ETAP I

Pytania wprowadzające:

1. *Czym się różnią testy funkcjonalne od niefunkcjonalnych?*

Testami funkcjonalnymi, określanymi też mianem testów czarnej skrzynki, nazywamy testy w których osoba sprawdzająca nie ma dostępudo wnętrza programu, ani kodu. Często do tego rodzaju testów wykorzystuje się osoby nie posiadające wiedzy z programowania. Ich zadaniem jest jak eksploatowanie programu w jak największym stopniu. Testyt te posiadają duża szansę wykrycia błędów wykonania programu.

Testy niefunkcjonalne natomiast dotyczą części programu, która jest niezależna od użytkownika. Sprawdzana jest jakość wykonanego programu, np: bezpieczeństwo, skalowalność czy czas w jakim aplikacja „odpowie" na wykonanie jakiejś czynności. Testy niefunkcjonalne obejmują bardzo szeroki zakres w którego skład wchodzą między innymi:

- Testowanie obciążeniowe i wydajnościowe
- Testowanie przeciążające i obciążenia
- Testowanie ergonomii
- Testowanie współpracy (międzyoperacyjności)
- Testowanie konwersji danych
- Testy bezpieczeństwa / Testy penetracyjne
- Testowanie instalacji
- Testy bezpieczeństwa (zabezpieczeń aplikacji, sieci)

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

Pojęcie smoke testów powstało na potrzeby testów sprzętu. Jak sama nazwa wskazuje, polega on na sprawdzeniu czy po uruchomieniu urządzenia nie ulatnia się z niego dym, iskry bądź inne niepokojące oznaki. W przypadku programów, należy sprawdzić każdą, możliwą część aplikacji, bez zagłębiania się w jej logikę. Jeśli nie zostaną wyrzucone żadne błędy krytyczne, czy bluescreen'y, wiemy że program jest sprawny i responsywny na tyle zeby mogły byc na nim przeprowadzone dalsze testy (np: sanity test lub test regresji).

Celem testów regresji jest zlokalizowanie ewentualnych błędów powstałych po dokonaniu zmian w kodzie programu. Typ tych testów pozwala określić czy na nowo otrzymanej wersji programu możliwe jest dokonanie dalszych testów. Dużą zaletą tych testów jest ich powtarzalność. Dzięki temu łatwo wykorzystać je w jakiejkolwiek automatyzaji, zwiększając przez to ich skuteczność oraz ograniczyć ewentualnie czynnik ludzki. Testy regresywne nie muszą koniecznie dotyczyć zmian wykonanych w samym programie.Mogą równiez odnosić się do tego samego programu używanego na nowszym systemie operacyjnym.

3. *Co jest celem testowania?*

Testowanie ma na celu zlokalizownie błędów w działaniu programu (weryfikację), jak również sprawdzenie funkcjonalności programu w rękach użytkownika (walidację). Głównym założeniem testowania jest sprawdzenie czy dany program spełnia oczekiwania zadane w specyfikacji oraz czy wykonana aplikacja jest przyjazna dla osoby, która będzie używać danego oprogramowania. Testowanie może zostać wdrożone na dowolnym etapie powstawania pogramu, weryfikując konkretne postępy w powstawaniu gotowego produktu.

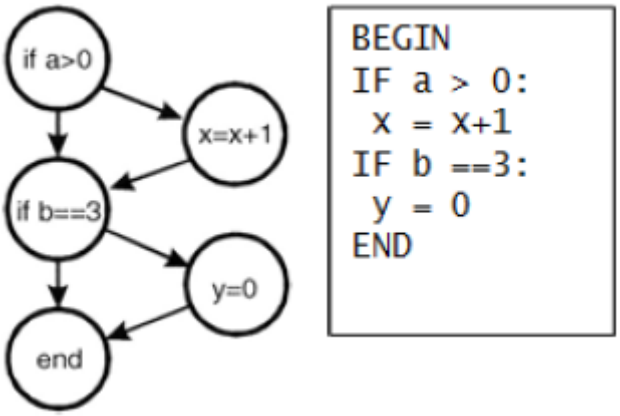
4. *Jak tester może się upewnić, że błąd został naprawiony?*

Tester zgłasza zaistniały błąd leaderowi zespołu. Ten przekazuje swoje uwagi programistom, a nastepnie oni naprawiają zaistniały problem. Gdy tester otrzyma feedback oraz poprawiony kod, ponawia swoje wcześniejsze kroki, dzięki którym uzyskał poprzednie wyniki. Jeżeli output jego działań jest nadal taki sam oznacza to, że błąd nie został należycie połatany. Jeśli jednak wynik jest inny od poprzedniego, należy upewnić się, że jest to oczekiwany rezultat, a nie kolejny feler.

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

Idealnym rozwiązaniem byłoby sprawdzenie aplikacji wraz z urządzeniem w warunkach laboratoryjnych, z góry ustalonych i precyzyjnie wyznaczonych przez aparaturę pomiarową temperaturach. Nieco prostszym rozwiązaniem jest jednak dostanie się do kodu programu i ustawienie, „na sztywno", wartości podawanych przez czujniki. Program powinien uznać, że zostały one zmierzone poprawnie i zacząć działać wedle podanych danych (pomiarów).

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



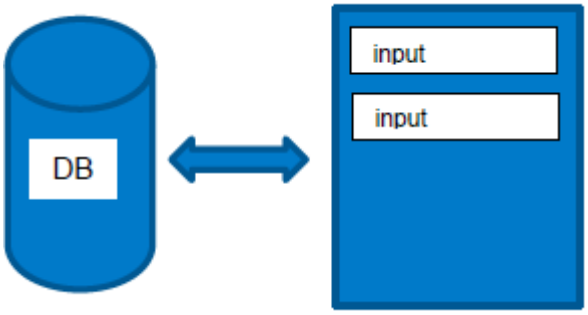
	B=!3	B==3
A>!0	xxx	xxx
A>0	xxx	xxx

Tylko 4 przypadki będą brane pod uwagę. Gdy **A** jest większe od 0 oraz **B** równe i nierówne 3, jak również **A** niewiększe od 0 wraz z tymi samymi warunkami dla **B.**

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?

- Ile użytkowników zawiera DB (data base)?
- Jakiego rodzaju dane są przechowywane w DB?
- Czy są jakieś ograniczenia w sposobie logowania (ilość znaków, duże litery)?
- Czy jest jakaś ograniczona ilość błędnego logowania?
- Sposób odzyskiwania straconych / zapomnianych haseł.

9. *Czym się różni metoda GET od POST?*

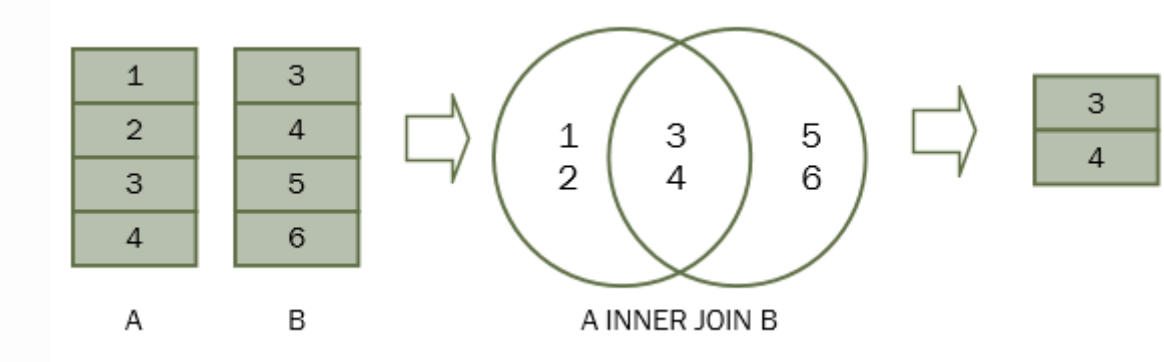
Są dwie metody przekazywania parametrów do innych stron: metoda „GET" i metoda „POST". Metody GET używa się kiedy parametrów jest niewiele. Dzieje się tak ponieważ parametry przekazuje się za pomocą adresu URL (np. http://www.blabla.pl/strona.php?par1=war1&par2=war2), którego długość jest ograniczona. Należy też pamiętać, że parametry są widoczne w pasku adresu przeglądarki, więc tej metody nie należy używać jeśli przekazywane są np. hasła. Jak widać, sam adres od parametrów oddzielany jest za pomocą znaku zapytania, a kolejne pary parametr=wartość za pomocą znaku „&". Tą metodę można wykorzystać także przy przekazywaniu parametrów przez odnośnik (zobacz dalej). Metoda „POST" do przekazywania parametrów wykorzystuje nagłówek zapytania – wystarczy wiedzieć, że metoda ta umożliwia przekazywanie dużo większych parametrów, a także że parametrów nie widać w pasku przeglądarki. Zależnie od metody, zmienne trafiają do odpowiednich tablic asocjacyjnych. Dane przesłane metodą GET trafiają do tablicy $_GET , a dane z metody POST to tablicy $_POST. Tablice $_GET i $_POST są *superglobalne*. Oznacza to, że są widoczne w każdym miejscu kodu PHP bez konieczności użycia składni globals.

10. *Czy HTTP jest protokołem zmiennostanowym*?

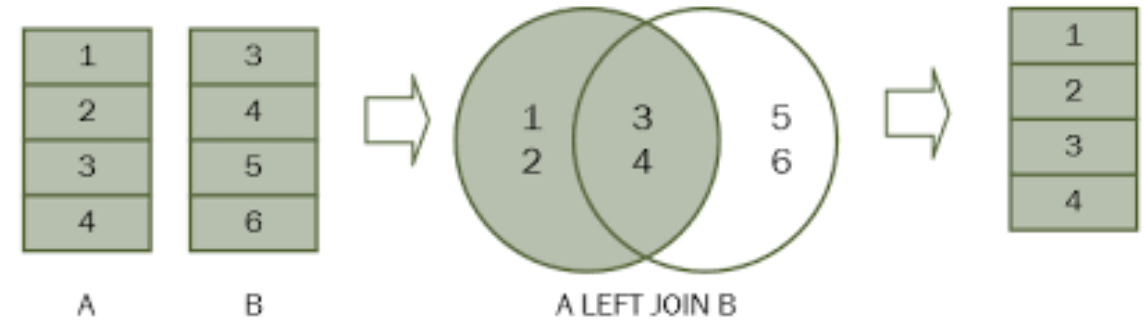
Hypertext Transfer Protocol, znany potocznie jako HTTP jest protokołem bezstanowym. Oznacza to, że ani po stronie serwera, ani po stronie klienta nie są przechowywane żadne informacje dotyczące wcześniejszych komunikacji pomiędzy określonym serwerem i klientem. Dzięki takiemu podejściu każde następne połączenie traktowane jest jako „nowe". Obecnie tą bezstanowość można obejść za pomocą popularnych ostatnio ciasteczek (cookies). Jest to zbiór par parametr=wartość przypisana do konkretnej domeny i przechowywana na komputerze użytkownika w formie pliku txt. Nie zapamiętują one jednakl danych użytkownika. Do tego uzywane są tzw. sesje.

11. *Czym różni się LEFT JOIN od INNER JOIN*?

INNER JOIN jest to funkcja, w wyniku której otrzymujemy złączenie wewnętrzne, którego tabela wynikowa, składa się ze wszystkich kolumn tabel wejściowych. Tabela wynikowa zawierać będzie tylko takie elementy, dla których warunki złączenia wewnętrznego będą spełnione (w logice trójwartościowej, wynik musi być TRUE). Wszystkie elementy dla których wynik nie będzie spełniony (FALSE oraz UNKNOWN), zostaną odrzucone. To jak szukanie wspólnej wartości w obu tabelach.



Łączenie LEFT JOIN działa dokładnie tak samo z jedną małą różnicą. Faworyzuje lewą tabelę i nie pozbywa się doklejanych wierszy jak miało to miejsce w INNER. Uzupełnia natomiast braki za pomocą pustych rekordów (NULL).



12. *W jakim katalogu, standardowo Linux trzyma pliki konfiguracyjne:*
    a. /boot
    b. /var
    c. /etc
    d. /cfg

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

Cp – polecenie służące głównie do kopiowania plików. Pliki mogą być skopiowane w tym samym katalogu, innym katalogu, a nawet innym systemie plików czy dysku twardym. Jeśli plik jest kopiowany do tego samego katalogu, musi mieć inną nazwę, niż plik pierwotny. W innym przypadku nazwa może być taka sama lub zmieniona. Komenda **cp** służy do kopiowania zawartości pliku lub katalogu podanego w parametrach *plik_źródłowy* lub *katalog_źródłowy* do pliku lub katalogu podanego w parametrach *plik_docelowy* lub *katalog_docelowy*. Jeśli *plik_docelowy* istnieje, zostanie on bez ostrzeżenia zastąpiony zawartością pliku źródłowego. Jeśli podanych zostanie więcej parametrów *plik_źródłowy*, parametrem docelowym musi być katalog. Jeśli w katalogu docelowym istnieje plik o takiej samej nazwie, zostanie on zastąpiony przez nowy plik. Aby umieścić kopię pliku *plik_źródłowy* w katalogu, należy w

parametrze *katalog_docelowy* podać ścieżkę do istniejącego katalogu. Podczas kopiowania do innego katalogu nazwy plików nie ulegają zmianie, chyba że użytkownik na końcu ścieżki poda nową nazwę pliku. Komenda **cp** umożliwia również kopiowanie całych katalogów do innych katalogów. Wymaga to podania opcji **-r** lub **-R**. Używając opcji **-R**, można kopiować pliki specjalne urządzeń. Opcja ta powoduje ponowne utworzenie plików specjalnych w nowym katalogu. Podanie opcji **-r** powoduje, że komenda **cp** spróbuje skopiować plik specjalny do pliku zwykłego.

Poniższe przykłady ilustrują użycie komendy **cp**:

- Aby utworzyć kopię pliku w katalogu bieżącym, wpisz:

  cp prog.c prog.bak

  Spowoduje to skopiowanie pliku **prog.c** do **prog.bak**. Jeśli plik **prog.bak** nie istnieje, komenda **cp** utworzy go. Jeśli plik istnieje, komenda **cp** zastąpi go kopią pliku **prog.c**.

- Aby skopiować plik z katalogu bieżącego do innego katalogu, wpisz:

  cp nowak /home/adam/klienci

  Spowoduje to skopiowanie pliku **nowak** do pliku **/home/adam/klienci/nowak**.

- Aby skopiować wszystkie pliki do innego katalogu, wpisz:

  cp /home/aneta/klienci/* /home/adam/kontrahenci

  Spowoduje to skopiowanie samych plików z katalogu **klienci** do katalogu **kontrahenci**.

- Aby skopiować określony zestaw plików do innego katalogu, wpisz:

  cp nowak kowalski milewicz /home/adam/klienci

  Spowoduje to skopiowanie plików **nowak**, **kowalski** i **milewicz** z bieżącego katalogu do katalogu **/home/adam/klienci**.

- Pliki można kopiować z użyciem znaków dopasowywania wzorca:

  cp programy/*.c .

  Spowoduje to skopiowanie plików, których nazwa kończy się na **.c**, z katalogu **programy** do katalogu bieżącego, który oznaczony jest pojedynczą kropką (**.**). Pomiędzy literą **c** a ostatnią kropką wymagany jest odstęp.

Aby sprawdzić działanie funkcji **cp** należy skupić się na pliku wyjściowym tej operacji. Za pomoca odpowiednich narzędzi skupiłbym się na porównaniu parametrów pliku wejściowego i wyjściowego. Pierwszymi krokami byłoby sprawdzenie cech pliku bez zaglądania do niego jak np: rozmiar. Kolejnym punktem na którym skupiłbym się jest sprawdzenie właściwej ścieżki, czy plik skopiował się do poprawnego katalogu, czy posiada odpowiednie rozszerzenie (.c czy .exe). Po upewnieniu się, że wszystko jest w normie zabrałbym się za zaglądnięcie do wnętrza pliku:

- Czy podczas kopiowania żadne dane nie zostały utracone?
- Czy żadne szumy nie wkradły się do pliku?
- Czy inne dane nie zostały wprowadzone do pliku (np: zmieniona jakaś nazwa wewnątrz)
- Czy plik po skopiowaniu nadal zachował swoje właściwości i po odpaleniu nadal wykonuje te same funkcje co pierwowzór?