## Тестова документація

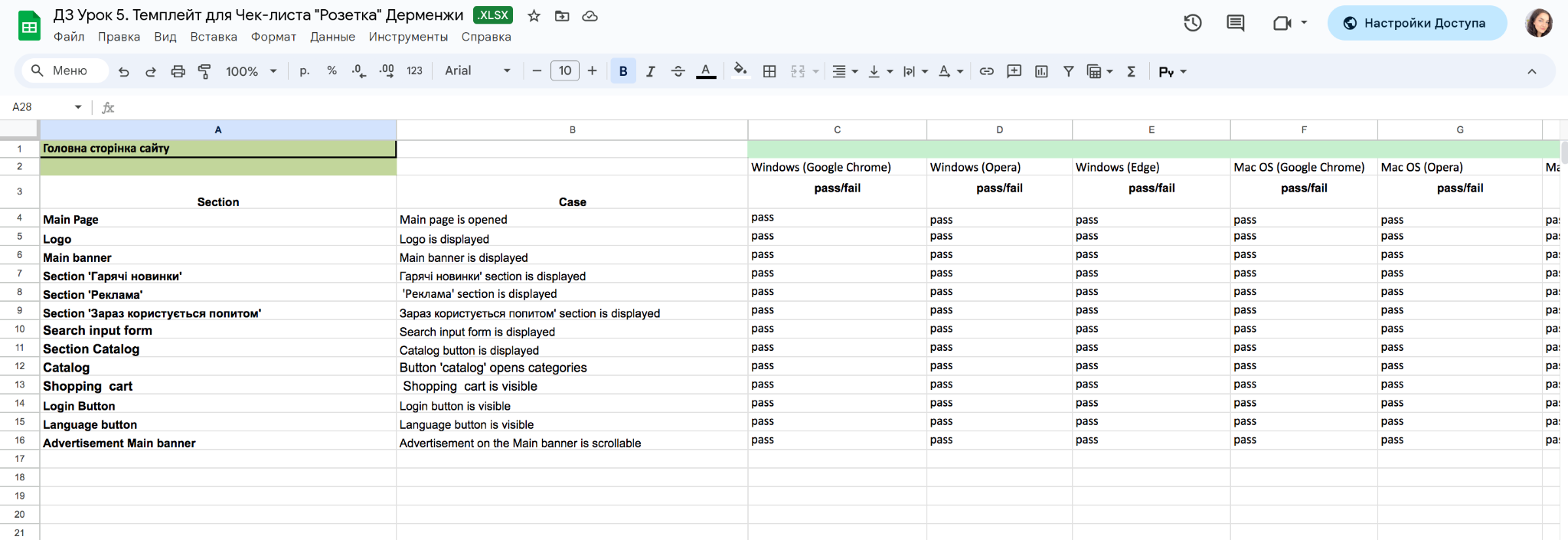
## Beet Seed — відпрацюй навички на базовому рівні.

1. Склади порівняльну таблицю трьох видів тестової документації:

| Назва тестової документації | № | Основні характеристики | Переваги | Недоліки |
| --- | --- | --- | --- | --- |
| Чек-ліст | 1 | Описує, які функції слід протестувати.  2 основні частини чек-ліста:   1. перелік функцій, реалізованих у конкретному продукті, працездатність яких підлягає перевірці; 2. список можливих тестів і помилок, які можуть існувати.   Використовується для розподілу завдань за рівнем кваліфікації та підтримки звітності та результатів тестування | * Використання чек-листів сприяє структуруванню інформації у співробітника. * При правильному записі необхідних дій у співробітника з'являється однозначне розуміння завдань. Це сприяє підвищенню швидкості навчання нових співробітників. * Чек-лісти допомагають уникнути невизначеності та помилок, що пов'язані з людським фактором. Збільшується покриття тестами програмного продукту. * Підвищується ступінь взаємозамінності співробітників. * Економія робочого часу. Написавши чекліст одного разу, його можна використовувати повторно, з огляду на актуальність інформації. Використання чек-листів – один із прийомів підвищення бас-фактора. У сфері розробки програмного забезпечення бас-фактор проекту («bus factor» – фактор автобуса) – це міра зосередження інформації серед окремих членів проекту. | Не вказано чіткі очікування від системи - не вказано ER, вказано тільки AR |
| Тест-кейс | 2 | Це набір дій, які виконуються для перевірки певної функції чи функціональності вашого програмного забезпечення.  Є 3 види тест-кейсів:   1. Positive test cases - очікується, що система, що тестується, працюватиме правильно 2. Negative test cases - очікується, що система, що тестується, буде несправною 3. Destructive test cases - тестовану систему навмисно зламано, щоб перевірити її можливості відновлення. Ці тести допомагають переконатися, що система може витримувати несподівані збої та продовжувати працювати належним чином. |  | * Не описує помилку * Не вказано AR, вказано тільки ER |
| Користувацький сценарій | 3 | У користувацьких сценаріях представлені варіанти того, як користувач може взаємодіяти з ПЗ, це перелік дій, сценарій, за яким користувач взаємодіє з додатком або програмою для виконання будь-якої дії та досягнення конкретної мети. | * Use Cases допомагають виявити невідповідності між вимогами й готовим програмним продуктом. * UC доводять, що програмне забезпечення працює коректно. | * Сценарії використання не є об’єктно орієнтованими. Вони орієнтовані на користувача. Для суто технічних і глибоко технологічних проєктів писати їх недоцільно – ви тільки марно витратите час та кошти. * Відсутні правила та формальні вимоги до того, як використовувати Use Cases, хто може їх розширювати, хто є акторами. Це доволі гнучка структура. * Не всі системи мають акторів, тобто не всі системи мають можливість ідентифікувати користувача як такого. У результаті ми отримуємо конфлікт, адже нам необхідно показати взаємодію двох систем за допомогою підходу, який орієнтований на користувача. * Use Cases засновані на підході Waterfall – вони виникли саме в цій методиці розробки. |

2. Склади чек-ліст для перевірки головної сторінки свого улюбленого інтернет-магазину (rozetka.ua, hotline.ua, silpo.ua тощо). Темплейт для чек-ліста можна взяти в кроці три теорії.

чек-ліст: https://docs.google.com/spreadsheets/d/1Ve06S--SWrZeP3LgpL-FGLcWle6YcgZ1/edit#gid=109836341



## Beet Sprout — детальніше заглибся в практику.

1. Виконай завдання попереднього рівня.

2. Склади 5 позитивних і 5 негативних тест-кейсів для перевірки основної функціональності твого улюбленого інтернет-магазину (rozetka.ua, hotline.ua, silpo.ua тощо)

test cases - <https://docs.google.com/spreadsheets/d/1Lx2FfbczzrsyDqO_m4jkpEUFyIUv2swL/edit#gid=533480428>

