QAEngineer
школа підготовки тестувальників

https://iqaengineer.com/ua

Техніки тест-дизайну використовуються для планування та створення тест-кейсів, що дозволяє забезпечити максимальне покриття функціональності, вимог та можливих дефектів програмного забезпечення. Ось декілька поширених технік тест-дизайну:

1. **Еквівалентність значень:** Ця техніка базується на припущенні, що якщо вхідні дані належать до одного класу еквівалентності, то результат повинен бути однаковим. Тестувальник обирає представника з кожного класу еквівалентності для включення до тест-кейсів.

Припустимо, у вас є форма реєстрації користувачів з полем "Вік", яке приймає значення від 18 до 65 років. Застосуємо техніку еквівалентності значень для створення тест-кейсів:

Вибірка з класу еквівалентності "Валідний вік":

a. Тест-кейс 1: Ввести вік 25 (знаходиться в межах допустимого діапазону).
b. Тест-кейс 2: Ввести вік 40 (знаходиться в межах допустимого діапазону).
c. Тест-кейс 3: Ввести вік 60 (знаходиться в межах допустимого діапазону).

Вибірка з класу еквівалентності "Неприпустимий вік":

d. Тест-кейс 4: Ввести вік 17 (менше мінімально допустимого значення).
e. Тест-кейс 5: Ввести вік 70 (більше максимально допустимого значення).

Вибірка з класу еквівалентності "Некоректні значення":

f. Тест-кейс 6: Ввести вік -5 (некоректне від'ємне значення).
g. Тест-кейс 7: Ввести вік 1000 (некоректне велике значення).

h. Тест-кейс 8: Ввести вік "дев'ять" (некоректне текстове значення).

2. **Границі значень:** Ця техніка передбачає тестування на межових значеннях. Вибираються значення, які знаходяться непрямо на межі допустимих діапазонів, такі як мінімальні та максимальні значення або значення, які призводять до зміни поведінки програми.

Задача: Перевірити функціональність розрахунку знижки в інтернет-магазині на основі загальної вартості замовлення.

Умови:

- Загальна вартість замовлення може бути в діапазоні від 0 до 1000 одиниць валюти.
- Знижка розраховується відповідно до таких правил:
  - Якщо загальна вартість замовлення менша або дорівнює 100 одиницям валюти, знижка не надається.
  - Якщо загальна вартість замовлення більша за 100 одиниць валюти, але менша або дорівнює 500 одиницям валюти, знижка становить 10% від загальної вартості замовлення.
  - Якщо загальна вартість замовлення більша за 500 одиниць валюти, знижка становить 20% від загальної вартості замовлення.

Тест-кейси:

Тест-кейс: Перевірка розрахунку знижки для мінімальної вартості замовлення. Вхідні дані:

a. Загальна вартість замовлення: 0 Очікуваний результат:
b. Знижка: 0 (знижка не надається)

Тест-кейс: Перевірка розрахунку знижки для значення в діапазоні без знижки. Вхідні дані:

c. Загальна вартість замовлення: 100 Очікуваний результат:
d. Знижка: 0 (знижка не надається)

Тест-кейс: Перевірка розрахунку знижки для значення в діапазоні з 10% знижкою. Вхідні дані:

e. Загальна вартість замовлення: 300 Очікуваний результат:
f. Знижка: 30 (10% від 300)

Тест-кейс: Перевірка розрахунку знижки для значення в діапазоні з 20% знижкою. Вхідні дані:

g. Загальна вартість замовлення: 600 Очікуваний результат:
h. Знижка: 120 (20% від 600)

Тест-кейс: Перевірка розрахунку знижки для максимальної вартості замовлення. Вхідні дані:

i. Загальна вартість замовлення: 1000 Очікуваний результат:
j. Знижка: 200 (20% від 1000)

3. **Парне тестування:** При використанні цієї техніки вхідні дані розбиваються на пари, і кожна пара використовується для створення тест-кейсу. Це дозволяє перевірити взаємодію між різними вхідними значеннями.

Припустимо, у нас є форма входу на веб-сайт, яка містить поля "Електронна пошта" і "Пароль". Ми хочемо протестувати цю форму, перевірити взаємодію різних комбінацій вхідних значень. Використаємо парне тестування для створення ефективних тест-кейсів:

Вхідні значення:

a. Електронна пошта: user1@example.com, user2@example.com
b. Пароль: password1, password2

Парні комбінації:

c. Комбінація 1: (user1@example.com, password1)
d. Комбінація 2: (user1@example.com, password2)
e. Комбінація 3: (user2@example.com, password1)
f. Комбінація 4: (user2@example.com, password2)

Створення тест-кейсів:

g. Тест-кейс 1: Ввійти в систему, використовуючи комбінацію 1
h. Тест-кейс 2: Ввійти в систему, використовуючи комбінацію 2
i. Тест-кейс 3: Ввійти в систему, використовуючи комбінацію 3
j. Тест-кейс 4: Ввійти в систему, використовуючи комбінацію 4

4. **Перебори (Combinatorial Testing)**: Ця техніка використовується для тестування комбінацій можливих вхідних значень. Застосування методів перебору дозволяє зменшити кількість тест-кейсів, що потрібно перевірити, зберігаючи при цьому важливі комбінації.

Уявімо, що у нас є програмне забезпечення для розрахунку вартості поїздки в таксі. Це програма, яка приймає наступні параметри: відстань поїздки (в кілометрах), тип автомобіля (економ, стандарт або преміум) і наявність знижки (так або ні).

Для перебору можливих комбінацій вхідних значень ми можемо скористатися технікою переборів.

Параметри:

- Відстань поїздки: 10 км, 20 км, 30 км
- Тип автомобіля: економ, стандарт, преміум
- Наявність знижки: так, ні

За допомогою техніки перебору, ми можемо створити наступні комбінації:

1. Відстань поїздки: 10 км, Тип автомобіля: економ, Наявність знижки: так
2. Відстань поїздки: 10 км, Тип автомобіля: стандарт, Наявність знижки: так
3. Відстань поїздки: 10 км, Тип автомобіля: преміум, Наявність знижки: так
4. Відстань поїздки: 20 км, Тип автомобіля: економ, Наявність знижки: так
5. Відстань поїздки: 20 км, Тип автомобіля: стандарт, Наявність знижки: так
6. Відстань поїздки: 20 км, Тип автомобіля: преміум, Наявність знижки: так
7. Відстань поїздки: 30 км, Тип автомобіля: економ, Наявність знижки: так
8. Відстань поїздки: 30 км, Тип автомобіля: стандарт, Наявність знижки: так
9. Відстань поїздки: 30 км, Тип автомобіля: преміум, Наявність знижки: так
10. Відстань поїздки: 10 км, Тип автомобіля: економ, Наявність знижки: ні
11. Відстань поїздки: 10 км, Тип автомобіля: стандарт, Наявність знижки: ні
12. Відстань поїздки: 10 км, Тип автомобіля: преміум, Наявність знижки: ні
13. Відстань поїздки: 20 км, Тип автомобіля: економ, Наявність знижки: ні
14. Відстань поїздки: 20 км, Тип автомобіля: стандарт, Наявність знижки: ні
15. Відстань поїздки: 20 км, Тип автомобіля: преміум, Наявність знижки: ні
16. Відстань поїздки: 30 км, Тип автомобіля: економ, Наявність знижки: ні
17. Відстань поїздки: 30 км, Тип автомобіля: стандарт, Наявність знижки: ні

18. Відстань поїздки: 30 км, Тип автомобіля: преміум, Наявність знижки: ні

Таким чином, ми створили 18 комбінацій для тестування, що дозволяє перевірити різні комбінації вхідних значень і впевнитися, що програма правильно розраховує вартість поїздки в залежності від цих параметрів.

5. **Поведінка межових значень:** Ця техніка ставить за мету перевірити поведінку програми на межі допустимих значень вхідних даних. Наприклад, випробовуються значення, що близькі до межі, але трохи поза нею, для перевірки коректності обробки.

Приклад використання техніки "Поведінка межових значень" для створення тест-кейсу:

Припустимо, що у нас є функція розрахунку вартості доставки товару в залежності від ваги замовлення. Допустимий діапазон ваги встановлено від 1 кг до 10 кг, а вартість доставки змінюється наступним чином:

- До 3 кг - вартість доставки 5 грн.
- Від 3 кг до 6 кг - вартість доставки 10 грн.
- Понад 6 кг - вартість доставки 15 грн.

Використовуючи техніку "Поведінка межових значень", ми можемо створити наступний тест-кейс:

Тест-кейс:

Вага замовлення: 1 кг. Очікуваний результат: Вартість доставки 5 грн.

Вага замовлення: 3 кг. Очікуваний результат: Вартість доставки 5 грн.

Вага замовлення: 4 кг. Очікуваний результат: Вартість доставки 10 грн.

Вага замовлення: 6 кг. Очікуваний результат: Вартість доставки 10 грн.

Вага замовлення: 7 кг. Очікуваний результат: Вартість доставки 15 грн.

Вага замовлення: 10 кг. Очікуваний результат: Вартість доставки 15 грн.

Вага замовлення: 11 кг. Очікуваний результат: Вартість доставки - недопустима вага, повідомлення про помилку.

6. Умовне покриття: Ця техніка спрямована на покриття різних логічних умов та гілок в програмному коді. Вибираються тест-кейси, які задовольняють різні комбінації умов, для перевірки коректності умовних конструкцій.

Уявімо, що у нас є програма для калькулятора, яка виконує операції додавання, віднімання, множення та ділення. Ми хочемо перевірити правильність роботи цієї програми з використанням умовного покриття.

Тест-кейс: Опис: Перевірка додавання чисел Умови:

- Числа A та B є цілими числами
- Число A менше за число B

Кроки:

1. Запустити калькулятор
2. Ввести значення A = 5
3. Ввести значення B = 10
4. Натиснути кнопку додавання
5. Перевірити, чи відображається правильний результат (A + B)
6. Закрити калькулятор

В даному прикладі тест-кейс забезпечує покриття умов, коли вхідні числа є цілими числами та коли число A менше за число B. Це допомагає перевірити, як програма впорається з такими умовами та правильно виконує операцію додавання.

7. **Аналіз помилок:** При цій техніці виходять з припущення, що програмні помилки часто повторюються. Аналізуються попередні помилки та використовуються для створення тест-кейсів, що охоплюють ці помилки.

Ці техніки тест-дизайну допомагають систематично планувати тестування та створювати ефективні тест-кейси, що спрямовані на виявлення помилок та забезпечення якісного програмного забезпечення.