Q1-https://martinfowler.com/articles/practical-test-pyramid.html adresindeki yazıyı okuyup özetini çıkarınız.

A1-Üretime hazır yazılım,üretime geçmeden önce test gerektirir.Önceden ve nadir de olsa halen bu,testerlar tarafından yapılır. Yazılım geliştirime disiplini geliştikçe test mantığı da değişti.Testi yapılacak yazılımı testerlardan alıp otomatikleştirmeye yönlenildi.Bu sayede testinin uzun sürmesi beklenen yazılımlar çok daha kısa sürelerde testinin yapılmasını olanak sağlanmış oldu.Etkili bir yazılım testi yaklaşımına sahip olmak,yazılımcıların hızlı ve güvenli hareket etmesini sağlar.

Test piramidi otamatik testler konusunda bilinmesi gereken anahtar bir kavramdır.Piramit katmanlardan oluşur ve her katman da farklı sayıda testler uygulanır.3 katman vardır.

-Unit Test

-Service Test

-User Interface Test

Test yazarken dikkat edilmesi gereken bazı hususlar vardır.Farklı ayrıntı düzeyine sahip testler yazılımalıdır ve ne kadar yüksek seviye alınırsa o kadar az test yapmak gerekir. Test yaklaşımında kullanılan bazı toollar ve libraryler şunlardır.

-Junit

-Mockito

-Wiremock

-Pact

-Selenium

-Rest-assured

Unit testler diğer tüm testlerin en dar kapsamıdır ve diğer testlerden büyük ölçüde fazla sayıda bulunur.Functional bir yazılım dilinde unit tek bir fonksiyondur.Oop kullanan bir yazılım dilinde unit bir methoddan classa kadar değişebilir.

Mock ve stub testleri iki farklı türde test doubledır.Yani 2sinden daha fazlası vardır.Gerçek bir class modüle veya funcitonu onun sahte versiyonuyla değiştirmemiz anlamına gelir.

Unit testler katman bağımsız tüm üretim kodu sınıfları için yazılabilir.Unit testler önemsiz olmayan tüm kodların test edilmesini sağlamalıdırlar.Aynı zamanda uygulamaya çok yakından bağlı olmamalıdırlar.Üretim kodu yeniden düzenlendiğinde unit testler bozulabilir.Bunun önlemek için de dahili kod yapısını yansıtmadan gözlemlenebilir davranış için testler yapılır.

Test yapısı ise tüm testler için iyi bir yapıdır.Test verileri ayarlanmalıdır.Method test altında aranmalıdır.Beklenen sonucun döndüğünü idda etmeliyiz.Given when then yapısı vardır.Bu yapı ise neler veriliyor neler ne zaman çağırılıyor ve sonrasında ne oluyor şeklinde bir yapıdır.Bu yapı testlerin kolay okunurluluğunu ve tutarlı olmasını sağlar.

Java için unit testler Junit ile yazılır.Testte gerçek bir class stub ile değiştirmek istenilirse Mockito kullanılır.

Integration testler uygulamanın dışında olan tüm parçalarla uygulamanın entegrasyonu test edilir.Şu alanlarda integration testler yazılabilir.

-Servicelerin rest apiına yapılan çağrılar

-Veritabanından okuma,veritabanına yazma

-Diğer uygulamaların apilarını çağırma

-Kuyruklardan okuma ve kuyruklara yazma

-Dosya sistemine yazma.

Test piramidinde integration testler unit testlerden daha üst seviyededir.

Service entegrasyonunda otomatik testler çalıştırılırken gerçek sunuculara takılmamalıdır.Entegrasyon testlerimizi yaparken kendi sahte sunucumuzu çalıştırabiliriz ve bu wiremock gibi toollar ile çok kolaydır.

Contract testlerde sistemimizi birçok küçük servise ayırmak,genellikle bu servicelerin belirli interfaceler aracılığıyla birbirleriyle iletişim kurması gerektiği anlamına gelir.Farklı uygulamalar arasındaki interfaceler farklı şekil ve teknolojilerde olabilir.Bunların bazıları

-Https aracılığıyla rest ve json

-gRpc gibi bir şey kullanan Rpc

-Queular kullanarak olaya dayalı bir mimari oluşturmak.

Her interface için consumer ve provider vardır.Consumer providerdan aldığı verileri işler.Provider gerekli tüm uç noktalarla bire rest api oluşturur ve consumer veri almak için bu apiya çağrı yapar.

UserInterface testleri uygulamamızın ui testlerinin doğru çalıştığını test eder.

End to end testlerde userinterfacelerle test yapmak uygulamayı test etmenin en uçtan uca şeklidir.

Userinterface uçtan uca testinde Selenium ve WebDriver protokolü birçokdeveloper için tercih edilen araçtır.

Test piramidinde ne kadar yükselinirse oluşturulan özelliklerin bir kullanıcının bakış açısından doğru çalışıp çalışmadığını test etme alanına girilme olasılığı o kadar artar

Testleri yazmak, okumak,anlamak ve çalıştırmak zaman alır.Bu yüzden gereksiz test yazmaktan kaçınılmalıdır.

İyi ve temiz bir test kodu yazmak için bazı dikkat edilmesi gereken hususlar vardır.

-Üretim kodu yazılır gibi test kodu yazılmalıdır.

-Her koşul ayrı ayrı test edilmelidir.

-Given when then yapısı test yazmak için iyi yapılardır.

-Okunabilirlik önemlidir.

Q2- Regression test nedir ? Kısaca açıklayınız.

A2-Regrasyon testi, yazılımdaki bir kod değişikliğinin ürünün mevcut işlevselliğini etkilemediğini doğrulamak için yapılan bir test türüdür.Bu test, ürünün yeni işlevler,hata düzeltmeleri veya mevcut özellikteki herhangi bir değişikliğe düzgün çalıştığından emin olmak için iyidir.Değişikliğin etkisini doğrulamak için önceden yürütülen test senaryoları yeniden yürütülür.

Q3-A/B test nedir ? Kısaca açıklayınız.

A3-A/B testing kullanıcı deneyimini ortaya çıkaran bir metodolojidir.Bu metodoloji ile 2 değişken üzerinden kullanıcıların etkileşimleri ölçümlenmeye çalışılır.2 farklı değişkeni,diğer etkenleri aynı bırakarak sistematik ve teorik analizlerle ölçer

Q4-Black box / white box test kavramlarını açıklayınız.

A4-White box testleri geliştirilen yazılımın kod yapısı bilinerek gerçekleştirilen test tasarım tekniğidir.Yazılımın iç yapısı ve iş akışlarıyla ilgilenir.Birim,tümleştirme ve sistem test seviyelerinde gerçekleştirilebilir.

Black box testleri ise sistemin functionalitysi ile ilgilenir.

Q5-Mutation test nedir ? Kısaca açıklayınız.

A5-Mutation test,kaynak koddaki belirli ifadeleri değiştirdiğimiz ve test entegrasyonlarının hataları bulabildiğini kontrol ettiğimiz bir yazılım testidir.Temelde unit test için kullanılan bir White box test türüdür.Mutated programdaki değişiklikler son derece küçük tutulur,böylece programın genel amacı etkilenmez.Mutasyon testinin amacı,mutant kodunu kaldıracak kadar sağlam olması gereken test vakalarının kalitesini değerlendirmektir.Bu yöntem, programda bir hata oluşturmayı içerdiği için hata tabanlı test stratejisi olarak da adlandırılır.

Q6-Behavior Driven Development (BDD) nedir, neyi amaçlamaktadır ?

A6-Yazılım süreçlerinin daha test odaklı gitmesini sağlayan bir yaklaşımdır.Testler,esas olarak sistem davranışına dayanmaktadır.Yazılım geliştirmeye başlamadan önce test senaryolarının yazılması desteklenmektedir.Bazı avantajları vardır.

-Kodun kalitesini artırarak temelde bakım maliyetini düşürür ve proje riskleri en aza indirilir.

-Developerlar işleyişi daha iyi öngörebildikleri için yazdıkları koda daha fazla güvenirler.

Q7-Agile test quadrant nedir ? Quadrant’ların kapsamını kısaca açıklayınız.

A7-Agile test quadrantları tüm agile test methodolojilerini dört temel çeyreğe ayıran bir araç olarak bilinir.Agile test quadrantları,tüm takımın anında iletişim kurmasına ve yüksek kaliteli bir ürün sunmasına yardımcı olur.Agile test quadrantların yardımıyla tüm test süreci anlaşılması çok kolay bir dilde açıklanabilir ve tüm takım ürün üzerinde etkili bir şekilde çalışabilir.

Quadrant1 : Takımı destekleyen teknolojiyle ilgili testler

Quadrant2 : Takımı destekleyen işe yönelik testler

Quadrant 3 : Ürünü eleştiren, işe yönelik testler

Quadrant 4 : Ürünü eleştiren teknolojiyle ilgili testlerdir.