Лабораторна робота 1
Інструментальний засіб *Jira Software*.
Тестування програмного забезпечення.

Мета роботи: Вивчити принцип роботи інструмент управління робочим процесом для команд розробників ПЗ *Jira Software*. Провести тестування вебсайту.

Теоретичні відомості

Питання про роль та місце психології у програмуванні стали виникати багато років тому, при становленні програмування як професійної діяльності. Наскільки справедливо розглядати розробку програмних продуктів лише з погляду обчислювальної математики?

В даний час переважає тверде переконання, що програмування – це рід людської діяльності, де люди створюють програми виключно для людей. Комп'ютер використовується лише як інструмент. Звідси випливає, що розробка програмного забезпечення, зокрема мультимедійних продуктів, має розглядатися з позицій людино-машинної системи.

Стандарт ISO 9241-210 ставить у центр процесу розробки користувальницькі вимоги. Однак концентрація виключно на користувачах може призвести до того, що продукт виявиться невигідним для бізнесу або важким з технічної точки зору. Тому для створення успішного продукту треба враховувати також бізнес-вимоги та технічну складову. Користувачів, бізнес та технології поєднує UX-технології, а точніше UX-тестування.

UX-тестування – комплекс заходів, спрямованих на виявлення будь-яких проблемних місць на вашому ресурсі: чи достатньо зрозумілий, логічний, зручний, чи правильно працюють всі його технічні елементи.

Результатом грамотного UX-тестування є перелік рекомендацій, що і яким чином потрібно змінити, щоб підвищити кількість конверсій та перетворити відвідувачів сайту на його постійних та відданих користувачів.

Тестування виявляє великі та дрібні проблеми інтерфейсу, кожна з яких відсіває ваших потенційних покупців.

Також UX-тестування показує, наскільки зрозумілий покупцям ваш інтерфейс, чи використовують вони його так, як ви задумали, чи зовсім іншим чином. Отже, показують, яким чином потрібно змінити *user flow* на сайті, щоб користувачам було зручно.

UX-тестування потрібно, якщо ви хочете перевірити існуючий інтерфейс на зручність користувальницьких сценаріїв, відзначити всі "проблемні" місця та покращити їх.

Тестувальнику доводиться працювати з величезною кількістю інформації, вибирати з безлічі варіантів вирішення завдань та винаходити нові. У процесі цієї діяльності об'єктивно неможливо пам'ятати усі думки, а тому продумування та розробку тест-кейсів рекомендується виконувати з використанням «test case».

Test Case – це документ, що описує сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації функції або її частини.

Навіщо потрібний test case?

1. Щоб описати детально, як будемо тестувати функцію.

2. Для будь-якого спеціаліста, який не знайомий із вимогами.

3. Для звітності.

Під час проведення тестування заповнюють так званий *Bug report* (звіт про знаходження помилки).

Bug report – докладна покрокова інструкція для відтворення помилки.

Атрибути *Bug report*:

Id – номер звіту в системі.

Summary – короткий опис проблеми, що явно вказує на причину помилки. Описується за принципом «Що? Де? Коли?».

Preconditions – умови, які мають бути дотримані для відтворення помилки.

Steps to reproduce – кроки відтворення помилки.

Actual result – фактичній результат, отриманий після кроків відтворення.

Expected result – очікуваний результат, прописаний у специфікації (вимогах).

Attachments – вкладення, що допомагають відтворити *bag* та прояснити ситуацію.

Additional info – додаткова інформація.

Author – упорядник *bag-report.*

Assigned to / Asignee – розробник, який має виправити *bag*.

Status – поточний стан *bag*.

Severity – критичність, показник, що відображає вплив дефекту на працездатність програми (зазвичай виставляє тестувальник).

Priority – пріоритет, показник, що визначає порядок робіт (більш інструмент для менеджера).

Environment – оточення, на якому було знайдено *bag* (ОС, ім'я та версія браузера).

Version – версія продукту, в якій було знайдено *bag.*

Можливі значення *Priority*.

High – найвищий пріоритет. Ця помилка має бути виправлена у максимально короткий термін. Наприклад: не працює кнопка «*Login*».

Medium – помилка, яку необхідно виправити в короткий термін, але вона не критична для даного продукту. Наприклад: не працює кнопка переходу на сторінку магазину в *Instagram.*

Low – помилка не вимагає швидкого вирішення, але обов'язково повинна бути виправлена. Наприклад: друкарська помилка в тексті на сторінці з описом одного з товарів.

Можливі значення *Severity.*

Blocker – помилка, яка призвела до повного блокування програм. Цей атрибут означає, що подальша робота неможлива, потрібно вжити термінових заходів для усунення отриманого результату. Наприклад: користувач не може оформити замовлення в інтернет-магазині, оскільки не відкривається кошик.

Critical – критична помилка, яка також потребує термінового вирішення. Це можуть бути збої в системі безпеки або помилка, через яку впав сервер. Система буде працювати з цією помилкою, але ключові функції можуть бути недоступні. Наприклад: користувач не може ввійти у свій аккаунт.

Major – значна помилка. Цей атрибут означає, що можливість працювати з цією функцією є, але при цьому певна частина логіки працює некоректно. Наприклад: не приходить сповіщення про нове повідомлення в месенджері.

Minor – бізнес-логіка працює коректна, але знайдена помилка інтерфейсу користувача. Наприклад: колір кнопки не відповідає основним кольорам сайту.

Trivial – помилка, що має найменшу серйозність, вона на зачіпає бізнес-логіку, користувач може навіть не потрапити на цю помилку. Наприклад: помилка в тексті; не робоче посилання в тексті статті.

При тестуванні програмного забезпечення використовуються інструментальні засоби. Інструментальний засіб управління тестуванням бере на себе всі рутинні технічні операції, які необхідно виконувати в процесі реалізації життєвого циклу тестування. Величезною перевагою також є здатність таких інструментальних засобів відстежувати взаємозв'язки між різними документами та іншими артефактами, взаємозв'язки між артефактами та процесами тощо, підпорядковуючи ці дії системі розмежування прав доступу та гарантуючи збереження та коректність інформації.

В даний час використовується значна кількість інструментальних засобів тестування. У цьому курсі ми будемо використовувати систему *Jira*.

Jira Software – інструмент управління робочим процесом для команд розробників ПЗ, які хочуть систематизувати і відстежувати свою роботу. Неймовірна гнучкість дозволяє налаштувати *Jira* відповідно до унікального робочого процесу команди.

1. Порядок виконання лабораторної роботи

1. Встановити он-лайн версію Jira Software за посиланням: https://www.atlassian.com/ru/software/jira/free

2. Створити новий проєкт.

3. Підготувати шаблон з тестування за зразком:

Preconditions:

Посилання https://d.goit.global/ua/qa/ відкрите в браузері

Steps to reproduce:

Натиснути на іконку “Фейсбук” (в футере)

Проінспектувати посилання

Actual result:

При натисканні на іконку «Фейсбук» (в футере) відкривається посилання на «Телеграм»

Expected result:

При натисканні на іконку «Фейсбук» (в футере) відкривається посилання на сторінку у «Фейсбук»

Environment:

Windows, Firefox 121.0

4. Протестувати сайт https://d.goit.global/ua/qa/ на помилки.
5. Знайти не менше 5 помилок.

2. Зміст звіту

Зробити скрін екрану. Зразок звіту: