МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт

з дисципліни «Якість програмного забезпечення та тестування»

для студентів спеціальностей

121 «Інженерія програмного забезпечення»

122 «Компютерні науки»

усіх форм навчання

2025

Методичні вказівки до виконання лабораторних робіт з дисципліни «Якість програмного забезпечення та тестування» для студентів спеціальностей 121 «Інженерія програмного забезпечення», 122 «Компютерні науки» усіх форм навчання /Укл.: В.Г. Костюк, О.В. Шитікова. – Запоріжжя: НУ «Запорізька політехніка», 2025. – 101 с.

Укладачі: В.Г. Костюк, асистент кафедри ПЗ,

О.В. Шитікова, доцент кафедри ПЗ.

Рецензент: А.О.Олійник, д.т.н., професор кафедри ПЗ.

Відповідальний
за випуск: С.О.Субботін, д.т.н., зав. кафедри ПЗ.

Затверджено
на засіданні кафедри
«Програмні засоби»
Протокол № 1
від «18» серпня 2025 р.

Рекомендовано до видання
НМК Факультету
комп'ютерних наук і технологій
Протокол № 2
від «10» вересня 2025 р.

ЗМІСТ

ЗАГАЛЬНІ ПОЛОЖЕННЯ

Методичні вказівки представляють собою необхідний засіб для успішного вивчення теоретичного матеріалу та практичного освоєння тестування програмного забезпечення в рамках дисципліни «Якість програмного забезпечення та тестування».

Мета роботи – створити підґрунтя для оволодіння усіма концепціями тестування програмного забезпечення: тестування вебпроєктів, функціональне тестування, тестування мобільних додатків, тестування ігор тощо. Лабораторний курс складається з дванадцяти робіт. Кожна робота виконується та здається студентом індивідуально. Студенти, що не підготовлені до роботи, а також, які не мають вірно оформленого звіту, до занять не допускаються.

Вимоги до оформлення та змісту звіту, а також контрольні запитання, надано в кожній лабораторній роботі. Студент повинен знати мету роботи, теоретичні відомості, методи та засоби тестування програмного забезпечення, підходи проведення різноманітних видів тестування.

Студент, який не здав попередньої роботи, не допускається до виконання наступних.

Звіт має містити:

– титульний аркуш (на ньому вказують назву міністерства, назву університету, назву кафедри, номер, вид і тему роботи, виконавця та особу, що приймає звіт, рік). Приклад наведено у додатку В;

– тему та мету роботи;

– завдання до роботи;

– лаконічний опис теоретичних відомостей згідно тематики;

– результати виконання лабораторної роботи;

– змістовний аналіз отриманих результатів та висновки.

Звіт виконують на білому папері формату А4 (210 × 297 мм). Текст розміщують тільки з однієї сторони листа. Розмір елементів тексту – 14пт. Міжрядковий інтервал – 1,5. Поля сторінки згори та знизу – 20мм, ліворуч – 30мм, праворуч – 10мм. Аркуші скріплюють за допомогою канцелярських скріпок або вміщують у канцелярський файл, але з’єднання листів стиплером категорично не допускається.

1 ЛАБОРАТОРНА РОБОТА №1
СТВОРЕННЯ БАГ-РЕПОРТІВ

Мета лабораторної роботи: отримання базових теоретичних та практичних навичок в розпізнаванні етапів тестування та створенні баг-репортів

1.1 Короткі теоретичні відомості

Баг полягає в основній ідеї тестування та характеризується як невідповідність фактичної роботи програмного забезпечення очікуваному результату, тобто технічному завданню чи специфікації від замовника. Перевірка на баги відбувається не тільки безпосередньо вже на етапі тестування готового продукту, а й на етапі аналізу техзавдання. Під тестування може потрапляти як інтерфейс у зручності користування, так і точність дизайну продукту, або функціональна складова бекенду програми. **Кожна помилка системи має бути задокументована.** І звіт, в якому вказані всі баги, називається баг-репорт. У ньому вказується детальний опис помилки, що сталася на етапі тестування та всі дії, що до неї призвели [1].

Кожен баг-репорт складається з кількох важливих атрибутів, які обов'язково вказуються у ньому:

– **Саммарі** (стислий опис помилки в одному рядку) – будується за принципом: що не працює? де не працює? коли? чи за яких умов не працює? Наприклад, «Не здійснюється оплата, після натискання на кнопку «Купити» на сторінці купівлі товару».

– **Опис помилки** (кроки відтворення помилки), має бути коротким і ясним програмісту. Тут вказуються всі кроки, тобто умови, виникнення бага, фактичний результат та очікуваний результат роботи програми.

– **Проєкт** – вказується найменування програми, сайту чи іншої програми, до якої належить наш баг, для подальшого відстеження усунення цього дефекту.

– **Компонент** – під ним слід розуміти частину програми, або ділянку розробки, в якій було допущено баг. Якщо у різних частинах продукту є схожі ділянки коду, рекомендується перевірити один баг на кількох ділянках розробки. Наприклад, каталог інтернет-магазину

передбачає наявність фільтрів для деталізації пошуку певного товару, але також на різних категоріях товару є ймовірність ділянки розробки коду, що повторюється, для фільтрів, які можуть бути просто скопійовані розробником. Якщо є баг в одній частині, він неодмінно буде і в іншій ділянці.

– **Серйозність** – тобто критичність помилки, або наскільки дана помилка впливає на працездатність нашого програмного забезпечення. Найчастіше це – несуттєвий (Minor), значний (Major), критичний (Critical) чи блокуючий (Blocker).

– **Пріоритет** – послідовність виправлення багів, виходячи з попереднього атрибуту бага. Зазвичай це високий (High), середній (Medium) і низький (Low). Важливо відзначити, що пріоритет тестувальник не проставляє – цим займається менеджер або той, хто займається розподілом завдань на програмістів.

– **Статус** – це фіксація етапу усунення бага у його життєвому циклі. Наприклад, відкритий (Open), закритий (Closed), новий (New) або перевідкритий (Reopen).

– **Версія білда**, тобто версія продукту, в якій виявлена помилка. Іноді цей атрибут може бути не вказаний, якщо немає версії продукту або баг відтворюється у всіх версіях програмного забезпечення.

– **Автор** – той учасник команди, який вносить інформацію про баг.

– **Оточення** – під це поняття може потрапляти ОС, розряд системи, браузер та його версія, у якому було виявлено помилку [2].

Одні з найважливіших атрибутів кожної помилки в баг-репорті є **серйозність і пріоритет**. І якщо рівень серйозності вказує тестувальник, то за пріоритетність їх усунення відповідає PO (Product Owner) з боку замовника або проджект менеджер (PM), оскільки є незначні помилки, які не ламають додаток, наприклад, найменування інтернет-магазину, в якому допущена помилка, але їх усунення першочергово важливе для престижу компанії замовника.

Класифікації ступеня серйозності багів та його вплив на працездатність програмного забезпечення.

Blocker – та помилка, яка призводить до непрацездатності нашої програми. Тобто її подальше використання користувачем стає неможливим.

Critical – є кілька варіантів: відхилення від логіки роботи програмного забезпечення, задуманого замовником; неробочий функціонал окремих елементів ПЗ (наприклад, відсутність переходу в кошик після натискання кнопки «Купити» на сайті інтернет-магазину); не збереження даних сеансу або користувача.

Major – є схожість з критичними помилками, проте в даному випадку програма працює, але не відповідає запитам замовника.

Minor – незначний дефект, що не порушує працездатність ПЗ. Наприклад, помилка в дизайні програми.

Trivial – найменш незначний баг програмного забезпечення, наприклад, текстова друкарська помилка. Вона ніяк не впливає на працездатність, але не відповідає бажаному результату технічного завдання. [2]

Пріоритети відповідають за порядок виправлення багів на проєкті. Існує три найпоширеніші пріоритети: **High, Medium і Low** (іноді їх можуть класифікувати цифрами за тією ж логікою). Баг High повинен виправитися в першу чергу, зазвичай цей пріоритет поширюється на Blocker і Critical ступені серйозності, які можуть зашкодити програмному забезпеченню. Відповідно, після виправляються Medium і Low пріоритети, які критично не впливають на працездатність функцій софту [2].

Для правильної назви заголовка бага в атрибуті опису можна використовувати золоте правило **«що?», «де?» і «коли?»**. Що – що не працює коректно, Де – місце, в якому виявлено помилку, коли – за яких умов. Чим коректнішим і зрозумілішим буде опис бага, тим швидше він буде усунений. Оскільки, якщо баг буде не відтворюваний, то найімовірніше програміст відхилить цю помилку і вона не буде виправлена [3].

Усі баг-репорти вносяться до баг-трекінгових систем. По суті, баг-трекінгові системи створені не тільки для фіксації багів. В них також вносяться і User-стори, технічні вимоги від PM реалізації у продукті та інша документація. Найпоширенішими баг-трекінговими системами є Jira (найбільш використовувана), Asana, Redmine, Trello и Bugzilla.

Після початкових усунень багів, розроблений **продукт тестується повторно**, окремо на відтворення бага, і окремо за функціоналом, який міг бути порушений під час виправлення помилок. Потрібно розділяти повторне тестування бага і повторне

тестування суміжного компонента, щоб не дублювати фіксацію однієї помилки.

Перед написанням баг-репорту **обов'язково відтворіть помилку кілька разів**. Оскільки вона могла виникнути у зв'язку з нестабільною роботою мережі, або через неполадки ПК та інші причини, не пов'язані з працездатністю самого програмного забезпечення. Якщо виникнення бага повторюється, необхідно відразу зафіксувати помилку та повідомити про неї програміста. Також краще заздалегідь перевірити помилку з різних браузерів та з мобільної версії, щоб максимально точно локалізувати місця, де щось не працює. Дуже часто, можна спостерігати помилки, які відтворюються тільки в конкретному браузері, наприклад, є в Chrome, тоді як у Firefox все буде працювати коректно. І навпаки. Перед надсиланням баг-репорту програмістам необхідно перечитати його кілька разів, перевірити на орфографічні помилки, ще раз оцінити стисліть і ясність написаного звіту [3].

Баг-репорт є одним із найважливіших документів тестувальника, він показує вашу грамотність та кваліфікованість.

Заповнення баг-репортів.

Поля:

– у темі, описі та результатах повинна дотримуватися послідовність «Що? Де? Коли?». Відповідати на запитання необхідно саме в такій послідовності;

– тему, опис та результати необхідно описувати у теперішньому часі, відповідаючи на питання: «Що відбувається?»;

– допускається формулювання теми, опису та результатів в баг-репортах за принципом «Що?, Де?» без вказівки «Коли?», в разі, якщо баг відображається на сторінці завжди без будь-яких додаткових дій/подій/умов (наприклад, в багах верстки);

– поле «Summary» (короткий опис дефекту) потрібно описувати за принципом «Що? Де? Коли». Тема повинна повністю вміщатись в рядку «Summary»;

– поле «Description» потрібно описувати одним реченням за принципом «Що? Де? Коли?» згідно «Summary». Можна додати 1 додаткове речення, якщо це є важливою інформацією і не вміщується в «Summary» через обмеження за кількістю символів [4].

–

Кроки:

– у першому кроці необхідно вказати посилання на головний домен, а в наступних кроках описувати послідовність дій. Посилання необхідно вказувати тільки в першому кроці;

– у кроках необхідно відповідати на запитання: «Що необхідно зробити?», без звернень до третьої особи. Наприклад: Відкрити ..., Натиснути ..., Звернути увагу ...;

– у кроках необхідно коротко писати, що зробити, куди натискати, не уточнюючи назви сторінки, поп-ап вікна у кожному кроці;

– у кроках необхідно вказувати які дані вводяться в поле (валідні або невалідні). У разі невалідних повинні бути наведені приклади (наприклад: «Ввести спецсимволи в поле «Name» (*, @ ,, /, <,>)»);

– необхідно скоротити кількість кроків. Як правило, в баг-репорті описується не більше 7-8 кроків. Частина інформації можна винести у передумови;

– в останньому кроці не потрібно детально описувати помилку і дублювати фактичний результат. Потрібно акцентувати увагу на сам елемент з дефектом (Наприклад: Pay attention to the ..., Звернути увагу на ...);

– кілька кроків варто обєднати в один, якщо їх більше 8 (наприклад, «Заповнити обовязкові поля форми валідними даними»);

– не варто об’єднувати багато кроків в один (наприклад: Click the «Login» button and fill the «Facebook Account» form with valid data and then click the «OK» button). Необхідно розділити складовий крок на кілька окремих кроків [4].

Результати:

– очікуваний результат слід вказувати після фактичного результату

– результати також необхідно описувати згідно теми за принципом «Що? Де? Коли?», відповідаючи по порядку на ці питання;

– результати не потрібно нумерувати та не варто писати кілька результатів в одному дефекті;

– краще уникати формулювань очікуваного і фактичного результатів, які відрізняються лише негативною формою. Очікуваний

результат необхідно детально описати, недостатньо просто додати заперечення в одне з речень [4].

Скріншот/відео:

– розширення скріншота має бути: «.png», «.bmp», «.jpeg»;

– на скріншоті потрібно відобразити курсор (вказівник миші);

– проблемне місце потрібно виділити червоним прямокутником і вказати на нього червоною стрілкою (прямокутником потрібно виділяти місце з помилкою, а не весь блок). За потреби зеленим прямокутником і зеленою стрілкою можна вказати правильний приклад або макет;

– на скріншоті/відео не повинна відображатися панель закладок браузера або інша зайва інформація (задній фон робочого столу, вікно програвача відеофайлів, сторонні повідомлення та ін.);

– на скріншоті/відео необхідно показати вікно браузера з адресним рядком;

– відео необхідно завантажувати формату «.avi», «.mp4», «.mkv» [4].

Інше:

– при тестуванні сайтів на десктопних пристроях в поле «OS» пишеться назва операційної системи (наприклад, Windows), в поле «OS Version» – версія операційної системи (наприклад, 7x64);

– необхідно вказати браузер і його версію в полі «Platform». Якщо баг повторюється в різних браузерах, їх потрібно вказати в розділі «Additional information»;

– обов'язково слід вказати версію браузера, так як поява багатьох дефектів залежить саме від версії браузера;

– пропущені деяких атрибутів в баг-репорті (тема, опис, кроки, результати, скріншот/відео, тестове оточення) є неприпустимим;

– у темі та результатах потрібно вказати, яка саме помилка відображається без застосування слів «некоректно/коректно», «невірно/вірно», «правильно/неправильно», «incorrect/correct», «wrong/right». Наприклад: Зображення товару розтягнуте (Що?) у формі «Надіслати другу» (Де?);

– необхідно перефразувати тему або результати, вказавши яка саме помилка відображається. Не варто писати «нічого не відбувається», «не працює», «ламається верстка», «nothing happens», «the layout is broken». Наприклад: Вікно завантаження зображення не

з’являється на сторінці редагування профілю ... The menu items are shifted to the left;

– в баг-репорт не варто використовувати слова на кшталт «можна/не можна», «можливо/неможливо», «possible/impossible», «can/can not». Необхідно точно описувати суть бага, вказуючи, в чому саме невідповідність;

– особисті займенники («ви», «ми», «я»), а також слова «користувач/user» краще не використовувати, перефразувавши опис баг-репорта від третьої особи, акцентуючи увагу на об’єкт дії: «сайт», «кнопка», «посилання» та ін.;

– тему та результати слід описувати повними реченнями з підметом і присудком;

– поряд з назвою елемента потрібно писати його тип (кнопка, посилання, поле і т.д.), так як на сайті може бути декілька елементів з однаковими назвами;

– не потрібно перекладати назви елементів сайту на мову оформлення звіту. Назви кнопок, полів, посилань, повідомлень необхідно вказувати саме так, як вони відображені на сайті;

– коли баг відображається після скоєної дії правильно писати саме «після» замість «при». Наприклад, «після натискання…», «після наведення...»;

– не потрібно створювати дублікати з описом одного і того ж дефекту, який відтворюється на різних сторінках або для різних елементів. У розділі «Additional Information» вказується перелік сторінок та елементів з однотипними багами;

– тему, опис, кроки, назви результатів необхідно писати з великої літери;

– необхідно використовувати лапки в назвах елементів сайту [4].

Приклад наведено на рисунках 1.1 – 1.2.

Номер ID	Проєкт	Категорія	Рівень доступу	Дата реєстрації	Дата зміни
Тестувальник		**Пріорітет**		**Серйозність**	
Відтворюваність		**Статус**		**Резолюція**	
Платформа		**Операційна система**		**Версія ОС**	
Тема					
Опис					
Кроки для відстворення					
Фактичний результат					
Очікуваний результат					
Додаткова інформація					
Прикріплені файли					

Рисунок 1.1 – Шаблон баг-репорту

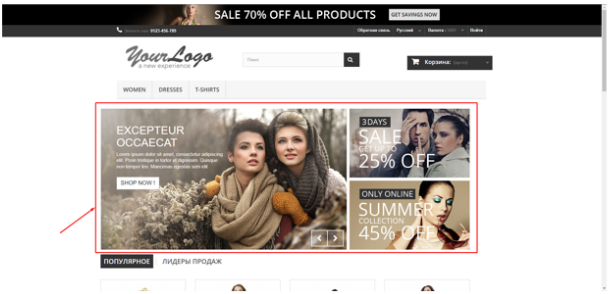
номер ID	Проєкт	Категорія	Рівень доступу	Дата реєстрації	Дата зміни
4	http://prestashop.qatestlab.com.ua/en/	верстка	загальний	20.11.2024	20.11.2024
Тестувальник	Круковський Євгеній	**Пріорітет**	низький	**Серйозність**	незначна
Відтворюваність	постійно	**Статус**	надіслано	**Резолюція**	відкритий
Платформа	Google Chrome ver.109.0.5414.168	**Операційна система**	Windows	**Версія ОС**	11 23H2
Тема	Банери на головній сторінці не вирівнянні по центру				
Опис	Банери на головній сторінці не вирівнянні по центру				
Кроки для відстворення	1.Відкрити головну сторінку сайту (http://prestashop.qatestlab.com.ua/en/) 2. Звернути увагу на банери				
Фактичний результат	Банери роздашовані ближче до лівої сторони				
Очікуваний результат	Банери розташовані по центру сторінки				
Додаткова інформація	Баг також відтворюється у браузері Opera v113.0.5230.62 та Edge ver. 127.0.2651.74				
Прикріплені файли					

Рисунок 1.2 – Приклад заповнення баг-репорту

1.2 Завдання до лабораторної роботи

1.2.1 Відкрити сайт, наприклад: https://www.automationexercise.com/.

1.2.2 Знайти 5 багів, та оформити за ними баг-репорти (рисунки 1.1 – 1.2).

1.2.3 Оформити звіт з лабораторної работи. Баг-репорти в форматі «.xlsx» або «.xls» додати до звіту. Звіт надати на перевірку

1.3 Зміст звіту

1.3.1 Тема та мета роботи.

1.3.2 Завдання до роботи.

1.3.3 Короткі теоретичні відомості.

1.3.4 Хід виконання завдання до лабораторної роботи.

1.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 1.4 за вибором студента).

1.4 Контрольні запитання

1.4.1 Визначити основні властивості та відмінності браузерів: Mozilla Firefox, Opera, Google Chrome, Edge. Обґрунтувати розповсюдженість використання зазначених браузерів.

1.4.2 Для чого призначені та які існують системи відслідковування помилок?

1.4.3 Що повинен містити якісний опис помилки?

1.4.4 Що відносять до атрибутів дефекту?

2 ЛАБОРАТОРНА РОБОТА №2
ВИДИ ТЕСТУВАННЯ

Мета лабораторної роботи: ознайомлення з видами тестування

2.1 Короткі теоретичні відомості

Види тестування програмного забезпечення.

Класифікація за ознаками

1 За ступенем автоматизації:

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

2 За ступенем підготовленості до тестування:

- тестування по документації (formal testing);
- тестування ad hoc або інтуїтивне тестування (ad hoc testing) — тестування без тест-плану та документації, що базується на методиці передбачення помилки на власному досвіді тестувальника.

3 За знанням системи:

- тестування чорної скриньки (black box);
- тестування білої скриньки (white box);
- тестування сірої скриньки (grey box).

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 або Димове тестування (smoke testing);
- тестування нової функціональності (new feature testing);
- регресивне тестування (regression testing);
- тестування при здачі (acceptance testing);
- бета-тестування (beta testing).

6 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 або Юзабіліті-тестування (usability testing);
– тестування інтерфейсу користувача (UI testing);
– тестування безпеки (security testing);
– тестування локалізації (localization testing);
– тестування сумісності (compatibility testing).

7 За ознакою позитивності сценаріїв:

– позитивне тестування (positive testing);
– негативне тестування (negative testing) [5]–[6].

2.2 Завдання до лабораторної роботи

2.2.1 Перелічити і описати не менше 10 видів тестування, про які ви знаєте. Кожен з видів тестування необхідно описувати своїми словами з поясненням. Важливо описати не тільки види тестування, а ще й вказати критерії класифікації, до яких відноситься той чи інший вид тестування (наприклад критерій по ступеню автоматизації: автоматизоване, напівавтоматизоване, ручне тестування). Розширення файлу з описом видів тестування має бути: «.doc», «.docx», або «.pdf». Файл необхідно додати до звіту

2.2.2 Створити схему з класифікацією видів тестування та застосувати кожен з видів до якогось предмету на вибір (наприклад, ліфт, кружка, стілець або будь-який інший предмет). Тести необхідно вигадувати самостійно, достатньо одного тесту для кожного виду тестування. Загалом має бути не менше 10 видів тестування та не менше 10 тестів для предмету.

Рекомендовані програми для оформлення схем: MS Word, Visio, XMind. Файл необхідно прикріпити до звіту у форматі «.png», «.jpeg», або «.jpg».

Приклади схем наведено на рисунках 2.1 та 2.2.

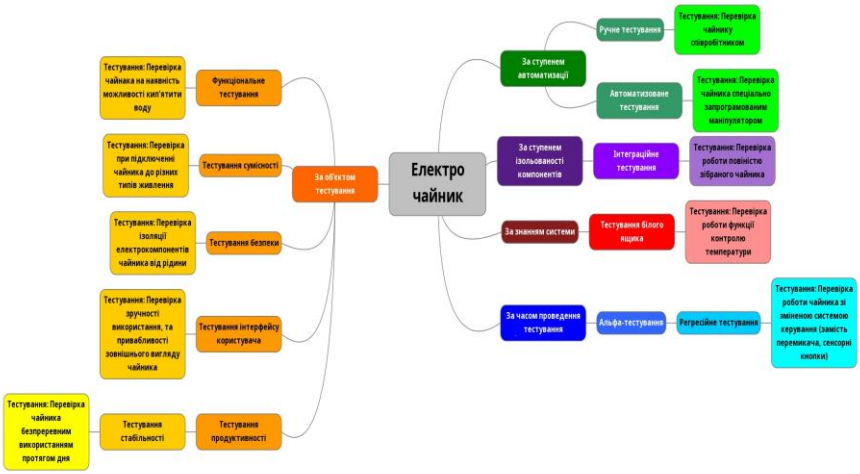


Рисунок 2.1 – Схема з класифікацією видів тестування електрочайника

Рисунок 2.2 – Схема з класифікацією видів тестування електромобіля

2.3 Зміст звіту

2.3.1 Тема та мета роботи.
2.3.2 Завдання до роботи.
2.3.3 Короткі теоретичні відомості.
2.3.4 Хід виконання завдання до лабораторної роботи.
2.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 2.4).

2.4 Контрольні запитання

2.4.1 Пояснити, коли слід використовувати той чи інший вид тестування, з переліку, який було описано в Завданні 2.2.1.

2.4.2 Для чого існує поділ тестування на види та підвиди?

2.4.3 Вказати переваги та недоліки використання тестових сценаріїв у тестуванні ПЗ на прикладі.

3 ЛАБОРАТОРНА РОБОТА №3 ВЕБТЕСТУВАННЯ, ЧЕКЛІСТИ ТА КРОСБРАУЗЕРНЕ ТЕСТУВАННЯ

Мета лабораторної роботи: отримання базових теоретичних та практичних навичок в розпізнаванні етапів тестування та створенні контрольного списку (checklist), отримання базових теоретичних та практичних навичок в кросбраузерному тестуванні.

3.1 Короткі теоретичні відомості

Чекліст – це зручний і структурований інструмент, який допомагає тестувальникам у проведенні перевірки програмного забезпечення. Він являє собою список завдань, кроків і критеріїв, які необхідно виконати для ретельної перевірки функціональності або інших аспектів ПЗ [7].

Використання чекліста в процесі тестування ПЗ надає кілька переваг:

– **нічого не пропустити!** Уявіть, що є величезний застосунок із безліччю функцій і можливостей. Без чекліста, можна забути перевірити якісь важливі аспекти та пропустити помилки. Чекліст допомагає не загубитись і бути впевненими, що покриті всі основні перевірки. Він нагадує про кожен крок, який потрібно виконати, щоб нічого не пропустити;

– **структурованість і організація.** У роботі тестувальника дуже важливо мати структуру й організацію. Чекліст це і надає: допомагає розбити процес на більш дрібні завдання і слідувати за порядком. Це особливо корисно, коли є великий обсяг роботи або обмежений час. Чекліст дає чітке уявлення про те, що потрібно робити і в якій послідовності;

– **уніфікація та консистентність.** У команді QA може працювати кілька тестувальників, і кожен із них може мати свої вподобання та методи роботи. Чекліст допомагає уніфікувати процес і встановити єдині стандарти для всіх. Коли кожен дотримується одного й того самого чекліста, можна бути впевненим, що всі основні

перевірки будуть виконані. Це підвищує консистентність і якість роботи команди [7]–[8].

З яких ключових моментів складається чекліст тестування? Їх кілька і вони допомагають організувати та впорядкувати процес перевірки:

– **завдання та функціональність.** Чекліст має містити список завдань і функціональності, які слід перевірити. Це може бути щось на кшталт “перевірити, що кнопка «Відправити» надсилає дані форми” або “переконатися, що функція пошуку знаходить усі результати, які повинна”. Простіше кажучи, це список дій і перевірок, які слід виконати, щоб переконатися, що все працює належним чином;

– **критерії оцінки** або очікувані результати, які слід порівняти з фактичними результатами перевірок. Наприклад, якщо перевіряється функція пошуку, критерієм оцінки може бути те, що за правильного запиту мають бути видані всі відповідні результати. Якщо це відбувається, завдання відзначається як виконане успішно. Якщо ні, значить, є недолік, якому потрібно приділити увагу;

– **пріоритети.** У чеклісті також можна вказати пріоритети для кожного завдання. Це допомагає визначити, які з них слід виконати першими, щоб зосередитися на найважливіших і критичних аспектах ПЗ. Пам’ятайте, може бути обмежений час або ресурси, тому вміння пріоритизувати – ключова навичка!

При проходженні чеклістів тестувальник зазначає статус навпроти кожного тестованого пункту. Можливі такі варіанти статусів:

– «Passed» – перевірка пройдена успішно, багів не знайдено;

– «Failed» – знайдений один або більше багів;

– «Blocked» – неможливо перевірити, тому що один з багів блокує поточну перевірку;

– «In Progress» – поточний пункт, над яким працює тестувальник;

– «Not run» – ще не перевірено;

– «Skipped» – не буде перевірятися з будь-якої причини [8].

Зразок чекліста наведено на рисунку 3.1.

Сайт http://prestashop.qatestlab.com.ua/en/	Браузер версія	Браузер версія	Браузер версія	Примітки/Посилання на баг-репорт
Реєстрація та Особистий профіль				
Перевірити реєстрацію на сайті				
Перевірити редагування профілю				
Перевірити зміну паролю в особистому кабінеті				
Перевірити додавання нової адреси				
Валідаційні повідомлення для обов'язкових полів:				
Порожні поля				
Пароль, який містить менше 6 символів				
Заповнення полів валідними даними				
Авторизація				
Повторний вхід після реєстрації				
Валідаційні повідомлення:				
Невалідна пошта (незареєстрований користувач)				
Невалідний пароль				
Зворотний зв'язок				
Перевірити валідацію полів				
Перевірити відправку листа / повідомлення				
Пошук				
Перевірити роботу пошуку за назвами товарів				
Перевірити перехід за підказками у випадаючому списку поля пошуку після введення ключового				
Перевірити роботу пошуку по різноманітним параметрам				
Перевірити сортування товарів за назвою				
Перевірити фільтрування за категоріями				
Коментарі				
Перевірити додавання коментаря-відгуку до товару				
Валідаційні повідомлення:				
Порожні поля				
Шаринг + фото				
Перевірити функцію шаринга фото у всіх				
Перевірити функцію перегортання фото				
Перевірити функцію збільшення фото (якщо є)				
Сторінка товару та Кошик				
Додати товар в кошик				
Видалити товар з кошика				
Правильне вирахування кількості товару в кошику				
Правильне вирахування загальної суми товару в				
Перевірити чи виводиться візуальне підтвердження				
Можливість повернутися до кошика та замовлення				
Можливість повернутися до кошика та замовлення				
Оновити кошик покупок				
Підписка на новини				
Перевірити можливість підписатися на розсилку				

Рисунок 3.1 – Чекліст

Вимоги до розмітки (верстки):

– чи немає помітних оку невідповідностей: поламані блоки, не стикування кольору, некоректне відображення тексту навколо зображень;

– якщо дизайн згідно ТЗ розрахований на певну ширину, то, незалежно від браузера, не повинна з'являтися горизонтальна прокрутка;

– під час зменшення розміру вікна менше мінімального згідно ТЗ не повинно нічого ламатися, фони не повинні «розпливатися»;

– сайт повинен виглядати однаково у всіх стандартних розширеннях екрану (1024x600, 1024x768, 1152x864, 1280x800,

1280x1024, 1440x900, 1680x1050, 1920x1080): не повинно нічого ламатися, не повинні різко обриватися фони при великих розширеннях;

– при відключених зображеннях, написи (особливо логотип та головне меню сайту) повинні залишатися читабельними, на всіх інформаційних картинках повинні бути підписи акуратним, невеликим, сірим шрифтом (відключення картинок: Chrome → Settings → Search settings → JavaScript → Site settings→ Images → вибрати опцію «Don`t allow sites to show images»; Firefox→ about:config → permissions.default.image → 2);

– працездатність при вимкненому JavaScript. Увесь критично важливий функціонал сайту повинен бути доступний без JS (відключення JavaScript: Chrome → Settings → Search settings → JavaScript → JavaScript (Content) → Don`t allow sites to use JavaScript; Firefox→ about:config → Accept the Risk and Continue → javascript.enabled → Toggle the «javascript.enabled» preference) [9]–[10].

Перевірка однотипності/симетричності в інтерфейсі:

– однотипність/симетричність використання відступів;

– ширина колонок та контентного поля;

– позиціонування елементів верхнього та нижнього колонтитулів на всіх сторінках;

– позиціонування банерів, правого та лівого блоку;

– розмір текстів;

– відступи між абзацами;

– відстані між рядками;

– елементи управління (кнопки, чекбокси, випадаючі списки);

– розмір/колір/тип шрифта;

– наявність стилів оформлення заголовків першого, другого та більше рівнів;

– коректне відображення тексту навколо зображень;

– напрямки тіней у всіх елементів управління;

– назви однакових елементів у різних місцях;

– чи мають елементи, що призначені для натискання, вказівник «pointer»; елементи, які можна перетягувати – вказівник «move» або «crosshair»; неактивні/недоступні елементи – вказівник «default» (рисунок 3.2) [10].

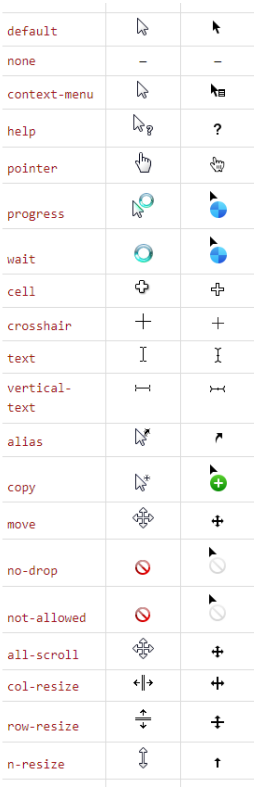


Рисунок 3.2 – Форма вказівника при наведенні на відповідний елемент

Кросбраузерність – властивість сайту відображатися та працювати у всіх браузерах ідентично. Під ідентичністю розуміється відсутність розвалів верстки та здатність відображати матеріал з однаковим ступенем читабельності. Поняття «кросбраузерність» дуже часто плутають з попіксельною відповідністю, що насправді є різними поняттями. Так як вебтехнології весь час розвиваються, прийнятну кросбраузерність можна забезпечити тільки для останніх версій різних браузерів. На практиці зазвичай обмежуються тільки найпопулярнішими браузерами, що суттєво може скоротити час на розробку сайту [11].

Правила кросбраузерності:

– під час застосування будь-яких змін в дизайні сайту (чи то новий інформер, рекламний блок, новий шаблон або ділянка шаблонів) – необхідно звертати увагу, який має вигляд дизайн в інших браузерах;

– потрібно тестувати сайти самостійно за допомогою встановлених на комп'ютері браузерів, інколи використовуючи спеціальні онлайн сервіси;

– необхідно здійснювати тестування кросбраузерності на всіх доступних пристроях (комп'ютер, ноутбук, планшет та ін.);

– завжди повинен бути встановлений на комп'ютері комплект найбільш популярних браузерів, які не вичерпують ресурси жорсткого диска, але якщо буде потрібно перевірити кросбраузерність сайту – вебмастер запускає їх та проводить необхідний аналіз;

– різні версії одного й того ж браузера будуть відображати сайт теж по-різному і це потрібно враховувати, але намагатися використовувати найостанніші версії.

Основні проблеми кросбраузерності сайту лежать на витоках його створення – під час створення шаблону. Існує багато готових шаблонів, з яких можна вибрати той єдиний, який буде однаково (або з незначними змінами) відображатися у всіх браузерах, а потім доопрацювати його під свій проєкт, але з постійним орієнтиром на різні браузери [11].

Інструменти тестування кросбраузерності верстки:

1 Browsershots, ймовірно, володіє найширшим набором браузерів серед безкоштовних інструментів тестування, включає в себе браузери, що працюють в Linux, Windows та BSD. Серед них є такі, про які рідко хто чув (наприклад, Galeon, Iceape, Kazehakase або Epiphany). Browsershots дозволяє тестувати як в останніх версіях кожного браузера, так і в застарілих версіях. Хоча Browsershots дозволяє тестувати у величезній кількості браузерів, варто пам'ятати, чим більший набір браузерів для тестування, тим довше доведеться чекати результату. Так що варто зупинитися на основних браузерах.

2 Browser Sandbox – інструмент, який буде корисним тільки користувачам Windows. Він підтримує такі браузери, як Firefox, Chrome, ChromiumCanary, Firefox Mobile, Safari, Opera та FirefoxNightly. В безкоштовному варіанті існує можливість тестування

тільки в останній версії певного браузера. Для тестування в старих версіях необхідно користуватися платною версією.

3 CrossBrowserTesting – сервіс, який використовує реальні пристрої для тестування сайту. Сервіс платний, але надає можливість тестувати більш як в 900 браузерах і приблизно в 40 операційних системах. Ще одна його особливість полягає в режимі live testing, в якому можна тестувати сайт в реальному оточенні, перевіряючи дієздатність AJAX, HTML-форм, JavaScript тощо.

4 Sauce Labs (безкоштовна та комерційна версії) – надає доступ до безлічі браузерів в різних ОС та встановлює з'єднання браузера з налагодженою віртуальною машиною. Також записується відео всієї сесії тестування. Sauce надає 200 хвилин безкоштовного тестування на місяць і дозволяє створювати тести автоматизованого тестування в браузерах (використовується Selenium).

5 Browsera (безкоштовна та комерційна версії) – забезпечує автоматизацію тестування сумісності. Він автоматично визначає відмінності в відображенні сторінок браузерами, тим самим спрощуючи процес тестування. Також визначаються помилки JavaScript, а в комерційній версії дозволяє тестувати сторінки за передплатою та сторінки, що вимагають авторизації. Також можна протестувати динамічні сторінки. Безкоштовна версія включає в себе достатньо обмежене число браузерів та низький дозвіл. Існують різні комерційні версії, які підтримують більшу кількість браузерів, забезпечують високий дозвіл та дозволяють тестувати «закриті» сторінки.

6 BrowserStack – ще один з найвідоміших сервісів для кросбраузерного тестування. Він також надає реальні пристрої для тестування і підтримує більш як 700 браузерів. Існує можливість локального тестування та швидкого отримання скріншотів на різних розширеннях екранів від 800-600 до 2048-1536. Сервіс платний, але для деяких open source проєктів пропонуються безкоштовні послуги [12].

3.2 Завдання до роботи

3.2.1 Завантажити зі сторінки дисципліни в навчальній системі Moodle шаблон чекліста (рисунок 3.1). Вказати прізвище та ім'я власника у його назві.

Додати мінімум 5-7 пунктів до існуючих пунктів контрольного списку на вкладці «Верстка», не повторюючись із вже доданими, та відмітити їх будь-яким кольором.

3.2.2 Провести тестування верстки сайту, наприклад https://www.automationexercise.com/, перевіривши всі пункти контрольного списку на вкладці «Верстка» у 3 браузерах (Firefox, Google Chrome, Edge, Opera тощо) в останній або передостанній версії.

3.2.3 Створити баг-репорти до 5 – 7 знайдених багів під час тестування верстки сайту із пункту 3.2.2.

3.2.4 У чеклісті вказати назву та версію браузерів, назву сайту. Результат перевірки відзначити «Passed/Failed». Для «Failed» вказати в примітці або нотатках посилання на баг-репорти (5-7 шт.), інші додати тему багу за принципом «Що? Де? Коли?».

3.2.5 Доповнений та пройдений чекліст в форматі «.xlsx» або «.xls» та баг-репорти додати до звіту. Звіт надати на перевірку

3.3 Зміст звіту

3.3.1 Тема та мета роботи.

3.3.2 Завдання до роботи.

3.3.3 Короткі теоретичні відомості.

3.3.4 Хід виконання завдання до лабораторної роботи.

3.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 3.4 за вибором студента).

3.4 Контрольні запитання

3.4.1 Що відносять до дефектів розмітки (верстки)? Навести приклади та обґрунтувати відповідь.

3.4.2 За яким принципом були додані нові пункти до чекліста? Обґрунтувати важливість нових пунктів при тестуванні сайту із завдання.

3.4.3 Для чого в примітці до пункту чекліста вказується посилання на звіти у системі відслідковування помилок? Обґрунтувати необхідність посилання на звіт в чеклісті.

3.4.4 Обґрунтувати необхідності створення пунктів та підпунктів в чеклісті.

3.4.5 Які існують переваги використання чеклістів порівняно з іншою подібною документацією для проведення тестування? Навести приклади та обґрунтувати відповідь.

3.4.6 Обґрунтувати необхідність та важливість кросбраузерного тестування.

3.4.7 За якими чинниками визначають, в яких браузерах проводити тестування?

3.4.8 Перерахувати існуючі розширення у різних браузерах для тестування вебінтерфейсів, вказати доцільність та ефективність застосування.

3.4.9 Для чого здійснюється та як впливає на роботу очистка історії (включаючи cookies, cache та ін.) у браузері? Навести приклад роботи браузера до очистки та після.

3.4.10 Назвати програму, яка була використана для тестування кросбраузерного тестування у лабораторній роботі, обґрунтувати вибір даної програми.

4 ЛАБОРАТОРНА РОБОТА №4
ТЕСТУВАННЯ ЗРУЧНОСТІ ВИКОРИСТАННЯ

Мета роботи: отримання базових теоретичних та практичних навичок у визначенні та застосуванні тестування зручності використання та складанні простого контрольного списку.

4.1 Короткі теоретичні відомості

Тестування зручності використання (usability testing) – метод тестування, спрямований на встановлення ступеня зручності використання, здатності до навчання користувача, зрозумілості та привабливості для користувачів розроблювального продукту в контексті заданих умов [13].

Тестування зручності користування дає оцінку рівня зручності програми за наступними пунктами:

- **продуктивність, ефективність (efficiency)** – скільки часу та кроків знадобиться користувачеві для завершення основних завдань програми, наприклад, розміщення новини, реєстрації, покупка та ін.? (менше – краще);
- **правильність** (**accuracy**) – скільки помилок зробив користувач під час роботи з додатком? (менше – краще);
- **активізація в пам'яті** (**recall**) – як багато користувач пам'ятає про роботу програми після припинення роботи з нею на тривалий період часу? (повторне виконання операцій після перерви має проходити швидше ніж у нового користувача);
- **емоційна реакція** (**emotional response**) – як користувач себе почуває після завершення завдання – розгублений, в стані стресу? Чи порекомендує користувач систему своїм друзям? (позитивна реакція – краще);
- простоту використання сайту або інтерфейсу;
- ефективність використання;
- запам'ятовуваність;
- помилки, їх кількість та серйозність;
- задоволення користувача (суб'єктивне) [14].

Правильне уявлення про тестування зручності використання та застосування в роботі приходить з досвідом та знаннями структури проєкту, а також вимог до проєкту. Також слід звернути увагу на те, що саме підлягає тестуванню:

- вебдодаток;
- мобільний додаток;
- десктоп-додаток.

У кожного з цих видів додатків можуть бути різні цілі, використовувані ресурси (наприклад пам'ять, енергія та ін.), оточення, ступінь досвіду користувача та багато іншого [13]–[14].

Розглянемо приклади.

1 Вебдодаток (інтернет-магазин).

Мета – користувачі повинні багаторазово користуватися сайтом, весь цикл повинен бути простий, містити якомога менше кроків, інтерфейс повинен бути інтуїтивно зрозумілий для будь-якого користувача.

Критично важливо перевірити простоту використання (основного функціоналу) та зрозумілість використання інтерфейсу без додаткового вивчення яких-небудь документів або правил використання.

2 Спеціалізований вебдодаток для формування бухгалтерських звітів.

Мета – користувач повинен мати можливість скласти звіти з усіма можливими даними, що зберігаються в БД в різній послідовності за певний період часу.

Не так важлива простота використання **як** функціональні можливості та правильність даних, тобто скоріше всього в складній системі працюватимуть користувачі, які заздалегідь навчені всім можливим алгоритмам роботи та знають де й що натискати.

Контрольний список – один з фундаментальних інструментів тестування: це документ, який описує, що має бути протестовано та дозволяє не забувати про важливі тести, фіксувати результати роботи та відстежувати статистику про статус програмного продукту [8].

Навіщо потрібен контрольний список (checklist)?

- не забути необхідні тести;
- для поділу завдань за рівнем кваліфікації;
- для збереження звітності та результатів тестування.

Що має бути в контрольному списку (checklist)?

- список перевірок *(з необхідним ступенем деталізації);*
- оточення перевірки: тестове оточення *(якщо є);*
- інформація про тестувальників;
- збірка, на якій проводилося тестування;
- результат перевірки [8].

4.2 Завдання до роботи

4.2.1 Завантажити приклад чекліста (рисунок 3.1).

4.2.2 Вказати прізвище та ім’я у назві контрольного списку.

4.2.3 Провести «Usability» тестування сайту, наприклад https://www.automationexercise.com/, перевіривши всі пункти контрольного списку на вкладці «Usability» у 3 браузерах (Firefox, Google Chrome, Edge, Opera або інший) в останній або передостанній версії. Додати мінімум 5-7 пунктів до існуючих пунктів контрольного списку на вкладці «Usability», не повторюючись з вже доданими, та відмітити їх будь-яким кольором.

4.2.4 Створити баг-репорти до 5-7 знайдених багів під час тестування зручності використання сайту із пункту 4.2.3, оформити та додати до звіту.

4.2.5 У чеклісті вказати назву та версію браузеру, назву сайту. Результат перевірки відзначити «Passed/Failed». Для «Failed» вказати в примітці або нотатках посилання на баг-репорт (5-7 шт.), інші додати тему багу за принципом «Що? Де? Коли?».

4.2.6 Доповнений та пройдений чекліст в форматі «.xlsx» або «.xls» та баг-репорти додати до звіту.

4.3 Зміст звіту

4.3.1 Тема та мета роботи.

4.3.2 Завдання до роботи.

4.3.3 Короткі теоретичні відомості.

4.3.4 Хід виконання завдання до лабораторної роботи.

4.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 4.4 за вибором студента).

4.4 Контрольні запитання

4.4.1 Що відносять до дефектів зручності використання? Навести приклади та обґрунтувати відповідь.

4.4.2 За яким принципом були додані нові пункти до чекліста? Обґрунтувати важливість нових пунктів при тестуванні сайту із завдання.

4.4.3 Якого принципу необхідно дотримуватися при поліпшенні зручності користування?

4.4.4 Які основні пункти та стовпці повинен містити контрольний список з тестування зручності використання (usability testing)?

4.4.5 Обґрунтувати важливість тестування зручності використання.

5 ЛАБОРАТОРНА РОБОТА №5
ФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ

Мета роботи: ознайомлення з функціональним тестуванням та отримання базових теоретичних та практичних навичок в пошуку функціональних багів під час тестування сайту.

5.1 Короткі теоретичні відомості

Функціональне тестування (functional testing) – вид тестування, при якому виявляється некоректна/неправильна робота функціоналу програми або процес перевірки відповідності поведінки системи першочергово заявленим функціональним вимогам [15].

Також, можна сказати, що **функціональне тестування –** це тестування програмного забезпечення з метою перевірки реалізованості функціональних вимог, тобто здатності програмного забезпечення в певних умовах вирішувати завдання, потрібні користувачам. Функціональні вимоги визначають, що саме робить програмне забезпечення, які завдання вирішує.

При функціональному тестуванні перевіряється коректність роботи системи, яка включає перевірку кожної з функцій програми, адекватність збережених і вихідних даних, методи їх обробки, обробка даних, що вводяться, методи зберігання даних, методи імпорту та експорту даних і т.д. залежно від специфіки додатку. При перевірці проєктів наводиться документація функціональних вимог, що спрощує перевірку [15].

При функціональному тестуванні проводиться перевірка наступних модулів сайту.

1 Реєстрація (registration, logging in).
2 Авторизація (authorization).
3 Новинний модуль (news).
4 Пошук на сайті (search).
5 Зворотній зв’язок (feedback).
6 Банери (banners).
7 Фотоальбоми – галереї (photo album).
8 Форум (forum).
9 Інтернет-магазин (online shop, e-shop).

10 Список, що випадає (drop-down list).

11 Коментарі, поширення в соціальних мережах (sharing).

12 Відео (video).

13 Модуль розсилки (mailout module).

14 Форми (forms) [16].

Під час роботи з даними важливу роль відіграє валідація даних (data validation). Перш ніж використовувати отримані від користувача дані, необхідно переконатися, що вони введені правильно і показують коректні значення [17].

Валідація (validation) – це процес перевірки даних на відповідність певним, заздалегідь відомим правилам (форматам, вимогам) [17].

Під час тестування необхідно перевіряти узгодженість валідаторів вхідних даних з логікою обробки цих даних додатком. Для тестування затвердження даних, треба розуміти, як вони повинні працювати, що можна вважати правильним, а що ні.

«Невалідні» дані, що не задовольняють певним обмеженням, можуть викликати збій у роботі програми.

Валідація даних здійснюється наступним чином.

1 Посимвольна перевірка.

2 Перевірка окремих значень.

3 Сукупність вхідних значень.

4 Перевірка стану системи після обробки даних [17].

5.2 Завдання до роботи

5.2.1 Завантажити приклад чекліста (рисунок 3.1).

5.2.2 Вказати прізвище та ім’я власника у назві контрольного списку.

5.2.3 Додати мінімум 5-7 пунктів до існуючих пунктів контрольного списку на вкладці «Функціонал», не повторюючись з вже доданими, та відмітити їх будь-яким кольором.

5.2.4 Провести функціональне тестування сайту, наприклад https://www.automationexercise.com/, перевіривши всі пункти контрольного списку на вкладці «Функціонал» мінімум в 3х браузерах (Firefox, Google Chrome, Edge, Opera або інший) в останніх або передостанніх версіях.

5.2.5 Створити баг-репорти до 5-7 знайдених багів під час функціонального тестування сайту із пункту 5.2.4, оформити та додати до звіту.

5.2.6 Результат перевірки за чеклістом відзначити «Passed/Failed». Для «Failed» вказати в примітці або нотатках посилання на баг-репорти (5-7 шт.), інші додати тему багу за принципом «Що? Де? Коли?».

5.2.7 Доповнений та пройдений чекліст в форматі «.xlsx» або «.xls» та баг-репорти додати до звіту.

5.3 Зміст звіту

5.3.1 Тема та мета роботи.

5.3.2 Завдання до роботи.

5.3.3 Короткі теоретичні відомості.

5.3.4 Хід виконання завдання до лабораторної роботи.

5.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 5.4 за вибором студента).

5.4 Контрольні запитання

5.4.1 Що відносять до дефектів функціонального тестування? Навести приклади та обґрунтувати відповідь.

5.4.2 За яким принципом були додані нові пункти до чекліста? Обґрунтувати важливість нових пунктів при тестуванні сайту із завдання.

5.4.3 Які основні пункти повинен містити контрольний список (checklist) з тестування функціоналу сайту?

5.4.4 Надати пояснення твердженню, що функціональне тестування є трудомістким процесом та може займати до 80% всього бюджету проєкту з тестування.

5.4.5 Тестування яких модулів сайту із завдання здійснювалось в першу чергу при функціональному тестуванні, навести приклади та обґрунтувати відповідь.

6 ЛАБОРАТОРНА РОБОТА №6
ТЕСТУВАННЯ ВЕБПРОЄКТІВ

Мета роботи: ознайомлення з основними етапами тестування вебдодатків та функціональними можливостями Інструментів розробника, проведення навантажувального тестування сайту.

6.1 Короткі теоретичні відомості

Будь-який вебдодаток складається з клієнтської та серверної частини, де клієнтом виступає браузер, для відображення даних при взаємодії з сервером. Перевага використання вебдодатків в тому, що всі стандартні функції в браузері виконуються незалежно від операційної системи. Проблеми виникають від різної реалізації HTML, CSS, DOM та інших специфікацій в продукті, що призводить до проблем при розробці та підтримці вебдодатків [9].

Етапи тестування вебдодатків.

1 Підготовчі роботи (вивчення отриманої документації, аналіз функціоналу, технічного завдання, кінцевих макетів сайту, складання тест-плану для подальшого тестування).

2 Тестування верстки вебсайтів: перевірка розташування елементів, відповідність їх позицій наданим макетам, оптимізація зображень і графіки, перевірка валідності коду та ін.

3 Функціональне тестування – найбільш тривалий етап перевірки ресурсу. Суть цього процесу полягає у перевірці всіх обов’язкових функцій сайту:

- інтеграція даних (Data Integration);
- перевірка полів даних (Data field checks);
- перевірка числових полів (Numeric field checks);
- перевірка буквено-цифрових полів (alphanumeric field checks);
- перевірка працездатності форм на сайті (наприклад, зворотній зв’язок, додавання коментаря в блог та ін.);
- перевірка роботи пошуку (включаючи релевантність результатів);
- перевірка гіперпосилань, пошук неробочих посилань;

– перевірка підгрузки файлів на сервер;

– перегляд на відповідність вмісту сторінок сайту вихідного контенту, наданого замовником;

– тестування взаємодії вебдодатків з сервером.

4 Usability тестування: проводиться для оцінки зручності продукту у використанні.

5 Тестування безпеки: перевірка, чи немає у користувачів доступу до службових/закритих сторінок; перевірка захисту всіх критично важливих сторінок (наприклад, розділу адміністрування сайту) від зовнішнього впливу.

6 Тестування продуктивності сайту: проводиться з метою визначення швидкодії сайту або його частини під певним навантаженням. Включає в себе такі види тестування:

– тестування навантаження;

– тестування швидкодії.

7 Обробка результатів. Звіт про виявлені в процесі тестування помилки передається учасникам проєкту, після чого керівник проєкту визначає відповідального за виправлення кожної з помилок. Далі визначається графік виправлення помилок, після чого проводиться повторне тестування з метою контролю якості виправлення помилок, а також відсутності нових. Дана процедура повторюється, поки сайт не буде відповідати специфікаціям технічного завдання [16].

Інструменти розробника (Dev Tools) – це потужний інструмент для налагодження коду вебсайтів, який за замовчуванням встановлений в браузерах Firefox, Google Chrome та в інших браузерах [18].

Крім веброзробки, даний інструментарій, разом із накопиченими знаннями з HTML та CSS, будуть дуже корисні тестувальникам при тестуванні вебсайтів і створенні баг-репортів. Грамотне оформлення баг-репортів, використання правильної термінології (назви елементів, стильових властивостей, помилок, які відображаються тощо), а також знаходження причини дефекту, допоможе розробникам швидко виправити дефект.

Основні функції Інструментів розробника:

– перегляд HTML, зміна стилів і верстки в режимі реального часу;

– перегляд CSS-метрик;

– аналіз використання та продуктивності мережі з великою точністю;

– налагодження та профілювання JavaScript;

– зручна консоль, яка надає детальну і корисну інформацію про помилки в JavaScript, CSS та XML;

– набір функцій логування сценаріїв JavaScript;

– інструменти для перегляду DOM сайту і багато іншого [19].

Для того, щоб відкрити Dev Tools можна:

– використати гарячі клавіші: Ctrl + Shift + I (Windows/Linux) або Command + Option + I (Mac);

– вибрати в браузері Google Chrome пункт меню «Додаткові інструменти => Інструменти розробника», в браузері Firefox – пункт меню «Веброзробка => Перемкнути інструменти»;

– натиснути клавішу F12 [18].

Тестування навантаження (Load Testing) – дозволяє зрозуміти чи здатен сайт/додаток стабільно працювати з навантаженням в допустимих межах і трохи перевищуючи ці межі. При цьому тестуванні перевіряється поведінка з високим навантаженням. Навантаженням може бути, наприклад, певна кількість одночасно працюючих користувачів додатка в певний проміжок часу. Такий тип тестування дозволяє визначити час відгуку важливих бізнес-транзакцій. Спостерігаючи за базою даних, сервером додатка і мережею можна визначити слабкі місця програми.

Навантажувальне тестування найчастіше застосовується для дослідження багатокористувацьких ресурсів і різноманітних загальних систем, але інші види програмних продуктів також можуть бути протестовані цим методом. Наприклад, можна перевірити, чи зможе графічний редактор впоратися з відкриттям зображення великого обсягу. Також можна з’ясувати, чи здатна система сформувати фінансовий звіт, проаналізувавши дані декількох місяців, або навіть років. Якісно створений навантажувальний тест забезпечить найбільш достовірними результатами.

Основною ідеєю навантажувального тестування є створення певного навантаження за допомогою однакових апаратних і програмних забезпечень, щоб відстежити індекс продуктивності продукту. Найефективнішим цей метод буде на ранніх етапах

розробки, тому що такий підхід дозволить отримати оптимальні результати вимірювання показників продуктивності системи [16].

Основні принципи навантажувального тестування.

1 Унікальність запитів. Під час складання сценарію слід враховувати реальні статистичні дані, вимоги, очікувану поведінку системи.

2 Час відгуку системи. Керуючи певним собі числом вимірів, можна визначити, до якого інтервалу часу потрапить той чи інший запит.

3 Розподіл системи залежить від часу відгуку. Кількість вузлів впливає на розкид часу відгуку системи, кожен з яких збільшує величину затримки при скануванні запитів. Цей факт варто врахувати під час складання вимог до продуктивності продукту.

4 Коректність відтворення навантажувальних профілів. Складність програмного забезпечення вимагає значних витрат часу і ресурсів на проєктування, програмування і подальшу підтримку. Розробка тестів і покриття функціоналу системи повинні бути збалансованими для отримання найбільш [16].

6.2 Завдання до роботи

6.2.1 За допомогою Dev Tools змінити властивості шрифту на сторінці будь-якого сайту (колір, розмір тощо), ширину поля. У відповідь додати до звіту скріншот з посиланням на цю сторінку (рисунки 6.1, 6.2).

6.2.2 Дослідити 3 зображення на будь-яких вебсайтах, визначити розмір цих зображень за допомогою Інструментів розробника. У відповідь додати до звіту скріншот з посиланнями на сторінки цих сайтів, вказати розміри зображень (рисунок 6.3).

6.2.3 Додати до звіту 2 скріншота з помилкою в консолі Dev Tools різних сайтів, також додати посилання на данні сторінки сайтів (рисунок 6.4).

6.2.4 Встановити програму JMeter для навантажувального тестування. Встановити Java 8+ (без встановленої Java на ПК програма JMeter працювати не буде).

6.2.5 У програмі JMeter в Тест-плані створити потік користувачів «Thread group».

6.2.6 Для потоку задати параметри: кількість користувачів на вибір від 50 до 150, період часу в секундах (від 2 до 5 хв), протягом якого буде виконуватись тест навантаження та кількість запитів від 1 до 25 на вибір (рисунок 6.5).

6.2.7 Зберегти налаштування на ПК файлом формату jmx.

6.2.8 Створити HTTP-запит для «Thread group» натиснувши праву кнопку миші – «Add» – «Sampler» – «HTTP-request».

6.2.9 В поле «Protocol [http]» ввести протокол згідно вибраного сайту, в поле «Server name or IP» ввести назву сайту без протоколу, в полі «HTTP-request» вибрати метод «GET-запиту» та зберегти налаштування.

6.2.10 Створити звіт для HTTP-запиту у вигляді таблиці та графічний – натиснувши праву кнопку миші – «Add» – «Listener» – «Summary Report» та «Add» – «Listener» – «Graph results». До звіту додати скріншоти (рисунки 6.6, 6.7).

6.2.11 Запустити виконання тест-плану.

6.2.12 Зробити звіт. Всі скріншоти додати до звіту.

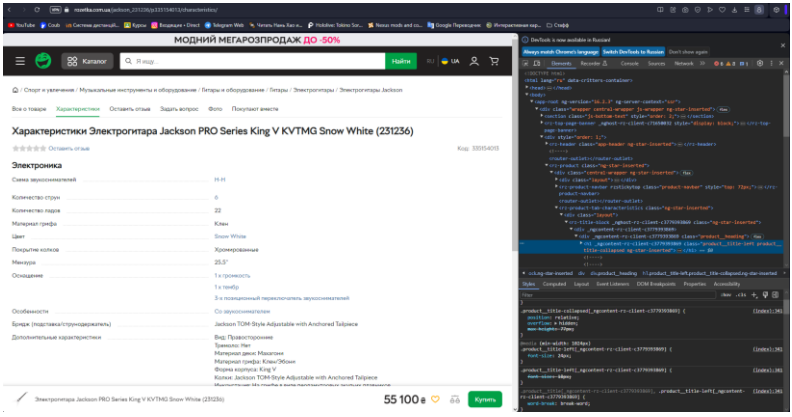


Рисунок 6.1 – Зображення до зміни параметрів шрифта: https://jmeter.apache.org/download_jmeter.cgi

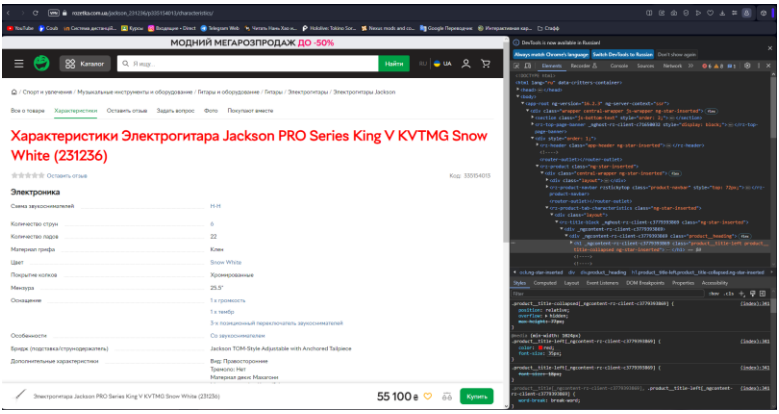


Рисунок 6.2 – Зображення після зміни параметрів шрифта: https://jmeter.apache.org/download_jmeter.cgi

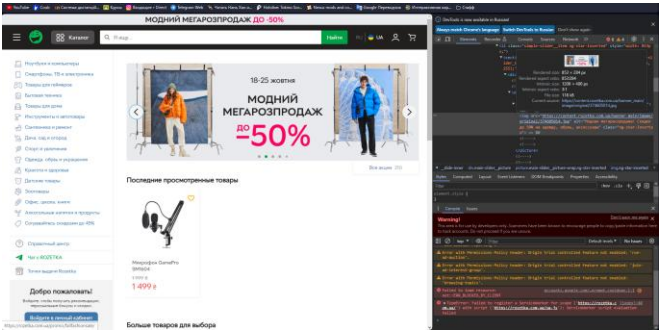


Рисунок 6.3 – Розмір картинки: 850 x 402 px. Посилання: https://prom.ua

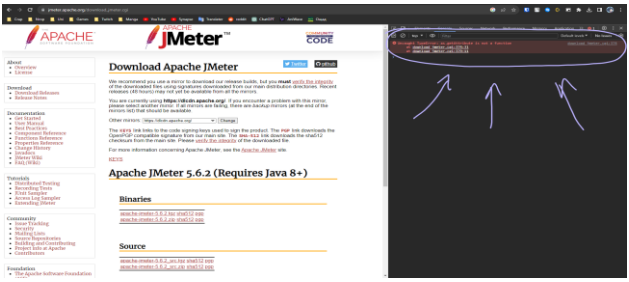


Рисунок 6.4 – Помилки на сайті https://jmeter.apache.org/download_jmeter.cgi

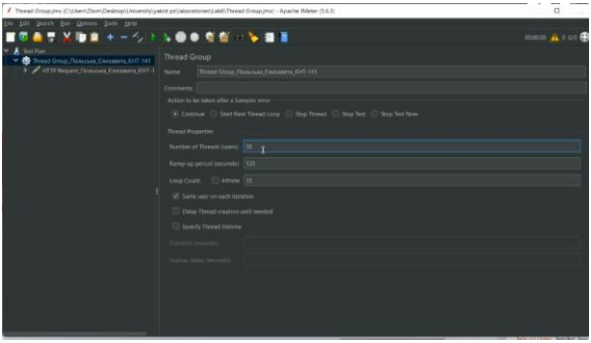


Рисунок 6.5 – Параметри проєкту

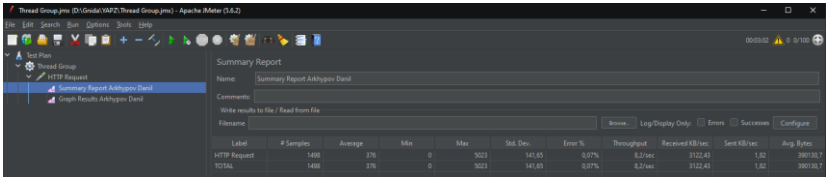


Рисунок 6.6 – Summary report

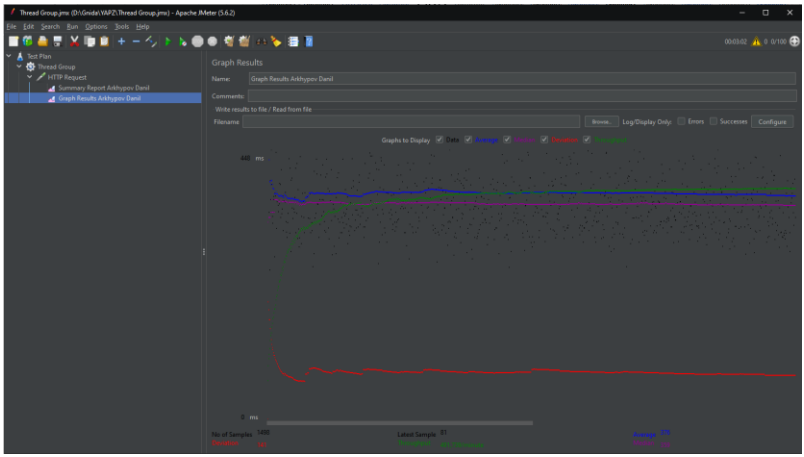


Рисунок 6.7 –Graph results

6.3 Зміст звіту

6.3.1 Тема та мета роботи.

6.3.2 Завдання до роботи.

6.3.3 Короткі теоретичні відомості.

6.3.4 Хід виконання завдання до лабораторної роботи.

6.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 6.4 за вибором студента).

6.4 Контрольні запитання

6.4.1 Яким чином можна оцінити проблемні частини сайту за допомогою Dev Tools? Відповідь обґрунтувати та навести приклади.

6.4.2 За допомогою якого програмного інтерфейсу можливо змінити «JavaScript» сценарії у реальному часі на сайті?

6.4.3 Як визначити за допомогою Dev Tools, коли почалося та закінчилося завантаження файлів на сайті? Відповідь обґрунтувати та навести приклади.

6.4.4 В якій послідовності завантажується «JavaScript»? Відповідь обґрунтувати та навести приклади.

6.4.5 Перерахувати, які помилки на сайті можна знаходити за допомогою інструментів розробника, навести приклади.

6.4.6 Навести приклади інструментів для навантажувального тестування, коротко розповісти про них.

6.4.7 Розповісти для чого проводиться навантажувальне тестування?

7 ЛАБОРАТОРНА РОБОТА №7
ТЕСТ-ДИЗАЙН ТА ТЕСТ-КЕЙСИ

Мета роботи: ознайомлення з техніками тест-дизайну, отримання базових теоретичних та практичних навичок в групуванні тест-кейсів, здобуття навичок створення тестових випадків.

7.1 Короткі теоретичні відомості

Тест-дизайн (test design) – це етап процесу тестування програмного забезпечення (ПЗ), на якому проєктуються і створюються тестові випадки (тест-кейси), відповідно з певними раніше визначеними критеріями якості і цілями тестування [20].

Тестовий випадок або тест-кейс (test case) – це сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації функції, що тестується, або її частини.

Тестовий набір (test suite):

– набір тестів, що реалізують бізнес-завдання, виконуване системою, що тестується;

– тестовий набір включає тестові сценарії, тестові дані або правила їх генерації [21].

До стандартних атрибутів тест-кейса відносять:

– **ID (Test Case ID) –** унікальний ідентифікатор необхідний для зручної організації зберігання і навігації у групі тестів (Test Suite), переважно генерується автоматично;

– **назва** – основна тема чи ідея тест-кейса, короткий опис його суті, який дозволяє зрозуміти призначення тест-кейсу. Може повторювати тему. Поле обов'язкове для заповнення;

– **тема (Summary) –** поле опису основної теми, яка лаконічно і точно описує його суть. Для уніфікованого підходу до викладу теми використовують конструкцію, яка відповідає на питання «Що перевіряємо? Де перевіряємо? З якими даними?». Поле обов'язково для заповнення;

– **попередні умови (Preconditions) –** список всіх необхідних підготовчих дій (налаштування програми, середовища тестування) для виконання даного тест-кейсу;

– **кроки для відтворення (Step actions)** – описують послідовність дій для відтворення тестового випадку, які повинні привести до очікуваного результату. Повинні бути короткими і зрозумілими;

– **очікувані результати (Expected results)** – визначають правильну реакцію програми на виконання даних кроків. Повинні бути зрозумілими, однозначними, простими;

– **історія редагування –** лаконічний журнал змін, де відображено ким, як і коли був змінений тест-кейс [20].

Тестові випадки поділяються за очікуваним результатом на **позитивні** та **негативні**.

Позитивний тест-кейс використовує тільки валідні дані і перевіряє, що додаток правильно виконав функцію, що перевіряється.

Негативний тест-кейс оперує як валідними, так і невалідними даними (мінімум 1 некоректний параметр) і ставить за мету перевірку виняткових ситуацій (спрацьовування валідаторів).

Тестові випадки зручно об'єднувати **за призначенням.**

1 Позитивні тест-кейси:

– перевірка функціоналу;

– перевірка дизайну/UI;

– перевірка безпеки.

2 Негативні тест-кейси [21].

При складанні тест-кейсів необхідно дотримуватися наступних принципів:

– тему тест-кейса необхідно описувати за принципом «Що перевіряється? Де перевіряється? З яким типом даних (валідні, невалідні)?»;

– наявності опису тест-кейса;

– наявність попередніх умов, де має бути вказано, яка форма, сторінка відкрита, але посилання слід вказувати тільки на головний домен сайту – не на конкретну сторінку;

– на кожен крок повинен бути очікуваний результат;

– в кроках має бути зазначений тип даних, що вводяться (валідні/ невалідні), у разі невалідних – повинні бути наведені приклади таких даних (наприклад, для поля введення електронної

пошти невалідними будуть неприпустимі спеціальні символи, без @, без домена та ін.);

– тест-кейс повинен бути завершеним і цілісним (наприклад, функція відновлення пароля повинна закінчуватися на успішній зміні пароля, а не на відправці листа з посиланням для зміни пароля) [20].

7.2 Завдання до роботи

7.2.1 Створити 6 тест-кейсів для сайту, наприклад https://www.automationexercise.com/, для наступного функціоналу:

− форма реєстрації нового користувача;

− форма авторизації користувача;

− форма відновлення паролю до акаунту.

Для кожної форми має бути по 2 тест-кейси – один з позитивним сценарієм (з валідними даними) і один з негативним сценарієм (з невалідними даними).

Приклад шаблону наведено на рисунку 7.1, заповнений тест-кейс на рисунку 7.2 (шаблон можна завантажити з навчальної платформи Moodle).

7.2.2 Зробити звіт та додати файли тест-кейсів у форматі «.xlsx» або «.xls».

Назва		
Тема: що перевіряється? Де перевіряється? З яким типом даних?		
Передумови		
1 2		
№	Кроки	Очікувана реакція
1		

Рисунок 7.1 – Шаблон тест-кейсу

Назва	Реєстрація користувача з валідними даними	
Тема:		
Перевірка роботи системи реєстрації користувача при введенні валідних даних		
Передумови		
1. Користувач не реєструвався на сайті, має валідні дані для вводу 2. Відкритий сайт http://prestashop.qatestlab.com.ua/en/, натиснути на кнопку "Sign in" на головній сторінці сайту 3. Відкрита форма реєстрації користувача		
№	Кроки	Очікуваний результат
1	Ввести валідний email в поле "Email address"	Поле "Email address" підсвічується зеленим кольором
2	Натиснути на кнопку "Create an account"	Система відкриває нову форму для введення додаткових даних
3	Заповнити поле "First name" валідними даними	Поле "First name" підсвічується зеленим кольором
4	Заповните поле "Last name" валідними даними	Поле "Last name" підсвічується зеленим кольором
5	У поле "Password" ввести валідний пароль (більше 6 символів)	Поле "Password" підсвічується зеленим кольором, введені дані приховуються крапками
6	Натиснути на кнопку "Register"	Система переходить на головну сторінку сайту, в правому куті головної сторінки виводиться ім'я та прізвище зареєстрованого користувача

Рисунок 7.2 – Приклад заповненого тест-кейсу

7.3 Зміст звіту

7.3.1 Тема та мета роботи.
7.3.2 Завдання до роботи.
7.3.3 Короткі теоретичні відомості.
7.3.4 Хід виконання завдання до лабораторної роботи.
7.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 7.4 за вибором студента).

7.4 Контрольні запитання

7.4.1 Дайте визначення поняттю тест-кейс та вкажіть його основні атрибути.
7.4.2 Вказати причини того, чому спочатку проводиться позитивне тестування, а потім негативне.
7.4.3 В чому полягає важливість використання тестових випадків у тестуванні ПЗ?
7.4.4 Вказати переваги та недоліки використання тестових випадків у тестуванні ПЗ на прикладі.
7.4.5 В чому полягає різниця між тест-кейсом та звітом про дефект?

8 ЛАБОРАТОРНА РОБОТА №8
ВИДИ ТЕСТУВАННЯ, ПОВ'ЯЗАНІ ЗІ ЗМІНАМИ

Мета роботи: ознайомлення з поняттями регресійного тестування (Regression Testing), димового тестування (Smoke Testing), санітарного тестування (Sanity Testing), тестування збірки (Build Verification Test), отримання базових теоретичних та практичних навичок в проведенні тестувань, пов'язаних зі змінами.

8.1 Короткі теоретичні відомості

Для підтвердження того факту, що баг/дефект був дійсно виправлений, після внесення необхідних змін, програмне забезпечення повинно бути протестоване ще раз. Нижче перераховані види тестування, які необхідно провести для підтвердження працездатності програми або правильності здійсненого виправлення дефекту:

- регресійне тестування (Regression Testing);
- димове тестування (Smoke Testing);
- санітарне тестування або перевірка узгодженості/справності (Sanity Testing);
- тестування збірки (Build Verification Test) [22].

Регресійне тестування (Regression Testing) – у більшості випадків, розглядають, як тестування, яке має на меті переконатися, що нова зміна коду в певному місці програми не вплинула негативно на інші доти працюючі частини програми. Регресійними можуть бути як функціональні, так і нефункціональні тести. Як правило, для регресійного тестування використовуються тест-кейси, написані на ранніх стадіях розробки і тестування. Це дає гарантію того, що зміни в новій версії програми не пошкодили вже існуючу функціональність. Рекомендується робити автоматизацію регресійних тестів, для прискорення подальшого процесу тестування і виявлення дефектів на ранніх стадіях розробки програмного забезпечення [23].

Залежно від контексту використання терміну регресійного тестування, може мати різний зміст. Сем Канер, наприклад, описав три основних типи регресійного тестування.

1 Регресія багів (Bug regression) – спроба довести, що виправлена помилка насправді не виправлена.

2 Регресія старих багів (Old bugs regression) – спроба довести, що нещодавня зміна коду або даних зламала виправлення старих помилок, тобто старі баги стали знову відтворюватися.

3 Регресія побічного ефекту (Side effect regression) – спроба довести, що нещодавня зміна коду або даних зламала інші частини продукту [22].

Димове тестування (Smoke testing). Назва цього виду тестування походить від швидкого способу перевірки інженерами електроприладів: при введенні в експлуатацію нового обладнання вважалося, що тестування пройшло вдало, якщо при увімкнені з установки не пішов дим.

В області тестування програмного забезпечення, димове тестування застосовується для поверхневої перевірки всіх модулів програми на предмет працездатності і наявності швидкого знаходження критичних і блокуючих дефектів. Мета такого тестування перевірити, що після чергової збірки програмного продукту немає явних грубих помилок.

Smoke тести повинні бути швидкими та легкими для того, щоб їх можна було запускати часто.

Санітарне тестування або перевірка узгодженості/справності (Sanity Testing) – це вузьконаправлене тестування, що є достатнім для доказу того, що конкретна функція працює згідно заявленим в специфікації вимогам. Використовується для визначення працездатності лише певної частини програми після внесення деяких незначних змін. Зазвичай виконується вручну.

На відміну від димового, санітарне тестування направлено вглиб функції, що перевіряється, в той час як димове направлено в ширину, для покриття тестами якомога більшої кількості функціоналу в найкоротші терміни.

Тестування збірки (Build Verification Test) – тестування спрямоване на визначення відповідності, випущеної версії, критеріям якості для початку тестування. За своїми цілями є аналогом димового тестування, спрямованого на прийняття нової версії в подальше тестування або експлуатацію. Вглиб воно може проникати далі, залежно від вимог до якості випущеної версії [22].

8.2 Завдання до роботи

8.2.1 Обрати реальний сайт, функціонуючий на даний момент (не тестовий), ознайомитись з функціоналом. Визначити функціонал сайту, до якого можна додати зміни (наприклад для зручності використання). У звіті потрібно навести посилання на даний сайт та вказати мінімум 5 пунктів функціоналу, який потрібно замінити або доповнити, також до кожної вказаної функції навести варіанти заміни або доповнення.

8.2.2 Дати розгорнуту відповідь в звіті у вигляді переліку функціоналу для заміни і того, на який потрібно замінити.

Наприклад: Функціонал «Реєстрація на сайті за допомогою електронної пошти» – доповнити функціоналом «Реєстрація через соціальні мережі».

Важливо! Сайти з доменом ru прийматись не будуть, оскільки використання подібних інтернет-ресурсів заборонено в Україні на рівні законодавства.

8.2.3 Зробити звіт та подати його на перевірку.

8.3 Зміст звіту

8.3.1 Тема та мета роботи.

8.3.2 Завдання до роботи.

8.3.3 Короткі теоретичні відомості.

8.3.4 Хід виконання завдання до лабораторної роботи.

8.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання із пункту 8.4.

8.4 Контрольні запитання

8.4.1 Обґрунтувати важливість та необхідність регресійного тестування, навести приклади.

8.4.2 Вказати можливу причину не виправлення знайдених дефектів, надати обґрунтовану відповідь.

8.4.3 В чому полягає різниця між регресією багів та регресію старих багів? Відповідь обґрунтувати та навести приклади.

9 ЛАБОРАТОРНА РОБОТА №9
ТЕСТ-ПЛАНИ

Мета роботи: ознайомлення з прикладами тест-планів, отримання базових теоретичних та практичних навичок в написанні тест-планів, ознайомлення з системою TestRail та створення тест-плану в TestRail

9.1 Короткі теоретичні відомості

Тест-план (**Test Plan**) – це документ, що описує весь обсяг робіт з тестування, починаючи з опису об'єкта, стратегії, розкладів, критеріїв початку та закінчення тестування, вимог до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення [24].

Rational Unified Process (RUP) – адаптивна ітеративна структура для процесу розробки ПЗ, що складається з чотирьох фаз життєвого циклу проєкту: початок, проєктування, побудова, впровадження [1].

Гарний тест-план повинен, як мінімум, описувати наступне.

1 Що потрібно тестувати? – Опис об'єкта тестування: системи, додатки, обладнання.

2 Що будете тестувати? – Список функцій та опис системи, що тестується, та її компонент окремо.

3 Як будете тестувати? – Стратегія тестування, а саме: види тестування та їх застосування по відношенню до об'єкта тестування.

4 Коли будете тестувати? – Послідовність проведення робіт: підготовка (Test Preparation), тестування (Testing), аналіз результатів (Test Result Analysis) у розрізі запланованих фаз розробки.

5 Критерії початку тестування:

- готовність тестової платформи (тестового стенду);
- завершеність розробки необхідного функціоналу;
- наявність всієї необхідної документації;

– витримка певного періоду без зміни початкового коду додатка Code Freeze (CF);

– витримка певного періоду без відкриття нових дефектів Zero Bug Bounce (ZBB).

6 Критерії завершення тестування – результати тестування задовольняють критеріям якості продукту:

– вимоги до кількості відкритих дефектів виконані;

– витримка певного періоду без зміни початкового коду додатка;

– витримка певного періоду без відкриття нових багів [24].

Види тест-планів.

1 Майстер Тест-план (Master Plan or Master Test Plan).

2 Тест-план (Test Plan).

3 План Приймального Тестування (Product Acceptance Plan) – документ, що описує набір дій, пов'язаних з приймальним тестуванням (стратегія, дата проведення, відповідальні працівники тощо).

Явна відмінність Майстер Тест-плану від просто Тест-плану в тому, що Майстер Тест-план є більш статичним в силу того, що містить у собі високорівневу (High Level) інформацію, яка не схильна до частої зміни в процесі тестування і перегляду вимог. Сам же детальний тест-план, який містить більш конкретну інформацію по стратегії, видам тестування, розклад виконання робіт, є «живим» документом, який постійно зазнає зміни, що відображають реальний стан справ на проєкті.

Зі зразками різних видів тест-планів можна ознайомитись у джерелі [25].

9.2 Завдання до роботи

9.2.1 Створити у TestRail обліковий запис та проєкт за посиланням https://secure.testrail.com/customers/testrail/trial/. Назва проєкту обов'язково має бути у форматі «Прізвище Ініціали» (наприклад «Шевченко Т.Г.»). Скріншот створеного проєкту слід додати у звіт з лабораторної роботи.

Важливо! Доступ до безкоштовної версії TestRail надається тільки на 14 днів.

У TestRail створити тест-план з довільною назвою та описом. Додати тест-ран, додати усі створені тест-кейси. Пройти (призначити статуси) усі тест-кейси (їх має бути мінімум 4) та прикріпити скріншот із діаграмою результатів.

У TestRail-проєкті створити мінімум 2 тестових набори та мінімум 4 тест-кейси відповідно до функціоналу тестового сайту, наприклад https://www.automationexercise.com/, (тест-кейси можна скопіювати із Лабораторної роботи «Тест-дизайн та тест-кейси»). В одному тестовому наборі має бути 2 тест-кейси (негативний та позитивний) з тестування однієї функції сайту. Тема тест-кейсу має бути сформульована за принципом «Що? Де? З яким типом даних?»

У TestRail створити та пройти тест-рани, зробти скріншоти .

Усі скріншоти додати до звіту.

9.2.2 Створити стислий тест-план для сайту, наприклад https://www.automationexercise.com/, згідно прикладу.

Приклад тест-плану наведено в Додатку А.

Порядок виконання:

Реєстрація у TestRail

Даний інструмент є платним, але існує безкоштовна пробна версія на 14 днів. Цього цілком достатньо для виконання лабораторної роботи.

Для початку процесу реєстрації акаунта слід перейти за посиланням https://www.testrail.com/ та вибрати кнопку «Try for Free».

Далі відкривається форма реєстрації акаунта та можливість вибору версії для встановлення на сервер чи робота у хмарі. Для ознайомлення та індивідуальної роботи достатньо буде онлайн-інструмента.

Після підтвердження вказаного імейла, користувач автоматично переноситься на особисту дошку, де і відбувається практично вся робота.

Створення проєкту

Щоб почати роботу у системі, потрібно створити проєкт. Він є основною організаційною одиницею TestRail. Зазвичай рекомендується створювати новий проєкт для кожного реального проєкту. Тестові прогони, результати тестів, етапи та інші дані безпосередньо пов’язані з кожним конкретним проєктом.

Спочатку треба натиснути на кнопку «Add project».

Далі відкривається форма для заповнення (рисунок 9.1). Поле «Ім'я проєкта» є обов'язковим, його анонс та варіанти організації тестових наборів:

– **єдине сховище для всіх випадків (рекомендовано).** Єдиний набір тестів (репозиторій) є простим в управлінні та достатньо гнучким для більшості проєктів з відсутністю або декількома одночасними версіями. Можна використовувати розділи та підрозділи для подальшої організації тестів.

– **єдине сховище з базовою підтримкою.** Єдиний набір тестів (репозиторій) із додатковою опцією для створення базових планів для керування декількома гілками тестів одночасно. Це ідеальний варіант, якщо потрібно тестувати кілька версій проєкта паралельно.

– **кілька наборів тестів для керування випадками.** Кілька наборів тестів можуть бути корисними для організації тестових випадків за функціональними областями та модулями програми на рівні тестового пакету. Це традиційний режим TestRail, який автоматично використовується для оновлених проєктів.

При реєстрації слід використати довільну пошту.

На формі реєстрації слід вибрати TestRail Cloud – для отримання онлайн версії середовища.

Назва проєкту повинна містити прізвище та ім'я студента.

На етапі створення проєкта слід вибрати форму організації – Use a single repository for all cases (recommended).

Зробити скріншот створеного проєкта та додати у звіт з лабораторної роботи. Скріншот має виглядати як на рисунку 9.2.

Останнім кроком натискається кнопка «Add project».

Після створення проєкту відображається меню, діаграма, перелік і кнопки для створення подій та тест-ранів, дії з проєктом (додавання та перегляд подій, тест-ранів, тест-кейсів) (рисунок 9.3).

Створення тест-кейсу

Спочатку треба натиснути на пункт меню «TEST CASES» (рисунок 9.4).

Якщо проєкт новий і тест-кейси для нього ще не додавались, тоді з'являється сторінка з досить обмеженими функціями: «Add Section» (додати розділ) та «Add Test Case» (додати тест-кейс).

Після натискання на кнопку «Add Test Case» з'являється форма для заповнення полів тест-кейса:

1 Title – тема тест-кейса (є обов'язковим).

2 Section – випадаючий список із варіантами вибору розділу, в якому буде відображатись створений тест-кейс. За замовчуванням створено розділ «Test cases» (є обов'язковим).

3 Template – шаблон форми заповнення тест-кейса (є обов'язковим). Є три варіанта вибору:

- **Test Case (Text)** встановлено за замовчуванням – має поле для передумов, одне текстове поле для всіх кроків та одне текстове поле для єдиного очікуваного результату.
- **Test Case (Steps)** – є поле для передумов та відрізняється тим, що має окреме текстове поле для кожного кроку та кожного очікуваного результату. За допомогою кнопки «Add Step» додається пара полів з нумерацією.
- **Exploratory Session** (дослідницька сесія) – замість передумов з'являється поле «Mission» для опису задач тест-кейса, та поле «Goals» для опису його цілей.

4 Type – тип тестування (є обов'язковим). Варіанти: Other (за замовчуванням), Accessibility, Automated, Compatibility, Destructive, Functional, Performance, Regression, Security, Smoke & Sanity, Usability.

5 Priority (є обов'язковим) – пріорітет тест-кейса. Доступні варіанти: Critical, High, Medium, Low.

6 Estimate – оцінка тест-кейса.

7 References – довідкова інформація/посилання на джерела.

8 Automation Type – поле заповнюється, якщо для поточного тест-кейса використовується автоматизоване тестування.

9 Preconditions – поле для опису передумов тест-кейса. Якщо використовується шаблон Exploratory Session, тоді даного поля не буле, але з'явиться поле «Mission» для опису задач тест-кейса.

10 Steps – поле для опису кроків тест-кейса.

11 Expected Result – очікуваний результат. При використанні шаблона Test Case (Text) буде одне поле, яке знаходиться під полем із кроками. Якщо ж вибрано шаблон Test Case (Steps) – полів для кроків та результатів буде однакова кількість (очікувані результати будуть описуватись для кожного кроку окремо), знаходитимуться вони одне біля одного. Кожна пара пронумерована.

Для кожного тест-кейсу слід вибрати шаблон Test Case (Steps).

В одному тест-сьюті (тестовому наборі) має бути два тест-кейси (негативний та позитивний) з тестування однієї функції сайту.

Після заповнення полів необхідними даними, натиснути на кнопку «Add Test Case».

Як тест-кейс буде створено, відображається сторінка із переглядом доданого прикладу (рисунок 9.5).

Створення тестових наборів

У TestRail тестові набори називаються **Section** та **Subsection** (розділи та підрозділи).

Для додавання нового розділу слід натиснути на кнопку «Add Section».

З'являється форма для заповнення назви розділу (обов'язково), опису та додавання медіафайлів (не обов'язково). Далі слід натиснути кнопку «Add Section» внизу форми.

Для створення підрозділу до даного розділу слід натиснути на кнопку «Add Subsection».

З'являється форма, аналогічна до створення розділу, яка заповнюється відповідними даними, та натискається кнопка створення підрозділу.

Після маніпуляцій по створенню одного розділу з назвою «Sign in» та двох підрозділів «Authorization» і «Registration», відображається така структура:

Раніше було створено тест-кейс по авторизації з валідними даними, але додано його в розділ за замовчуванням (який, до речі, можна видалити або відредагувати). Тож цей приклад можна легко перенести у необхідний розділ/підрозділ. Для цього необхідно лише навести мишкою на створений тест-кейс, затиснути та перетягнути його в необхідне місце.

З'являється меню з опціями **Move here** (перемістити тест-кейс), **Copy here** (скопіювати тест-кейс) або **Cancel** (відмінити дію).

Спосіб організації розділів залежить від розміру проєкта. Якщо є багато тестових випадків, рекомендується додатково розбити розділи та створити підрозділи для певних функцій.

Створення тест-плану

Якщо потрібно керувати кількома тестовими прогонами та конфігураціями для одного проєкту, стають у пригоді тест-плани TestRail. План тестування дає змогу запускати кілька тестових прогонів одночасно, якщо потрібно протестувати кілька конфігурацій.

Це може бути все, що потрібно для тестування проєктів, наприклад, різні операційні системи або браузери.

Для створення тест-плану потрібно перейти на вкладку «TEST RUNS & RESULTS», натиснути на кнопку «Add Test Plan».

З'являється форма для заповнення назви тест-плана (Name) – обов'язково, етапу (Milestone) та опису (Description) – не обов'язково (рисунок 9.6).

Далі слід натиснути на кнопку «Add Test Plan».

Тест-план створено. Відображається діаграма активності та прогресу в роботі з ним.

Створення та проходження тест-ранів.

Перейти на вкладку «TEST RUNS & RESULTS», натиснути на кнопку «Add Test Run»:

З'являється форма для заповнення з такими полями:

– **Name** – назва тест-рана (є обов'язковим);

– **References** – ідентифікатори/посилання на зовнішні тікети;

– **Milestone** – етап, якому належить цей тестовий прогін;

– **Assign To** – користувач, якому призначені тести нового тестового запуску. Цій людині буде надіслано електронний лист, якщо сповіщення електронною поштою ввімкнено;

– **Description** – опис мети цього тестового запуску;

– **Include all test cases** – параметр, з яким включено всі тестові випадки в цей тестовий запуск. Якщо нові тест-кейси додаються до репозиторію, вони також автоматично включаються в цей запуск;

– **Select specific test cases** – вибір тестових випадків, які потрібно включити в цей тестовий запуск. Нові тести не додаються автоматично до цього циклу;

– **Dynamic Filtering** – автоматичне додавання тестів на основі вибору фільтра. Нові тест-кейси автоматично додаються до серії, якщо вони відповідають фільтру (якщо серія не закрита).

Після заповнення всіх необхідних полів слід натиснути на кнопку «Add Test Run». Відображається сторінка із діаграмою та переліком включених до цього тест-рана тест-кейсів.

На рисунку 9.6 тестові випадки ще не пройдені, тому мають статус «Untested», а сам тест-ран пройдено на 0%. Для того, щоб відмітити проходження тестового випадку, треба натиснути на випадаючий список «Untested», який знаходиться справа від назви тест-кейсу.

Кожен з них має п’ять статусів:

– **Untested** (не протестовано) – за замовчуванням всім новим тестам присвоюється даний статус. Після того, як результат було додано до тесту, він більше ніколи не отримає цей статус;

– **Passed** (тест-кейс успішно пройдено) – коли очікувані результати співпадають із фактичним проходженням теста;

– **Blocked** (проходження тест-кейса заблоковано) – наразі тест неможливо виконати через певну зовнішню залежність;

– **Retest** (тест-кейс слід пройти ще раз) – наприклад, якщо спочатку тест не вдався і розробник вирішив проблему, можна призначити тест-кейс для повторного тестування;

– **Failed** (тест-кейс не вдалося пройти/провалено) – якщо один із зазначених кроків тесту призвів до помилки або якщо очікуваний результат відрізняється від фактичного результату тесту.

Після вибору однієї з опцій (в даному випадку Passed), з’являється невелика форма із результатом перевірки, яка має такі поля:

– **Status** – статус тест-кейса (є обов’язковим). В залежності від варіанту вибору змінюється колір віконечка;

– **Comment** – опис результата теста;

– **Assign To** – призначення тест-кейса іншому члену команди;

– **Version** – тестова версія;

– **Elapsed** – скільки часу тривав тест;

– **Defects** – перелік ID баг-репортів, які були оформлені під час поточного проходження тест-кейса.

При створенні тест-рана слід обрати параметр «Include all test cases», щоб всі створені тест-кейси було додано автоматично.

Пройти необхідно всі тест-кейси (їх має бути мінімум 4).

Зробити скріншот пройденого тест-рана та додати у звіт з лабораторної роботи. Скріншот має виглядати як показано на рисунку 9.7.

Результатом проходження тест-кейсів поточного тест-рана буде подібна діаграма з відсотками кожного із статусів. Можна переглянути результати тестування та активність тестування прогонів, етапів і цілих проєктів на сторінках окремих ресурсів. Зробити скріншот одного тест-рана на вибір з усіма атрибутами, оскільки

вмістити усі тест-кейси з усіма атрибутами на одному скріншоті неможливо.

Після усунення дефекту можна скористатися функцією повторного запуску, щоб ще раз виконати тест та вивести нові результати [26].

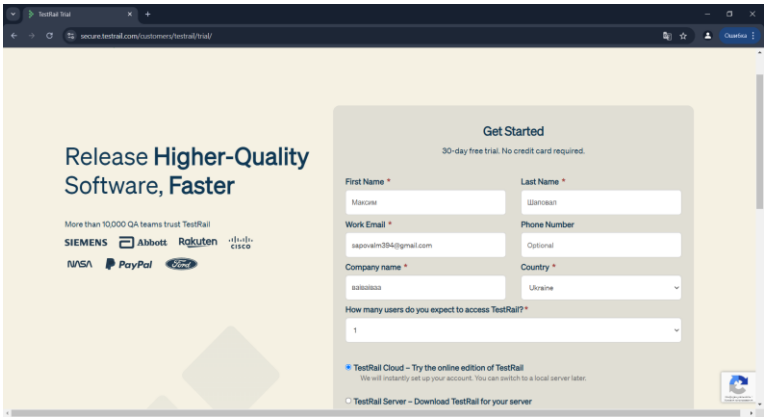


Рисунок 9.1 – Реєстрація на сайті TestRail

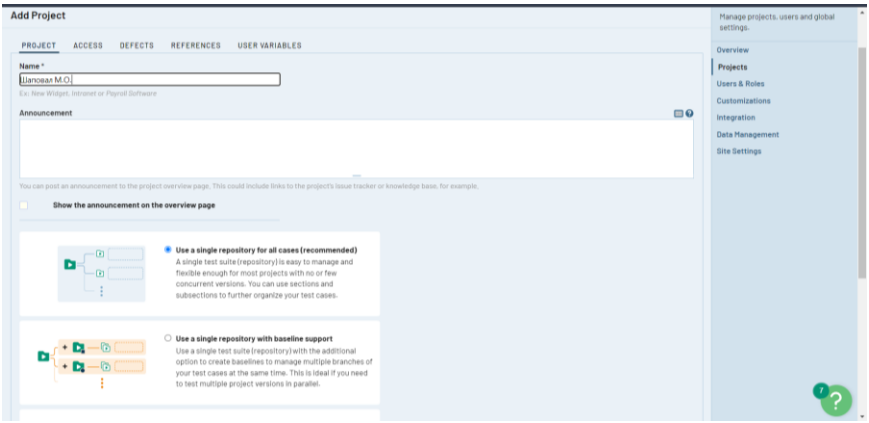


Рисунок 9.2 – Створення проєкту

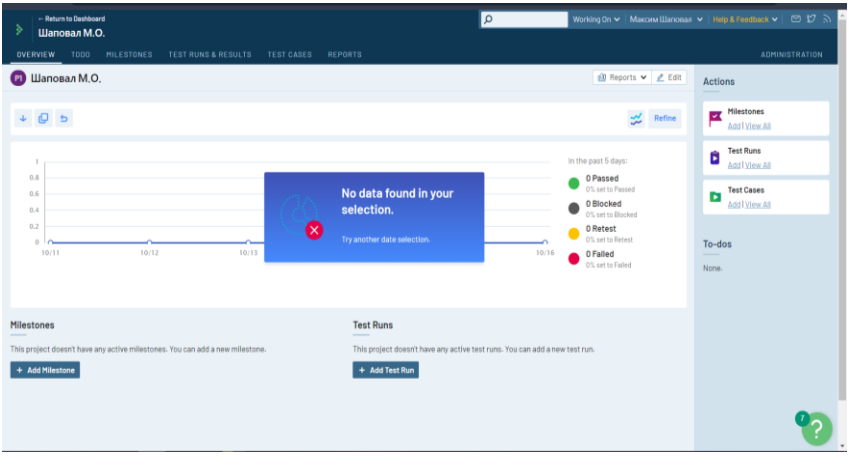


Рисунок 9.3 – Створений проєкт

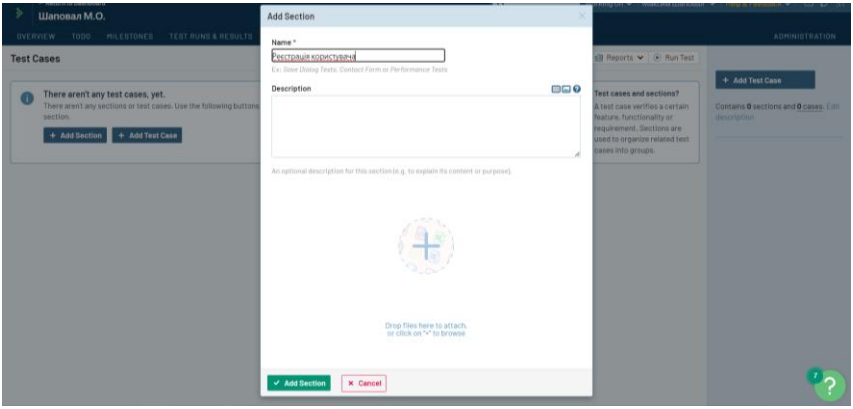


Рисунок 9.4 – Створення секції «Реєстрація користувача»

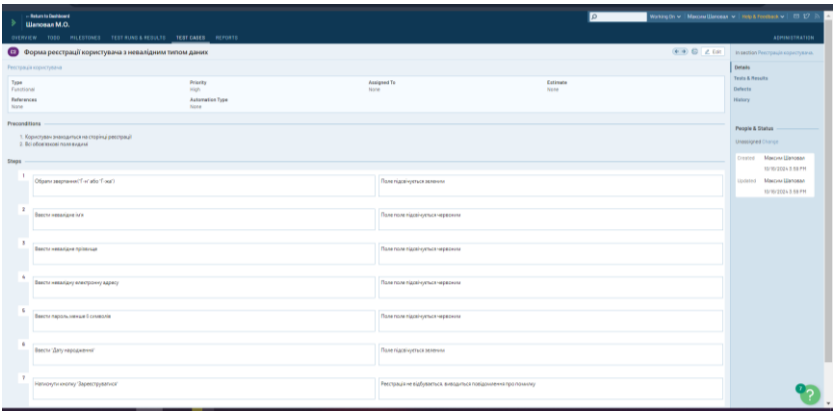


Рисунок 9.5 – Створений тест-кейс «Форма реєстрації користувача з невалідним типом даних»

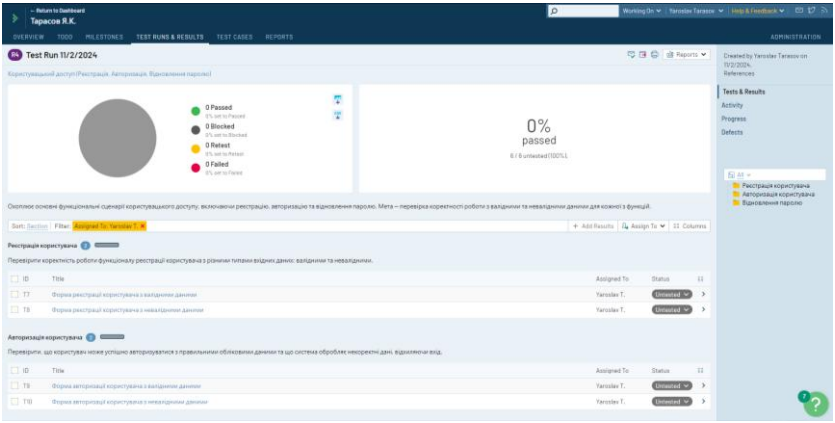


Рисунок 9.6 – Створений тест-ран

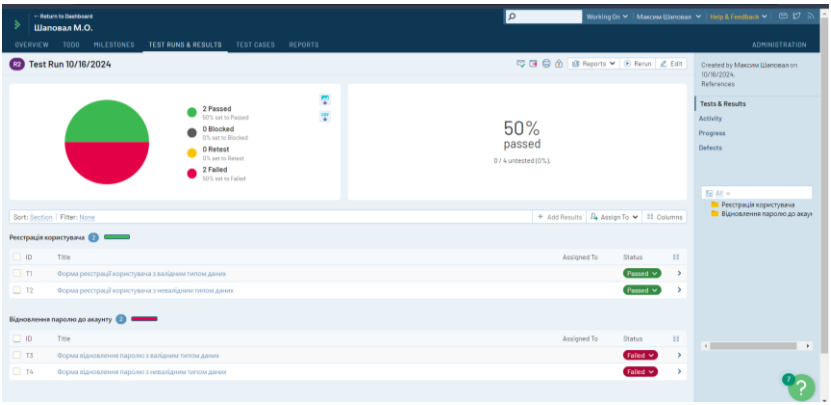


Рисунок 9.7 – Діаграма з відсотками кожного із статусів

9.3 Зміст звіту

9.3.1 Тема та мета роботи.
9.3.2 Завдання до роботи.
9.3.3 Короткі теоретичні відомості.
9.3.4 Хід виконання завдання до лабораторної роботи.
9.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 9.4 за вибором студента).

9.4 Контрольні питання

9.4.1 Для чого використовується система TestRail? Розповісти про основні можливості системи.
9.4.2 В чому різниця тест-планів і тестової стратегії?
9.4.3 З якою метою пишуть тест-плани, чи можна провести тестування програмного продукту без тест-плану? Надану відповідь обґрунтувати та навести приклади на практиці.
9.4.4 З якою метою проводиться рецензування та затвердження тест-плану?
9.4.5 Хто затверджує та рецензує тест-план?

10 ЛАБОРАТОРНА РОБОТА №10
ЗВІТИ ПРО ТЕСТУВАННЯ

Мета роботи: ознайомлення з поняттям звітності в тестуванні та інструментарієм для створення тестової звітності, знайомство з системою Jira та отримання базових теоретичних та практичних навичок в складанні звітів про тестування.

10.1 Короткі теоретичні відомості

Звіт потрібен для того, щоб дати розробнику інформацію про поточний стан програмного продукту. Повна інформація про стан продукту є тільки у команди, що приймала участь в процесі тестування, яка перевірила всі функції додатку, володіє інформацією про кількість дефектів та динаміку зміни кількості, наприклад, порівняно з попереднім білдом [27].

Звіт про результати тестування в основному потрібен:

− менеджеру проєкту;

− лідеру команди розробників;

− лідеру команди тестувальників;

− замовнику.

Зазвичай для складання звітів про тестування використовуються спеціальні програми або вебсервіси. Найбільш популярними є Jira, TestRail, Asana, Basecamp, Wrike, Trello та ін.

Якщо звіт пишеться для розробника або менеджера, то ніяких запитань щодо надання звіту не виникає – необхідна термінологія та опис проблем буде легко сприйматися людиною, яка буде читати представлений звіт. Але якщо поставлена мета написати звіт для людей, які не знайомі зі специфікою та технічною стороною програмного забезпечення, то з’являється задача, для вирішення якої краще використовувати графічний вид надання інформації: графіки та діаграми [27].

Це може бути, наприклад, графік із пройденими тест-кейсами з кількістю дефектів (рисунок 10.1).

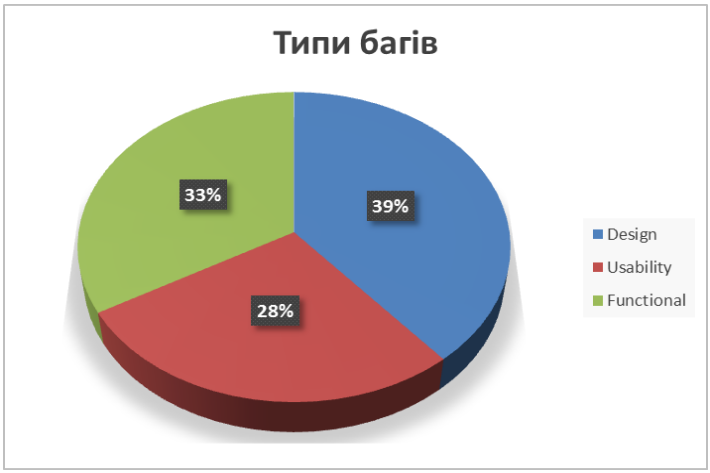


Рисунок 10.1 – Діаграма пройдених тест-кейсів і кількості дефектів

Також це може бути, наприклад, графік пройдених тестових випадків (test case) по модулям, який наочно покаже, який обсяг роботи в кожному з модулів вже виконаний та допоможе виявити проблеми.

На додаток до діаграми або графіка необхідно надати зведену таблицю з усіма даними щодо тестування (таблиця 10.1).

Таблиця 10.1 – Зведена таблицю з підсумковими даними тестування

Модуль	Загальна кількість ТК по модулю	Проходження	
		passed	not passed yet
Модуль 1	6	1	5
Модуль 2	8	6	2
Модуль 3	1	0	1

Приклад звіту сайту у Додатку Б.

10.2 Завдання до роботи

10.2.1 Створити обліковий запис і проєкт з розробки програмного забезпечення в Jira з назвою в форматі «Прізвище Ініціали» (наприклад, «Коваленко І.О.») за посиланням https://www.atlassian.com/try/cloud/signup та додати скріншот створеного проєкту до звіту з лабораторної роботи (рисунок 10.2).

10.2.2 Створити скрам-дошку в проєкті з назвою в форматі «Прізвище Ініціали» (наприклад «Коваленко І.О.») та додати скріншот до звіту з лабораторної роботи (рисунок 10.3).

10.2.3 Оформити в Jira один баг-репорт з сайту, наприклад https://www.automationexercise.com/. Прикріпити скріншот баг-репорта до звіту з лабораторної роботи (рисунок 10.4)

10.2.4 Скласти звіт про тестування сайту із пункту 10.2.3, виходячи з результатів виконання лабораторних робіт «Кросбраузерне тестування», «Тестування вебпроєктів», «Функціональне тестування». Приклад звіту наведений у Додатку Б.

Порядок виконання.

1 Реєстрацію потрібно здійснити безкоштовно на сайті https://www.atlassian.com/try/cloud/signup, використавши електронну адресу та довільну назву сайту (рисунок 10.2).

Важливо! Зверніть увагу, що створений обліковий запис в Jira доступний тільки протягом 14 днів. Після цього акаунт видаляється і для виконання завдання потрібно буде реєструвати новий.

На етапі «Додавання відомостей про проєкт» потрібно **вказати назву проєкту «Прізвище Ініціали» студента**, а також **обрати шаблон проєкту замість Kanban – Scrum**.

Після реєстрації потрібно створити проєкт з назвою «Прізвище Ініціали» через відповідну кнопку «Create project». **Важливо обрати тип проєкту «Company managed».**

Зробити скріншот створеного проєкту та прикріпити до звіту з лабораторної роботи. Скріншот має виглядати як на рисунку 10.2.

На скріншоті не має відображатись панель закладок браузера, проте обов’язково має бути відображено адресний рядок браузера. Скріншот додати до звіту.

2 Створення дошки здійснюється шляхом проходження наступних кроків: Your work → Boards → View all boards → Create board → Create a Scrum board → Board from an existing project.

На скріншоті не має відображатись панель закладок браузера, проте обов'язково має бути відображено адресний рядок браузера. Скріншот додати до звіту (рисунок 10.3.).

3 Тема баг-репорта повинна бути сформована за принципом «Що? Де? Коли?».

В розділі «Опис» потрібно описати кроки відтворення дефекту та результати.

Додати тестове середовище та прикріпити скріншот/відео (Зверніть увагу, якщо на формі створення баг-репорта відсутні деякі необхідні поля, то їх можна додати натиснувши кнопку «Configure fields»).

Скріншот повинен бути з адресним рядком проте не має відображатись увімкнена панель закладок браузера.

Якщо опис теми, опису та кроків з результатами займає чимало місця, то всі атрибути баг-репорту можуть не поміститись на одному скріншоті. В такому випадку слід або зробити складний скріншот з двох простих або зменшити масштаб вебсторінки. Скріншот додати до звіту.

При створенні звіту обов'язково необхідно вказувати

1 Склад команди.

2 Терміни виконання, за які складається звіт.

3 Опис процесів тестування.

4 Зміни тестової моделі, доповнення ТК.

5 Відсоток пройдених ТК.

6 Критичні та блокуючі проблеми та вжиті заходи по їх усуненню.

7 Результати регресу (акцент на невирішених проблемах).

8 План на наступну ітерацію\тиждень\місяць.

Пункти 3, 4, 6 та 8 варто писати з урахуванням цільової аудиторії звіту, 7 пункт варто вказувати тоді, коли проводилося «регрестестування», зазвичай цей пункт фігурує в «версійних» звітах. Пункт 8 з підсумкового звіту виключається.

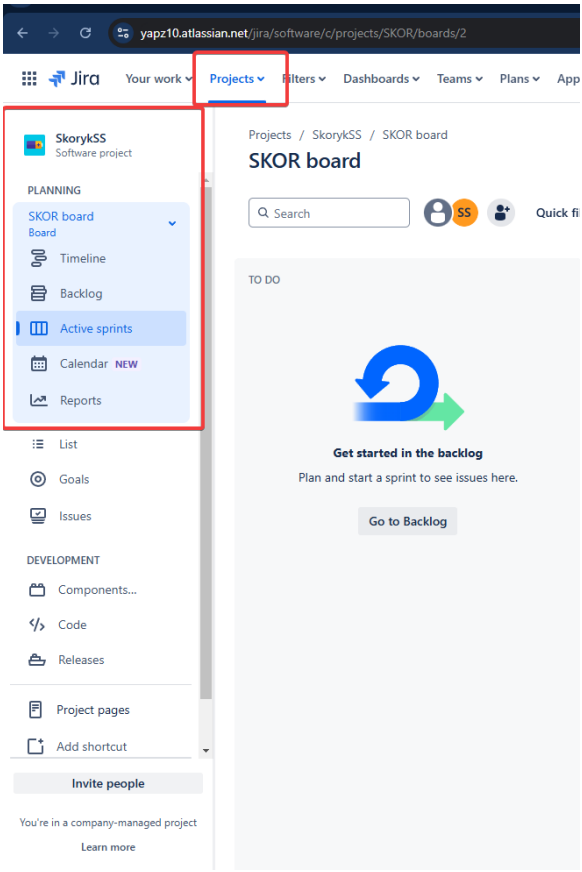


Рисунок 10.2 – Створений проєкт у Jira

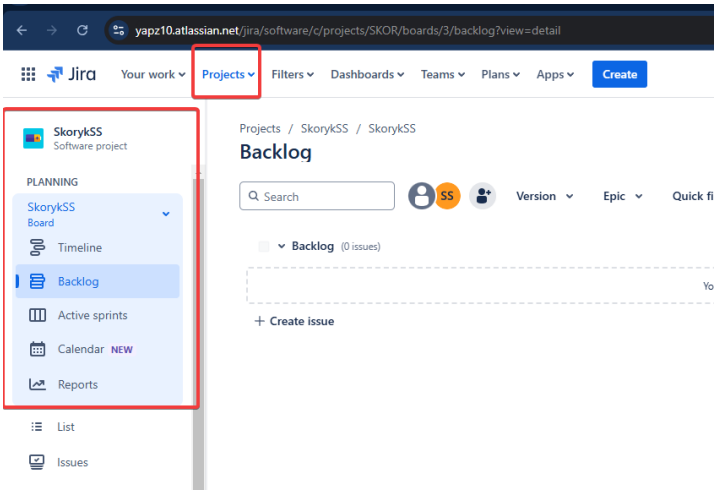


Рисунок 10.3 – Створена скрам-дошка «SkorykSS»

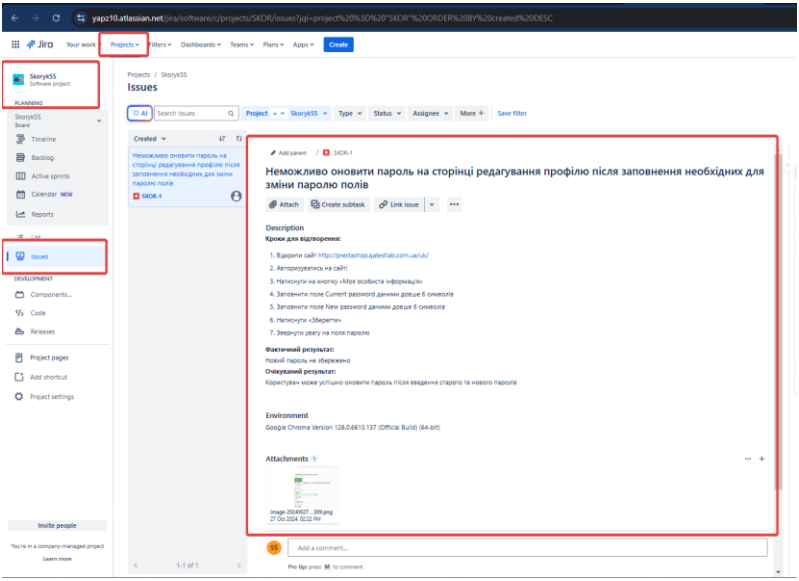


Рисунок 10.4 – Баг-репорт на тему «Неможливо оновити пароль на сторінці»

10.3 Зміст звіту

10.3.1 Тема та мета роботи.
10.3.2 Завдання до роботи.
10.3.3 Короткі теоретичні відомості.
10.3.4 Хід виконання завдання до лабораторної роботи.
10.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 10.4 за вибором студента).

10.4 Контрольні питання

10.4.1 Які основні пункти входять до звіту тестування? Обґрунтувати їх необхідність.
10.4.2 На яку аудиторію орієнтувалися при написанні звіту про тестування? Надану відповідь обґрунтувати.
10.4.3 Яку інформацію обов'язково потрібно вказувати при написанні будь якого звіту про тестування?
10.4.4 Які додаткові документи можуть бути додані до звіту про тестування?
10.4.5 Назвати найпоширеніші системи для управління проєктами та подачі стислих звітів.

11 ЛАБОРАТОРНА РОБОТА №11
МОБІЛЬНЕ ТЕСТУВАННЯ

Мета роботи: ознайомлення з основними відмінностями мобільних вебсайтів та додатків від їх аналогів на ПК, отримання базових теоретичних та практичних навичок тестування додатка на мобільному пристрої, складання контрольного списку для тестування додатка.

11.1 Короткі теоретичні відомості

Тестування мобільних додатків – процес, під час якого програмна частина додатку для мобільних пристроїв тестується на функціональність, зручність та змістовність. Таке тестування може бути ручним або автоматизованим [28].

Основні відмінності мобільних та десктопних додатків:

- розмір екрану;
- датчики та пристрої введення;
- наявність телефонних функцій;
- енергоспоживання;
- робота з мережею;
- особливості платформи;
- вузька спеціалізація;
- норми та рекомендації [28].

Мобільний вебсайт – спеціалізований сайт, адаптований для перегляду та функціонування на мобільному пристрої.

Тестування вебдодатків має дати оцінку якості мобільних вебдодатків при використанні різних браузерів на різних мобільних пристроях. На відміну від мобільних додатків, вебдодатки зазвичай дозволяють звертатися до функцій базового серверу через тонкий мобільний клієнт. Таким чином, крім аналізу функціональності та поведінки, виконання вимог до якості обслуговування (*Quality of Service – QoS*), зручності використання, безпеки та конфіденційності, тестування мобільних вебдодатків повинно враховувати ще й зв’язність [28].

Мобільний додаток – це спеціально розроблений додаток під конкретну мобільну платформу або декілька платформ.

Спеціально призначені для тестування програми встановлюються та запускаються на мобільних пристроях і, як правило, звертаються до спеціальних інтерфейсів API (наприклад, до API GPS) та апаратних компонентів (наприклад, до камери). Вебдодатки складаються з сервера додатків та клієнтського ПЗ, виконуваного в середовищі браузерів, через які користувачі отримують доступ до сервісів [29].

Тестування додатків, спеціально призначених для мобільних пристроїв, дає змогу оцінити якість мобільних програм, які завантажуються та виконуються на різних мобільних пристроях обраної мобільної платформи. Тестування спрямоване на аналіз функціональності та поведінке (у тому числі підтримки специфічних для пристрою функцій – наприклад, управління за допомогою жестів), виконання вимог до QoS, зручності використання, безпеки та конфіденційності.

У баг-репорті при тестуванні сайту на мобільному пристрої назву браузера та його версію необхідно вказати в розділі «Additional Information».

У першому кроці не треба вказувати посилання на сайти, де можна завантажити додаток, достатньо написати його назву в полі «Additional Information».

Під час написання баг-репортів необхідно використовувати спеціальну термінологію щодо дій на мобільному пристрої (наприклад: tap, swipe left/right/up/down, pinch in/out) [28].

Приклад баг-репорта наведено на рисунку 11.1.

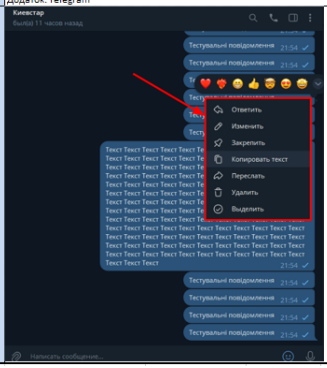
номер ID	Проєкт	Категорія	Рівень доступу	Дата реєстрації	Дата зміни
1	Telegram	функціонал	загальний	14.02.2025	
Тестувальник	Аделєв Дмитро	Пріорітет	середній	Серйозність	незначна
Відтворюваність	постійно	Статус	отримано	Резолюція	відкритий
Платформа	Mobile	Операційна система	Android	Версія ОС	12
Тема	Відсутне сповіщення про копіювання тексту повідомлення.				
Опис	Відсутне сповіщення про копіювання тексту повідомлення.				
Кроки для відстворення	1. Відкрити додаток 2. Перейти до чату де є повідомлення 3. Натиснути на повідомлення правою кнопкою миші 4. Натиснути скопіювати повідомлення 5. Звернути увагу екран				
Фактичний результат	Повідомлення про копіювання не з'являється на десктопній версії				
Очікуваний результат	Після натискання на кнопку "скопіювати повідомлення" має з'явитись повідомлення про успішне копіювання				
Додаткова інформаціs	Додаток: Telegram				
Прикріплені файли					

Рисунок 11.1 – Баг-репорт на тему «Неможливо оновити пароль на сторінці редагування профілю»

11.2 Завдання до роботи

11.2.1 Встановити будь-який безкоштовний додаток з магазину на мобільний пристрій та провести тестування мобільного додатку, використовуючи контрольний список.

11.2.2 Завантажити зі сторінки дисципліни в навчальній системі Moodle шаблон чекліста (рисунок 3.1).

11.2.3 Вказати прізвище та ім’я власника у назві контрольного списку.

11.2.4 Додати мінімум 5-7 пунктів, які відповідають тестуванню на мобільному пристрої, до існуючих пунктів контрольного списку на вкладці «Моб.пристрій (додаток)», не повторюючись з вже доданими, та відмітити їх будь-яким кольором.

11.2.5 Провести тестування мобільного додатку, перевіривши всі пункти контрольного списку на вкладці «Моб.пристрій (додаток)».

11.2.6 Вказати назву/модель мобільного пристрою та версію ОС. Результат перевірки відзначити «Passed/Failed». Для «Failed» вказати в примітці або нотатках посилання на баг-репорти (не менше 5 штук), для всіх інших сформулювати тему багу за принципом «Що? Де? Коли?».

11.2.7 Доповнений та пройдений чекліст в форматі «.xlsx» або «.xls» та баг-репорти додати до звіту.

11.3 Зміст звіту

11.3.1 Тема та мета роботи.

11.3.2 Завдання до роботи.

11.3.3 Короткі теоретичні відомості.

11.3.4 Хід виконання завдання до лабораторної роботи.

11.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 11.4 за вибором студента).

11.4 Контрольні питання

11.4.1 Дайте визначення поняттю «тестування мобільних додатків».

11.4.2 Вказати особливості при формуванні контрольного списку для тестування мобільного додатку.

11.4.3 Перелічіть основні програми для налагодження додатків (Android/iOS).

11.4.4 Обґрунтувати або спростувати доцільність використання емуляторів при тестуванні мобільних додатків.

12 ЛАБОРАТОРНА РОБОТА №12
ТЕСТУВАННЯ ІГОР

Мета роботи: ознайомлення з видами тестування ігор, отримання базових теоретичних та практичних навичок в розпізнаванні дефектів в іграх за видами..

12.1 Короткі теоретичні відомості

Види тестування ігор [30].

1 Функціональне тестування. Мета – виявити відхилення від функціональних вимог. Зводиться до багаторазового проходження гри, виявленню неполадок й умов, в яких їх можна помітити.

2 Навантажувальне тестування. При тестуванні ігор доцільно створити ситуації, які вимагають великого обчислювального навантаження. Таким чином, тестувальник може перевірити продуктивність системи в стресовій ситуації. Так легше помітити й вчасно виправити потенційно ненадійні ділянки коду.

3 Тестування локалізації. Ігри часто перекладають на мови тих країн, в яких їх планують випускати на ринок. У деяких випадках, перекладачі не можуть надати абсолютно точний переклад, який би повністю відповідав подіям гри. Навіть правильно перекладена фраза може не відповідати ситуації та не сприйматися носієм мови. Тому, після локалізації корисно провести тестування гри мешканцями тих країн, де передбачається реалізація кінцевого продукту.

4 Тестування на сумісність. Найчастіше програмування ігор проводять на персональних комп’ютерах або ноутбуках. Однак багато ігор можуть бути призначені для інших пристроїв: ігрових приставок, мобільних телефонів, комунікаторів та ін. Розробка здійснюється на симуляторах даних пристроїв, однак вони можуть значно відрізнятися від оригіналу. Тому, надалі можуть виникнути труднощі при запуску гри на оригінальному пристрої. Крім того, слід звернути особливу увагу на ліцензування програмного забезпечення.

При будь-яких відхиленнях гру можуть повернути на доопрацювання, що займає додатковий час та втрату грошових

коштів. Отже, дуже важливо перевірити гру на відповідність заявленим параметрам апаратного забезпечення.

12.1.1 Ігрові механіки (game mechanics) [30]

Ігрові механіки – внутрішньо-ігрові механізми, які допомагають гравцям досягти мети гри, іншими словами це визначений набір правил та петель зворотного зв'язку, які призначені для створення сценаріїв комп'ютерної гри (gameplay), які приносять задоволення від ігрового процесу та є будівельними блоками, які можуть застосовуватися та комбінуватися з ігровим й не ігровим контекстом.

Досягнення (Achievement). Віртуальне або фізичне втілення будь-якого звершення. Зазвичай їх розглядають як нагороди.

Приклад: Значок, рівень, нагорода, бали, по суті все, що можна прийняти за нагороду може бути «ачівкою».

Механіка призначеної зустрічі (Appointment Dynamic). Динаміка, в якій для досягнення успіху необхідно повернення в заздалегідь певний момент часу для вжиття будь-яких дій. Механіку призначеної зустрічі часто об'єднує міцний зв'язок з режимами інтервальних винагород або вибіркових.

Приклад: При поверненні в гру в певний час, користувач отримує якесь заохочення.

Уникнення (Avoidance). Акт, що спонукає гравця до дій, за допомогою не винагороди, а уникнення покарання. Викликає ряд послідовних дій узгоджених за часом з розкладом.

Приклад: Натискайте важіль кожні 30 секунд, щоб не отримати удар струмом.

Поведінковий контраст (Behavioral Contrast). Теорія, яка визначає те, як сильно може змінюватися поведінка відповідно до змін очікувань.

Приклад: За виконання квесту гравцеві дають 100 досвіду в перший раз. Надалі – по 5000. При виконанні цього ж квесту, скажімо, в десятий раз – йому дають 100 досвіду. Як наслідок – швидше за все гравець перестане виконувати цей квест. Приклад психогогічної маніпуляції із повсякдення: спочатку штучно роблять погіршення умов, потім повертають до попереднього стану, що психологічно сприймається як суттєве покращення.

Поведінковий імпульс (Behavioral Momentum). Тенденція в поведінці гравців до продовження виконання тих дій, якими вони займалися раніше.

Приклад: Прикладом може стати фарм і гринд в різних іграх, тобто постійне виконання однієї і тієї ж дії, з надією отримати бажаний результат/нагороду.

Вдячна продуктивність (Blissful Productivity). Ідея про те, що в процесі гри ви відчуваєте себе щасливіше важко працюючи, ніж відпочиваючи і розслабляючись.

Приклад: Провівши на роботі 6 годин ви, в результаті, отримали менше ніж провівши ці ж 6 годин у своїй улюбленій грі. Мається на увазі така собі залежність від позитивних ефектів і схвалення в іграх. І чим більше проводиться в них часу, тим більше цей показник.

Механіка каскадної подачі інформації (Cascading Information Theory). Теорія говорить про те, що для досягнення належного рівня розуміння в кожному описовому моменті гри інформація повинна подаватися мінімально можливими фрагментами.

Приклад: Подача сюжету в іграх. Туторіали.

Ланцюги подій (Chain Schedules). Практика зв'язування винагород з серіями непередбачених ситуацій. Гравці схильні сприймати це як прості індивідуальні події. Відкриття одного кроку в послідовності часто розглядається гравцем як окрема винагорода.

Приклад: Вбити 10 орків, щоб потрапити в печери дракона, де кожні 30 хвилин з'являється дракон.

Товариські ігри (Companion Gaming). Товариська гра – це гра, в яку можуть грати одночасно користувачі різних платформ.

Приклад: Ігри доступні на iphone, facebook, xbox з безшовним перехресно-платформним ігровим процесом.

Обставини (Contingency). Проблема, яку гравець повинен подолати в тричастинній парадигмі графіків винагород.

Приклад: 10 орків блокують твій шлях.

Відлік (Countdown). Механіка, в якій гравець наділений обмеженою кількістю часу, для здійснення яких-небудь дій. Створює графік активності, який примушує первісну активність гравця драматично збільшуватися, поки не скінчиться час, і не відбудеться примусове завершення.

Приклад: Ігри з таймерами для отримання максимально можливої кількості очок. Рівні на час. Є варіанти екстремальних часових умов, наприклад, встигнути в магазин за півгодини, до його закриття о 10, при тому що зараз 9:50.

Стримувальні фактори (Disincentives). Ігровий елемент, який використовує штраф (або зміну ситуації) для того, щоб викликати поведінкові зміни.

Приклад: Втрата очок здоров'я, що прискорюють пастки.

Нескінченні ігри (Endless Games). Ігри, які не володіють певною кінцівкою. Більшість додатків і казуальних ігор, з оновлюваним контентом або ігри, де статичний (але позитивний) стан є нагородою сам по собі.

Приклад: Різного роду казуальні джампери та ранери, ММО ігри (Massively multiplayer online games).

Припинення (Extinction). Припинення – це термін, який використовується для опису дії зупинки надання винагороди. Зазвичай це призводить до зростання агресивності у гравців, так як вони відчувають себе обдуреними, не отримавши очікуваної нагороди. Як правило, викликає негативний поведінковий імпульс.

Приклад: Той же, що і в поведінковому контрасті, тільки тепер гравцеві взагалі не дають досвід.

Механіка фіксованих проміжних винагород (Fixed Interval Reward Schedules) забезпечують нагороду після фіксованої кількості часу.

Приклад: В іграх-фермах після 30 хвилинного очікування з'являється урожай.

Механіка фіксованих відносин винагород (Fixed Ratio Reward Schedule) передбачає нагородження після вчинення певної кількості дій.

Приклад: Для виконання квесту – вбити дракона, врятувати принцесу.

Механіка проміжних винагород (Interval Reward Schedules) дозволяє отримувати нагороду після певної кількості часу. Існує два різновиди: змінний і фіксований.

Приклад: Зачекайте N хвилин, зберіть орендну плату.

Лотерея (Lottery). Ігрова механіка, при якій переможець визначається виключно завдяки випадку.

Приклад: Численні види азартних ігор і миттєвих лотерей.

Модифікатори (Modifiers) – пристосування, що впливає на виконання інших дій при використанні. Зазвичай модифікатори заробляються шляхом виконання ряду завдань або ключових обов'язків.

Приклад: Модифікатор X2, що подвоює кількість очок при виконанні певної дії.

Конфіденційність (Privacy). Ідея, яка полягає в тому, що деяка інформація є приватною і не призначена для публічного розповсюдження.

Приклад: Ваги, які щодня публікують вашу вагу в Twitter (реально існуючий і доведений спосіб підтримки дієти). Публічне поширення вашого місцезнаходження кожного разу, коли ви вживаєте якихось дій (це вторгнення в особисте життя і його слід уникати).

Прогресивна динаміка (Progression Dynamic). Динаміка, при якій успіх частково відображається і вимірюється в процесі виконання деталізованих завдань.

Приклад: Різні індикатори, зростання з 1 рівня Паладина до 60, прогресбари.

Механіки співвідношення нагород (Ratio Reward Schedules). Співвідношення нагород надає винагороду по завершенню кількох дій. Існує два різновиди: змінна і фіксована.

Приклад: Убий 10 орків – отримай поліпшення.

Реальний час і час з затримкою (Real-time v. Delayed Mechanics). Інформація, яка надходить в реальному часі, вільна від затримок. Інформація з затримкою надходить тільки після закінчення певного інтервалу часу.

Приклад: Кількість очок, оновлюється в реальному часі, викликає негайну реакцію (задоволення або розчарування). Затримання привносить двозначність, яка може викликати велику активність завдяки відсутності визначеності в розстановці сил.

Посилення (Reinforcer). Винагорода, що отримується в разі виконання очікуваної дії в тричастинній парадигмі режимів нагородження.

Приклад: Отримання рівня після знищення 10 орків.

Розподіл фізичних благ (Rolling Physical Goods). Фізичне благо (володіє реальною вартістю), яке може бути виграно ким завгодно на постійній основі, якщо цей хтось підпадає під певні характеристики.

Приклад: Бути першим у будь-чому.

Статус (Status). Ранг або рівень гравця. Гравців часто стимулює бажання досягти більш високого рівня або статусу.

Приклад: Білий паладин 90 рівня у WoW.

Механіка змінних проміжних винагород (Variable Interval Reward Schedules). Механіка змінних проміжних винагород надає винагороду по закінченню приблизно рівних відрізків часу.

Приклад: Зачекайте, поки після 30 хвилин як закінчеться гра з'являється зброя. Часті перевірки не прискорюють появу. Зазвичай гравці погано усвідомлюють це.

Механіка змінних відносин винагород (Variable Ratio Reward Schedule). Механіка змінних відносин винагород надає винагороди після приблизного, але точно невідомої кількості дій.

Приклад: Необхідно знищити приблизно 20 кораблів і отримаєте рівень. Відвідайте кілька локацій (приблизно 5) і отримаєте значок.

Весело один раз, Весело завжди (Fun Once, Fun Always). Концепція того, що дія приносить задоволення, скільки б разів вона не повторювалася. Зазвичай це має відношення до найпростіших прикладів активності. Також, дуже часто існує обмежувальний поріг для загального рівня задоволення, принесеного однією дією.

Приклад: Різного роду завдання на дослідження місцевості.

Віртуальні предмети (Virtual Items). Цифрові призи, нагороди, об'єкти, знайдені під час гри. Часто їх можна продати або подарувати.

Приклад: Екіпірування в RPG, подарунки в Facebook, значки тощо.

12.1.2 Порядок заповнення баг-репортів по іграм [31]***:***

– при тестуванні ігор на ПК в розділі «Platform» вказують «PC». При тестуванні ігор на мобільних пристроях в розділі «Platform» вказують «Mobile»;

– під час оформлення баг-репортів по локалізації, в темі (Summary) необхідно вказувати: пристрій, на якому було здійснено перевірку, мову локалізації (абревіатура), місце розташування (рівень, розділ, локація тощо), короткий опис бага. Наприклад: «iPad 2. ZH. Квест 356. Відображається текст «unknown 14740» на кнопці діалогу після дебрифінгу квесту # 356» або «Win. Міні-гра «Археологічна експедиція». Текст [id = 7418] відрізняється від локкіта у вікні туторіала»;

– при тестуванні ігор в кроках відтворення (Steps to Reproduce) необхідно вказувати платформу, на якій проводилася перевірка з версією ОС і номер білда, на якому виконувалося тестування,

наприклад: iPad Air iOS 12.1.1 Build # 90 або Win10 x64 Pro Patch # 961 ST3;

– під час оформлення баг-репортів по іграх на ПК, в темі (Summary) необхідно вказувати: платформу, на якій проводилася перевірка, місце розташування (рівень, розділ, локація і т.д.), короткий опис бага. Наприклад: Win. Ангар. Не відображаються гусениці у моделі танка після вибору танка «IC-7»;

– під час оформлення баг-репортів по іграх на мобільному пристрої в темі (Summary) необхідно вказувати: назву та модель мобільного пристрою, на якому проводилася перевірка, місце розташування (рівень, розділ, локація), короткий опис бага. Наприклад: iPad Pro 11. Lvl 1. Текст кнопок відображається квадратами у грі після поразки;

– назву гри необхідно вказати в розділі «Additional Information».

Приклад баг-репорту тестування гри наведено на рисунку 12.1.

номер ID	Проект	Категорія	Рівень доступу	Дата реєстрації	Дата зміни
6	GTA 5	функціонал	загальний	28.10.2024	
Тестувальник	Попов Кирило КНТ-121	**Пріорітет**	середній	**Серйозність**	незначна
Відтворюваність	постійно	**Статус**	надіслано	**Резолюція**	відкритий
Платформа	PC	**Операційна система**	Windows	**Версія ОС**	10 Pro
Тема	6: Win. Неправильна логіка керування автомобілем ботами, якщо шлях є заблокованим				
Опис	Якщо розташувати автобус на мосту поруч з ділянкою без огорожі, заблокувавши рух транспорта, то боти на автомобілях будуть спрямовувати машини у прірву на інший міст, що наражає їх на небезпеку				
Кроки для відстворення	Платформа: Windows 10 Pro 22H2 Збірка: 19045.5011 1. Авторизуватися у застосунку Rockstar Studio 2. Завантажити гру GTA 5 3. Перейти у головному меню гри до сюжетної версії GTA 5 4. Обрати будь-якого з наявних трьох головних персонажів сюжетної версії 5. Помітити та вкрасти будь-який автобус або довгий автомобіль та розташувати його вздовж ділянки руху на мості у центрі міста поруч з прірвою так, щоб боти не могли проїхати вперед 6. Звернути увагу на подальші дії ботів на автомобілях				
Фактичний результат	Створені комп'ютером персонажі-боти в грі, не маючи можливості проїхати далі, їдуть вперед у прірву, що призводить до потенційного руйнування автомобілів або їхнім зниченням разом з водієм та пасажирами				
Очікуваний результат	Створені комп'ютером персонажі-боти в грі, не маючи можливості проїхати далі, їдуть назад, шукаючи альтернативні способи та маршрути доїхати до необхідного місця, або чекають, доки машина, що їм заважає проїхати вперед, не покине відповідну ділянку на мосту				
Додаткова інформація	Гра: GTA 5 Online Посилання на Epic Games Store: https://store.epicgames.com/en-US/p/grand-theft-auto-v				
Прикріплені файли					

Рисунок 12.1 – Баг-репорт на тему «Неможливо оновити пароль на сторінці редагування профілю після заповнення необхідних для зміни паролю полів»

12.2 Завдання до роботи

12.2.1 Встановити сервіс цифрового розповсюдження ігор Steam https://store.steampowered.com/about (створити акаунт у встановленому клієнті Steam та встановити безкоштовну гру у ранньому доступі) або завантажити та встановити будь-яку безкоштовну гру з Інтернету.

12.2.2 Зробити 3 баг-репорти для функціональних багів та 3 баг-репорти для багів локалізації або юзабілі та додати до звіту.

12.3 Зміст звіту

12.3.1 Тема та мета роботи.
12.3.2 Завдання до роботи.
12.3.3 Короткі теоретичні відомості.
12.3.4 Хід виконання завдання до лабораторної роботи.
12.3.5 Висновки, що відображують результати виконання роботи та їх критичний аналіз. Також додати відповіді на контрольні запитання (3 запитання із пункту 12.4 за вибором студента).

12.4 Контрольні питання

12.4.1 Вказати та охарактеризувати існуючі платформи для тестування ігор.
12.4.2 Що входить до фокус-тестування ігор? Навести приклади знайдених дефектів та обґрунтувати відповідь.
12.4.3 Вказати основну мету проведення тестування локалізації. Навести приклади знайдених дефектів та обґрунтувати відповідь.
12.4.4 Які дефекти в грі вважаються критичними? Навести приклади знайдених дефектів та обґрунтувати відповідь.
12.4.5 Вказати особливості тестування казуальних ігор. Навести приклади та обґрунтувати відповідь.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1 Крепич С.Я. Якість програмного забезпечення та тестування: базовий курс : навч. посіб. / С.Я. Крепич, І.Я. Співак. – Тернопіль: ФОП Паляниця В.А., 2020. – 479с. URL: http://dspace.wunu.edu.ua/handle/316497/39773 (дата звернення 30.08.2025).

2 Основні атрибути баг-репорта. URL: https://training.qatestlab.com/blog/course-materials/attributes-bug-report/ (дата звернення 30.08.2025).

3 Чому грамотне оформлення баг-репортів таке важливе. URL: https://training.qatestlab.com/blog/course-materials/proper-bug-report/ (дата звернення 30.08.2025).

4 10 найчастіших помилок при оформленні баг-репортів. URL: https://training.qatestlab.com/blog/helpful-materials/10-common-mistakes-in-bug-reports/ (дата звернення 30.08.2025).

5 Types Of Software Testing: Different Testing Types With Details. URL: https://www.softwaretestinghelp.com/types-of-software-testing/ (дата звернення 30.08.2025).

6 Огляд видів тестування. URL: https://training.qatestlab.com/blog/technical-articles/review-the-types-of-testing/ (дата звернення 30.08.2025).

7 Checklist – чудовий помічник для QA. URL: https://qagroup.com.ua/publications/checklist-chudovyj-pomichnyk-dlia-qa/ (дата звернення 30.08.2025).

8 Що таке чеклісти та як з ними працювати. URL: https://training.qatestlab.com/blog/technical-articles/work-with-checklist/ (дата звернення 30.08.2025).

9 Software Testing Help. «Web Testing: Complete guide on testing web applications». URL: http://www.softwaretestinghelp.com/web-application-testing/ (дата звернення 30.08.2025).

10 Тестування верстки. Основні вимоги. URL: https://training.qatestlab.com/blog/technical-articles/testing-layout-basic-requirements/ (дата звернення 30.08.2025).

11 Кросбраузерне тестування: навіщо та кому потрібно його проводити. URL: https://training.qatestlab.com/blog/technical-articles/cross-browser-testing/ (дата звернення 30.08.2025).

12 Інструменти для тестування кросбраузерності верстки. URL: https://training.qatestlab.com/blog/technical-articles/tools-for-testing-cross-browser-layout/ (дата звернення 30.08.2025).

13 What is Usability Testing? Software UX. URL: https://www.guru99.com/usability-testing-tutorial.html (дата звернення 30.08.2025).

14 Дружній інтерфейс: як зробити дизайн сайту зручним. URL: https://pbb.lviv.ua/statti-i-novyny/statti-shchodo-stvorennia-saitu/jak-zrobyty-dyzayn-saytu-zruchnym/ (дата звернення 30.08.2025).

15 Що таке функціональний баг та як його знайти. URL: https://training.qatestlab.com/blog/technical-articles/what-is-a-functional-bug-and-how-to-find-it/ (дата звернення 30.08.2025).

16 Як тестувати вебсайт: основні етапи і поради. URL: https://brainlab.com.ua/uk/blog-uk/yak-testuvati-veb-sayt-osnovn-etapi-poradi (дата звернення 30.08.2025).

17 Валідація. Тестування валідації. URL: https://training.qatestlab.com/blog/technical-articles/validation-testing/ (дата звернення 30.08.2025).

18 Chrome DevTools: налаштування, можливості та способи перевірки коду. URL: https://dou.ua/lenta/articles/chrome-dev-tools-guide/ (дата звернення 30.08.2025).

19 Faster HTML/CSS Workflow with Chrome Developer Tools. URL: https://webdesign.tutsplus.com/uk/articles/faster-htmlcss-workflow-with-chrome-developer-tools--webdesign-8314 (дата звернення 30.08.2025).

20 Що таке цілісність тест-кейсу або як правильно описати тест-кейс, щоб перевірити функціонал? URL: https://training.qatestlab.com/blog/course-materials/test-case-description/ (дата звернення 30.08.2025).

21 How to Write Test Cases in Software Testing with Examples. URL: https://www.guru99.com/test-case.html (дата звернення 30.08.2025).

22 Види тестування, пов’язані зі змінами. Кросбраузерність. URL: https://qalearning.com.ua/theory/lectures/material/regression-testing-crossbrowser/ (дата звернення 30.08.2025).

23 What is regression testing? URL: https://www.guru99.com/regression-testing.html (дата звернення 30.08.2025).

24 Test plan. URL: https://en.wikipedia.org/wiki/Test_plan (дата звернення 30.08.2025).

25 Test Plan Templates. URL: https://strongqa.com/qa-portal/testing-docs-templates/test-plan (дата звернення 30.08.2025).

26 Робота з системою TestRail. URL: https://training.qatestlab.com/blog/technical-articles/work-with-testrail-system/ (дата звернення 30.08.2025).

27 Навчальний посібник із підсумкових звітів про тестування: навчання за допомогою прикладів і шаблонів. URL: https://www.guru99.com/uk/how-test-reports-predict-the-success-of-your-testing-project.html (дата звернення 30.08.2025).

28 Особливості тестування додатків на мобільних пристроях. URL: https://training.qatestlab.com/blog/technical-articles/testing-mobile-devices/ (дата звернення 30.08.2025).

29 Інструменти для тестування на мобільних пристроях. URL: https://training.qatestlab.com/blog/technical-articles/tools-for-testing-on-mobile-devices-part-1/ (дата звернення 30.08.2025).

30 Термінологія геймера. URL: https://training.qatestlab.com/blog/technical-articles/gamer-terminology/ (дата звернення 30.08.2025).

31 Тестування продуктивності в іграх, поняття FPS. URL: https://training.qatestlab.com/blog/technical-articles/testing-performance-games-fps/ (дата звернення 30.08.2025).

ДОДАТОК А
ПРИКЛАД ТЕСТ-ПЛАНУ

А.1 Вступ

А.1.1 Мета

Метою складання тест-плану є опис процесу тестування сайту Prestashop (повна адреса http://prestashop.qatestlab.com.ua/en/). Документ дозволяє отримати уявлення про планові роботи з тестування проєкту.

А.1.2 Вихідні дані

Prestashop – сайт, що дозволяє переглядати та купувати товари магазину, створювати списки бажаних товарів, переглядати інформацію про магазини.

А.1.3 Мета тестування

Метою тестування сайту Prestashop є перевірка роботи всіх його функціональних можливостей на різних версіях браузерів з типовими сценаріями його використання. Частина часу (приблизно 20%) буде використана для тестування нетипових або тих, що потенційно викликають помилки в роботі, сценаріїв використання.

Результатом процесу тестування будуть наступні матеріали:

− висновок команди тестування відносно загального стану, що дає розробникам і менеджерам цього продукту уявлення щодо коректності роботи сайту в різних браузерах;

− звіт про результати тестування поточного покриття, типові сценарії використання, відображення в різних браузерах;

− задокументовані баги в баг-трекері замовника.

Тестування буде проводитися вручну, методом «неформального» тестування (ad-hoc testing) з позиції кінцевого користувача сайту.

А.2 Умови для тестування

Вебсайт повинен задовольняти потреби користувача в активностях, пов'язаних з переглядом та замовленням товарів, створенням списків бажаних товарів, переглядом інформації про магазини, створенням та входом в особистий кабінет.

А.3 Стратегія процесу тестування

Наведений нижче план тестування є формальним, так як для побудови розгорнутого плану необхідно розуміння поточного стану проєкту. В результаті першого прогону функціональних тестів в тест-план будуть занесені зміни і покращення. Перший прогон функціональних тестів надасть точне уявлення про рівень стабільності системи і буде чітко визначено набір тестів, які будуть виконані в кожній конфігурації.

Такий підхід дозволить отримати розгорнутий звіт по тестованому продукту і зосередить максимальну увагу на вузьких місцях.

Замовнику будуть надаватись щоденні звіти про хід тестування, знайдені дефекти, пропозиції щодо покращення роботи продукту і його дизайну. Всі виявлені дефекти будуть занесені в баг-трекер замовника у вигляді окремих тікетів для подальшого виправлення.

В процесі тестування сайту Prestashop буде використано ad-hoc тестування через відсутність специфікації, а також через обмеженість ресурсів на формалізацію тестів.

Планується п'ять етапів проведення процесу тестування:

− перший етап полягає в аналізі ТЗ, складанні тест-плану, а також часткового прогону функціональних тестів;

− другий етап буде пов'язаний з детальним прогоном функціональних тестів та з виявами і описом дефектів;

− на третьому етапі буде проведено тестування кросбраузерності з описом знайдених дефектів;

− четвертим етапом є перевірка виправлених розробниками багів і проведення регресивного тестування;

− п'ятий етап полягає в тестуванні дизайну продукту з описом знайдених дефектів.

Таким чином, досягається максимальна деталізація глибини тестування, що, в свою чергу, дозволяє більш точно визначити витрачені ресурси, а також дозволяє розробникам проєкту виправляти дефекти на ранніх етапах.

ОС, затверджена до перевірки, – Windows 11.

Браузери, затверджені до перевірки:

- Brave 1.58.137;
- Google Chrome 117.0.5938.150;
- Microsoft Edge 117.0.2045.60.

Тестування безпеки і стрес-тестування не проводиться через нестачу часу на тестування.

А.3.1 Типи тестування

А.3.1.1 Функціональне тестування

Мета: Виявлення функціональних помилок, невідповідностей ТЗ і очікуванням користувача шляхом реалізації стандартних, а також нетривіальних тестових сценаріїв.

Опис процесу.

1 Особистий кабінет:

– реєстрація;
– авторизація;
– інформація про покупця;
– картки та адреси доставки;
– список бажаних товарів;
– історія замовлень.

2 Кошик:

– видалення товару;
– змінення кількості товару;
– перехід до оформлення замовлення.

3 Оформлення замовлення.

4 Пошук.

5 Зворотний зв’язок.

6 Підписка на новини.

7 Сторінки соц. Мереж.

8 Банери:

– відображення банерів;
– перехід до відповідних сторінок.

9 Перелік товарів:
– фільтрування;
– сортування.
10 Сторінка товару:
– додавання до кошику;
– додавання до списку бажаних товарів;
– поділитися в соц. мережах;
– придбати товар;
– перегляд фото та інформації про товар.

А.3.1.2 Тестування кросбраузерності

Мета: Перевірити коректну роботу і дизайн сайту в різноманітних браузерах.

Браузери до перевірки:
- Brave 1.58.137;
- Google Chrome 117.0.5938.150;
- Microsoft Edge 117.0.2045.60.

А.3.1.3 Стрес-тестування

Мета: Виявити слабкі місця в роботі сайту шляхом використання об’ємних даних, довгих посилань, невалідних даних в пошуковій системі сайту.

Опис процесу: Цей вид тестування і тест-кейси до нього обговорюються і описуються окремо разом з розробником сайту.

А.3.1.4 Регресивне тестування і перевірка вирішених дефектів

Мета: Перевірка змін зроблених на сайті для того, щоб впевнитися, що нова версія не має помилок в уже протестованих частинах сайту.

В ході регресивного тестування будуть проведені такі види тестувань:
- верифікаційні тести;
- тестування версій;
- тестування суміжного функціоналу.

А.3.1.5 Тестування дизайну

Мета: Перевірка відповідності дизайну продукту макетам специфікації.

Опис процесу:

- реєстраційні/авторизаційні форми;
- контакти та розташування магазинів;
- корзина;
- сторінки сайту.

А.4 План робіт

Таблиця А.4.1 – План робіт

Задача	Об'єм роботи	Дата початку	Дата закінчення
Складання тест-плану і чекліста	12 год	14.11.2024	12.12.2024
Коригування тест-плану і чекліста	1 год		
Виконання тестування другого етапу	Brave 1.58.137 – 2 год G Chrome 117.0.5938.150 – 2 год M Edge 117.0.2045.60 – 2 год	22.11.2024	23.11.2024
Виконання тестування третього етапу	Brave 1.58.137 – 1,5 год G Chrome 117.0.5938.150 - 1,5 год M Edge 117.0.2045.60 – 1,5 год	23.11.2024	24.11.2024
Аналіз тестування	4 год		
Підведення підсумків	3 год		

А.5 Кінцеві результати

А.5.1 Висновок

Кінцевим підсумком проведення тестування повинен стати оформлений кінцевий результат процесу тестування з описаними дефектами, а також рекомендаціями про покращення продукту з точки зору кінцевого користувача.

ДОДАТОК Б
ПРИКЛАД ЗВІТУ З ТЕСТУВАННЯ САЙТУ

Б.1 Вступ

Б.1.1 Умови тестування

Було вивчено роботу інтернет-магазину «prestashop.qatestlab» на комп'ютері під управлінням операційної системи Windows 10, версії Pro.

До головних програмних умов тестування відносимо:

– безперервне інтернет-з'єднання зі швидкістю не менше 10 Мбіт/с;

– браузери Google Chrome 95, Internet Explorer 11, Safari 15;

– включений у браузерах JavaScript.

До мінімальних програмних умов тестування належать:

– процесор: 1 гігагерц (ГГц) або швидший процесор;

– оперативна пам'ять: 1 гігабайт (ГБ) для 32-розрядних або 2 ГБ для 64-розрядних;

– місце на жорсткому диску: 32 ГБ для 64-бітної і 32-бітної ОС;

– відеокарта: з підтримкою DirectX 10;

– розширення екрану: 576 x 1024, мінімальний розмір діагоналі для основного дисплея 7 дюймів або більше.

При тестуванні у зазначених вище умовах сайт показав свою працездатність. Відмінності при тестуванні в декількох браузерах були незначні. Завантаження сторінок проходило швидко в порівнянні з аналогічними сайтами.

Тестування та складання звіту проводилось в складі одного співробітника – Петренка Сергія.

Б.1.2 Цілі проєкту prestashop.qatestlab

Prestashop.qatestlab – сайт, типу інтернет-магазин. Головні цілі такого проєкту:

– автоматизація процесу купівлі-продажу одягу;

– підвищення ресурсної ефективності купівлі-продажу одягу – часової, трудової, матеріальної;

– як наслідок підвищення доходу замовника.

Об’єктами процесу купівлі-продажу є жіночий верхній одяг, який розбивається на наступні 3 категорії: сукні, плаття та футболки.

Суб’єктами процесу є жінки, переважного середнього віку.

Б.1.3 Завдання проєкту prestashop.qatestlab

Вебсайт має найбільш ефективно налагодити роботу «клієнту» та «продавця» в системі. Для цього передбачено:

– розташування актуального товару;

– пропонування знижок, акцій та додаткових пропозицій;

– ведення особистого профілю клієнту, з його даними та вподобаннями;

– ведення кошику покупця, звідки проводиться оформлення замовлення;

– надання усієї інформації про магазин, інтеграція функцій інших сервісів (соціальних мереж, гугл-карт).

Система побудована за базовою архітектурою клієнт-сервер та представляє собою інтернет-магазин.

Б.1.4 Найчастіші типи активності відвідувачів сайту prestashop.qatestlab

Найчастіші типи активності відвідувачів сайту наступні:

– реєстрація нового користувача;
– авторизація користувача;
– відновлення паролю;
– зміна локалізації (мови сайту, валюти);
– вказання адреси для доставлення товару;
– перегляд історії покупок;
– зміна особистої інформації профілю користувача;
– робота зі списком побажань;
– «зворотній зв’язок»;
– пошук товару на сайті;

– фільтрація товарів за категорією (за кольором, за розміром, за кількістю товарів, за стилем, за тканню, за покроєм, за виробником, за кондицією використане/невикористане);
– вибір інтервалу прийнятної ціни;
– перегляд та дій з товаром;
– «поділитися» товаром у соціальних мережах;
– коментування;
– робота з «кошиком» покупок;
– оформлення покупки (товар, кількість, знижки, адреса, вказання реквізитів);
– пошук магазинів за радіусом на карті.

Б.1.5 Коментарі

Аудиторія сайту складається з постійної клієнтів та нерегулярних відвідувачів. Постійна аудиторія забезпечує стабільну відвідуваність і популярність, що важливо для власника компанії. Отже, необхідно збільшувати число постійних відвідувачів. Для формування постійної аудиторії ключове значення мають два фактори: зручність роботи і довіра відвідувачів. Останнім часом все більшого значення набуває також можливість спілкування, створення спільноти активних відвідувачів.

Б.2 Тестування сайту prestashop.qatestlab

Тестування сайту було проведене у ручному режимі, за визначеними чеклістами тестування.

Діаграму тестування, згідно до статусу пунктів тестування, зображено на рисунку Б.2.1.

Рисунок Б.2.1 – Діаграма тестування, згідно до статусів пунктів тестування

Згідно до побудованої діаграми, можна зробити висновок, що третина тестів завершилася не успішно, тобто можна констатувати знайдення багів. На їх основі зробимо пошук, обґрунтування та шляхи виправлення недоліків сайту.

Тепер, серед знайдених багів, зробимо огляд їх типів. Діаграму тестування, згідно до типу помилки, зображено на рисунку Б.2.2.

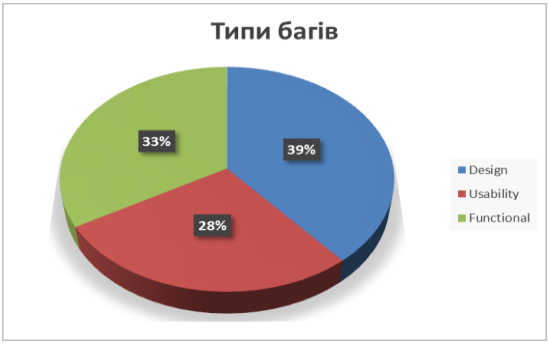


Рисунок Б.2.2 – Діаграма розподілу багів за типами

Згідно до побудованої діаграми, можна зробити висновок, що кількість знайдених багів за типами (дизайн, зручність використання, функціональні помилки) приблизно рівна, а тому екстенсивне вирішення проблем, пов’язаних з цими багами, буде теж приблизно рівним.

Проаналізуємо серйозність помилок, застосовуючи діаграму, зображену на рисунку Б.2.3.

Рисунок Б.2.3 – Діаграма розподілу багів за серйозністю

Згідно до побудованої діаграми, можна зробити висновок, що кількість проблем з підвищеною серйозністю доволі велика у загальному числі багів – 67%. Однак «блокуючих» багів, що вимагають негайного вирішення, не так багато. На них необхідно сконцентруватися у першу чергу.

За рівнем проблематичності найменше виділяється Google Chrome 95. При вирішенні проблеми кросбраузерності та підтримки Internet Explorer, компанії необхідно проаналізувати ринок користувачів, та частоту використання цього браузеру. Можливо від підтримки цього браузеру варто відмовитися у зв'язку зі строками та доцільності розробки. Однак вирішення може і не витрати багато часу, у той час коли втрати клієнтів знизять дохід.

Тепер наведемо графік проходження тестів, та, як наслідок, знаходження багів у часі.

Б.2.1 Тестування зручності використання

Виходячи зі знайдених багів, визначимо основні недоліки:

– зворотній зв'язок;
– правильні кольорові акценти;
– актуальність інформації для користувача;
– наявність спливаючих підказок;
– щільність елементів керування.

Б.2.1.1 Зворотній зв’язок

Зворотній зв’язок в контексті зручності використання – це система автоматичного реагування на дії користувача, завдяки чому користувач впевнюється у правильності виконуваних дій.

Користувач має бачити результати своєї взаємодії зі сайтом. У разі безуспішності такого результату – користувач отримує пояснюючі повідомлення графічного, рідше звукового виду.

Група проблем, пов’язаних зі зворотнім зв’язком, містить наступні пункти:

– відсутність графічного виділення на сторінках першочергових можливостей системи. Через це користувач може втратити свої головні можливості роботи на сайті;

– відсутні пояснення алгоритму дій для користувача, для складання та оформлення його першого замовлення. Це призводить до складнощів, у першу чергу, для нових та недосвідчених користувачів, які роблять оформлення покупки у перший для них раз;

– деякі неочевидні елементи сторінки інтуїтивно незрозумілі та не мають пояснень у будь-якій формі. Другорядні елементи та їх сенс не тільки є незрозумілими, а й можуть відволікати користувача від суті;

– написання та успішне публікування коментарів ніяк не подається користувачеві, як зроблена та закінчена дія; Таким чином користувач відчуватиме неповноцінність та не важливість його власної точки зору на товар;

– відсутні звукові підтвердження та повідомлення користувачу. Таким чином ігнорується 10% потоку сприйняття користувача інформації.

Рекомендації:

Згідно до проблем «зворотного зв’язку» пропонується:

– графічне виділення кольором, який є прийнятний за даного дизайну, найважливіших об’єктів системи: 3 категорії одежі – сукні, плаття та футболки; кошик та товари у списку товарів. Беручи до уваги, що стандартними кольорами для оформлення сайту є білий-сірий-чорний, пропонується взяти будь-який стандартний колір та застосовувати його та його похідні повсюдно на сайті, наприклад червоний;

– створити пояснення алгоритму дій для користувача, для складання та оформлення його першого замовлення. Для цього необхідно створити або модальні, тимчасові вікна, поверх сторінки, або тимчасові повідомлення з продуманою структурою верстки сторінки на початку її створення. Також можливий варіант створення «гайду» для нових користувачів, який не є обов'язковий, і від якого користувач може відмовитися. Для реалізації цієї можливості необхідно відслідковувати нових користувачів та їх перше замовлення;

– банери на головній сторінці, банери на сторінках з категоріями одягу мають містити відображення адреси посилання при наведені, до натискання на них відповідно. Поля для заповнення при авторизація, створення коментаря та повідомлення адміністратору сайту мають бути поясненні при наведені, звідки користувач дізнається їх формат;

– після написання коментарю користувач отримує повідомлення «Дякуємо за вашу думку», при оформлені покупки – повідомлення «Оригінальний вибір», а при виході з системи – «Дякуємо за проведений разом з вами час». Таким чином проходить і спілкування з користувачем на пряму;

– звукові сигнали при невдалих діях користувача не рекомендується – це може підвищити кількість отриманого їх стресу при взаємодії з системою. Однак при визначних діях, наприклад при підтвердженні покупки, користувач має отримувати незначний. прийнятний звуковий сигнал, або невелику мелодію.

Б.2.1.2 Кольорові акценти

При розміщенні чогось на сторінці, що є головним елементом предметної області чи її першопричиною, кольорові акценти мають бути направлені в першу чергу на нього.

Тут основною проблемою є елемент сторінки – список товарів. Через те що його головна задача, це надавати інформацію про товар, одиниця товару має містити – назву, вартість та зображення товару. Другорядні елементи цікавлять користувача або при докладному ознайомлені, або при фільтрації чи пошуку. Перевантаженість

другорядної інформацією у списку товарів добре можна побачити на рисунку Б.2.4.

Рекомендації:

– залишити видимою лише головну інформацію про товар – назву, вартість та фотографію, назву виділити, змінивши шрифт та підвищуючи товщину;

– виключити з відображення наступну необов'язкову інформацію: чи є товар у наявності, наявні кольори товару. Інформацію про знижку залишити, виділивши червоним кольором. При наведені прибрати кнопку швидкого перегляду товару, кнопку «більше», а другорядну інформацію, описану вище, навпаки вивести;

– відображена інформація, будь-то кнопки чи плашки, має бути подана у чорно-білому форматі, окрім інформації про знижку.

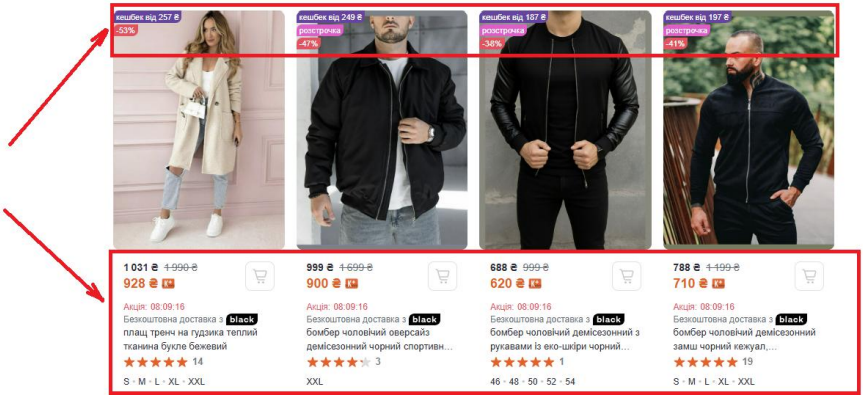


Рисунок Б.2.4 – Перевантаженість другорядною інформацією у списку товарів

Б.2.1.3 Актуальність інформації для користувача

Програмним продуктом користуються два типи користувачів. Функціональні можливості як зареєстрованих (постійних), так і незареєстрованих (нових) відрізняються, однак на сайті представлені тотальним форматом, без розділення. Перевантаження нових користувачів інформацією може злякати їх, та відвернути від

продукту, а для постійних користувачів відсутність інформації напроти – спонукає їх покинути платформу та знизити дохід. Така низька ефективність сайту, у плані помилкового відображення інформації, з точки зору її актуальності, веде до зниження конверсії сайту.

Рекомендації:

Усі недоступні незареєстрованому користувачу функції скрити від його зору. Слід прибрати:

– кошик та можливість додати товар у кошик, через те, що неавторизований клієнт не може оформлювати покупки в магазині;

– можливість додати товар до списку бажаних товарів, тому що список бажаних товарів відсутній;

– можливість написання коментарю, через те, що коментар має містити інформації про автора та не може бути анонімним;

– з футеру сайту прибрати усю інформацію про акаунт «My account» (My orders, My credit slips, My addresses, My persona info).

ДОДАТОК В
ПРИКЛАД ОФОРМЛЕННЯ ТИТУЛЬНОГО АРКУШУ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

кафедра «Програмні засоби»

ЗВІТ

з лабораторної роботи № 1

з дисципліни «Якість програмного забезпечення та тестування»

на тему: «**СТВОРЕННЯ БАГ-РЕПОРТІВ**»

Виконав:

студент групи КНТ-113 А.Б. Іваненко

Прийняв:

ст. викладач В.Г. Костюк

2025