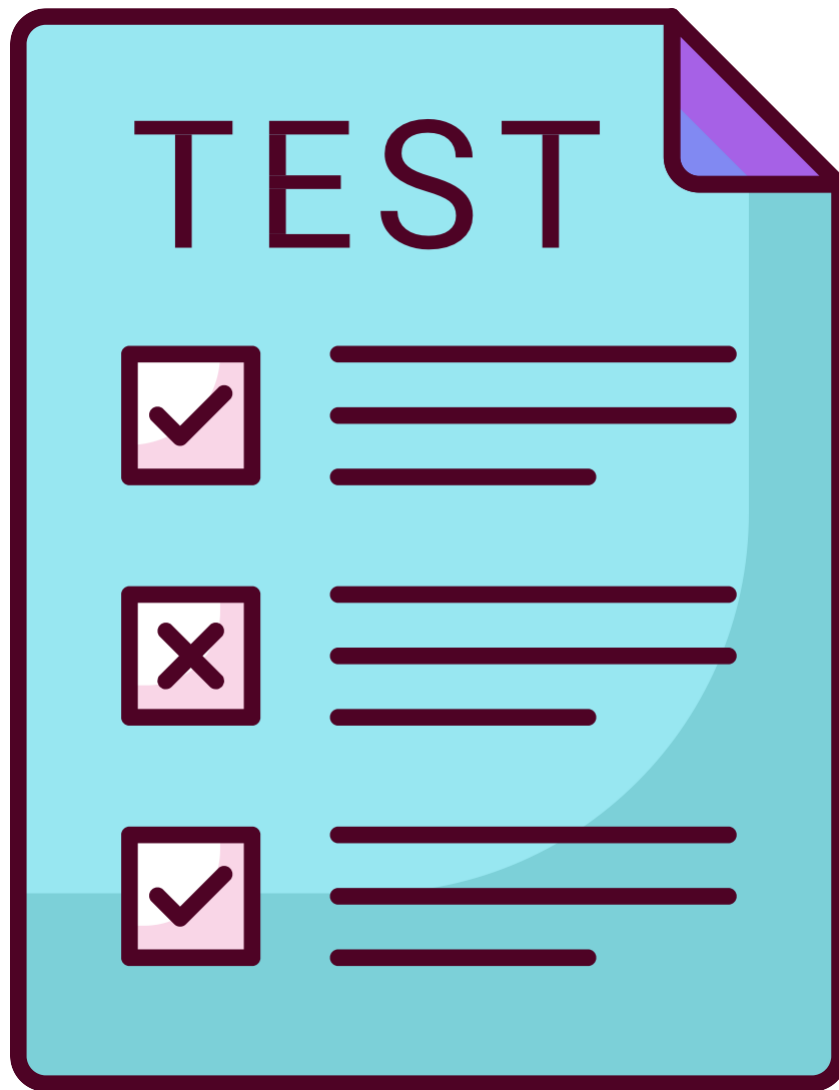




YAZILIM GEÇERLEME VE SINAMA

Black - Box Testi

Black-Box



Black-box testi, yazılımın iç yapısını, kodunu veya algoritmasını dikkate almadan sadece dışarıdan gözlemlenebilir davranışını test etme yöntemidir. Bu test türü, yazılımın kullanıcı tarafından görülen arayüzü ve fonksiyonları üzerinden gerçekleştirilir. Ana odak, yazılımın belirlenen gereksinimleri karşılayıp karşılamadığıdır.

Black-Box Testinin Önemi ve Uygulama Alanları

Black-box testi, yazılımın fonksiyonel doğruluğunu değerlendirmek için önemlidir. Bu test yöntemi, özellikle **kullanıcı arayüzü, sistem entegrasyonları, ve genel kullanıcı deneyimi** gibi alanlarda kritik öneme sahiptir. Kullanıcıların yazılımı nasıl deneyimleyeceğini anlamak ve yazılımın belirtilen gereksinimleri karşılayıp karşılamadığını test etmek için kullanılır.

Neden Black-Box Testi?

- **Kullanıcı Perspektifi:** Testçi, son kullanıcının gözünden yazılıma bakar ve kullanıcı deneyimini doğrudan değerlendirir.
- **Uzmanlık Gerektirmez:** Testçinin yazılımın iç yapısı veya kodlama dilleri hakkında derin bilgi sahibi olması gerekmez.
- **Geniş Kapsamlı Test:** Farklı kullanım senaryolarını ve kullanıcı etkileşimlerini kapsar.

Black- Box Testinin Avantajları ve Sınırlamaları

Avantajları:

- **Kullanıcı Perspektifi:** Yazılımın son kullanıcı tarafından nasıl deneyimleneceğine odaklanır.
- **Esneklik:** Kod bilgisi gerekmediğinden, farklı arka planlardan gelen testçiler tarafından uygulanabilir.

Sınırlamaları:

- **İç Yapıya Erişim Yok:** Yazılımın iç yapısındaki hataları tespit etmek zor olabilir.
- **Kapsamlı Test Zorluğu:** Tüm olası giriş kombinasyonlarını test etmek zaman alıcı ve zor olabilir.

Black- Box Testi Nasıl Yapılır?

1. Gereksinimlerin Analizi:

- İlk adım, yazılımın ne yapması gerektiğini anlamaktır. Bu, yazılımın işlevsel gereksinimleri, kullanıcı hikayeleri ve kullanım senaryoları üzerinden yapılır.

2. Test Senaryolarının Hazırlanması:

- Yazılımın beklenen davranışlarını temsil eden test senaryoları oluşturulur. Bu senaryolar, yazılımın farklı özelliklerini ve işlevlerini kapsar.

Black- Box Testi Nasıl Yapılır?

3. Testlerin Yürütülmesi:

- Test senaryoları, yazılım üzerinde uygulanır. Testçiler, belirlenen girişleri sağlar ve yazılımın çıktılarını gözlemler.

4. Sonuçların Değerlendirilmesi:

- Elde edilen sonuçlar, beklenen sonuçlarla karşılaştırılır. Bu, yazılımın gereksinimleri karşılayıp karşılamadığını belirlemek için yapılır.

5. Raporlama ve Geri Bildirim:

- Test sonuçları raporlanır ve bulunan sorunlar, geliştirme ekibine geri bildirim olarak sunulur.

Black-Box Testi Türleri

1. Fonksiyonel Test
2. Kullanılabilirlik Testi
3. Arayüz Testi
4. Entegrasyon Testi

Black- Box Testi Türleri

1. Fonksiyonel Test

- Fonksiyonel test, yazılımın belirli işlevlerinin, tanımlanan gereksinimlere ve beklentilere uygun olarak çalışıp çalışmadığını kontrol eder.

Özellikleri:

- **Test Senaryoları:** İşlevsellikle ilgili spesifik gereksinimler temel alınarak test senaryoları oluşturulur.
- **Girdi/Çıktı Kontrolü:** Her işlev için beklenen girdi ve çıktılar belirlenir ve yazılımın bu beklentilere uyup uymadığı test edilir.
- **Hata Ayıklama:** İşlevlerin yanlış veya beklenmedik sonuçlar vermesi durumunda bu durumlar tespit edilir.

Black- Box Testi Türleri

Uygulama Alanları:

- Bir web uygulamasındaki kayıt formunun çalışması, bir alışveriş sitesindeki ödeme sisteminin doğruluğu gibi spesifik işlevlerin testi.

Black- Box Testi Türleri

2. Kullanılabilirlik Testi

- Kullanılabilirlik testi, yazılımın kullanım kolaylığına ve kullanıcı deneyimine odaklanır.

Özellikleri:

- **Kullanıcı Deneyimi:** Yazılımın ne kadar sezgisel ve kullanıcı dostu olduğunun değerlendirilmesi.
- **Erişilebilirlik:** Farklı kullanıcı grupları için yazılımın erişilebilirliğinin test edilmesi.
- **Gerçek Kullanıcı Geri Bildirimleri:** Gerçek kullanıcıların deneyimlerini taklit eden senaryolar kullanılır.

Black- Box Testi Türleri

Uygulama Alanları:

- Bir mobil uygulamanın navigasyonunun kolaylığı, web sitelerindeki bilgi erişiminin basitliği gibi kullanıcı arayüzü ve deneyimi ile ilgili özellikler.

Black- Box Testi Türleri

3. Arayüz Testi

- Arayüz testi, kullanıcı arayüzünün ve etkileşimlerinin doğru şekilde çalışıp çalışmadığını kontrol eder.

Özellikleri:

- **Arayüz Elemanları:** Düğmeler, menüler, metin kutuları gibi arayüz elemanlarının doğru çalışıp çalışmadığına bakılır.
- **Hata Mesajları:** Yanlış veya eksik girdilerde uygun hata mesajlarının gösterilip gösterilmediği test edilir.
- **Grafik ve Tasarım Öğeleri:** Görsel öğelerin düzgün yüklenip yüklenmediği ve doğru gösterilip gösterilmediği kontrol edilir.

Black- Box Testi Türleri

Uygulama Alanları:

- Web sitelerindeki form alanlarının doğru şekilde veri toplaması, uygulamalardaki geçiş efektlerinin düzgünlüğü gibi arayüzle ilgili detaylar.

Black- Box Testi Türleri

4. Entegrasyon Testi

- Entegrasyon testi, yazılımın diğer sistemlerle veya modüllerle olan etkileşimini ve entegrasyonunu test eder.

Özellikleri:

- **Sistemler Arası Etkileşim:** Farklı yazılım sistemleri veya modülleri arasındaki veri akışı ve iletişimin doğru olup olmadığını kontrol eder.
- **Bağımlılıkların Testi:** Bir modülün veya sistemin diğerleri üzerindeki etkilerinin değerlendirilmesi.
- **Arayüzlerin Uyumu:** Farklı sistemler arasındaki arayüzlerin birbiriyle uyumlu çalışıp çalışmadığına bakılır.

Black- Box Testi Türleri

Uygulama Alanları:

- Bir CRM sisteminin şirketin e-posta sunucusuyla entegrasyonu, e-ticaret sitesinin ödeme geçidiyle olan bağlantısının doğruluğu gibi sistemler arası etkileşimler.

Örnek Vakalar ve Senaryolar

Gerçek dünya örnekleri ve test senaryoları, black-box testinin uygulamalarını ve önemini daha iyi anlamak için kullanılır. Örneğin, bir e-ticaret web sitesinde alışveriş sepeti işlevinin test edilmesi, bu tür bir testin tipik bir örneğidir.

Örnek Senaryolar 1

Örnek olarak, bir e-ticaret web sitesinin "Ürün Arama" özelliğini test edeceğiz.

Senaryo: E-Ticaret Sitesinde Ürün Arama Testi

1. Testin Amacı: Web sitesinin "Ürün Arama" özelliğinin doğru ve etkin bir şekilde çalıştığını doğrulamak.

2. Test Adımları:

a. Girdi Hazırlama:

Aranacak örnek ürün isimleri (örneğin, "kitap", "telefon", "ayakkabı").

Hatalı veya var olmayan ürün isimleri (örneğin, "xyz123", " ", özel karakterler).

b. Test Senaryoları:

- **Senaryo 1:** Geçerli bir ürün ismi girildiğinde, ilgili ürünlerin listelenmesi.
- **Senaryo 2:** Geçersiz veya var olmayan bir ürün ismi girildiğinde, uygun bir hata mesajının gösterilmesi veya sonuç bulunamadığının belirtilmesi.
- **Senaryo 3:** Boş bir arama yapıldığında, uygun bir uyarı mesajı veya genel ürün listesinin görüntülenmesi.

c. Test İşlemleri:

- Web sitesine girilir.
- Arama kutusuna çeşitli girdiler (geçerli/geçersiz) yazılır ve arama butonuna basılır.
- Her senaryo için beklenen sonuçlar gözlemlenir.

3. Beklenen Sonuçlar:

- **Senaryo 1 için Beklenen Sonuç:** Geçerli bir ürün ismi girildiğinde, ilgili ürünlerin listelendiği ve arama sonuçlarının uygun olduğu gözlemlenir.
- **Senaryo 2 için Beklenen Sonuç:** Geçersiz veya var olmayan bir ürün ismi girildiğinde, "Ürün bulunamadı" tarzında bir mesajın veya uygun bir hata mesajının gösterilmesi.
- **Senaryo 3 için Beklenen Sonuç:** Boş bir arama yapıldığında, "Arama terimi girin" gibi bir uyarı mesajı veya genel ürün listesinin görüntülendiği görülür.

4. Testin Değerlendirilmesi:

- Her bir senaryo için gözlemlenen sonuçlar, beklenen sonuçlarla karşılaştırılır.
- Eğer tüm senaryolar beklenen sonuçları veriyorsa, test başarılı kabul edilir.
- Aksi takdirde, bulunan sorunlar raporlanır ve geliştiricilere bildirilir.

Örnek Senaryolar 2

Basit Bir Hesaplama Fonksiyonu

Diyelim ki bir hesaplama fonksiyonumuz var. Bu fonksiyon, iki sayıyı alıp toplamını döndürüyor. Black-box testi yaparken, bu fonksiyonun doğru çalışıp çalışmadığını anlamak için çeşitli giriş değerleri ile test ederiz.

Örnek Senaryolar 2

```
def add(a, b):  
    return a + b
```

Black-Box Test Senaryoları:

1. Normal Durum Testi:

- **Giriş:** add(2, 3)
- **Beklenen Çıktı:** 5
- **Amaç:** Fonksiyonun normal şartlar altında doğru çalışıp çalışmadığını test etmek.

2. Sıfır Değeri Testi:

- **Giriş:** `add(0, 5)`
- **Beklenen Çıktı:** 5
- **Amaç:** Bir girişin sıfır olduğu durumda fonksiyonun doğru çalışıp çalışmadığını kontrol etmek.

3. Negatif Sayı Testi:

- **Giriş:** `add(-2, 3)`
- **Beklenen Çıktı:** 1
- **Amaç:** Negatif sayılarla fonksiyonun doğru çalışıp çalışmadığını test etmek.

4. Büyük Sayılar Testi:

- **Giriş:** `add(1000, 2000)`
- **Beklenen Çıktı:** 3000
- **Amaç:** Büyük sayılarla fonksiyonun doğru çalışıp çalışmadığını kontrol etmek.

5. Tip Uyuşmazlığı Testi:

- **Giriş:** `add("a", "b")`
- **Beklenen Çıktı:** Tip hatası veya uygun olmayan bir çıktı
- **Amaç:** Yanlış veri tipleri ile fonksiyonun nasıl davrandığını görmek.

Örnek Senaryolar 2

Bu test senaryoları, fonksiyonun farklı durumlar altında nasıl davrandığını anlamak için kullanılır. Black-box testi yaparken, fonksiyonun iç yapısını veya kodunu bilmek gerekmez; sadece dışarıdan gözlemlenebilir davranışlarına odaklanılır.

Black-Box Test Teknikleri

1. Denklik Paylarına Ayırma - Eşdeğer Aralık Testi (Equivalence Partitioning)
2. Sınır Değer Analizi Testi (Boundary Value Analysis)
3. Karar Tablosu Testi (Decision Table Testing)
4. Durum Geçiş Testi (State Transition Testing)
5. Çiftler Arası Test (Pairwise Testing)
6. Kullanım Durumu Testi (Use Case Testing)

Eşdeğer Aralık Testi

Tekniğin arkasındaki fikir, bir dizi test koşulunu aynı kabul edilebilecek gruplara veya kümelere bölmektir. Bölümlendirme genellikle girdiler, çıktılar, dahili değerler, zamanla ilgili değerler ve arayüz parametrelerini içeren test nesneleri için gerçekleştirilir. Maksimum test kapsamıyla minimum test senaryoları oluşturulur. Bu teknik, özellikle geniş bir girdi değeri aralığıyla uğraşırken kullanışlıdır.

Eşdeğer Aralık Testi

- Belirli varsayımlar üzerine çalışır:
 - Sistem, bir aralık içindeki tüm test giriş varyasyonlarını aynı şekilde ele alacaktır.
 - Giriş koşullarından biri geçerse, aralık içindeki diğer tüm giriş koşulları da geçecektir.
 - Giriş koşullarından biri başarısız olursa, aralık içindeki diğer tüm giriş koşulları da başarısız olacaktır.

Eşdeğer Aralık Testi

- İki test aşağıdaki durumlarda eşdeğer kabul edilir:
 - Aynı şeyi test ederler (fonksiyon modülü, sistemin bir parçası).
 - Testlerden biri bir hatayı yakalarsa, diğerinin de onu yakalama olasılığı yüksektir.
 - Bunlardan biri hatayı yakalamazsa, diğerinin de onu yakalama olasılığı düşüktür.

Eşdeğer Aralık Testi

Çok fazla eşdeğerlik sınıfı, birden fazla testin gereksiz (aşırı) olma olasılığını artırır.

Çok az eşdeğerlik sınıfı, hatanın atlanma olasılığını artırır.

Eşdeğer Aralık Testi

Test senaryosu tasarımı 2 adımda gerçekleştirilir:

- Eşdeğerlik sınıflarını belirleme

Giriş, çıkış: gereksinimden ipucu

Geçerli sınıflar: kabul edilebilir giriş değerleri

Geçersiz sınıflar: kabul edilemez giriş değerleri

- Her eşdeğerlik sınıfından bir temsilci seçme
- Test vakalarını tanımlama

Eşdeğer Aralık Testi

Örnek: Sipariş sayfasındaki ‘Miktar’ alanı için gereklilik: Miktar alanı sayısal bir alandır, yalnızca tam sayı (0-255) içermelidir, alfabetik (A - Z), (a-z) ve özel karakterler içermemelidir.

Test senaryoları:

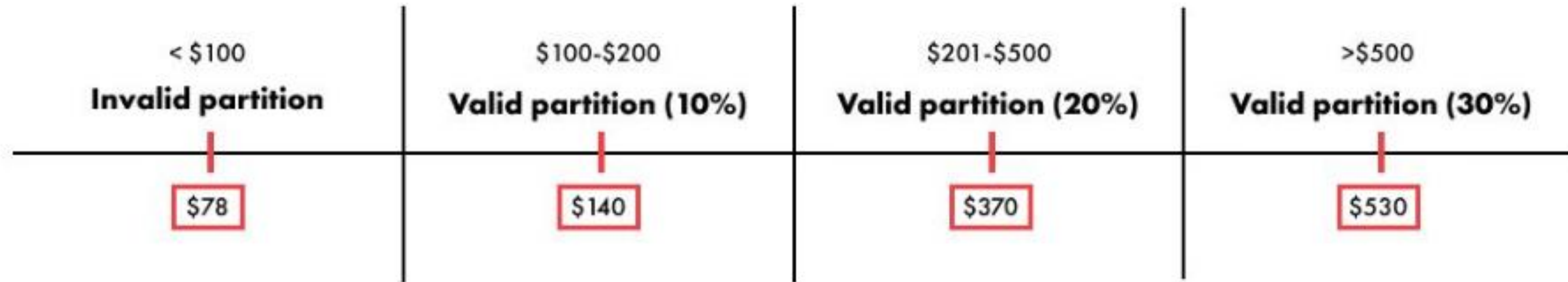
1. -5 >> Başarısız
2. 50 >> Başarılı
3. 330 >> Başarısız
4. Kelime >> Başarısız
5. %(!@ >> Başarısız

Eşdeğerlik Sınıfları	Durum
<0	geçersiz
0-255	geçerli
>255	geçersiz
kelime	geçersiz
özel karakter	geçersiz

Eşdeğer Aralık Testi

Örnek: İndirim, alışveriş sepetinin toplam tutarına bağlı olarak hesaplanır. Toplam tutar 100-200\$ aralığındaysa indirim %10, toplam tutar 201-500\$ aralığındaysa %20 ve toplam tutar 500\$'dan fazlaysa indirim %30'dur. Bu senaryoda, 100\$'ın altındaki tutar için bir geçersiz bölüm ve üç geçerli bölüm belirleyebiliriz.

Eşdeğer Aralık Testi



Eşdeğer aralık testini uygulamak için her bölümden bir değer alabiliriz: ilk geçerli bölümden 140\$, ikinci geçerli bölümden 370\$, üçüncü geçerli bölümden 530\$ ve geçersiz bölümden 78\$. Şimdi dört test senaryonuz var ve tüm tanımlı bölümler kapsandığı için %100 kapsama ulaştık.

Eşdeğer Aralık Testi

Eşdeğer aralık testi, test senaryolarını belirlemek için tek başına bir yöntem değildir. Sınır değer analiziyle desteklenmesi gerekir.

Sınır Değer Analizi Testi

Bu teknik eşdeğer aralık yönteminin bir uzantısıdır ve geçerli ve geçersiz aralıkların sınırlarını test etmek için kullanılır. Bu sınırlar, yanlış olma olasılığı en yüksek olan kırılma noktalarıdır. Ayrıca, bu iki kara kutu test tekniği - eşdeğer aralık ve sınır değer analizi - birleştirilebilir. Sınırın minimum ve maksimum değerlerini test etmek için iki değerli sınır analizi; sınırdan önceki, sınırdaki ve sınırın hemen üzerindeki değerleri test etmek için üç değerli sınır analizi kullanılır.

Sınır Değer Analizi Testi

Invalid partition	Valid partition (10%)	Valid partition (20%)	Valid partition (30%)
\$99	\$100	\$200	\$201
		\$500	\$501

İki değerli sınır analizini uygulamak için her sınırın minimum ve maksimum değerini test edeceğiz: 99\$, 100\$, 200\$, 201\$, 500\$ ve 501\$. Altı test senaryosu var ve tüm tanımlanmış sınırlar kapsandığı için bunları test ederek %100 kapsama ulaştık.

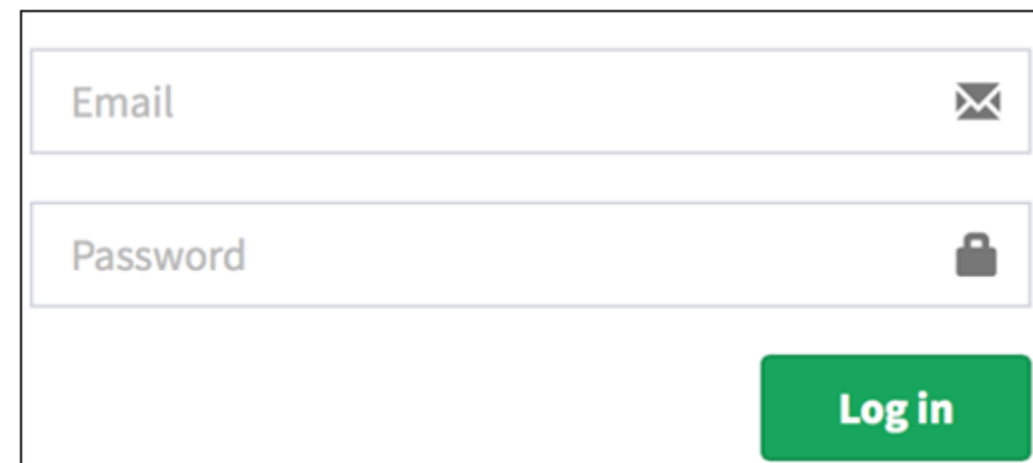
Karar Tablosu Testi

Bu test tasarım tekniği tanımlanmış gereksinim ya da iş kurallarına bağlı olarak testi yapılacak koşulların sayısının fazla olduğu durumlarda tercih edilir. Bu koşullar çok fazla olduğunda mevcut durum için bir tablo oluşturulur ve bu durumlar sonucunda çıkacak sonuçlar tabloya yerleştirilir. Test edilecek koşulların fazla olması halinde mevcut durum bir tablo üzerine aktarılarak karar tablosu oluşturulur. Her bir duruma karşı yazılım vereceği cevaplar tabloya yerleştirilir.

Karar Tablosu Testi

Örnek: Giriş ekranı için bir karar tablosu oluşturalım.

Tüm olası koşulların sayısını bulmak için test cihazı 2^n formülünü kullanır; burada n girdi sayısını belirtir.



A diagram of a login form. It consists of a rectangular container with a thin black border. Inside, there are two input fields stacked vertically. The top field is labeled 'Email' and has an envelope icon on its right side. The bottom field is labeled 'Password' and has a padlock icon on its right side. Below these fields, on the right side of the container, is a green rectangular button with the text 'Log in' in white.

Karar Tablosu Testi

Kullanıcı doğru kullanıcı adı ve şifreyi girerse ana sayfaya yönlendirilecektir. Girdilerden herhangi biri yanlışsa bir hata mesajı görüntülenecektir.

D - Kullanıcı adı/Şifre doğru

Y - Kullanıcı adı/Şifre yanlış

Durum 1: Kullanıcı adı/Şifre doğru - Ana sayfaya yönlendirme

Durum 2: Kullanıcı adı yanlış şifre doğru - Hata mesajı

Durum 3: Kullanıcı adı doğru şifre yanlış - Hata mesajı

Durum 4: Kullanıcı adı/Şifre yanlış - Hata mesajı

Karar Tablosu Testi

Koşullar	Durum 1	Durum 2	Durum 3	Durum 4
1. Kullanıcı adı	D	Y	D	Y
2. Şifre	D	D	Y	Y
Eylemler				
Ana sayfaya yönlendirme (Kullanıcı adı/Şifre doğru)	X			
Hata mesajı (Kullanıcı adı yanlış Şifre doğru)		X		
Hata mesajı (Kullanıcı adı doğru Şifre yanlış)			X	
Hata mesajı (Kullanıcı adı/Şifre yanlış)				X

Durum Geçiş Testi

Durum geçiş testi, bir sistemin girişe göre farklı durumlar arasında doğru şekilde geçiş yaptığını doğrulamak için kullanılan bir tekniktir. Tüm olası durumları, geçişleri ve olayları haritalamak için bir durum geçiş diyagramı veya tablosu oluşturmayı içerir. Daha sonra test senaryoları, sistemin tanımlanmış kurallara göre durumlar arasında doğru şekilde geçiş yaptığını doğrulamak için tasarlanır.

Durum Geiş Testi

Bu teknik, iş akışı sistemleri, kullanıcı arayüzleri veya karmaşık uygulamalar gibi birden fazla durum ve geişe sahip sistemler için özellikle yararlıdır. Sistemin bir durumdan diğerine geerken beklendiğı gibi davranmasını sağlar.

Durum Geiř Testi

Örnek: Çevrimiçi alışveriş uygulaması olabilir. Bu uygulamada, ilk aşamada alışveriş sepeti boştur, müşteriler sepete ürün ekler, hesabına giriş yapar, ödeme işlemine geçer, ödeme ve gönderim ayrıntılarını girer ve siparişlerini onaylar.

Bu geçişleri test etmek, sistemin durum değişikliklerini doğru şekilde ele almasını ve alışveriş sürecinin farklı aşamalarında tutarlı davranışı sürdürmesini sağlar.

Çiftler Arası Test

Bir yazılım uygulamasının çıktısı birçok faktöre bağlıdır, örneğin giriş parametreleri, durum değişkenleri ve ortam yapılandırmaları. Sınır değer analizi ve eşdeğer aralık analizi gibi teknikler, bireysel faktörler için olası değerleri belirlemekte yararlı olabilir. Ancak tüm bu faktörler için tüm olası değer kombinasyonlarını test etmek pratik değildir. Bu nedenle bunun yerine tüm faktörleri tatmin edecek bir kombinasyon alt kümesi üretilir.

Çiftler Arası Test

Birden fazla parametre içeren uygulamalar için test senaryosu tasarlamakta oldukça faydalıdır. Testler, bir sisteme her bir giriş parametresi çifti için, bu parametrelerin tüm olası ayrı kombinasyonları olacak şekilde tasarlanır.

Dikkatlice seçilmiş test senaryoları kullanılarak, giriş parametre çiftlerinin testlerini "paralelleştirerek" sistemi test etmek, tüm parametrelerin tüm kombinasyonlarının kapsamlı test edilmesinden çok daha hızlı yapılabilir.

Kullanım Durumu Testi

Kullanım Durumu Testi, bir kullanıcının yazılımla sahip olabileceği tüm olası senaryoları veya etkileşimleri belirlemeyi ve belgelemeyi içerir. Bu, sistemin davranışını bir son kullanıcı perspektifinden anlamaya yardımcı olur ve tüm işlevlerin kapsamlı bir şekilde test edilmesini sağlar. Ayrıntılı kullanım durumları oluşturarak, test edenler neyin test edilmesi gerektiğine dair net bir yol haritasına sahip olabilir ve bu da kritik işlevleri kaçırma olasılığını azaltır.

Kullanım Durumu Testi

Örneğin, yeni bir mobil bankacılık uygulamasını test ettiğinizi düşünün. Kullanım durumu testi ile, işlem yapan bir müşteri veya hesapları yöneten bir yönetici gibi farklı kullanıcıların yerine geçeceksiniz. Oturum açmaktan para transferine ve kişisel bilgileri güncellemeye kadar tüm hareketleri yapacaksınız.

Kalite Güvencesi ve Risk Değerlendirmesi

Black-box testi, yazılımın kalite güvencesinin önemli bir parçasıdır. Kullanıcı ihtiyaçlarını ve beklentilerini karşılayıp karşılamadığını değerlendirir ve potansiyel riskleri ortaya çıkarabilir.

Yazılım Geliştirme Yaşam Döngüsü ile İlişkisi

Yazılımın geliştirme sürecinin çeşitli aşamalarında, özellikle de son safhalarda uygulanır. Bu, yazılımın piyasaya sürülmeden önce kararlı ve kullanıcı beklentilerine uygun olduğunu doğrulamak için kritik öneme sahiptir.