МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ПРОГРАМУВАННЯ

Пояснювальна записка

до магістерської дипломної роботи

магістра
(бакалавра, магістра)

на тему: «Методи та засоби тестування для оцінювання показників якості програмного продукту»

здобувач 2 курсу групи ІСТ-23зм

спеціальність: 126 “Інформаційні системи та технології”

Кравець О.В. _____
(ПІБ здобувача) *(підпис)*

Керівник роботи д.т.н., доц. Лифар В.О. _____
(вчене звання, науковий ступінь, ПІБ) *(підпис)*

Рецензент д.т.н., проф. Меняйленко О.С. _____
(вчене звання, науковий ступінь, ПІБ) *(підпис)*

Київ – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
(повне найменування факультету)
Кафедра Інформаційних технологій та програмування
(повне найменування кафедри)
Освітньо-кваліфікаційний рівень Магістр
(бакалавр, магістр)
Спеціальність 126 "Інформаційні системи та технології"
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП

_____ д.т.н., проф. Захожай О.І.
(підпис)

«__» _____ 20___ року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кравець Олені Вікторівні
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи: Методи та засоби тестування для оцінювання показників якості програмного продукту

Керівник роботи Лифар Володимир Олексійович, д.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від « 06 » грудня 20 24 року № 361/15.15-С
2. Строк подання студентом роботи до захисту 12 грудня 2024 року

3. Вихідні дані: матеріали науково – дослідної практики; науково – методична література; дані інтернет – мереж; інформація про тестування ПЗ; інструменти для процесу розробки.

4. Зміст основної частини (перелік питань, які потрібно розробити):
1) Теорія тестування програмного забезпечення, його типи та види.
2) Інструменти для тестування.
3) Тестування ПЗ на основі технічної документації.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): _____

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання « 04 » листопада 20 24 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання на виконання роботи	04.11.2024	
1	Ознайомлення з темою та постановкою задачі дипломного проекту	09.11.2024	Виконано
2	Вивчення спеціальної літератури і технічної документації	14.11.2024	Виконано
3	Аналізування основ автоматизованого та мануального тестування Веб-додатків	17.11.2024-18.11.2024	Виконано
4	Написання першого розділу щодо аналізу предметної області	19.11.2024-23.11.2024	Виконано
5	Аналізування інструментів для тестування ПЗ	24.11.2024	Виконано
6	Написання другого розділу щодо опису інструментів тестування та їх застосування	25.11.2024-27.11.2024	Виконано
7	Опис тестів за допомогою тест-кейсів і тест-плану	28.11.2024-30.11.2024	Виконано
8	Реалізація практичної частини завдання у третьому розділі	01.12.2024-04.12.2024	Виконано
9	Оформлення пояснювальної записки та узгодження її з керівником	07.12.2024	Виконано
10	Здача пояснювальної записки на кафедру	11.12.2024	Виконано
11	Підготовка доповіді та презентації	11.12.2024	Виконано

Здобувач ..... Кравець О.В.

(підпис) *(прізвище та ініціали)*

Керівник роботи ..... Лифар В.О.

(підпис) *(прізвище та ініціали)*

РЕФЕРАТ

Пояснювальна записка до дипломного проекту на тему «Методи та засоби тестування для оцінювання показників якості програмного продукту» на здобуття освітньо-кваліфікаційного рівня «Магістр» зі спеціальності «Інформаційні системи та технології».

Роботу викладено на 63 сторінках друкованого тесту, містить 13 таблиць, 4 рисунки та перелік джерел з 17 найменувань.

Метою роботи є вивчення та аналіз засобів тестування якості програмного продукту для оцінювання показників його ефективності.

Предметом дослідження є моделі та методи тестування веб-додатків.

Об'єктом дослідження є методи оцінювання якості програмного забезпечення, а також його надійності.

Методи досліджень: здійснюється порівняння різних методів тестування програмного забезпечення та обираються найоптимальніші з них; використовується безперебійне з'єднання і управління структурою за реалізації системи; для розробки програмних засобів системи задіяні методи об'єктно - орієнтованого програмування.

Програмне забезпечення для виконання завдання:

Selenium IDE, Jira Software.

Результати дослідження: проведено аналіз існуючих методів та засобів тестування веб-додатків; на підставі цих даних було розкрито усі теоретичні основи для розробки систем тестування; виконано проектування самої системи, встановлено усе необхідне програмне забезпечення, обрано визначені бібліотеки і сервіси.

Ключові слова: МАНУАЛЬНЕ ТЕСТУВАННЯ, АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ, ТЕХНІЧНА ДОКУМЕНТАЦІЯ, JAVA, SELENIUM, BUG, ЧЕК-ЛИСТ, ТЕСТ-ПЛАН, ТЕСТ-КЕЙС, ВЕБ-ДОДАТОК.

ABSTRACT

Explanatory note for the diploma project on the topic **"Methods and tools for testing to evaluate the quality indicators of a software product"** for obtaining the educational qualification level of "Master" in the specialty **"Information Systems and Technologies"**.

The work is presented on 63 pages of printed text, contains 4 figures, 13 tables and a list of references with 17 sources.

The aim of the work is to study and analyze the tools for testing software quality to evaluate its efficiency indicators.

The subject of the study is the models and methods for testing web applications.

The object of the study is the methods for assessing software quality as well as its reliability.

Research methods: The study involves comparing different software testing methods and selecting the most optimal ones. It uses seamless connection and structural management for system implementation, and object-oriented programming methods for the development of software tools for the system.

Software used for the task: Selenium IDE, Jira Software.

Research results: The analysis of existing methods and tools for testing web applications has been conducted. Based on this data, all theoretical foundations for the development of testing systems were outlined. The system design was completed, all necessary software was installed, and specific libraries and services were selected.

Keywords:

MANUAL TESTING, AUTOMATED TESTING, TECHNICAL DOCUMENTATION, JAVA, SELENIUM, BUG, CHECKLIST, TEST PLAN, TEST CASE, WEB APPLICATION.

ЗМІСТ

ВСТУП

Актуальність теми. Інформаційні та комп'ютерні технології оточують наше повсякденне життя. Безліччю речей навколо нас керує програмне забезпечення. Воно контролює мобільні телефони, комп'ютери, автомобілі і кредитні картки.

Кожен з нас хоч раз в житті зіткався з тією чи іншою проблемою у роботі програм: дуже прикро, коли ніяк не завантажується сайт, банкомат не віддає картку, текстовий редактор не дає закінчити роботу з документами. Все це ускладнює нам життя і, як мінімум, дратує.

У зв'язку із стрімким розвитком ІТ – ринку та «роботизацією» усіх сфер нашої діяльності, одним із головних стає питання надійності програмних продуктів і захисту від цифрових загроз, а саме – питання кібербезпеки. Людський фактор вчиняє помилки, і для різних програмних систем існують різні рівні ризику. Саме тому, перш ніж ефективно і безпечно використовувати будь – який програмний продукт, його потрібно протестувати. Загальна перевірка працездатності програмного забезпечення і виявлення недоліків на етапі розробки забезпечується різними методами та засобами тестування.

Значно покращити надійність і продуктивність системи дозволяє тестування, яке проводиться на всіх етапах розробки. Команда професійних тестувальників під час перевірки повинна переконатися в тому, що програмний продукт не відхиляється від виконання усіх задокументованих функцій. Тому критично важливим є включення тестування в життєвий цикл розробки програмного забезпечення, це забезпечує високу якість кінцевого ІТ- продукту.

Той факт, що тестування впроваджується саме на ранніх стадіях роботи над продуктом, означає, що буде значно знижено витрати на усунення різних помилок. Тож, застосування перевірочних заходів у будь – яких ІТ – рішеннях скасовує суттєві іміджеві та економічні втрати. Але слід знати, що для абсолютної якості вихідного програмного забезпечення зазвичай недостатньо використання одного методу тестування, необхідно будувати цілу систему тестування, з використанням інструментів автоматизованого тестування.

Мета і завдання дослідження. Метою роботи є вивчення засобів та методів тестування для оцінювання показників якості програмного продукту.

Об’єкт дослідження: процеси розроблення веб-додатків.

Предмет дослідження: моделі, засоби та методи тестування веб-додатків.

Методи дослідження: Для вирішення проблеми надійної перевірки функціонування веб – продуктів застосовуються методи з дослідження системного аналізу.

Наукова новизна одержаних результатів:

- зініційовано використання методу модульної системи контролю програмного забезпечення у поєднанні із декількома методами тестування;

- розроблено модель складного тестування веб-додатків за допомогою декількох методів.

Практичне значення отриманих результатів:

запропоновані методи та засоби тестування сприяють виявленню помилок на різних етапах розробки, що дозволяє підвищити надійність і функціональність програмного продукту;

аналіз існуючих методів і інструментів дозволяє вибрати найефективніші з них для конкретних типів проектів, що зменшує час і ресурси, необхідні для перевірки програмного забезпечення;

результати роботи можуть бути використані в компаніях для створення та впровадження систем тестування веб-додатків або інших типів програмного забезпечення.

РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування, або Quality Assurance (QA) – важливий етап, який проходить будь-який програмний продукт перед релізом. Він визначає якість і рівень готовності програми, наявність помилок і відповідність вимогам клієнта.

В процесі роботи QA-інженери використовують різні технології, методології і рівні тестування ПЗ для перевірки функціональних і не функціональних можливостей продукту. Кожен з них має свою специфіку, відправну точку і завдання.

Під методологією розуміються різні комбінації ідей, способів і принципів тестування програм, до яких фахівці вдаються під час роботи над проектом. У деяких випадках, крім експертів, для перевірки ПЗ збирається фокус-група, яка допомагає аналізувати поведінку користувачів і виявляти інтуїтивно незрозумілі функції [1].

1.1 Основні типи тестування програмного забезпечення

Існує кілька ознак, за якими прийнято робити класифікацію типів тестування. Зазвичай виділяють наступні:

1) За об'єктом тестування:

- функціональне тестування (functional testing);
- тестування продуктивності (performance testing):
- навантажувальне тестування (load testing);
- стрес-тестування (stress testing);
- тестування стабільності (stability / endurance / soak testing);
- тестування зручності використання (usability testing);
- тестування інтерфейсу користувача (ui testing);
-тестування безпеки (security testing);
- тестування локалізації (localization testing);

- тестування сумісності (compatibility testing).

2) За знанням системи:

- тестування чорного ящика (black box);
- тестування білого ящика (white box);
- тестування сірого ящика (gray box).

3) За ступенем автоматизації:

- ручне тестування (manual testing);
- автоматизоване тестування (automated testing);
- напівавтоматизоване тестування (semiautomated testing).

4) За ступенем ізольованості компонентів:

- компонентне (модульне) тестування (component / unit testing);
- інтеграційне тестування (integration testing);
- системне тестування (system / end-to-end testing).

5) За часом проведення тестування:

- Альфа-тестування (alpha testing):
- тестування при прийманні (smoke testing);
- тестування нової функціональності (new feature testing);
- регресійне тестування (regression testing);
- тестування при здачі (acceptance testing);
- Бета-тестування (beta testing).

6) За ознакою позитивності сценаріїв:

- позитивне тестування (positive testing);
- негативне тестування (negative testing).

7) За ступенем підготовленості до тестування:

- тестування за документацією (formal testing);

- ед хок (інтуїтивне) тестування (ad hoc testing).

1.2 Методи тестування програмного забезпечення

Глибоке розуміння методів тестування якості програмного забезпечення дозволяє розробникам та тестувальникам надавати виняткові програмні продукти. Ретельно аналізуючи всі аспекти програмного забезпечення, від функціональності та зручності використання до продуктивності та безпеки, тестування QA закладає основу для створення надійного, безпечного та зручного для користувача програмного забезпечення, яке процвітає на конкурентному цифровому ринку.

1.2.1 Мануальне та автоматизоване тестування

Мануальне (ручне) тестування.

Ручне тестування (Manual testing) — процес ручної перевірки програмного забезпечення на помилки. Тестувальник має відігравати роль користувача програми й використовувати властивості програми для знаходження помилок у роботі програми. Для професійного тестування тестувальник часто користується написаним планом тестування з варіантами тестування (test cases).

Великою перевагою ручного тестування є те, що Ви при проходженні сценаріїв можете побачити реальні проблеми, з якими зіткнеться кінцевий користувач. Ручне тестування набагато гнучкіше, ніж автоматичне. Використовуючи автоматизовані засоби, важко змінити значення в програмі, після початку тестування. При ручному тестуванні, Ви можете швидко перевірити результати, і це дозволить Вам побачити, який функціонал працює найкраще.

Автоматизоване тестування.

Автоматизоване тестування програмного забезпечення (automated testing) — частина процесу тестування на етапі контролю якості в процесі розробки програмного забезпечення. Воно використовує програмні засоби для виконання

тестів і перевірки результатів виконання, що допомагає скоротити час тестування і спростити його процес.

Автоматизоване тестування найкраще застосовувати, коли Ви працюєте над великим проектом, і у тестовій системі буде багато користувачів. Самі більші переваги автоматизованого тестування - це відносна швидкість і ефективність. Після того, як Ви створили тест, для повторення результату, треба тільки ввести необхідні вихідні дані і запустити його. Все інше буде зроблено автоматично. Недоцільно використовувати автоматичне тестування на маленьких та короткострокових проектах, оскільки витрати на автоматичне тестування будуть перевищувати вигоду. Тому, найбільш ефективним буде використання ручного тестування [2].

1.2.2 Тестування «чорної скриньки» (Black Box Testing)

Тестування методом «чорної скриньки» — це метод тестування програмного забезпечення, при якому перевіряється робота програми без знання її внутрішньої побудови та схеми роботи. Іншими словами, не маючи доступу до коду програми. Цей метод тестування може бути застосований практично до будь-якого рівня тестування програмного забезпечення: модульного, інтеграційного, системного або приймального тестування. Іноді цей метод називають тестуванням на основі специфікації.

Під «чорною скринькою» слід розуміти об'єкт дослідження, внутрішній устрій якого невідомо. Поняття «чорна скринька» було запропоновано У. Р. Ешбі. У кібернетиці воно дозволяє вивчати поведінку систем, тобто їх реакцій на різноманітні зовнішні впливи і в той же час абстрагуватися від їх внутрішнього устрою.

При цьому виді тестування конкретні знання коду програми, внутрішньої структури та знань програмування загалом не потрібні. Тестувальник знає, що саме має робити програмне забезпечення, але не знає, як саме вона це робить. Наприклад, тестувальник знає, що на певні вхідні дані програма повертає пев-

ний незмінний результат, але не знає про те, як саме програмне забезпечення створює вихідні дані [3, 4].

Переваги:

- тестування проводиться з позиції кінцевого користувача і може допомогти виявити неточності і протиріччя в специфікації;
- тестувальнику немає необхідності знати мови програмування і заглиблюватися в особливості реалізації програми;
- тестування може проводитися фахівцями, незалежними від відділу розробки, що допомагає уникнути упередженого ставлення;
- можна починати писати тест-кейси, як тільки готова специфікація.

Недоліки:

- тестується тільки дуже обмежена кількість шляхів виконання програми;
- без чіткої специфікації (а це швидше реальність на багатьох проектах) досить важко скласти ефективні тест-кейси;
- деякі тести можуть виявитися надмірними, якщо вони вже були проведені розробником на рівні модульного тестування [5].

1.2.3 Тестування «білої скриньки» (White Box Testing)

Тестування білого тесту - тестування внутрішнього кодування та інфраструктура програмного рішення. Основна увага зосереджена на зміцненні безпека, потік входів і виходів через додаток, а також покращення дизайну та зручності використання. Інші назви тестування білого ящика: тестування чистого ящика, тестування відкритого ящика, структурне тестування, тестування на основі коду та тестування скляного ящика.

Тестування білого ящика включає тестування програмного коду на наступних частинах:

- внутрішні отвори для безпеки;
- розбиті або погано структуровані шляхи в процесі кодування;
- потік конкретного вводу через код;
- очікувані результати;

- функціональність умовних циклів;
- тестування кожної процедури, об'єкта та функції на індивідуальній основі.

Ми можемо проводити тестування на системному, інтеграційному та одиничному рівнях розробка програмного забезпечення. Крім того, одна з основних цілей тестування «білої скриньки» — налагодити робочий процес для програми. Він передбачає перевірку серії попередньо визначених вхідних даних на очікувані або бажані вихідні дані, тому, якщо певний вхідний сигнал не призводить до очікуваного результату, сталася помилка.

Переваги випробувань White Box:

- оптимізація коду шляхом пошуку прихованих помилок;
- легка автоматизація тестових випадків для білого поля;
- тестування більш ретельне, тому зазвичай перевіряються всі кодові шляхи;
- тестування може розпочатися на початку проекту, навіть якщо графічний інтерфейс недоступний.

Недоліки випробувань White Box:

- тестування на білому полі може бути складним і дорогим;
- розробники, які зазвичай використовують тестові кейси для білих скриньок, не люблять цього. Тому розробники не перевіряють білу скриньку детально, що може призвести до виробничих помилок;
- крім того, для тестування білої коробки, професійні інструменти необхідні та глибокі знання програмування та впровадження;
- тестування білої скриньки займає багато часу, потрібен час для повного тестування великих програм [6].

1.2.4 Тестування «сірої скриньки» (Grey Box Testing)

Метод сірої скриньки — комбінація методів білої скриньки та чорної скриньки, яка полягає в тому, що до частини коду та архітектури у тестувальника доступ є, а до частини — ні. Зазвичай говорять про методи білої або чорної скриньки стосовно тих чи інших частин застосунка, при цьому розуміючи,

що «застосунок цілком» тестується за методом сірої скриньки. Деякі автори визначають метод сірої скриньки як протиставлення методам білої і чорної скриньки, особливо підкреслюючи, що за методом сірої скриньки внутрішня структура об'єкта, що тестується, відома частково і з'ясовується в міру дослідження.

Цей підхід, безперечно, має право на існування, але у своєму граничному разі він вироджується до стану «частину системи ми знаємо, частину — не знаємо», тобто до тієї ж комбінації білої і чорної скриньок. Методи білої та чорної скриньки не є конкуруючими або взаємовиключними — навпаки, вони гармонійно доповнюють один одного, компенсуючи таким чином недоліки.

У таблиці 1.1 наглядно показано переваги та недоліки методів тестування програмного забезпечення.

Таблиця 1.1 - Переваги та недоліки методів білого, чорного та сірого ящиків

Black Box Testing	White Box Testing	Grey Box Testing
Це найменш трудомісткий і вичерпний вид	Найбільш вичерпний і трудомісткий вид тестування	Частково трудомісткий та вичерпний вид
Не підходить для тестування за алгоритмом	Підходить для тестування за алгоритмом	Не підходить для тестування за алгоритмом
Можна зробити лише методом проб і помилок	Будову системи та внутрішні дані можна краще перевірити	Будову системи та внутрішні дані можна перевірити, якщо вони відомі
Внутрішня робота програми не повинна бути відомою	Тестер повністю володіє знаннями про внутрішню будову додатку	Дещо відомо про внутрішню будову
Тестування базується на зовнішніх очікуваннях, внутрішня поведінка програми невідома	Внутрішня будова системи повністю відома, і тестер може відповідно розробляти сценарії випробувань	Тестування проводиться на основі діаграм баз даних високого рівня та діаграм потоку даних
Виконується кінцевими користувачами, а також тестерами та розробниками	Зазвичай виконується тестерами та розробниками	Виконується кінцевими користувачами, а також тестерами та розробниками

1.3 Рівні тестування програмного забезпечення

Виділяють два основні рівні тестування програмного забезпечення: функціональне та нефункціональне, а також окремі види тестування.

1.3.1 Функціональне тестування

Функціональне тестування – це те, що запобігатиме потребі в дорогому і трудомісткому ремонті в майбутньому, а також підтримає задоволеність клієнтів. У процесі розвитку QA як професії виникла дуже велика кількість видів і типів тестування.

Існують різні класифікації, а також підходи до опису видів тестування залежно від того, яку мету ставить перед собою фахівець, що безпосередньо проводить тестування. Об’єктивно новий вид тестування можна уявити в будь-який момент, якщо просто структуровано й тезово пояснити своє бачення і підхід на цей вид тестування. Але все ж таки є загальноприйняті види тестування і їх всього два – це функціональне і нефункціональне тестування.

Функціональне тестування — це процес перевірки працездатності програмного забезпечення, унаслідок якого порівнюють фактичну поведінку системи на відповідність із функціональними вимогами замовника.

Для функціонального тестування можна виділити чотири основні рівні (іноді їх ще називають типи функціонального тестування):

- модульне тестування (Unit testing) – найчастіше проводиться розробниками. Це приклад функціонального тестування, де тестується певний ізольований модуль додатка обмежено у власному середовищі. Жодні зовнішні чинники та взаємозв’язок з іншими модулями програмного забезпечення не враховуються на цьому рівні тестування;

- інтеграційне тестування (Integration testing) – рівень тестування, за якого перевіряється, як окремі компоненти можуть взаємодіяти між собою. Без урахування зовнішніх факторів;

- системне тестування (System testing) – рівень тестування, де перевіряється працездатність програмного забезпечення цілком. Можна сказати, що на цьому рівні й проводять порівняння фактичної поведінки системи на відповідність із функціональними вимогами замовника;

- приймальне тестування (Acceptance testing) – завершальний рівень тестування, який зазвичай проводять замовник, або ж його представник, а також може проводити кінцевий користувач. Проводиться з метою отримання оцінки про готовність програмного забезпечення та відповідність його до вимог замовника;

Виконуючи приймальні тести на системі, тестувальна група визначить, як буде працювати програма під час постійного використання. Існують також юридичні та договірні вимоги щодо прийняття системи в роботу [7].

Форми приймального тестування. Не менш важливими є методології перевірки споживачів (типи перевірки прийнятності), які допомагають зміцнити впевненість у запуску продукту і, таким чином, призводять до успіху товару на ринку.

Етапи альфа- та бета-тестування в основному зосереджуються на виявленні помилок вже перевіреного продукту, і вони дають чітке уявлення про те, як продукт насправді використовується користувачами в режимі реального часу. Вони також допомагають отримати досвід роботи з продуктом до його запуску, а цінний зворотний зв'язок ефективно впроваджується для підвищення зручності використання продукту.

Цілі та методи альфа- та бета-тестування переключаються між собою залежно від процесу, застосованого в проекті, і можуть бути змінені, щоб відповідати процесам.

Обидва ці методи тестування заощадили тисячі доларів на масштабних випусках програмного забезпечення для таких компаній, як Apple, Google, Microsoft тощо.

Альфа-тестування.

Цей тест є першим етапом тестування і проводитиметься серед команд (розробників та команд із забезпечення якості). Тестування модулів, тестування інтеграції та тестування системи у поєднанні відомі як альфа - тестування. Під час цієї фази в додатку буде перевірено наступне:

- орфографічні помилки;

- несправні посилання;

- додаток буде протестоване на машинах з найнижчою специфікацією для перевірки часу завантаження та будь-яких проблем із затримкою.

Бета-тестування.

Цей тест виконується після успішного альфа-тестування. При бета - тестуванні частина ймовірних користувачів тестує додаток. Бета-тестування також відоме як тестування перед випуском. Бета-тестові версії програмного забезпечення інколи розподіляються серед широкої аудиторії в Інтернеті, частково для того, щоб виконати тест в реальних умовах. Якщо розроблюване програмне забезпечення призначене для внутрішнього користування в певній компанії, то для тестування залучають співробітників замовника. На цьому етапі аудиторія буде тестувати наступне:

- користувачі встановлюють, запускають додаток та надсилають свої відгуки команді проекту;

- типографічні помилки, заплутаність потоку додатків і навіть збої;

- отримуючи зворотний зв'язок, команда проекту може вирішити проблеми перед тим, як випустити програмне забезпечення фактичним користувачам;

- чим більше виправлених проблем, які вирішують реальні проблеми користувачів, тим вище буде якість вашої програми;

- наявність більш якісної програми під час випуску для широкої громадськості підвищить задоволеність клієнтів [8].

Різницю між альфа- і бета-тестуванням надано у таблиці 1.2.

Таблиця 1.2 - Відмінності між альфа- і бета-тестуванням

Альфа-тестування	Бета-тестування	Дії
Підвищує якість продукту і забезпечує готовність до бета-тестування.	Підвищує якість продукту, інтегрує дані про клієнта в готовий продукт і забезпечує готовність до випуску.	Що робить
Ближче до кінця процесу розробки, коли продукт перебуває в майже повністю працездатному стані.	Безпосередньо перед запуском.	Коли проводиться
Зазвичай дуже довго протягом багатьох ітерацій. Альфа-тестування нерідко триває в 3-5 разів більше тривалості бета-тестування.	Зазвичай тільки кілька тижнів (іноді до декількох місяців) з невеликою кількістю основних ітерацій.	Як довго проводиться
Майже виключно включає перевірку якості ПЗ (баги).	Зазвичай включає в себе маркетинг, підтримку, документацію, якість і інжиніринг (в основному, всю групу продуктів).	Що включає в себе
Зазвичай виконується тестувальниками, розробниками, а іноді і «друзями і сім'єю». Фокусується на тестуванні, яке буде емулювати ~ 80% клієнтів.	Випробувано в «реальному світі» з «реальними клієнтами» і зворотний зв'язок може охоплювати кожен елемент продукту.	Хто проводить
Безліч помилок, збоїв, відсутніх документів і функцій.	Кілька помилок, менше збоїв, більшість документації і функцій завершені.	Що отримують тестувальники
Більшість відомих критичних проблем виправлені, деякі функції можуть бути змінені або додані в результаті раннього зворотного зв'язку.	Велика частина зібраної зворотного зв'язку розглядається і/або застосовується в майбутніх версіях продукту. Виконуються тільки важливі/критичні зміни.	Налагодження

Продовження таблиці 1.2

Тестують за методологією, показниками ефективності. Хороший альфа-тест задає чітко визначені критерії і вимірює продукт по відношенню до цих орієнтирів.	Тестують із застосуванням реальності та уяви. Бета-тести досліджують межі продукту, дозволяючи клієнтам досліджувати кожен елемент продукту в своєму рідному середовищі.	Як тестують
Є відмінне уявлення про те, як працює продукт і чи відповідає він критеріям дизайну (і чи «бета-готовий» він).	Є уявлення про те, що ваш клієнт думає про продукт і про те, що він може відчувати, коли купує його.	Результат

1.3.2 Нефункціональне тестування

Це тестування нефункціональних аспектів програмного забезпечення, таких як продуктивність, зручність використання, надійність, безпека, тощо. На відміну від функціонального тестування, котре фокусується на перевірці правильності виконання програмним забезпеченням певних функцій, нефункціональне тестування оцінює наскільки добре програмне забезпечення працює в різних умовах.

Основні типи нефункціонального тестування:

Тестування продуктивності (Performance Testing):

оцінка роботи системи з точки зору швидкодії та стабільності, в умовах визначеного робочого навантаження.

Існують різні причини, які сприяють зниженню продуктивності програмного забезпечення:

- затримки мережі;
- обробка на стороні клієнта;
- обробка транзакцій в базі даних;
- балансування навантаження між серверами;
- відображення даних.

Тестування ефективності розглядається як один із важливих та обов'язкових типів тестування з точки зору наступних аспектів:

- швидкість (тобто час відгуку, надання та доступ до даних);
- ємність;
- стабільність;
- масштабованість.

Це може бути або якісна, або кількісна перевірка, і її можна розділити на різні під види, такі як навантажувальне тестування (Load testing) та стрес тестування (Stress testing).

Навантажувальне тестування (Load Testing):

перевірка поведінки системи під очікуваним користувацьким навантаженням. Цей тип тестування визначає максимальну потужність програмного забезпечення та його поведінку у піковий час.

Більшу частину часу тестування навантаження проводиться за допомогою автоматизованих інструментів, таких як Load Runner, AppLoader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test тощо.

Стрес тестування (Stress Testing):

визначає надійність програмного забезпечення, шляхом його тестування в умовах, що виходять за межі нормальної роботи та перевищують навантаження на систему.

Основна мета - протестувати програмне забезпечення, застосовуючи навантаження на систему та переймаючи ресурси, що використовуються програмним додатком для визначення точки зламу. Це тестування може бути проведено шляхом тестування різних сценаріїв, таких як:

- вимкнення або перезапуск мережевих портів випадковим чином;
- увімкнення та вимкнення бази даних;
- запуск різних процесів, що споживають такі ресурси, як процесор, пам'ять, сервер тощо.

Об’ємне тестування (Volume Testing):

проводиться для оцінки здатності системи обробляти певні обсяги даних (зазвичай рівні або близькі до максимальних). Об’ємні тести можна використо-

вувати, для перевірки того, чи немає втрати даних або помилок, коли до системи додаються великі обсяги даних.

Тестування на відмову та відновлення (Failover and Recovery Testing):

підвид тестування продуктивності, котрий перевіряє здатність системи відновлюватись після можливих збоїв, що виникли у зв'язку з помилками програмного забезпечення, відмовою устаткування, проблемами мережі, тощо.

Тестування витривалості (Endurance Testing, Stability Testing):

містить у собі тестування системи в умовах значного навантаження, впродовж тривалого періоду часу. Допомагає з'ясувати можливість системи витримувати постійне необхідне навантаження, протягом визначеного періоду часу. Здебільшого використовується для перевірки витоків пам'яті, часу відгуку, правильності підключення та закриття з'єднань (наприклад БД), тощо.

Тестування надійності (Reliability Testing):

перевіряє здатність програмного забезпечення працювати без збоїв протягом певного періоду часу, або певної кількості операцій.

Тестування зручності використання (Usability Testing):

Оцінює користувацький інтерфейс програмного забезпечення та зручність роботи з ним.

Зручна для користувачів система повинна виконувати наступні п'ять цілей:

- проста для вивчення;
- проста для запам'ятовування;
- ефективна у використанні;
- задовільна у використанні;
- просту для розуміння.

Тестування графічного інтерфейсу користувача (Graphical User Interface Testing, GUI Testing):

перевірка візуальних компонентів програми (таких як: кнопки, меню, іконки, текстові поля, списки, форми, зображення та інші інтерактивні елементи). Тестування GUI має на меті забезпечення відповідної функціональності та есте-

тики користувацького інтерфейсу, що призводить до покращення користувацького досвіду та загальної якості продукту.

Тестування безпеки (Security Testing):

виявляє вразливості, загрози та ризики в системі, для запобігання потенційним зловмисним атакам; включає тестування програмного забезпечення з метою виявлення будь-яких недоліків та вразливих місць з точки зору безпеки та вразливості.

Нижче наведено основні аспекти, які перевіряються під час тестування безпеки:

- конфіденційність;
- цілісність;
- автентифікація;
- доступність;
- авторизація;
- програмне забезпечення захищене від відомих і невідомих загроз;
- дані програмного забезпечення захищені;
- програмне забезпечення відповідає всім нормам безпеки;
- перевірка та валідація вхідних даних.

Тестування сумісності (Compatibility Testing):

перевіряє коректну роботу програмного забезпечення в різних середовищах, таких як браузери, пристрої, операційні системи, тощо.

Тестування переносимості (Portability Testing):

перевіряє наскільки легко програмне забезпечення може бути перенесено з одного середовища в інше.

Тестування масштабованості (Scalability Testing):

тип навантажувального тестування, що проводиться для визначення здатності програмного забезпечення масштабуватися відповідно користувацького навантаження, кількості транзакцій, обсягу даних, тощо. Іншими словами, тестування масштабованості перевіряє, як система буде працювати під час раптового стрибка або падіння навантаження, від запитів користувачів.

Тестування підтримки (Maintainability Testing):

перевіряє легкість модифікації програмного продукту після випуску, з метою виправлення дефектів, покращення продуктивності та інших характеристик, або для адаптації продукту до нових умов.

Інсталяційне тестування (Installation Testing):

перевіряє безперешкодне встановлення додатку (програмного продукту) та його належну роботу після інсталяції. Перевірка може містити кроки, котрі повинен виконати користувач для встановлення додатку, а також його початкову функціональність.

Тестування документації (Documentation Testing):

процес перевірки документації на наявність помилок та прогалин. Включає в себе перевірку таких аспектів документації як наприклад: повнота, точність, актуальність, зрозумілість, відсутність граматичних помилок, тощо.

Тестування відповідності (Compliance Testing):

перевірка відповідності програмного продукту визначеним стандартам, угодам або правилам законодавства, та іншим подібним приписам.

Тестування локалізації (Localization Testing):

процес перевірки того, що програмний продукт локалізований і спроможний функціонувати за призначенням на визначених ринках. Гарантує, що програмне забезпечення належним чином адаптовано для цільової аудиторії (мова, культурні традиції, регіональні стандарти). Включає перевірку таких аспектів як: переклад інтерфейсу користувача, формати дати та часу, формати чисел, валютні символи, тощо.

Тестування інтернаціоналізації (Internationalization Testing):

перевіряє програмне забезпечення на предмет адаптивності та пристосованості до різних мов, географічних регіонів і культурних уподобань, без внесення змін до коду. Тестування інтернаціоналізації по суті перевіряє основу для локалізації, що дозволить в майбутньому легко адаптувати програмний продукт для певних регіонів і мов.

Отже, не зважаючи на те, що зазвичай основний фокус спрямовано на функціональне тестування, нефункціональне тестування є також важливим, для

створення високоякісного продукту, котрий відповідає очікуванням користувачів і добре працює в реальних умовах [9].

1.3.3 Види тестування, пов’язані зі змінами

До окремих видів тестування можна додати ті, які необхідно виконувати в разі, якщо відбуватимуться зміни в нашому продукті. Це можуть бути такі види тестування:

- *регресійне тестування (Regression testing)* – вид тестування ПЗ, який проводиться після внесення в програму змін. Перед тестуванням обирається список тест-кейсів, за якими проводитиметься оцінювання ПЗ на предмет появи нових відхилень, а також на те, що попередній функціонал працює добре і без змін.

- *димове тестування (Smoke testing)* – вид тестування ПЗ, що перевіряє базову функціональність, тобто перевірка того, що основні функції програми працюють без відхилень і помилок. Це дає змогу переходити до тестування вужчих модулів і напрямів роботи ПЗ.

- *тестування чистоти (Sanity testing)* – так само як і димове тестування, перевіряє основний ключовий функціонал, але не так глибоко. У пріоритеті перевірка саме ключових областей, на які можуть вплинути зміни та нові функції вашого ПЗ.

- *адаптаційне тестування (Adaptation Testing)* – перевірка того, що програма успішно адаптується до нових, що виникли внаслідок змін, вимог.

- *тестування оновлень (Patch testing)* – проводиться в разі, якщо зміни надаються у вигляді патча або оновлення. Перевіряється їх коректність і стабільність. Основною метою є перевірка, що оновлення встановлюються без збоїв і не призводять до небажаних побічних ефектів і багів.

Це далеко не всі види тестування, які можуть бути пов’язані зі змінами програмного забезпечення. Що стосується функціонального тестування, то воно часто підлягає автоматизації, а також застосуванню певних технік, підходів та інструментальних засобів, які дають змогу частково виключити людське

втручання в процес тестування. Це окремий високо структурований підхід до тестування, який може охоплювати кроки з вибору інструментів автоматизації, розроблення тест-кейсів для автоматизації, налаштування оточень та інтеграцію з CI/CD, подальшу підтримку з моніторингом результатів, а також отримання та збирання звітності для подальшого подання замовнику.

Автоматизація тестування, звісно, корисна, але необхідно розуміти, що це трудомісткий процес, що вимагає вкладень і грамотного ведення всіх процесів. Тому перш ніж розпочинати ці процеси, необхідно переконатися у її доцільності.

1.4 Постановка задачі дослідження

Людина схильна помилятися, недоліки і помилки можуть призвести до порушення роботи на всіх стадіях розробки програмного забезпечення. При цьому наслідки некоректної роботи ПЗ можуть бути самими різними — від незначних до катастрофічних.

Для програмного забезпечення будь-якої складності неможливо вилучити всі проблеми. Проте тестування допомагає виявити більшість помилок, з якими може зіткнутися користувач й потенційно зменшує ризик виникнення проблем з програмою у майбутньому.

Магістерська атестаційна робота присвячена дослідженню важливої галузі у сучасній інформатиці — тестування програмного забезпечення. Даний вид діяльності активно застосовується у процесі життєвого циклу розробки ПЗ та безпосередньо впливає на забезпечення якості ПЗ. Тому вивчення проблем тестування та розробка інструментарію для його проведення є актуальним завданням.

Для досягнення поставленої мети протягом дослідницької практики були вирішено таке завдання, як дослідження ефективності існуючих методів і засобів тестування за наступними критеріями оцінки:

- час (швидкість) тестування;
- повнота тестування;

- повторюваність, неодноразове використання;
- навантаження;
- надійність.

1.5 Висновки до розділу

Під час проведення дослідження було проаналізовано основні методи та підходи з тестування програмного забезпечення. Були розглянуті поняття ручного та автоматизованого тестування. Були вказані ситуації, коли необхідно вводити автоматизоване тестування та засоби, за допомогою яких це проводиться.

Проаналізовано переваги та недоліки різних підходів до тестування програмного забезпечення. Можна сказати, що автоматизація тестування програмного забезпечення буде виправданою, якщо хоча б частково будуть виконуватись зазначені умови. В такому випадку автоматизація тестування дозволить значно знизити витрати на тестування, а відповідно й розробку програмного продукту. Також це дозволить покращити якість програмного забезпечення, що розробляється або підтримується. Під час регресійного тестування це може значно зменшити час на його виконання й аналіз виявлених помилок, якщо такі будуть.

РОЗДІЛ 2 ІНСТРУМЕНТИ ДЛЯ ТЕСТУВАННЯ ВЕБ-ДОДАТКІВ

Ці інструменти допомагають забезпечити якість, продуктивність та безпеку веб-додатків на всіх етапах розробки.

Досить складним завданням є вибір оптимального інструменту для тестування.

2.1 Основні критерії вибору інструментів

Треба враховувати важливі фактор, обираючи програми для тестування програмного забезпечення:

- вимоги до проектів:

потрібно розуміти вимоги до свого проекту, оскільки це перший фактор, який слід враховувати перед придбанням інструментів тестування;

- багаторазовість:

потрібне програмне забезпечення, яке має функцію тестування повторного використання. Це означає, що вам не доведеться постійно переписувати кроки тестування;

- легке створення тесту:

потрібно отримати пробну версію та переглянути її функції автоматизації, щоб перевірити функціональність створення тестів. Крім того, краще шукати інструменти, які керуються даними та мають можливість самовідновлення;

- бюджет:

враховуючи рівень свого бюджету, обирати інструмент відповідно до свого проекту та вимог. Також краще придбати платний інструмент, оскільки він має надійні функції, підтримку та розширені функції;

- звітність:

переконайтеся, що інструменти тестування програмного забезпечення надають точні та кілька способів звітування, як-от чіткі ознаки невдалих тестів або кроків, знімки екрана для невдалих кроків тощо;

- support:

інструменти тестування коду повинні забезпечувати належне навчання та підтримку 24/7. Постачальник також повинен передбачати підтримку співпраці та інтеграції. Згідно з нашим оглядом, такі інструменти, як Testrail, Zephyr Xray, Testpad і Testmo, настійно рекомендуються[11].

2.2 Топ - 3 інструменти у 2024 році

Інструменти з відкритим вихідним кодом підходять як для малих, так і для великих проектів; забезпечують високу якість продукту, дозволяючи командам зекономити ресурси та зосередитися на ефективності процесу тестування.

Інструменти тестування з відкритим вихідним кодом – це програми, які підтримують низку заходів тестування. Це програмне забезпечення дає змогу планувати, будувати, тестувати виконання, реєструвати дефекти тощо. Ці засоби тестування використовуються для перевірки стійкості та ретельності програмного забезпечення.

Нижче наведено кілька найкращих інструментів тестування з відкритим вихідним кодом із їхніми популярними функціями [12]:

1. Selenium.
2. Katalon Studio.
3. UFT One.

Дані засоби автоматизації досить впевнено вирішують проблеми тестування протягом вже тривалого часу, тому вони вважаються найкращими. Дані інструменти обираються за такими ознаками:

- підтримують тестування API та сервісів ;
- виконують паралельне виконання тестів - це скорочує час, який витрачається на виконання паралельних тестів;
- підтримують багато відомих мов програмування.

Розглянемо кожний з них, оцінимо їхні переваги та недоліки у таблиці 2.1.

Таблиця 2.1 - Порівняльна характеристика інструментів тестування

Інструмент тестування	Selenium	Katalon Studio	UFT One
Підтримка мов програмування	Java, .Net (C#), Python, Ruby, PHP, Perl, JavaScript	Java, Groovy	VBScript
Підтримує роботу на базі систем	Windows, macOS та Linux	Windows, Linux, OS X	Windows
Види тестування	Web app	Web, API, Mobile, Desktop apps	Web, Desktop, Mobile, RPA apps
Рік випуску	2004	2015	1998
Вартість використання на рік	400$	759$	3200$
Вимоги до навичок у програмуванні	Вимагаються навички переважно для інструментів інтеграції	Вимоги відсутні. Рекомендується для тестових скриптів	Вимоги відсутні. Рекомендується для тестових скриптів
Складність у використанні	Потрібні навички встановлення та використання	Легкий як у встановленні, так і у використанні.	Складно встановлюється, потрібні навички для використання

2.2.1 Характеристика інструменту автоматизації Selenium

Selenium взаємодіє з веб-браузерами, використовуючи вбудовану підтримку автоматизації браузера і надаючи набір інструментів і бібліотек, що да-

ють змогу тестувальникам і розробникам автоматизувати веб-браузери та моделювати взаємодію користувачів із веб-додатками.

Принципи роботи інструменту:

- тестувальник або розробник створює тестовий сценарій, використовуючи Selenium WebDriver API або будь-яку іншу мову програмування;

- API Selenium WebDriver надсилає команди у веб-браузер, використовуючи вбудовану підтримку автоматизації браузера, наприклад ChromeDriver або GeckoDriver;

- браузер отримує команди і виконує відповідні дії, наприклад натискання на посилання, заповнення форми або перехід на нову сторінку;

- браузер надсилає результат дії Selenium WebDriver;

- Selenium WebDriver зберігає результати і продовжує виконання наступної команди тестового сценарію;

- після завершення тестового сценарію Selenium генерує звіт із результатами тесту, включно з будь-якими помилками або збоями.

Selenium може автоматизувати веб-браузери, що працюють у різних операційних системах, як-от Windows, macOS і Linux, і підтримувати різні браузери, як-от Chrome, Internet Explorer, Safari і Firefox, а також може виконувати тести в кількох браузерах паралельно з використанням Selenium Grid. Автоматизуючи веб-браузери, Selenium дає змогу тестувальникам і розробникам швидко й ефективно виконувати автоматичні тести, забезпечуючи якість і надійність веб-додатків.

Для автоматизації тестування селен підтримує кілька мов програмування. Виберіть ту, яка відповідає навичкам вашої команди і вимогам проекту. Java і Python є популярним вибором через їхню велику підтримку та ресурси спільноти:

1. Встановлення Selenium: Selenium можна інтегрувати у ваш проект різними способами залежно від обраної вами мови програмування. Наприклад, якщо ви використовуєте Java, ви можете додати Selenium як залежність у свій проект Maven або Gradle.

2. Налаштування WebDriver: Selenium взаємодіє з веб-браузерами через WebDriver. Кожен браузер (наприклад, Chrome, Firefox) має свій власний WebDriver, який вам необхідно завантажити та налаштувати у своєму проекті.

3. Інтегроване середовище розробки (IDE): виберіть IDE, що підтримує обрану вами мову. Популярні IDE включають Eclipse і IntelliJ IDEA для Java і PyCharm для Python.

Вибрати можна будь-який, залежно від особистих уподобань.

Просунуті концепції Selenium:

1. Selenium Grid: для великомасштабного автоматизованого тестування селену і паралельного виконання використовується Selenium Grid – інструмент автоматизації тестування селену. Це дає змогу одночасно запускати кілька тестів у різних браузерах і операційних системах.

2. Інтеграція з платформами тестування: інтегруйте Selenium з платформами тестування, такими як JUnit або TestNG. Ці платформи надають структуру для написання тестів, затвердження умов і створення звітів.

3. Безперервна інтеграція (CI): інтегруйте тести Selenium у свій конвеєр CI/CD за допомогою таких інструментів, як Jenkins. Це гарантує автоматичний запуск тестів під час кожної збірки, що спрощує безперервне тестування.

4. Обробка помилок і налагодження: розвивайте навички обробки помилок і налагодження. Розуміння того, як читати трасування стека і журнали, має вирішальне значення для виявлення проблем у ваших тестових сценаріях.

Для ефективного використання Selenium необхідно розуміти загальні проблеми та методи їх вирішення:

- проблеми синхронізації: усуваються за допомогою відповідних стратегій очікування;

- сумісність браузера: потрібно регулярно оновлювати версії WebDriver і запускати тести сумісності, щоб забезпечити безперебійну роботу в різних браузерах;

- зміна веб-елементів: потрібно використовувати стабільні та надійні локатори, регулярно оновлювати свої тести, щоб відображати зміни в користувацькому інтерфейсі веб-додатка;

- підтримка тестування: потрібно регулярно проводити рефакторинг і перевірку тестових сценаріїв, видаляти застарілі тести й оновлювати наявні, щоб вони залишалися актуальними[13].

2.2.2 Характеристика інструменту Katalon Studio

Katalon studio – безкоштовний інструмент для автоматизації тестування від компанії Katalon Studio з графічним інтерфейсом. Може також застосовуватися для тестування API.

API (Application Programming Interface) – це інтерфейс для зв'язку компонентів програмного забезпечення.

Функціональне тестування проводиться на стороні користувацького інтерфейсу, а тестування API дозволяє обійти візуальний інтерфейс і з'єднатися безпосередньо з додатком або сайтом за допомогою відправки запитів серверу.

Працює це таким чином:

1. Клієнт, звертаючись до ресурсу, робить запит.

2. Запит відправляється на той сервер, який може виконати цей запит.

3. Сервер знаходить необхідний ресурс і відправляє відповідь назад клієнту.

Для безпосереднього тестування API-запитів існує безліч інструментів, але ми розглянемо Katalon studio, значною перевагою якого є можливість об'єднувати рівні UI і Business (служби API/Web) для різних операційних систем.

Основні плюси даного інструменту:

- безкоштовна програма, але без відкритого коду;
- підтримує як SOAP, так і REST API;
- легка у встановленні та обігу;
- малий поріг входження;
- має функціонал для тестування веб сервісів і REST API;
- дозволяє ділитися записаними тестами з колегами за допомогою самої програми;

- використовує вбудовану систему пошуку за ключовими словами для створення сценаріїв[14].

2.2.3 Характеристика інструменту UFT

QTP є інструментом автоматизованого функціонального тестування. Він виконує автоматизовані тести програми для виявлення різноманітних помилок.

Його розробник - Mercury Interactive, пізніше його придбала HP, а зараз-MicroFocus. Повна назва QTP – це «QuickTest Professional», у той час, як UFT означає «уніфіковане функціональне тестування».

UFT One, який раніше був відомий як UFT, зараз досить прогресивний комерційний інструмент. Тестує веб-програми, персональні настільні комп'ютери, додатки RPA та смартфони. Він має гарні можливості для тестування API, а також зручний вибір для тестування AUT, що працює в Інтернеті, на персональних комп'ютерах та на мобільних пристроях.

Підтримуючи кілька платформ для цільового тестованого додатка (AUT), UFT One пропонує:

- автоматизацію регресії та функціонального тестування додатків;
- тестер як технічного, так і нетехнічного характеру може використовувати Micro Focus QTP;
- забезпечує запис і відтворення;
- тестує однаково добре як настільні, так і веб-додатки;
- проводить тестування бізнес–процесів (BPT);
- тестування QTP засновано на скриптовій мові VB;
- для автоматизації додатків UFT від Micro Focus використовує VBScript;
- підтримує найбільший набір б'єктів серед розробки програмного забезпечення, таких як SAP, Oracle та ін.;
- інструмент QTP допомагає тестувальникам безперервно виконувати функціональне тестування.

Розглянемо порівняльні характеристики інструменту UFT у таблиці 2.2.

Таблиця 2.2 – Порівняльні характеристики інструменту UFT

Переваги автоматизації QTP	Недоліки автоматизації QTP
Підтримує різноманітні надбудови такі як Oracle, Java, SAP, NET, веб-форми, програмне забезпечення People та ін.	UFT не підтримує багатопоточність.
Використовує активний екран для запису сценаріїв та допомагає тестувальнику посилатися на властивості об'єкту екрану	Користувачеві потрібно поновити ліцензію для того, щоб отримати онлайн-підтримку від HP
Має відмінний процес ідентифікації об'єкту, що дозволяє покращити існуючі тести навіть через активний екран	Інтеграція UFT або пов'язування з іншими інструментами є дуже дорогими
Його можна інтегрувати з такими інструментами управління тестуванням, як Quality Center, Test Director і Winrunner.	У порівнянні з іншими інструментами, в UFT час виконання значно більший при меншій кількості сценаріїв
Постачається із вбудованим IDE	Імміграція даних в UFT є перешкодою
Підтримує популярні середовища автоматизації: підхід до тестування на основі ключових слів, модульний підхід до тестування, підхід до тестування на основі даних та ін.	UFT найбільш дорогий у порівнянні з іншими засобами автоматизації. Його ліцензія, а також додаткові збори за будь-які надбудови вартують дорожч
Простота в обслуговуванні	UFT працює не у всіх версіях браузерів

Нові функції, які включає остання версія QTP/UFT (уніфіковане функціональне тестування), наведені у таблиці 2.3.

Таблиця 2.3 - Можливості останньої версії QTP/UFT

Нові можливості	Докладніше
Підтримка OS і браузера	Підтримка Windows 10; виконує тестування у браузері Safari
Розширення UFT у магазині Chrome	Автоматичне отримання оновлень у Chrome
Підтримка об'єктів середовища виконання Windows	Можливість записувати та запускати тести на Windows, створювати репозиторії об'єктів за допомогою Windows
Нові технології, що підтримуються	СДК 1.8/ XenDesktop 7/ SAP Web Dynpro ABAP для NetWeaver 7.40 и т. ін.
Розширена підтримка різних технологій	Розпізнає об'єкти з веб-застосунків Siebel та взаємодіє з ними, SAP-додатки та ін.; Нові методи тестових об'єктів FlexTable доступні для розширених елементів керування сіткою даних.

Якщо ми потребуємо швидкого виведення великого проекту на ринок - інструменти автоматизації тестування є досить надійним вибором: вони дозволяють писати, керувати тестовими кейсами. Крім того, вони є відмінними помічниками командам, котрі використовують Agile-методології, DevOps або CI/CD.

Швидке та ефективне тестування програмного забезпечення відіграє вирішальну роль у життєвому циклі розробки.

Якісний інструмент допомагає командам економити дорогоцінний час і гроші, при цьому збільшуючи тестове покриття.

2.3 Висновки до розділу

Інструменти перфоманс-тестування є важливою складовою комплексного підходу до тестування програмного забезпечення. За допомогою цих інструментів тестувальники перевіряють реакцію програмного забезпечення на навантаження і стрес, виконуючи імітацію того, що відбувається при користуванні споживачем вашим продуктом.

На ринку є багато різних інструментів для тестування ефективності програмного забезпечення. Обрати вірного кандидата на посаду дуже нелегко, але у другому розділі ми описали інструменти для автоматизованого тестування; розглянули найголовніші критерії його вибору. Ми порівняли три найпопулярніші: проаналізували їхні переваги і недоліки, зробили оцінку їх характеристик.

РОЗДІЛ 3 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення (ПЗ) — це процес перевірки роботи системи для виявлення помилок, оцінки її якості та забезпечення відповідності заданим вимогам. Воно є невід'ємною частиною життєвого циклу розробки ПЗ, оскільки дозволяє гарантувати стабільність, безпеку та високу якість продукту.

3.1 Опис Web-додатку

У даному розділі описується сам Web-додаток, за допомогою якого, на основі технічної документації, буде складено тест – план; перевірка функціональності сайту; написання тест- кейсів.

Для тестування було обрано сайт комп’ютерної школи Hillel lms, що навчає таким сучасним технологіям як Java, FrontEnd, QA та іншим спеціальностям: https://lms.ithillel.ua/.

Hillel lms- це сайт, який має наступні основні функції: користувач сайту може переглядати перелік та опис курсів для навчання; розклад занять; має можливість авторизуватися, переглядати відео-уроки; спілкуватися в чаті та ін.

Проводитиметься чотири етапи процесу тестування:

- перший етап - складання тест-плану і чек-листа, із частковим використанням функціональних тестів;
- другий етап - тестування верстки сайту;
- третій етап - буде проведено автоматизовані і ручні тести;

Отже, завдяки всім етапам перевірки Web-додатку буде досягнуто максимальної ефективності тестування. Це, в свою чергу, дозволить виправляти дефекти і недоліки на ранніх етапах розробки.

ОС, що використовуватиметься під час тестування: Windows 10.

Браузер, що використовуватиметься під час тестування: Google Chrome – версія 131.0.6778.86 (Офіційна збірка), (64 бит).

Тестування безпеки і стрес-тестування не проводиться.

3.2 Інструменти для створення автотестів

Ми виберемо Selenium IDE — інструмент для автоматизованого тестування веб-додатків, який працює як плагін для браузерів (Google Chrome або Mozilla Firefox). Він дозволяє записувати, редагувати, виконувати і зберігати тестові сценарії без потреби у програмуванні.

Переваги використання Selenium IDE:

- простота у використанні;
- можливість запису без програмування;
- підтримка перевірки елементів і текстів;
- швидке тестування базових функцій сайту.

Selenium IDE підходить для швидкого створення тестів, але для складних тестових сценаріїв рекомендується використовувати Selenium WebDriver разом із мовами програмування (Java, Python, C# тощо).

3.3 Вибір мови сценаріїв для реалізації автотестів

Вибір мови програмування для тестування програм залежить від кількох факторів. Вони можуть варіюватися залежно від типу проекту, вимог команди, середовища тестування і навіть особистих уподобань тестувальників. Ось основні аспекти, які впливають на цей вибір:

Тип тестування.

Для UI-тестів часто обирають мови, які підтримують популярні фреймворки, такі як: Python (Selenium, Playwright), JavaScript (Cypress, Puppeteer), Java (Selenium).

Мова програмування проекту.

Тестові скрипти часто пишуть тією ж мовою, що й сам продукт, оскільки це спрощує інтеграцію тестів у проект і полегшує взаємодію між тестувальниками і розробниками.

Інструменти та фреймворки.

Доступність і зручність інструментів відіграють важливу роль: Selenium: підтримує Java, Python, C#, Ruby; Cypress: написаний на JavaScript, Playwright: підтримує Python, JavaScript, TypeScript, C#.

Складність і масштаб тестів.

Для швидкого написання простих тестів обирають мови з коротким синтаксисом і великою кількістю готових бібліотек. Для масштабних проектів із розгалуженими тестовими сценаріями можуть обрати мову з більш суворою структурою.

Інтеграція з DevOps.

Важливо врахувати, чи підтримується обрана мова CI/CD-інструментами (Jenkins, GitLab CI, Azure DevOps). Наприклад, Python і Java мають широкі можливості інтеграції з DevOps-циклами.

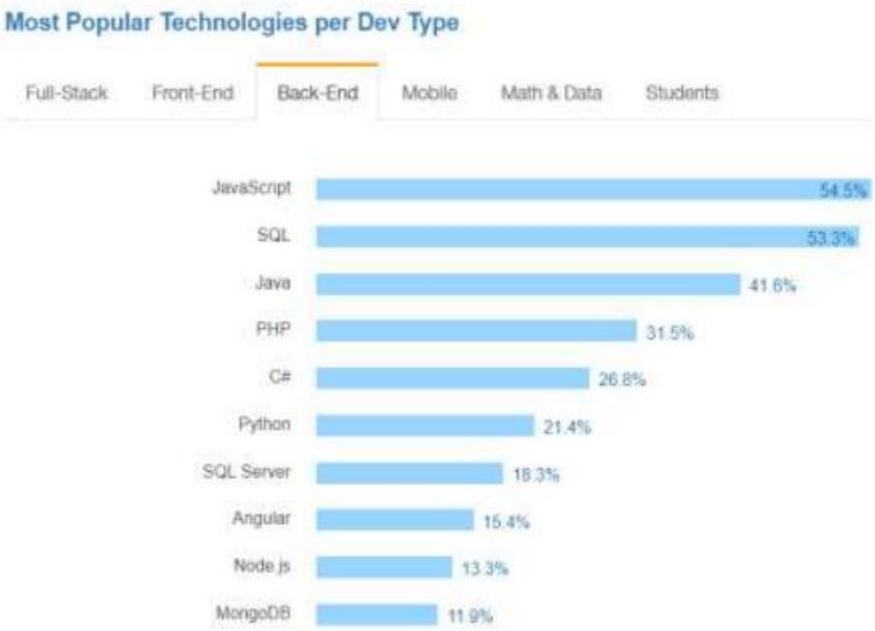
Отже, для більшості задач у тестуванні підходять Python (завдяки простоті і універсальності) та Java (завдяки підтримці великих корпоративних проєктів). Однак остаточний вибір залежить від особливостей проекту, наявних інструментів і досвіду команди.

Java є однією з найпопулярніших мов програмування, зі зрозумілим і простим синтаксисом. Вона підходить до самих різноманітних платформ. На Java пишуть майже весь софт: мобільні застосунки, промислове і наукове ПЗ, ігри, корпоративні рішення.

Згідно рейтингу IEEE, Java визнано лідером серед мов програмування; вона є найдешевшою та найпростішою для написання сайтів для різноманітних видів бізнесу, Інтернет-магазинів.

Java є третьою за популярністю мовою програмування, після JavaScript та SQL, згідно з StackOverflow (рис.3.1):

Рисунок 3.1 – Рейтинг популярності внутрішніх технологій



Загружаємо інсталяційний файл Java SE:
https://www.oracle.com/ua/java/

3.4 Опис документації

Після закінчення загрузки усіх додатків можна переходити до опису документації.

Перший етап складається зі складання чек-листа і тест-плану. Далі буде виконано частковий прогін функціональних тестів.

3.4.1 Чек лист як форма систематизації роботи

Чек-лист у тестуванні – це перелік дій, умов або сценаріїв, які потрібно виконати чи перевірити для того, щоб упевнитися в якості програмного продукту. Він допомагає забезпечити покриття критичних аспектів тестування і уникнути пропусків важливих функцій чи сценаріїв.

Складемо чек-лист для перевірки сайту онлайн-школи Hillel lms**,** який охоплює різні аспекти функціональності, продуктивності та зручності використання. Чек-лист (Таблиця 3.1) розділено на категорії для зручності.

Таблиця 3.1 – Чек-лист для перевірки сайту онлайн-школи Hillel lms

Перевірка	Коментар	Статус
1. Головна сторінка	Перевірити доступність головної сторінки сайту.	пройшов
	Верифікувати працездатність форми входу (логін і пароль): коректні облікові дані; некоректні дані (неправильний логін/пароль); порожні поля.	пройшов
2. Перевірити відновлення пароля	Введення існуючого e-mail..	пройшов
	Введення неіснуючого e-mail або порожнє поле	пройшов
3. Перевірити навігацію по сайту	Тестувати можливість завантаження файлів до LMS (домашні завдання, документи). Верифікувати відображення розкладу занять.	пройшов
	Перевірити, чи надходить підтвердження на email.	пройшов
4. Перфоманс	Виміряти швидкість завантаження сторінки (Google PageSpeed Insights).	пройшов
	Перевірити час завантаження головної сторінки при низькій швидкості Інтернету.	пройшов
	Перевірити, чи немає зависань під час навігації по сайту.	пройшов
5. Зручність використання (UI/UX)	Перевірити коректність відображення елементів сайту (меню, кнопки, поля вводу).	пройшов
	Перевірити наявність помилкових повідомлень при введенні некоректних даних.	пройшов
	Перевірити мову інтерфейсу (українська/російська/англійська, якщо доступно).	пройшов

Продовження таблиці 3.1

6. Продуктивність	Перевірити швидкість завантаження головної сторінки.	пройшов
	Перевірити час відгуку сервера при бронюванні квитка.	пройшов
	Перевірити стабільність роботи сайту під навантаженням.	пройшов
7. Безпека	Чи шифруються дані під час авторизації (перевірка HTTPS)?	пройшов
	Чи можна здійснити успішну авторизацію без валідних даних?	пройшов
	Чи блокує система багаторазові невдалі спроби авторизації?	не пройшов
8. Кросбраузерність	Перевірити роботу сайту в Google Chrome, Mozilla Firefox, Safari, Microsoft Edge.	пройшов
9. Тестування інтеграцій	Перевірити коректність роботи на різних версіях браузерів.	не пройшов
	Перевірити роботу інтеграції з поштовими сервісами (підтвердження покупки).	пройшов
	Перевірити інтеграцію з API сеансів та бронювання.	пройшов

Даний чек-лист підходить для проведення мануального тестування. Для пунктів, які не пройшли перевірку, потрібно фіксувати баги (невідповідність очікуваному результату) в системі управління (наприклад, Jira).

3.4.2 Тест план як стратегія тестування ПЗ

Тест-план – це документ, який описує стратегію, підхід, обсяг і графік проведення тестування програмного продукту. Він допомагає організувати процес тестування, визначити його цілі, а також розподілити ролі та ресурси між учасниками.

Тест-план для сайту комп’ютерної школи Hillel lms надано у таблиці 3.2.

Таблиця 3.2 – Інформація про створення тест-плану

Дата	Версія	Вид змін	Автор
26.11.2024	1.0	Створення	Кравець О.В.

Опис тест-плану включає основні розділи, які визначають стратегію, мету, обсяг, методи та інші аспекти тестування. В описанні даного тест-плану присутні такі пункти:

1) Вступ (Introduction);

2) Мета документа;

3) Вихідні дані;

4) Мета тестування.

Типи тестування (Types of Testing): функціональне тестування, тестування кросбраузерності, регресивне тестування.

Мета складання даного тест-плану полягає в описі процесу перевірки сайту Hillel lms (адреса сайту: https://lms.ithillel.ua/). Опис надає можливість виконати практичну роботу з тестування сайту.

Вихідні дані.

Hillel lms - сайт комп’ютерної школи, що навчає сучасним технологіям: Java, FrontEnd, QA та іншим спеціальностям.

Мета документа.

Метою тестування сайту Hillel lms є забезпечення стабільної, функціональної та безпечної роботи сайту онлайн-школи, що включає тестування основних функцій, перевірки навантаження та продуктивності, сумісності з різними платформами, а також дотримання стандартів безпеки й зручності для кінцевих користувачів.

У процесі тестування створюється низка документів і звітів, які відображають виконану роботу, виявлені проблеми та загальну якість програмного продукту. Ці результати дозволяють команді оцінити якість продукту та прийняти рішення про його готовність до запуску.

Умови тестування.

Web-сайт повинен всебічно задовольняти потреби користувачів у таких активностях, як можливість перегляду переліку та опису курсів для навчання; розкладу занять; можливість авторизуватися, переглядати відео-уроки; спілкуватися в чаті та ін.

Правильно підготовлені умови забезпечують точність і ефективність тестування, що є запорукою якісного релізу сайту кінотеатру.

Стратегія процесу тестування.

Проводитимуться наступні етапи процесу тестування:

а) складання чек-листа і тест-плану

б) тестування верстки сайту

в) проходження авто тестів

г) тестування зручності користування Web-сайтом та рекомендації щодо покращення.

ОС, що використовуватиметься під час тестування: Windows 10.

Браузер, який буде використовуватиметься під час тестування: Google Chrome.

Типи тестування.

1.Функціональне тестування.

Мета: перевірити, чи працюють основні функції сайту відповідно до вимог та специфікацій, гарантувати, що користувачі можуть використовувати ключові функції.

2. Тестування кросбраузерності.

Мета: перевірка коректної роботи проекту, будемо перевіряти на браузері. Google Chrome.

3. Регресивне тестування.

Мета: переконатися у збереженні працездатності існуючого функціоналу, виявити побічні ефекти внесених змін, підтвердити стабільність системи, гарантувати якість продукту перед релізом.

Перевіримо тільки його працездатність, адже ми тестуємо вже абсолютно функціональний сайт.

План робіт показано у таблиці 3.3.

Таблиця 3.3 – План робіт з тестування сайту

Задача	Об’єм роботи	Дата початку	Дата закінчення
Складання тест-плану	3 години	24.11.2024	24.11.2024
Виконання тестування	10 годин	25.11.2024	26.11.2024
Аналіз тестування	1 година	26.11.2024	26.11.2024
Підведення підсумків	2 години	27.11.2024	27.11.2024

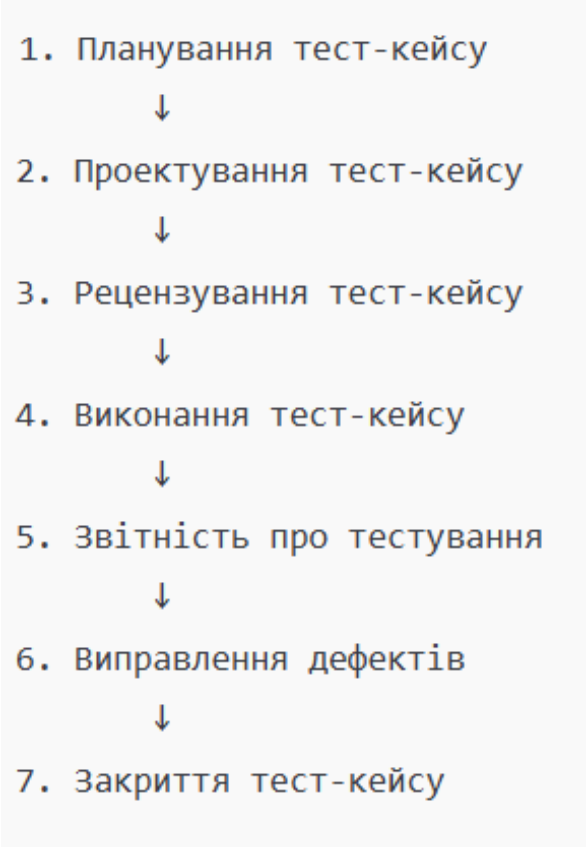
Кінцеві результати тестування відображають загальний стан програмного продукту після проходження всіх етапів тестування. Це дозволяє команді розробників та зацікавленим сторонам оцінити готовність продукту до запуску.

3.4.3 Тест-кейс як тестовий артефакт

Тест-кейс — це документ, який описує конкретну перевірку певної функціональності або компонента системи. Тест-кейси використовуються для планування та організації тестування, щоб упевнитися, що програмне забезпечення працює відповідно до вимог.

Тест-кейси важливі для структурування процесу тестування, забезпечення покриття всіх функціональних вимог, відслідковування помилок і дефектів у процесі тестування, визначення результатів тестування (чи працює система згідно з вимогами). Вони допомагають організувати роботу тестувальників і забезпечити належний рівень якості продукту. Схему життєвого циклу тест-кейсу показано на рис. 3.2.

Рис. 3.2 – Схема життєвого циклу тест-кейсу



Ключові моменти життєвого циклу тест-кейсу:

- кожен етап життєвого циклу тест-кейсу має свою мету та завдання;
- важливо, щоб тест-кейси були чіткими, детальними та перевіреними, щоб уникнути помилок під час тестування.
- цей цикл допомагає забезпечити високу якість продукту і вчасне виявлення дефектів;

Для сайту Hillel lms створимо певну кількість тест-кейсів, опишемо найбільш актуальні кроки тестування. Розглянемо усі сценарії: як позитивні, так і негативні.

Опис функціональності логіну показано у таблиці 3.4.

Таблиця 3.4 – Функціональність логіну

ID	Назва	Опис	Кроки	Очікуваний результат
1	Логін з коректними даними	Перевірка успішного входу з валідними обліковими даними	1. Відкрити сайт. 2. Ввести коректний логін. 3. Ввести коректний пароль. 4. Натиснути кнопку "Увійти".	Користувач успішно входить у систему, відображається головна сторінка LMS.
2	Логін з некоректними даними	Перевірка роботи з невалідними обліковими даними	1. Відкрити сайт. 2. Ввести неправильний логін або пароль. 3. Натиснути кнопку "Увійти".	Відображається повідомлення про помилку: "Некоректний логін або пароль".
3	Логін з порожніми полями	Перевірка роботи системи при відсутності введених даних у поля логіну та паролю	1. Відкрити сайт. 2. Натиснути кнопку "Увійти", не заповнюючи поля.	Відображається повідомлення про необхідність введення логіну та паролю.

Опис функціональності відновлення пароля показано у таблиці 3.5.

Таблиця 3.5 – функціональність відновлення пароля

ID	Назва	Опис	Кроки	Очікуваний результат
1	Успішне відновлення пароля	Перевірка відправлення посилання на відновлення пароля	1. Натиснути "Забули пароль?". 2. Ввести валідний e-mail. 3. Натиснути кнопку "Відновити".	На e-mail надходить посилання для відновлення пароля.
2	Відновлення пароля з неіснуючим e-mail	Перевірка реакції системи на введення e-mail, що не зареєстрований	1. Натиснути "Забули пароль?". 2. Ввести неіснуючий e-mail. 3. Натиснути кнопку "Відновити".	Відображається повідомлення: "Користувач із таким e-mail не зареєстрований".

Опис навігації по сайту показано у таблиці 3.6.

Таблиця 3.6 – Навігація по сайту

ID	Назва	Опис	Кроки	Очікуваний результат
1	Перевірка роботи меню	Перевірити, чи всі посилання в меню працюють	1. Перейти на головну сторінку. 2. Натиснути на кожен пункт меню.	Всі посилання відкривають відповідні сторінки без помилок.
2	Робота кнопки "Вихід"	Перевірити, чи працює кнопка виходу з системи	1. Увійти у систему. 2. Натиснути кнопку "Вийти".	Користувач успішно виходить із системи, і його перенаправляє на сторінку логіну.

Опис розділу «Домашні завдання» показано у таблиці 3.7.

Таблиця 3.7 – Домашні завдання

ID	Назва	Опис	Кроки	Очікуваний результат
1	Завантаження файлу завдання	Перевірка, чи можна завантажити файл до системи	1. Перейти до сторінки домашнього завдання. 2. Натиснути "Завантажити файл". 3. Вибрати файл.	Файл успішно завантажується, і відображається статус "Файл завантажено".
2	Видалення файлу із завдання	Перевірка, чи можна видалити завантажений файл	1. Завантажити файл. 2. Натиснути кнопку "Видалити" біля файлу.	Файл успішно видаляється, і статус завдання змінюється на "Файл не завантажено".
3	Завантаження невалідного файлу	Перевірка реакції системи на завантаження файлу неправильного формату	1. Перейти до сторінки домашнього завдання. 2. Завантажити файл із неправильним форматом.	Відображається помилка: "Формат файлу не підтримується".

Опис кросбраузерного тестування показано у таблиці 3.8.

Таблиця 3.8 – Кросбраузерне тестування

ID	Назва	Опис	Кроки	Очікуваний результат
1	Перевірка відображення даних	Перевірити, чи розклад відображається правильно	1. Увійти у систему. 2. Перейти до розкладу занять.	Відображається актуальний розклад занять із правильною датою і часом.

Дані тест-кейси можна використовувати як базу для ручного або автоматизованого тестування сайту Hillel lms.

3.5 Автоматизоване тестування системи веб-додатку

Selenium є проектом, у рамках якого розробляється серія програмних продуктів із відкритим вихідним кодом:

- Selenium WebDriver,
- Selenium Grid,
- Selenium IDE.

Розглянемо процес тестування веб-сайту https://lms.ithillel.ua/ за допомогою Selenium IDE.

Встановлення розміру вікна:

- Command: *set window size;*
- Target: Встановлення розміру вікна браузера на 926x697 пікселів;
- Мета: Забезпечення стабільності тестів для різних елементів на сторінці.

Наведення на елемент (mouse over):

- Command: *mouse over;*
- Target: Елемент із селектором css=.menu-aside__selection-box;
- Мета: Перевірка поведінки сайту при наведенні курсору на певний елемент (наприклад, відкриття підменю).

Прибирання курсору з елемента (mouse out):

- Command: *mouse out;*
- Target: Той самий елемент css=.menu-aside__selection-box;
- Мета: Перевірка коректного повернення елемента до початкового стану після того, як курсор покинув область.

Клік по елементу:

- Command: *click;*
- Target: css=.menu-aside__link--store > .menu-aside__icon > .ng-tns-c144-1:nth-child(2);
- Мета: Емуляція кліку на елемент (наприклад, відкриття підменю чи сторінки).

Перехід до активного елемента:

- Command: *click;*
- Target: css=.store-page-aside-list__link--active > .ng-tns-c292-3:nth-child(2);
- Мета: Натискання на посилання, яке стало активним після попередніх дій.

Додатковий клік:

- Command: *click;*
- Target: css=.store-page-aside-list__item:nth-child(2) .ng-tns-c292-3:nth-child(2);
- Мета: Тестування реакції сайту на взаємодію з підпунктами меню.

Перевірка тексту (assert text):

- Command: *assert text;*
- Target: css=.store-page-category-header;
- Value: Безкоштовні спринти;

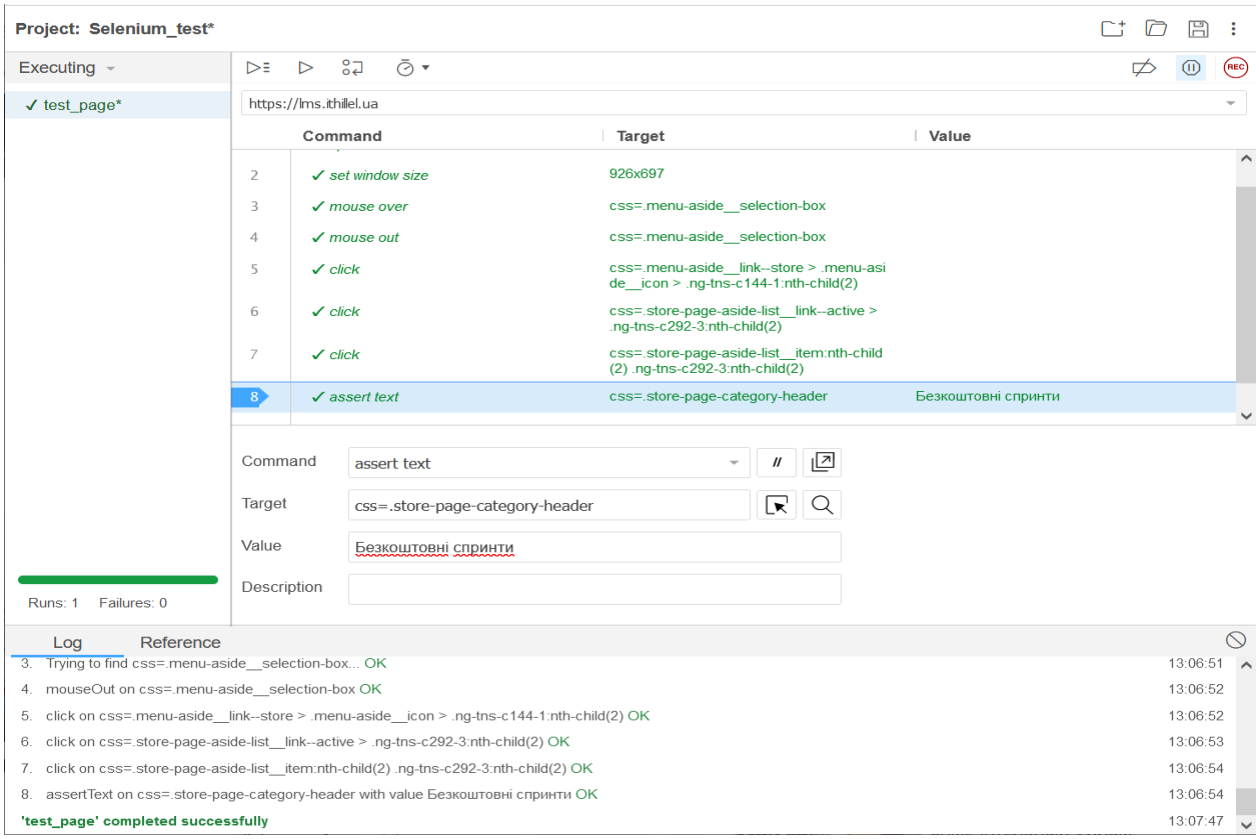
-Мета: Перевірка відповідності тексту на сторінці очікуваному результату. У цьому випадку перевіряється, чи відображається заголовок "Безкоштовні спринти".

Завершення тесту:

- Результат: Усі кроки тесту виконалися успішно (позначені як OK). Результат перевірки тесту показано на рисунку 3.3;
- Висновок: Сайт працює згідно з очікуваними функціональними вимогами для цього сценарію.

На рисунку 3.3 бачимо результати перевірки тесту за допомогою Selenium IDE.

Рис. 3.3 - Результат перевірки тесту за допомогою Selenium IDE



Цей тестовий сценарій можна використовувати для: перевірки коректної роботи меню та підменю; тестування інтерактивних елементів на сторінці; перевірки візуального відображення тексту на веб-сторінці.

3.6 Мануальне тестування системи веб-додатку

Мануальне тестування передбачає виконання перевірки продукту без участі програмного забезпечення, тобто, вручну. Це означає, що весь процес тестування здійснюється конкретною людиною, яка, проводить тести та фіксує результати.

У ручному тестуванні використовуються різноманітні інструменти, які допомагають тестувальникам ефективно організовувати свою роботу, фіксувати результати тестування та забезпечувати якість програмного забезпечення. Ось основні категорії та приклади інструментів:

а) інструменти для відслідковування багів (Bug Tracking). Дозволяють документувати знайдені помилки та відстежувати їхнє виправлення:

- Jira – популярний інструмент для трекінгу задач і багів.
- Bugzilla – потужний та безкоштовний трекер багів.
- Redmine – управління проектами та відслідковування помилок.
- MantisBT – простий у використанні трекер багів.

б) Інструменти для управління тестами. Ці інструменти допомагають створювати, організовувати та відслідковувати тест-кейси, тест-плани та звіти:

- TestRail – управління тест-кейсами та звітами.
- Zephyr (плагін до Jira) – інтеграція з Jira для управління тестами.
- Test – управління всіма етапами тестування.
- PractiTest – хмарний інструмент для управління тестуванням.

в) інструменти для тестування API. Ручне тестування API допомагає перевірити взаємодію клієнта і сервера:

- Postman – для виконання API-запитів.
- Insomnia – інший інструмент для тестування REST API.
- Swagger – перевірка API через його документацію.

г) інструменти для управління тестами. Ці інструменти допомагають створювати, організовувати та відслідковувати тест-кейси, тест-плани та звіти:

- TestRail – управління тест-кейсами та звітами.
- Zephyr (плагін до Jira) – інтеграція з Jira для управління тестами.
- qTest – управління всіма етапами тестування.
- PractiTest – хмарний інструмент для управління тестуванням.

Ці інструменти допомагають значно підвищити ефективність ручного тестування, зменшити кількість рутинної роботи й покращити якість звітів про результати тестування.

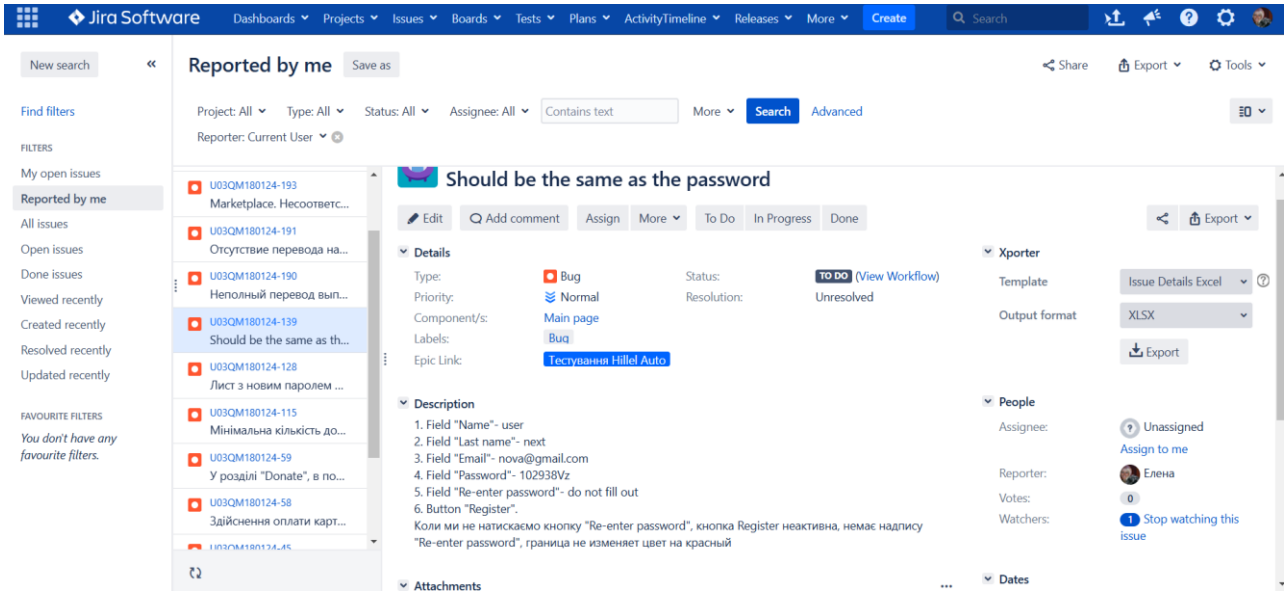
Розглянемо процес тестування веб-сайту https://lms.ithillel.ua/ за допомогою Jira Software - популярного інструменту для трекінгу задач і багів.

Jira — це інструмент для управління проектами та завданнями, розроблений компанією Atlassian. Він широко використовується для:

- трекінгу завдань: створення, призначення та відстеження прогресу задач;
- управління проектами: підтримка методологій Scrum, Kanban та Agile;
- тестування і баг-трекінг: зручне ведення звітів про помилки (bugs) і їхній статус;
- командна робота: полегшує співпрацю завдяки гнучким інструментам фільтрації, спільного доступу та комунікації.

На рисунку 3.4 видно інтерфейс Jira Software, що використовується для управління завданнями та процесом тестування.

Рисунок 3.4 – Результат тестування у Jira Software



Опис процесу тестування:

Мета тестування: Перевірка поля "Re-enter password".

Кроки тестування:

- заповнення полів "Name", "Last name", "Email" і "Password";
- поле "Re-enter password" залишене порожні;
- натискання кнопки "Register".

Очікуваний результат:

- кнопка "Register" має бути неактивною;
- повідомлення про необхідність повторного введення пароля або зміна кольору межі поля на червоний для підсвічення помилки.

Результат:

- поведінка не відповідає очікуванням;
- поле "Re-enter password" не виконує перевірку, кнопка активна, а візуального попередження немає.

Поточний статус:

- завдання має статус "To Do" (ще не розпочато);
- проблему зареєстровано як Bug (Баг) зі звичайним пріоритетом.

Баг-трекінг — це процес виявлення, реєстрації, відстеження та виправлення програмних помилок (багів).

Основні етапи:

- виявлення: знаходження помилки під час тестування чи використання;
- реєстрація: документування багу в спеціальних системах (наприклад, Jira), із зазначенням деталей;
- пріоритизація: визначення важливості та впливу помилки;
- виправлення: усунення проблеми розробниками;
- перевірка: перетестування, щоб переконатися, що баг виправлено.

Це важлива частина процесу забезпечення якості мануального тестування (QA) у розробці ПЗ.

3.7 Висновки до розділу

У третьому розділі було розглянуто процес тестування програмного забезпечення на прикладі веб-додатку https://lms.ithillel.ua/, що включав опис застосованих методів і інструментів для забезпечення якості продукту.

Опис веб-додатку. Система **LMS Hillel** була охарактеризована як сучасний веб-додаток для управління навчальним процесом, що вимагає детального тестування для забезпечення його надійності, зручності та функціональності.

Інструменти для створення автотестів. Було обґрунтовано вибір **Selenium IDE** як інструменту для автоматизації тестування, оскільки він забезпечує легкість у використанні, можливість запису сценаріїв тестів і підтримку різних браузерів.

Вибір мови сценаріїв. Вибір мови програмування був обґрунтований з огляду на її сумісність з інструментами автоматизації. Наприклад, Java, як одна з основних мов, забезпечила ефективність і зручність написання автотестів.

Опис документації:

- чек-листи забезпечили систематизацію та чітку структуру перевірок;
- тест-план визначив стратегію, обсяги і завдання тестування;
- тест-кейси деталізували кроки перевірки функціоналу веб-додатку.

Автоматизоване тестування. Проведено тестування за допомогою **Selenium IDE**, що дозволило виконати регресійні тести і перевірити функціональність веб-додатку на основі автоматизованих сценаріїв.

Мануальне тестування. Використовуючи **Jira Software**, було організовано процес ручного тестування, включаючи створення та відстеження дефектів. Це дало змогу виявити критичні баги, які могли вплинути на користувацький досвід.

Загальний висновок. Застосування комбінованого підходу до тестування, що включає автоматизовані та мануальні методи, дозволило забезпечити комплексну перевірку веб-додатку. Використання чек-листів, тест-плану і тест-кейсів сприяло систематизації процесу тестування, а вибір інструментів

Selenium IDE і Jira Software гарантував якісну організацію робіт. Такий підхід допоміг оцінити якість програмного забезпечення, виявити критичні недоліки та надати рекомендації щодо їх усунення.

ВИСНОВКИ

У ході виконання дипломної роботи на тему **«Методи та засоби тестування для оцінювання показників якості програмного продукту»** було проведено ґрунтовний аналіз теоретичних основ, інструментів та практичних підходів до тестування програмного забезпечення, а також здійснено апробацію методів тестування на реальному веб-додатку.

1. Аналіз методів тестування

У першому розділі було розглянуто основні типи та методи тестування програмного забезпечення, зокрема:

- Функціональне, нефункціональне та регресійне тестування;

- Методи «чорної», «білої» та «сірої скриньки», що дозволяють забезпечити всебічний підхід до перевірки ПЗ;

- Оцінено рівні тестування, включаючи модульне, інтеграційне та системне тестування, що є основою ефективного процесу контролю якості.

На основі проведеного аналізу було визначено завдання дослідження — порівняння інструментів та методів для створення системи тестування веб-додатків.

2. Вибір інструментів для тестування

У другому розділі було проаналізовано інструменти для тестування веб-додатків і надано характеристику топ-3 рішень у 2024 році. Основна увага була зосереджена на:

- Selenium, як найпопулярнішому інструменті автоматизації з відкритим вихідним кодом.

- Katalon Studio, що пропонує інтегровані рішення для тестування.

- UFT, що забезпечує широкі можливості для автоматизації корпоративних додатків.

Проведено порівняння інструментів за критеріями функціональності, зручності використання, підтримки та продуктивності, що дозволило обрати оптимальне середовище для тестування веб-додатків.

3. Практичне тестування веб-додатку

У третьому розділі виконано тестування веб-додатку https://lms.ithillel.ua/. Основні аспекти:

- проведено автоматизоване тестування з використанням Selenium IDE, що дозволило перевірити ключові сценарії роботи системи;

- використання Jira Software дало змогу ефективно організувати процес мануального тестування та управління дефектами;

- документація (чек-листи, тест-план, тест-кейси) систематизувала тестування та полегшила відстеження результатів;

- на основі автоматизованих і мануальних тестів було виявлено та описано критичні дефекти веб-додатку.

Значення отриманих результатів

Виконані роботи дозволили:

- розробити підхід до комплексного тестування, який може бути використаний для інших проектів;

- сформувати рекомендації для покращення функціональності та надійності веб-додатку;

- підтвердити ефективність комбінованого підходу до тестування із застосуванням автоматизації та ручного аналізу.

Загальний підсумок

Результати дослідження підтвердили важливість системного підходу до тестування програмного забезпечення, зокрема для веб-додатків. Вибір відповідних методів і інструментів, таких як Selenium IDE і Jira Software, забезпечує ефективну організацію процесу тестування, виявлення дефектів і підвищення загальної якості продукту.

Практичне значення дипломної роботи полягає в її застосовності для реальних проектів, що вимагають всебічної перевірки програмного забезпечення з використанням сучасних інструментів.

ПЕРЕЛІК ПОСИЛАНЬ

1. [Електронний ресурс]. – Режим доступу: https://avada-media.ua/services/urovni-i-metody-testirovaniya-po/.

2. Канер Кем, Фолк Джек, Нгуен Енг Кек. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес- приложений. – Киев: Диа-Софт, 2001. – 544 с.)

3. Forgács, István; Kovács, Attila (2019). Практичне проектування тестів: Вибір традиційних та автоматизованих методів проектування тестів. ISBN 1780174721.

4. Milind G. Limaye (2009). Software Testing. Tata McGraw-Hill Education. с. 216. ISBN 978-0-07-013990-9.

5. [Електронний ресурс]. – Режим доступу: https://qalight.ua/baza-znaniy/white-black-grey-box-testuvannya/.

6. [Електронний ресурс]. – Режим доступу: https://uk.itpedia.nl/2018/02/05/white-box-testing-onder-de-loep/.

7. Башмаков А.И. Интеллектуальные информационные технологии: Учеб. пособие. / А.И. Башмаков, И.А. Башмаков. – М.: Изд-во: МГТУ им. Н.Э. Баумана, 2005. – 304 с.

8. [Електронний ресурс]. – Режим доступу: https://uk.myservername.com/alpha-testing-beta-testing/.

9. [Електронний ресурс]. – Режим доступу: https://www.it-notes.wiki/software-testing/non-functional-testing/.

10. [Електронний ресурс]. – Режим доступу: https://dan-it.com.ua/uk/blog/vidy-funkcionalnogo-i-nefunkcionalnogo-testirovanija/.

11. [Електронний ресурс]. – Режим доступу: https://www.guru99.com/uk/testing-tools.html.

12. [Електронний ресурс]. – Режим доступу: https://www.guru99.com/uk/best-open-source-testing-tools.html.

13. [Електронний ресурс]. – Режим доступу:

https://foxminded.ua/selenium/.

14. [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/api-testing-tools/.

15. [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/selenium-webdriver/.

16. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Apache_Maven/.

17. [Електронний ресурс]. – Режим доступу: https://www.guru99.com/uk/manual-testing-tools.html/.