# Regression test nedir ? Kısaca açıklayınız.

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

# A/B test nedir ? Kısaca açıklayınız

A / B testi (bölme testi veya kova testi olarak da bilinir) hangisinin daha iyi performans gösterdiğini belirlemek için bir web sayfasının veya uygulamanın iki sürümünü birbiriyle karşılaştırmanın bir yöntemidir. AB testi, esas olarak, bir sayfanın iki veya daha fazla varyantının rasgele kullanıcılara gösterildiği bir deneydir ve hangi varyasyonun belirli bir dönüşüm hedefi için daha iyi performans gösterdiğini belirlemek için istatistiksel analiz kullanılır.

# Black box / white box test kavramlarını açıklayınız.

**Kara Kutu (Black Box) Yaklaşımı**

Bu yaklaşımda sızma testinin yapılacağı sistem ile ilgili bir bilgi önceden verilmez. Test edici bir bilgisayar korsanı gibi sisteme sızmaya çalışır. Hedef sisteme sızmak için sistemle ilgili bilgi toplanır; zafiyetler ve açıklar taranır. Harici veya son kullanıcı bakış açısından test etmeyi içerir. Test edicinin yanlışlıkla sisteme zarar verme ihtimali bulunmaktadır.

**Beyaz Kutu (White Box) Yaklaşımı**

Sızma testini yapacak ekibe firmada bulunan tüm sistemler hakkında bilgiler verilir. Black box yaklaşımına göre kuruma daha fayda sağlanır. Zafiyetleri bulmak kolaylaşır ve önlem alınması için geçen süre de azalmaktadır. Ekibin zarar verme riski çok azdır.

# Mutation test nedir ? Kısaca açıklayınız.

Mutasyon Testi, kaynak koddaki belirli ifadeleri değiştirdiğimiz(mutant) ve test senaryolarının hataları bulabildiğini kontrol ettiğimiz bir tür yazılım testidir. Temelde birim testi(unit test) için kullanılan bir beyaz kutu testi türüdür. Mutant programdaki değişiklikler son derece küçük tutulur, bu nedenle programın genel hedefini etkilemez.

# Behavior Driven Development (BDD) nedir, neyi amaçlamaktadır ?

BDD yaklaşımı ilk olarak 2009 yılında Dan North tarafından ortaya atılmıştır. Açılımı Behavior Driven Development olan BDD, TDD yaklaşımının karmaşıklığı gidermek amacıyla ortaya çıkmıştır. Yazılım süreçlerinin daha test odaklı gitmesini sağlayan bir yaklaşımdır. TDD yaklaşımında olduğu gibi burada da yazılım geliştirmeye başlamadan önce test senaryolarının yazılması desteklenmektedir.

BDD’nin en güzel yanlarından biri konuşma dilinde test senaryoları yazmamıza olanak sağlamasıdır. İş analistleri müşteri ile yaptığı görüşmeler sonrasında ihtiyacı anlayarak user storyler oluşturmakta ve oluşturulan bu user storyler üzerinden de test senaryoları hazırlanarak koda dökülmektedir. Kısacası müşterinin ihtiyacı konuşma dilinde koda döküldüğü için müşteri ile ortak bir dilde buluşmaya olanak sağlayan bir yaklaşımdır aslında. Kolay bir yazım yöntemi olmasıyla birlikte “Given”, “When”, ”Then” olarak 3 başlıkta senaryolar kurulmaktadır.

# Agile test quadrant nedir ? Quadrant’ların kapsamını kısaca açıklayınız.

Brian Marick tarafından 2003 yılında geliştirilen bu model, farklı test türlerini sınıflandırmak ve bir test için gereken kaynakları erken bir aşamada belirlemek için kullanılır.X ekseni ve Y ekseni kuralına göre 4 çeyreğe dağıtılır

**Çeyrek 1**

Bu kadran, geliştirme aşamasında ekibi destekleyen teknolojiye yönelik testleri açıklar. Bu testler her zaman otomatiktir ve birincil amaçları, işlevlerin çalışır durumda olduğundan emin olmaktır. Yazılımın üzerine inşa edildiği temeli doğrular.

Örnekler — Birim testi, Entegrasyon testi, Bileşen testi

İdeal olarak, testlerin çoğu, bir ürün geliştirilirken aynı anda yazıldığı için 1. Çeyrekte yazılır.

**Çeyrek 2**

Geliştirmeyi destekleyen, işle ilgili testlerdir. Çeyrek 2 testleri temel olarak soruları yanıtlamak ve bilgileri keşfetmek için kullanılır. Bu da, oluşturulmakta olan uygulamanın özelliklerini doğrular.

Örnekler — Fonksiyonel testler, otomatik test altında UI testleri. Bu çeyrekteki manuel test örnekleri, web sayfaları gibi prototiplerin kullanılmasını içerir.

Bu testler geliştirme sırasında ve sonrasında yapılır. Bir yazılımın son sürümünden önce hataların ortaya çıkarılmasına yardımcı olurlar.

**Çeyrek 3**

Bu kadran, ürünü eleştiren işle ilgili testleri içerir. Esas olarak manuel testler içerir, ancak buna otomasyon eklenebilir. Ürünün mevcut çalışma durumu hakkında değerli geri bildirim sağlarlar. Daha çok kullanıcı odaklıdırlar. Kullanıcıların ürünle nasıl etkileşimde bulunduğunu belirlemeye çalışarak kullanıcı deneyimine odaklanırlar. Çeyrek 3 testleri, uygulamanın iş akışını anlamaya çalışır.

Örnekler — Keşif testi, Kullanılabilirlik testi, A/B testi.

Geliştirmenin tamamlanmasından önce veya sonra yapılabilirler. Çeyrek 3 testinin arkasındaki ana amaç, nelerin daha fazla geliştirilebileceğini bulmak ve yazılımdaki hataları keşfetmektir.

**Çeyrek 4-**

Bu kadran, ürünü de eleştiren teknolojiye yönelik testlere sahiptir. Tüm testler bu kadran altında otomatikleştirilir. Burada test etmek için özel araçlar kullanılır. Quadrant 3 testlerinin temel amacı, uygulama hakkında hedeflenen bilgileri sağlamaktır.

Örnekler — Performans testleri, Yük testleri, Güvenlik testleri, Güvenilirlik testleri.

Uygulamada neyin en önemli olduğuna bağlı olarak Quadrant 4 testleri yapılır. Üründeki özelliklerin önceliğine odaklanırlar. Daha sonra bilinçli kararlar vermek için analiz için kullanılabilecek verileri ölçmek için kullanılırlar.

Çevik Test Çeyreği Metodolojisi, ürünün tüm planlama, geliştirme ve yayınlama aşaması boyunca kullanılabilir.

Not: Çevik Test Çeyrekleri, 4 çeyreğin tümünden testlerin gerekli olduğunu veya testlerin yapılması gereken herhangi bir kronolojik sırayı belirtmez.