1. Czym się różnią testy funkcjonalne od niefunkcjonalnych?

Testy funkcjonalne wykonujemy aby zweryfikować czy działanie systemu informatycznego jest zgodne ze specyfikacją funkcjonalną zawartą w dokumencie analitycznym, wymaganiami funkcjonalnymi oraz wybranymi wymaganiami utrzymaniowymi wynikającymi bezpośrednio z potrzeb biznesowych Zamawiającego lub domniemaniach przez testerów.

Natomiast testy niefunkcjonalne służą do pomiarów charakterystyk systemu i oprogramowania, które nie są związane z określoną funkcją lub działaniem użytkownika, takie jak skalowalność systemu, wydajność, bezpieczeństwo czy czas odpowiedzi aplikacji.

2. Co to są 'smoke testy' i 'testy regresji'? Kiedy je stosujemy?

Testowanie regresywne polega na powtórzeniu wcześniej opracowanych testów na już przetestowanym programie. Wykonywane są po modyfikacjach w tym programie lub jego środowisku, w celu wykrycia nowych usterek, niezamierzonych zmian lub usterek odsłoniętych na skutek wykonanych zmian. Usterki te mogą występować w testowanym oprogramowaniu, jak również w innych powiązanych lub niepowiązanych modułach.

Testowanie regresyjne może być przeprowadzone jeśli modyfikacje systemu nie były znaczące. W przeciwnym przypadku konieczna jest także modyfikacja samych wariantów testu.

„Smoke testy" to uproszczona forma testowania regresyjnego zwana testem na dym. Jest tp podzbiór wszystkich zdefiniowanych/ zaplanowanych przypadków testowych, które pokrywają główne funkcjonalności modułu lub systemu, mający na celu potwierdzenie, że kluczowe funkcjonalności programu działają, bez zagłębiania się w szczegóły. „Smoke testy" powinny być wykonywane reguralnie, w celu wskazania obszarów które należy poddać pełnym testom regresyjnym, dobrą praktyką jest również posiadanie kilku zestawów smoke testów.

3. Co jest celem testowania?

- Wykrycie istniejących błędów
- Nabieranie zaufania do poziomu jakości
- Dostarczanie informacji potrzebnych do podejmowania decyzji
- Zapobieganie defektom
- Udowodnienie, że oprogramowanie nie zawiera błędów

4. Jak tester może się upewnić, że błąd został naprawiony?

Wykonując test potwierdzający

5. Testujesz aplikacje termometr która wykonuje pomiar temperatury. Co byś zrobił aby przetestować zachowanie aplikacji przy skrajnych wartościach -50C i 200C ?

Sprawdzenie temperatury 200C w piekarniku z termo-obiegiem

Sprawdzenie temperatury -50C w niskotemperaturowej zamrażarce np.VT407(lub odwiedzając Syberię ;))

6. Ile przypadków testowych potrzeba, aby pokryć wszystkie możliwości?

4:

- a > 0, b=3
- a>0, b≠3
- a≤0, b=3
- a≤0, b≠3

7. Dany jest input „wiek", który przyjmuje wartości od 18 do 60. Twoim zadaniem jest przetestować go za pomocą techniki wartości brzegowych. Jakie wartości wpisujesz do inputu? Podaj wszystkie liczby, które wpisujesz.

Wiek: 17, 18, 60, 61

8. Dołączasz do projektu w trakcie develepmentu aplikacji, do której nie ma dokumentacji. Schemat logowania do aplikacji wygląda następująco: Jakie pytania zadasz analitykowi, zanim przystąpisz do testów logowania?

- Skąd wziąć dane testowe do logowania?
- Czy baza danych jest lokalna czy zdalna?
- Jak powinna wyglądać obsługa błędów logowania:
    - Złe hasło,
    - Nieistniejący użytkownik
    - Konto nieaktywowane
- Ponieważ schemat jest niedokładny to zapytałabym, które pole służy do wprowadzenia hasła, a które do nazwy użytkownika (nie są opisane)?
- W jaki sposób następuje wysłanie formularza- na schemacie nie ma przycisku do wysyłania?
- Czy jest limit nieudanych prób logowania?

9. Czym się różni metoda GET od POST?

Metody GET używa się kiedy parametrów jest niewiele, natomiast metodę POST umożliwia przekazywanie dużo większych parametrów.

Parametry w metodzie GET są widoczne w pasku adresu przeglądarki, więc tej metody nie należy używać jeśli przekazywane są np. hasła. W metodzie POST parametrów nie widać w pasku adresu przeglądarki.

W metodzie GET parametry przekazuje się za pomocą adresu URL, można też przekazywać parametry przez odnośnik. Metoda „POST" do przekazywania parametrów wykorzystuje nagłówek zapytania.,

10. Czy HTTP jest protokołem zmiennostanowym?

Nie, bezstanowym

11. Czym różni się LEFT JOIN od INNER JOIN?

- LEFT JOIN jest typem złączenia w którym z lewej tabeli pobierane są wszystkie rekordy, rekordy te uzupełniane są rekordami z prawej tabeli, o ile uda się dopasować je po kluczu, jeśli się nie uda to wstawiane są wartości null
- INNER JOIN natomiast pobiera jedynie te rekordy z lewej i prawej tabeli, które uda się dopasować po kluczu(tworzy zbiór wspólny obu tabel)

Przez klucz w tym wypadku należy rozumieć to co znajduje sie w klauzuli "ON"

12. W jakim katalogu, standardowo Linux trzyma pliki konfiguracyjne:

c. /etc

13. Jak przetestowałbyś bashową komendę cp? (argumenty funkcji można pominąć)

- bez żadnych argumentów
- z jednym argumentem – istniejącym plikiem źródłowym
- z jednym argumentem – nieistniejącym plikiem źródłowym
- z dwoma argumentami – istniejącym plikiem źródłowym i nieistniejącym plikiem docelowym w tym samym katalogu co plik źródłowy
- z dwoma argumentami – nieistniejącym plikiem źródłowym i istniejącym plikiem docelowym w tym samym katalogu co plik źródłowy
- z dwoma argumentami – istniejącym plikiem źródłowym i istniejącym plikiem docelowym w tym samym katalogu co plik źródłowy
- z dwoma argumentami – istniejącym plikiem źródłowym i nieistniejącym plikiem docelowym w innym(istniejącym) katalogu niż plik źródłowy
- z dwoma argumentami – nieistniejącym plikiem źródłowym i istniejącym plikiem docelowym w innym(istniejącym) katalogu niż plik źródłowy
- z dwoma argumentami – istniejącym plikiem źródłowym i istniejącym plikiem docelowym w innym(istniejącym) katalogu niż plik źródłowy
- z dwoma argumentami – istniejącym plikiem źródłowym i nieistniejącym plikiem docelowym w innym(nieistniejącym) katalogu niż plik źródłowy
- z dwoma argumentami – nieistniejącym plikiem źródłowym i istniejącym plikiem docelowym w innym(nieistniejącym) katalogu niż plik źródłowy
- z dwoma argumentami – istniejącym plikiem źródłowym i istniejącym plikiem docelowym w w innym(nieistniejącym) katalogu niż plik źródłowy