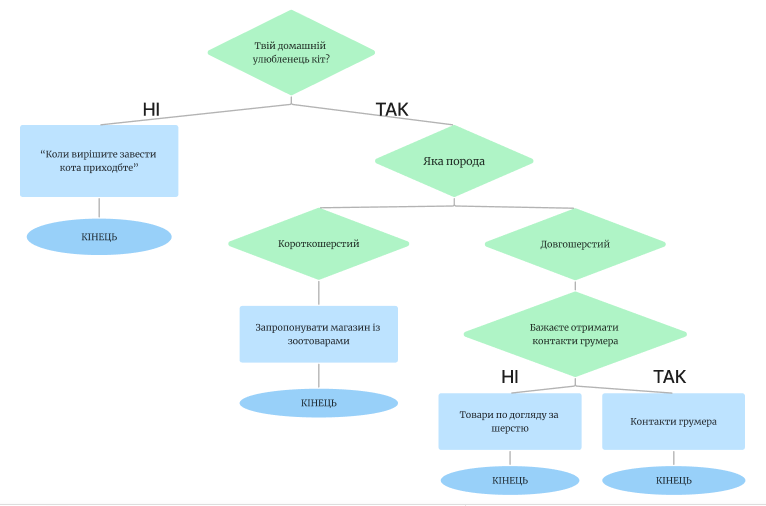
| Must have рівень:   1. Зроби порівняння статичних та динамічних технік тестування. Наведи переваги та можливі обмеження при використанні кожної з них.  |  | Статистична техніка тестування | Динамічна техніка тестування | | --- | --- | --- | | Основна інформація | Методика тестування ПЗ, при якій ПЗ тестується без запуску кода. Це також процес виявлення та усунення помилок і дефектів в різних супровідних документах (наприклад, специфікація вимог) | Методика тестування, що перевіряє функціональність програми, коли виконується код. Включає в себе тестування ПЗ в режимі реального часу шляхом надання вхідних даних і вивчення результату поведінки програми. | | Перевага №1 | Знижує вартість виправлення багів | Додаток тестується з точки зору користувача, що підвищує якість ПЗ | | Перевага №2 | Істотно скорочує зусилля по виправленні помилок | Може бути автоматизоване | | Перевага №3 | Покращує обмін критичної і важливої інформації між членами команди. | Виявлення складних помилок, які могли вислизнути на етапі код рев'ю. | | Перевага №4 | Відгуки, отримані в ході цього тестування, допомагають покращити функціонування процесу, що також допомагає команді уникнути подібних дефектів і багів. | Це ретельне дослідження, яке розглядає всю функціональність програми, тому якість відповідає найвищим стандартам. | | Обмеження №1 | Займає багато часу так як в основному він виконується вручну | Оскільки динамічне тестування являє собою складний процес, воно займає багато часу. | | Обмеження №2 | Перешкоджає виявленню вразливостей, представлених в середовищі виконання | Висока вартість проведення тестування | | Обмеження №3 |  | Знайдені баги виявляються пізніше в життєвому циклі розробки | | Висновок | Отже мета статичного тестування – запобігання дефектів програмного забезпечення.  Незважаючи на те, що статичне тестування вимагає багато часу, все ж варто витратити час на запобігання появи дефектів на останніх етапах розробки продукту. Тому статичне тестування по праву вважається важливим кроком на шляху до розробки ПЗ без помилок. | Отже мета динамічного тестування - пошук і усунення багів. Динамічне тестування також відіграє важливу роль. Завдяки безпосередньому виконанню тестів програмного забезпечення (перевірки функціональної поведінки, продуктивності, надійності та інших важливих аспектів) команда може перевірити і підтвердити якість і ефективність ПЗ. | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Середній рівень:   1. Виконай завдання попереднього рівня. 2. Наступне твердження стосується покриття рішень:   *Коли код має одну ‘IF” умову, не має циклів (LOOP) або перемикачів (CASE), будь-який тест, який ми виконаємо, дасть результат 50% покриття рішень (decision coverage).*  *if ( h > 21 )*  *{*  *print : “ end lesson ”*  *}*  *else*  *{*  *print: ‘ continue’*  *}*  Яке твердження є коректним?   * 1. Коректно. Будь-який тест кейс надає 100% покриття тверджень, таким чином покриває 50% рішень.   2. **Коректно. Результат будь-якого тесту умови IF буде або правдими, або ні.**   3. Некоректно. Один тест може гарантувати 25% перевірки рішень в цьому випадку.   4. Некоректно, бо занадто загальне твердження. Ми не можемо знати, чи є воно коректним, бо це залежить від тестованого ПЗ.  1. Є псевдокод:   Switch PC on -> Start MS Word -> IF MS Word starts THEN -> Write a poem -> Close MS Word.  Скільки тест кейсів знадобиться, щоб перевірити його функціонал?   * 1. **1 – для покриття операторів, 2 – для покриття рішень**   2. 1 – для покриття операторів, 1 – для покриття рішень   3. 2 – для покриття операторів, 2 – для покриття рішень   4. 2 – для покриття операторів, 1 – для покриття рішень  1. Скільки потрібно тестів для перевірки тверджень коду:     1. **2**    2. **1**    3. 3    4. 4 |
| Програма максимум:   1. Виконай завдання двох попередніх рівнів. 2. Продовжуємо розвивати стартап для застосунку, який дозволяє обмінюватися фотографіями котиків.   Є алгоритм:  *Запитай, якого улюбленця має користувач.*  *Якщо користувач відповість, що має кота, то запитай, яка порода його улюбленця: «короткошерста чи довгошерста?»*  *Якщо клієнт відповість «довгошерста», то запитай: «ви бажаєте отримати контакти найближчого грумера?»*  *Якщо клієнт відповість «так», то скажи: «Надайте адресу найближчої котячої перукарні»*  *Інакше*  *Скажи: «Запропонуй магазин з товарами по догляду за шерстю»*  *Закінчити*  *Інакше*  *Скажи «Запропонуй обрати магазин із зоотоварами»*  *Закінчити*  *Якщо клієнт не має кота*  *Скажи “Коли вирішите завести улюбленця – приходьте”*  *Закінчити*  Завдання:   1. Намалюй схему алгоритму (в інструменті на вибір, наприклад, у вбудованому Google Docs редакторі, [figjam](https://www.figma.com/figjam/) чи [miro](https://miro.com/)) 2. Який потрібен мінімальний набір тест-кейсів, щоб переконатися, що всі запитання були поставлені, всі комбінації були пройдені та всі відповіді були отримані? |



Оскільки в нашому алгоритмі 4 вітки з варіантами розвитку подій, то нам необхідно буде створити 4 тест-кейси, щоб переконатися, що всі запитання були поставлені, всі комбінації були пройдені та всі відповіді були отримані