YAZILIM TESTİ

Zeliha Öznük

Yazılım testi Neden Önemlidir?

- Günümüzde, teknolojinin sürekli gelişmesiyle yazılım sektörü de büyük bir atılım göstermeye ve kendi içinde yeni dallar oluşturmaya başladı. Daha doğrusu teknoloji artık yazılım sektörüne endeksli olarak büyüyor da diyebiliriz. Sektördeki bu büyümenin rekabeti de arttırdığını söylemek hiç de zor değil.

- Yazılım sektöründe firmalar hızlı bir şekilde değer kazanabildiği gibi aynı şekilde piyasadan kısa sürede çekilmek zorunda kalabiliyor. Rekabetin ve tüketimin bu kadar yüksek olduğu bir ortamda şirketlerin ayakta kalabilmek için ürün kalitelerini koruyup memnuniyet oranlarını sürekli yüksek tutmaları gerekiyor.

- Ürün kalitesinden bahsettiğimiz anda da ürünün hatadan arındırılmış olması önkoşul olarak karşımıza çıkıyor. Bu önemli aşamada yazılım test mühendislerine de büyük işler düşüyor. Çünkü bir şirketin ürünü güncel ve kaliteli tutabilmesi onu sürekli profesyonel bir şekilde test etmesinden geçiyor.

- Yazılım test mühendisliği neden önemlidir sorusunu birkaç maddede böylece açıklayabiliriz.

Test, hataları ve kusurları tespit eder

- Test, söz konusu yazılım programını kullanırken ortaya çıkabilecek tüm olası sorunları belirlemeye yardımcı olur. Örneğin, bir simülasyon, müşteri memnuniyetsizliğine ve maliyetli rahatsızlıklara yol açabilecek kullanıcı arayüzündeki kusurları ortaya çıkarabilir. Bu tür sorunlar, kontrol edilmediği takdirde bu yazılım programına güvenen kuruluşta onarılamaz hasara neden olabilir.

Yazılım testi, yüksek ürün maliyetinden ve mali zarardan korur.

- Bu maddeyi bir örnekle açıklayabiliriz. Şirketinizin yüksek kullanıcılı bir yazılım ürününe sahip olduğunu varsayalım. Uygulamanıza yeni bir özellik (güncelleme) sağlıyorsunuz. Bu güncellemeyle birlikte sistemde bir hata (bug) oluşuyor ve binlerce kullanıcınız bu durumdan ciddi bir şekilde etkileniyor. Etkilenen kullanıcılarınız müşteriniz olmayı bırakıp aynı özelliklere sahip yeni bir uygulama kullanmaya başlıyor. Bu varsayım gerçekleşmiş olsaydı şirketinizde çok ciddi mali kayıplara sebep olacaktı.
- Buna ek olarak muhtemelen o müşterileri geri kazanmak şirketiniz için çok zor olacağı gibi sektörde bıraktığınız kötü izlenim yeni müşteriler kazanmanızı da zorlaştıracaktı.
- Diğer açıdan bakacak olsaydık eğer, yani yaptığınız güncellemeyi son müşteriye sunmadan önce profesyonel bir yazılım test ekibi test etmiş olsaydı, muhtemelen hata çok daha erken fark edilerek düzeltilmesi için ilgili yazılım ekibine tekrar yönlendirilecekti. Hata çözülmüş bir şekilde yapılan güncelleme sayesinde ise müşteri kayıpları yaşanmayacaktı. Böylece yüksek zararlardan korunmuş olunacaktı.

Üründe oluşabilecek güvenlik açıklarını minimuma indirir.

- En önemli noktalardan bir tanesi elbette yazılım ürününün güvenliğidir.
- Kişisel ve finansal bilgilerin (kredi kartı vs.) kullanılıp saklandığı uygulamalar güvenlik açısından çok dikkat edilmesi gereken uygulamalardır. Bu tarz uygulamalarda kullanıcı, bilgilerinin güvenli bir şekilde saklandığından emin olmak ister. Çünkü güvenlik sistemlerindeki küçük açıklar bile hackerlar tarafından kötüye kullanılabilmektedir.
- Özellikle bankacılık uygulamaları hackerların bu açıkları kullanarak büyük finansal zararlara sebebiyet verdikleri alanlardır. Bu nedenle bu uygulamaların güvenliği her zaman profesyonel test ekipleri tarafından test edilmelidir.
- Bazı durumlarda da kötü niyetli yazılım mühendisleri tarafından uygulamalarda arka kapı diye tabir ettiğimiz bilinçli açıklar bırakılabilmektedir. Böyle durumların önüne geçebilmek için de yazılım ekibinden bağımsız olarak çalışan test mühendisleri ile çalışılmalıdır.

Yazılım testi ürün kalitesini arttırır.

- Ürününüzün kalitesini istediğiniz düzeye çekebilmeniz ve piyasaya planladığınız kalitede sunabilmeniz için yazılım testi büyük önem arz etmektedir.
- Bu konuya da şöyle bir örnek verebiliriz. Şirketinizin geliştirmiş olduğu uygulama çok sayıda müşteri tarafından kullanılabileceği için birçok farklı tarayıcıdan giriş sağlanabilir. Fakat uygulama her tarayıcıda istenildiği gibi çalışmayabilir ve uyumluluk sorunları yaşanabilir. Bu durumda ürününüzün kalitesi beklenenden düşük demektir. Sorun yaşayan kullanıcılarınız ürün kalitesini beğenmediği takdirde müşteriniz olmayı reddedecektir.
- Yazılım test mühendisleri yine bu noktada devreye girerek oluşabilecek sorunları engelleyebilmektedir. Çünkü profesyonel yazılım ekiplerinde, uygulamaları farklı tarayıcılar üzerinden çeşitli test araçları kullanarak yüzlerce hatta binlerce senaryo ile test etme imkanı vardır. Bu kapsamda gerçekleştirilen testler ile olası hatalar kolayca bulunabilir ve ortaya çıkabilecek hata sayısı minimuma düşürülüp maksimum kalite sağlanabilir.

Müşteri memnuniyetini sağlar.

- Şirket sahipleri için asıl amaç her zaman maksimum müşteri memnuniyetine ulaşmaktır. Çünkü müşteri memnuniyeti sektörde sürekliliği sağlar. Müşteri memnuniyeti yüksek ürünlerin reklamı kulaktan kulağa hızlı bir biçimde yayılabilmektedir. Memnuniyeti kazanmak ise gerçekten çok zordur, hele ki ilk izlenimde kullanıcı tarafından uygulama beğenilmediyse! Memnuniyeti arttırmak için ise ürünün müşterinin bakış açısıyla test edilmesi gerekir. Bu nedenle, genellikle şirkette üründen sorumlu iş birimi tarafından, UAT(Kullanıcı Kabul Testi) adı verilen testler uygulanarak, ürünün son kullanıcıya sunumundan önce son kullanıcı gözüyle işlevselliği doğrulanmalıdır. Bu testin başarılı olması halinde ürün pazara sunulur. Bu sayede ilk izlenimde oluşacak kötü durumlar engellenmiş ve yüksek müşteri memnuniyeti sağlanmış olacaktır. Ayrıca yapılan performans testleri, ürünün farklı koşullarda verdiği tepkiyi ölçümleyerek geliştirme aşamasında gerekli önlemlerin alınmasını sağlar. Ürünün bu testlerden başarıyla geçmesinden sonra pazara sunulması ile müşteri memnuniyetini olumsuz etkileyebilecek durumların önüne geçilebilmektedir.

- Maksimum müşteri memnuniyetine ulaşmak için yapılan testler uzun vadede şirkete büyük kazançlar sağlayacaktır.

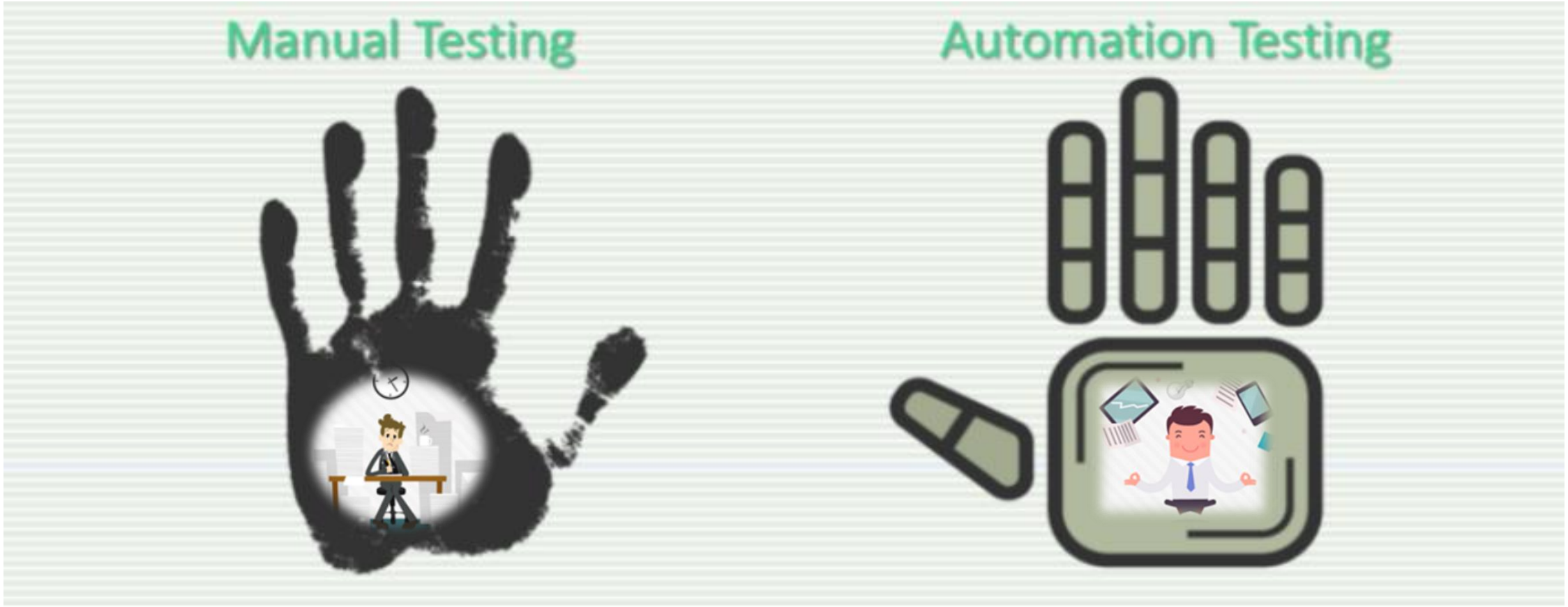
Yazılım Test Süreci Aşamaları nelerdir?

- 1- Testin planlanması
- 2- Test tasarımı yapılması
- 3- Testin gerçekleştirilmesi
- 4- Hata raporlama yapılması
- 5- Test sonuç raporları oluşturulması ve paylaşılması

Yazılım Testlerinin 7 Prensibi

- 1) Testler, hataların yokluğunu değil varlığını gösterir.
- 2) Detaylı ve %100 test etmek imkansızdır.
- 3) Erken aşamada test yapmak zaman ve paradan tasarruf etmeyi sağlar.
- 4) Uygulamadaki hataların çoğunluğu sadece belli yerlerde kümelenir.
- 5) Zehirlenme (veya antibiyotik) paradoksu.
- 6) Test, içeriğe göre değişkenlik gösterir.
- 7) Hatasızlık bir yanılgıdır.

Manuel Test ve Test Otomasyonu

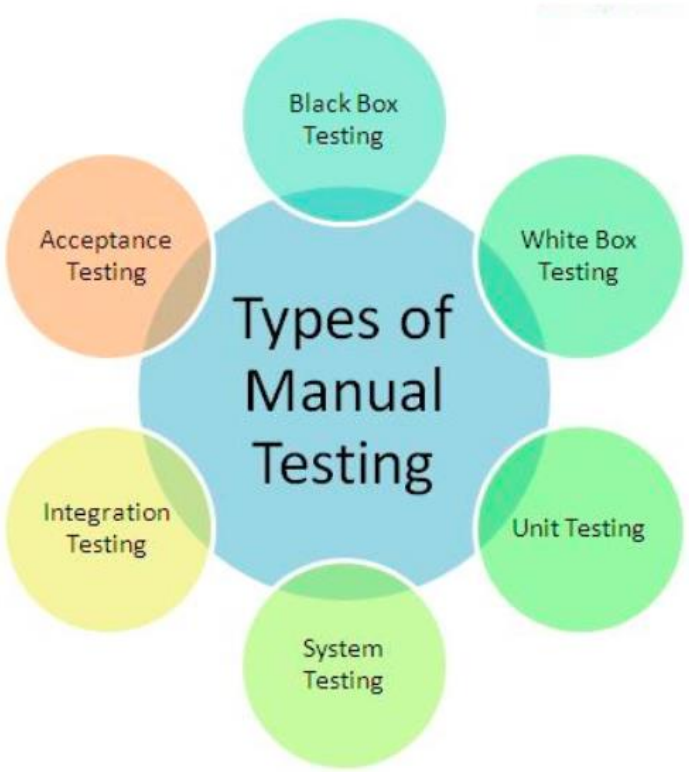


Manuel Test

- Manuel test, diğer ismiyle fonksiyonel test isminden de anlaşılacağı üzere test mühendisinin kendini son kullanıcının yerine koyarak bütün fonksiyonaliteyi test etmesi demektir. Test senaryolarını otomatik bir araç kullanmadan manuel olarak yürütür. Tüm senaryolar ve durumlar test cihazı tarafından son kullanıcı düşünülerek yapılır.

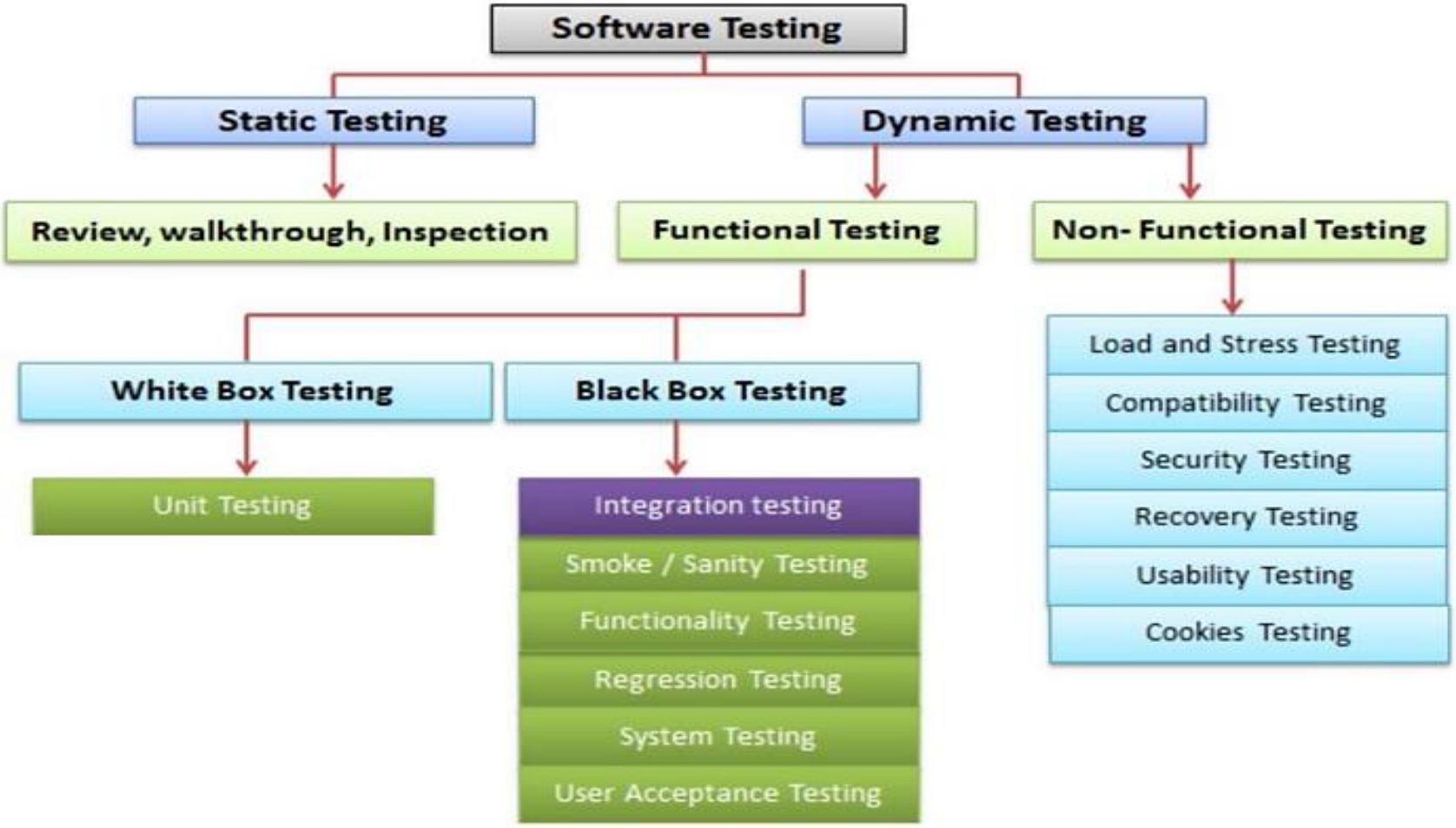
- Butonlar doğru çalışıyor mu? Kullanıcı bu özelliği doğru bir şekilde kullanabiliyor mu? Bu özellik işe yarıyor mu? gibi soruların cevabını arar. Manuel test hem görünür hem de görünür olmayan hataları yakalamaya olanak sağlar. Geliştirilen her yeni özellik için manuel test zorunludur. Manuel testin başarılı olmasının en önemli kuralı, yazılımın gereksinimlerinin ve hedeflerinin testi yapan kişi tarafından kusursuz bir şekilde bilinmesidir.

Manuel Test Teknikleri;



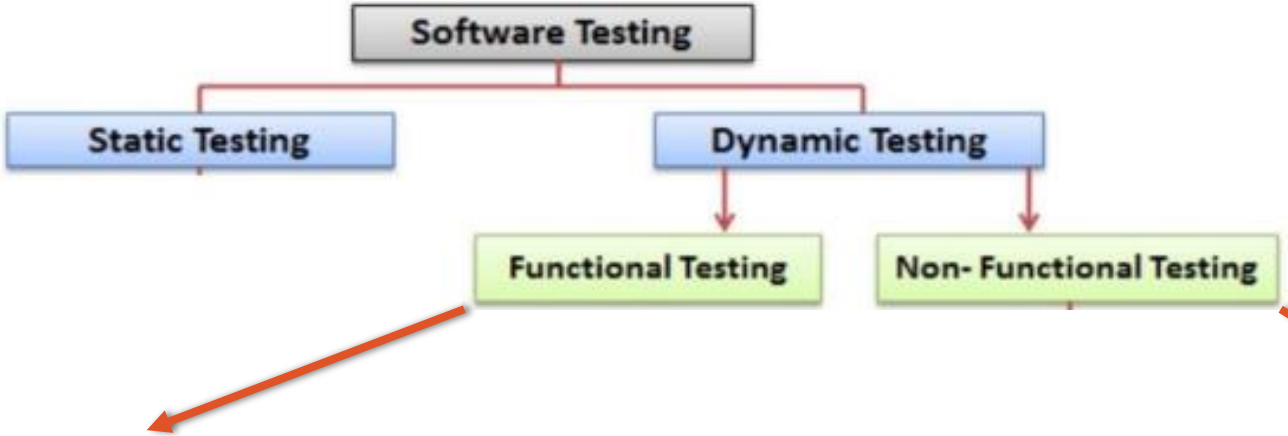
- · Beyaz kutu (WhiteBox) Testi
- · Kara kutu (BlackBox) Testi
- · Kabul (Acceptence) Testi
- · Birim (Unit) Testi
- · Sistem (System) Testi
- · Entegrasyon (Integration) Testi

Yazılım Test Teknikleri Nelerdir?



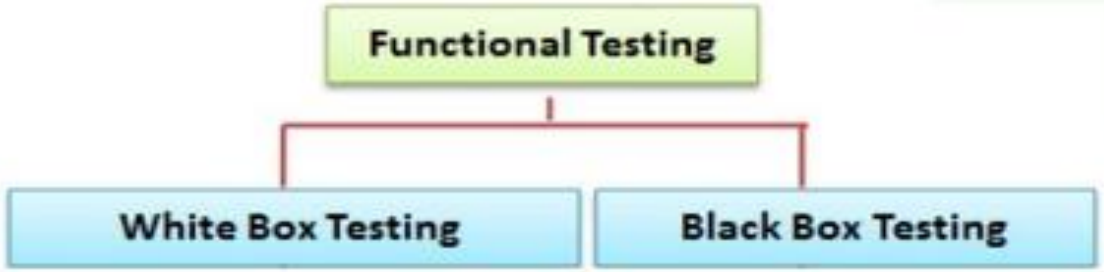
Software Testing

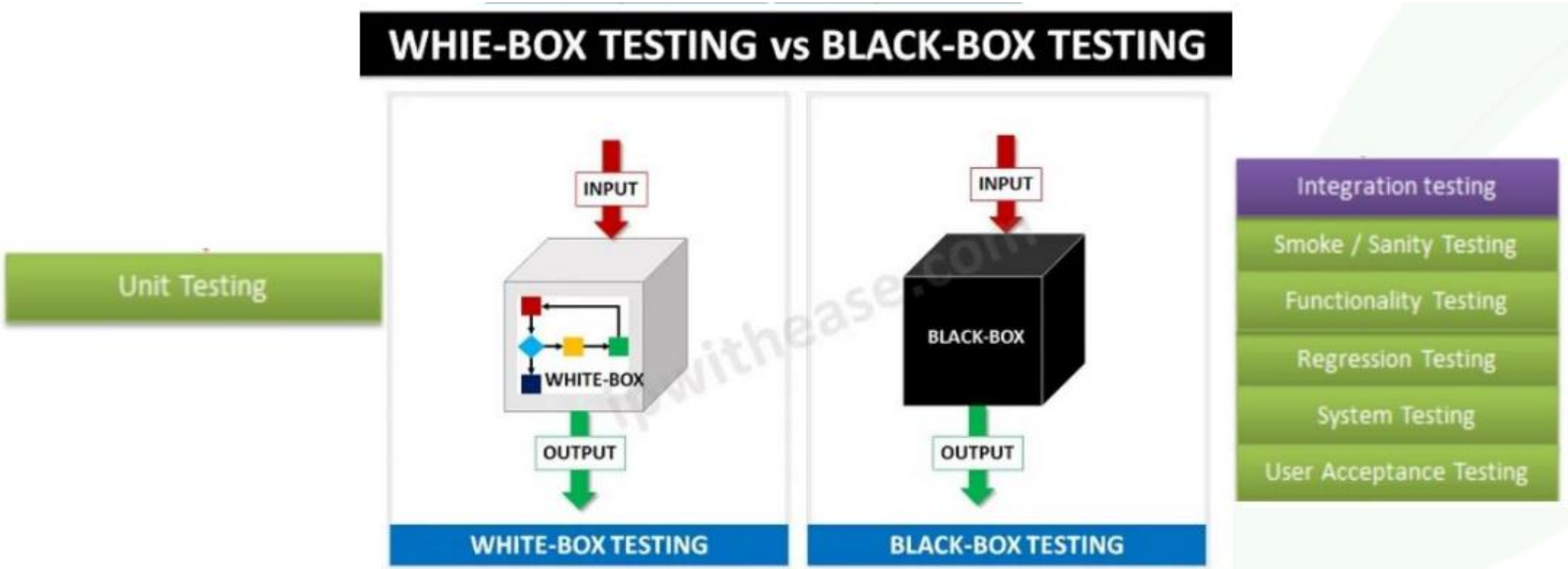
- **Statik test** kod yürütülmeden kodun veya diğer proje dokümanlarının manual olarak gözden geçirilmesidir. Statik testler dinamik testlere geçilmeden önce yapılmalıdır. Projenin başlarında gözden geçirme yoluyla tespit edilen hataların çözülmesi ilerleyen aşamalarda bulunmasından daha az maliyetlidir.
- **Dinamik Test:** Kodun bütününü çeşitli yöntemlerle test etmeye yarayan metottur. Bulunan tüm hatalar çözülmeden ve testin sonlandırma kriterleri sağlanmadan sona ermez. Test edilecek yazılımın türüne göre, uygulanma metotları farklılık gösterebilir.



Fonksiyonel Test, testini gerçekleştirdiğimiz uygulamanın her bir fonksiyonunun verilen gereksinimlere uygun olarak çalışıp çalışmadığını doğrulayan test türüdür. Fonksiyonel test Black Box test altında
kullanıldığı için uygulamanın kaynak kodu ile ilgili değildir. Bu testi gerçekleştirirken odak
noktası daima uygulamanın ana işlevlerinin kullanıcı dostu olmasıdır.

Fonksiyonel olmayan test, Uygulamanın işlevsel olmayan özelliklerinin test edildiği bir test türüdür. Amaç sistemin hazır olup olmadığını belirlemektir. Bileşenlerin veya sistemin kalite özellikleri test edilir. Yazılımın kalitesinde ve doğru çalışmasında işlevsel testler kadar önemlidir. Örneğin sistemi aynı anda kaç kullanıcı kullanabilir sistem yeterince güvenli midir? Bu gibi soruların karşılığını almak için sistem test edilir.


Functional Testing
White Box Testing
Black Box Testing


WHIE-BOX TESTING vs BLACK-BOX TESTING
Unit Testing
INPUT
WHITE-BOX
OUTPUT
WHITE-BOX TESTING
INPUT
BLACK-BOX
OUTPUT
BLACK-BOX TESTING
Integration testing
Smoke / Sanity Testing
Functionality Testing
Regression Testing
System Testing
User Acceptance Testing

Beyaz kutu testinde, kodun içine girilerek kodun doğruluğu ve kalitesi test edilir. Bu test türünde kod erişimi zorunludur. Kod yapısı ve tasarımına yönelik testler gerçekleştirilir. Örneğin, gereksiz bir kod bloğu tespit edilebilir veya kodun okunabilirliğini arttırmaya yönelik durumlar tespit edilebilir. Kodda erken bulunacak hatalar Kara Kutu(Black Box) testlerini de kolaylaştırmaktadır. Beyaz kutu testleri çoğunlukla geliştiriciler tarafından uygulanır.

Kara kutu testleri; kodun yapısı(structure), tasarımı(design) ve uygulanışı(implementation) ile ilgilenmez. Kara kutu testlerinde girdi ve çıktı değişimine göre sistemin nasıl çalıştığı test edilir. Kara kutu test çeşitleri çoğu yazılım test uzmanı tarafından yaygın olarak kullanılan test çeşitleridir.

Kara kutu test teknikleri:

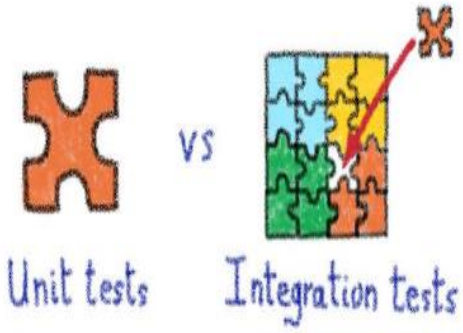
1- Equivalence Partitioning (Eşit Bölümlere Ayırma Tekniği)
2- Boundary Value Analysis (Sınır Değerleri Analizi)
3- Decision Table (Karar Tablosu)
4- State Transition Table (Durum Geçiş Tablosu)
5- Use Case Testing (Kullanım Durumları Testi)

UNIT TEST (BİRİM TESTİ)- whitebox

- Her modülü test ettikten sonra, entegre modüllerin istenen çıktıyı verdiğinden emin olmak için bir araya getirilir ve test edilir. Tipik olarak her yazılım, farklı yazılım geliştiriciler tarafından kodlanmış farklı yazılım modüllerinden oluşur. Bu test her bir modülün herhangi bir kusur olmadan mükemmel bir şekilde etkileşime girmesini sağlamayı amaçlar. Bu modüllerin verileri arasındaki iletişime odaklanır. Entegrasyon testi, dizi testi ve iş parçacığı testi olarak da bilinir.

- Birim testi sırasında gözden kaçmış olabilecek kusurları, birim testi yapıldıktan sonra istemcilerin gereksinimlerinde meydana gelen değişiklikler gibi faktörlerden kaynaklanabilecek kusurları, harici donanım arayüzlerinden kaynaklanan hataları, yazılımlar arasındaki etkileşimlerden kaynaklanan hataları ortadan kaldırmak için entegrasyon testi gereklidir.

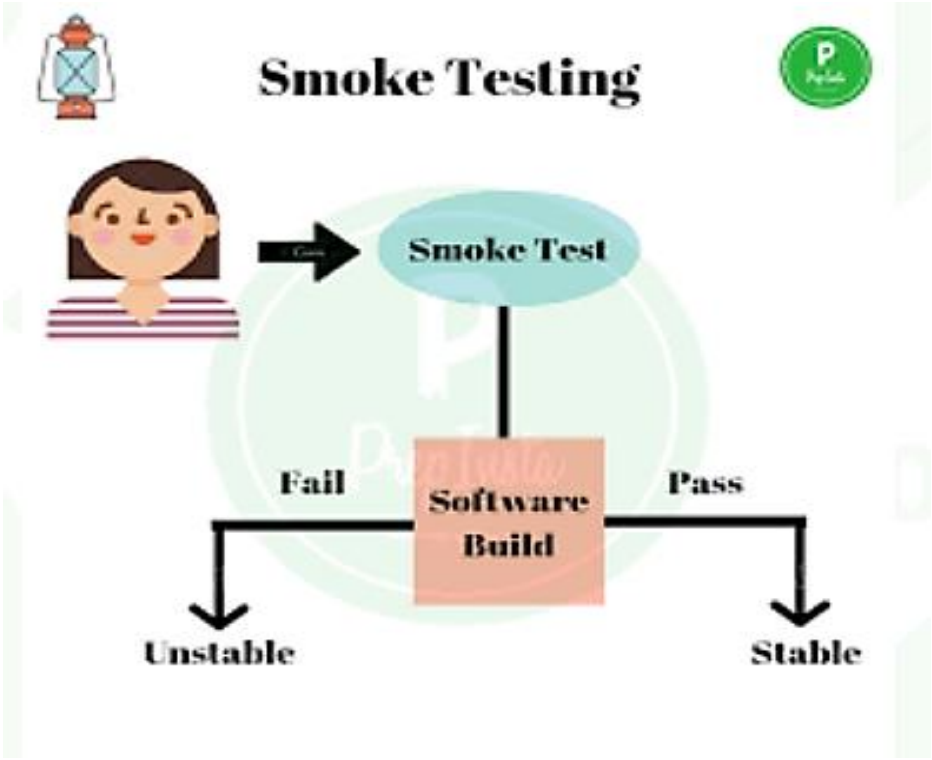
INTEGRATION TEST(Entegrasyon Testi)

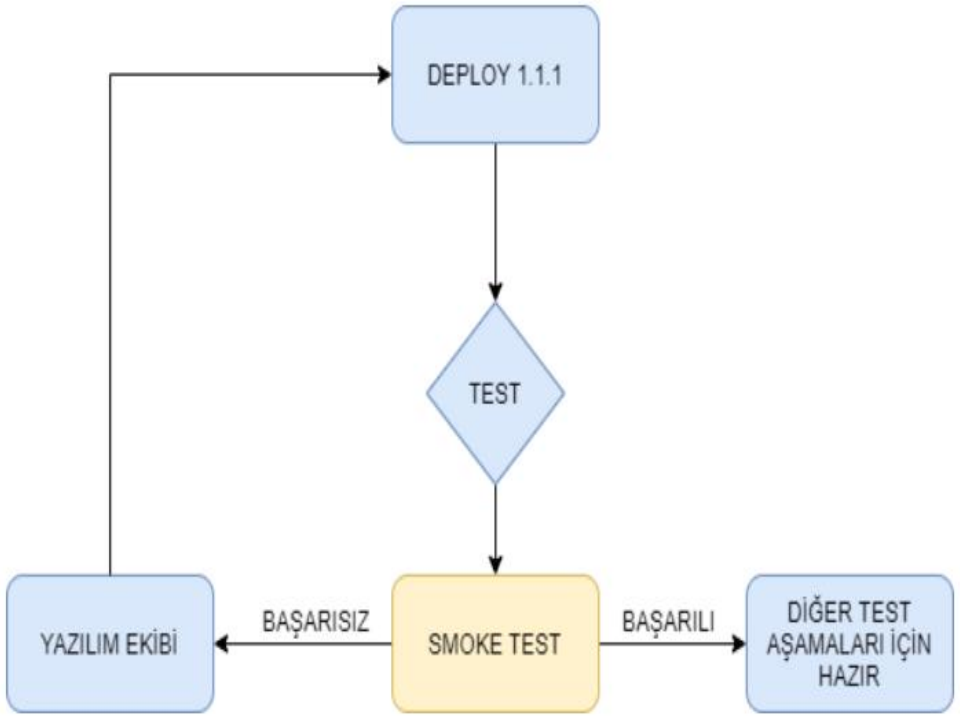


- Her modülü test ettikten sonra, entegre modüllerin istenen çıktıyı verdiğinden emin olmak için bir araya getirilir ve test edilir. Tipik olarak her yazılım, farklı yazılım geliştiriciler tarafından kodlanmış farklı yazılım modüllerinden oluşur.
- Bu test her bir modülün herhangi bir kusur olmadan mükemmel bir şekilde etkileşime girmesini sağlamayı amaçlar. Bu modüllerin verileri arasındaki iletişime odaklanır. Entegrasyon testi, dizi testi ve iş parçacığı testi olarak da bilinir.
- Birim testi sırasında gözden kaçmış olabilecek kusurları, birim testi yapıldıktan sonra istemcilerin gereksinimlerinde meydana gelen değişiklikler gibi faktörlerden kaynaklanabilecek kusurları, harici donanım arayüzlerinden kaynaklanan hataları, yazılımlar arasındaki etkileşimlerden kaynaklanan hataları ortadan kaldırmak için entegrasyon testi gereklidir.

SMOKE TEST(Duman Testi)

- Duman testi projede en önemli işlevlerin çalışmasını sağlamayı amaçlayan kapsamlı olmayan bir test türüdür. Temel sistemin aynı kalması için tüm sistem birlikte test edilmelidir. Bu teste önemli olan büyük resmi görmektir. Amaç uygulamanın teste devam edecek düzeye gelip gelmediğini kontrol etmektir. Duman testlerini evreleme ve üretim ortamlarında erken ve sık sık tekrarlamak gerekmektedir.





- Proje deploya alındığından test ekibi tarafından smoke teste tabi tutulur. Smoke test projenin en temel en önemli fonksiyonlarının bozulup bozulmadığını kontrol eder.
- Eğer proje genel olarak çalışır durumdaysa ve smoke test kısmından başarı elde ettiyse daha detaylı testler için hazır demektir.
- Fakat başarısız sonuç aldıysa geliştirici ekibe kırılan yerler bildirilir. Eksiklik ve bozukluk olan yazılımın fonksiyonel teste gönderilmesi mantıklı olmayacaktır. Böylece, ürünün daha kaliteli ve kullanılabilir hale gelmesine olanak sağlanır.
- Kısacası test, ürünü kaliteli kılan bir yapıdır.

- **Manuel Duman testi**

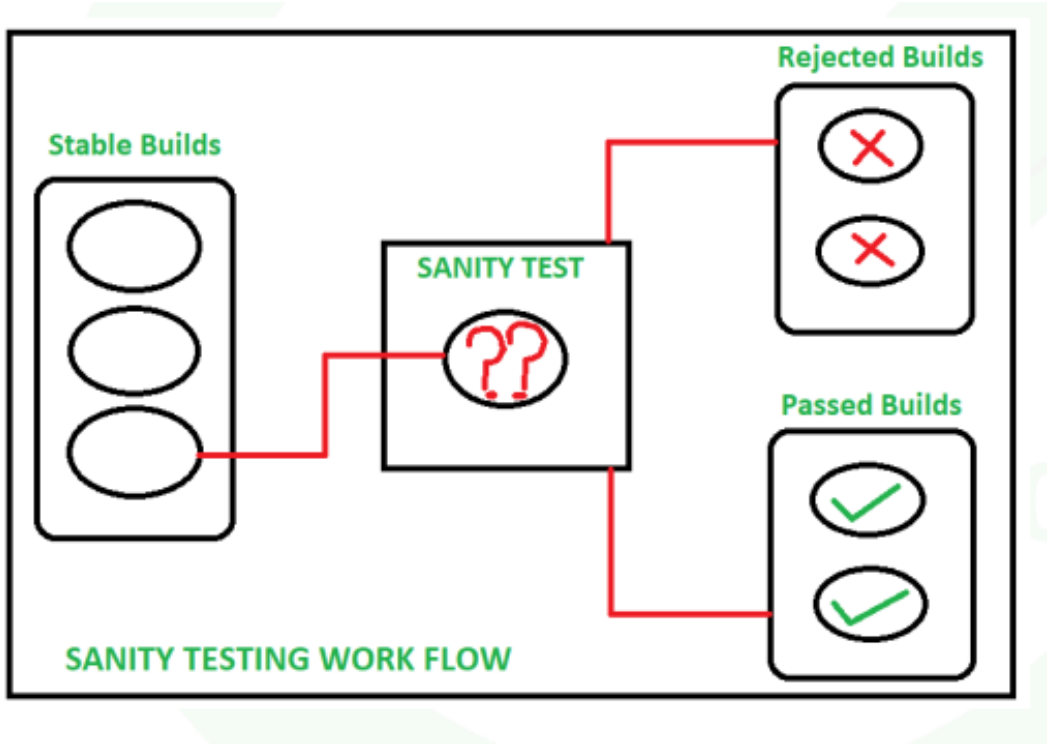
- Genelde duman testi manuel olarak yapılır. Test geçerse, fonksiyonel teste devam edilir. Test başarısız olursa, yapı reddedilip düzeltme için geliştirme ekibine geri gönderilir ve yeni sürümle duman testi yeniden başlatılır. Duman testi yeni yapı üzerinde gerçekleştirilir ve sistemin doğruluğunu korumak için eski yapılarla entegre edilir.

- **Otomasyon ile Duman Testi**

- Yeni yazılım versiyonu dağıtıldığında manuel olarak testi tekrarlamak yerine, yapıya karşı kaydedilen duman testi durumları yürütülür. Ana işlevlerin hala düzgün çalışıp çalışmadığını doğrular. Test başarısız olursa, yapıyı düzeltebilir ve yapıyı hemen yeniden dağıtabilirler. Bu sayede zamandan tasarruf edebilir ve kaliteli bir yapı sağlanır. Test mühendisi, otomatik bir araç kullanarak, yazılım oluşturmada gerçekleştirilen tüm manuel adımları kaydeder.

SANITY TEST

- Uygulama üzerinde küçük bir hata giderildiğinde veya küçük değişiklikler ardından yapılan testlerdir. Sanity testi test sürecinde uygulamanın bir iki işlevine odaklanırken duman testi tüm önemli işlevlerin çalıştığından emin olmak için yapılır. Bu test sayesinde hatalar daha erken bir aşamada keşfedilir.

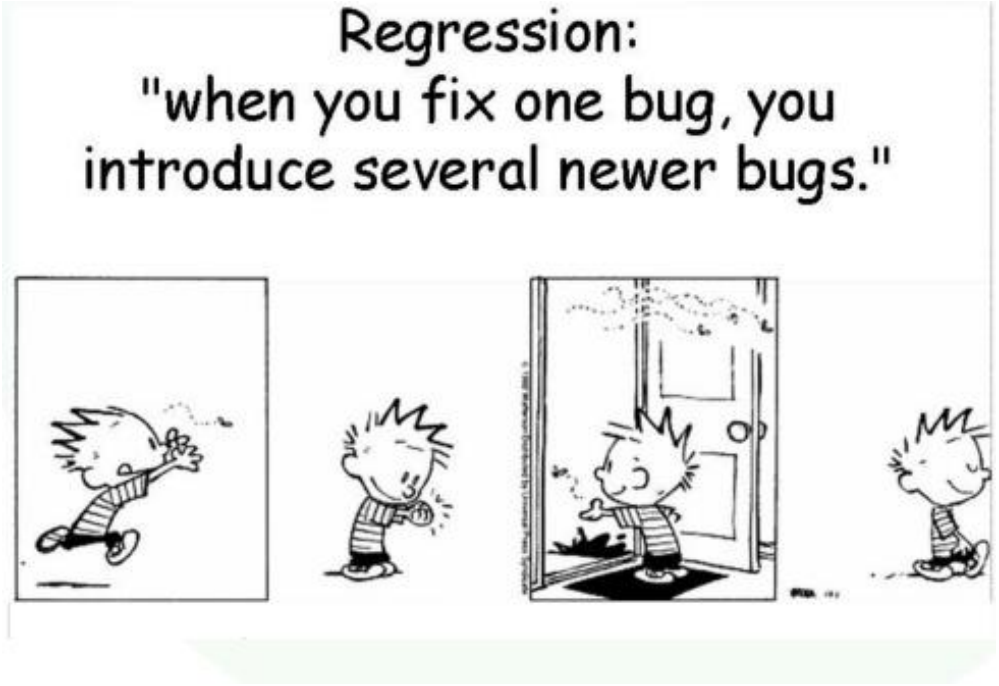


REGRESSION TEST(Regresyon Testi)

- Bu test türünde yazılım üzerinde yapılan değişiklik sonucunda uygulama kodunun bozulup bozulmadığını doğrulamak için yapılan test türüdür. Değişiklik sonrasında mevcut özelliklerin çalışıp çalışmadığını kontrol etmek için yapılan test türüdür.

- 1. **Kısmi Regresyon**, kodda değişiklikler yapıldığında ve bu birimin değişmemiş veya halihazırda mevcut olan kodla entegre edildiğinde bile kodun iyi çalıştığını doğrulamak için yapılır.

- 2. **Tam Regresyon**, kodda bir dizi modülde bir değişiklik yapıldığında ve ayrıca başka herhangi bir modüldeki bir değişikliğin etkisinin belirsiz olması durumunda yapılır. Değiştirilen kod nedeniyle herhangi bir değişikliği kontrol etmek için ürün bir bütün olarak test edilir



Sistem Testi (E2E – END TO END Test)

- Sistem Testi, yazılım gereksinim testlerinin tamamlanmasından sonra, sistem gereksinimlerine göre oluşturulan testleri kapsar.
- Yazılım tarafında yapılan Birim Testi(Unit Test) ve Entegrasyon Testi(Integration Test) adımlarından sonra yapılan sistem testleri, daha çok işlevi tamamlanan yazılımın, güvenlik, güvenilirlik, performans gibi faktörler altında yapılan test işlemlerini kapsar.
- Sistem testinin amacı, bir uygulamanın tasarlandığı gibi işlevleri gerçekleştirirken doğruluğunu ve eksiksizliğini doğrulamaktır.
- Gerçek sonuçlar ve beklenen sonuçlar sıraya girdiğinde veya farklılıklar, müşteri girdisine göre açıklanabilir veya kabul edilebilir olduğunda sistem testi tamamlanmış olarak kabul edilir.
- • İki şekilde ifade edilebilir
- 1) Farklı fonksiyonların birleştiği uçtan uca senaryolar
- 2) Aynı modülün backend ve ui tarafında ele alındığı senaryolar

Ad-hoc Testing (Gecici Test –Maymun Testi)

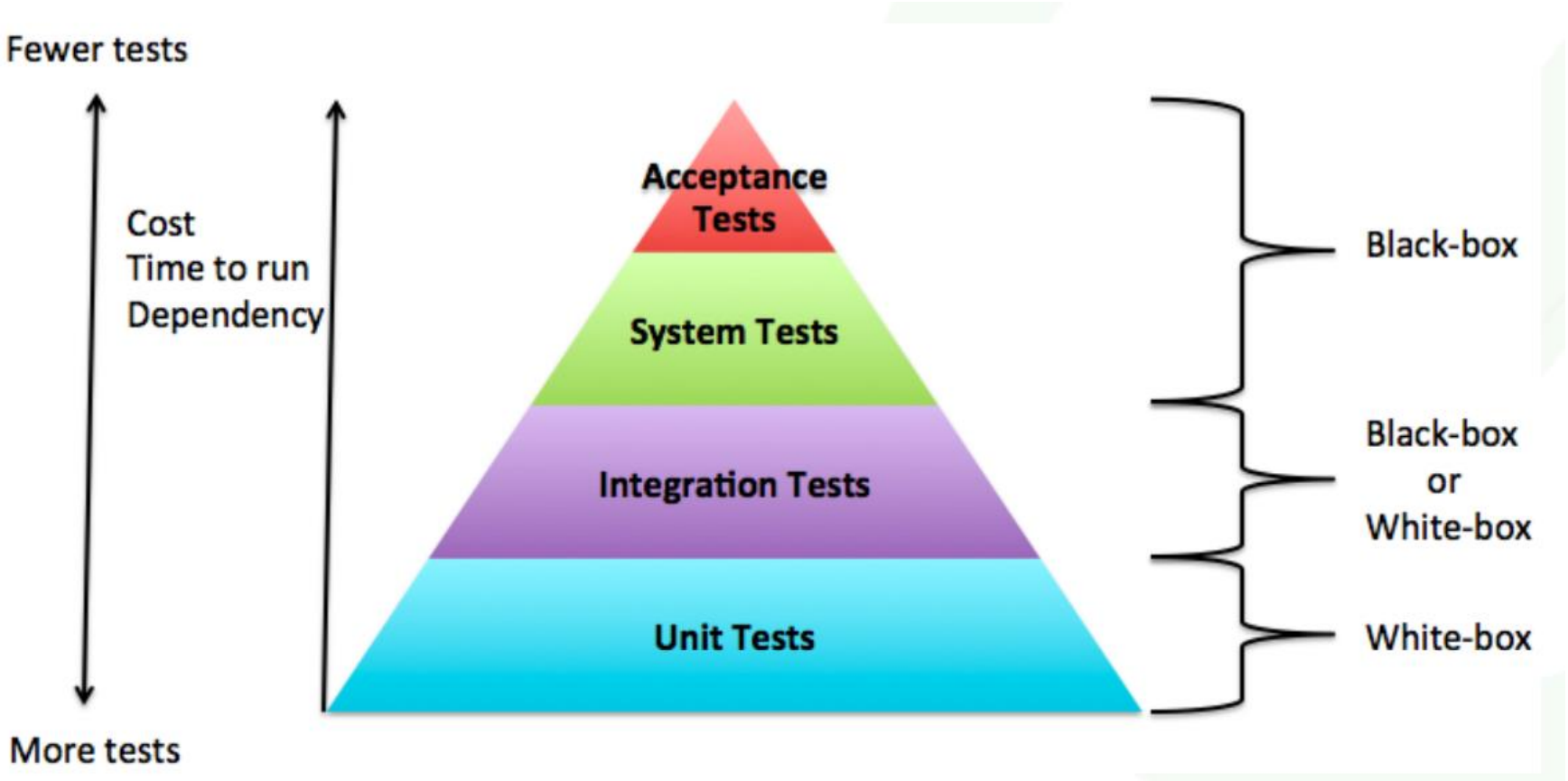
- Rastgele Test veya Maymun Testi olarak da bilinir, herhangi bir planlama ve belge olmadan bir yazılım testi yöntemidir.

- Testler herhangi bir resmi prosedür veya beklenen sonuç olmadan gayri resmi ve rastgele yapılır.

User Acceptance Testing UAT (Kullanıcı Kabul Testi)

- Bu testtin amacı yazılımı iş gereksinimlerine göre doğrulamaktır. Amaç yazılımın kullanıcı ve müşteri tarafından kabul edilip edilmeyeceğini belirlemek için test edilmesine olanak sağlamaktır. Bu doğrulamalar iş gereksinimlerine aşina olan son kullanıcılar tarafından gerçekleştirilir. Fonksiyonel, sistem ve regresyon testleri gerçekleştirildikten sonra kullanıcı kabul testleri gerçekleştirilir. Yazılım yayınlanmadan gerçekleştirilen son testtir. Beta testi olarakta adlandırılır

Test Piramidi – Test Hiyerarşisi



BLACK BOX (KARA KUTU) TEST TENİKLERİ

1- Denklik Paylarına Ayırma - Equivalence Partitioning (EP)

- Denklik paylarına ayırma test tekniğinin amacı benzer özelliklere sahip test senaryolarını gruplandırılarak daha az test senaryosu yazılmasıdır. Bu tekniğe göre aynı çıktıyı veren test girdi grubu için bir tane test senaryosu yazılması yeterli görülür. Bu test tekniği tüm test seviyelerinde uygulanabilir
- Geçerli ve geçersiz test girdilerinin (denklik sınıflarının) aşağıdaki gibi olduğunu söyleyebiliriz.
- • Geçersiz Aralık: 0 ve 0'dan küçük sayılar
- • Geçerli Aralık: 1-120 arasındaki sayılar
- • Geçersiz Aralık: 121 ve 121'den büyük sayılar
- Denklik sınıflarının doğru belirlenmesi test tasarımınızı ve dolayısıyla yapacağınız testlerin kalitesini olumlu şekilde etkileyecektir.

2 - Sınır Değer Analizi - Boundary Value Analysis

- Sınır değer analizi test tekniği tüm test seviyelerinde uygulanabilir. Uygulaması kolay hata bulma becerisi yüksektir. Bu teknik özellikle denklik paylarına ayırma test tekniği ile kullanılırsa daha etkili sonuçlar elde edilebilir. Yine doğum tarihi hesaplama uygulaması üzerinden örnek vermek gerekirse;

	Formül	**Test Girdisi**
Geçersiz Aralık:	Alt sınır değeri – 1	0
Geçerli Aralık:	Alt sınır, Alt sınır + 1 ve Üst Sınır -1, Üst Sınır	1,2 – 119,120
Geçersiz Aralık:	Üst Sınır + 1	121

Sınır değer analizi tekniğinde görüldüğü üzere bir sınır değerinin testi için 3 farklı test senaryosu yazılabiliyor.

3- Karar Tablosu Testi - Decision Table Testing

- Karmaşık iş kurallarına sahip sistemlerin test edilmesinde kullanılan test tekniğidir. Karar tablosu test tekniğinin en büyük avantajı test sırasında gözden kaçabilecek olasılıkların net bir şekilde listelenerek gözden kaçırma riskinin en düşük seviyelere indirilmesidir. Gelin bu test tekniğini örnekle daha iyi anlamaya çalışalım.

Durum	**Kural 1**	**Kural 2**	**Kural 3**	**Kural 4**
Kullanıcı adı doğru mu ?	Evet	Evet	Hayır	Hayır
Parola doğru mu ?	Evet	Hayır	Evet	Hayır
Aksiyon				
Oturum açma başarılı	Evet			
Oturum açma başarısız		Evet	Evet	Evet

4- Kullanım Durum Senaryosu Testi - Use Case Testing

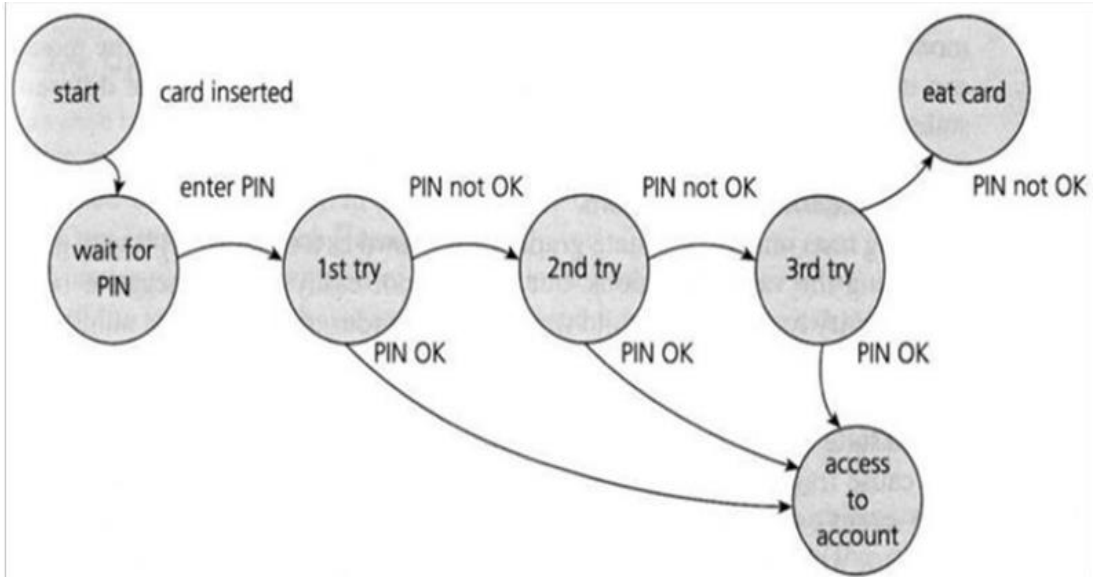
- Kullanım durum senaryoları sistem ile aktör arasındaki etkileşimi göstermek için yazılan senaryolar topluluğudur. Kullanım durum senaryosu testinin amacı sistemi baştan sona her bir senaryonun teker teker bütün sistemi kapsayacak şekilde test edilmesidir. Kullanım durum senaryolarında ana senaryo ve alternatif senaryolar bulunur. Ana senaryolarda sistemin yerine getirmesi gereken adımlar yer alır. Alternatif senaryolarda ise sistemin uç noktaları için test adımları yazılır. Örneğin bir kullanıcı tanımlama ekranında ana senaryo ekranda yer alan tüm zorunlu alanların doldurularak kaydedilebilmesi iken alternatif senaryoda ekranda yer alan bir veya birden fazla zorunlu alanın boş bırakılarak sistemin verdiği tepkinin ölçülmesi olabilir.

Ekran Adı	Kullanıcı Tanımlama ve Güncelleme
Tanım	Yetkili kullanıcının kullanıcı tanımlama ve güncellemesi amaçlanmıştır.
Aktörler	Yetkili Kullanıcı
Gereksinim Numarası	SRS-KM-1
Ön Koşullar	Yetkili kullanıcı sisteme giriş yapmış olmalıdır.
Ana Senaryo	1. Yetkili kullanıcı ekranda yer alan aşağıdaki alanları doldurur/seçer; a. Adı* b. Soyadı* c. TCKN* d. E-Posta* e. Cep Telefonu f. Adres 2. Yetkili kullanıcı, **Kaydet** butonuna basar. 3. Sistem, kullanıcı kaydetme işleminin başarıyla tamamlandığına dair bilgilendirme mesajı verir.
Alternatif Senaryo	1a) Yetkili kullanıcı ekranda yer alan zorunlu alanları boş bırakarak kaydet butonuna basar. 2a) Yetkili kullanıcı sistemde tanımlı olan kullanıcının TCKN bilgisini kullanarak yeni kullanıcı tanımlamaya çalışır. 2b) Yetkili kullanıcı sistemde tanımlı olan kullanıcının E-Posta bilgisini kullanarak yeni kullanıcı tanımlamaya çalışır.
Son Koşullar	3a) Sistem kaydetme işleminden sonra işlem kaydını (log) veri tabanına kaydeder.

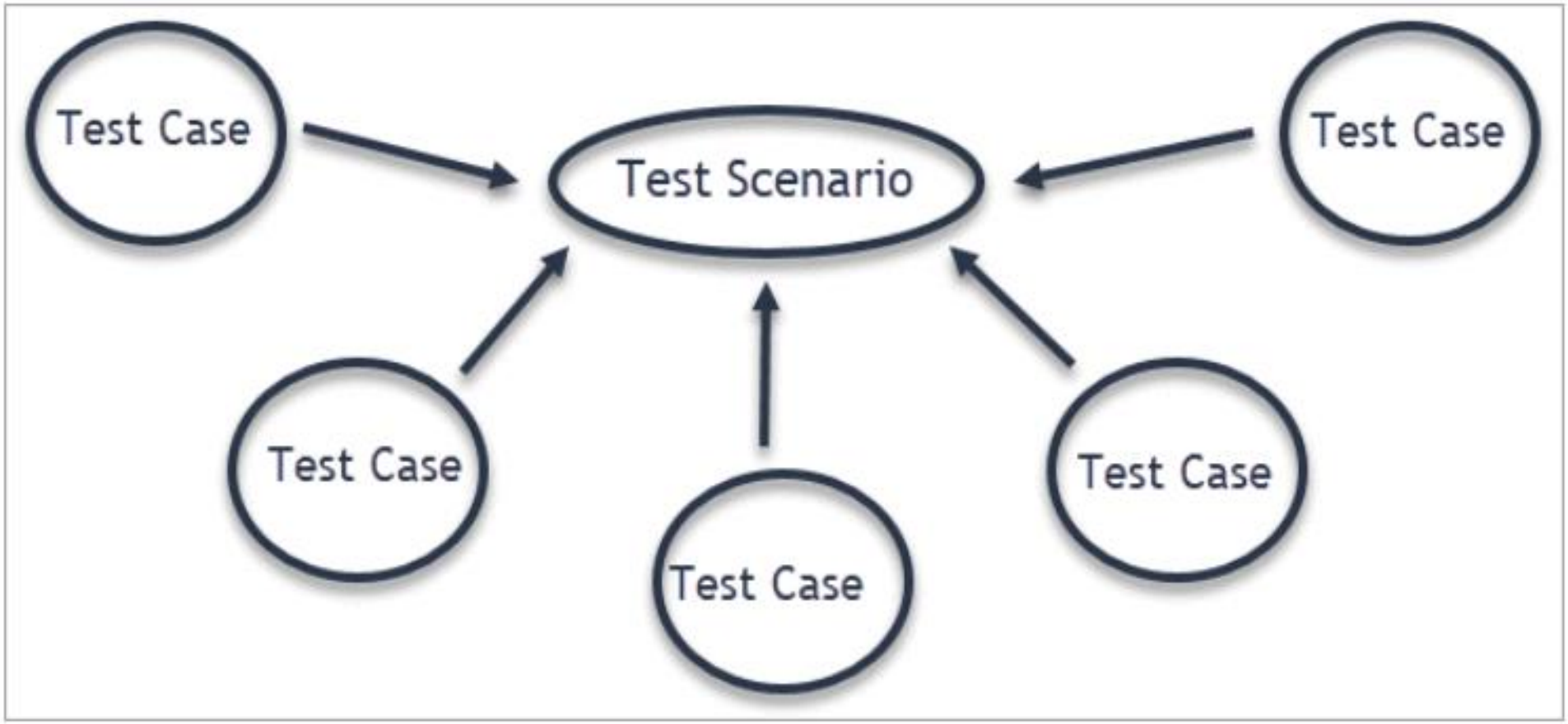
5- Durum Geçiş Testi - State Transition Testing

- Durum geçiş testinin amacı belli iş kurallarına bağlı olarak şartların oluşmasına ve bir durumdan diğerine geçilerek bir noktada sonlanması durumunu test etmektir. Bu tür sistemlerde bu durumlar durum geçiş diyagramı ile gösterilir. Durum geçiş test tekniğini bir örnekle ele alalım.

- Yanda yer alan ATM örneğinde ilk olarak PIN kodu bekleniyor daha sonra girilen PIN kodu kontrol ediliyor eğer PIN kodu doğruysa kullanıcı hesabına erişebiliyor 3 hatalı PIN kodu girişinde ise ATM kartı yutuyor. Durum geçiş test tekniği genellikle gömülü sistemlerin testlerinde kullanılmaktadır. Kara kutu test teknikleri ile projelerinizde daha efektif test senaryoları tasarlayabilirsiniz.



Test Senaryosu



Test senaryosu nedir?

- Test senaryosu, test gerekliliklerinin, fonksiyonelitiye göre üst seviye bir sınıflandırılmasıdır. Bu dokümantasyonda test edilecek fonksiyonların senaryolarını dokümante etmeniz beklenir ve çok detaylı yazılmazlar.
- Kafanızda daha iyi canlandırmanız adına biraz daha açmak gerekirse. Test edilecek bir modülünüz var, bu modülün fonksiyonlarını test etmek için belirlediğiniz kriterlerin dokümantasyonudur.

Test Senaryoları Neden Önemlidir?

- Test Senaryoları, yazılım uygulamasının gerçek dünyadaki durumlara göre değerlendirilmesine yardımcı olur.
- Test senaryoları belirlendiğinde, test kapsamını ikiye ayırmaya yardımcı olur.
- İşlevselliklerin öncelikli olarak test edilmesi, yazılım uygulamasının başarılı bir şekilde uygulanmasına büyük ölçüde yardımcı olur.
- Test senaryoları önceliklendirildikçe, en önemli işlevler kolaylıkla tanımlanabilir ve öncelik sırasına göre test edilebilir. Bu, önemli işlevlerin çoğunun düzgün çalışmasını ve bunlarla ilgili kusurların gerektiği gibi yakalanmasını ve düzeltilmesini sağlar.
- Test senaryoları, yazılımın iş süreci akışını belirler ve böylece uygulamanın uçtan uca test edilmesi mümkündür.

Test Senaryosu Adımları:

- Scenario Type (Senaryo Tipi)
- Priority (Öncelik)
- Test Name (Test Adı)
- Test Steps (Test Adımları)
- Pre-condition (Ön koşul)
- Description (Açıklama)
- Expected Result (Beklenen)
- Status (Durum)
- Notes (Notlar)

1) Scenario Type (Senaryo Tipi):

- Yapılan teste göre aşağıdaki uygun senaryo tiplerinden biri yazılır.
- **Fonksiyonellik Testi (Functional Testing):** Uygulamaya ait fonksiyonların test edilmesidir.
- **Stres Testi (Stress Testing):** Fazla sayıda veri girişi, çok sayıda sorgu ve büyük nümerik değerler olduğu zaman kullanılan test türüdür. Bu sayede sistemin verdiği tepki ve sistemin dayanıklılığı gözlenir.
- **Yükleme Testi (Load Testing):** Sistemin performansını derecelendirmek veya ağır yükler ve çok sayıda veri girişinde sistemin nerede ve hangi koşullarda çökeceğinin belirlendiği test çeşididir.
- **Araştırma Testi:** Test edilecek uygulamanın test öncesinde öğrenilerek, hakkında fikir edinilmesidir.
- **Kullanılabilirlik Testi(Usability Testing):** Kullanıcı arayüzü eğer önemli bir yere sahipse ve ihtiyaçlar belirli bir kullanıcıya göre belirleniyorsa bu test tipi kullanılmalıdır.
- **Duman Testi (Smoke Testing):** Mantık testi olarak da bilinmektedir. Duman testi uygulamanın büyük testlere hazır olup olmadığını belirleyerek küçük testleri başarılı bir şekilde geçtiğini gösterir.
- **Yenilenme Testi**: Uygulamanın herhangi bir hataya karşı ne kadar sürede eski haline geleceğinin belirlendiği test çeşididir. Sistem gereksinimlerine göre yenilenme hızı ve tip belirlenir.
- **Seviye Testi:** Uygulama tarafından çok büyük veri işlemi yapılarak, sistem uç noktasına kadar zorlanır ve sistemin uç noktalarının belirlenir.
- **Ad-Hoc Testi:** Yapılacak olan bütün testlerin sürelerinin belirlenmesidir.

2) Priority (Öncelik) :

- Yazılan adımın öncelik derecesi bu alana yazılır. Teste öncelik her zaman gözden geçiren tarafından belirlenmelidir. 4 çeşit öncelik derecesi mevcuttur.
- Critical (Kritik)
- High (Yüksek)
- Medium (Orta)
- Low (Düşük)

- 3- **Test Name (Test Adı):** Yapılan testin adıdır. Örn: Yeni kullanıcı tanımı.
- 4- **Test Steps (Test Adımları) :** Yazılan test senaryosu kaç adımdan oluştuğu bu alandan takip edilir.
- 5- **Pre-condition (Ön koşul):** Teste başlanmadan önce yerine getirilmesi gereken işlemlerdir. İhtiyaç ve gereksinim olduğu durumlarda bu alan doldurulur ve belirtilen ön koşul sağlandıktan sonra test edilmesi gerektiği anlaşılır.
- 6- **Description (Açıklama):** Yapılacak testin girdileri bu alana yazılır.
- 7- **Expected Resu(Beklenen):** Test yürütüldükten sonra sistemin çıktısının ne olması gerektiğinin belirtildiği alandır. Girdiler alanında yazan test senaryosu gerçekleştikten sonra beklenen çıktılar kontrol edilir. Sistemde gerçekleşen çıktı ile beklenen çıktı karşılaştırılır.
- 8- **Status (Durum):** Passed ve Failed olmak üzere 2 tip durum mevcuttur. Gerçek sonuç beklenen sonuçla aynı ise status alanına Passed, değilse Failed değerleri yazılır.
- 9- **Notes (Notlar):** Yukarıda açıklanan alanları destekleyecek bazı özel koşullar veya beklenen, gerçekleşen sonuçlarla ilgili herhangi bir sorun mevcutsa bu alana yazılır.

Test Senaryoları:

Test Senaryoları	Test Durumları
Eposta alanına fazla karakter girilmesi	Negatif
Şifre alanına fazla karater girilmesi	Negatif
Eposta alanına eposta formatında bir değer girilmemesi	Negatif
Eposta ve şifre alanlarına doğru bir değer girilmesi	Pozitif
Epostası doğru Şifresi yanlış bir değer girilmesi	Negatif
Alanlara hiç bir değer girilmeden formun gönderilmesi	Negatif
Gönderme butonuna iki kere tıklanması	Negatif
Daha önce üye olmayan bir eposta ile login denemesi	Negatif

Scenario Type	Priority	Test Name	Test Step	Preconditions	Description	Expected Result	Status	Notes
Functional	Medium	Yeni Fatura Oluşturma	Step 1	Fatura oluşturma yetkisi olan kullanıcı ile login olunması	Yeni Fatura giriş ekranına giriş sağlanır.	Ekranın başarıyla açılması.	Passed	
Functional	Medium	Yeni Fatura Oluşturma	Step 2		Yeni kayıt ekle butonuna tıklanır.	Ekranın yeni kayıt modunda açılması.	Passed	
Functional	Medium	Yeni Fatura Oluşturma	Step 3		Müşteri bilgisi seçilir.	Seçilen değerin başarıyla ekranda görüntülenmesi.	Passed	
Functional	Medium	Yeni Fatura Oluşturma	Step 4		Fatura senaryo bilgisi seçilir.	Senaryo tipi alanının seçilen değer olarak dolması.	Passed	
Functional	Medium	Yeni Fatura Oluşturma	Step 5	Zorunlu olan tüm alanların doldurulmuş olması.	Diğer zorunlu alanlar doldurularak Kaydet butonuna tıklanır.	Faturanın başarıyla kaydedilmesi.	Passed	

Test durumu (Test Case) nedir?

- Test durumu, başarılı ve başarısız yöntemlerle koşulabilecek test senaryosu prosedürleridir.
- Aslında türkçe olarak anlatacak olursak, bir test senaryosu sonunda başarılı bir işlem gerçekleşmesini bekliyorsanız başarılı bir test durumunuz olur. Tam tersi olarak, test senaryosu sonunda başarısız bir işlem bekliyorsanız negatif bir test durumunuz olur.
- Tabii bu noktada, pozitif test durumları başarılı, negatif test durumları başarısızdır yargısı oluşmamalıdır. Negatif test durumlarınızın sonunda negatifi yakalamalısınız, pozitif test durumlarınızda ise pozitifi yakalamalasınız. Eğer test sonuçlarınızda elinizdeki sonuçlar eşleşiyorsa, testinizi başarılı ilan edebilirsiniz.

Analiz dokümanındaki her gereksinime karşılık sadece bir senaryo mu olmalı mıdır?

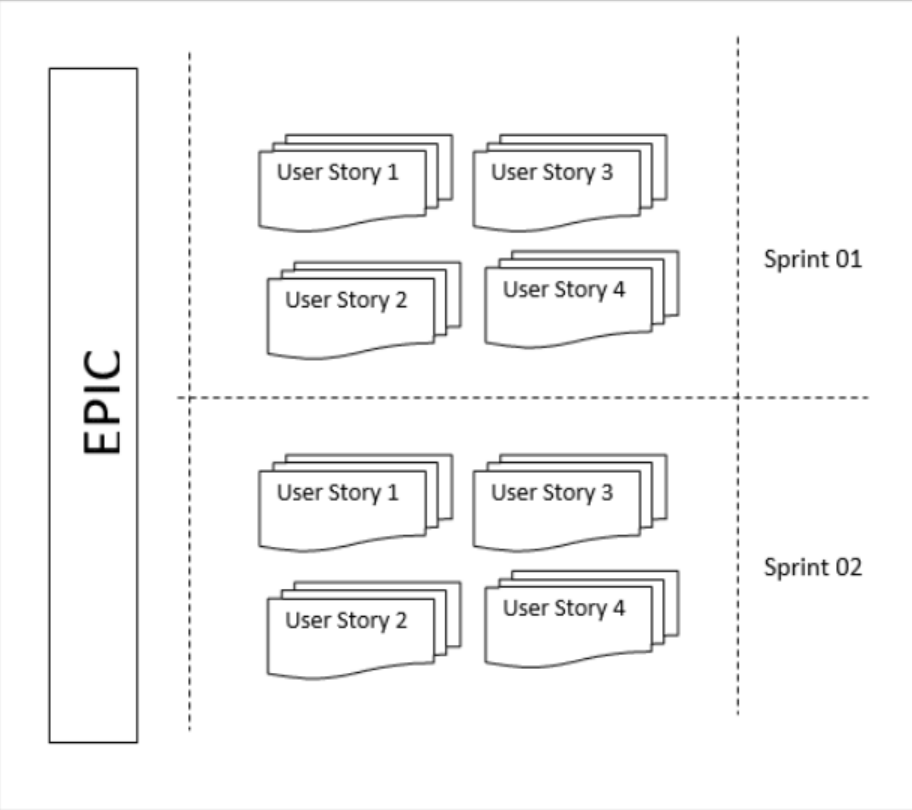
- Aslında bu soruya hem evet, hem hayır denebilir. Buna sebep ise; testin içerik bağımlı olmasıdır. Eğer test ettiğiniz fonksiyon sadece bir butona tıklandığında içerik detayının getirilip getirilmediği ise burada sadece bir test senaryosu yeterli olabilir. Ancak, bir uçaktaki yere yaklaşma sistemini test ediyorsanız sanırım bir gereksinme karşılık 100 testiniz bile olabilir.

- Anlatmak istediğim, aşağıdaki örneği de inceleyecek olursanız, bir ekrandaki testleri bir senaryo altında toplayıp, yüzlerce adıma sahip bir test senaryosu oluşturmayın. Senaryolarınızı mümkün olduğunca bölüp, test adımlarını her test senaryosu için ayrı yazmanız daha doğru olacaktır. Böylece, geliştirme ekibiniz de sistemde nelerin test edildiğini başlık halinde görme şansına sahip olacaktır.

Test Adımı (Test Step) nedir?

- Test adımı, bir ekrandaki test edilecek şeylerin tek tek yazıldığı yer değildir. Test adımı, bir test senaryosunun gerçekleştirilebilmesi için gerekli adımlarının açıklandığı dokümantasyondur. Aşağıdaki örnekte de görebileceğiniz gibi, yazılan senaryoyu gerçekleyen işlemleri yaptıran açıklamalar ve bu atılan adımlar doğrultusunda, sistemin bize her adımda vermesi gerektiği cevapları içeren dokümantasyondur.
- Örnekleyecek olursak:
- Fonksiyon: Login Fonksiyonu
- Ön Koşul:
- • Kullanıcının login olabilmesi için, daha önceden başarılı bir üye kaydı olmalıdır.
- Gereksinimler:
- • Kullanıcının eposta adresini girmesi için bir text box olmalıdır.
- • Kullanıcının şifresini girmesi için bir text box olmalıdır.
- • Kullanıcının bu bilgileri server a göndermesi için bir buton olmalıdır.

Test Senaryolarının Uygulanması

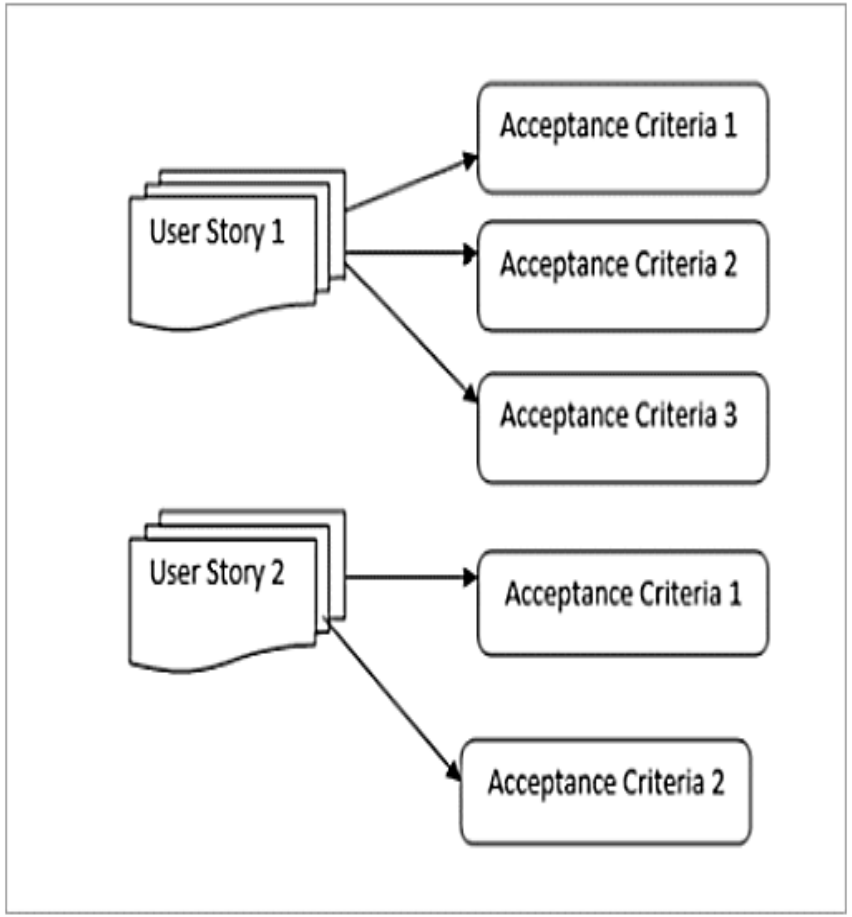


- Epic / İş Gereksinimleri oluşturulur.
- Epic Örneği : Bir Gmail hesabı oluşturduğunuzu düşünün. Epic, bir uygulamanın ana özelliği veya bir iş gereksinimi olabilir.
- Destanlar, sprintler arasında daha küçük kullanıcı hikayelerine bölünmüştür.
- Kullanıcı hikayeleri Epics'ten türetilmiştir. Bu kullanıcı hikayeleri, paydaşlar tarafından temel alınmalı ve onaylanmalıdır.

- Test senaryoları, nihai hale getirilmiş ve temel alınan kullanıcı öykülerinden veya BRS'den (İş Gereksinim Belgesi), SRS'den (Sistem Gereksinim Şartname Belgesi) veya FRS'den (İşlevsel Gereksinim Belgesi) elde edilir.
- Test uzmanları test senaryolarını yazar.
- Bu test senaryoları, kuruluşa bağlı olarak Takım Lideri, İş Analisti veya Proje Yöneticisi tarafından onaylanır.
- Her test senaryosu en az bir kullanıcı hikayesine bağlanmalıdır.
- Olumlu ve olumsuz test senaryoları tanımlanmalıdır.

Kullanıcı hikayeleri şunlardan oluşur: Kabul kriterleri gibi :

- **Kabul Kriterleri**, koşulların bir listesidir veya müşteri gereksinimleri için niyet durumudur. Kabul kriterleri yazılırken müşterinin beklentileri ve ayrıca yanlış anlamalar dikkate alınır.
- Bunlar, bir kullanıcı hikayesi için benzersizdir ve her kullanıcı hikayesinin, bağımsız olarak test edilebilecek en az bir kabul kriterine sahip olması gerekir.
- Kabul kriterleri, hangi özelliklerin kapsam dahilinde olduğunu ve hangilerinin bir proje için kapsam dışında olduğunu belirlemede yardımcı olur. Bu kriterler, işlevsel ve işlevsel olmayan özellikleri içermelidir.
- İş Analistleri kabul kriterlerini yazar ve Ürün Sahibi bunları onaylar.
- Veya bazı durumlarda, ürün sahibi kriterleri kendisi yazabilir.
- Kabul kriterlerinden test senaryoları elde edilebilir.



Excel'deki Test Senaryosu Dokümanında olması gereken Title'lar;

- * **Test Case No**: Test Numarası (Kesin Olmalıdır)
- * **Priority** (Öncelik)
- * **Test Name**: Test Senaryosu Adı (Kesin Olmalıdır)
- * **Test Step**: (Test Adımı, Adımları (Kesin Olmalıdır)
- * **Result**: Beklenen Sonuç (Kesin Olmalıdır)
- * **Status**: Durumu, Done/Undone - Pass/Fail (Kesin Olmalıdır)
- * **Test Datası**
- * **BUG** (Hata) Sayısı yazılır
- * **Notes** (Detay ve Notlar)
- * **Sprint No** (Agile & Scrum Çalışmasında Kullanılır)

Usert_Story_ID	Description	Acceptance Criteria	Priority	Validation
US_001	Amazon Search	Gecersiz e-posta ile giris yapilmamali	Hight	

User Story ID	Test Case ID	Test Objective	Pre-Condition	Steps	Test Data	Expected Result	Actual Result	Status
US_001	TC_001	Amazon Log In Test	https://www.amazon.com	1- Kullanici Amazon web sitesine gider		Kullanici gecersiz e-mail ile giris yapamamali	Kullanici gecersiz e-mail ile giris yapamadi	PASS
				2- login butonuna tiklar				
				3- gecersiz e-posta gonderir, faker kullanarak	Fake email			
				4- gonderdigi e-postanin ekran goruntusunu alir				
				5- continiue'a tiklar				
				6- There was a problem mesajini dogrular				
				7- Need help e tiklar				
				8- forgot your password e tiklar				
				9- Password assistance metnini dogrular				
				10- Geri gider				
				11- Create your account butonuna tiklar				
				12- Create account metnini dogrular				
				13- sayfayi kapatir				

Test Case'te Olması Gereken Alanlar

A	B	C	D	E	F	G	H	I
User Story ID	Test Case ID	Test Objective	Pre-Condition	Steps	Test Data	Expected Result	Actual Result	Status
US 0002	TC_001	Gecersiz kullanıcı adı ile erişim sağlanamaz	Login erisilebilir olmalidir //www.facebook.com/ da	1_ https://www.facebook.com/ gidi 2_kullanici adi textbox a tiklayiniz 3_yanlis bir kullanici adini giriniz 4_sifre textbox ina tiklayiniz 5_dogru bir kullanici sifresi giriniz 6_login butonuna tiklayiniz	URL 3: https://www.facebook.com kullanici adi ="pes_etmek_yok" sifre ="yapabilirim"	kullanici erisim elde edememelidir	kullanici erisim elde edemedi	Pass
US 0002	TC_002	Geçersiz şifre ile erişim sağlanamaz	Login erisilebilir olmalidir //www.facebook.com/ da	1_ https://www.facebook.com/ gidi 2_kullanici adi textbox a tiklayiniz 3_dogru kullanici adini giriniz 4_sifre textbox ina tiklayiniz 5_yanlis bir kullanici sifresi giriniz 6_login butonuna tiklayiniz	URL 3: https://www.facebook.com kullanici adi = techproedusa@gmail.com sifresi = "yanlis_sifre"	kullanici sifre hatasi elde etmelidir ve giris izni verilmemelidirr	Email adresi yanlis veerisim elde edilemedi	Fail
US 0003	TC_003	Geçersiz kullanıcı adı ve şifre ile erişim sağlanamaz	Login erisilebilir olmalidir //www.facebook.com/ da	1_ https://www.facebook.com/ gidi 2_kullanici adi textbox a tiklayiniz 3_yanlis kullanici adi giriniz 4_sifre textbox ina tiklayiniz 5_yanlis sifre giriniz 6_login butonuna tiklayiniz	URL 3: https://www.facebook.com username ="yanlis_username" password ="yanlis_sifre"	kullanici erisim elde edememelidir	kullanici erisim elde edemedi	Pass

Otomasyon Testi

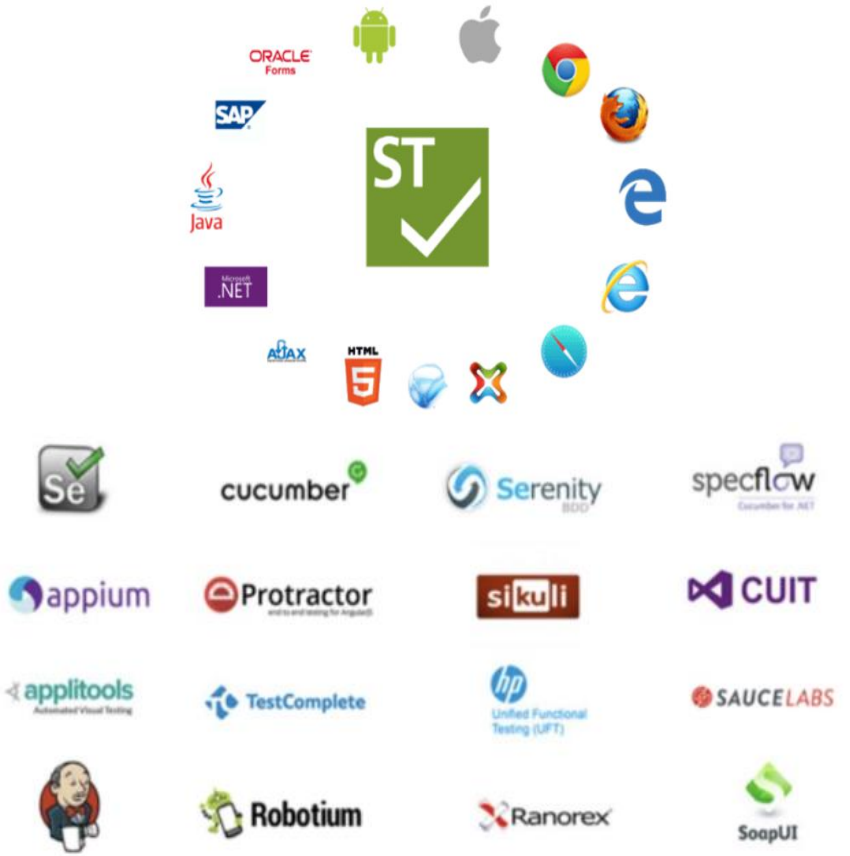
Test planı yürütülürken manuel bir müdahaleye gerek kalmadan yapılan teste otomasyon testi denir. Test senaryolarını yazmak ve yürütmek için otomasyon araçları kullanılır.

Neden Otomasyona İhtiyaç Duyarız?

- Otomasyonun temel mantığı, yapılan değişikliklerin nereyi bozduğunu daha kolay tespit edebilmek. Ürünün üzerinde otomasyon dediğimiz test aşaması gerçekleştiğinde ve devamlı koşan test kümeleri olduğunda kırılan yerleri bulmamız, görmemiz, onarmamız daha kolay oluyor. Gözle görülemeyecek, fark edilmesi zaman alacak sorunları hızlıca tespit edebiliyoruz.

Test Otomasyon Araçları Nelerdir?

- Her geçen yıl test dünyasına yepyeni toollar, frameworkler ve araçlar kazandırılıyor. Kaliteli yazılıma erişmek ve bunu sürdürebilmek adına yazılım testine gereken önemin verilmesi gerekir. UI otomasyon toolları cazip ve olmazsa olmazlardandır.
- Nedir bu otomasyon toolları? Selenium, Cypress, Ghost Inspector, WatiN, Appium gibi toolları örnek verebiliriz.

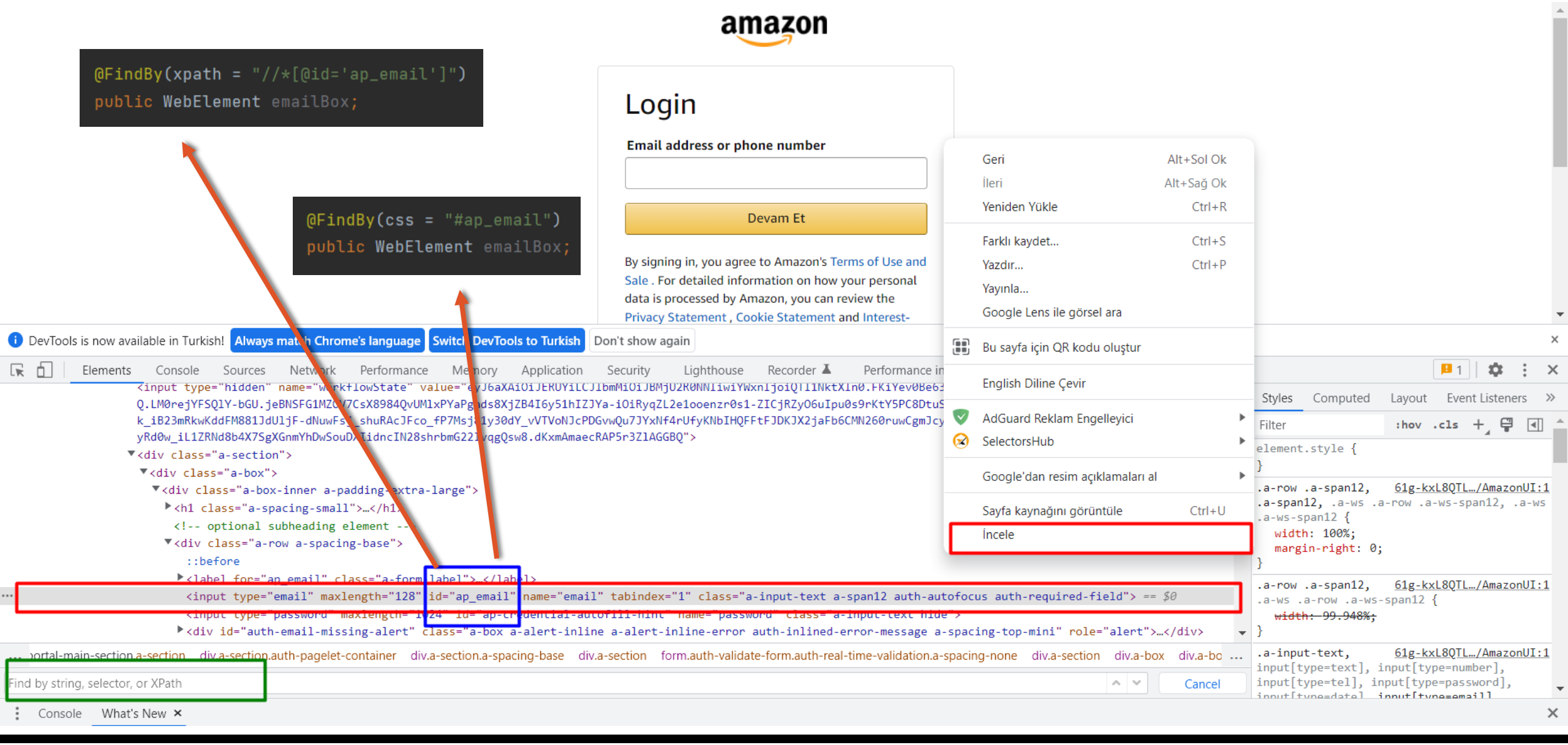


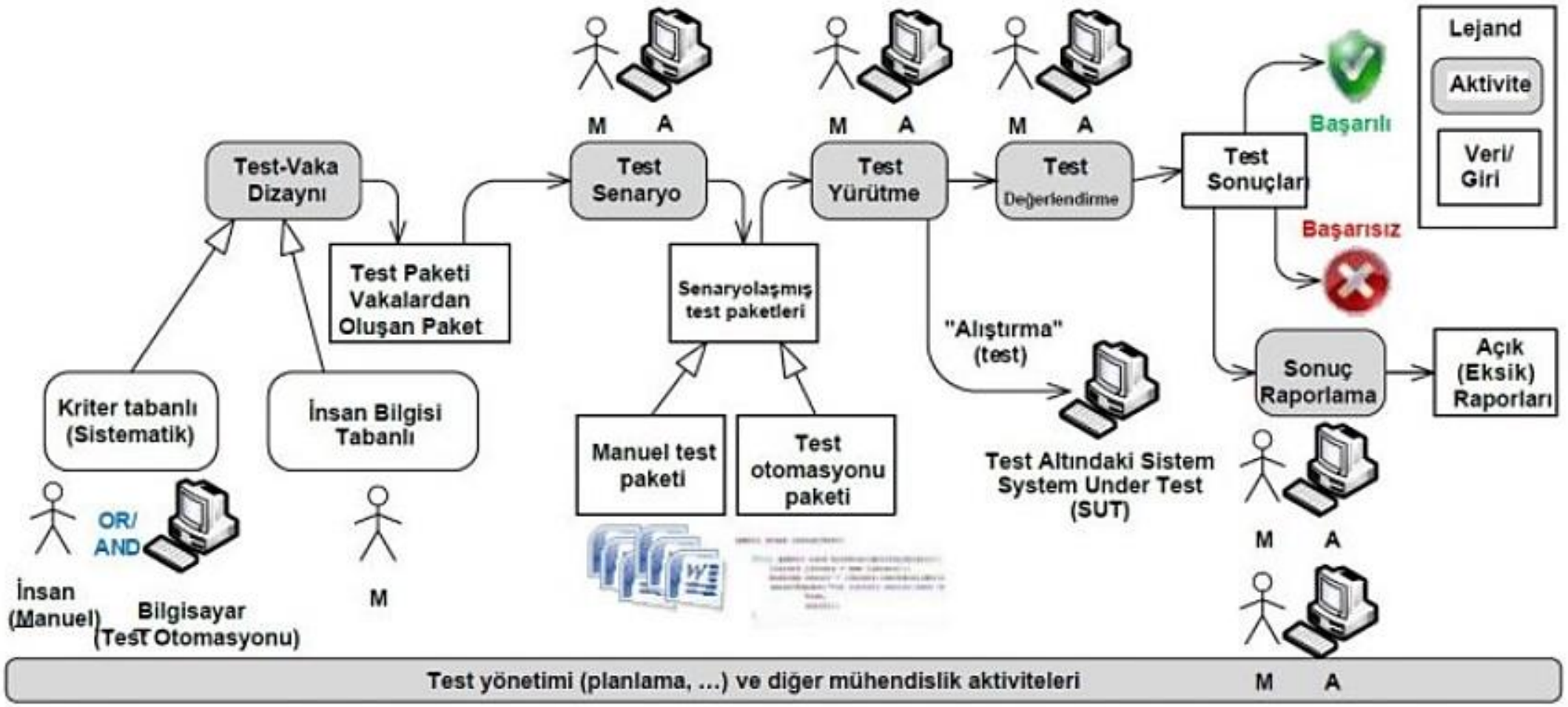
Peki bir web sitesinin test yazılımına uygun dizayn edildiğini anlamak kolay mıdır?

- Çok kolay. Otomasyon kısmında sayfadaki her bir input, textbox, button gibi alanlara element adı veriyoruz ve bu elementleri yakalamak için çeşitli yöntemlerimiz var. CssSelector, Xpath, Id bunlardan birkaçı.

```
@FindBy(xpath = "//*[@id='ap_email']")
public WebElement emailBox;
```

```
@FindBy(css = "#ap_email")
public WebElement emailBox;
```




Lejand
Aktivite
Veri/ Giri
Test-Vaka Dizaynı
Test Paketi Vakalardan Oluşan Paket
Kriter tabanlı (Sistematik)
İnsan Bilgisi Tabanlı
OR/ AND
İnsan (Manuel)
Bilgisayar (Test Otomasyonu)
M
Test Senaryo
Senaryolaşmış test paketleri
Manuel test paketi
Test otomasyonu paketi
Test Yürütme
Test Değerlendirme
"Alıştırma" (test)
Test Altındaki Sistem System Under Test (SUT)
Test Sonuçları
Başarılı
Başarısız
Sonuç Raporlama
Açık (Eksik) Raporları
M A
Test yönetimi (planlama, ...) ve diğer mühendislik aktiviteleri

- **Toparlayacak olursak:**
- Her yiğidin yoğurt yiyişi tabi ki farklıdır. Dokümantasyon konusunda takımdan takıma fark eden gruplama tarzları olabilir. Ekipten ekibe, kullanım farkları olabilir. Tüm bunların yanında global bırakılması gereken kısım ise; bir gereksinim için bir senaryo yazıp altına tüm ekranın testini bir senaryo üzerinde takip edilmemesi kısmıdır.
- Testi daha takip edilebilir, geliştirme ekiplerine daha faydalı hale getirebilmek adına test senaryolarınızı ayırın. Ayrılmış test senaryoları ile, yazılımcı arkadaşlara başarılı ve başarısız testleri aktardığınızda, kodun neresine daha çok çaba harcamaları gerektiğini bile ayırabilirler. Bu durumda da bulgu kümelenmelerini(defect clustering) bile çözen bir yapıya gidebilirsiniz.
- Sizlere tavsiyem, testin bu yapısını üst yönetimlerinize aktarmaya çabalayınız. Bu durumda sizlere test senaryo sayılarını değil, kodun kalitesini sormaya başlarlar. Bu durumda da, kalite için tek başınıza değil; bir ekip olarak çalışır hale gelirsiniz. Bu durumda müşterilerinize ve oradan da tekrar size mutluluk olarak yansır.
- Bol bulgulu günler dilerim. Unutmayın bulunan her bulgu, mutlu bir müşteri demektir.

TEŞEKKÜR EDERİM

 https://github.com/zelihaznk

 https://www.linkedin.com/in/zelihaoznuk/