1. Czym się różnią testy funkcjonalne od niefunkcjonalnych?

Testów funkcjonalnych użyjemy jeśli zależy nam na przetestowaniu jakiejś funkcji wykonywanej przez oprogramowanie. Natomiast jeśli będziemy chcieli np. przetestować niezawodność, użyteczność systemu (czyli atrybuty jakościowe) zastosujemy wówczas testowanie niefunkcjonalne.

Testowanie funkcjonalne oparte jest na analizie specyfikacji funkcjonalnej modułu lub systemu, specyfikacji wymagań, przypadkach użycia.

Natomiast **testowanie niefunkcjonalne** to testowanie atrybutów modułu lub systemu, które nie odnoszą się do jego funkcjonalności np. niezawodności, efektywności, pielęgnowalności i przenaszalności. Podczas testów niefunkcjonalnych testowane są **cechy jakościowe** części lub całości systemu. Brane są pod uwagę aspekty oprogramowania, które nie są związane z określoną funkcją lub działaniem użytkownika, takie jak skalowalność, bezpieczeństwo czy czas odpowiedzi aplikacji np. ile osób może się zalogować w jednym czasie? Testy niefunkcjonalne są wykonywane na wszystkich poziomach testów.

Testowanie funkcjonalne polega na sprawdzeniu „co" system robi, natomiast niefunkcjonalne „jak" system działa.

Testy funkcjonalne odpowiadają na pytanie jak system wykonuje funkcje, które ma za zadanie udostępniać włączając w to komendy użytkownika, operacje na danych, wyszukiwanie, procesy biznesowe i ekrany użytkownika. Testy funkcjonalne pokrywają zarówno funkcje dostępne dla użytkownika poprzez interfejs aplikacji jak i operacje typu back-end.

Testy funkcjonalne dotyczą funkcji lub innych cech oraz współdziałania z innymi systemami. Natomiast **testy niefunkcjonalne** to testy wymagane do zmierzenia właściwości systemów i oprogramowania, mogące zostać określone ilościowo na zmiennej skali, takich jak czasy odpowiedzi w testach wydajnościowych. Mogą się odnosić do modelu jakości oprogramowania.

W testach funkcjonalnych (testy czarnej skrzynki lub black box testing) tester nie ma dostępu do kodu testowanej aplikacji. Testy wykonywane są przez osobę, która nie tworzyła aplikacji w oparciu o dokumentację oraz założenia funkcjonalne. W zakres testów metodą czarnej skrzynki wchodzi badanie wartości brzegowych oraz definiowanie klas równoważności.

W tym rodzaju testów oprogramowanie jest sprawdzane pod kątem wymagań funkcjonalnych. Przypadki testowe pisane są pod kątem sprawdzenia czy aplikacja zachowuje się w oczekiwany sposób.

Testowanie funkcjonalne obejmuje:

- Testy jednostkowe
- Testy dymne / testy kondycji
- Testowanie integracyjne (testowanie zstępujące, testowanie wstępujące)
- Testy interfejsu i użyteczności
- Testy systemowe
- Testy regresji

- Testy Alfa i Beta
- Testy akceptacyjne użytkownika
- Testowanie metodami białej skrzynki oraz metodą czarnej skrzynki

Testowanie niefunkcjonalne obejmuje:

- 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?

Smoke test (test dymny)

Test dymny (smoke test) to 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 test wywodzi się z testowania sprzętu – podłączamy nowe urządzenie i sprawdzamy czy nie… wybuchnie. Jeśli pojawi się dym lub ogień wtedy wiemy, że coś poszło nie tak. Nazwa określa po prostu szybkie testy które w jakimś stopniu potrafią sprawdzić czy aplikacja działa.

Somke testy mogą być wykonywane na podstawie istniejących przypadków testowych lub przy użyciu narzędzi do testów automatycznych.

Przeznaczeniem smoke testów jest “dotknięcie” każdej z części aplikacji bez zagłębiania się w logikę jej działania. **Chodzi o upewnienie się, czy najbardziej krytyczne funkcje działają.**

Smoke test mówi nam, czy program/system da się uruchomić, czy jego interfejsy są dostępne i czy reagują na działania użytkownika. Jeżeli smoke test nie powiedzie się nie ma powodu aby przechodzić do dalszych testów.

Cechy smoke test ("testu dymnego") to:

- **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).

Kiedy stosujemy smoke test?

Smoke test przeprowadzany jest przez programistów tuż przed oddaniem wersji aplikacji lub przez testerów, przed zaakceptowaniem otrzymanej do testów aplikacji.

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 (testowanie regresywne)

Jest to ponowne przetestowanie uprzednio testowanego programu po dokonaniu w nim modyfikacji, w celu upewnienia się, że w wyniku zmian nie powstały nowe defekty lub nie ujawniły się defekty w niezmienionej części oprogramowania. Czyli upewniamy się, że aplikacja działa po dokonaniu w niej modyfikacji, poprawieniu błędów lub po dodaniu nowej funkcjonalności.

Co dają testy regresji?:

- 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.

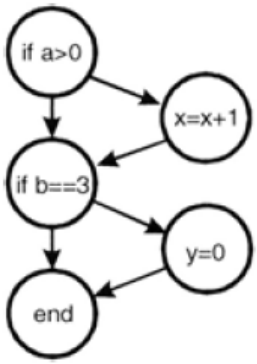
Kiedy stosujemy testowanie regresywne?

Testy regresyjne mogą zostać wykonane na wszystkich poziomach testowych i zajmują się funkcjonalnością, zdolnościami niefunkcjonalnymi i testowaniem strukturalnym.
Testy regresywne mogą być przeprowadzane dla kompletnego produktu lub jego części. Testy te są przeprowadzane po zmianie oprogramowania lub jego środowiska. Testy regresywne powinny być przeprowadzane po smoke testach lub testach typu sanity.

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

Aby pokryć wszystkie możliwości potrzeba 4 przypadków testowych – aby pokryć wszystkie 4 ścieżki.

```
BEGIN
IF a > 0:
 x = x+1
IF b ==3:
 y = 0
END
```



4. Co to jest testowanie zwinne?

Testowanie zwinne jest to metoda testowania stosowana w projektach korzystających z metodologii zwinnych takich jak programowanie ekstremalne (XP), traktujące wytwarzanie jako klienta testowania i kładąca nacisk na metodę „najpierw przygotuj testy". Jest to związane z wytwarzaniem sterowanym testami : czyli sposobem wytwarzania oprogramowania, w którym przypadki testowe są przygotowywane i często automatyzowane zanim powstanie oprogramowanie, które będzie testowane za ich pomocą.

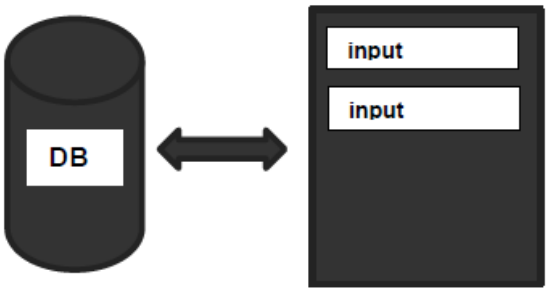
5. Dany jest input „wiek", który przyjmuje wartości od 18 do 60. Twoim zadaniem jest przetestować go za pomocą techniki wartości brzegowych. Jakie wartości wpisujesz do inputu, podaj wszystkie liczby, które wpisujesz.

Wiek:	17,18,19,59,60,61

Wartość brzegowa to wartość znajdująca się wewnątrz, pomiędzy lub tuż przy granicy danej klasy równoważności. Istnieje duże prawdopodobieństwo, że oprogramowanie będzie się błędnie zachowywać dla wartości na krawędziach klas równoważności. Wartości brzegowe to na ogół minimum i maksimum (o ile istnieją) klasy równoważności.

Ponieważ minimum i maximum klas równoważności to liczby 18 i 60, powinniśmy przetestować wiek dla wartości 17 oraz 61. W praktyce testujemy także dla wartości , znajdujących się pomiędzy wartościami 18 i 60 czyli dla 19 i 59.

6. 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?

Pytania:

a) Na jaką platformę stworzona jest aplikacja?
b) Jaka przeglądarka obsługuje tą aplikację?
c) Czy transmisja jest szyfrowana i jaką metodą?
d) Jakie warunki musi spełniać login?
e) Jakie warunki musi spełniać hasło?
f) Czy jest opcja automatycznego wylogowywana? Jeśli tak to po jakim czasie?
g) Czy jest możliwość łączenia się z bazą danych za pomocą różnych protokołów sieciowych (ftp, http itd.)?
h) Po jakiej stronie są przechowywane dane podczas pracy z tą aplikacją (klienta czy serwera)?

7. Co jest celem testowania?

Cele testowania:

- znajdowanie usterek
- nabieranie zaufania do poziomu jakości
- dostarczanie informacji potrzebnych do podejmowania decyzji
- zapobieganie defektom

Przykłady:

- W testowaniu wytwórczym (np. modułowym, integracyjnym lub systemowym) głównym celem może być wywołanie wielu awarii, aby zidentyfikować i poprawić usterki występujące w oprogramowaniu.

- W testach akceptacyjnych celem może być potwierdzenie, że system działa tak jak powinien oraz nabranie pewności, że spełnia wymagania.

- Celem może być też ocena jakości oprogramowania, by dostarczyć klientom informacji o ryzyku związanym z wydaniem systemu w danej chwili.

- Testowanie pielęgnacyjne sprawdza, czy nie wprowadzono nowych usterek podczas wykonywania zmian.

- W testowaniu produkcyjnym celem może być ocena atrybutów systemu np. niezawodności, dostępności.

Zadanie praktyczne:

Zgłoszenie błędu w MANTIS

ID: 0000091