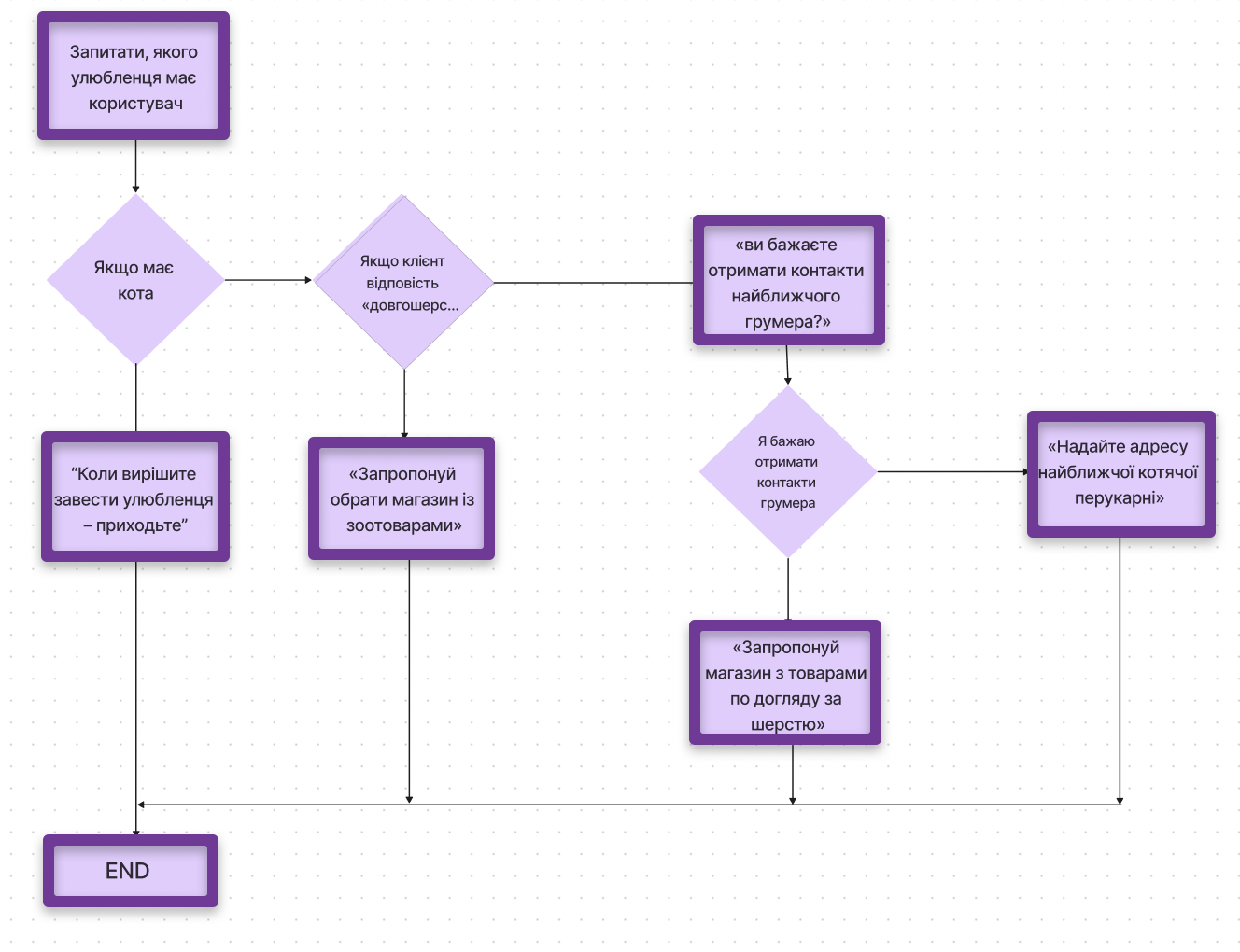
| Must have рівень:   1. Зроби порівняння статичних та динамічних технік тестування. Наведи переваги та можливі обмеження при використанні кожної з них.  |  | Статистична техніка тестування | Динамічна техніка тестування | | --- | --- | --- | | Основна інформація | Це тип тестування програмного забезпечення, при якому програмний додаток тестується без виконання коду. З метою пошуку помилок проводиться ручний або автоматизований перегляд коду, документів вимог та дизайну документації. Основною метою статичного тестування є підвищення якості програмних додатків шляхом виявлення помилок на ранніх стадіях процесу розробки програмного забезпечення. | Під час динамічного тестування виконується код. Перевіряється функціональна поведінка програмної системи, використання пам'яті/процесора та загальна продуктивність системи.  Основна мета цього тестування - підтвердити, що програмний продукт працює відповідно до бізнес-вимог. Це тестування також називається технікою виконання або валідаційним тестуванням. | | Перевага №1 | Раннє виявлення та виправлення будь-яких помилок кодування. | Він виявляє вразливості в середовищі виконання. | | Перевага №2 | Зниження вартості на ранніх стадіях розробки на основі обсягу доопрацювань, необхідних для виправлення помилок. | Автоматизовані інструменти забезпечують гнучкість у виборі об'єктів для сканування. | | Перевага №3 | Скорочення термінів розробки. | Це дозволяє аналізувати додатки, в яких ви не маєте доступу до фактичного коду. | | Перевага №4 | Цей процес також допоможе отримати краще уявлення про проблеми з якістю, знайдені в програмному забезпеченні. | Динамічне тестування також може перевірити коректність результатів статичного тестування. | | Перевага №5 | Це відносно швидко, якщо використовуються автоматизовані інструменти. | Динамічне тестування може бути застосоване до будь-якого додатку. | | Обмеження №1 | Це займає багато часу, якщо проводити його вручну. | Складніше відстежити вразливість до точного місця в коді, що призводить до збільшення часу на усунення проблеми. | | Обмеження №2 | Автоматизовані інструменти хороші настільки, наскільки хороші правила, за якими вони сканують. | Автоматизовані інструменти настільки хороші, наскільки хороші правила, за якими вони сканують. | | Обмеження №3 (і т.д.) | Вони не знаходять вразливостей, впроваджених у середовищі виконання. | Динамічне тестування займає багато часу, оскільки виконується додаток/програма або код, що потребує багато ресурсів. | | Висновок | Незважаючи на те, що статичне тестування вимагає тривалого часу, воно варте того, щоб запобігти появі дефектів на останніх етапах розробки продукту. Тому статичне тестування по праву вважається основним кроком на шляху до розробки програми без помилок. | Динамічне тестування в життєвому циклі програмного забезпечення (STLC) має величезне значення. Від безпосереднього тестування програмного забезпечення до перевірки функціональності, ефективності, надійності та інших важливих елементів програмного забезпечення. | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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).*  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? |



Statement coverage = 4 TC

Decision coverage = 4 TC