CloudTest Öğrenme Notları (Learning CloudTest Notes)

1. CloudTest Nedir? (What is CloudTest?)

CloudTest, Huawei'nin CodeArts TestPlan aracıyla sunduğu, test süreçlerini bulut tabanlı (cloud-based) yönetmeye imkan tanıyan bir platformdur. Genel olarak "cloud testing" kavramı, yazılım testlerinin fiziksel cihazlar yerine bulut ortamında çalıştırılmasını ifade eder.

Böylece, farklı işletim sistemleri, cihaz türleri, tarayıcılar ve ağ koşullarında uygulamanın nasıl davrandığını gözlemleme fırsatı verir. Kendi bilgisayarımda sınırlı test yapmak yerine, dünyanın farklı bölgelerindeki cihazlara bağlanarak uygulamayı test etmek mümkün olur.

Amacı (Purpose):

- Donanım yatırımı yapmadan, çok geniş test senaryolarını gerçekleştirebilmek.
- Test süreçlerini hızlandırmak ve standartlaştırmak.
- CI/CD pipeline'larıyla uyumlu bir şekilde, otomatik testleri yazılım geliştirme sürecine entegre etmek.

Sağladığı Avantajlar (Advantages):

- **Hız ve Esneklik (Speed & Flexibility):** Aynı anda onlarca hatta yüzlerce cihazda paralel test yapılabilir.
- **Maliyet Avantajı (Cost Efficiency):** Donanım satın alma zorunluluğu ortadan kalkar, yalnızca kullanılan kaynak için ödeme yapılır (pay-as-you-go modeli).
- **Gerçek Cihaz ve Lokasyon Testleri (Real Device & Location Testing):** Uygulamanın Türkiye, Almanya veya Amerika gibi farklı bölgelerde nasıl çalıştığı görülebilir.
- **CI/CD Entegrasyonu (CI/CD Integration):** Pipeline içerisine eklenerek her deploy sonrası otomatik test çalıştırılabilir.
- **Uyumluluk ve Güvenlik Testleri (Compatibility & Security Testing):** Performans, enerji tüketimi, uyumluluk ve güvenlik açıkları sistem tarafından raporlanır.

Kullanım Alanları (Use Cases):

- Mobil uygulama testleri (Mobile App Testing)
- Web uygulama testleri (Web App Testing)
- API testleri (API Testing)
- Yük ve performans testleri (Load & Performance Testing)

Benim anladığım kadarıyla CloudTest'in en büyük gücü, yerel test ortamı kurmaya gerek kalmadan çok çeşitli senaryoyu aynı platformda deneyebilmek. Bu durum özellikle büyük projelerde zaman ve maliyet açısından ciddi avantaj sağlıyor.

---

2. Test Case Şablonu (Test Case Template) Nedir?

Test case şablonu, test senaryolarının standart bir form üzerinden yazılmasını sağlar. Yazılım testi sürecinde her kişinin aynı formatı kullanması, hem okunabilirliği hem de tekrar kullanılabilirliği artırır.

Şablon İçeriği (Template Fields):

- **Kimlik (ID):** Her test için benzersiz bir kimlik (örneğin TC_01).
- **Başlık (Title):** Testin amacı → "Kullanıcı doğru şifreyle giriş yapabilmeli".
- **Açıklama (Description):** Senaryonun kapsamı ve hedefi.
- **Ön Koşullar (Preconditions):** Testin başlayabilmesi için gerekli durumlar.
- **Adımlar (Test Steps):** Yapılması gereken işlemler.
- **Beklenen Sonuç (Expected Result):** Testin başarılı sayılması için sağlanması gereken çıktı.
- **Öncelik (Priority):** High / Medium / Low.
- **Son Koşullar (Postconditions):** Test sonrası sistemin durumu.

Neden Standart Kullanılır (Why Standard?):

- Herkesin aynı formatta yazması iletişim kolaylığı sağlar.
- Eksik veya karmaşık senaryoların önüne geçilir.
- Testlerin tekrar edilebilirliği ve kalite kontrolü artar.

Benim bakış açımla: Bu şablon, sadece yazım düzeni değil, aslında ekip içinde ortak bir dil kurmak için de kritik. Çünkü testin okunurluğu, yazılım geliştirme sürecindeki diğer ekip üyeleri için de önem taşıyor.

---

3. Test Case Kontrol Listesi (Test Case Checklist) Analizi

Test case yazıldıktan sonra onun kaliteli olup olmadığını ölçmek için bir kontrol listesi kullanılır. Bu liste, hem testin amacına uygun yazılıp yazılmadığını hem de farklı senaryoları kapsayıp kapsamadığını gösterir.

Checklist Soruları (Checklist Questions):

- Senaryo net mi, açık ve anlaşılır mı (Clear)?
- Acceptance Criteria ile uyumlu mu (Aligned with Acceptance Criteria)?
- Pozitif (Positive) ve negatif (Negative) test senaryoları var mı?
- Tek bir fonksiyonu mu test ediyor (Single Purpose) yoksa çok mu karmaşık?
- Ön koşullar ve beklenen sonuç açıkça yazılmış mı?

Pozitif ve Negatif Senaryo (Positive & Negative Scenarios):

- Pozitif: Kullanıcı doğru şifre girdiğinde giriş yapılabilmeli.
- Negatif: Yanlış şifre girildiğinde sistem kullanıcıyı bilgilendirmeli.

Buradan çıkardığım şey şu: Kaliteli bir test case sadece başarılı senaryoyu değil, olası hataları da kapsamalı. Çünkü çoğu hata zaten "beklenmeyen senaryolar"dan doğuyor.

---

4. Yazılmış Test Case'leri CloudTest'e Aktarmak (Importing Test Cases into CloudTest)

Çoğu zaman test case'ler Excel ya da CSV dosyalarında tutulur. CloudTest, bu dosyaları içeri aktararak (import) test case'lerin otomatik oluşmasını sağlar.

Import Süreci (Import Process):

1. CloudTest arayüzünde "Import Test Cases" seçilir.
2. Dosya formatı (CSV/Excel) belirlenir.

3. **Eşleştirme (Mapping):** Dosyadaki sütunlar CloudTest alanlarıyla bağlanır.
    - Excel “Case Title” → CloudTest “Title”
    - Excel “Expected Result” → CloudTest “Expected Result”
4. Dosya yüklenir ve sistem test case’leri otomatik olarak oluşturur.

Avantajı (Benefit):

- Yüzlerce test case’i manuel girmek yerine birkaç dakika içinde yüklenebilir.
- Zaman tasarrufu sağlar ve hata payını azaltır.

Bu özellik bana şunu gösteriyor: Büyük ölçekli projelerde manuel iş yükünü azaltmak için bu tarz otomasyon adımları kritik önem taşıyor.

---

5. Test Case’leri Requirement ile İlişkilendirmek (Associating Test Cases with Requirements)

CloudTest’in güçlü yanlarından biri de test case ↔ requirement ilişkisidir. Bu sayede hangi gereksinimin hangi test ile doğrulandığı net bir şekilde takip edilebilir.

Nasıl Yapılır (How?):

- Requirement (örn: “Kullanıcı giriş yapabilmeli”) seçilir.
- İlgili test case bu requirement’a bağlanır.
- Böylece **Traceability Matrix (İzlenebilirlik Matrisi)** oluşur.

Avantajları (Benefits):

- Gereksinimlerin doğrulandığı net şekilde görülür.
- Eksik testler kolayca fark edilir.
- Regülasyonlu alanlarda (finans, sağlık vb.) yasal olarak zorunlu olan izlenebilirlik sağlanır.

Benim için çıkarım: Bu özellik, QA süreçlerini daha şeffaf ve ölçülebilir hale getiriyor. Ayrıca ekipler arası iletişimde “bu gereksinim test edildi mi?” sorusuna direkt cevap verilebilmesini sağlıyor.

---

6. Test Görevi (Test Task) Oluşturmak ve Test Case’leri Çalıştırmak (Creating Test Tasks & Executing Test Cases)

Test Task Nedir (What is a Test Task?):
Test case’lerin çalıştırılabilmesi için CloudTest üzerinde oluşturulan görevdir.

Nasıl Oluşturulur (Steps):

1. CloudTest’te “New Test Task” seçilir.
2. Göreve isim verilir (örneğin: “Login Module Tests”).
3. Çalıştırılacak test case’ler eklenir.
4. Test ortamı (cihaz, tarayıcı, lokasyon) belirlenir.
5. Çalıştırma başlatılır.

Çalıştırma ve Raporlama (Execution & Reporting):

- Testler manuel ya da otomatik çalıştırılabilir.
- Çıktılar: Başarılı (Pass) / Başarısız (Fail).
- Başarısız adımlar için log ve ekran görüntüleri kaydedilir.
- Sonuçlar Excel/PDF olarak indirilebilir.

Burada benim öğrendiğim şey: Sadece test case yazmak değil, onları organize etmek ve çıktısını alabilmek de sürecin tamamlayıcı bir parçası.

Bugün Öğrendiklerimin Özeti (Summary of Today’s Learning)

Bugün, CloudTest üzerine detaylı bir araştırma yaptım. Platformu doğrudan deneyimleyemedim çünkü ücretli bir servis, fakat resmi dokümanları ve kaynakları inceleyerek teorik bilgiyi edindim.

Test case yazımında şablon kullanmanın yalnızca format açısından değil, ekip içinde ortak bir dil kurma açısından da önemli olduğunu kavradım. Checklist mantığının kalite kontrol sürecinde kritik rol oynadığını gördüm. Excel/CSV’den test case import etme özelliği sayesinde büyük projelerde manuel iş yükünün nasıl azaltılabileceğini fark ettim.

Requirement ↔ Test Case ilişkisinin QA süreçlerinde izlenebilirlik sağladığını ve raporlama sürecini çok daha şeffaf hale getirdiğini teorik düzeyde anladım. Ayrıca test task oluşturma ve çalıştırma süreçlerini adım adım öğrendim; uygulama yapamasam bile mantığını kavramak benim için faydalı oldu.

Sonuç olarak: Bugün pratik deneyimim olmadı ama teorik düzeyde yazılım test süreçlerini uçtan uca nasıl planlayabileceğimi, hangi noktada hangi aracın devreye girdiğini ve neden önemli olduğunu öğrenmiş oldum. Gelecekte uygulama fırsatı bulduğumda, bu teorik temel sayesinde çok daha bilinçli ilerleyebileceğim.