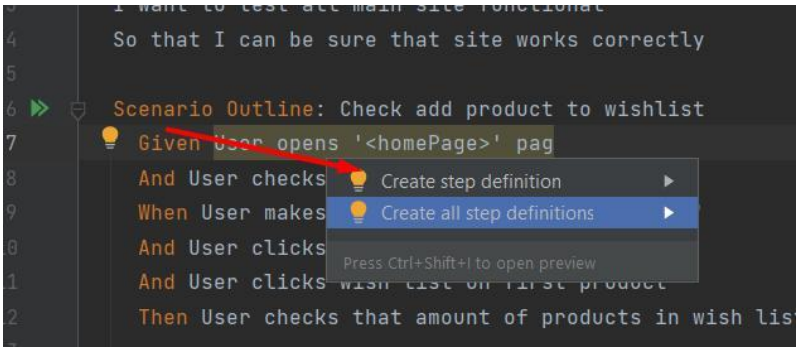


Рисунок 4.12 — Створення нового визначення кроку

Після цього вам буде запропоновано вибрати клас із crocampa, тоді в цьому класі буде створено метод, подібний до нашого кроку у файлі функцій. Це рекомендується, щоб уникнути помилок.

Тепер увесь код написаний, і все, що потрібно зробити, це запустити автоматизовані тести та переконатися, що все працює належним чином.

4.2 Запуск автотесту

Для проведення тестів вам потрібен браузер, завантажений проект і середовище розробки Intellij IDEA для якісного графічного інтерфейсу. Відкриваємо проект в Intellij IDEA, натиснувши відповідну кнопку в нашому середовищі: Файл -> Відкрити (рисунок 4.13).

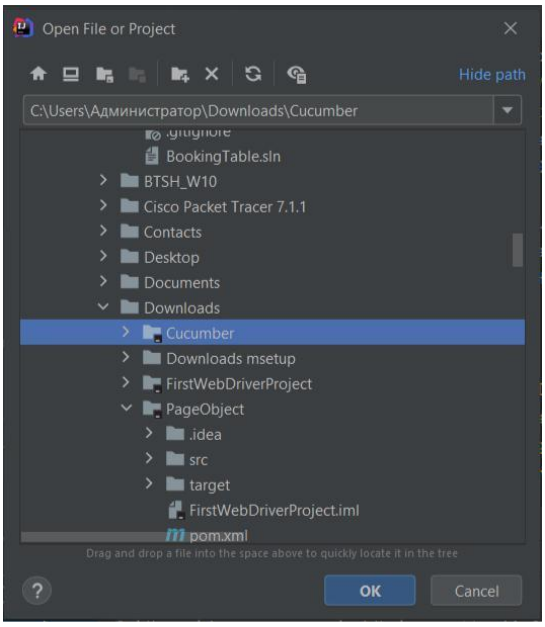


Рисунок 4.13 — Вікно вибору проекту

Потім вибирається фіч-файл та запускаються усі сценарії, кожний з яких є набором тестів. Запуск відбувається за натисканням на «зелені трикутники» як зображено на рисунку 4.14.

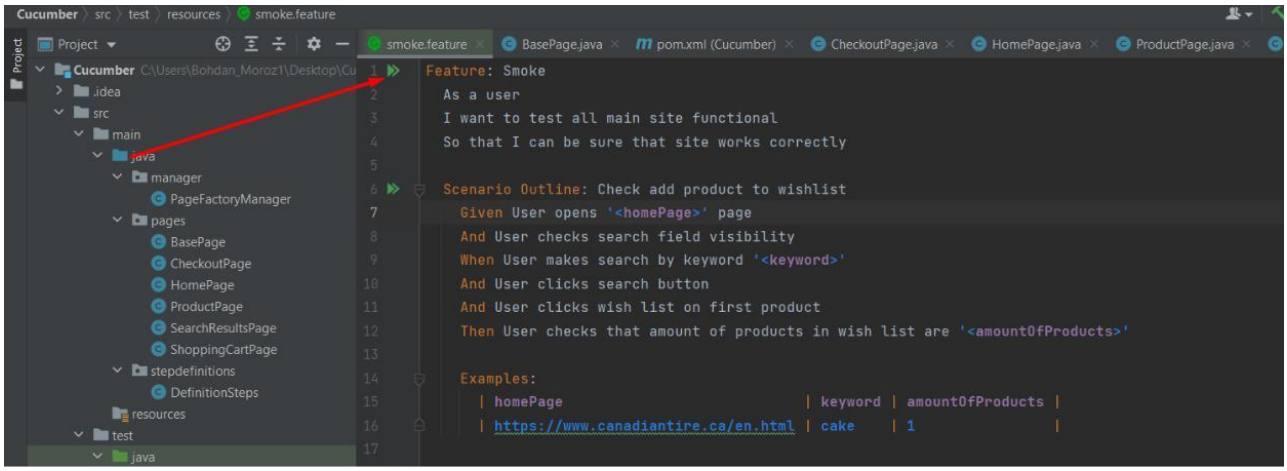


Рисунок 4.14 — Запуск всіх сценаріїв

Якщо потрібно запустити лише один з вибраних тестів, то він запускається Scenario (рисунок 4.15).

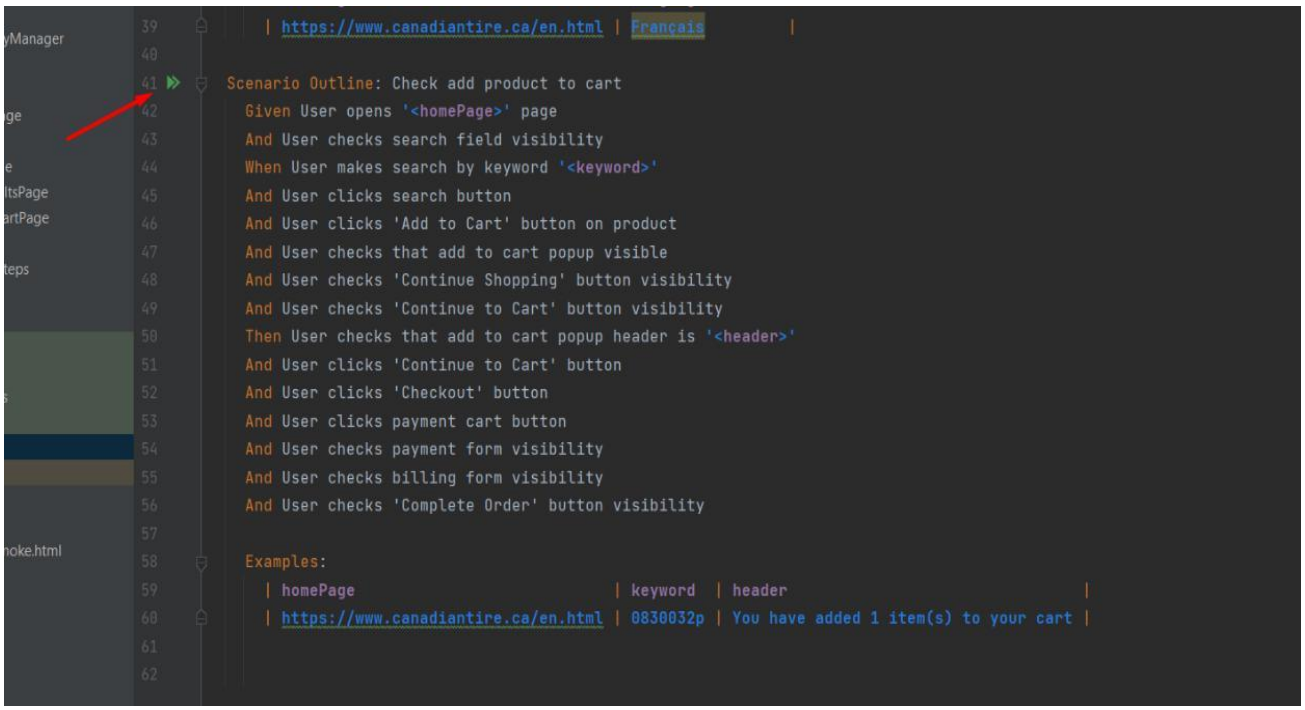


Рисунок 4.15 — Запуск одного сценарію

На завершення перевіряється результат тестування. Якщо результат

успішний, як це зображено на рис. 4.16, можна перейти до формування звітів. При неуспішному результаті тестування необхідно проаналізувати причини. Пошук причини негативного результату можна здійснити через перегляд стеку викликів або покрокового виконання тестового сценарію у Debug.

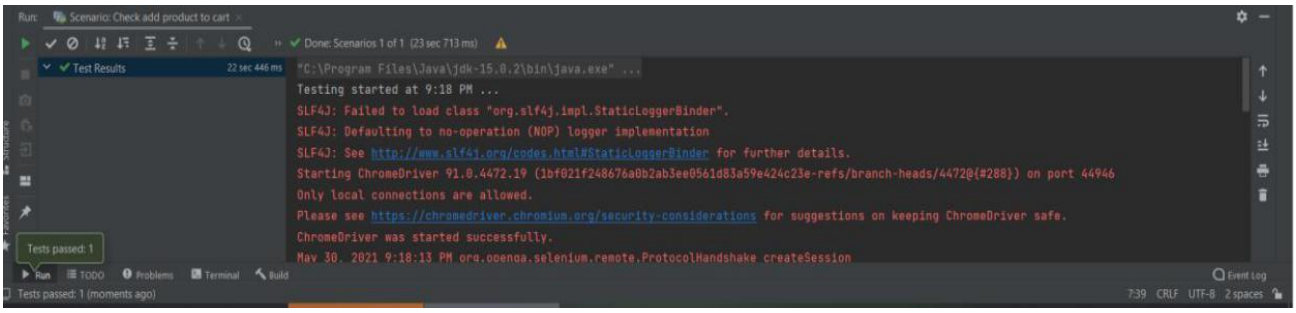


Рисунок 4.16 — Повідомлення про успішно пройдені тести

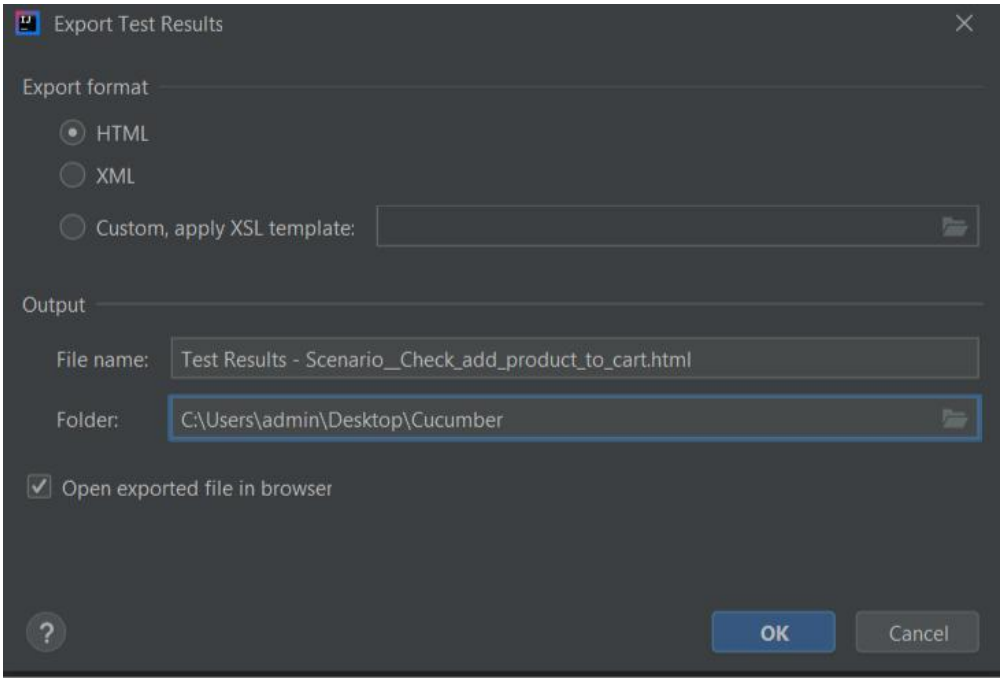


Рисунок 4.17 — Вибір місця для розміщення звіту та його формат

Успішне завершення тесту дозволяє у подальшому обробити та проаналізувати його результати. Для створення звіту потрібно вибрати його бажаний формат та місце на диску для його (рисунок 4.17). Найзручнішим є збереження звіту у html форматі, що дозволяє переглядати його у будь-якому браузері (рисунок 4.18).

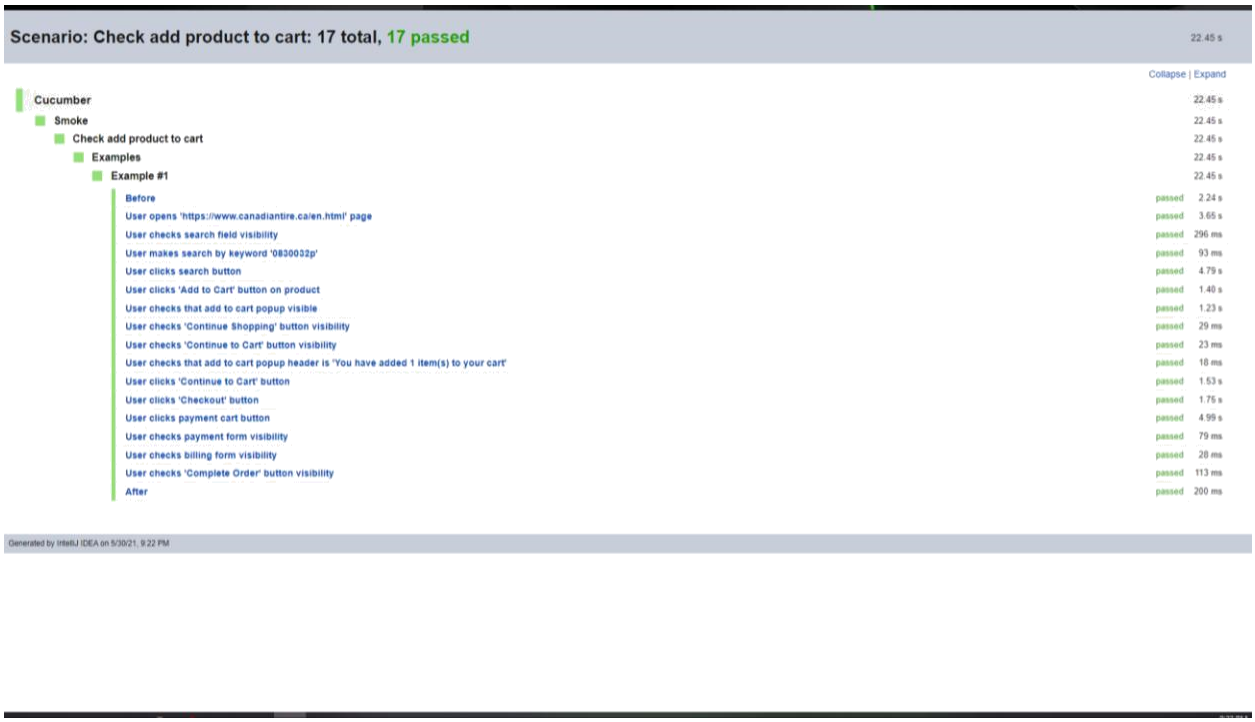


Рисунок 4.18 — Вигляд звіт, що був збережений у форматі html, при його перегляді у браузері

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку Інформаційно інтегрованої системи оцінювання готельного бізнесу. Метою розробки є створення інформаційної інтегрованої системи оцінювання готельного бізнесу.

Особливістю розробки є використання удосконаленого підходу до оцінювання функціонування готельного бізнесу, який дозволяє об'єднувати інформаційні та управлінські ресурси окремих готелів у єдиний інформаційний простір, що надає значні можливості для збору та аналізу даних.

Аналогом може бути Katalon Studio за ціною 4200 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 5.1.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Крите-рій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах
Ринкові переваги					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 5.1

3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні.	Потрібні незначні фінансові ресурси.	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фі-	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у ВПК	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій

Продовження табл. 5.1

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію про-	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	4	4
Наявність аналогів на ринку	3	3	3
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	3
Експлуатаційні витрати	4	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	4	4	4
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	3	4
Термін реалізації ідеї	4	4	3
Супровідна документація	3	4	3
Сума	44	43	42
Середньоарифметична сума балів	(44+43+42) / 3 = 43		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що використовується удосконалений метод щодо оцінювання функціонування

готельного бізнесу, який дозволяє об'єднувати інформаційні та управлінські ресурси окремих готелів у єдиний інформаційний простір, що надає значні можливості для збору та аналізу даних.

Таблиця 5.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

Визначаємо основну заробітну плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \quad ,(5.1)$$

де М — місячний посадовий оклад конкретного розробника (дослідника), грн.;

Тр — число робочих днів за місяць, 20 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	27300	1365,00	28	38220,000
Програміст	22500	1125,00	28	31500,000
Всього				69720,00

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

Визначаємо заробітну плату додаткову заробітну плату розробників, які брати участь в розробці обладнання/програмного продукту. Додаткову заробітну плату прийнято розраховувати як 14,5 % від основної заробітної плати розробників та робітників:

$$Зд = Зо \cdot 14,5\ \% / 100\ \% \quad (5.2)$$

$$Зд = (69720,00 \cdot 14,5\ \% / 100\ \%) = 10109,40 \ \text{(грн.)}$$

Розраховуємо нарахування на заробітну плату розробників. Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Нз = (Зо + Зд) \cdot 22\ \% / 100\% \quad (5.3)$$

$$Нз = (69720,00 + 10109,40) \cdot 22\ \% / 100\ \% = 17562,47 \ \text{(грн.)}$$

Визначаємо витрати на амортизацію обладнання, яке використовувалось для проведення розробки. Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \ \text{[грн.]}. \quad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років

tвик — термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп’ютер балансова вартість якого становить 40000 грн., термін його корисного використання згідно

податкового законодавства – 2 роки, а термін його фактичного використання – 1,40 міс.

$$A_{обл} = \frac{40000}{2} \times \frac{1{,}4}{12} = 2333{,}33 \; грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.5. Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, Внем.ак. : 11000 + 300 = 11300 грн.

Таблиця 5.5 — Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп’ютер та комп’ютерна периферія	40000	2	1,40	2333,333
Офісне обладнання (меблі)	25000	4	1,40	729,167
Приміщення	1000000	20	1,40	5833,333
Всього				8895,83

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл

електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.5)$$

де В — вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,4 кВт;

Ф — фактична кількість годин роботи обладнання, годин.

Кп — коефіцієнт використання потужності, Кп = 0,9.

$$Ве = 0,9 \cdot 0,35 \cdot 8 \cdot 28 \cdot 6,2 = 437,472 \text{ (грн.)}$$

Розраховуємо інші витрати та загальновиробничі витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.6)$$

де Ннзв — норма нарахування за статтею «Інші витрати».

$$Ів = 69720,00 * 80\% / 100\% = 55776 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні

(загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$H_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \quad (5.7)$$

де Ннзв – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Ннзв = 69720,00 * 130 % / 100 % = 90636 (грн.)

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

Взаг = 69720,00+10109,40+17562,47+8895,83+11300+437,47+55776+
+90636 = 264437,17 грн.

Розрахуємо загальні витрати на науково-дослідну (науково-технічну) роботу та оформлення її результатів. Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (5.8)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

ЗВ = 264437,17 / 0,5 = 528874 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

Розрахунок проведемо для випадку коли розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем. В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.9)$$

де ±ΔЦо — розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором, зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

Цо — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, Цо = Цб ± ΔЦо;

Цб — вартість програмного продукту у році до впровадження результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

р — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2023 році ϑ =18%.

Припустимо, що при прогнозованій ціні 3099 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 100 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 2000 шт., протягом другого року – на 2500 шт., протягом третього року на 3000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П1 = (0*100 + (3099 + 100)*2000)* 0{,}8333* 0{,}45) * (1 - 0{,}18) = 1905884{,}924 \text{ грн.}$$

$$\Delta П2 = (0*100 + (3099 + 100)*(2000+2500)* 0{,}8333* 0{,}45) * (1 - 0{,}18) = 4426616{,}073 \text{ грн.}$$

$$\Delta П3 = (0*100 + (3099 + 100)*(2000+2500+3000)* 0{,}8333* 0{,}45) * (1 - 0{,}18) = 7377693{,}455 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 13710194,45 грн.

Оцінимо ефективність вкладених інвестицій та періоду їх окупності. Розраховуємо приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.10)$$

де ΔП — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т — період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}05…0{,}15$;

t — період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

$$ПП = (1905884{,}924/(1+0{,}1)1)+(4426616{,}073/(1+0{,}1)2)+(7377693{,}455/$$
$$/(1+0{,}1)3) = 1732622{,}66 + 3658360{,}391 + 5542970{,}289 = 10933953{,}34 \text{ грн.}$$

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (511)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 528874 = 1057748{,}69 \text{ грн.}$$

Тоді абсолютний економічний ефект Еабс або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Еабс = ПП – PV, \quad (5.12)$$
$$Еабс = 10933953{,}34 – 1057748{,}69 = 9876204{,}64 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (5.13)$$

Тж — життєвий цикл наукової розробки, роки.

$$Ев = \sqrt[3]{(1 + 9876204{,}64/1057748{,}69} - 1 = 1{,}178$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \qquad (5.14)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d = (0,09…0,14);

f — показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

Так як Ев > τmin, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{5.15}$$

Ток = 1 / 1,178 = 0,85 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,85 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 528874 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,85 роки.

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи був створений потужний інструмент — інформаційна мережева система для тестування вебсайтів, яка відповідає всім визначеним вимогам і володіє достатнім рівнем модульності для майбутнього розвитку та розширення. У ході роботи було проведено аналіз інформаційних систем, розглянуті різновиди існуючих систем автоматизованого тестування та виконано варіантний аналіз інструментів для розробки вебсайтів.

На основі цього аналізу було створено веб-додаток для реалізації інтегрованої інформаційної системи для оцінювання готельного бізнесу. На даний момент розроблена система може бути використана для тестування вебсайтів міжнародних готелів та пошуку інформації. У часи збройних конфліктів власникам готелів важливий безперебійний доступ до їхніх вебсайтів, а біженцям та переселенцям потрібна оперативна і достовірна інформація.

Систему, у подальшому, можна рекомендувати як потужний і корисний інструмент для автоматизованого функціонального тестування вебсайтів будь-якого змісту. У майбутньому розвитку проекту планується розширити можливості, зокрема, інтеграцію з білд-сервером Jenkins для впровадження безперервної інтеграції та тестування на всіх етапах розробки вебсайтів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Інтернет як інструмент просування послуг для індустрії гостинності. [Електронний ресурс]. Режим доступу: https://infotour.in.ua/skorbenko.htm.

2. Білозубенко В. С. Поляков, М. П., Шаблій, С. Є. Перспективи розвитку глобальної готельної індустрії / В. С. Білозубенко, М. П. Поляков, С. Є. Шаблій // Економічний простір, 2021, №165, С. 18 – 22.

3. Інноваційні технології у готельному господарстві : навч. посіб. / Н. М. Влащенко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2023. – 150 с.

4. Автоматизація готельного бізнесу. [Електронний ресурс]. Режим доступу: http://plenet.com.ua/?p=1118.

5. Казакова Н. А. Інноваційний розвиток інформатизації готельного бізнесу в умовах глобалізації світового господарства / Н. А. Казакова, А. С. Перепелиця, М. В.Сідоров // Світова економіка та міжнародні відносини, 2017, №1, С. 21 – 25.

6. A. Munir et al., “IFCIoT: Integrated Fog Cloud IoT: A Novel Architectural Paradigm for the Future Internet of Things,” IEEE Consumer Electronics Magazine, vol. 6, PP. 74-82, 2017.

7. David Adelson INTELITY Forecasts Hotel Technology Trends for 2017. [Електронний ресурс]. Режим доступу: https://intelity.com/blog/intelity-forecast-of-hotel-technology-in-2017/.

8. Що таке парсинг і для чого використовується? [Електронний ресурс]. Режим доступу: https://dalistrategies.com/ua/shho-take-parsing-i-dlya-chogo-vikoristovuietsya/

9. Функции Google Spreadsheets в помощь SEO-специалисту. [Електронний ресурс]. Режим доступу: https://it-rating.ua/funktsii-google-spreadsheets-v-pomosch-seo-spetsialistu.

10. In-Depth SEO Audit with Website Spider. [Електронний ресурс]. Режим доступу: https://netpeaksoftware.com/spider.

11. ComparseR. [Електронний ресурс]. Режим доступу: https://download-basket.giveawayoftheday.com/web-development/comparser/

12. Screaming Frog SEO Spider. [Електронний ресурс]. Режим доступу: https://www.screamingfrog.co.uk/seo-spider/

13. Парсер Booking.com. [Електронний ресурс]. Режим доступу: https://www.screamingfrog.co.uk/seo-spider/

14. Selenium automates browsers. [Електронний ресурс]. Режим доступу: https://www.selenium.dev/.

15. HP QuickTest Professional. [Електронний ресурс]. Режим доступу: https://www.wikiwand.com/ru/HP_QuickTest_Professional.

16. Rational Functional Tester. [Електронний ресурс]. Режим доступу: https://www.automation-consultants.com/rational-functional-tester/.

17. Тестування ПЗ (види тестування). [Електронний ресурс]. Режим доступу: https://drukarnia.com.ua/articles/testuvannya-pz-vidi-testuvannya-JInS1.

18. Види тестування сайтів. [Електронний ресурс]. Режим доступу: https://webtune.com.ua/statti/web-rozrobka/vydy-testuvannya-sajtiv/.

19. Руководство: Cucumber + Java. [Електронний ресурс]. Режим доступу: https://habr.com/ru/articles/332754/.

20. WebDriver. [Електронний ресурс]. Режим доступу: https://www.selenium.dev/documentation/webdriver/.

21. JUnit 5 User Guide. [Електронний ресурс]. Режим доступу: https://ac2epsilon.github.io/TRANS/TESTNG/JUnit5UserGuide.html.

22. JetBrains IntelliJ IDEA. [Електронний ресурс]. Режим доступу: https://itpro.ua/product/jetbrains-intellij-idea/?tab=description.

ДОДАТКИ

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри__ОТ__

проф., д.т.н.. Азаров О.Д.

_____

" ____ " _____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

" Інформаційно інтегрована система оцінювання готельного бізнесу "

08-54.МКР.021.00.000 ПЗ

Керівник роботи к.т.н. доц. каф. ОТ

_____ Тарновський М. Г

Студент групи 1КІ–22м

_____ Щербань К. П.

ВНТУ 2023

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність роботи обумовлена необхідністю перевірки достовірності інформації, що розміщується на сайтах готелів.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — вдосконалення інформаційної системи тестування сайтів міжнародних готелів для оцінювання доступності та достовірності представленої на них інформації.

2.2 Призначення розробки — створення системи для автоматичного тестування сайтів готелів.

3 Вихідні дані для виконання МКР

3.1 Призначення системи — тестування веб-сайтів готелів;

3.2 При тестуванні використовувати інформацію з баз даних міжнародних систем бронювання;

3.3 Підтримка можливості змінювати тестові сценарії.

4 Вимоги до виконання МКР

4.1 Провести аналіз предметної області;

4.2 Провести аналіз методів та засобів тестування веб-застосунків;

4.3 Розробити системи тестування сайтів готелів;

4.4 Оцінити комерційний потенціал розробки.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Обгрунтування актуальності теми. Аналіз предметної області	19.09.2023	15.10.2023	Вступ, Розділ 1
2	Аналіз методів та засобів тестування веб-застосунків	16.10.2023	29.10.2023	Розділ 2
3	Розробка системи тестування сайтів готелів	30.10.2023	7.11.2023	Розділ 3
4	Тестування системи	8.11.2023	19.11.2023	Розділ 3
5	Підготовка економічної частини	20.11.2023	3.12.2023	Розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	4.12.2023	10.12.2023	ПЗ, графічний матеріал і презентація
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	11.02.2023	15.02.2023	Оформленні документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового

керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21

ДОДАТОК Б

Архітектура системи

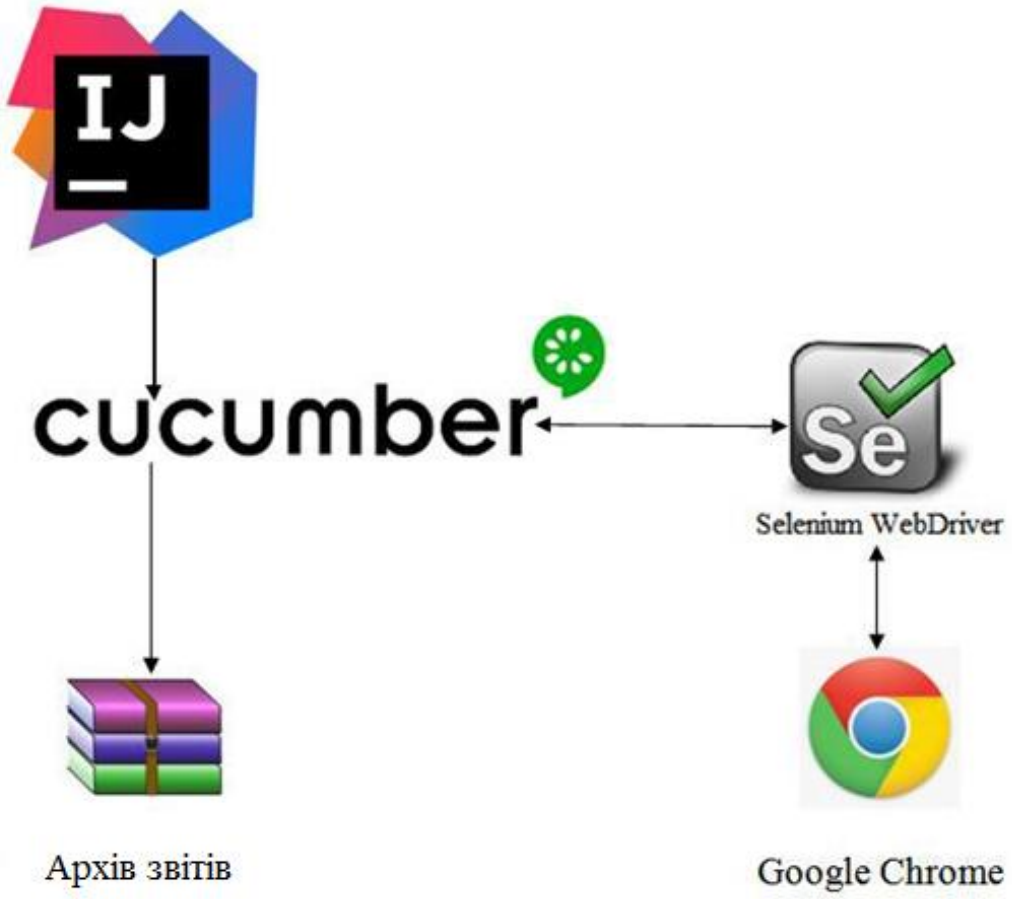


Рисунок Б.1 — Архітектура системи

ДОДАТОК В

Схема основних модулів системи

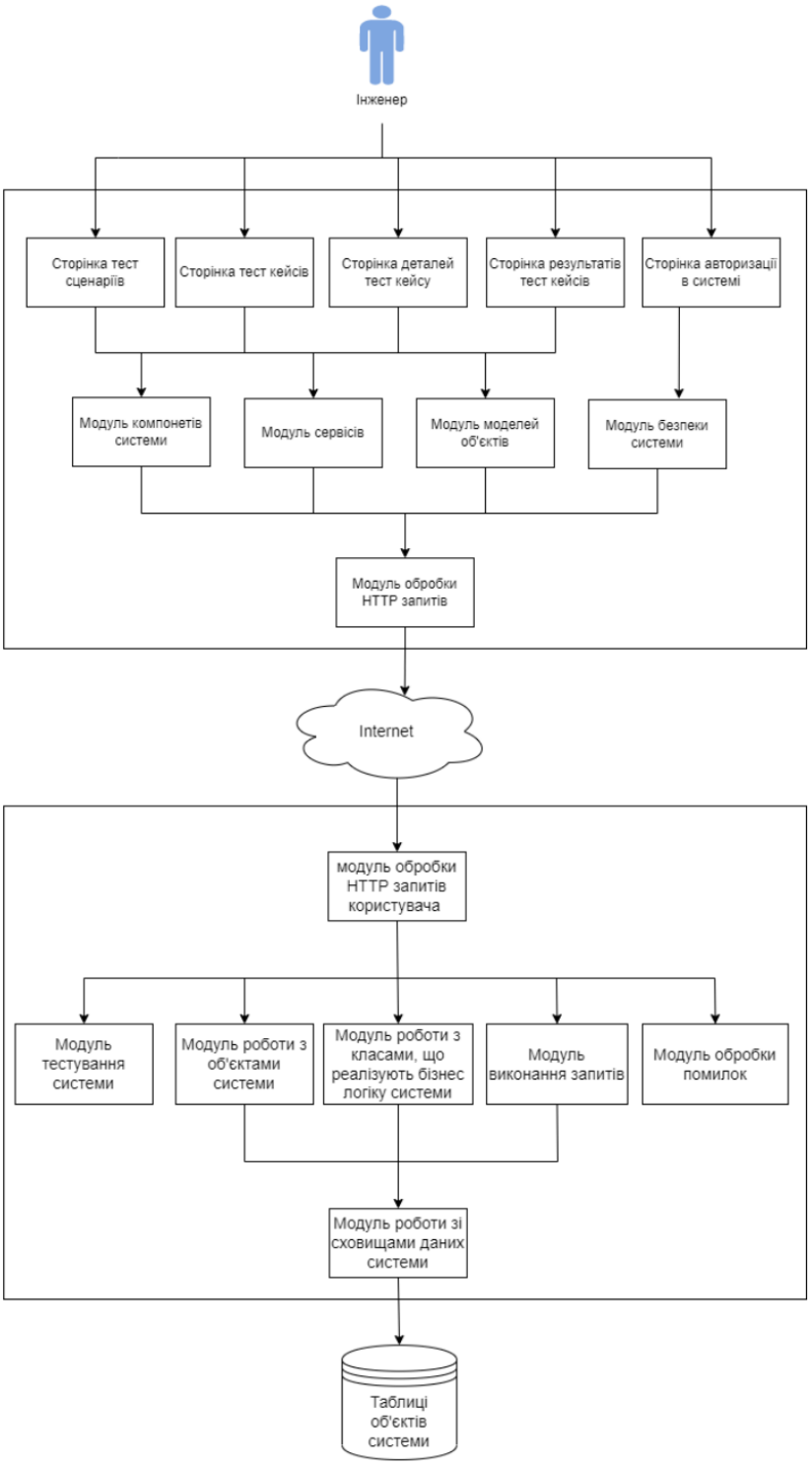


Рисунок В.1 — Схема основних модулів системи

ДОДАТОК Г

Діаграма прецедентів

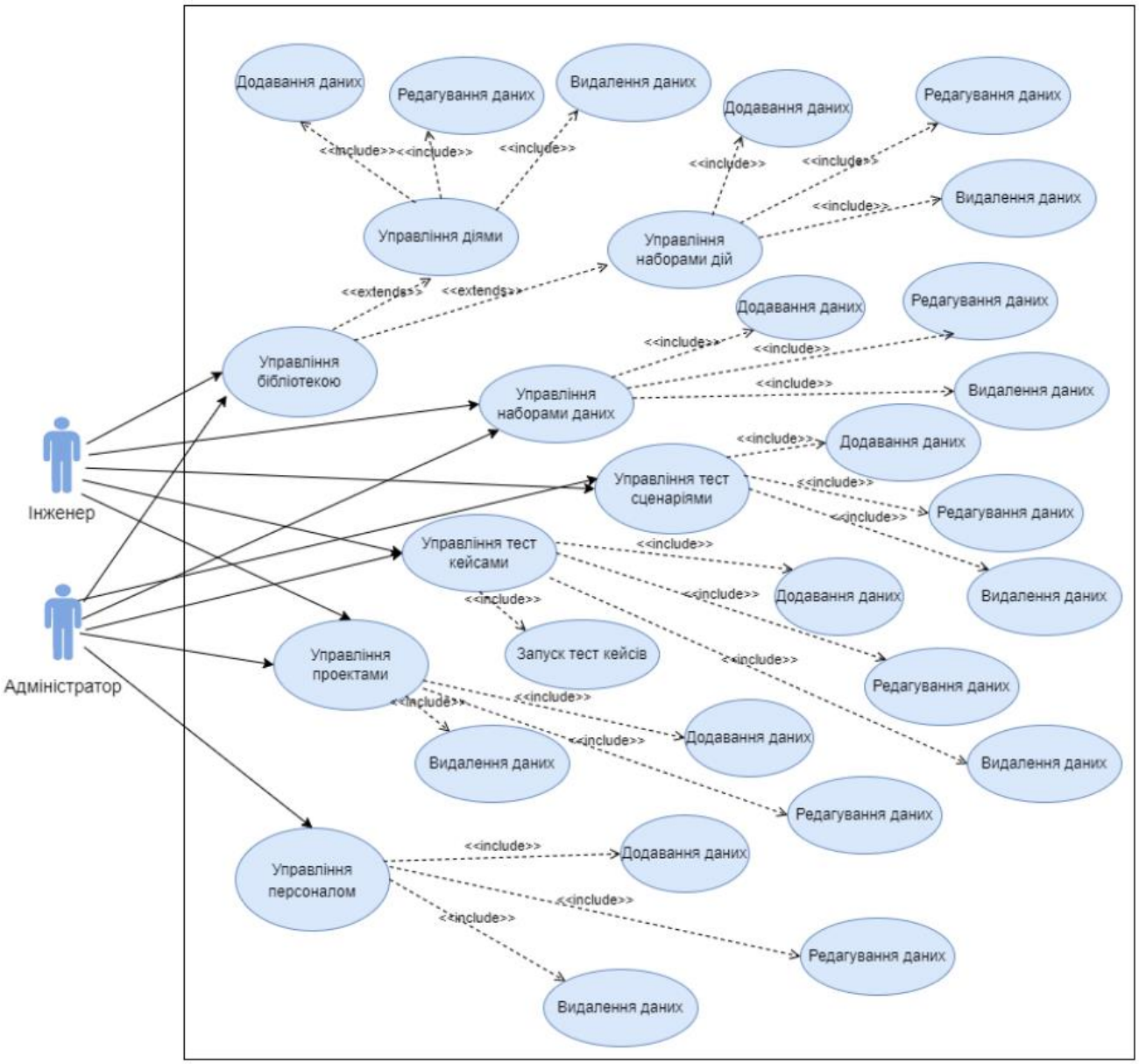


Рисунок Г.1 — Діаграма прецедентів

ДОДАТОК Ж

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: «Інформаційно інтегрована система оцінювання готельного бізнесу»

Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра обчислювальної техніки

Показники звіту подібності Unicheck

Оригінальність 86,4% Схожість 13,6%

Аналіз звіту подібності (відмітити потрібне):

- ✔ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Щербань К. П.

Керівник роботи _____ Тарновський М.Г.