SDLC & AGILE

1) Yazılım Testi Nedir?

Bir yazılımın, kendisinden beklenen özellikleri karşılayıp karşılayamadığını incelemek amacıyla yapılan işlemlerdir. Bu şekilde yazılımdaki hatalar bulunup düzeltilebilir ve gereksinimlere uygun hale getirilebilir

2) Yazılım Gereksinimleri Belirtimi nedir? SRS Nedir?

SRS, amacı, desteklenecek ana iş süreçleri, özellikler, temel performans parametreleri ve davranış dahil olmak üzere geliştirilecek bir yazılım ürününün kapsamlı bir tanımını sağlamak olan bir belgedir. Bu nedenle, geliştirme sürecine rehberlik eden ve herkesi doğru yolda tutan bir harita görevi görür.

Bir SRS genellikle, yazılım geliştirme sürecinin en erken aşaması olan gereksinim mühendisliği aşamasının sonunda imzalanır. Hem işlevsel hem de işlevsel olmayan gereksinimleri içerir. İşlevsel gereksinimler, bir yazılım sisteminin ve bileşenlerinin işlevini (bir üniversite kütüphane sistemini tanımlarken kitapların önceden ayırtılması gibi) tanımlarken, işlevsel olmayan gereksinimler, yazılım sisteminin ve bileşenlerinin (güvenlik veya hizmet gibi) performans özelliklerini tanımlar.

3) Yazılım Geliştirme Yaşam Döngüsü SDLC Nedir?

SDLC mümkün olan en kısa sürede, en yüksek kalitede ve en düşük maliyetle yazılım üreten bir süreçtir. SDLC, bir organizasyonun iyi test edilmiş ve üretimde kullanıma hazır yüksek kaliteli yazılımı hızla üretmesine yardımcı olan iyi yapılandırılmış bir aşama akışı sağlar.

SDLC altı aşamadan oluşur:

1.Aşama: Planlama

“Ne istiyoruz?” SDLC’ nin bu aşamasında ekip, analiz edilen gereksinimlerin uygulanması için gereken maliyeti ve kaynakları belirler. Ayrıca ilgili riskleri detaylandırır ve bu riskleri azaltmak için alt planlar sunar. Başka bir deyişle, ekip projenin fizibilitesini ve projeyi en düşük riski göz önünde bulundurarak nasıl başarılı bir şekilde uygulayabileceklerini belirlemelidir

2. Aşama: Tasarım

Ekip, ürün için geniş bir dizi gereksinim ve hedef üzerinde anlaşmaya vardıktan sonra, bir sonraki adım bir tasarım spesifikasyonu oluşturmak olacaktır. Bu, “Bunu nasıl inşa edeceğiz?” Sorusuna cevap vermelidir. Bir ürün ekibi için bu aşama, önerilen çalışmanın öncelik sırasının belirlenmesini, ürünün yol haritasının oluşturulması aşamasıdır. Bu hem geliştirme hem de ürün ekiplerindeki herkesin amaçlarının daha net bir resmini elde etmesine yardımcı olacaktır.

3.Aşama: Uygulama veya Kodlama

Bu, mühendislik ekibinin ürünü fiilen kodladığı aşamadır. Bu aşamada geliştirme ekibi, yol haritasında iletilen üst düzey genel bakışı bir dizi taktiksel görev, bitiş tarihi ve günlük çalışma programlarına dönüştürür.

4. Aşama: Test

Ekip, yazılımın bir sürümünü tamamladıktan sonra, onu bir test ortamına yayınlayacak. Burada, QA ekibi ve geliştiriciler, herhangi bir kusur, hata veya diğer sorunları tespit etmek için uygulamanın tüm alanlarını test edecek.

5. Aşama: Dağıtım

Bu aşamada ekip, tüm kusurları düzelttiğinden ve yazılımın üzerinde anlaşmaya varılan hedeflere ve özelliklere göre oluşturulduğundan emindir.
Şimdi yazılımı üretim ortamına bırakma zamanı. Bu, ürünün genel olarak müşterilerin satın alması ve kullanması için hazır olacağı anlamına gelir.

6. Aşama: Bakım

Yazılımın artık canlı olması ve müşteriler tarafından kullanılmasıyla birlikte, geliştirme ekibinin odak noktası onu korumaya kayacaktır.

Geliştiricilerin iyileştirmeler, hata düzeltmeleri ve yeni özellikler için istekleri karşılamaya hazır olmaları gerekecektir. Bu talepler birçok kaynaktan (satışlar, yöneticiler, müşteriler) gelecek, ancak ürün yönetimi ekibi, geliştiricilerin üzerinde çalışması için bu girişimlerden hangisinin ürün yol haritasına gireceğini belirleyecektir.

4) Yazılım Testi Yaşam Döngüsü STLC Nedir?

Yazılım Testi Yaşam Döngüsü, kalite hedeflerinin karşılandığından emin olmak için belirli bir sırayla yürütülmesi gereken belirli adımları olan bir test sürecini ifade eder. STLC sürecinde her faaliyet planlı ve sistematik bir şekilde yürütülür. Her aşamanın farklı hedefleri ve çıktıları vardır. STLC'de farklı organizasyonların farklı aşamaları vardır; ancak temel aynı kalır.

STLC'nin 8 aşaması vardır:

1. Gereksinim Aşaması:

STLC'nin bu aşamasında, gereksinimleri analiz edin ve inceleyin. Diğer ekiplerle beyin fırtınası yapın ve gereksinimlerin test edilebilir olup olmadığını bulmaya çalışın. Bu aşama, testin kapsamını belirlemeye yardımcı olur. Herhangi bir özellik test edilebilir değilse, azaltma stratejisinin planlanabilmesi için bu aşamada bunu bildirin.

2. Planlama Aşaması:

Pratik senaryolarda, Test planlama, test sürecinin ilk adımıdır. Bu aşamada, test hedeflerine ulaşılmasına yardımcı olacak faaliyetleri ve kaynakları belirleriz. Planlama sırasında, ölçümleri, bu ölçümleri toplama ve izleme yöntemini de belirlemeye çalışırız.

Test Planlama Faaliyetleri

Çeşitli test türleri için test planı/strateji belgesinin hazırlanması

- Test aracı seçimi
- Test çabası tahmini
- Kaynak planlaması ve rol ve sorumlulukların belirlenmesi
- Eğitim gereksinimi

Çıktılar

- Test planı/strateji belgesi
- Efor tahmin belgesi

3. Analiz Aşaması:

Bu STLC aşaması, test edilecek 'NE' i tanımlar. Test koşullarını temel olarak gereksinimler belgesi, ürün riskleri ve diğer test temelleri aracılığıyla belirleriz. Test koşulu, gerekliliğe kadar izlenebilir olmalıdır.

Test koşullarının tanımlanmasını etkileyen çeşitli faktörler vardır:

- Seviyeler ve test derinliği
- Ürünün karmaşıklığı
- Ürün ve proje riskleri
- Yazılım geliştirme yaşam döngüsü dahil.
- Test yönetimi
- Ekibin becerileri ve bilgisi.
- Paydaşların mevcudiyeti.

Test koşullarını detaylı bir şekilde yazmaya çalışmalıyız. Ayrıntılı test koşulu yazmanın en önemli avantajı, test senaryoları test koşulu bazında yazılacağı için test kapsamını artırmasıdır, bu detaylar daha detaylı test senaryolarının yazılmasını tetikleyerek sonunda kapsamı artıracaktır.

Ayrıca, testin çıkış kriterlerini tanımlayın, yani testi ne zaman durduracağınız bazı koşulları belirleyin.

4. Tasarım aşaması:

Bu aşama test edilecek 'NASIL' ı tanımlar. Bu aşama aşağıdaki görevleri içerir:

- Test durumunu detaylandırın. Kapsamı artırmak için test koşullarını birden çok alt koşula ayırın.
- Test verilerini tanımlayın ve alın

- Test ortamını belirleyin ve kurun.
- Gereksinim izlenebilirlik metriklerini oluşturun
- Test kapsamı ölçümleri oluşturun.

5. Uygulama Aşaması:

Bu STLC aşamasındaki ana görev, ayrıntılı test senaryolarının oluşturulmasıdır. Test senaryolarına öncelik verin, hangi test senaryosunun regresyon paketinin bir parçası olacağını da belirleyin. Test senaryosunu tamamlamadan önce, test senaryolarının doğruluğundan emin olmak için incelemenin yapılması önemlidir. Ayrıca, gerçek yürütme başlamadan önce test senaryolarının onayını almayı unutmayın.

6. Yürütme Aşaması:

Adından da anlaşılacağı gibi bu, gerçek uygulamanın gerçekleştiği Yazılım Test Yaşam Döngüsü aşamasıdır. Ancak infazınıza başlamadan önce, giriş kriterinizin karşılandığından emin olun. Test durumlarını yürütün, herhangi bir tutarsızlık durumunda hataları günlüğe kaydedin. İlerlemenizi izlemek için eşzamanlı olarak izlenebilirlik metriklerinizi doldurun.

Test Yürütme Faaliyetleri

- Plana göre testleri yürütün
- Başarısız durumlar için hata kayıtlarını belgelendirin
- Düzeltilen hataları tekrar test edin
- Hataların kapanışı için takip edin

7. Sonuç Aşaması:

- Bu STLC aşaması, çıkış kriterleri ve raporlamaya odaklanır. Projenize ve paydaşlarınızın seçimine bağlı olarak, haftalık raporun günlük raporunu vb. Göndermek isteyip istemediğinize karar verebilirsiniz.
- Gönderebileceğiniz farklı rapor türleri vardır (DSR — Günlük durum raporu, WSR — Haftalık durum raporları), ancak önemli olan, raporun içeriğinin değişmesi ve raporlarınızı kime gönderdiğinize bağlı olmasıdır.
- Proje yöneticileri test geçmişine aitse, projenin teknik yönüyle daha çok ilgilenirler, bu nedenle teknik konuları raporunuza dahil edin (geçen test vakalarının sayısı, ortaya çıkan kusurlar, önem derecesi 1 kusurlar, vb.).
- Ancak, üst paydaşlara raporlama yapıyorsanız, teknik konularla ilgilenmeyebilirler, bu nedenle onları test yoluyla hafifletilen riskler hakkında bildirin.

8.Kapanış Aşaması:

Kapatma faaliyetleri için görevler şunları içerir:

- Testin tamamlanıp tamamlanmadığını kontrol edin. Tüm test senaryolarının kasıtlı olarak yürütüldüğü veya azaltıldığı. Açılan önem 1 kusuru olmadığını kontrol edin.
- Alınan dersler toplantısı yapın ve alınan dersler belgesi oluşturun. (Neyin iyi gittiğini, iyileştirmelerin kapsamını ve nelerin geliştirilebileceğini dahil edin)

Test Döngüsü Kapanış Faaliyetleri,

- Süre, Test kapsamı, Maliyet, Yazılım, Kritik İş Hedefleri, Kalite temelinde döngü tamamlama kriterlerini değerlendirin
- Test metriklerini yukarıda maddelere dayanarak hazırlayın
- Projeden öğrenilenleri belgeleyin
- Test kapanış raporu hazırlayın
- Ürünün müşteriye niteliksel ve niceliksel olarak raporlanması.
- Hata dağılımının türüne ve ciddiyetine göre test sonucu analizi
- Test Döngüsü Kapanışının Çıktıları
- Test Kapanış raporu,
- Test metrikleri

5) STLC ve SDLC arasındaki fark nedir?

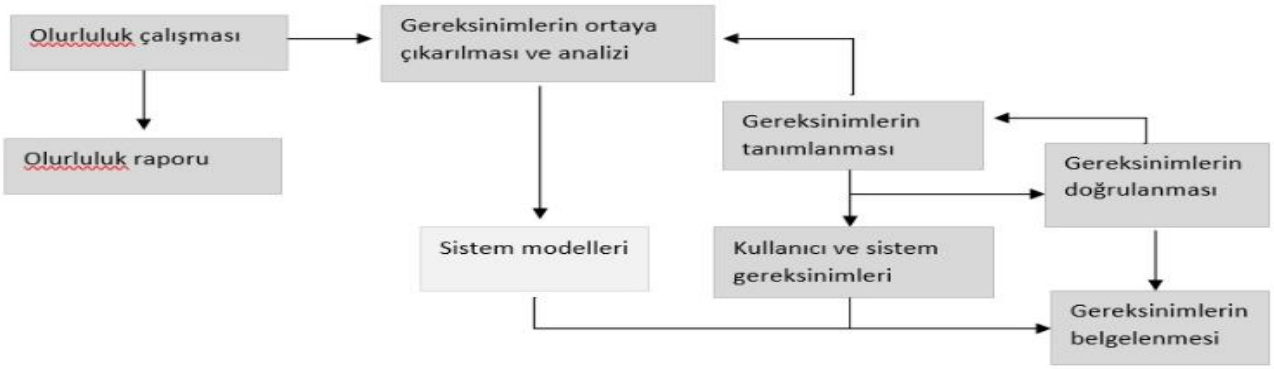
- SDLC, yazılım geliştirme sürecinde yer alan tüm standart aşamaları tanımlarken, STLC süreci, ürünün kalitesini iyileştirmek için çeşitli etkinlikleri tanımlar.
- SDLC bir Geliştirme Yaşam Döngüsü iken STLC bir Test Yaşam Döngüsüdür.
- SDLC'de geliştirme ekibi yüksek ve düşük seviyeli tasarım planlarını oluştururken, STLC'de test analisti Sistem, Entegrasyon Test Planını oluşturur.

- SDLC'de gerçek kod geliştirilir ve tasarım belgelerine göre gerçek çalışma gerçekleştirilirken, STLC test ekibi test ortamını hazırlar ve test durumlarını yürütür.
- SDLC yaşam döngüsü, bir ekibin yazılımın başarılı bir şekilde geliştirilmesini tamamlamasına yardımcı olurken, STLC aşamaları yalnızca yazılım testini kapsar.

Parametre	SDLC	STLC
Menşei	Geliştirme Yaşam Döngüsü	Yaşam Döngüsünün Test Edilmesi
Amaç	SDLC yaşam döngüsünün ana amacı, test ve diğer aşamalar dahil olmak üzere yazılımın başarılı bir şekilde geliştirilmesini tamamlamaktır.	STLC aşamasının tek amacı test etmektir.
Şartlı toplantı	SDLC'de iş analisti gereksinimleri toplar ve Geliştirme Planı oluşturur	STLC'de, QA ekibi işlevsel ve işlevsel olmayan belgeler gibi gereksinim belgelerini analiz eder ve Sistem Test Planı oluşturur.
Yüksek ve Düşük Seviyeli Tasarım	SDLC'de geliştirme ekibi, yüksek ve düşük seviyeli tasarım planlarını oluşturur	STLC'de, test analisti Entegrasyon Test Planını oluşturur
Kodlama	Gerçek kod geliştirilir ve tasarım belgelerine göre gerçek çalışma gerçekleşir.	Test ekibi test ortamını hazırlar ve yürütür
Bakım	SDLC aşaması, dağıtım sonrası destekleri ve güncellemeleri de içerir.	Test uzmanları, yerleştirilen bakım kodunu kontrol etmek için regresyon takımları, genellikle otomasyon betikleri yürütürler.

6) Gereksinim nedir? (Yazılım Gereksinimi)

Yazılım gereksinimleri geliştirilmesi istenen üründen nelerin beklendiğinin net açık bir şekilde ifade edilmesidir. Amacına uygun ürünlerin oluşması için beklenenlerin, istenilenlerin yani yazılım gereksinimlerinin net bir şekilde tespit edilmesi önemlidir. Genel de yazılım gereksinimleri belirtilmeden önce taleplerin tanımlanması yani iş birimleri (müşteri, kullanıcı vb.) tarafından gelen istekler ya da şirket içi birimlerden gelen talepler adımı bulunur. Taleplerin yönetilmesi için ayrı departmanların olduğu şirketlerde talep yöneticileri tarafından başlatılan bu süreçte bu isteklerin gereksinime dönüştürülmesi aşaması genelde İş Analistleri tarafından yapılır. İş analistleri gereksinim analizine ilk önce isteklerin, taleplerin yapılabilirliğinin olup olmadığına yani olurluluk çalışması ile başlarlar. Bu çalışma ile başlayan sürecin devamı şekildeki gibi işleyerek ilerlemektedir.

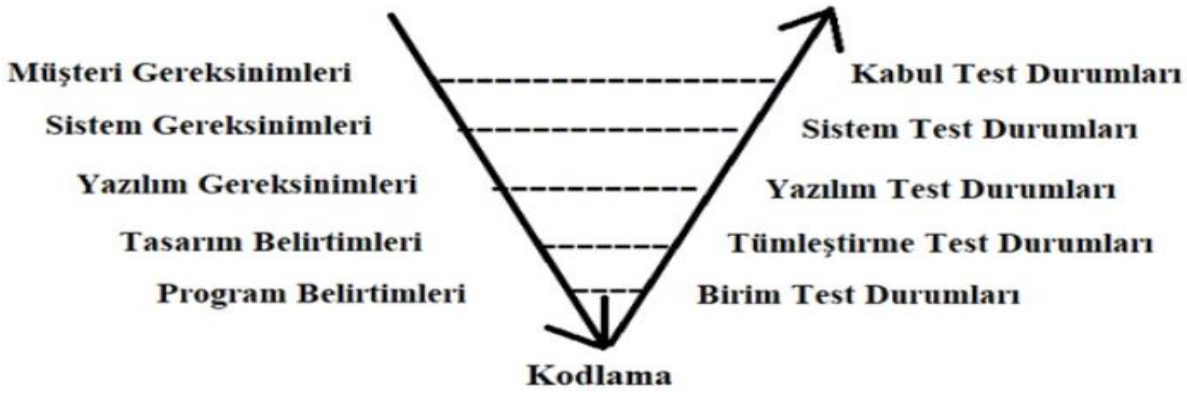


7) Yazılım gereksinimi-İhtiyaç nereden geliyor?

Doğru anlaşılmış bir ihtiyaç doğru yazılımın geliştirilmesi için atılan en önemli adımdır. Yazılım gereksinimlerinin başarılı şekilde çıkarılması geliştirilecek ürünün başarılı olması açısından, geliştiriciler de oluşacak belirsizliklerin önüne geçmesi açısından çok önemlidir. Zaman olarak büyük tasarruf sağlamasının yanında paydaşlar arasında iletişim sorunlarının da oluşmamasına yardım etmesi faydaları olarak verilebilir. Gereksinimlerin oluşturulmasından sonra yapılması gereken adım belirlenen gereksinimleri dokümantasyonu yapılmadan önce paydaşlar tarafından

gözden geçirme işlemi (review) ile doğrulanmasıdır. Gözden geçirmelerde gereksinimler doğruluk, açıklık, anlaşılırlık, yapılabilirlik, tutarlılık, izlenebilirlik, uyumluluk, test edilebilirlik gibi bir dizi kriter açısından gözden geçirilir.

Gereksinim Analizi yapılan farklı gereksinim türleri bulunabilir. Müşteri, Sistem, Kullanıcı vb. gereksinimleri kendi içlerinde belli kriterlere göre oluşturulan gereksinimlerdir. Gereksinimler test mühendislerinin referans alabileceği ve testlerin tasarlanmasında kullanılan bir girdi olarak düşünülebilir. Gereksinim doğru tanımlanmış, net açık ve anlaşılır olduğunda hem yazılım mühendisleri hem test mühendisleri kargaşanın içinde olmadan işlerini daha hızlı ve verimli şekilde yapmış olacaklardır.



Aslında gereksinimleri sağlamak, açıklığa kavuşturmak, belirtmek ve önceliklendirmek proje paydaşlarının sorumluluğundadır. Fakat bu, sizin oturup proje paydaşlarından tek tek isteklerini sıralamalarını bekleyeceğiniz anlamına gelmiyor. Düzenli toplantılar organize ederek, onları ne istediklerini daha ayrıntılı olarak belirtmeleri ve hatta belki de orijinal gereksinimlerini yeniden gözden geçirmeleri ve değiştirmeleri için motive edebilirsiniz.

8) Yazılım testi ne zaman başlar?

Yazılım testi, yazılımın yaşam döngüsünün sadece sonunda yer alan bir aşama değildir. Yazılım testi süreçtir ve yazılım tamamlandıktan sonra da devam eder. Örneğin ürün teslim edildikten sonra ürüne yeni özellikler eklenecek, ürün geliştirilecektir. Bu aşamada yine yazılım testi gerekli olacaktır.

• Test, gereksinimlerin test edilmesiyle başlar (en olası yanıt gibi görünen kodlama aşamasından sonra değil).

• İlk etapta gerekliliğin doğru olduğundan emin olmalıyız. Yanlış gereksinimle hata oluşturmak hatalara sebebiyet verir.

9) Gereksinimin iyi mi kötü mü olduğu nasıl anlaşılır?

Peki Gereksinimler Nasıl Olmalıdır? -SMART

Spesific-Açık olmalı

Measurable- Ölçülebilir olmalı

Attaninable- Uygulanabilir olmalı

Realisable- Amaca uygun olmalı

Testable- Test edilebilir olmalı

10) Neden test ediyoruz?

• Hatasız uygulama oluşturmak için.
• Memnun son kullanıcı ve müşteriye.
• Daha fazla gelir elde etmek için harika ürünler oluşturmak.
• Test etmeyi seviyorum ve test etmek benim tutkum.

11) Test edenin ana sorumluluğu nedir?

· Gereksinimlerin ne kadar karşılandığından emin olmak,
· Ürünün kalite düzeyine karşı güven oluşturmak,
· Ürünün test işleminin tamamlandığını ve müşterinin beklediği şekilde çalıştığını onaylamak,
· Hataları ve kusurları bulmak ve önlemek,
· Yetersiz yazılım ve kalite riskini azaltmak,
· Yasal standartlarına uymasını sağlamak bunlardan başlıcalarıdır.

12) %100 test mümkün mü?

Hiçbir uygulama % 100 hatasız değildir. Test yapmak, yazılımda bug olmadığını göstermez. Testin amacı, yazılımda hataların olduğunu göstermektir; yazılımda hata kalmadığını ispatlamak değildir. Bu madde, müşterilerin, proje yöneticilerinin ve yönetim ekibinin inandığı çok yaygın bir efsanedir. Yazılım testi, mümkün olduğu kadar test edebileceğimiz **işlevselliğin önceliğine** dayanan risk tabanlı bir faaliyettir. %100 test mümkün olmasa da %100 müşteri memnuniyeti sağlanmaya çalışılır.

13) Test hiyerarşisi nedir?

- Birim Testi (Unit Testing): Dinamik test sürecinin ilk aşaması olmakla beraber, hataların erken bulunup düzeltilebilmesi açısından da bu sürecin en önemli aşamasını oluşturur. Mikro ölçekte yapılan bu testte, özel fonksiyonlar veya kod modülleri (fonksiyonlar, veri yapıları, nesneler vb.) test edilir. Bu test, test uzmanlarınca değil programcılar tarafından yapılır ve program kodunun ayrıntıları ile içsel tasarım biçiminin bilinmesi gerekir. Uygulama kodu çok iyi tasarlanmış bir mimaride değilse oldukça zor bir testtir.
- Tümleyim Testi (Integration Testing): Bir uygulamanın farklı bileşenlerinin beraberce uyum içinde çalışıp çalışmadığını sınamak için yapılan bir testtir. Bileşenler, modüller, bağımsız uygulamalar, istemci/sunucu uygulamaları biçiminde olabilirler. Bu tür testlere, özellikle istemci/sunucu uygulamaları ve dağıtık sistemlerin testinde başvurulmaktadır. Bunun yanısıra uygulamaya yeni işlevsel elemanlar ya da program modülleri eklendikçe sürekli test edilmesi işlemine de “Artımsal Tümleyim Testi” adı verilir. Test uzmanları ve/veya programcılar tarafından gerçekleştirilen testlerdir.
- Regresyon Testi (Regression Testing): Uygulamada ve uygulama ortamlarında gerekli değişiklikler ve sabitlemeler yapıldıktan sonra yeniden yapılan testlere çekilme (regresyon) testi denilir. Böylece, önceki testlerde belirlenen sorunların giderildiğinden ve yeni hatalar oluşmadığından emin olunur. Uygulamanın kaç kez yeniden test edilmesi gerektiğini belirlemek güçtür ve bu nedenle, özellikle uygulama geliştirme döneminin sonlarına doğru yapılır.
- Zorlanım – Performans Testi (Performance Testing): Bu test, çoğu kez "yük testi" ile aynı anlamda kullanılmaktadır. Aynı zamanda, beklenmedik (normal olmayan) ağır yükler, belirli eylemler ve taleplerin çok fazla artışı, çok yoğun sayısal işlemler, çok karmaşık sorgulamalar vb. ağır koşullar altında olan bir sistemin işlevsellik testi (iş yapabilme testi) olarak da tanımlanabilmektedir. Bir web sitesi için sistem tepkisinin hangi noktada azaldığı veya yanıt veremez olduğunu belirlemek için yapılan testler, performans testine örnek teşkil edebilir.
- Kullanıcı Kabul Testi (User Acceptance Testing): Son kullanıcı veya müşteri siparişine (veya isteklerine) dayanan son test işlemidir. Kullanıcıların, uygulamayı “kabul” etmeden önce, söz konusu uygulamanın gereksinimlerini ne ölçüde karşılayıp karşılamadığını belirleyip, geri dönüş yapabileceği testlerdir.

- Beyaz Kutu Test Tekniği (White Box Testing Technic) : Beyaz kutu test tekniğinin en genel tabiri kod testidir. Projenin hem kaynak kodu hem de derlenmiş kodu test edilir. Bu tür testler, uygulama kodunun iç mantığı üzerindeki bilgiye bağlıdır. Yazılım kodundaki deyimler, akış denetimleri, koşullar vb. elemanlar sınanır.
- Kara Kutu Test Tekniği (Black Box Testing Technic) : Test ekipleri tarafından en çok kullanılan teknik olan kara kutu test tekniği adından da anlaşılacağı gibi uygulamanın sadece derlenmiş kodu üzerinden test edilmesi olarak bilinir. Bu test tekniğinde, yazılımın programatik yapısı, tasarımı veya kodlama tekniği hakkında herhangi bir bilgi olması gerekli değildir. Yazılımın gereksinimine duyulan şeylere yanıt verip veremediği ve işlevselliği sınanmaktadır.

14) Pozitif test nedir? Mutlu Yol testi

• Uygulamayı geçerli girişlerle test etme. "Mutlu Yol" testi olarak da adlandırılır.

15) Uyumluluk testi nedir?

Uyumluluk, müşteri memnuniyetini sağlamak için işlevsel olmayan bir testtir. Yazılım uygulamanızın veya ürününüzün farklı tarayıcılarda, veri tabanında, donanımda, işletim sisteminde, mobil cihazlarda ve ağlarda çalışacak kadar yetkin olup olmadığını belirlemektir.

Uyumluluk testi, tüm platformlar için aynı şekilde çalışan uygulamanın kontrol edilmesidir.

Genellikle, geliştirme ekibi ve test ekibi uygulamayı tek platformda test eder. Fakat uygulama üretimde yayınlandıktan sonra müşteri ürünümüzü farklı platformda test edebilir ve uygulamada kalite açısından layık olmayan hatalar bulabilir.

Bu tür sorunları azaltmak ve müşterilerinizi üzmemek için uygulamayı tüm platformlarda test etmeniz önemlidir.

16) Riske dayalı test nedir?

Risk tabanlı test (RBT) esasen risklere bağlı olarak projeler için yapılan bir testtir. Riske dayalı test stratejileri, testin yürütülmesi sırasında doğru testleri önceliklendirmek ve vurgulamak için risklerden yararlanır.

Her türlü işlevselliği kontrol etmek için yeterli zamanın olmayabileceği düşünüldüğünde, risk tabanlı test esas olarak en büyük etkiyi ve başarısızlık olasılığını taşıyan işlevselliği test etmeye odaklanır.

17) Bu regresyon takımını oluşturmak ne kadar sürdü?

• 3 yıl sürdü; 2 test cihazı 1 manuel test cihazı + 1 otomasyon test cihazı çalıştırdığımızda:
o serbest bırakılmadan önce
o büyük hata düzeltmesinden sonra
o büyük yeni işlevlerden sonra
• test senaryolarını tuttuğumuz ve ekip olarak bir sprintte birden fazla uygulanacak kararı aldığımız yer bazı senaryoları test edersiniz.

18) Regresyon paketini çalıştırırken karşılaşmış olduğunuz zorlukları bize anlatır mısınız?

Regresyon testi canlıda çalışan kodun üzerinde yapılan değişikliklerin kontrolü için kullanılır. Bu değişiklikler yeni bir fonksiyon, hata çözümü ya da performans geliştirmesi olabilir. Regresyon testleri genellikle değişiklikler son aşamaya geldiğinde ve yazılımın yeni sürümü yayınlamadan önce gerçekleştirilir. Regresyon testlerinin öncelikli amacı, uygulamanın kritik alanlarının hala beklendiği gibi çalıştığını kontrol etmektedir.
Regresyon testleri:

- Yazılımın değişiklik sonrasında son kalitesinin kontrol edilmesini
- Daha önce çıkan hataların düzeldiğinin kontrolünü
- Yazılım ekibinin ürün hakkında güveninin artmasını sağlar

Regresyon testleri maliyeti büyük testlerdir. Yazılım firmasında maliyet = zamandır ve sürümün hızlı şekilde canlı ortama aktarılması büyük önem taşır. Bu sebeple regresyon test adımları seçiminde bazı önemli noktaları göz önüne almalıyız. Bunlar:

- Kullanıcıların yoğun olarak kullanıldığı alanlar
- Genellikle hata çıkan uygulama alanları
- Ana fonksiyonlar
- Yüksek karmaşık fonksiyonlar
- Son değişikliklerin yapıldığı alanlar
- Önemli entegrasyonlar

Regresyon testlerinin doğru şekilde planlanması ve gerçekleştirilmesi, yazılım projelerinin başarıya ulaşması için önemlidir. Bu nedenle canlı geçişleri öncesi, regresyon testleri gerçekleştirerek yazılıma duyulan güveni arttırabiliriz.

19) Kaç environments ortamınız var?

• Geliştirme Ortamı (Development Environment)
o Birim testi
o Test ortamından daha az kararlı

• Test ortamı (Test Environment)
o Manuel test burada gerçekleşir
o Üretim ortamını tam olarak çoğaltır
o Değişiklikler aralıklarla dağıtılır
o Burada otomatik duman testleri yapılır
§ Uygulamanın diğer büyük testleri gerçekleştirmek için yeterince kararlı olduğundan emin olmak için test ortamına karşı çalışır faaliyetler.
§ Değişiklikler Test ortamına her dağıtıldığında çalıştır
§ Geliştirme ortamında çalıştırılabilir
o Otomasyon testleri burada çalıştırılır
o Otomatik Entegrasyon testleri burada çalıştırılır

• Üretim Öncesi Ortam (Pre-production Environment)
o UAT ortamı
o Demo burada gerçekleşir
o yük/performans testi burada gerçekleşir
o Değişiklikler büyük aralıklarla dağıtılır
o Burada otomatik majör regresyon testleri (yayınlanmadan önce)
§ UAT ortamına karşı çalışır
§ Yeni değişikliklerin herhangi bir kusurla sonuçlanıp sonuçlanmadığını öğrenmek için
§ Büyük hata düzeltmelerinden ve her sürümden sonra çalışır
§ Bu test, test planında kararlaştırılır
o Çok kararlı
• Üretim ortamı (Production environment)

20) Regresyon testinin hangi bölümü otomatikleştirilmelidir?

• Kararlı testler
• Sık tekrarlanan otomasyon adayları
• Basittir ve test cihazı girişi gerektirmez iyidir

Uygulamanın ne sıklıkla test edileceğini belirlemek zordur, bu yüzden uygulama geliştirme döneminin sonlarına doğru yapılır. Bu sayede, bütün bir projenin ne kadar değişim yaşadığı ve güncellemelerden ne derece etkilendiği daha net bir şekilde belirlenebilir. Uygulama altyapısı tamamen tanımlanmış duruma getirildiğinde regresyon testleri yapılmaya başlanır.

21) Regresyon testlerinizin etkili olduğundan nasıl emin oluyorsunuz?

• Regresyon testleri, kusurları yakalamaya izin verecek kadar geniş ve ayrıntılı olmalıdır. Yinelenen testi de ortadan kaldırabilirsiniz. Belirli durumlarda, test senaryolarını ve otomatik testleri mümkün olduğunca birleştirin.

22) Yazılımda bir dizi kritik hata düzeltildi. Raporlarla ilgili tüm hatalar tek bir modüldedir. Test yöneticisi sadece raporun modülünde regresyon testi yapmaya karar verir.

• Regresyon testi diğer modüllerde de yapılmalıdır çünkü bir modülün sabitlenmesi diğer modülleri etkileyebilir.

23) Regresyonunuzu nasıl yürütürsünüz? Ne sıklıkla, kaç VM, kaç gün, kaç test?

• Regresyon her sürümden önce planlanır ve yılda iki kez yayınlarız (İlkbahar sürümü ve sonbahar sürümü).
• Büyük bir hata düzeltmesi olduğunda da gerileme olur.
• Yaklaşık 500 özellik dosyası ve 1300 senaryo.
• Regresyon testleri jenkins tarafından başlatılır. Testler, jenkins sunucusunda (VM) yürütülür. Linux sunucum RedHat.
• En son çalıştırma 12 saatten fazla görünüyor. Bir diğer cevap ise;

o Bir dizi regresyon testi oluşturdum. Regresyon etiketli özellik dosyalarıdır. Ve jenkins'te tekmeleyen bir işim var regresyon testlerindendir. Testi tetiklemek için maven komutunu kullanır. Maven komutu şu etiket adını içerir:
mvn testi -D cucumber.options =”--tags @Regression”.
o Yürütmenin sonunda, jenkins ayrıntılı test adımları ve ekran görüntüleri içeren HTML raporu oluşturur.

24) Fonksiyonel test nedir?

Fonksiyonel Test, yazılım sistemini işlevsel gereksinimlere/spesifikasyonlara karşı doğrulayan bir tür yazılım testidir. Fonksiyonel testlerin amacı, uygun girdi sağlayarak ve çıktıyı Fonksiyonel gereksinimlere göre test edildiği bir tür kara kutu testidir.
Fonksiyonel test, esas olarak kara kutu testini içerir ve uygulamanın kaynak koduyla ilgilenmez. Bu test, Kullanıcı Arayüzü, API'ler, Veritabanı, Güvenlik, İstemci/Sunucu iletişimi ve test edilen uygulamanın diğer işlevlerini kontrol eder. Test, manuel olarak veya otomasyon kullanılarak yapılabilir.

25) Fonksiyonel olmayan test nedir?

Fonksiyonel olmayan test türleri 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.

26) Birim testi nedir? Hiç birim testi yaptınız mı?

Birim testi yazılımın fonksiyon, sınıf, arayüz gibi en küçük parçalarının test edilmesidir. Bu testin amacı yazılımın en küçük parçasının dahi beklendiği gibi çalışmasıdır. Birim testi geliştiriciler tarafından geliştirme süresi boyunca yapılır. Birim test tek bir kod bölümünü izole eder ve doğruluğunu kontrol eder. Birim testler manuel veya otomatik olarak gerçekleştirilebilir. Birim testlerinin yapılması ile bulunan hatalar projenin erken safhalarında keşfedildiği için hatanın daha kolay düzeltilmesini ve maliyetten tasarruf etmeyi sağlar.
• Geliştiriciler tarafından yapıldığı için henüz birim testi yapmadım. Ama bunu öğrenebileceğimi ve gerekirse yapabileceğimi düşünüyorum.

27) Bileşen testi nedir?

• Uygulamanın her bir bileşenini ayrı ayrı test etme. Uygulamada bir bileşen olabilir. Bir bileşen vardır bağımsız işlevsellik. Eski. amazon.com'da Satıcı işlevi bir bileşen olabilir. Alıcı başka olabilir bileşen. Ayrıca, Amazon prime videoları başka bir bileşen olabilir.

28) 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.

- ➢ Smoke test ne kadar sıklıkla yapılır?
- ✓ Genelde her gün yapılan testlerdir diyebiliriz. Dikkat edilmesi gerekir dönemsel olarak dikkat edilmelidir.

• TEST sırası:
Kod-> Birim Testi-> Entegrasyon Testi-> Sağlık Testi-> Duman Testi ve Fonksiyonel Test
• Projemizde; oturum açma, kullanıcıyı görüntüleme, kullanıcı ayrıntı sayfası, yeni kullanıcı oluşturma ve görev oluşturma
• Bu beş modülde, geliştirici ilk olarak tüm ana işlevleri yürüterek duman testini gerçekleştirecektir. Gibi modüller; kullanıcı, geçerli oturum açma bilgileriyle oturum açabilir veya giremez, oturum açtıktan sonra yeni kullanıcı oluşturulabilir veya oluşturulamaz, yaratıldı, görüntülendi veya görüntülenmedi vb.

29) Kara kutu testi nedir? Farklı kara kutu test teknikleri nelerdir?

• Kara kutu testi, yazılımın iç yapısını bilmeden test etmek için kullanılan yazılım test yöntemidir.

7 Farklı Kara Kutu Test Tekniği

1. Equivalence Partitioning Design

Denklik paylarına ayırma test tekniği de denebilir. En basit tanımıyla aynı özelliği gösteren test koşullarının gruplandırılması ve belirlenen grupların her birinden seçilen kısıtlı veri seti ile uygulamaların test edilmesidir.

2. Boundary Value Technique Design

Sınır değer tekniğidir. Yazılımda hataları bulmak için yazılımdaki tek noktaya odaklanmak yerine değişkenin alt ve üst sınırlarını kontrol ederek yapılan testlerdir. Yapılan yazılımların sınır değerlerinde hata çıkma olasılığının ufak farkla fazla olması sebebiyle kullanılır.

3. Decision Table Test Design

Karar tablosu test tekniğidir. Gereksinimleri belirlenmiş çok koşullu karmaşık yazılımlar için kullanılır. İş kuralları bir tablo üzerine yazılıp girdiler için alınabilecek değerler ve hangi girdi için hangi çıktının alınabileceği bu tabloda gösterilir. Görülebilecek bütün test senaryoları bu tabloya yazılır.

4. Classification Tree Test Design

Sınıflandırma ağacı test tasarımı tekniğidir. Test için kullanılan nesnenin hiyerarşik olarak denklik paylarının bir ağaç görünümünde modelleyerek bu payların birbirleriyle olan ilişkileri üzerinden test senaryoları üreten bir tekniktir.

5. Pairwise Test Design Technique
İkili test tasarım tekniğidir. Testi yapılan yazılımın fonksiyonunun birden fazla kombinasyona sahip olduğu durumlarda test senaryo sayısını önemli ölçüde azaltmaya yarayan test tekniğidir. Bir girdi girilip, bir çıktı alınan senaryolardan ziyade iki girdi girilip hata bulunabilirliğini arttırmaktadır. Bu teknik zamandan önemli ölçüde kazanç sağlamaktadır.
6. State Transition Test Design Technique
Durum geçişi test tasarımı tekniğidir. Yazılımın geçmişteki ve şu anki durumları değerlendirilerek yapılır. Durumlar arası geçişler test edilir. Başarılı sonlanan geçişler ve başarısız geçişler bir diyagram haline getirilir. Genellikle gömülü veya elektro-mekanik sistemler üzerinde kullanılan bir tekniktir.
7. Use Case Test Design Technique
Bu teknikte fonksiyonel olan ve fonksiyonel olmayan gereksinimler kullanılarak test senaryosu üretilir. Bu teknikteki en önemli özellik senaryoların açık ve detaylı olmasıdır. Temel olarak test yöntemi bir kullanıcı ve bu kullanıcının sistemden beklediği durumların oluşup oluşmadığını ölçümler.

30) Denklik bölümleme testi nedir? (Equivalence partitioning testing)

• Denklik bölümleme testi, uygulama giriş testi verilerini her birine bölen bir yazılım test tekniğidir. Test senaryolarının türetilebileceği eşdeğer verilerin en az bir kez bölümlenmesi. Bu test yöntemi ile süreyi kısaltır. Yazılım testi için gereklidir.

• Örnek: Bir not hesaplama sistemini test ederken, test eden kişi 90'dan 100'e kadar olan tüm puanların bir not vereceğini belirler.

A, ancak 90'ın altındaki puanlar olmaz.

• Girdi ve çıktı kapsamını elde etmek için hangi teknik kullanılabilir? İnsan girişine uygulanabilir, arayüzler aracılığıyla giriş yapılabilir, entegrasyon testinde bir sisteme veya arayüz parametrelerine.

31) Sınır değer testi nedir? (Boundary value testing)

• Giriş ve çıkış denklik sınıflarının kenarlarında, altında ve üstünde sınır koşullarını test edin.

• Örneğin, maksimum 1000$ ve minimum 100$ çekebileceğiniz bir banka uygulaması diyelim. Değer testi, ortada vurmak yerine sadece kesin sınırları test ediyoruz. Bu, maksimumun üzerinde test ettiğimiz anlamına gelir, limit ve minimum limitin altında.

• Örneğin, kredi kartımın: Aktivasyon tarihi alt sınırdır. Son kullanma tarihi 10/2019 üst sınırdır. 0 dolar daha düşük harcama limiti için sınır. 25.000$, harcama limiti için üst sınırdır.

32) Sınır değer analizi neden iyi test durumları sağlar?

• Çünkü değerler aralığının 'kenarlarına' yakın farklı durumların programlanması sırasında hatalar sıklıkla yapılır.

33) Karar tablolarını neden kullanırız? (decision tables)

• Eşdeğerlik paylaştırma ve sınır değer analizi teknikleri genellikle belirli durumlara veya girdilere uygulanır.

Ancak, farklı girdi kombinasyonları farklı eylemlerin gerçekleştirilmesine neden oluyorsa, bunu kullanarak göstermek daha zor olabilir. Daha çok kullanıcı arayüzüne odaklanma eğiliminde olan eşdeğerlik paylaştırma ve sınır değer analizi.

• Spesifikasyona dayalı diğer iki teknik, karar tabloları ve durum geçiş testi daha çok iş odaklıdır. Mantık veya iş kuralları. Bir karar tablosu, şeylerin kombinasyonlarıyla (örneğin girdiler) başa çıkmanın iyi bir yoludur.

• Bu teknik bazen 'neden-sonuç' tablosu olarak da anılır. Bunun nedeni, ilişkili bir mantık olmasıdır. Bazen karar tablosunun türetilmesine yardımcı olmak için kullanılan 'neden-sonuç grafiği' adı verilen diyagram oluşturma tekniği.

34) Beyaz kutu testi nedir ve beyaz kutu testi türlerini listeler mi? (white box testing)

Yazılım test tasarım tekniklerinden Beyaz Kutu Testi, şeffaf kutu testi de denebilir. Tıpkı ismi gibi beyaz bir kutu içerisinden bakılarak yazılımın kodunun iç yapısının bilinerek, ölçümlenerek test senaryolarının tasarlandığı tekniktir. Bu yöntemdeki ana amaç kod parçacıklarının tek tek test edilerek aslında en küçük parçacık halinde bile sağlıklı bir şekilde çalıştırılabildiğinin görülmesidir. Gereksinimler sonucu belirlenen girdilerin kodun bir parçasındaki çıktıları karşılayıp karşılamadığı test edilir. Yazılımın işlevselliği test edilmez.

Yazılım test uzmanları tarafından da kullanıldığı gibi çoğunlukla yazılım geliştiriciler tarafından kullanılır. Bu yöntem birim, entegrasyon ve sistem testi seviyelerinde kullanılır.

2 Farklı Beyaz Kutu Test Tasarım Tekniği

1. Komut Test Tasarım Tekniği

Komut kodda direkt olarak gerçekleşmesini istediğimiz bir işleve denir. Int y=5 bir komuttur. Komut test tasarımı ise yazılım içerisinde yer alan bütün komutların üzerinden en az bir kez geçerek hepsinin test edilmesiyle sağlanır.

2. Karar Test Tasarım Tekniği

Karar kapsamı kodumuzun içerisindeki if else, while, switch gibi birden fazla sonuç oluşturabilecek karar noktalarıdır. Bu test tasarım tekniğinde karar noktalarından her bir koşul üzerinden bir kez geçildiği durumlar değerlendirilir. Karar noktalarının üzerinden mutlaka geçilmesi gerekmektedir.

35) Beyaz kutu testinde neyi doğrularsınız?

• Koddaki güvenlik açıklarını doğrulayın

• Koddaki eksik veya bozuk yolları doğrulayın

• Belge spesifikasyonuna göre yapının akışını doğrulayın

• Beklenen çıktıları doğrulayın

• Uygulamanın tam işlevselliğini kontrol etmek için koddaki tüm koşullu döngüleri doğrulayın

• Satır kodlaması ile satırı doğrulayın ve %100 testi kapsar

36) Gri Kutu Testi Nedir? (Gray Box Testing)

Bu yaklaşım türünde ağ içerisinde bulunan veya sistemdeki herhangi bir hizmeti kullanan çalışanların veya yüklenicilerin erişim yetkilerinin değerlendirildiği güvenlik testi yaklaşımıdır.

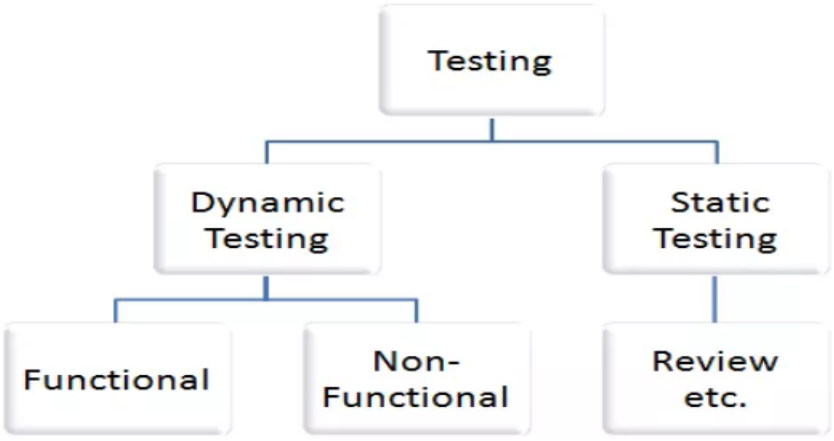
• Gri kutu testi, kara kutu ve beyaz kutu testinin melezidir.

• Gri kutu testinde test mühendisi, bileşenin kodlama bölümü bilgisine sahiptir ve test senaryoları veya test tasarlar. Sistem bilgisine dayalı veriler.

• Bu test cihazında kod bilgisi vardır, ancak bu beyaz kutu testi bilgisinden daha azdır. Bu bilgiye dayanarak test senaryoları tasarlanır ve yazılım uygulaması, bir kara kutu ve test cihazı olarak test muamelesi altında uygulamayı test eder.

37) Statik ve dinamik test arasındaki fark nedir?

- Statik test programı çalıştırmadan yapılırken Dinamik test programı çalıştırarak yapılır.
- Statik test, hataları bulmak için kodu, gereksinim belgelerini ve tasarım belgelerini kontrol ederken Dinamik test, yazılım sisteminin işlevsel davranışını, bellek / CPU kullanımını ve sistemin genel performansını kontrol eder.
- Statik test, kusurların önlenmesi ile ilgilidir, Dinamik test ise kusurları bulmak ve düzeltmekle ilgilidir.
- Dinamik test doğrulama sürecini gerçekleştirirken statik test doğrulama sürecini gerçekleştirir.
- Statik test derlemeden önce gerçekleştirilirken Dinamik test derlemeden sonra gerçekleştirilir.
- Statik test teknikleri yapısal ve ifadeyi kapsarken, Dinamik test teknikleri Sınır Değer Analizi ve Eşdeğer Bölümlemedir.
- • Statik test: Statik test sırasında kod yürütülmez ve yazılım belgeleri kullanılarak gerçekleştirilir.
- • Dinamik test: Bu testi gerçekleştirmek için kodun yürütülebilir bir biçimde olması gerekir.



38) Entegrasyon Testi Nedir?

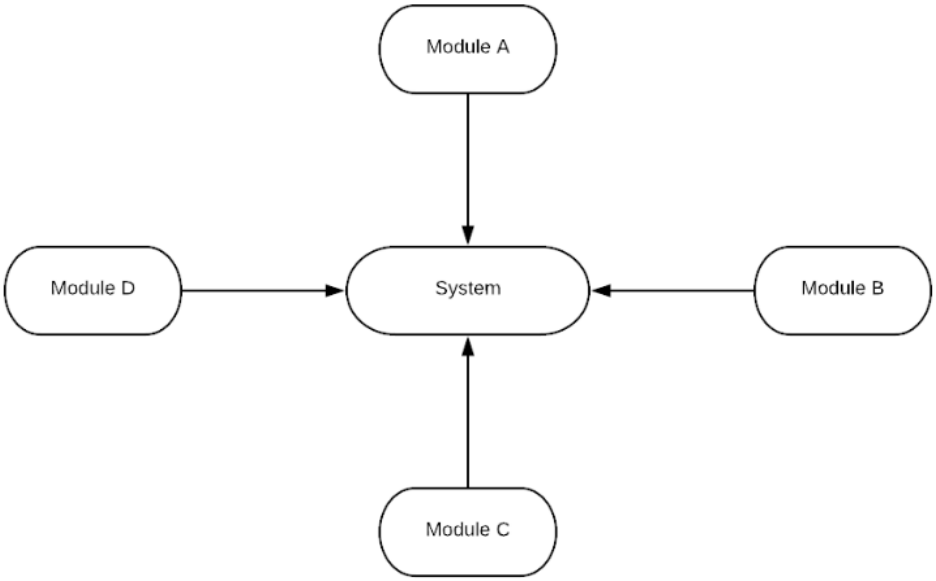
Entegrasyon birleştirme demektir. Entegrasyon testi bir yazılımın bileşenlerinin birbirine entegre edilmesi sırasında yapılabileceği gibi iki farklı yazılımın birbirine entegre edilmesi sırasında da yapılabilir. Bu yüzden entegrasyon testi, birim entegrasyon testi ve sistem entegrasyon testi olarak farklı test seviyelerinde yapılabilir.

Yazılım geliştirme uzmanlarının birim test sırasında ayrı ayrı test ettikleri bileşenler birbirine entegre edildikleri zaman hataya sebep olabilirler. Entegrasyon testi, sistemin bu farklı bileşenlerinin(birimlerinin) birlikte doğru çalışıp çalışmadıklarını test etmeyi amaçlar.

Entegrasyon testleri dört'e ayrılır:

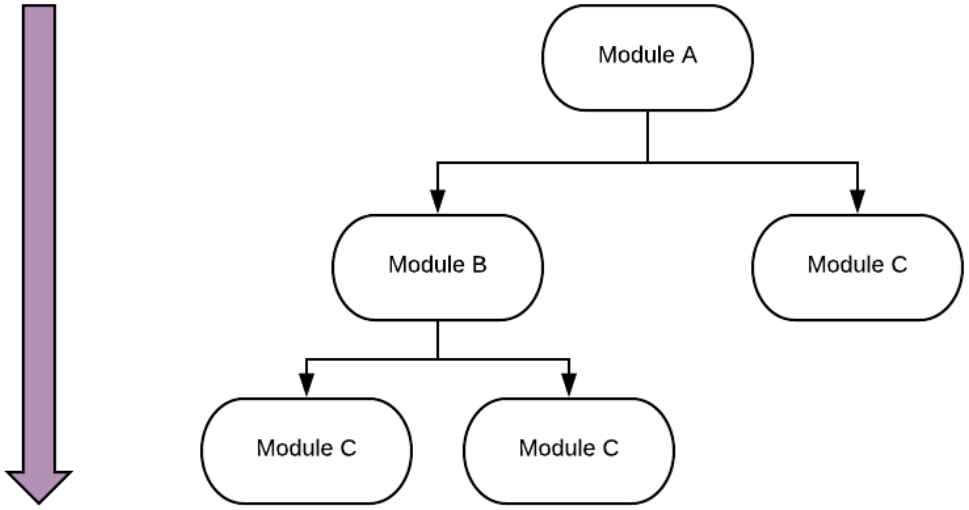
1) Big Bang Integration Test:

En yaygın kullanılan entegrasyon test tipidir. Geliştirilmiş tüm modüller bir araya getirilerek yapılan testtir. Hızlı ve kolay bir şekilde birbirleri ile beraber çalıştıklarında anlam ifade eden modüllerin doğruluğunu sağlar fakat birim başı metot doğruluğunun gözden kaçınılması olasıdır.



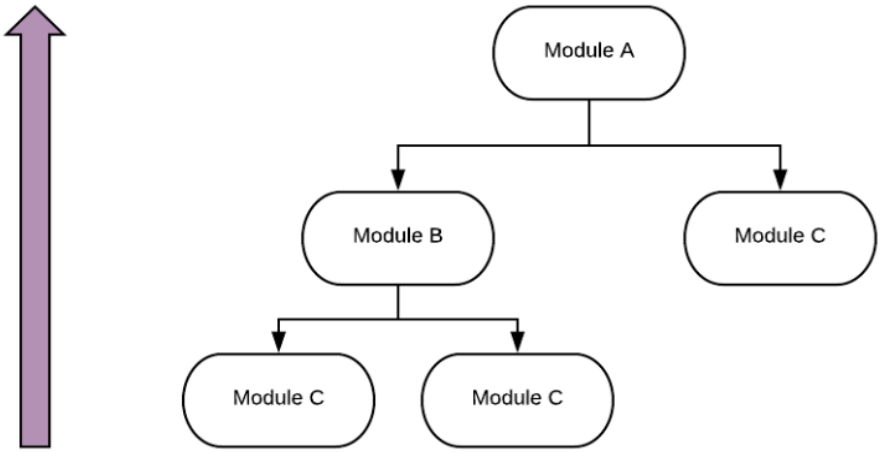
2) Top-Down Integration Test:

Bu entegrasyon testindeki amaç ise modüller arası geçiş yapılırken hatalı olan modülün kolay bir şekilde bulunabilmesini sağlamaktır. Test işlemi yukarıdan aşağı doğru gerçekleşmektedir ve her birinin test işleminden başarılı bir şekilde geçerek ilerlemesi gerekmektedir. Her bir modül testleri stub olarak adlandırılmaktadır. Modül ağacının son bacaklarında ise her bir stub kendi içerisinde test edilerek test işlemi sonuçlandırılır.



3) Bottom-Up Integration Test:

Bu test yöntemi ise Unit Testler ile beraber ilerlemektedir. Alt tarafta bulunan tüm stublar, Unit Testlerden geçirilerek yukarıya doğru ilerlenir. Top-Down'da olduğu gibi yukarıya ilerlerken Unit Testler aracılığı ile her test başarılı olarak sonuçlanmalıdır. Tüm stublar için Unit Testler oluşturulduktan sonra bir üst seviyede hepsi bir ele alınarak test işlemi yapılır. Bu test tipindeki amaç ise stublardan başlayarak hataların en kısa sürede bulunabilmesidir.



4) Sandwich/Hybrid Integration Test:

Modüllerin bir kısmı Top-Down, bir diğer kısmı ise Bottom-Up tiplerini kullanılarak gerçekleştirilen test tipidir. Bu karma tipteki amaç ise bazı modülleri gruplara ayırabilirken diğer modülleri ise ayrı bir şekilde test edebilmektir.

39) Scalability (Ölçeklenebilirlik) Nedir?

Ölçeklenebilirlik Testi, bir uygulamanın performansının kullanıcı isteklerinin veya diğer bu tür performans ölçüm özniteliklerinin sayısını artırma veya azaltma becerisi açısından ölçüldüğü, işlevsel olmayan bir test metodolojisidir.

Ölçeklenebilirlik testi, donanım, yazılım veya veri tabanı düzeyinde gerçekleştirilebilir. Bu test için kullanılan parametreler bir uygulamadan diğerine farklılık gösterir, bir web sayfası için kullanıcı sayısı, CPU kullanımı, ağ kullanımı olabilirken, bir web sunucusu için işlenen isteklerin sayısı olabilir.

40) Depolama Testi Nedir? (Storage Testing)

Depolama testi, yazılım geliştirme yaşam döngüsünün bir parçasıdır, özellikle uygulama veya programın veri dosyalarını doğru dizinlerde depolayıp okumadığını ve beklenmedik bir şekilde sonlandırılması için yeterli miktarda depolama alanı ayırıp ayırmadığını görmek için test edildiği test aşamasıdır. Yer olmaması nedeniyle oluşmaz.

Depolama testi ayrıca, üreticilerin kendileri veya üçüncü taraf sertifika kuruluşları tarafından yapılan bir kıyaslama biçimi olarak depolama cihazlarının ve sistemlerinin kapsamlı test edilmesini ifade eder.

• Depolama Testinde, verileri DB'ye depolamaktan sorumlu olan uygulamanın bu işlevlerini test ederiz.

• Son kullanıcı tarafından GUI'de veya ön uçta girilen veriler, veri tabanında depolanan verilerle aynıdır.

• Depolama testi, uygulamanın ön ucundan alınan verilerin doğru yerde ve doğru yerde saklandığını belirler.

41) Stres Testi Nedir?

Stres testi, olağandışı yoğunluğunda üzerinde beklenmedik yoğunluklar sırasında, bilişim sisteminizin tüm bileşenlerinin davranışı ve yanıt sürelerinin belirlenmesini sağlayan bir test olarak gerçekleştirilir. Böylece tüm bilişim sisteminizin, kriz aşamasındaki tepki süreleri ve kapasiteleri belirlenir. Bunun yanında bileşenlerin yanıt vermekte güçlük çekmesi aşamasının yanında kırılma aşamasındaki tepki ve davranışların net bir şekilde belirlenmesi, stres testi aracılığıyla gerçekleştirilir. Bu sayede tüm sisteminizin kapasitesel güncellemelerinin, doğru bir şekilde planlanması ve kapasitesinin de net çerçevelerde belirlenmesi sağlanmış olur.

Stres testi aracılığıyla herhangi bir sektörde kullanılmakta olan ya da kullanılmaya yeni başlanacak tüm sistemlerin kriz aşamalarındaki davranış ve kapasiteleri belirlenebilir. Kullanım oranı ve sisteme erişen kullanıcı sayısı oranına göre, gereken

kapasitesel artırım planlamaları, stres testi sayesinde daha odaklı bir şekilde gerçekleştirilebilir. Bununla birlikte olağandışı şekilde gerçekleşen sistem yüklenmeleri sırasında, sistemin hangi aşamada kendisini korumaya almasının belirlenmesi de stres testi sayesinde belirlenebilir. Böylece olası kötü niyetli saldırıların sistemde geçici ve kalıcı hasar oluşturmasının önüne geçilmiş olur. Dolayısıyla kullanıcılara en verimli hizmet sunumunun hangi kapasite özelliklerine sahip bir sistemler sunulabileceğinin belirlenmesi sağlanır.

42) Bakım Testi (maintenance testing) Nedir?

İşletimde olan bir sisteme yapılan değişiklerin veya değişmiş bir ortamın işletimde olan bir sisteme etkisinin test edilmesidir.

• Mevcut yazılımın değiştirilmesi, taşınması veya kullanımdan kaldırılmasıyla tetiklenir.

43) Test Koşum Nedir? (Test Harness)

Yazılım Testinde Test Harness, test yürütmesini otomatikleştirmek için gerekli olan saplamalar, sürücüler ve diğer destekleyici araçlardan oluşan bir koleksiyondur. Test donanımı, bir test kitaplığı kullanarak testleri yürütür ve test raporları oluşturur. Test koşum takımı, test senaryoları, hedef dağıtım bağlantı noktası (TDP), test edilen kaynak dosya, taslaklar vb. gibi bir testi derlemek ve çalıştırmak için gereken tüm bilgileri içerir.

Test Harness'in kullanıldığı iki bağlam vardır

- Otomasyon testi: Test komut dosyalarını, bu komut dosyalarını çalıştırmak için gerekli parametreleri ve analiz etmek için sonuçları toplar.
- Entegrasyon testi: Birleştirilmiş davranışın beklendiği gibi olup olmadığını kontrol etmek için birbiriyle etkileşime giren iki kod birimi veya modülü bir araya getirmek için kullanılır.

• Test donanımı, uygulamayı farklı testlerde çalıştırarak test etmek için gereken bir yazılım ve test verileri koleksiyonudur.

Stres, yük, veriye dayalı ve davranışını ve çıktılarını izleme gibi koşullar. Test Harness iki ana parça içerir:

o Test yürütme motoru

o Test komut dosyası deposu

• Otomasyon testi, testlerin yürütülmesini kontrol etmek ve gerçek sonuçları beklenen sonuçlarla karşılaştırmak için bir aracın kullanılmasıdır.

Sonuçlar. Ayrıca test ön koşullarının oluşturulmasını da içerir.

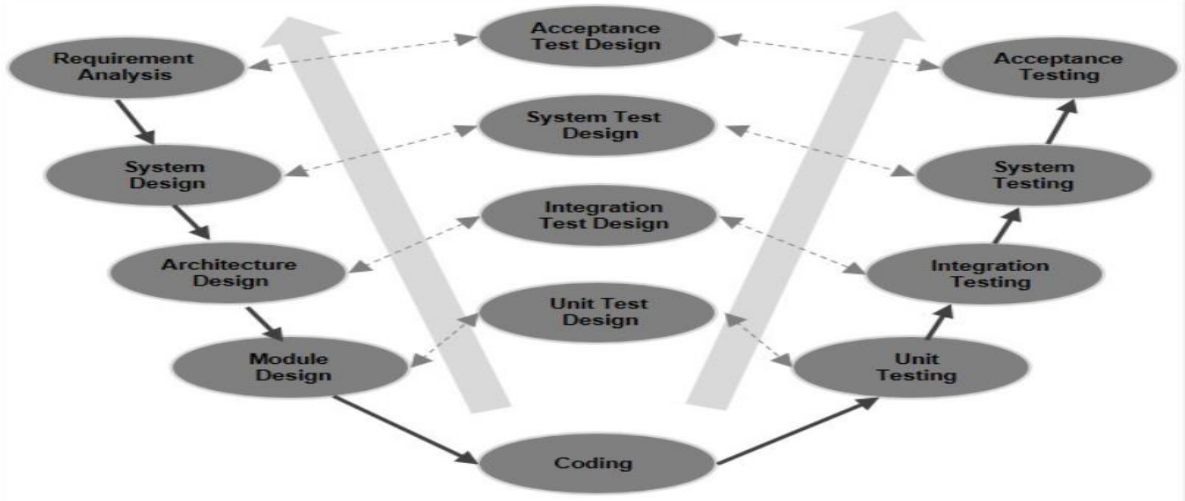
44) Test kapsamı nedir?

• Test kapsamı, kaç tane test senaryosuna sahip olduğumuz ve bu test senaryolarının hangi fonksiyonel alanı kapsadığı anlamına gelir.

45) V- Model Nedir?

V şekli üzerinde sıralı bir şekilde süreçlerin gerçekleştirildiği modeldir. Verification and Validation model olarak da bilinir.

Waterfall modelinin uzantısıdır ve her geliştirme aşamasına karşılık gelen bir test aşamasının ilişkilendirilmesine dayanır, bir sonraki aşama ancak önceki aşama tamamlandıktan sonra başlayabilir.



46) Test yakalama ve tekrar oynatma olanakları sağlayan test araçlarının kullanımından en çok aşağıdakilerden hangisinin yararlanması muhtemeldir?

• Regression testing (Gerileme testi): Regresyon testi canlıda çalışan kodun üzerinde yapılan değişikliklerin kontrolü için kullanılır. Bu değişiklikler yeni bir fonksiyon, hata çözümü ya da performans geliştirmesi olabilir. Regresyon testleri genellikle değişiklikler son aşamaya geldiğinde ve yazılımın yeni sürümü yayınlamadan önce gerçekleştirilir. Regresyon testlerinin öncelikli amacı, uygulamanın kritik alanlarının hala beklendiği gibi çalıştığını kontrol etmektedir.

• Integration testing = Entegrasyon birleştirme demektir. Entegrasyon testi bir yazılımın bileşenlerinin birbirine entegre edilmesi sırasında yapılabileceği gibi iki farklı yazılımın birbirine entegre edilmesi sırasında da yapılabilir. Bu yüzden entegrasyon testi, birim entegrasyon testi ve sistem entegrasyon testi olarak farklı test seviyelerinde yapılabilir.

• System testing = Sistem testi, eksiksiz ve entegre bir sistemde gerçekleştirilir. Gereksinimlere göre sistemin uygunluğunun kontrol edilmesini sağlar. Bileşenlerin genel etkileşimini test eder. Yük, performans, güvenilirlik ve güvenlik testlerini içerir.

• User Acceptance Testing = Kullanıcı kabul testi (UAT), yazılım test sürecinin son aşamasıdır. UAT sırasında, gerçek yazılım kullanıcıları, spesifikasyonlara göre, gerçek dünya senaryolarında gerekli görevleri yerine getirebildiğinden emin olmak için yazılımı test eder.

47) Kabul testi nedir? (Acceptance testing)

Agile süreçlerde sprint içerisinde yer alan story'lerin geliştirmesi ve test ekiplerince testleri tamamlandıktan sonra genel olarak product owner'lar tarafından gerçekleştirilen test aktivitesidir. Waterfall ve V-Model süreçlerde ise genel olarak bu testleri analist ya da iş talebinde bulunan iş birimleri yapmaktadır. Kullanıcı Kabul Testlerinin amacı talep edilen işin içerdiği kullanıcı ihtiyaçlarının karşılanıp, karşılanmadığının belirlenmesi ve canlıya çıkışı için onay kararının verilmesidir.

Kullanıcı kabul testi için örnek bir giriş ve çıkış kriterleri aşağıdaki tabloda özetlenmiştir.

Giriş Kriterleri	Çıkış Kriterleri
• Kod Yazıldı • Kod Gözden Geçirildi • Birim Testleri Yapıldı • Birim Testlerinde Tespit Edilen Hatalar Düzeltildi • Testler Koşturuldu • Tespit Edilen Yüksek ve Orta Oncelik Seviyesindeki Tüm Hatalar Çözümlendi • UAT'de Kullanılacak Test Verileri Hazırlandı	• Tüm UAT Test Senaryoları Koşuldu • Tüm Kabul Kriterleri Test • Hedef Kalite Seviyesi Yakalandı (>=%90). (Passed Test Cases / Passed + Failed Test Cases) • Tespit Edilen Defect'ler Çözümlendi (Priority >= Major)

48) UAT (Kullanıcı Kabul Testi) ve Sistem testi arasındaki fark nedir?

SIT	UAT
Bu, test uzmanları ve geliştiriciler tarafından gerçekleştirilir.	Bu, son kullanıcılar ve müşteriler tarafından gerçekleştirilir.
Alt birimlerin / birimlerin entegrasyonu burada kontrol edilir. Arayüzler test edilecektir.	Tüm tasarım burada kontrol edilir.
Bireysel birimler, sistemin gereksinimlere göre çalışacağı şekilde entegre edilir ve test edilir.	Sistem, kullanıcının istediği şekilde ürünün ana işlevselliği açısından bir bütün olarak test edilir.
Test uzmanlarının gereksinimlerine göre yapılır.	Ürünün son kullanıcı tarafından nasıl kullanılması gerektiğine ilişkin kullanıcı bakış açısına göre yapılır.
Sistem kurulur kurulmaz SIT gerçekleştirilir.	UAT, nihayet ürünün piyasaya sürülmesinden hemen önce gerçekleştirilir.

- Sistem testlerinde testerlar sadece elindeki bilgiye göre hareket eder
- UAT ise tüm aşamalarıyla elinde olan her şeyle testini gerçekleştirir ve daha geniş olarak testi yapar müşteri işi istediği gibi test etme olarak hareket eder. Her ihtiyacı olanı test eder.

49) Sürekli entegrasyon nedir?

Sürekli entegrasyon, geliştiricilerin yazdıkları yeni kodu kod tabanına günde en az bir kez ekleyerek geliştirme döngüsü boyunca daha sık entegre ettiği bir yazılım geliştirme sürecidir. Entegrasyon sorunlarını, düzeltmenin daha kolay olduğu daha erken dönemde belirlemek için derlemenin her yinelemesinde otomatikleştirilmiş test gerçekleştirilir. Bu, yayın için gerçekleştirilen son birleştirmede de sorunların önlenmesine yardımcı olur. Genel olarak sürekli entegrasyon, derleme sürecinin kolaylaştırılmasına yardımcı olur; bu da yazılım kalitesinin daha yüksek, teslimat zamanlamalarının daha öngörülebilir olmasını sağlar.

50) Kod, üretim ortamına nasıl dağıtılır?

• Yerelden

o Git'e çek ve gönder kullanarak kodu kontrol edin (şirketimde bu SVN'dir)

o Birim testlerini çalıştırın

o Değişiklikleri sunucuya dağıtın

§ Geliştirici kodu her kontrol ettiğinde jenkins tarafından otomatik olarak yapılır.

§ Dev ortamındaki değişiklikleri dağıttıktan sonra

• Geliştirme ortamından

o Jenkins tarafından yapılan Test ortamındaki değişiklikleri dağıtın

o Programlanabilir veya manuel olarak tetiklenebilir

• Test Dağıtımından-> Jenkins tarafından yapılan değişiklikler

• Ön üretimden (From Pre-production)

51) Çevik Çerçeve? (Agile Framework)

• Rol: PO, SM, Takım

• Törenler: -Sprint Planlama, Günlük Scrum, Sprint İnceleme, Sprint Retro, Bakım Seansı

• Artifacts: Ürün biriktirme listesi, - Sprint biriktirme listesi, - Tükenmişlik tablosu

52) Çevik nedir? (Agile)

Yazılım geliştirme modellerinden olan Agile, "Çevik ve Atik" anlamına gelen yazılım geliştirmede kullanılan proje yönetimine özel bir yaklaşımdır. Bu yöntem ekiplerin yazılım geliştirme süreçlerinin öngörülemezliğine cevap vermesine yardımcı olur. Agile yazılım geliştirme, bazı değer ve prensipleri baz alarak bir dizi metot ve uygulamayı altında toplayan bir şemsiyedir. Agile, belirsiz bir ortamda dahi başarılı olabilmek için değişime karşılık verebilme ve geliştirme yapabilme esnekliği, kapasitesi ve yeteneğidir.

• Çevik, şelale metodolojisine alternatif olan yinelemeli ürün geliştirme metodolojisidir.

• Scrum: Ekip, bir sonraki sprint için iş miktarını planlar

• Kanban: Sprint planlaması yok, hikayeler olduğu gibi alınır, ancak yine de her şeye sahipsiniz

53) Neden Çevik'e ihtiyacımız var? Şelale ve Çevik?

• Şelale metodolojileri aşağıdaki dezavantajlara sahip olduğundan;

- ✓ Gereksinim değiştirilemez veya belge imzalandıktan sonra değiştirilmesi zor değildir.
- ✓ Şelalede bir aşamayı tamamlamadan bir sonraki aşamaya geçemezsiniz. Örneğin, kodlama aşaması tamamlanmadan test başlatılamaz.
- ✓ Müşteri, geliştirme yaşam döngüsünün son aşamasına kadar ne elde edeceğini göremez.
- ✓ Üretime gitmek daha uzun sürer. Ürün pazara girdiğinde, zaten modası geçmiş olabilir.

• Çevik aşağıdaki avantajlara sahiptir:

- ✓ Değişiklik memnuniyetle karşılanır. Örneğin, sprint demosundan sonra müşteri bir şeyi beğenmezse, geri bildirimlerini alabilir ve ürünü iyileştirebiliriz. Gereksinim değişikliği tamam.
- ✓ Yinelemeli geliştirme süreci olduğundan, geliştirme ekibi bir işlevsellik geliştirebilir, geri bildirim alabilir ve bir sonraki yinelemeyi iyileştirebilir. Böylece ürün sürekli olarak geliştirilecektir.
- ✓ Scrum master yardımıyla atıklar çevik olarak elimine edilir. Örneğin bloke olursam bekleyip zamanımı boşa harcamam gerekmez. Ekip üyeleri birbirleriyle verimli bir şekilde iletişim kurduğundan, mükerrer çabayı önleyerek daha üretken olabiliriz.
- ✓ Şelale, C#_.NET gibi araçları ve platformu vurgular, ancak çevik insanları vurgular. En iyi araca sahip olabilirsiniz ama sonunda insanlar bu araçları kullanıyor. İlham veren insanların daha az paraya veya daha az kaynağa sahip olsalar bile harika ürünler yapabileceğine inandım.

54) Önceki projelerinizde ne tür Agile metodolojisi kullandınız?

• Extreme programlama (XP), Kanban ve Scrum'ı duydum. Ama sadece scrum ile çalıştım.

55) Scrum, Çevik bir çerçevedir, değil mi? Diğer birkaç Çevik çerçeveyi adlandırın.

• Evet, Scrum bir Çevik çerçevedir.

Diğer birkaç Çevik çerçeve:

- ➢ Feature Driven Development (Özellik Odaklı Geliştirme)
- ➢ Test Driven Development (Test Odaklı Geliştirme)
- ➢ Kanban

56) Scrum'daki farklı roller nelerdir?

Scrum takımı bir Scrum Master, bir Ürün sahibi ve geliştiricilerden oluşur. Scrum takımında ne hiyerarşi ne de altgruplar vardır. Scrum takımı, her seferinde tek bir hedefe yani ürün hedefine (Product Goal) odaklanan profesyonellerden oluşan kapsayıcı ve uyumlu bir birimdir.

Pig Roller; Scrum sürecine dahil olanlar, projede asıl işi yapan kişilerdir. Bunlar Scrum Master, Product Owner , Geliştirme Takımıdır.

Chicken Roller; Scrum'ın işleyişinde aktif olarak yer almayan kişilerdir.Müşteriler ve Satıcılar gibi

- ❖ Geliştiriciler (Developers): Scrum takımındaki geliştiriciler, kendini her Sprintte kullanışlı bir Arttırım yaratmaya adamış kişilerdir. Geliştiricilerin ihtiyaç duyduğu belirli beceriler genellikle geniştir ve çalışma alanına göre değişiklik gösterir. Bununla birlikte, Geliştiriciler her zaman aşağıdakilerden sorumludur:
- ❖ Ürün Sahibi (Product Owner): Ürün Sahibi, Scrum Takımının çalışmasının sonucu olan ürünün, değerini maksimize etmekten sorumludur. Bunun nasıl yapılacağı kuruluşlar, Scrum Takımları ve bireyler arasında büyük ölçüde değişebilir.
- ❖ Scrum Master: Scrum Master, Scrum Kılavuzunda tanımlandığı gibi Scrum'ı kurmaktan sorumludur. Bunu, hem Scrum Takımı hem de organizasyon içinde herkesin Scrum teorisini ve uygulamasını anlamasına yardımcı olarak yapar. Scrum Master, Scrum Takımının etkinliklerinden sorumludur. Bunu, Scrum Takımının Scrum çerçevesi içinde uygulamalarını iyileştirmesini sağlayarak yaparlar.

• Product Owner aslında projenin paydaşıdır.

- ✓ Proje gereksinimlerini ekip önünde temsil eder.
- ✓ Ne inşa edileceğine dair bir vizyona sahip olmaktan ve detaylı vizyonunu ekibe iletmekten sorumludur.
- ✓ Agile scrum yazılım geliştirme projesinin başlangıç noktasıdır.

• Scrum takımı, belirli bir projenin gerçekleştirilmesi için performans sergileyen bireylerin toplu katkılarıyla oluşur.

- ✓ Ekip, talep edilen ürünün zamanında teslimi için çalışmakla yükümlüdür.

• Scrum master – Scrum master, scrum ekibinin taahhüt edilen görevleri düzgün bir şekilde yerine getirip getirmediğini kontrol eden scrum ekibinin lideri ve koçudur.

- ✓ Sprint hedefine etkili bir şekilde ulaşabilmeleri için ekibin verimliliğini ve üretkenliğini artırmaktan da sorumludur.

57) Scrum takımını nasıl tanımlarsınız?

Bir Scrum takımı, istenen ürünü veya ürün parçacığını (increment) sunmak için birlikte çalışan bir gruptur. Genellikle projeyi yürütmek için gereken farklı becerilere sahip bir ekipten oluşur. Scrum takımları kendi kendine organize olmayı becerebilen ve cross-functional ekip üyelerinden oluşmalıdır.

- Kendi kendine organize olmak: Bir lidere bağlı kalmadan çalışmanın nasıl yapılacağına karar verebilmektir.
- Cross-functional (İşlevsel): Ekip dışı üyelere bağlı kalmadan, farklı yeteneklere sahip olan ve ortak bir hedef için çalışan bir grup insandır. Bir elin beş parmağı gibi.

• 5 rock yıldızını bir araya getirmeniz onların bir takım oldukları veya harika bir ürün oluşturabilecekleri anlamına gelmez.

• Benim için ekip, aynı hedefi paylaşan, aynı yönde ilerleyen, birbirine güvenen ve harika bir ürün oluşturmak için birbirleriyle etkili bir şekilde iletişim kuracak ve işbirliği yapacak olan bir grup insandır. Yıldız bir birey değil, bir yıldız takımı olmalı.

58) Scrum Master'ın sorumlulukları nelerdir?

• İzleme ve izleme

• Gereksinimleri doğru anlamak

• Proje hedefine ulaşmak için çalışın

• Proses kontrol master ve kalite master

• Takımı müfrezelerden koruyun

• Takımın performansını iyileştirmek

• Toplantıları yönetin ve sorunları çözün

• Çatışmaların ve engellerin çözümü

• İletişim ve raporlama

59) Negatif test durumu nedir?

Negatif testler yazılımın çalışmadığı noktaları göstermeyi amaçlar. Test esnasında olumlu sonuç verecek veriler yerine olumsuz sonuç verecek veriler kullanılır. Bu test ile yazılımın olumsuz durumlarda nasıl çalıştığını, olumsuz sonuçlarda ne gösterdiğini, birimin dayanıklılığını test edebiliriz.

• Negatif test senaryoları, yıkıcı bir şekilde test etme fikrine dayalı olarak oluşturulur. Örneğin, uygulamaya uygun olmayan girdiler girilirse ne olacağını test etmek.

60) "Scrum of Scrums" teriminden ne anlıyorsunuz?

Bir Scrum of Scrums, kaynak teslim ekiplerine gömülü bağlantılara sahip delegelerden oluşan sanal bir ekiptir. Tipik organizasyonel hiyerarşiler veya proje tabanlı ekiplerle karşılaştırıldığında, bu birbirine bağlı ekip yapıları iletişim yollarını azaltır. Amaç, daha küçük, bağımsız ekipleri koordine etmektir. Scrum of Scrums uygulayan ekipler yalnızca teslimatı koordine etmekle kalmaz, aynı zamanda her sprint sonunda tam entegre bir ürün sağlar. Bu nedenle, Scrum of Scrums, müşterilere değer sağlayan bir yayın ekibi olarak hareket eder. Kuruluşlar genellikle bu yaklaşımı çevikliği ölçeklendirmek ve daha büyük ve karmaşık ürünlerin teslimatını organize etmek için ilk adım olarak kullanır.

• Şu anda yedi ekibin üzerinde çalıştığı aktif bir projeyi varsayalım. Her takım kendi scrum toplantısını yönetmekten sorumludur. Ancak farklı ekiplerle koordine ve iletişim kurmak için ayrı bir scrum toplantısı düzenlemek gerekir. Her takımdan, büyükelçi olarak bilinen ve takımını scrum'larda temsil etmekten sorumlu olan bir takım lideri vardır.

• Scrum ekipleri arasında koordinasyonu sağlamak için düzenlenen scrum toplantısı, scrum of scrum olarak bilinir.

61) Sevk edilebilir ürün/artış? (product/increment)

• Ürünün parçası yapılır ve her sprintten ek işlevsellik kazanmaya devam eder.

• Artış, geliştirme ekibinin Bitti Tanımı ile uyumlu olmalıdır

- ✓ Ürün eki teslim edildiğinde "Bitti Tanımı"nı karşılaması gerekir.

- ✓ Kabul kriterleri karşılandı
- ✓ Ürün sahibi, kullanıcı hikayelerini kabul eder.

• Artış, P.O tarafından kabul edilebilir olmalıdır

62) BurnDown Grafiği nedir?

Burndown chart, belirli bir zaman aralığında yapılması planlanmış iş – zaman ilişkisini takip etmek amacıyla kullanılan bir grafiksel sunum çeşididir. Dikey eksende puan (story point) değerleri, yatay eksende ise zaman (gün) değerleri bulunur.

• İşin tamamlanma hızının ve ne kadar işin kaldığını gösteren grafik gösterimidir.

63) Verification(Doğrulama) ve Validation(Geçerlilik) nedir?

Verification(Doğrulama) Nedir?

- Yazılım süreçlerinde verification işlemini yapan rolü yazılımcılar üstlenir.
- Çıktının, ürünün şartlarında belirtilen özelliklere uygun olup olmadığını doğrulamak için yapılan bir işlemdir. Yani yazılımcının, elde ettiği kod çıktısıyla dokümanda belirtilen standart ve gerekliliklere uygun olup olmadığının kontrolünü yaptığı işlemler bütünüdür.
- Yazılım süreçlerinde genellikle ilk olarak verification işlemi yapılır.
- Verification işlemi yapılarak daha sonra yapılacak olan validation işleminde gözden kaçan bir durum tespit edilebilir.

Validation(Geçerlilik) Nedir?

- Yazılım süreçlerinde validation işlemini yapan rolü testçiler üstlenir.
- Çıktının veya ürünün, müşterinin beklentilerini karşılayıp karşılamadığını denetlemek içindir. Yani testçinin, yazılımcıdan gelen kodu müşterinin istediği asıl ürün ile karşılaştırıp doğru ürün üretilip üretilmediğinin kontrolünü yaptığı işlemler bütünüdür.
- Yazılım süreçlerinde genellikle validation işlemi, verification işleminden sonra ikinci işlem olarak yapılır.
- Validation işlemi yapılarak daha önce yapılan verification işleminde gözden kaçan bir durum tespit edilebilir.

• Verification, test edenler ve geliştiriciler tarafından geliştirme sırasında gerçekleşir; geliştirme aşamasında yazılımı değerlendirme ve belirli bir uygulamanın ürününün belirtilen gereksinimleri karşılayıp karşılamadığına karar verme sürecidir.

• Test uzmanları tarafından Validation; geliştirme süreci sonunda yazılımın değerlendirilmesi ve müşteri gereksinimlerini karşılayıp karşılamadığının kontrol edilmesi sürecidir.

64) Ready'nin tanımı nedir?

• Kabul Kriterleri temizlendi/incelendi

• Puan/saat verilir

65) Kullanıcı Hikayesi Nedir?

• (Not: temel olarak, bir kullanıcı hikayesi sadece bir gerekliliktir) Kullanıcı hikayesi, kısa ve basit bir tanım olup, minimum sevk edilebilir üründür.

• Normalde şöyle görünür:

<son kullanıcı> olarak <aksiyon> yapmak istiyorum, böylece <fayda> olabilirim.

o Amazon kullanıcısı olarak internetten alışveriş yapabilmem için giriş yapabilmeliyim.

66) “Sevk edilebilir(shippable)” dediniz, bununla ne demek istiyorsunuz?

• Kullanıcı adı alanına kullanıcı adımı koymak istiyorum diyemezsiniz.

• Böylece kullanıcı adımı oraya yazabilirim. Tam işlevsellik olmalıdır. Kullanıcı adını koymak, gönderilebilir bir işlev değildir.

Ancak giriş yapabilmek tam bir işlevselliktir. Gönderilebilir derken bunu kastediyorum.

67) Otopark nedir? (parking lot)

• Agile'da şu anlama gelir: Toplantıda, diğer insanlarla gerçekten ilgili olmayan bir sorununuz olduğunda, toplantıda o konuyu tartışmaya devam etmemeliyiz çünkü diğer insanların zamanını boşa harcıyoruz.

<Otopark maddesi yapalım> demek toplantıdan sonra o konuyla ilgilenen herkes konuşabilir.

68) Sprint iş akışı nedir?

Sprint İş Listesi, Ürün Parçasında hangi fonksiyonların olacağına ve bu fonksiyonları “Bitti” tanımına uygun bir Ürün Parçasına dönüştürmek için gerekli olan işe dair bir öngörüdür. Sprint İş Listesi, Geliştirme Takımının Sprint Hedefine ulaşmak için gerekli gördüğü tüm işleri görünür kılar.

• Bir hikâyenin yapılacaklar arasında nasıl ilerlediği ve yaşam döngüleri- bir şey engellendiğinde ne olur, vb.

69) En son projenizdeki çevik deneyim? (Agile experience)

• Sprintimiz 4 haftadır ve her 3 sprint'te bir sürüm döngüsü olarak yayın yaptık

• Ekibimde 7 kişi var. 3 geliştirici (Shwan, Simon, Sinan), 1 otomasyon (Me) ve 1 fonksiyonel test cihazı (Usman), ayrıca 1 SM (Yasin) ve 1 PO (Simon B.).

• Sprint Planlama Toplantısı ile bir sprint başlatıyoruz ve

o ekibin öncelikli özellikleri ve ürün biriktirme listesi öğeleri hakkında tartışırız ve

o Uygulamanın geliştireceğimiz kısmını öğreniyoruz.

o Hız ve kapasiteye göre hikaye seçimi

§ Hız: Bir sprintte teslim edilen hikâye noktalarının/demo'nun sayısı. Örneğin: ekip 30 hikâye puanı planlamışsa (İş değeri); bir sprintte kullanıcı hikayelerinin değeri ve planlandığı gibi teslim edilebiliyorsa takımın hızı 30'dur

§ Kapasite: Bir sprint için mevcut toplam saat sayısı Takımın kapasitesidir. Tatil ve PTO saatlerini hesaplar

o Bu toplantı her hafta yapılır ve yaklaşık 1 saat sürer. Görevler için bazı tahminler vermek için Sprint Grooming toplantısını yaptığımızdan daha genel bir fikir ediniriz.

§ Ekip, SM ve PO bir araya gelerek iş öğelerinin alakalı ve kullanışlı olmasını sağlar

§ Kullanıcı hikayeleriyle ilgili P.O'ya sorular sorun

§ Kabul kriterlerini yeniden tanımlayın

§ Yeni hikayeler yazmak

§ Destanları kullanıcı hikayelerine bölmek

§ Doğru tahminde bulunmak/eksik/fazla tahminden kaçınmak için hikâyeyi anlayın

Nasıl tahmin edersiniz?

Tecrübelerime ve hikâyenin karmaşıklığına dayanarak ve daha önce üzerinde çalıştığım bir şey.

• Sprint başladıktan sonra Günlük Standup Toplantısı yapıyoruz

o her gün sabah ve dün ne yaptığımızı, bugün ne yapacağımızı ve herhangi bir engelleyici olup olmadığını tartışıyoruz.

o Sadece sprint hakkındaki bilgileri senkronize ediyoruz.

• Sprint sonu, genellikle Sprint Demo/İnceleme Toplantısı yaparız.

o Sadece müşteriye sprint oluşturduğumuzu göstermek içindir (PO geri bildirimde bulunabilir)

o Ekibimde bir SDET olarak, bazen sunum yaptım ve konferans odasındaki işlevleri gözden geçirdim.

o Müşteri veya paydaşlar ya da iş adamları bilmediklerini soruyorlar.

• Sprint Demo'nun ardından Sprint Retrospektif Toplantısı yapıyoruz.

o Sprint Retro'da son sprintte neyin iyi olduğunu, ne gibi hatalar yaptığımızı konuşuyoruz.

o Bunları gözden geçirip bir daha aynı hataları yapmamaya özen gösteriyoruz.

o İyi bir şey ve iyileştirmeler yapsaydık, yapmaya devam ederdik.

o Sprint gözden geçirme toplantısında veya sprint sonunda yapılan bu toplantı; 2-3 saat sürer.

70) Epik (Epic) nedir?

• Epic, tek bir sprintte tamamlayamayacağınız büyük bir kullanıcı hikayesidir.

• Örneğin, bir kullanıcı olarak yerel mağazayı ziyaret etmek zorunda kalmamak için çevrimiçi satın almak istiyorum. Bu hikaye çok büyük ve bir sprintte tamamlanamaz. Yani kullanıcı hikayesi yerine Epic diyebiliriz. Aşağıdakiler gibi birden çok kullanıcı hikayesine bölünmelidir:

o Bir müşteri olarak hesabımı görebilmek için giriş yapabilmek istiyorum.

o Bir müşteri olarak, satın alabilmek için bir ürün arayabilmek istiyorum.

o Bir müşteri olarak, satın alacağım ürünün ödemesini yapabilmek için ödemeye devam edebilmek istiyorum.

o Bir müşteri olarak hesabımı koruyabilmek için çıkış yapabilmek istiyorum.

o Gördüğünüz gibi< Bir müşteri olarak satın alabilmek istiyorum...> birden fazla kullanıcı hikayesine bölünebilir. Ekip, her sprintte bir veya daha fazla kullanıcı hikayesi seçebilir.

71) Kabul kriterleri nedir?

Kabul kriterleri (Acceptance Criteria — AC), bir yazılım ürününün bir kullanıcı, müşteri veya başka bir sistem tarafından kabul edilebilmesi için karşılaması gereken koşullardır. Her kullanıcı hikayesi için benzersizdirler ve son kullanıcının bakış açısından özellik davranışını tanımlarlar. İyi yazılmış kabul kriterleri, bir geliştirme aşamasının sonunda beklenmedik sonuçlardan kaçınmaya yardımcı olur ve tüm paydaşların ve kullanıcıların elde ettiklerinden memnun olmalarını sağlar.

• Kabul kriterleri, kullanıcı hikayesinin başarılı bir şekilde geliştirilip geliştirilmediğini bilmemizin yoludur.

• Bir hikâyenin ne zaman "bittiğini" ve beklendiği gibi çalıştığını belirlemek için kullanıcının bakış açısından tanımlanan gereksinimlerin ifadeleri

• 3 parça örneği

- ✓ Giriş -> geçerli e-posta adresi
- ✓ Süreç -> işaretleme mesajlaşma
- ✓ Sonuç -> pazarlama mesajı tasarımı, pazarlama tarafından sağlanan özelliklerle eşleşir.

72) Sıçandeliği nedir? (rat hole)

• Çevik ekipte çok fazla iletişim olduğundan, ekip birçok şeyi tartışmak zorundadır. Ancak bazen tartışma bir konu için çok uzun sürer ve bu gerçekten verimli olmaz. Bunun <rat deliği> olduğunu söyleyeceğiz, bu, bu konuyu fazla uzatmamamız ve ilerlememiz gerektiği anlamına geliyor.

73) Ne tür Test vakaları? (Test cases)

• Farklı senaryoları ele alıyorum

- ✓ Pozitif (Positive)
- ✓ Negatif (Negative)
- ✓ Sınır Değer Analizi (Boundary Value Analysis)

74) Test Case Nedir?

Test case'ler gereksinimlere göre hazırlanan input'lar, olaylar ya da aksiyonlar ve bunlar sonucu oluşması beklenen sonuçların belirtildiği dokümanlardır.

• Test durumu, Test Edilen Uygulamaya karşı kontrol etmek için özel bir koşuldur. Test adımları, ön koşullar, test ortamı ve çıktılar hakkında bilgi içerir.

• Test durumu, işlevselliği ve test adımlarını açıklar.

o Test Durumu Kimliği o Test Case ID

o Adım numarası o Step number

o İşlevin açıklaması o Description of the functionality

o Beklenen sonuç o Expected result

o Gerçek Sonuç o Actual Result

75) Genellikle bir haftada kaç Test cases (regresyon takımınızda) tamamlarsınız?

• 10 küçük test vakası, 7-8 orta, 2-3 büyük

• VEYA Projeye bağlıdır. Şirketimde 2000 test vakamız var. 4 Stay'de yaklaşık 700 test vakamız var

76) Regresyon takımınızın çalıştırmanız ne kadar sürer?

• Projeye bağlıdır. Mevcut projemde, regresyon paketindeki 2000 test senaryosundan 1500 civarında zaten otomatikleştirilmiştir. Paralel yürütme gerçekleştirmek için 10 sanal makine kullanırsak, otomatikleştirilmiş test senaryolarının yürütülmesi 2 ila 3 gün sürer. Ayrıca, manuel test cihazları bazı manuel test senaryolarını yürütür, ancak ne kadarını yürüttüklerinden emin değilim. Bazı önemli test senaryolarını yalnızca önceliklendirmeden sonra yürüttüklerine inanıyorum.

77) Otomatik komut dosyanızı çalıştırdığınızda ne yaparsınız veya regresyon çalıştırdığınızda ne yaparsınız?

• İlk önce betiğimi çalıştırmam gerekiyor. Komut dosyası yürütme işlemi tamamlandıktan sonra, herhangi bir başarısız test durumu olup olmadığını görmek için çalıştırma sonucunu analiz etmem gerekiyor. Başarısız test durumları varsa, bunun meşru uygulama sorunu nedeniyle mi yoksa bazı komut dosyası sorunlarından mı kaynaklandığını belirlemem gerekiyor. (script, otomasyon kodu sorunu nedeniyle de başarısız olabilir) eğer uygulama sorunundan kaynaklanıyorsa, manuel olarak yeniden oluşturmaya ve çoğaltabilirsem bir kusur kaydetmeye çalışacağım. Senaryomdan kaynaklanıyorsa, düzeltmem gerekiyor. Ancak çoğu zaman durum böyle değildir.

78) Bir testi otomatikleştirmek (otomasyona dökmek) için attığınız adımlar(steps) nelerdir?

• İşlevi öğrenin (Learn the functionality)

- ✓ Okuma gereksinimleri
- ✓ B.A ile bilgi transferi oturumu
- ✓ Takım arkadaşlarına sor

• Manuel olarak test edin (Manually test it)

- ✓ Her adımı doğru anladığımdan emin olmak
- ✓ Beklenen sonuçları anlayın

• Otomatikleştirin (Automate it)

- ✓ POM sayfaları oluşturun
- § Kullanacağım gerekli öğeleri/yöntemleri ekleyin ve PageFactory tasarım desenini ekleyin
- § Singleton deseniyle bir sürücü sınıfı oluşturun

• TestNG İddialarını kullanarak testleri doğrulayın

79) Otomasyona karşı manuel konumun yüzde kaçı? (Manuel ve otomasyon yüzdeliğin)

• %80-85 otomasyon %15-20 manuel

80) Manuel test yerine otomatik testi ne zaman seçersiniz?

• Test senaryoları yüksek öncelikli test senaryolarıysa.

• İşlevsellik kritik işlevsellik ise.

• Shakeout veya duman testi (smoke test) test durumları.

• Test senaryoları çok uzunsa ve yürütülmesi çok zorsa. Önceliğe dayalı regresyon testi durumları.

• Mümkün olduğu kadar otomasyona geçmeliyiz.

81) Sprint'inizde otomasyonu ne zaman yapıyorsunuz?

• Geliştiricilerin kendi rolleri bittiğinde

• QA/test ortamına kod dağıtıldığında

• Test çerçevesi kurulduğunda

• Tüm manuel testler yapıldığında

• Duman testleri (smoke test) geçtiğinde

82) Test Planı Nedir?

Yazılım Test Planı, test kapsamını ve faaliyetlerini açıklayan belgedir. Bir projedeki herhangi bir yazılımın resmen test edilmesinin temelini oluşturur.

• Test planı, test kapsamını açıklayan bir word belgesidir.

o Yüksek seviye test döngüsü

o Arıza yaşam döngüsü

o Giriş Kriterleri (teste başlamak için gerekenleri tanımlar)

o Çıkış Kriterleri (testin ne bittiğini tanımlar)

TEST PLANI İLE İLGİLİ ISTQB TERİMLERİ: https://www.yazilimtestmerkezi.com/post/yazilim-test-plani

83) Test planlarındaki tablolar nelerdir?

• Test tasarımı, kapsamı, test stratejileri, yaklaşımı Test plan belgesinin içerdiği çeşitli detaylardır.

o Test durumu tanımlayıcısı

o Kapsam

o Test edilecek özellikler

o Test edilmeyecek özellikler

o Test stratejisi ve Test yaklaşımı

o Test çıktıları

o Sorumluluklar

o Personel ve eğitim

o Risk ve Beklenmedik Durumlar

84) Test planı ile KG planı arasındaki fark nedir?

• Bir test planı, ürünü test etmek için ne yapılması gerektiğini ortaya koyar ve kalite kontrolünün hataları ve kusurları belirlemek için nasıl çalışacağını içerir.

• Öte yandan bir KG planı, hataların ve kusurların test edilmesi ve düzeltilmesinden daha çok önlenmesi ile ilgilidir.

85) Akran değerlendirmesi nedir? (peer review)

• Akran değerlendirmeleri, aynı ekipte çalışan kişiler arasında yapılan incelemelerdir. Örneğin, bir QA mühendisi tarafından yazılmış bir test senaryosu, bir geliştirici ve/veya başka bir QA mühendisi tarafından gözden geçirilebilir.

86) Bir sistemi veya modülü yeterince test etmek için yeterli test senaryosu oluşturulduğunda nasıl anlarsınız?

• Her gereksinimi karşılayacak en az bir test senaryosu olduğunda yeterli test senaryosunun oluşturulduğunu söyleyebilirsiniz. Bu, uygulamanın tasarlanan tüm özelliklerinin test edilmesini sağlar.

• A2-İşte bu nedenle gereksinim izlenebilirlik matrisine ihtiyacımız var. Test senaryoları tarafından kaç gereksinimin karşılandığını ve kaç tanesinin hala RTM'den kaldığını söyleyebiliriz.

87) Test senaryolarını kim onaylar?

• Test senaryolarının onaylayıcısı bir kuruluştan diğerine değişir. Bazı kuruluşlarda, QA lideri test senaryolarını onaylarken, bir diğeri bunları akran incelemelerinin bir parçası olarak onaylayabilir.

88) Test planlarını ve test senaryolarını kim yazar?

• Test planları genellikle kalite güvence lideri tarafından yazılırken, test uzmanları genellikle test senaryoları yazar.

89) Test tasarım tekniğinin amacı nedir?

• Test koşullarının belirlenmesi ve Test senaryolarının belirlenmesi.

90) Test senaryosu ve Test komut dosyası arasındaki fark nedir?

• Test senaryosu terminolojisi çoğunlukla Manuel Test için kullanılırken Test Komut Dosyası çoğunlukla Otomasyon Testi için kullanılır

• Test durumu, girdi değerlerini, beklenen çıktıyı ve testin yürütülmesi için ön koşulları belirten bir belgedir. Aynı zamanda senaryonun nasıl test edileceğine ilişkin alt düzey ayrıntıların bir düzenidir.

• Yazılım testindeki bir test komut dosyası, sistemin beklendiği gibi çalıştığını test etmek için test edilen sistemde gerçekleştirilecek bir dizi talimattır.

91) Bir test stratejisine neler dahil edilmelidir?

• Test stratejisi, uygulamanın nasıl test edileceğine ve tam olarak neyin test edileceğine (kullanıcı arayüzü, modüller, süreçler, vb.) ilişkin bir plan içerir. Test için sınırlar belirler ve manuel mi yoksa otomatik testin mi kullanılacağını belirtir.

92) Komut dosyası başarısız olduğunda ne yapacaksınız?

• Tecrübelerime göre, başarısızlığı belirleyeceğim,

o Eğer bu uygulama hatası, senkronizasyon hatası, script sorunu veya ortam arızasından kaynaklanıyorsa, öncelikle sonucu Jenkins üzerinden çoğaltarak analiz ediyorum, sadece fail olanı çalıştırıyorum,

o Senkronizasyon sorunundan kaynaklanıyorsa, örtük, açık ve bazı özel beklenen koşulları kullanarak ekstra zaman ekleyeceğim,

o Komut dosyası sorunuysa, komut dosyamda hata ayıklayacağım (tanımlayacağım) ve düzelteceğim, istisnaları analiz edeceğim,

o gerçek kusur ise, o zaman kusuru kaydedeceğim.

93) Test Senaryosu nedir?

- Test edilen uygulamanın uçtan uca işlevselliğinin beklendiği gibi çalıştığından emin olun
- Testi yapan kişinin, test altındaki uygulamayı nasıl kullandıklarını kontrol etmek ve eylemi gerçekleştirmek için ayağını son kullanıcıların ayakkabılarına koyması gerekir.
- T.S, kendisiyle ilişkili birçok test senaryosuna sahip olabilir, T.S'yi çalıştırmadan önce senaryo için test senaryolarını düşünmemiz gerekir.
    - Test Senaryosu: Oturum açma sayfasını doğrulayın
- ✓ Test Durumu 1: Geçerli bir kullanıcı adı ve şifre girin
- ✓ Test Durumu 2: Parolanızı sıfırlayın
- ✓ Test Durumu 3: Geçersiz kimlik bilgilerini girin
    - Her test durumunda, yürütme için ayrıntılı adımlar ve koşul bulunur
    -

94) Gereksinim İzlenebilirlik Matrisi (RTM) nedir?

Gereksinimler İzlenebilirlik Matrisi (RTM), doğrulama işlemi boyunca gereksinimleri bağlayan bir belgedir. Gereksinimler İzlenebilirlik Matrisinin amacı, bir sistem için tanımlanan tüm gereksinimlerin test protokollerinde test edilmesini sağlamaktır.

• RTM, tüm test senaryolarının gereksinimi karşılayıp karşılamadığından emin olmak için kullanılır. Excel sayfası gibidir.

95) İşlevsel özellikler veya herhangi bir sistem ve geliştirme belgesi yoksa bir sistem için test geliştirmek için ne yapılabilir?

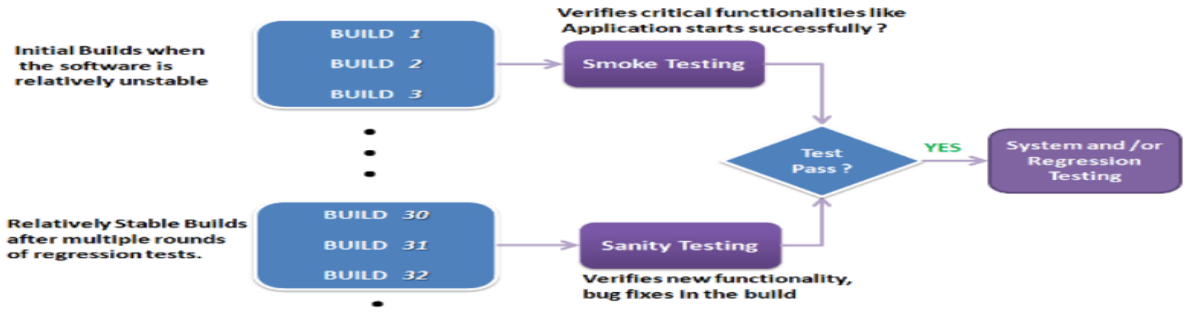
• İşlevsel özellikler veya sistem geliştirme belgeleri olmadığında, test eden kişi ürüne ve koda aşina olmalıdır. Piyasada benzer ürünleri bulmak için araştırma yapmak da faydalı olabilir.

96) Fonksiyonel test türleri nelerdir?

- Birim Test
- Smoke Test
- Sanity Test
- Entegrasyon Test
- Regresyon Test
- Kullanıcı Kabul Testi
- Yerelleştirme Testi
- Ulusallaştırma Testi'dir.

97) Sanity Testing ve Smoke Testing arasındaki fark nedir?

- Adını bir borunun işlevselliğini test etmek için bir ucundan verilen dumanın diğer ucundan çıktığının test edilmesinden alan smoke test, kapsamlı bir teste girmeden önce temel fonksiyonların kontrol edildiği testtir. Bu test ile versiyonun tüm sistem üzerindeki stability'si onaylanır. Bu test ardından Q&A (Kalite) grubu testlerine geçilir.
- Sanity test ise kullanıcıya verilebilecek kadar olgun olan bir versiyon üzerine yeni eklenen bir özellik veya bug çözümü ardından yapılan ilk testtir. Aslında sadece bu özelliklerin ve bug'ların çözümü kontrol edilerek devamındaki testlere engel bir durum olup olmadığı test edilir.
- Yani smoke test ile ürünün ana fonksiyonları, sanity check ile çözülen bug'ları ve eklenen özellikleri test edilir.
- Regression test ile ise yeni gelen çözüm ve özelliklerin ürünün fonksyionlarında bir gerilemeye sebep olup olmadığı kontrol edilir. O nedenle regression test daha çok sanity check ve/veya smoke test ardından koşulur bir testtir.
- Regression test, functional testten farklı olarak ilerlemiş bir yazılım üzerinde daha çok koşulur. Functional test ise spec dokümanına bağlı kalınarak, spec'te yazılan isteklerin çalışıp çalışmadığının sonucunu veren testtir.
- Sanity Testing ve Smoke Testing her ne kadar aynı işe yarar gibi görünse de aslında aralarında önemli temel bir fark var. Ama öncelikle benzeştikleri kısmı yazayım; ikisi de sistemin kabaca test edilmesidir. Aradaki fark, test edilen kısımda ortaya çıkıyor. Smoke Testing yapılan geliştirme sonucu tüm sistemdeki en temel ve kritik özelliklerin test edilmesi iken, Sanity Testing yapılan geliştirme ve etkilenen yakın çevresinin kabaca test edilmesidir (Sanity testing ayrıca bug fixler sonucunda da aynı şekilde kullanılmaktadır.). Sanity Testing başarılı olduğunda asıl kapsamlı testlere geçilir.



98) Sanity Testing'de hangi adımlar yer alır?

• Sanity Testing, duman testine çok benzer. Bir bileşenin veya uygulamanın en temel düzeyde çalıştığından ve daha ayrıntılı testlere devam edecek kadar kararlı olduğundan emin olmak için yapılan ilk testtir.

99) WinRunner ve Rational Robot arasındaki fark nedir?

• WinRunner işlevsel bir test aracıdır ancak Rational Robot hem işlevsel hem de performans testi yapabilir. Ayrıca WinRunner'ın 4 doğrulama noktası ve Rational Robot'un 13 doğrulama noktası vardır.

100) QA ve testing arasındaki fark nedir?

- Kalite Güvencesi, yazılım geliştirme sürecinde ortaya çıkabilecek sorunların veya hataların önlenmesine yardımcı olur. Yazılım testi ise, meydana gelmiş olabilecek herhangi bir arızanın tespit edilmesine ve düzeltilmesine yardımcı olur. Test, ürünün hatasız ve güvenli olduğunu garanti ederken, QA, ürün tasarımının ve işlevselliğinin son kullanıcının beklentilerini karşıladığını doğrular. KG, hedeflere ulaşmaya yardımcı olan tüm politikaları, eğitimi, ekip oluşturmayı ve araçları kapsayan bir yönetim tarzıdır. Öte yandan test, çeşitli uygulamalar için çeşitli testleri kapsayan bir prosedürdür.

- Sunulan fikirlerin bir sonucu olarak, yazılım testinin ve kalite güvencesinin önemini benzersiz bir şekilde değerlendirebiliriz. Her ikisinin de yazılım geliştirme döngüsünde yüksek bir değere sahip olduğunu ve gelişmiş kullanılabilirlik ve işlevselliğe katkıda bulunduğunu belirtir. Belirtilen hedefe ulaşmak için profesyonel bir tavır ve etkili iletişimi sürdürürken ürün kalitesine odaklanarak her ikisi de aynı yönde çalışır.

• KG'nin hedefleri, test etme hedeflerinden çok farklıdır.

• KG'nin amacı hataları önlemektir, uygulama ise testin amacı hataları bulmaktır.

101) Rastgele (random) testi açıklayın.

Rastgele test, geliştiricilerin bir yazılım ürünü için dahili koda bakmadıkları bir kara kutu testidir. Bunun yerine, sonuçların ne olduğunu görmek için sisteme rastgele girdiler girerler. Yaygın bir örnek, bu tamsayılara göre sonuç döndüren bir yazılım işlevini test etmek için rasgele tamsayıların kullanılmasıdır.

• Rastgele test, uygulamanın rastgele oluşturulan girdi verilerini nasıl işlediğini kontrol etmeyi içerir. Veri türleri genellikle yok sayılır ve veri alanına rastgele bir harf, sayı ve diğer karakterler dizisi girilir.

102) Kalite Kontrol ve Kalite Güvencesi arasındaki fark nedir?

• Kalite kontrol (QC) ve kalite güvencesi (QA) yakından bağlantılıdır ancak çok farklı kavramlardır. QC, geliştirilmiş bir ürünü değerlendirirken, QA'nın amacı, geliştirme sürecinin, sistemin veya uygulamanın gereksinimleri karşılayacağını kesinleştirecek bir seviyede olmasını sağlamaktır.

- ✓ Kalite Kontrol (QC), bir ürünün üretim aşamasında yapılan fiziksel kontrollerdir.
- ✓ Kalite Güvence (QA), planlı ve sistematik tüm teknik ve yönetimsel işlemleri içine alan, ürün için optimum kalite seviyesini hedefleyen sistemdir.

103) Bir proje geliştirmede QA'nın rolü nedir?

• QA ekibi, geliştirme için yürütülecek sürecin izlenmesinden sorumludur.

• QA ekibinin sorumlulukları, test yürütme sürecini planlamaktır.

• QA Lead, zaman çizelgelerini oluşturur ve ürün için bir Kalite Güvence planı üzerinde anlaşmaya varır.

• QA ekibi, QA sürecini ekip üyelerine iletti. QA ekibi, test senaryolarının gereksinimlere göre izlenebilirliğini sağlar.

104) Bir QA veya Test yöneticisini iyi yapan nedir?

• Yazılım geliştirme süreci hakkında bilgi

• Verimliliği artırmak için ekip çalışmasını iyileştirin

• Yazılım, test ve kalite güvence mühendisleri arasındaki işbirliğini geliştirin

• Kalite Güvence süreçlerini iyileştirmek.

• İletişim yetenekleri.

• Toplantıları yönetebilme ve onları odaklanmış halde tutabilme

105) Regresyon testi ile yeniden test arasındaki fark nedir?

• Regresyon testi, bir modülde veya sistemde yapılan değişikliklerin önceki sürümler üzerinde olumsuz bir etkisi olmadığından emin olmak için testler yapıyor. Yeniden test etme, yalnızca aynı testi yeniden çalıştırmaktır. Regresyon testi, yaygın olarak sorulan manuel test görüşme sorularıdır ve bu nedenle bu konuyu anlamak için daha fazla araştırmaya ihtiyaç vardır.

106) Hata şiddeti ve hata önceliği arasındaki farkı açıklayın.

• Hata ciddiyeti, hatanın uygulama veya sistem üzerindeki etkisinin seviyesini ifade ederken hata önceliği, bir düzeltme ihtiyacının aciliyet seviyesini ifade eder.

• Ağırlık genellikle finansal kayıp, çevreye verilen zarar, şirketin itibarı ve can kaybı olarak tanımlanır. Bir kusurun önceliği, bir hatanın ne kadar hızlı düzeltilmesi ve canlı sunuculara dağıtılması gerektiği ile ilgilidir.

107) Sistem testi ile entegrasyon testi arasındaki fark nedir?

• Sistem testi için tüm sistem bir bütün olarak kontrol edilirken entegrasyon testi için ayrı modüller arasındaki etkileşim test edilir.

108) İşlevsel ve yapısal testler arasındaki farkı açıklayın.

• İşlevsel test, test cihazının sistem veya uygulamanın spesifikasyona göre çalıştığını doğruladığı davranışsal veya kara kutu testi olarak kabul edilir. Yapısal testler ise koda veya algoritmalara dayanır ve beyaz kutu testi olarak kabul edilir.

109) Pilot ve Beta testi arasındaki fark nedir?

• Bu ikisi arasındaki farklar aşağıda listelenmiştir:

o Ürün son kullanıcıya sunulmak üzereyken bir beta testi, geliştirme döngüsünün ilk aşamasında pilot testler yapılır.

o Beta testinde, uygulamanın kullanıcı gereksinimlerini karşıladığından ve herhangi bir gösterici içermediğinden emin olmak için uygulama birkaç kullanıcıya verilirken, pilot test durumunda ekip üyesi uygulamanın kalitesini artırmak için geri bildirimde bulunur.

110) Alfa Testi (Alpha Testing) Nedir?

Alfa Testi (Alpha Testing), ürün neredeyse kullanılabilir durumdayken, geliştirme sürecinin sonuna doğru yapılan bir test türüdür. Alfa Testi, bir tür kabul testidir. Nihai ürünü son kullanıcılara sunmadan önce olası tüm sorunları ve hataları belirlemek için gerçekleştirilir.

• Geliştiricinin sitesinde son kullanıcı temsilcileri tarafından yayın öncesi testi.

111) Başarısızlık nedir?

• Başarısızlık, belirtilen davranıştan ayrılmadır.

112) Test karşılaştırıcıları nelerdir?

• Bir yazılıma bazı girdiler koyarsanız, ancak yazılımın doğru sonucu üretip üretmediğine asla bakmazsanız, bu gerçekten bir test midir?

• Testin özü, yazılımın doğru sonucu üretip üretmediğini kontrol etmektir ve bunu yapmak için yazılımın ürettiği ile üretmesi gerekeni karşılaştırmalıyız.

• Bir test karşılaştırıcısı, bu karşılaştırmanın yönlerini otomatikleştirmeye yardımcı olur.

113) Yazılım testi sırasında Risk analizinin nasıl gerçekleştirileceğini açıklayın?

• Risk analizi, uygulamadaki risklerin belirlenmesi ve test edilmeleri için önceliklendirilmesi sürecidir. Aşağıdaki risklerden bazıları şunlardır:

1. Yeni Donanım. 1.New Hardware.

2. Yeni Teknoloji. 2. New Technology.

3. Yeni Otomasyon Aracı. 3. New Automation Tool.

4. Kod tesliminin sırası. 4. Sequence of code delivery.

5. Uygulama testi kaynaklarının kullanılabilirliği. 5. Availability of application test resources.

• Bunları üç kategoride önceliklendiriyoruz:

o Yüksek büyüklük: Hatanın uygulamanın diğer işlevleri üzerindeki etkisi.

o Orta: Uygulamada tolere edilebilir ancak arzu edilmez.

o Düşük: tolere edilebilir. Bu tür bir riskin şirket işi üzerinde hiçbir etkisi yoktur.

114) İpek Testi Nedir? (Silk Test)

• Silk Test, uygulamanın regresyon ve işlevsellik testlerini gerçekleştirmek için geliştirilmiş bir araçtır. Silk Test, Window, Java, web veya geleneksel istemci/sunucu tabanlı uygulamaları test ederken kullanılan bir araçtır.

• Silk Test, test planının hazırlanmasına ve bu test planlarının yönetilmesine, veri tabanına doğrudan erişim ve sahanın validasyonunun sağlanmasına yardımcı olur.

115) Master Test Planı ile Test Planı arasındaki fark nedir?

• Ana Test Planı, uygulamanın tüm test ve risk içeren alanlarını içerirken, Test senaryosu belgesi test senaryolarını içerir.

• Ana Test planı, uygulamanın genel gelişimi sırasında yürütülecek her bir ayrı testin tüm ayrıntılarını içerirken, test planı, testin kapsamını, yaklaşımını, kaynaklarını ve gerçekleştirme takvimini tanımlar.

• Ana Test planı, uygulama üzerinde yapılacak tüm testlerin açıklamasını içerirken, test planı yalnızca birkaç test senaryosunun açıklamasını içerir. Birim testi, Sistem testi, beta testi vb. gibi test döngüsü sırasında

• Tüm büyük projeler için Master Test Planı oluşturulur, ancak küçük proje için oluşturulduğunda buna test planı adını verdik.

116) Bir test ne zaman başarılı olarak kabul edilir?

• Testin amacı, uygulamanın gereksinimlere göre çalışmasını sağlamak ve mümkün olduğunca çok sayıda hata ve bug keşfetmektir. Bu, daha fazla işlevi kapsayan ve daha fazla hata ortaya çıkaran testlerin en başarılı olarak kabul edildiği anlamına gelir.

117) Kusur nedir?

• Beklenen sonuç gerçek sonuçla eşleşmediğinde, bu kusurdur.

118) Kusur yoğunluğunu tanımla?

• Hata yoğunluğu, kod satırı başına toplam hata sayısıdır.

119) Kusur Yaşam Döngüsü (DLC) nedir?

• Yeni -> Atandı -> Aç > Düzeltildi -> Yeniden Test Edildi -> Kapat

• New -> Assigned -> Open > Fixed -> Retested -> Close

120) Kusur kategorileri nelerdir?

• Yanlış (Wrong): Gereksinimler uygulamada yanlış uygulanmış.

• Eksik (Missing): Müşteri ve uygulama tarafından verilen gereksinim bu uygulamayı karşılayamadığında.

• Ekstra (Extra): Son müşteri tarafından verilmeyen, ürüne dahil edilen bir gereklilik. Bu her zaman spesifikasyondan bir farklılıktır ancak ürünün kullanıcısı tarafından arzu edilen bir özellik olabilir.

121) Bir kusur bulduğunuzda ne yapmalısınız?

• Bir kusur bulursam, bunu bildirmeden önce, bunun geçerli bir kusur olduğundan emin olmak için gereken hatayı yeniden oluştururum.

• Küçük bir sorunsa, geliştirici masasına gideceğim ve o hemen düzeltebilir.

• Büyük bir sorunsa, JIRA'mı açarım ve kusuru kaydederim.

• Hata olup olmadığından emin değilsem KOBİ (konu uzmanı, uygulamayı herkesten daha iyi bilen kişi demektir) ile görüşeceğim.

122) Geliştirici (developer) bir kusur olmadığını söylüyorsa ne yapmalı?

• Her zaman gerçek bir kusur olduğundan emin oluyorum, bu yüzden onu yeniden üretiyorum.

• Ekran görüntülerini alıyorum ve kusuru yeniden oluşturmak için tüm adımları veriyorum.

• Aslında şu anki projemde karşılaştığım en büyük zorluklardan biri de bu.

123) Bir programı test edip hataların %100'ünü bulabilir misiniz?

• Bir uygulamadaki tüm hataları bulmak çoğu zaman imkansızdır çünkü kaç tane hata olduğunu hesaplamanın bir yolu yoktur. Programın karmaşıklığı, programcının deneyimi vb. gibi böyle bir hesaplamaya dahil olan birçok faktör vardır. Bu Kılavuz test mülakat soruları, test uzmanları tarafından düşünülen en zor sorulardır.

124) Hata ayıklama ve test etme arasındaki fark nedir?

• Hata ayıklama ve test etme arasındaki temel fark, hata ayıklamanın tipik olarak, aynı zamanda hata ayıklama aşamasında hataları düzelten bir geliştirici tarafından gerçekleştirilmesidir. Öte yandan test etme, hataları düzeltmek yerine bulur. Bir testçi bir hata bulduğunda, genellikle bir geliştiricinin düzeltebilmesi için bunu rapor eder.

125) Test nasıl yapılmalıdır?

• Test, uygulamanın teknik gereksinimlerine göre yapılmalıdır.

126) İyi bir test olarak kabul edilen nedir?

• Bir nesnenin veya sistemin işlevselliğinin çoğunu kapsayan test, iyi bir test olarak kabul edilir.

127) Test ne zaman durdurulmalı?

• Test edilen sistemin risklerine bağlıdır. Testi durdurabileceğiniz bazı kriterler vardır.

o Son Tarihler (Test, Yayın)

o Test bütçesi tükendi

o Hata oranı belirli bir seviyenin altına düşer

o Belirli bir yüzde ile tamamlanan test vakaları

o Testlerin sona ermesi için alfa veya beta dönemleri

o Kodun, işlevselliğin veya gereksinimlerin kapsamı belirli bir noktaya kadar karşılanıyor

128) Yukarıdan aşağıya ve aşağıdan yukarıya testler arasındaki fark nedir?

• Yukarıdan Aşağıya test, sistemle başlar ve birim düzeyine kadar iner.

• Ters yönde aşağıdan yukarıya test kontrolleri, genel sisteme arayüz için birim seviyesi. Her ikisinin de değeri vardır ancak aşağıdan yukarıya testler genellikle hataları düzeltme maliyetinin daha düşük olduğu geliştirme döngüsünün başlarında hataları keşfetmeye yardımcı olur.

129) Oluşturduğunuz yürütülebilir dosyaların ortalama boyutu nedir?

• Bu, deneyimlerimizle ilgili basit bir röportaj sorusudur. Yürütülebilir dosyalar oluşturduğunuz herhangi birinin boyutunu biliyorsanız, bu bilgiyi sağlamanız yeterlidir.

130) Ön uç ve arka uç üzerinde testler yaptınız mı?

• Front-End'i test ettiğimde, aslında uygulamayı açıp UI üzerinde test yaparak UI'yi test ediyorum. Kullanıcı arayüzünde herhangi bir şey yaptıysam, değişikliğin veritabanında da yapılıp yapılmadığını görmek için arka uç testi yapmam gerekiyor. Örneğin bir veli iletişim bilgilerini güncellediğimde veya yeni bir uygulama oluşturduğumda, veri tabanına bağlanıyor ve verilerde değişikliklerin uygulanıp uygulanmadığını veya yeni uygulamanın oluşturulup oluşturulmadığını kontrol ediyorum.

131) Ön Uç Testi ile Arka Uç testi arasındaki fark nedir?

• Ön Uç Testi, Grafik Kullanıcı Arayüzü üzerinde yapılırken Arka Uç Testi, veritabanlarının test edilmesini içerir.

• Ön uç, kullanıcının etkileşimde bulunabileceği web sitesi görünümünden oluşurken, arka uç durumunda verileri depolamak için gerekli olan veritabanıdır.

• Son kullanıcı, ön uç uygulamasının GUI'sine veri girdiğinde, girilen bu veri veritabanında depolanır. Bu verileri veritabanına kaydetmek için SQL sorguları yazıyoruz.

132) Test sırasında bulduğunuz en zor problem nedir?

• (Bu, bir örnek vermeniz gereken basit bir mülakat sorusudur}. Cevabınız işinizi belirleyeceğinden, bu en zor manuel test mülakat sorularından biridir. Sorun çözme becerileriniz, yeni şeyler öğrenme hevesiniz ve işe olan bağlılığınız cevaplarınızda kendini gösterecek şekilde cevap vermelisiniz.

133) Scrum'da karşılaştığınız zorluk nedir?

• Scrum, çapraz fonksiyonel ekibi vurguladığından (bu, geliştiricinin test edebilmesi ve testçilerin geliştirme yapabilmesi gerektiği anlamına gelir) geleneksel bir QA testçisi olarak geliştirme ekibinin bir parçası olmak zordur. Çünkü genellikle QA'lar nasıl kod yazılacağını bilmiyorlar. Bu yüzden kendimi çok rekabetçi tutmak zorundayım. Ne zaman vaktim olsa Java gibi daha çok kodlama öğreniyorum.

• Zaman değişikliği sorunu-> Veritabanına girdiğim bir tarihi kaydettiğimde, daha önce

134) Otomasyon Testi Nedir?

• İnsan müdahalesini azaltan otomatik test yapma süreci, bu otomasyon testidir.

• Otomasyon testi, QTP, Selenium, WinRunner vb. otomasyon araçları yardımıyla yapılır.

• Otomasyon testinde, uygulamayı test etmek için test komut dosyasını çalıştıran bir araç kullanıyoruz; bu test komut dosyası manuel veya otomatik olarak oluşturulabilir. Test tamamlandığında, araçlar otomatik olarak test raporunu oluşturur ve rapor eder.

135) Ne zaman otomatikleştireceksin?

• Çok fazla manuel çaba gerektiriyorsa. En az bir kez manuel çalıştırıyorum ve ondan sonra otomatikleştiriyorum.

• Otomasyon, çoğu tekrarlayan işlevsellik için iyidir.

136) Hangi testler otomatikleştirilebilir?

• Regression tests • Smoke tests • Functional tests • API • Database

137) Ne zaman otomatikleştirmeyeceksiniz?

• İşlevsellik değişmeye devam ederse

• İşlevsellik tüm proje boyunca yalnızca bir kez kullanılıyorsa

• Geçici test otomatikleştirilemez.

138) Bir scrum sprint'in süresi nedir? Sprintin ne kadar sürüyor?

• Mevcut projemde script döngümüz 4 haftadır. Buradaki sprintin ne kadar sürüyor? 2 hafta mı 4 hafta mı? (bazen soru sormak iyidir. ATM gibi davranmamanız gerektiğini unutmayın. Genelde sadece soru cevaplayanları unuturlar. Bir denge olmalı.)

• Ekibimiz 7 kişiliktir. 1 SM, 1 PO, 3 geliştirici, 1 MT, 1 AT

139) Hız nedir?

• Hız, takımın sprint baskıda ilerleme hızıdır.

• İki farklı scrum takımıyla karşılaştırılamayacağını da söyleyebilirim.

140) Scrum'daki engeller hakkında ne biliyorsunuz? Engellere birkaç örnek veriniz.

• Engeller, scrum ekibinin karşılaştığı ve çalışma hızını yavaşlatan engeller veya sorunlardır.

• Eğer bir şey scrum ekibinin işi "Bitti" hale getirmesini engellemeye çalışıyorsa, bu bir engeldir.

• Engeller herhangi bir biçimde gelebilir. Bazı engeller şu şekilde verilmiştir:

- ✓ Kaynak eksik veya hasta ekip üyesi
- ✓ Teknik, operasyonel, organizasyonel sorunlar
- ✓ Yönetim destekleyici sistem eksikliği
- ✓ İş sorunları
- ✓ Hava durumu, savaş vb. gibi dış sorunlar
- ✓ Beceri veya bilgi eksikliği

• Çözüm: Takım çalışması, sıkı çalışma, iyi iletişim kurma, çevrimiçi bağlantı, mentorluk ve eğitim

141) Çevik ve Scrum arasındaki fark ve benzerlik nedir?

• Çevik geniş bir yelpazedir, proje yönetimi için kullanılan bir metodolojidir, Scrum ise Çevik'in sadece süreci ve adımlarını daha kısa bir şekilde tanımlayan bir şeklidir.

• Agile bir uygulama iken, scrum bu uygulamayı sürdürmek için bir prosedürdür.

• Çevik'in projeleri adım adım veya aşamalı olarak tamamlamayı içermesi benzerliği. Çevik metodoloji, doğası gereği yinelemeli olarak kabul edilir. Çevik'in bir formu olan Scrum, Çevik'inkiyle aynıdır. Aynı zamanda artımlı ve yinelemelidir.

142) Artış nedir? Açıklayınız.

• Artış, bir sprint sırasında tamamlanan tüm ürün biriktirme listesi öğelerinin toplamıdır.

• Her artış, kümülatif olduğu için önceki tüm sprint artış değerlerini içerir.

• Hedefimize ulaşmak için bir adım olduğu için sonraki sürümde mevcut modda olmalıdır.

143) "Yapı kırıcı" nedir? "build breaker"

• Yapı kırıcı, yazılımda bir hata olduğunda ortaya çıkan bir durumdur.

• Bu ani beklenmeyen hata nedeniyle, derleme işlemi durur veya yürütme başarısız olur veya bir uyarı üretilir.

• Test edenin sorumluluğu, hatayı ortadan kaldırarak yazılımı normal çalışma aşamasına geri getirmektir.

144) Daily stand-up'tan ne anlıyorsunuz?

• Günlük stand-up, tüm ekibin aşağıdaki üç soruya yanıt bulmak için yaklaşık 15 dakika boyunca toplandığı günlük bir toplantıdır (en çok tercihen sabahları yapılır).

- ✓ Dün ne yapıldı? Bugün için planın nedir?
- ✓ Görevinizi tamamlamanızı engelleyen herhangi bir engel veya engel var mı?

• Günlük stand-up, ekibi motive etmenin ve o gün için bir hedef belirlemelerini sağlamanın etkili bir yoludur.

145) Scrumban hakkında ne biliyorsun?

• Scrumban, yazılım geliştirme için Scrum ve Kanban tabanlı bir modeldir.

• Bu model, sürekli bakım gerektiren, çeşitli programlama hataları olan veya ani değişiklikler olan projeler için özel olarak kullanılmaktadır.

• Bu model, bir programlama hatası veya kullanıcı hikayesi için bir projenin minimum sürede tamamlanmasını teşvik eder.

146) Çevik kalite stratejilerinden bazılarını belirtin?

• Yineleme (• Iteration)

• Yeniden düzenleme (• Refactoring)

• Dinamik kod analizi (• Dynamic code analysis)

• Kısa geri bildirim döngüleri (• Short feedback cycles)

• İncelemeler ve inceleme (• Reviews and inspection)

• Standartlar ve Talimatlar (• Standards and guidelines)

• Kilometre taşı incelemeleri (• Milestone reviews)

147) Çevik Manifesto ve İlkeleri hakkında bilginiz var mı? Kısaca açıklayın. (Agile Manifesto & its Principles)

• Çevik/scrum rolüne aday olan çoğu kişinin uçlarda olması gereken teori budur.

• Bu sorunun bir parçası olarak mümkün olduğunca dört manifesto değeri ve 12 ilke açıklanmalıdır.

• %100 doğru bir şekilde açıklanmasa bile iyi olmalı, ancak değerlerin ve ilkelerin niyetleri ortaya çıkmalı örn.

• Manifesto

- ✓ Süreçler ve araçlar üzerinden bireyler ve etkileşimler
- ✓ Kapsamlı dokümantasyon üzerinden çalışan yazılım
- ✓ Sözleşme müzakeresi üzerinden müşteri işbirliği
- ✓ Bir planı takip ederek değişime yanıt verme

• Rehber ilkeler

- ✓ Müşteri Memnuniyeti
- ✓ Karşılama Değişen Gereksinimler
- ✓ Çalışan Yazılımlar Sıklıkla Teslim Edilir (Aylar yerine Haftalar)
- ✓ İş İnsanları ve Geliştiriciler Arasında Yakın, Günlük İşbirliği
- ✓ Proje, güvenilmesi gereken motive olmuş bireyler etrafında inşa edilmiştir.
- ✓ Yüz Yüze Konuşma en iyi iletişim şeklidir
- ✓ Çalışan yazılım, ilerlemenin birincil ölçüsüdür
- ✓ Sürdürülebilir kalkınma, sabit bir tempoyu koruyabilme
- ✓ Teknik mükemmelliğe ve iyi tasarıma sürekli dikkat
- ✓ Sadelik- Yapılmayan iş miktarını en üst düzeye çıkarma sanatı - esastır
- ✓ Kendi kendini organize eden ekiplerden en iyi mimariler, gereksinimler ve tasarımlar ortaya çıkar
- ✓ Ekip düzenli olarak nasıl daha etkili olunacağını düşünür ve buna göre kendini ayarlar

148) Burn-up ve Burn-down charts çizelgelerinin kullanımı nedir?

• burn-up chart bir projede tamamlanan iş miktarını gösterirken, burn-down chart bir projeyi tamamlamak için kalan iş miktarını gösterir.

• Bu nedenle, bir projenin ilerlemesini izlemek için burn-up ve burn-down charts kullanılır.

149) Çevik modelin herhangi bir dezavantajı var mı? Öyleyse, açıkla.

• Evet, Çevik modelin bazı dezavantajları var, bazıları şöyle;

o Bir görevi tamamlamak için gereken çaba hakkında bir tahminde bulunmak kolay değildir. Gereken toplam çaba hakkında bir fikir edinmek zorlaştığından, büyük projeler söz konusu olduğunda daha sorunlu hale gelir.

o Bazen, projenin tasarımına ve dokümantasyonuna gerektiği gibi odaklanmak mümkün değildir.

o Müşteri gereksinimlerinin tam olarak anlaşılmaması durumunda nihai proje müşteri gereksinimlerini karşılamayacaktır. Böylece müşteri memnuniyetsizliğine yol açacaktır.

o Yalnızca Çevik metodolojilerde önemli deneyime sahip lider önemli kararlar alabilir. Çok az deneyimi olan veya hiç tecrübesi olmayan ekip üyeleri karar alma süreçlerine dahil olmazlar, bu nedenle bilgilerini ilerletme şansları olmaz.

150) Çevikte Sıfır Sprint ve Spike tanımlayın.

• Sıfır Sprint, Agile'da ilk sprintin hazırlık aşaması olarak tanımlanabilir.

o Projeye fiilen başlamadan önce yapılması gereken bazı faaliyetler vardır.

o Bu aktiviteler Sıfır sprint olarak kabul edilir; bu tür faaliyetlere örnekler: geliştirme ortamının oluşturulması, birikmiş işlerin hazırlanması vb.

• Spike, sprintler arasında alınabilecek hikâye türüdür.

o Sivri uçlar, araştırma, tasarım, prototip oluşturma ve keşif gibi tasarım veya teknik konularla ilgili faaliyetler için yaygın olarak kullanılır.

o İki tür sivri uç vardır – fonksiyonel sivri uçlar ve teknik sivri uçlar.

151) Scrum Master'ın rolü nedir?

• Scrum ustası, Scrum takımının hem lideri hem de koçudur.

• SM, takıma hizmet etmek ve performansı etkileyebilecek her türlü bloktan korumaktan sorumludur.

• SM'nin ana rolü, takımını sprint hedefine ulaşmak için motive etmektir.

• Her üyenin Çevik ve Scrum ilkelerinin ve uygulamalarının uygulanmasına aşina olduğu, kendi kendini organize eden ve motive olmuş bir ekip oluşturmaya odaklanmıştır.

• SM, taahhüt edilen görevleri düzgün bir şekilde yürütüyorlarsa, scrum ekibi üzerinde uygun bir kontrol sağlar.

• Sprint hedefine etkili bir şekilde ulaşabilmeleri için ekibin verimliliğini ve üretkenliğini artırmaktan da sorumludur.

152) Scrum'daki bir hikâye noktası hakkında ne biliyorsunuz?

• Scrum'daki bir hikâye noktası, belirli bir görevi gerçekleştirmek veya tamamlamak için gereken toplam çabaların tahmini için birimdir.

153) Scrum metodolojisinde Sashimi'nin rolü nedir?

• Sashimi, Scrum metodolojisinde önemli bir rol oynar.

• Sashimi, geliştiriciler tarafından oluşturulan tüm fonksiyonların tamamlandığını kontrol etmek için Scrum tarafından kullanılan bir tekniktir.

• Bu teknik kullanılarak bir ürünün yapısında kullanılan analiz, tasarım, kodlama, test ve dokümantasyon gibi tüm gereksinimler kontrol edilir ve ancak bundan sonra ürün teşhir edilir.

154) Çevik test teriminden ne anlıyorsunuz?

• Çevik test, tamamen yazılım geliştirmenin çevik ilkelerine dayanan bir yazılım testi uygulamasıdır. Gereksinimlerin ürün sahibi ve ekip arasındaki işbirliğinin sonucu olduğu yinelemeli bir metodolojidir.

Çevik ilke ve uygulamalar, projenin başarıyla tamamlanması ile müşteri gereksinimlerini karşılamak için uygulanmaktadır.

155) Scrum yerine şelale(waterfall) kullanılması önerildi mi? Evet ise, ne zaman olduğunu açıklayın.

• Evet, bazen Scrum yerine şelale modeli kullanılması önerilir.

• Müşteri gereksinimlerinin basit, iyi tanımlanmış, tam olarak anlaşılmış, öngörülebilir olması ve proje tamamlanana kadar değişikliğe uğramaması durumunda yapılır.

156) Scrum neden projeler için otomatik test kullanımını teşvik ediyor?

• Scrum, projenin mümkün olan en hızlı şekilde teslim edilmesini sağlamak için otomatik (otomatik performans veya otomatik regresyon) testinin kullanılmasını teşvik eder. Otomasyon için kullandığınız bazı araçları açıklayabilirsiniz.

157) Çevik için bazı yaygın matrisleri açıklayın.

• Hız -> Hız, son 3-4 sprintteki ortalama puan sayısıdır. Öykülerin onaylanmış tüm tahminlerinin toplamı ile ölçülür. Kapasite, ilerleme vb. hakkında bir fikir verir.

• Kümülatif Akış Şeması -> Bunun yardımıyla tekdüze iş akışı üzerinde bir inceleme yapılır. Bu diyagramda/grafikte, x ekseni zamanı, y ekseni ise efor sayısını temsil eder.

• İş Kategorisi Tahsisi -> zaman yatırımı hakkında hızlı bilgi veren önemli bir faktördür, yani zamanın nereye yatırıldığı ve zaman faktörü olarak hangi göreve öncelik verilmesi gerektiği.

• Zaman Kapsamı -> Test sırasında bir koda verilen zamandır. Test takımı tarafından çağrılan kod satırı sayısının ve ilgili kod satırlarının toplam sayısının bir faktörü olarak yüzde olarak hesaplanır.

• Sağlanan İş Değeri -> Takımın çalışma verimliliğini ifade eden bir terimdir. İş hedeflerine öncelik, karmaşıklık ve yatırım getirisi düzeyine göre sayısal değerler 1,2,3 vb. atanır.

• Kusur Giderme Farkındalığı -> Ekibin kaliteli bir ürün teslim etmesine yardımcı olan faktördür. Aktif sayıdaki kusurların belirlenmesi, bunların bilinmesi ve giderilmesi, yüksek kaliteli bir ürün sunmada önemli bir rol oynar.

• Hata Çözüm Süresi -> Ekip üyelerinin hataları (hataları) tespit ettiği ve hata çözümü için bir öncelik belirlediği bir prosedürdür. Hataları/hataları düzeltme veya kusur çözme prosedürü, kusur resminin temizlenmesi, kusur tespitinin planlanması, kusur tespitinin tamamlanması, oluşturulması ve çözüm raporunun ele alınması gibi birçok süreçten oluşur.

• Sprint Burndown Matrisi -> Sprint Burndown grafiği, Scrum döngüsü sırasında uygulanmayan veya uygulanan sprintlerin sayısını temsil eden bir grafiktir. Bu matris, sprint ile tamamlanan işi takip etmeye yardımcı olur.

158) Çevik modeli kullandığınız bazı metodolojileri ve geliştirmeleri adlandırın.

• Çevik modelin kullanılabileceği bazı metodolojiler ve geliştirmeler şunlardır:

o Kristal metodolojileri

o Yalın yazılım geliştirme

o Dinamik geliştirme ve Özellik odaklı geliştirme

159) Scrum M/Ürün O/Agile ekip üyesi olarak deneyiminizi paylaşın ve birincil sorumluluklarınız nelerdi?

• Bu sorudaki püf nokta, anlatırken kendi kendini organize eden ve motive eden bir ekip gösterip göstermediğinizdir.

160) Projenizdeki sprintlerin/yinelemelerin uzunluğu ne kadardı?

• Buradaki fikir, ne tür bir ortamda çalıştığınızı yargılamaktır. Bu uzunluk başlangıçta sabitlendi mi ve hiç değişmedi mi gibi bir takip sorusu kesinlikle olacak. Bu uzunluktan daha uzun veya bundan daha az denediniz mi?

161) "Planning Poker" tekniği hakkında ne biliyorsunuz?

• Scrum Poker olarak da bilinen planlama pokeri, planlama ve tahmin için kullanılan kart tabanlı çevik bir tekniktir. Poker planlama tekniğinin bir oturumunu başlatmak için, ürün sahibi tarafından çevik kullanıcı hikayesi okunur.

• Poker planlama tekniğinde gerçekleştirilen adımlar şunlardır:

o Her tahminci, hikaye puanlarını, ideal günleri veya takımın tahmin için kullandığı başka bir şeyi belirtmek için 0, 1, 2, 3, 5 vb. değerlere sahip bir deste poker kartına sahiptir.

o Her tahminci, ürün sahibiyle bir görüşme yapar ve daha sonra bağımsız tahminlerine dayalı olarak özel olarak bir kart seçer.

o Tüm tahminciler tarafından aynı değere sahip kartlar seçilirse, tahmin olarak kabul edilir. Değilse, tahminci tahminlerinin yüksek ve düşük değerlerini tartışır.

o Sonra yine her tahminci özel olarak bir kart seçer ve ortaya çıkarır. Bu poker planlama süreci, genel bir anlaşmaya varmak için tekrarlanır.

162) Projelerinizde kullanıcı hikâye haritalama ve hikâye tahminini nasıl yaptınız?

• Poker planlama, tişört, beden ölçülendirme gibi herhangi bir tahmin tekniği kullandınız mı? Projenizde hangi tekniği kullanırsanız kullanın, bunu çok net bir şekilde belirtin.

163) Çevik test metodolojisinin diğer test metodolojilerinden farkı nedir?

• Çevik test metodolojisi, tüm test sürecinin birden çok küçük kod segmentine bölünmesini içerir. Her adımda, bu kod segmentleri teste tabi tutulur.

• Ekip iletişimi, optimal sonuçlar için stratejik değişiklikler ve diğerleri gibi çevik test metodolojilerinde yer alan bir dizi ek süreç vardır.

164) Scrum ekip üyelerini idare ederken projenizde karşılaştığınız en büyük zorluk nedir?

• Scrumın ilk aşamalarında genellikle karşılaşılan zorluklar, hızın dengelenmesi, ekip üyelerinin çatışmaları, zaman boksuna bağlı kalma vb.

o Uygulama test edilecek kadar kararlı olmalıdır. o Her zaman zaman kısıtlaması altında test etme

o Gereksinimleri anlamak.

o Alan bilgisi ve iş kullanıcı perspektifi anlayışı.

165) İlk önce hangi testler yapılmalı?

• Komple Uygulamanın Test Edilmesi.

o Regresyon testi.

• Nitelikli test uzmanlarının olmaması.

o Değişen gereksinimler.

• Kaynak, araç ve eğitim eksikliği

166) Scrum Master sertifikanız var mı?

• Sertifikalı bir scrum master iseniz, sertifika sınavı, alınan puan ve sertifika sınavını geçme yılı gibi sertifikanızın ayrıntılarını paylaşmanız yeterlidir. Sertifikanız yoksa, belirli bir alandaki deneyiminizi belirtin ve vurgulayın. Ayrıca, yakın gelecekte sertifikaya yatırım yapmayı planlıyorsanız görüşmeciye bildirin.

167) Çevik sertifikanız var mı? Neden bu sertifikayı seçtiniz?

• Bir projeyi en kısa sürede tamamlamak için Çevik ve Scrum metodolojileri kullanılır.

• Çevik ilkelerin uygulanması müşteri memnuniyeti sağlarken, scrum gereksinimlere göre esnek özelliği ile bilinir.

168) Daha önce offshore ekibiyle çalıştınız mı?

• Hayır, çalışmıyorum...

• Offshore, temel olarak ekibin farklı bir ülkede bulunmasına rağmen hala şirketiniz tarafından istihdam edildiği anlamına gelir.

169) Ortak UI test otomasyon araçları nelerdir?

• Selenium

o Cucumber

o TestNG

• Appium

• Protractor

• Winium

• UFT/QTP

• Katalon Studio

170) Test Yazılımı Nedir? Test eşyası mı?

• Uygulamanın test edilmesinde yardımcı olan yazılımın alt kümesidir.

• Test yazılımları, testleri planlamak, tasarlamak ve yürütmek için gereklidir. Testte kullanılan belgeler, komut dosyaları, girdiler, beklenen sonuçlar, kurulum ve ek yazılım veya yardımcı programları içerir.

• Test yazılımı, bir yazılım paketini test etmek için gerekli olan tüm yardımcı programlar ve uygulama yazılımlarının birleşimine verilen terimdir.

Özeldir çünkü;

o Farklı amaç

o Kalite için farklı metrikler ve

o Farklı kullanıcılar

171) Bir istemci veya sunucu ortamı testi nasıl etkiler?

• İstemci veya sunucu teknolojileri ile çalışırken veri aktarım veri aktarım hızı, donanım ve sunucu vb. gibi testi etkileyen birçok çevresel faktör vardır, testler kapsamlı olacaktır.

• Zaman sınırımız olduğunda entegrasyon testi yapıyoruz. Çoğu durumda, istemci veya sunucu ortamı için uygulamanın yeteneklerini incelemek için yük, stres ve performans testini tercih ederiz.

Kaynak:

- Albert Document

Zeliha Öznük