1. Czym się różnią testy funkcjonalne od niefunkcjonalnych?

Testy funkcjonalne	**Testy niefunkcjonalne**
- to inaczej testy wymagań funkcji np. systemu, podsystemu, modułu udokumentowanych lub znanych testerom - zawierają opis zachowania systemu „Co system robi?" - wykonuje je się na wszystkich poziomach testów - są mierzalne jeśli chodzi o pokrycie (wszystkie funkcje zostały przetestowane) - są podatne na automatyzację - tester ma dostęp do kodu testowanej aplikacji - testy są wykonywane przez osobę, która nie tworzyła aplikacji w oparciu o dokumentację oraz założenia funkcjonalne - pozawalają na wykrycie błedów związanych z np. brakiem implementacji funkcjonalności opisanych przez wymagania ale nie pozwalaja precyzyjnie określić miejsca gdzie błąd występuje - przypadki testowe tworzone są pod kątem sprawdzenia czy aplikacja zachowuje się w oczekiwany sposób - testy funkcjonalne pokrywają funkce dostępne dla użytkownika jak np. interfejs jak i operacje typu back-end (bezpieczeństwo czy, wpływ aktualizacji na działanie systemu) - często wykonywane pod koniec pod koniec cyklu wytwarzania oprogramowania ale mogą i powinny być uruchamiane znacznie wcześniej **Testowanie funkcjonalne obejmuje** : testy jednostkowe, dymne, integracyjne, testy interfejsu i użyteczności, systemowe, regresji, alfa i beta, testy akceptacyjne użytkownika, testowanie metodami białej i czarnej skrzynki.	- może być wykonywane na wszystkich poziomach testów - w większości przypadków wykorzystują techniki czarnoskrzynkowe - właściwości niefunkcjonalne opisują następujące cechy testowanego produktu: Wydajność (testy wydajności, obciążeniowe i przeciążające), użyteczności, ergonomii, współpracy (międzyoperacyjności) niezawodności, efektywności, przeznaczalności, możliwości procedury instalacyjnej - testy niefunkcjonalne odpowiadają na pytania „Jak działa system?"

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

Smoke testy
Termin "smoke test" funkcjonuje obecnie głównie w kontekście procesów wytwarzania oprogramowania, a zwłaszcza kontroli jakości. W kontekście testowania oprogramowania i złożonych systemów informatycznych należy rozumieć smoke test (przekładany czasem na "test dymny") jako:

- **test pobieżny** (będzie to często test przeszukujący "wszerz", a nie "w głąb"; często będzie to test najbardziej typowej ścieżki czynności, którą może przebyć potencjalny użytkownik)**,**
- **test zajmujący niewiele czasu** (ma szybko udzielić informacji koniecznych do podjęcia decyzji, co zrobimy dalej w ramach testowania)**,**
- **test poszukujący bardzo wyraźnych problemów** (test zdany pomyślnie powinien wykluczyć zachodzenie ewidentnej awarii na swojej ścieżce),
- **test dopuszczający do kolejnego etapu prac** (zwłaszcza w kontekście zaangażowania w tym kolejnym etapie znaczących zasobów).

Mogą być zaprojektowane zarówno we wczesnych etapach wytwarzania oprogramowania, kiedy ustalane są najistotniejsze ścieżki przechodzenia przez aplikację - możemy na przykład wybrać najbardziej priorytetowe albo ryzykowne przypadki użycia (use case'y) i zaprojektować smoke test, który przejdzie przez te z najistotniejszych dla klienta, które pokrywają najwięcej obszarów aplikacji za jednym zamachem.

Nie wyklucza to możliwości tworzenia dodatkowych smoke testów później w procesie wytwarzania, gdy na przykład wydarzenia podczas projektu wskazują na prawdopodobieństwo zachodzenia "krytycznej" awarii. Bardzo typowe będzie dla zespołu testowego wykonywanie smoke testu, który konkretnie ma upewnić zespół, że, przykładowo, po ostatnim wdrożeniu najnowszej wersji systemu, ta wyjątkowo groźna awaria w znanym nam miejscu nie występuje.

Testy regresji

upewnienie się, że aplikacja działa po dokonaniu w niej modyfikacji, poprawieniu błędów lub po dodaniu nowej funkcjonalności.

Cecha: powtarzalność

Co dają testy regresywne:

- Wyszukanie błędów powstałych w wyniku zmian kodu/środowiska.
- Ujawnienie wcześniej nie odkrytych błędów.
- Jest to dobry kandydat do automatyzacji ze względu na swoją powtarzalność.

Iteracyjne metodologie oraz krótkie cykle w których dostarczane są kolejne funkcjonalności powodują, że testy regresywne pozwalają upewnić się czy nowe funkcjonalności nie wpłynęły negatywnie na istniejące już i działające części systemu.

Testy regresywne mogą być przeprowadzane dla kompletnego produktu lub jego części. Jeżeli pomiędzy cyklami testów nie można przeprowadzić pełnych testów regresywnych aplikacji (koszty, czas), przypadki testowe, które włączamy do testu, powinny być dobierane na podstawie:

- Jakie błędy zostały poprawione, jakie rozszerzenia czy zmiany zostały wprowadzone
- Których obszarów aplikacji zmiany te dotyczą najbardziej
- Jaki jest wpływ wprowadzonych zmian na inne części systemu

Testy regresywne powinny być przeprowadzane po smoke testach lub testach typu sanity. Ten typ testów pozwala upewnić się czy otrzymana nowa wersja aplikacji jest "testowalna" i czy warto zaczynać z nią pracę.

3. Co jest celem testowania

Wykrywanie podstawowych błędów, by je usunąć,

-Sprawdzanie zgodności z innymi aplikacjami,

-Ułatwianie podjęcia decyzji interesariuszom o wypuszczaniu lub niewypuszczaniu produktu,

-Minimalizacja kosztów pomocy technicznej,

-Kontrola zgodności z wymaganiami i z prawnymi uregulowaniami.

- W testach akceptacyjnych głównym celem może być potwierdzenie, że system działa tak jak powinien oraz nabranie pewności, że spełnia wymagania.
- Testowanie pielęgnacyjne często zawiera testy sprawdzające, czy nie wprowadzono nowych usterek podczas wykonywania zmian. Głównym celem testowania produkcyjnego może być ocena atrybutów systemu takich jak niezawodność lub dostępność

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

Poprzez wykonanie Re-testu (powtórzenie testu, test potwierdzający) – powtórne wykonanie przypadku testowego, którego pierwotne wykonanie wykazało awarię (nieprawidłowe działanie) testowanego produktu. Celem re-testu jest sprawdzenie, czy defekt będący przyczyną awarii został usunięty.

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

- 51 °C, -50 °C, 200 °C , 201 °C

6. Ile przypadków testowych potrzeba, aby pokryć wszystkie możliwości?
Aby pokryć wszystkie możliwości potrzebne są 4 przypadki testowe

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.

17, 18, 19, 59, 60, 61 zgodnie z teorią wartości brzegowych.

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

- Czy przeprowadzono walidację logowania? Czy jest określona ilość znaków minimalna i maksymalna. Czy nie przekracza np. maksymalnej ilości znaków
- Czy zostało sprawdzone użycie znaków takich jak apostrof, cudzysłów, średniki, ukośniki

- Czy sprawdzono logowanie pod kontem użycia białych znaków?

- Czy został stworzony przypadek testowy dla funkcji logowania np.

Kroki do wykonania:
podaj login i hasło [zgodnie ze zdefiniowaną walidacją]

Oczekiwany rezultat:
Poprawny login / hasło - zalogowany
Niepoprawny login / hasło - niezalogowany

9. Czym się różni metoda GET od POST?

Wszystkie transakcje WWW - a zatem także wysyłanie zawartości formularza - są realizowane przy użyciu protokołu HTTP. W stosunku do formularzy zastosowanie znajdują dwie spośród nich: GET oraz POST.

Podstawowymi różnicami między dwoma sposobami przesyłania danych z formularza są:

Metoda **GET** przesyła dane w adresie strony, po znaku *?*. i przyjmują postać:
http://gdzies.w.sieci/kat/strona.php?imie=Jan&nazwisko=Nowak&plec=M&wiek=35Action wskazuje na adres strony, która się otworzy po wysłaniu formularza. . Najczęściej służy to do podania podstrony serwisu, którą chcemy obejrzeć, wersji językowej lub specjalnej wersji do wydruku. Metoda GET powinna być stosowana tam, gdzie dane odwołanie powinno być adresowalne, to znaczy gdy w URL może być przechowywany pewien stan aplikacji, stały w czasie, do którego użytkownik mógłby chcieć wrócić

Metoda **POST** przesyła dane w sposób niezauważalny dla zwykłego użytkownika. Istnieje w ramach protokołu HTTP i wykorzystywana jest najczęściej do wysłania informacji z formularza znajdującego się na stronie internetowej przesłanych od klienta do serwera. Metoda POST powinna być stosowana tam, gdzie dane odwołanie jest formą interakcji z użytkownikiem, niemożliwą do utrwalenia w formie adresu.

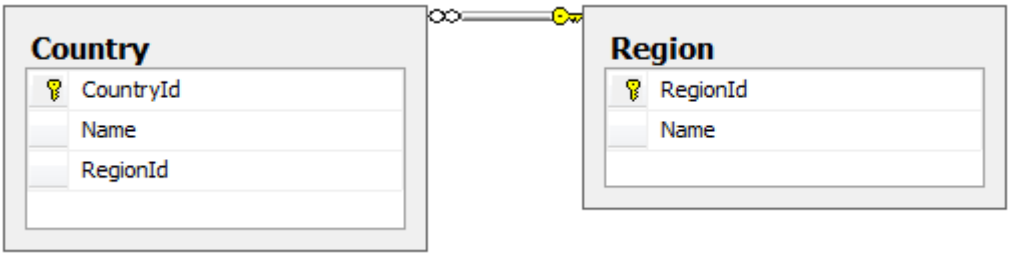
10. Czy HTTP jest protokołem zmiennostanowym?

HTPP zaliczane jest to protokołów bezstanowych- nie zachowuje żadnych informacji o poprzednich transakcjach z klientem (po zakończeniu transakcji wszystko "przepada"). Pozwala to znacznie zmniejszyć obciążenie serwera, jednak jest kłopotliwe w sytuacji, gdy np. trzeba zapamiętać konkretny stan dla użytkownika, który wcześniej łączył się już z serwerem.

11. Czym różni się LEFT JOIN od INNER JOIN?

System między innymi Left Join i Inner Join został stworzony w taki sposób ,aby wyrażać związki pomiędzy rekordami bez dublowania informacji. Na przykaldzie SQL

Tabelka Country zawiera ID , nazwę kraju oraz id regionu. ID regionu może być puste. Dany kraj może nie posiadać regionu.




```
SQL
INSERT INTO [dbo].[Region]
           ([RegionId]
           ,[Name])
     VALUES
           (1,'Europe'),(2,'Asia'),(3,'Africa'),
           (4,'South America'),(5,'North America'),(6,'Australia')

INSERT INTO [dbo].[Country]
           ([CountryId]
           ,[Name]
           ,[RegionId])
     VALUES
           (1,'Poland',1),(2,'Japan',2),
           (3,'Algeria',3),(4,'Argentina',4),
           (5,'New Zealand',NULL),(6,'Federated States of Micronesia',NULL)
```

REGIONY : “5,North America” oraz “6,Australia” nie są powiązane z żadnym krajem.

KAJE : “5,'New Zealand'” i “6,'Federated States of Micronesia'” nie mają swojego regionu.

INNER JOIN zwraca wiersze , w których obie wartość w obydwu tabelkach się zgodziły.

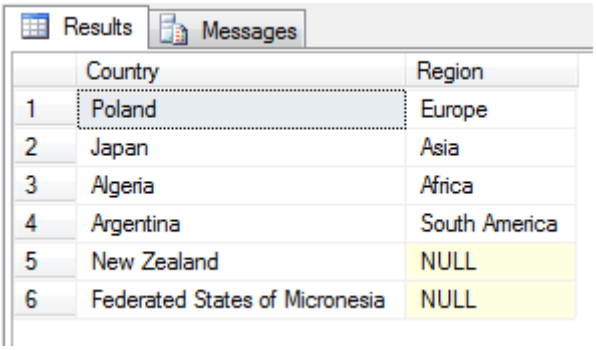
Wyniki tego zapytania powinneń zwrócić tylko kraje powiązane z regionami jak i regiony, które mają przypisane kraje. Rekordy zostały powiązane po RegionId.

Kraje nie mające RegionID zostały zignorowane. Podobny los trafił na regiony, które nie zostały przypisane do żadnego kraju. Np. wpisujemy komendę (SELECT * FROM Table 1t INNER JOIN Table t2 ON t1. Colld = t2.Colld)

Results | Messages

	Country	Region
1	Poland	Europe
2	Japan	Asia
3	Algeria	Africa
4	Argentina	South America

LEFT JOIN : na podstawie przykładu zwraca wszystkie wiersze tabelki po lewej plus wiersze, które mają uzupełnienie z prawej tabelki. Jeśli nie ma żadnych wartości odpowiadającej prawej tabelce jest zwracana wartość NULL. Np. wpisujemy komendę (SELECT * FROM Table t1 LEFT OUTER JOIN Table t2 ON t1.Colld = t2.Colld)



Results | Messages

	Country	Region
1	Poland	Europe
2	Japan	Asia
3	Algeria	Africa
4	Argentina	South America
5	New Zealand	NULL
6	Federated States of Micronesia	NULL

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

a. /boot

b. /var

c. /etc

d. /cfg

Odp. etc

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

Można przetestować komendę bash cp. Sprawdzając trzy warianty. W najbardziej podstawowym użyciu polega na podaniu dwóch argumentów, z czego pierwszy to plik (lub pliki), który ma zostać skopiowany, a drugi to miejsce do którego ma zostać skopiowany. Jeśli drugim argumentem jest nazwa pliku, to pierwszy plik zostanie skopiowany do drugiego (należy pamiętać, że podczas kopiowania nie wyskakuje żadne pytanie, czy na pewno chcemy plik podmienić/nadpisać, jeśli już o takiej samej nazwie istniał wcześniej).

$ cp [opcje] [-T] źródło cel
$ cp [opcje] źródło katalog
$ cp [opcje] -t katalog źródło

Sprawdzamy czy działa polecenie kopiowania plików

joanna@dd: ~/Desktop/test$ ls
kat1 kat2 kat3 plik1
joanna@dd:~/Desktop/test$ cp plik1 plik2
joanna@dd: ~/Desktop/test$ ls
kat1 kat2 kat3 plik1 plik2
joanna@dd:~/Desktop/test$[]
Czy działa kopiowanie plików do katalogu:

joanna@dd: ~/Desktop/test$ cp plik1 plik2 kat1/
joanna@dd: ~/Desktop/test$ cp plik* /home/joanna/Desktop/test/kat2
joanna@dd: ~/Desktop/test$ cp plik* ~/Desktop/test/kat3
joanna@dd: ~/Desktop/test$ ls kat*
kat1:
plik1 plik2
kat2:
plik1 plik2
kat3:
plik1 plik2

joanna@dd:~/Desktop/test$[]

Czy działa katalogu z zawartością: cp -r
joanna@dd: ~/Desktop/test$ cp -r kat1/ kat2/
joanna@dd: ~/Desktop/test$ cp -r plik1/ /home/joanna/Desktop/test/kat3
joanna@dd: ~/Desktop/test$ ls kat2/ kat3/
kat2:
kat1 plik1 plik2
kat3:
kat1 plik1 plik2
joanna@dd:~/Desktop/test$[]

Dodatkowo możemy sprawdzić kopiowanie z potwierdzeniem nadpisania jeśli chcemy mieć pewność że podczas kopiowania nie dojdzie do zastąpienia pliku bez powiadomienia. cp –i (interactive). Dostaniemy komunikat cp : overwrite ‘plik2’? y

joanna@dd: ~/Desktop/test$ cp -i plik1 plik2
cp: overwrite ‘plik2’? y
joanna@dd:~/Desktop/test/kat2$[]