**SDLC & Agile**

**Test Tipleri**

SDlc’de, Software testing, yazılım kalitesini doğrulamak için en sık kullanılan bir tekniktir. Black-box testing gecerlilik için yapılır. (i.e. are we building the right software?) and white box testing genellikle doğrulama için yapılır (i.e. are we building the software right?).

* **The black box** testi aynı zamanda Functional testing olarak da adlandırılır ve verilen bilgilere dayanarak test senaryoları tasarlayan bir teknik. Black box testing sistemin iç yapısının bilinmesine gerek duyulmadan icra edilen, çoğunlukla test mühendisleri tarafından yapılan bir **test** tekniğidir.
* **White box** testing Beyaz kutu testi, işlevselliğinin aksine bir uygulamanın iç yapılarını veya çalışmalarını test eden bir yazılım testi yöntemidir. Beyaz kutu testinde, test senaryolarını tasarlamak için sistemin iç perspektifinin yanı sıra programlama becerileri de kullanılır
* **Unit testing** code temelli developerlar tarafından yapılan testlerdir. her birimi ayrı ayrı test etmek için yapılır. bu test genellikle küçük code birimleri için yapılır ve classdan daha büyük değildir.
* **Integration testing** unit testinden sonra küçük birimlerin bir grup halinde çalışabilmesi için yapılan testir. Integration testing (Entegrasyon testinin) amacı, uygulamanın farklı bileşenlerinin hala müşteri gereksinimlerine göre çalışmasını sağlamaktadır. Birim testi tamamlandıktan sonra entegrasyon testi başlar.
* **System testing** Sistem testi, Test Ekibi tarafından gerçekleştirilen kara kutu testidir ve sistem testinin başlangıcında tüm sistem kontrollü bir ortamda yapılandırılır. Sistem testinin amacı, bir uygulamanın tasarlandığı gibi işlevleri gerçekleştirirken doğruluğunu ve eksiksizliğini doğrulamaktır.
* **User Acceptance testing** Son kullanıcının veya müşterinin spesifikasyonlarına veya son kullanıcılar/müşteriler tarafından sınırlı bir süre boyunca kullanıma dayalı nihai test. Son Asama olarak her şey gözden geçirilir.
* **Regression Testing** her sürümden önce yapılan değişikliklerin sürümün güvenirliğini sağlamak için yapılır. Değişiklikler sisteme yeni hatalar getirmediğinden emin olmak için yapılan bir testir. mevcut functionality’lerin iyi çalışmasını sağlamak için yeniden yürütülen test caselerin tam veya kısmi seçiminden ibarettir. Yeni değişiklikler ile birlikte programın bir bütün halinde çalışmasını sağlamak için yapılan testtir.
* **Functional Testing** bitmiş uygulama için yapılır ve programın . We can say that a functional test verifies specific action or function of the code, in other words, it tests functionalities of the application: Log-in, Log-out, Browse, Search, Check out.
* **Performance testing** bir iş yükü altında bir yazılım programının hızını, yanıt hızını ve kararlılığını belirleme işlemidir. Performans testi bir laboratuarda yapılan nicel testleri içerebilir veya sınırlı senaryolarda üretim ortamında gerçekleşebilir
* **Smoke testing**, or Sanity testing, En önemli functionality’lerin duzenli olarak çalışıp calismadigini gözden gecirmek icin yapilir. Programı bilmenin en önemli yoludur.

* **Positive and negative testing**, Positive testing doğru bilgilerle programın test edilmesidir. bu testte tester sadece doğru bilgilerle programın beklenildiği gibi hareket edip etmediğini öğrenir. Negative testing programın yanlış bilgilerle test edilmesidir. Negative test programın yanlış bilgilerle beklenildiği gibi sonuç verip vermediğini belirtir.
* **Ad-Hoc Testing** bir test planı veya dokumansız yapılan testtir. Bu yüzden herhangi bir döngüye veya programa ihtiyaç yoktur. Ad-hoc testing 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.

.

**Java**

* **What is an Exception?**

Hata gibi runtime/execution sırasında meydana gelir.

* **kaç tip exceptions biliyorsunuz?**
* kontrol edilmiş exceptions

Ör: Thread.sleep(ms); → Java bunu compile edemez eğer bu sorunu çözmez iseniz.

* kontrol edilmemiş/Runtime exceptions

Ex: ArithmeticException (int i=100/0;), NoSuchElementException

Bu exceptions runtime da meydana gelir ve developer isterse bu sorunu çözer. Java sorunu çözmen işin sizi zorlamaz.

* **Exception ile mücadele edebilmek için neler yaparsın?**

try...catch block.

* **Finally block nedir?**

finally block java exception handling icin kullanilir. Finally block u sadece try ile ve ya try-catch ile birlikte kullanabiliriz.finally block icindeki code her zaman calisir.

* **finally block'un calismadigi zamanlar hangileridir ?**
* System.exit(0); bu kod java programini durdurur ve finally block calismaz
* JVM ===> heap hafizasi dolu oldugu zaman çalışmaz (outOfMemoryError)
* stack hafizasi dolu oldugu zaman çalışmaz (StackOverFlow error)

**Selenium**

1. **What is Selenium?**

Selenium web testing için otomasyon tool idir. Selenium WebDriverdan oluşur. Selenium WebDriver web aplikasyonlarını otomasyon etmek için browserların native metodlarını kullanır.

1. **Kaç dil vardır selenium desteklediği?**

8: Java, C#, Groovy, Perl, PHP, Python, Ruby and Scala.

1. **Seleniumun avantajları nelerdir ?**

* Selenium açık kaynaktır ve lisans masrafları yoktur
* birçok dili destekler Ör: Java, ruby, python etc.
* Multi-browser testingi destekler.
* Internette birçok kaynak vardır.

1. **Seleniumun dezavantajları nelerdir?**

* Selenium sadece web-based applications destekler
* Herhangi bir raporlama için ilgili yeteneklerin üçüncü taraf araçlarına bağlıdır.
* HP UFT gibi ticari araçlara kıyasla araç için satıcı desteği yoktur
* Selenyum kullanarak masaüstü uygulamalarını test edemiyoruz.

1. **Selenium WebDriver kullanarak otomatikleştiremediğimiz bazı senaryoları listelemek ister misiniz?**

* Captcha otomasyonu Selenium WebDriver kullanarak imkansızdır
* Selenium WebDriver ile bar code okumayız.

1. **Selenium Webdriver'ı çalıştırmak için önkoşullar nelerdir?**

JDK, Eclipse, WebDriver (selenium standalone jar file), browser,

1. **HTML kodunu incelemek için ne kullanıyorsunuz?**

* The Chrome Developer Tools (Dev Tools for short) -> Sayfada herhangi bir noktaya sağ tık yaparak inspect e tıklarsanız
* Page Inspector - Firefox Developer Tools -> Sayfada herhangi bir noktaya sağ tık yaparak inspect e tıklarsanız
* Developer Tools (Internet Explorer) -> F12’e tuşlayın

# Real Interview Questions

1. Kendinden bahseder misiniz?
2. Test tecrübenden bahseder misiniz?
3. Hala çalışıyor musun? Neden iş değiştirmek istiyorsunuz?
4. Şuanki frameworkunden bahseder misiniz?
5. Nerede başlarsınız otomasyona? Nasıl başlarsın?
6. bir user storyi nasıl test ettiğini bize anlatır mısın?
7. Bana takımından bahseder misiniz?
8. projenizden bahseder misiniz?
9. Java da kendinizi puanlar mısınız?
10. Bir son tarih var ve yapacak çok işiniz var ve zamanında bitirmek imkansız görünüyor. Bununla nasıl mücadele edersin?