Відповіді на контрольні питання

1. Навіщо потрібні тест-кейси?

Тест-кейси потрібні для того, щоб **системно перевіряти якість продукту** і впевнитися, що він працює так, як очікується. Вони виконують кілька важливих функцій:

- **Перевірка функціональності** – чи працюють усі частини продукту відповідно до вимог.
- **Виявлення помилок** – допомагають знайти дефекти ще до того, як продукт потрапить до користувача.
- **Документування процесу тестування** – описують, що саме перевірялося, як і з яким результатом.
- **Відтворюваність** – інший тестувальник може повторити ті самі кроки й отримати той самий результат.
- **Прозорість для команди** – розробники, менеджери й замовники бачать, що саме перевірено.
- **Оцінка якості продукту** – дозволяють зрозуміти, чи готовий продукт до використання.

Наприклад:

Назва: Перевірка входу в систему з правильними даними

Pre-condition: Користувач має зареєстрований акаунт

Кроки:

1. Ввести правильний логін
2. Ввести правильний пароль
3. Натиснути "Увійти"

Expected Result: Система відкриває головну сторінку

Post-condition: Користувач авторизований

2. Основні атрибути Test Case?

Основні атрибути **Test Case** – це структурні елементи, які роблять його зрозумілим, повторюваним і корисним для перевірки якості продукту.

Основні атрибути Test Case:

- **ID (унікальний ідентифікатор)** – номер або код для швидкого пошуку та посилання.
- **Назва (Title)** – короткий опис того, що перевіряється.
- **Pre-condition (попередні умови)** – стан системи або дані, які повинні бути перед виконанням тесту.
- **Test Steps (кроки виконання)** – послідовність дій, які потрібно виконати.
- **Expected Result (очікуваний результат)** – що саме має відбутися після виконання кроків.
- **Post-condition (стан після тесту)** – що залишається після виконання тесту, чи готова система до наступних дій.
- **Priority (пріоритет)** – наскільки важливий цей тест для перевірки (високий, середній, низький).
- **Test Data (тестові дані)** – конкретні значення, які використовуються під час тестування.
- **Actual Result (фактичний результат)** – що реально відбулося під час виконання тесту.
- **Status (статус виконання)** – Passed / Failed / Blocked / Not Executed.
- **Author (автор)** – хто створив тест-кейс.
- **Date (дата створення/оновлення)** – коли був написаний або змінений.

3. Типи тест-кейсів.

Типи тест-кейсів за рівнем тестування

- **Unit Test Cases (модульні)** – перевіряють окремі функції чи компоненти системи.
- **Integration Test Cases (інтеграційні)** – перевіряють взаємодію між кількома модулями.
- **System Test Cases (системні)** – перевіряють роботу всієї системи як єдиного цілого.
- **Acceptance Test Cases (приймальні)** – перевіряють відповідність продукту вимогам замовника.

Типи тест-кейсів за метою

- **Функціональні** – перевіряють, чи функції працюють відповідно до вимог.
- **Нефункціональні** – перевіряють характеристики, що не пов’язані з функціональністю: продуктивність, безпека, зручність використання.
- **Регресійні** – перевіряють, чи нові зміни не зламали існуючий функціонал.
- **Smoke (димові)** – швидка перевірка базової працездатності системи.
- **Sanity (адекватності)** – перевірка конкретної функції після внесення змін.

Типи тест-кейсів за видом перевірки

- **Позитивні** – перевіряють правильну поведінку системи при коректних даних.
- **Негативні** – перевіряють реакцію системи на некоректні або неочікувані дані.
- **Edge Cases (граничні)** – перевіряють поведінку на межових значеннях (мінімум, максимум).
- **Exploratory (дослідницькі)** – менш формалізовані, використовуються для пошуку несподіваних помилок.

4. Що таке негативний тест-кейс?

Негативний тест-кейс — це тест, який перевіряє, **як система поводиться при некоректних або неочікуваних даних чи діях користувача**. Його мета — переконатися, що продукт правильно обробляє помилки, не «ламається» і видає зрозуміле повідомлення про помилку.

5. Що повинен знати тестувальник?

Що повинен знати тестувальник

1. Основи тестування

- Принципи тестування (чому тестування потрібне, які його цілі).
- Життєвий цикл розробки ПЗ (SDLC, STLC).
- Типи тестування: функціональне, нефункціональне, регресійне, smoke, sanity тощо.
- Види тест-кейсів (позитивні, негативні, граничні).
- Робота з баг-репортами (як правильно описати помилку).

2. Тестова документація

- Як писати тест-кейси, чеклісти, тест-плани.
- Як вести звітність про результати тестування.
- Використання систем управління тестами (TestRail, Zephyr, Qase).

3. Інструменти

- Системи баг-трекінгу (Jira, Trello, YouTrack).
- Системи контролю версій (Git, GitHub, GitLab).
- Інструменти автоматизації тестів (Selenium, Cypress, Playwright).
- Інструменти для API-тестування (Postman, Swagger).
- Інструменти для роботи з базами даних (SQL).

4. Технічні знання

- Основи програмування (Java, Python, JavaScript — хоча б базовий рівень).
- Робота з базами даних (запити SQL, перевірка даних).
- Основи мереж (HTTP, HTTPS, REST, SOAP).
- Робота з ОС (Windows, Linux, macOS).

5. Soft Skills

- Уважність до деталей.
- Аналітичне мислення.
- Вміння чітко формулювати думки (щоб баг-репорти були зрозумілими).
- Комунікація з командою (розробники, менеджери, замовники).
- Тайм-менеджмент і пріоритизація задач.

6. Скільки основних принципів тестування?

7 принципів тестування

1. **Тестування показує наявність дефектів, але не доводить їх відсутність** – тестування може виявити помилки, але ніколи не гарантує, що їх немає.
2. **Вичерпне тестування неможливе** – перевірити всі можливі комбінації даних і сценаріїв нереально, тому тестування завжди вибіркове.
3. **Раннє тестування економить час і гроші** – чим раніше знайдено дефект, тим дешевше його виправити.
4. **Дефекти концентруються в певних місцях** – більшість помилок зазвичай зосереджені в обмеженій кількості модулів або функцій.
5. **Парадокс пестициду** – якщо постійно виконувати одні й ті самі тести, вони перестають знаходити нові помилки. Тести потрібно оновлювати.
6. **Тестування залежить від контексту** – підхід до тестування різний для банківської системи, мобільного додатку чи ігрового сервісу.
7. **Відсутність помилок — не головна мета** – навіть якщо система технічно бездефектна, але не відповідає бізнес-вимогам, вона неякісна.