STLC (Yazilim Testi Yasam Dongusu) - 30 Temmuz 2022 Cumartesi

Tester urunun, bellli kriterlere bagli olarak musterinin urunle ilgili kriterlerini karsilayip karsilamadigini kontrol eder.

Yazilim Testi Nicin Onemlidir:
- Yazilimin guvenilirligini kontrol etmek.
- Yazilim testi arizaya neden olabilecek herhangi bir hata icermemesini saglar.
- Yazilim bir Dev ekibi tarafindan olusturulur. Sifir hata ile yazilim olusturulmaz. Testing mutlaka donguye dahil edilmelidir.
- Test yazilimin tam olarak gereksinmlerde belirtildigi gibi davranmasini ve calismasini saglar.
- Ihtiyaca (requirement) uygun olmayan her sey bir hatadir.
- **QA gorevi yazilimin kalitesini arttirmaktir.**

YAZILIM KALITE GUVENCESI
Tum yazilim gelistirme surecidir.

STLC ASAMALARI:

Her sprintte tekrar eden asamalardir.

Gereksinimlerin Analizi : Testin baslangici gereksinimlerdir... FRD ile gelir > BA hazirlar...
Test Planinin Hazirlanmasi :
Test Case Hazirlanmasi:
Test ortamlarinin Hazirlanmasi: (environment Setup) INTERVIEW
Testin Kosulmasi (Test Execution) INTERVIEW
Test Isleminin Tamamlanmasi.

1) Requirement Analysis (Gerekesinimlerin Analizi)
   Sadece USER STORY de aciklanan yazilim gereksinimleri gozden gecririz.
   Icecek Istiyorum:
   -Icecegimde semsiye istiyorum.
   -1 dilim limon istiyorum
   -Buyuk bir bardakta istiyorum
   -Icecek soguk olmali
   -Kolay icebilmek icin bir pipet istiyorum.
2) Test Planing (Test Planlamasi)
   - Test Plani Belgesi , Urun Aciklamasi, Yazilim gereksinimi Spesifikasyonu (Software Requirement Specifications SRS) veya kullanim senaryosu (Use Case Documents) vb belgelerden turetilir.
   – tes faaliyetlerinin kapsam, kaynaklar ve programini aciklar..
3) TEST CASE Development
   – Tester`larin olusturdugu bir dokumandir... !!!!
   – Bir yazilimin beklendigi gibi davrandigini dogrulamak icin olusturulan talimatlar listesi diyebiliriz.

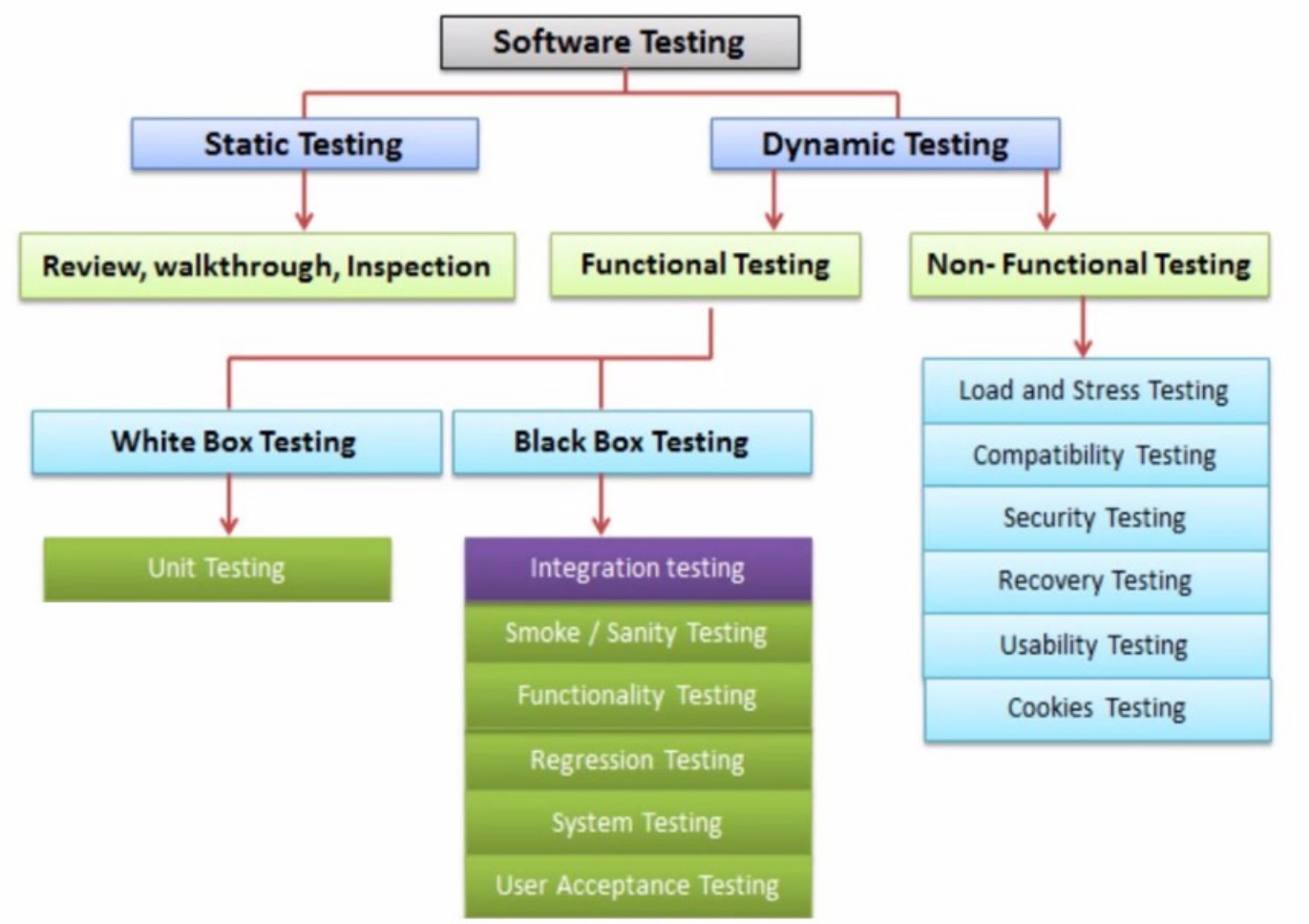
Test Senaryosu Olusturmak
Test Case No. - Kesin Olmalidir.
Priority - Oncelik
Test Name – Kesin olmalidir.
Test Step – Kesin Olmalidir.
Result – Beklenen Sonuc. Kesin olmalidir.
Status -
Test Datasi -
.

4) TEST ENVIRONMENT SETUP (Test Ortami Kurulumu)

testimizi nerede ve hangi ortamda gerceklestirecegimizi bilmeliyiz ve test verilerine sahip olmaliyiz.

5) Test Execution (Testin Uygulanmasi)

ÇOK ÇOK ONEMLI



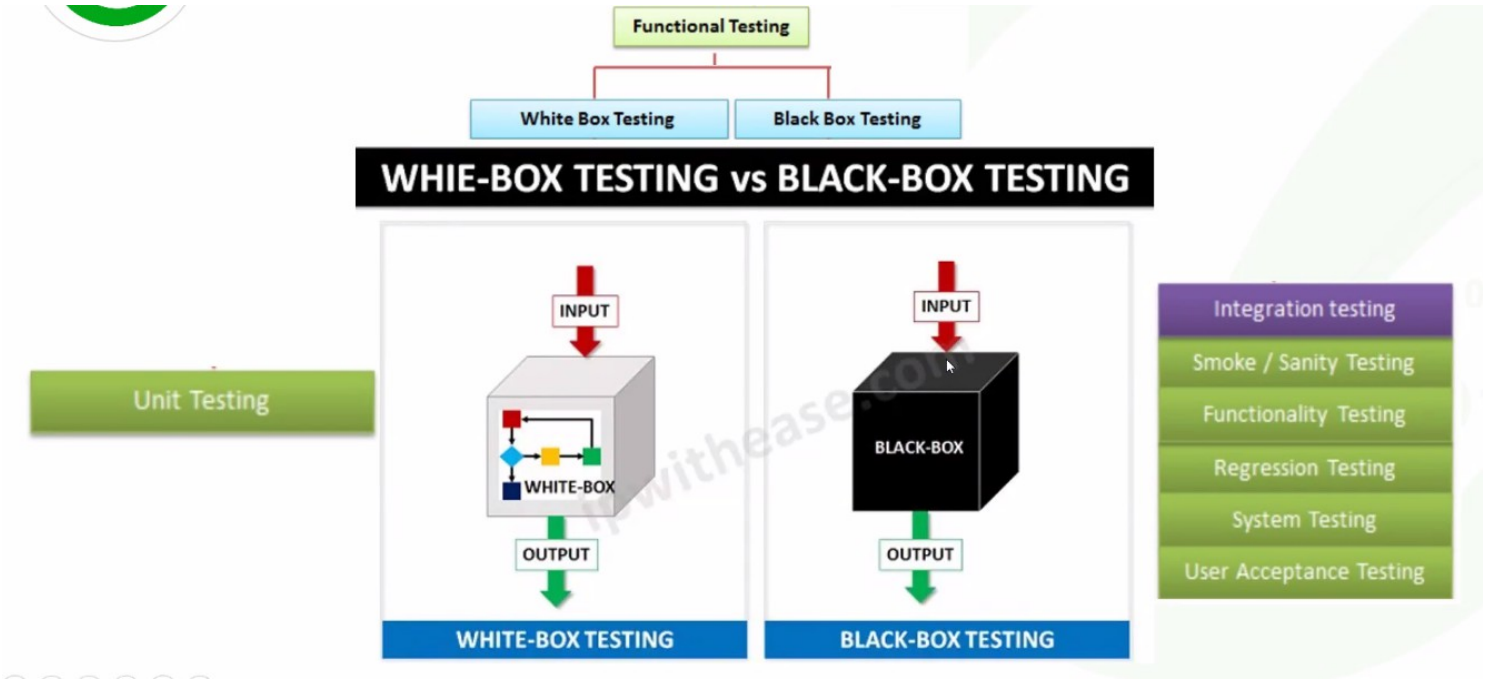
Static Test...:
Statik test kod yurutulmeden kodun veya diger proje dokumanlarinin manual olarak gozden gecirilmesidir.
Statik testler dinamik testlere gecilmeden once yapilmalidir.
Projenin baslarinda gozden gecirme yoluyla tespit edilen hatalarin cozulmesi ilerleyen asamalarda bulunmasindan daha az maliyetlidir.

Dinamk Testing:
Kodun butununu cesitli yontemlerle test etmeye yarayan metottur. Bulunan tum hatalar cozulmeden ve testin sonlandirma kriterleri saglanmadan sona ermez. Test edilecek yazilimin turune gore, uygulanma metotlari farklilik gosterebilir.

Functional Test: Uygulamanin ana islevlerinin test edilmesidir. Trandyol testi hem blackbox hem de functional testtir.

Non-Functional Test: Islevsel olmayan ozellikler test edilir. Ornegin sistemi ayni anda kac kullanici kullanabiliyor? Sistem yeterince guvenli midir? Gibi sorularin cevabini aliriz.

Functional Test 2ye ayrilir..



WhiteBox Test: (Saydam Kutu Testi) : icerikteki kodlarin gorundugu testlerdir.
Developerlarin yaptigi testtir. **Unit test** olarak da bilinir.

1. Kod mutlaka gorunmelidir.
2. Gereksiz kodlarin ayiklanmasi bu asamada olur.
3. Black box testlerinin kolaylasmasini saglar.
4. Kodun Yapisi (Structure) , Tasarimi(design) , Uygulanisi(Implementation) ile ilgili testtir.

BlackBox Testing : trandyol uzerinde yapilan test BlackBox testtir. Kodu gormeden yapilan test.
Kodun Yapisi (Structure) , Tasarimi(design) , Uygulanisi(Implementation) ile ilgilenmez.
\ Girdi cikti degisimlerine gore nasil calsitigi test edilir.
BlackBox test teknikleri QA`ler tarafindan yaygin kullanilan Test Cesitleridir.

UNIT TEST: Developerlarin bir yazilimin tamamlanmis her unit`ini bolumunu test etmesidir.
Unit testler bittikce bir araya gelen her unit birlestirilip QA`ye gonderilir.
Ihtiyac durumunda Unit Test`i QA de yapar.
Entegrasyon Testi , Dizi Testi veya İş Parçacığı testi olarak tanimlanir.

Integration Test:
Her modulu test ettikten sonra, entegre modullerin istenen ciktryi verdiginden emin olmak icin bir araya getirilir ve test edilir.
Tipik olarak her yazilim, farkli yazilim gelistiriciler tarafindan kodlanmis farkli yazilim modullerinden olusur.
Bu test her bir modulun herhangi bir kusur olmadan mukemmel bir sekilde etkilesime girmesini saglamayi amaclar.
Bu modullerin verileri arasindaki iletisime odaklanir.
Entegrasyon testi, dizi testi ve is parcacigi testi olarak da bilinir.
Birim testi sirasinda gozden kacmis olabilecek kusurlari, birim testi yapildiktan sonra istemcilerin

gereksinimlerinde meydana gelen degisiklikler gibi faktorlerden kaynaklanabilecek kusurlari, harici donanim arayuzlerinden kaynaklanan hatalari, yazilimlar arasindaki etkilesimlerden kaynaklanan hatalari ortadan kaldirmak icin entegrasyon testi gereklidir.

Smoke Test (Duman Testi): Kritik fonsiyonlarin testidir.
Duman testi projede en onemli islevlerin calismasini saglamayi amaclayan kapsamli olmayan bir test turudur.
Temel sistemin ayni kalmasi icin tum sistem birlikte test edilmelidir.
Bu testte onemli olan buyuk resmi gormektir.
Amac uygulamanin teste devam edecek duzeye gelip gelmedigini kontrol etmektir.
Duman testlerini evreleme ve uretim ortamlarinda erken ve sik sik tekrarlamak gerekmektedir.
Hergun yapilmasi gereken rutin testlerdir.

SANITY TEST:
Uygulama uzerinde kucuk bir hata giderildiginde veya kucuk degisiklikler ardindan yapilan testlerdir. Sanity testi test surecinde uygulamanin bir iki islevine odaklanirken duman testi tum onemli islevlerin calistigindan emin olmak icin yapilir. Bu test sayesinde hatalar daha erken bir asamada kesfedilir.
Bulunan bir hata sonucu, hata düzeltildikten sonra hatanın giderilip giderilmediğini anlamak için yapılan bölgesel küçük test.

REGRESSION TEST:
Bu test turunde yazilim uzerinde yapilan degisiklik sonucunda uygulama kodunun bozulup bozulmadigini dogrulamak icin yapilan test turudur. Degisiklik sonrasinda mevcut ozelliklerin calisip calismadigini kontrol etmek icin yapilan test turudur.
Projemiz ilerlerken, eklenen bir modül sonrası yapılan, eklediğimiz modülün sisteme uyup uymadığı veya sorun çıkartıp çıkartmadığını ölçmemiz için yapılan test. 2YE AYRILIR:

1. **Kismi Regresyon**, kodda degisiklikler yapildiginda ve bu birimin degismemis veya halihazirda mevcut olan kodla entegre edildiginde bile kodun iyi calistigin dogrulamak icin yapilir.
   Sprint sonlarinda yapilan bir testtir.
   Bir module yapilan eklemeler olursa, **o modulun** onceki kodlarinin bu eklemeyle baglantili olarak dogru calistiginin testidir.

2. **Tam Regresyon** ; kodda bir dizi modulde bir degisiklik yapildiginda ve ayrica baska herhangi bir moduldeki bir degisikligin etkisinin belirsiz olmasi durumunda yapilir. Degistirilen kod nedeniyle herhangi bir degisikligi kontrol etmek icin urun bir butun olarak test edilir.
   2-3 ayda bir yapilan bir testtir. Sure sarti tabu degildir. Ne zaman istenirse yapilabilir.
   Bir ekleme yapildiktan sonra **butun modullerin** tam anlamiyla calismasinin tes edilmesidir.

SISTEM TESTI: (E2E - End To End Test)
- Sistem Testi, yazilim gereksinim testlerinin tamamlanmasindan sonra, sistem gereksinimlerine gore olusturulan testleri kapsar.
-
- Yazilim tarafinda yapilan Birim Testi(Unit Test) ve Entegrasyon Testi(Integration Test) adimlarindan sonra yapilan sistem testleri, daha cok islevi tamamlanan yazilimin, guvenlik, guvenilirlik, performans gibi faktorler altinda yapilan test islemlerini kapsar.

- Sistem testinin amaci, bir uygulamanin tasarlandigi gibi islevleri gerceklestirirken dogrulugunu ve eksiksizligini dogrulamaktir.
- Gercek sonuclar ve beklenen sonuclar siraya girdiginde veya farkliliklar, musteri girdisine gore aciklanabilir veya kabul edilebilir oldugunda sistem testi tamamlanmis olarak kabul edilir.

- Ayni fonsiyonunu fakli arayuz ve platformlarda ayni sekilde test edilmesidir.
  Mesela trendyol giris ekraninin API, SQL, Selenium v.b. Platformlarda farkli testlere tabi tutulmasidir.

Iki sekilde ifade edilebilir:

1) Fakli fonsiyonlarin birlestigi uctan uca senaryolar.
2) Ayni modulun backend ve UI tarafindan ele alindigi senaryolar.

AD-Hoc Testing (Maymun Testi)

User Acceptance Testing UAT (Kullanici Kabul Testi)

Bu testtin amaci yazilimi is gereksinimlerine gore dogrulamaktir.
Amac: yazilimin kullanici ve musteri tarafindan kabul edilip edilmeyecegini belirlemek icin test edilmesine olanak saglamaktir.
Bu dogrulamalar is gereksinimlerine asina olan son kullanicilar tarafindan gerceklestirilir.
Fonksiyonel, sistem ve regresyon testleri gerceklestirildikten sonra kullanici kabul testleri gerceklestirilir.
Yazilim yayinlanmadan gerceklestirilen son testtir.
Beta testi olarak ta adlandirilir.