SDLC= SOFTWARE DEVELOPMENT LIFE CYCLE (YAZILIM GELISTIRME YASAM DONGUSU)

SDLC Genel Bakış : SDLC, Yazılım Geliştirme Yaşam Döngüsü, yazılım endüstrisi tarafından yüksek kaliteli yazılım tasarlamak, geliştirmek ve test etmek için kullanılan bir süreçtir. SDLC, müşteri beklentilerini karşılayan veya beklentilerinde ötesinde bir ürün ortaya koyan, öngörülen zaman içerisinde tamamlanan ve maliyeti (bütçe) doğru bir şekilde hesaplanan yüksek kaliteli yazılım üretmeyi amaçlar.

SDLC yazılım endüstrisi tarafından yuksek kaliteli yazilim tasarlamak, gelistirmek ve test etmek icin kullanilan bir surectir.

Standart Yazılım Geliştirme yaşam döngüsü aşağıdakilerden oluşur:

1: Planning and Requirement Analysis (Planlama ve İhtiyaç Analizi) (Project manager ve .Business Analyst tarafından planlama, ihtiyac ve Rısk analizi yapılır bu asamada)

- İhtiyaç analizi SDLC'nin en önemli ve temel aşamasıdır.
- Müşteriden gelen fikirler de göz önünde bulundurularak ekibin kıdemli üyeleri tarafından gerçekleştirilir.
- Bu bilgiler daha sonra temel proje yaklaşımını planlamak için kullanılır.
- Kalite güvence gerekliliklerinin planlanması ve projeyle ilişkili risklerin belirlenmesi de planlama aşamasında yapılır.
- Minimum risklerle projeyi başarıyla uygulamak için izlenebilecek teknik yaklaşımlar planlanır.

2: Defining Requirements (Gereksininleri tanimlama ve dokumantasyon ile yol harıtası belirlenir)

- İhtiyaç analizi yapıldıktan sonraki adım, ürün gereksinimlerini açıkça tanımlamak ve belgelendirmektir
- Müşteriden / işletmeciden onay alınır.
- Bu, proje yaşam döngüsü boyunca tasarlanacak ve geliştirilecek tüm ürün gereksinimlerini içeren 'BRD' (Business Requirement Document) – İş Gereksinimi Belgesi ile yapılır.

3: Designing the product architecture (Ürün dizaynını tasarlama)

- BRD (Business Requirement Document) Dizaynırların geliştirilecek ürün için en iyi dizaynla ortaya çıkacakları referanstır.
- BRD'de belirtilen gereksinimlere dayanarak, ürün mimarisi için genellikle birden fazla tasarım yaklaşımı önerilmektedir ve bir DDS (Design Document Specification) - Tasarım Belgesi Spesifikasyonu'nda belgelenmektedir.

4: Building or Developing the Product (Ürünü oluşturma veya geliştirme)(DEVELOPER)

- SDLC'nin bu aşamasında gerçek gelişme başlar ve ürün inşa edilir.
- Yazılımcılar (Developers), kuruluşları tarafından tanımlanan kodlama yönergelerine uymak zorundadır.
- Kodlama için C ++, Java, dot Net vs. gibi farklı üst düzey programlama dilleri kullanılır.

5: Testing the Product (Ürünü test etme)(TESTER)

- Bu aşama, ürün BRD'de tanımlanan kalite standartlarına ulaşıncaya kadar,ürün kusurlarının rapor edildiği, izlendiği, sabitlendiği(fixlendiği) ve tekrar test edildiği ürünün sadece test aşamasını ifade eder.
- Ürün iş beklentilerini karşılamalıdır(requirement specifications)
- STLC => Software Testing Life Cycle

6: Deployment in the Market and Maintenance (Ürünü pazarlama ve Bakım)DEVOPS

- Ürün test edildikten ve onaylandıktan sonra (hazır olduğunda), resmi olarak uygun görülen şekilde release edilir(piyasaya sürülür).
- Ürün piyasaya sunulduktan sonra mevcut müşteri tabanı için bakımı yapılır.

SDLC'nin avantajları ve dezavantajları nelerdir?

Avantajlari: tahmin edilebilirligi, maliyet ve zaman planlamasi her asamasinin belirlenmesi ve riski en aza indirmesi

Dezavantajları_Uzmanlik gerektiren bir sistem Maliyeti yuksek ve tecrube isteyen bir sistemdir. Buyuk komplex sistemler icin kullanilmalidir. Detaylidir zaman alicidir

SDLC TEAM

1.Project manager (PM)=tum proje risklerini ve komunikasyonu yonetir. Proje yöneticisi, takımdaki herkesin rolünü bilmesini ve yerine getirmesini ve bu rollerin gerçekleştirileceği inancına göre hareket etmesini sağlamaktan sorumludur.

2.Business Analyst (BA)=bir kuruluşu analiz eden ve süreçlerini ve sistemlerini tasarlayan, iş modelini değerlendiren kişidir ve teknolojiyle entegrasyonunu sağlar. Dokumantasyonulari olusturur ve problemlere teknik cozumler saglar

3.Developer (Dev)= Tekniği yüksek olur ve yazılımı yaparak application oluşturur. Daha iyi anlayacagımız manada kaliteli guvenlik goz onunde bulundurulacak sekilde code yazan kimselerdir

4.Quality Analyst (QA)= Kalite kontrolü sağlayacak şekilde Software Development Life Cycle ın bütün prensiplerini uygulamakla yükümlüdür. Uygulamanin tum arti ve eksilerini analiz eder ve gereken ozellikleri test eder

Scrum Takımının Rolleri

- Scrum Ekibi = Product Owner, Scrum Master ve Development team'den oluşur.
- Scrum Takımları kendi kendini organize eder ve Cross-Functional (Çok fonksiyonludur).
- Kendi kendini düzenleyen ekipler, ekip dışındaki kişiler tarafından yönlendirilmek yerine işlerini en iyi şekilde nasıl gerçekleştireceklerini seçerler.
- Cross-functional ekipler, ekibin bir parçası olmayan ve başkalarına bağlı kalmadan işi tamamlamak için gerekli tüm yetkinliklere sahiptir.
- Scrum'daki ekip modeli esnekliği, yaratıcılığı ve üretkenliği optimize etmek için tasarlanmıştır.
- Product Owner (Stakeholder olabilir)
- Scrum master
- Development team (Developers and Testers)

PROJECT MANAGER (PM)

- Project manager: Proje yöneticisi, takımdaki herkesin rolünü bilmesini ve yerine getirmesini ve bu rollerin gerçekleştirileceği inancına göre hareket etmesini sağlamaktan sorumludur.
- Proje planının geliştirilmesi
- Proje sahipleri (Stakeholder) ile yakından ilişki kurar
- Takım içerisindeki iletişimi sağlar
- Proje riskini yönetir
- Proje çizelgesini hazırlar
- Proje bütçesini yönetir
- Projede çıkabilecek karışıklıkları (conflicts) önler (Kriz yönetiminden sorumludur)
- Proje dağılımını yönetir

BUSINESS ANALYST (BA)

- Business Analyst: bir kuruluşu (gerçek veya varsayımsal) analiz eden ve süreçlerini ve sistemlerini tasarlayan, iş modelini değerlendiren kişidir Teknolojiyle entegrasyonunu sağlar.

Sorumluluklar:

- Business sorunları ve teknoloji çözümleri arasında bir köprü vazivesi görür
- Requirements yönetimini ve iletişimini sağlar.
- Alınan kararların anlaşılır bir dile dökülmesini sağlar.
- Business Requirement Document (BRD) oluşturur (alınan tum kararların ve gereksinimlerin dokümü)
- Functional Requirement Document (FRD) oluşturur (yazılımı yapılacak olan bütün maddelerin dokümü)
- Bir takım functional requirement toplandıktan sonra use cases oluşturur
- Akış şemasını oluşturur.(flowchart)

DEVELOPER (DEV)

- • Developer: Tekniği yüksek olur ve yazılımı yaparak application oluşturur.
- Daha iyi anlayacagımız manada code yazan kimsedir.

Sorumluluklar:

- Kendilerine aktarılan software requirement dokümanını toparlar ve gereken application ve programın oluşumunu sağlar
- Beklentileri ve gereksinimleri (costumer requirement) karşılayacak yüksek kalitede code yazarlar.
- Software dokümanını oluşturur ve önceki dökumanları günceller.

QUALITY ANALYST (QA)

- Kalite kontrolü sağlayacak şekilde Software Development Life Cycle ın bütün prensiplerini uygulamakla yükümlüdür.
- Her hangi bir organizasyonun ürünlerini ve hizmetini beklenen kalite standartlarını karşılayacak şekilde oluşturulmasını sağlar.
- Oluşturlulan application ın istenilen plan çerçevesinde yapılmasını sağlar.
- Oluşturlulan application ın end-user (kullanıcılar) ların ve business temsilcilerinin beklentilerini karşılayacak seviyede olmasını sağlarlar.
- Application daki hatalar Quality Analyst tarafından bulunmalıdır ki Developerlar bulunan hataların üzerinde calışıp sorun teşkil etmeyecek ürün ortaya koyabilsinler (minimum seviyede bug).
- Testing yapılmasının amacı her hangi bir application da oluşabilecek hataların ortaya cıkarılmasıdır.
-

Roller ve sorumluluklar:

- Işletmecinin requirement larını analiz etmek
- Test edilen application ı iyi anlamak
- Test plan ve test Strategy oluşturulması
- Test case ve test scenario hazırlanması
- Test data (verilerinin) hazirlanmasi (test case)
- Test case lerin execute edilmesi
- Defect tracking (Hataların takibi)
- Yeniden test etmek
- Oluşan hatalarin bildirimi (Developer için)
- Raporların hazırlanması
- Ekip içinde inceleme toplantıları yapmak

SÖZLÜK

- Sprint => Bir takım hedeflerin tamamlanması için 2-4 hafta süre.
- Product Backlog (Ürün İş Listesi) => tüm proje boyunca yapılacak öğelerin (functioanlity) bir listesidir.
- Sprint Backlog (Sprint İş Listesi) => bir sprint sırasında yapılacak öğelerin (functionality) bir listesidir.
- Epic => Application da oluşturulacak büyük işlevde gereksinimler(functionality)(Backlog items)
- User story (Kullanıcı hikayesi)=> Application da oluşturulacak kucuk işlevde gereksinimler (functionality) (backlog items).
- Burndown chart => bir sprint'te kalan işi gösteren bir grafiktir.
- Product Owner (Ürün Sahibi)=> Product Backlog öğelerini yönetmekten sorumludur. (Stakeholder olabilir)
- Development team => Developers and Testers
- Scrum master => Team adına yapılan tüm organizasyonların yönetimini takip eder, Product Owner ve Development team arasında köprü vazifesi görür ama team in yöneteminden sorumlu değildir. Oluşabilecek engelleri ortadan kaldırır.
- Developer => uygulamayı oluşturmak için kod yazan kimseler .
- Functional tester: uygulamayı manuel olarak test eden kimseler.
- Automation Tester(Cross Functional tester) => uygulamayı test etmek için kod yazan kimseler.
- Test case => adım adım neyi test edeceğinizi açıklayan senaryolarınızdır
- Test plan =>Test kapsamı, yaklaşım kaynakları ve amaçlanan test faaliyetlerinin zamanlamasını tanımlayan giriş ve çıkış kriterleri.
- Test execution => Scriptler hazır olduğunda, çalıştırır veya manuel olarak testinizi gercekleştirirsiniz.
- Test data => functionality (leri) test etmek için kullanmanız gereken veriler
- Acceptance criteria =>functionalityi test etmenin ayrıntılı tanımı
- Expected result => yapılan testin beklenen sonucu
- Actual result => testi yaptıktan sonra ortaya cıkan netice.
- Environments (test yapilan ortamlar (URL)=> Dev,Test,Stage,Prod

SDLC TOPLANTILARI

Not) Grooming meeting→user storyler degerlendirilir ve tum sorular PO ya sorularak anlasilir hale getirilir. (PO aktif rol oynar) Sprint baslamadan once Grooming meeting toplantisi yapilir.

1) The Sprint_ Sprint kendi içinde, ön görülen zaman diliminde tüm işleri ve diğer tümolayları içeren bir etkinliktir.

2) Spring planning meeting→ Puanlama yapilir, sorumluluklar paylasilir.

- Bu, her Sprint'i başlatan etkinliktir ve Product Owner ve Development ekibinin Sprint'e hangi Ürün İş Listesi Öğelerinin (PBI'ler) dahil edileceğini tartıştığı yerdir.
- Product Owner Sprint'e potansiyel olarak dahil edilmek üzere her PBI'ya öncelik verme hakkına sahip olsa da, Geliştirme ekibi yanıt vermeye, sorunları gündeme getirmeye ve gerektiğinde geri itmeye teşvik edilir.
- Development team daha sonra hız, kaynaklar ve mevcut zaman ve kaynakları etkileyebilecek faktörler hakkında bilgi sahibi olduklarında Sprint'te kaç PBI sunabileceklerini tahmin eder.
- Sprint Planlama Toplantısının sonucu, herkesin kabul ettiği gerçekçi ve ulaşılabilir bir Sprint Hedefi ve Sprint İş Listesi elde etmektir.

3) Daily stand-up & (Daily scrum) (Günlük Scrum)

- Scrum zamanınızı ve kaynaklarınızı verimli bir şekilde kullanmayı amaçlar.
- Günlük Scrum 15 dakika kadar sürer. Ayağa kalkmak zorunlu değildir. Ancak, birçok takım bu toplantıyı kısa ve öz tutmak için yararlı bir teknik bulmaktadır.
- Günlük Scrum, Geliştirme Ekibinin check-in yapması, Sprint Hedefi'ne ulaşma yolundaki ilerlemeyi değerlendirmesi ve önümüzdeki 24 saat boyunca faaliyetlerini gözden geçirmesi ve planlaması için bir fırsattır.
- Genelde dün neler yaptık? Bugün ne üzerinde çalışacağız? Her hangi bir sıkıntı (impediment veya Blocker) ile karşılaştık mı?

4) Demo→ Sprint review (Sprint İncelemesi) Business yetkililerine demo sunum yapilir (QA Tester aktif rol alir)

- Sprint Review genellikle Sprint'in son gününde yapılır ve Stakeholder'lara (müşteriler, yönetim ve ilgili ve ilgilenilen diğer herkes) "done" (tamamlanmış) yapılan işi gösterme fırsatı verir.
- Sprint sırasında üretilen çalışma özelliklerini göstermenin yanı sıra, gelecekteki sprint'ler için çalışmayı yönlendirmeye yardımcı olabilecek Ürün İş Listesi'ni ekleyebileceğiniz yararlı geri bildirimler de alıyorsunuz.

5) Sprint Retro (Sprint Retrospective) (Geriye dönük Sprint Değerlendirmesi)→ Neler guzel gecti? Neler daha iyi olabilir? Bir sonraki sprinte neleri degistirmeyi planliyoruz? Herhangi bir impediment var mi?

- Sprint Retrospective Sprint'teki son toplantıdır. Scrum ekibi gelecekteki Sprint'ler için nelerin geliştirilebileceğini ve nasıl yapmaları gerektiğini gözden geçirir.
- Ne tür engellerle (impediment) karşılaştılar ve hangi fikirlerin ve güncellemelerin daha fazla gelişim sağlamalarına katkıda bulunduğunu değerlendirirler.

Scrum Artifacts (Scrum Eserleri)

- Product Backlog (Ürün İş Listesi) => tüm proje boyunca yapılacak öğelerin (functioanlity) bir listesidir.
- Sprint Backlog (Sprint İş Listesi) => bir sprint sırasında yapılacak öğelerin (functionality) bir listesidir.

SDLC nın en cok kullanılan 2 modelı MODELLERI WATERFALL VE AGILE

Waterfall (genelde devlet projelerinde kullanılır) Doğrusal-sıralı model olarak da adlandırılır. • Kullanımı ve anlaması çok kolaydır. • Her aşamanın bir sonraki aşama başlamadan tamamlanması gerekir. • Fazlandırmalar yapılır ve bu fazlardan birinin sonucu diğerinin girdisi olarak kabul edilir. Kucuk projelerde daha kulanislidir. Yonetimi kolaydir. Musteri endeksli degildir ve geri donusumler zor karsilanir. Her asama bitmeden bir digeri baslamaz. Test asamasi en son yapilir.

- Waterfall yöntemi, yazılım geliştirmede kullanılan erken SDLC yaklaşımıdır.
- Waterfall da her aşama bir önceki aşama tamamen bittikten sonra başlıyor.
- Waterfall modelinde fazlar çakışmaz.
- Tüm bu aşamalar, ilerlemenin aşamalar boyunca sürekli olarak aşağı doğru
- (şelale gibi) aktığı görülen biçimde birbirine kademelendirilir.

Aşamaları

1) Gereksinimlerin oluşturulması
2) Tahmini değerlendirme ve plan
3) Dizayn oluşumu
4) Coding yapılır
5) Piyasa sunum

Waterfall Avantajları • Anlamak ve kullanmak kolaydır. • Modelin sağlamlığı nedeniyle kolay yönetilir. Her aşamada belirli çıktılar ve inceleme süreci vardır. • Aşamalar ardı ardına işlenir ve tamamlanır. • Gereksinimlerin iyi anlaşılabildiği küçük projelerde iyi uygulanabilir. • Aşamalar açıkça tanımlanır. • İşleri sıralamak kolaydır. • Süreçler ve sonuçlar kolay iyi dokümante edilir.

Waterfall - Dezavantajları • Son aşamaya kadar uygulama çalışır durumda olmayabilir. • Sonuçlar açısından yüksek risk ve belirsizlikler vardır. • Karmaşık ve object-oriented projeler için uygun değildir. • Uzun ve devam eden projeler için zayıf bir modeldir. • İhtiyaçların değişme riskinin yüksek olduğu projeler için uygun değildir. Bundan dolayı bu modelde risk ve belirsizlik yüksektir. • Aşamaların ilerlemesini ölçmek zordur. • Herşey önceden belirlendiği için değişen gereksinimleri karşılayamaz.

Agile (Cevik) projeleri yönetmek ve tamamlamak için yaygın olarak kullanılan bir geliştirme yöntemidir.

Dinamiktir Musteri memnuniyetini esas alir. Yenilik ve degisimlere aciktir. Aktif toplantilar halinde surekli gelisim gosterir.

Agile Model, süreç uyumluluğu ve müşteri memnuniyetine odaklanarak hızlı bir şekilde sürekli çalışan uygulamalar oluşturmayı hedefler • Küçük iterasyonlarla üretim yaparak ürünü ortaya çıkarır • Her iterasyon 1-3 hafta arasında sürer • Her iterasyonda farklı konularda eşzamanlı olarak çalışan yetenekli takımlar ile gerçekleştirilir – Planlama – Gereksinim analizi – Tasarım – Geliştirme – Unit Test – Kabul Testi • İterasyonun sonunda çalışan uygulama müşteriye gösterilir

Agile musteri endekslidir. Agile gelisim ve yeniliklere aciktir.

1. Agile - Avantaj / Dezavantaj • Agile son dönemde yazılım dünyasında büyük oranda kabul görmüş bir yöntemdir. Bununla birlikte bu yöntem her ürün için uygun olmayabilir. – Avantajlar • Çok gerçekçi bir yaklaşım sağlar • Ekip çalışmasını ve birbirini eğiten elemanları destekler • İşlevler hızlı geliştirilir ve gösterilir • Az kaynak ihtiyacı vardır • Sabit ve değişken gereksinimlere uygundur • İhtiyaçların bir parçasını karşılayan ve sürekli çalışan uygulamalar sunar • Planlanmış zamanlarda ürün çıkartmayı sağlar • Az ya da hiç planlama gerekli olmayabilir • Kolay yönetilir • Geliştiricilere esneklik kazandırır

2. SDLC – Agile - Avantaj / Dezavantaj • Dezavantajlar – Müşteri etikileşimine bağlı olarak müşteri açık değilse ekip yanlış yönde yönlenir – Minimum dökümantasyon olduğundan yüksek bireysel bağımlılık vardır – Ürün gereksinimleri sürekli değişebileceğinden maliyetler peşinen tahmin edilemez. – Müşterilerden sürekli geri bildirim almak sizi alaşağı edebilir ya da bazı müşterilerin zamanı ve hiç ilgisi olmayabilir. – Müşteriler geri bildirim vermek istediklerinde akıllarına gelebilecek yeni istekler ek fazları ortaya çıkarır. Bu durum proje maliyetlerini ve süresini uzatabilir. – Müşteri ile iletişimi zor olan büyük Kurumsal yapılarda uygulamak zor olacaktır.

Jira proje yonetim aracidir_ Bug tracking tool

Software Testing (Yazılım testi) hiyararşisi

Functional Test

Whitebox(beyaz kutu testi)(kodlar üzerinden yapılan test) teknik bilgi gerektirir.
Developer'lar tarafından yapılır.Kodların iç yapisini ele alır.

- Dahili güvenlik açıkları
- Kodlama işlemlerinde bozuk veya kötü yapılandırılmış yollar
- Kod üzerinden belirli girdilerin akışı
- Beklenen çıktı
- Koşullu döngülerin işlevselliği (conditional loops)
- Her ifadenin, nesnenin ve fonksiyonun ayrı ayrı test edilmesi

(**unit test** whitebox'a girer) Unit Test (Birim testi) dinamik testin ilk düzeyidir ve önce Developerlarin ve ardından test mühendislerinin sorumluluğundadır.Birim testi, functionality hazır olunca bireysel olarak test edilir (Birbirinden ayrı bir şekilde)

Blackbox(karakutu testi) (kodlar gorunmez urun uzerinden test yapilir).illa teknik bir alt yapi gerektirmez. Kodlar uzerinden değil ürün üzerinde test yapilir

Blackbox Testleri(karakutu testleri) (tester'lar yapar)

Smoke test (tester'lar yapar, devepoler'larda bilir)
Unit Test, integration Testi, System/End To End Test ve UAT içerisinde kritik olan en önemli functionality'lerin düzenli olarak çalışıp çalışmadığını gözden geçirmek için yapılır.
(sabah ilk iş olarak smoke test yapılır)

integration (Entegrasyon)Testi

- (kompanentleri birarada çalıştırma testi)
- Birim testi tamamlandıktan sonra başlar
- Amaç; Uygulamanın farklı bileşenlerinin müşteri gereksinimlerine göre çalışmasını sağlamaktır
- Gerçek ve beklenen sonuçlar aynı olduğunda veya farklılıklar istemci girdisine göre açıklanabilir / kabul edilebilir olduğunda, entegrasyon testinin tamamlandığı kabul edilir.

System/ End To End Test
(baştan uca sistemin tamamının uyum içinde çalışması testi)

- Entegrasyon testinin tamamlanmasının ardından sistem testi başlatılır.

UAT (User Acceptance Test)
(kullanıcı kabul testi)

- Son kullanıcının veya müşterinin spesifikasyonlarına veya son kullanıcılar/müşteriler tarafından sınırlı bir süre boyunca kullanıma dayalı nihai test.
- Son Asama olarak her sey gözden gecirilir.

Regression test (Gerileme Testi) (testerler yapar)
Unit Test, integration Testi, System/End To End Test ve UAT içerisinde yapılan değişikliklerin yan etkilerinin izlenmesi ve mevcut olan hatalarında başka hatalar oluşturup-oluşturmadığının kontrol edilmesi için yapılan testlerdir. Regresyon Testi Gereksinimi; Yazılıma yeni özellik eklendiginde, Defect fix edildiginde, Performans sorunu düzeltmesi halinde, her splintten sonra veya 3 aylık dönemlerde yapilir
Minor(küçük) regression=> tüm functionalitylerin bir sprint sonunda test edilmesi
Major regression=> 3 aylik bir donemde tum reguirement (functinality)larin test edilmesi

ReTesting= hata anında yapılır. Hataların fix edilip edilmediğini görmek için yapılır

Ad-hoc (monkey) herhangi bir test yaptığımızda buna monkey test denir

Sanity test= yeni oluşturulan kritik özellikler sanity teste girer

Non-Functional Testing

Performance testing; uygulamanın hızı test edilir

Stress testing ; toplu bir şekilde birçok taraftan yük ve hız testi (alışılmadık derecede ağır yükler altındayken sistem fonksiyonel testi, belirli eylemlerin veya girdilerin ağır tekrarı, büyük sayısal değerlerin girilmesi, bir veritabanı sistemine büyük karmaşık sorgular vs. aktarimi.)

Load Testing; (Sistemin yanıt süresinin hangi noktada düştüğünü veya başarısız olduğunu belirlemek için bir uygulamayı bir dizi yük altında test etmek gibi bir uygulamayı ağır yükler altında test etme.)

Security test Sistemin tüm potansiyel açık kapıları ve zayıflıkları araştırılır.

Test Teknikleri

Equivalence partitioning =kategorilere bölerek test etme (eşit bölümlere ayırma)

Boundary value analysis = min ve max değerler üzerinden test etme (sınır değer Analizi)

SOFTWARE TESTING LIFE CYCLE (STLC)

Gereksinim analizi (Requirement Analysis)

Test planı (Test Planing)

Test senaryosu geliştirme (Test Case Development)

Test ortamı kurulumu (Test Environment Setup)

Test uygulaması (Test Execution)

Test döngüsü kapanışı (Test Cycle Closure)

YAZILIM TESTİNE GİRİŞ

- Yazılımın güvenilirliğini kontrol etmek için yazılım testi gereklidir
- Yazılım testi, sistemin her türlü arızaya neden olabilecek herhangi bir hata içermemesini sağlar.
- Yazılım testi, ürünün müşterinin gereksinimine uygun olmasını sağlar
- Sonunda, yazılım, hepsi farklı bakış açılarına ve yaklaşımlara sahip bir Developer ekibi tarafından geliştirilir. En zeki insan bile hata yapma eğilimindedir. Sıfır hata ile yazılım oluşturmak mümkün değildir dolayisiyla Testing yaşam döngüsünü yazilim geliştirme döngüsüne muhakkak dahil edilir.

SOFTWARE TESTING OVERVIEW _ YAZILIM TESTİNE GENEL BAKIŞ

✓Yazılım testi olmadan yazılım kalitesine ulaşamazsınız.
✓Testerlar gerçek kodlamaya dahil olmasalar bile, kodun kalitesini artırmak için Developerlarla yakın çalışmalıdırlar.
✓En iyi sonuçlar için yazılım testi ve kodlamanın iç içe gitmesi önemlidir.
✓Test, yazılımın kalitesini değerlendirmeye yardımcı olur.
✓Tüm test faaliyetleri planlama gerektirir.
✓Test, yazılımın tam olarak gereksinimlerde belirtildiği gibi davranmasını ve uygulanmasini sağlar.
✓Ihtiyaca (requirement) uygun olmayan her şey bir hatadir(defect).
✓İyi test edilmiş yazılım kaliteli ve daha iyi geribildirim ve yorumlar anlamına gelir (iyi feedbackler).

What is 'Software Quality Assurance'? ('Yazılım Kalite Güvencesi' nedir?)

✓Yazılım KG, tüm yazılım geliştirme sürecini içerir - sürecin izlenmesi ve iyileştirilmesi, üzerinde anlaşılan süreçlerin, standartların ve prosedürlerin takip edilmesini ve sorunların bulunmasını ve ele alınmasını sağlar.
✓What is 'Software Testing'? ('Yazılım Test Etme' nedir?)
✓Yazılım testi, geliştirilen bilgisayar yazılımının doğruluğunu, bütünlüğünü ve kalitesini belirlemek için kullanılan bir süreçtir.
✓Ürünün son kullanıcılara sunulmadan önce düzeltilebilmesi için yazılımda hata bulmak amacıyla yapılan bir dizi etkinliği içerir.
✓Basit bir ifadeyle, yazılım testi, gerçek sonuçların beklenen sonuçlarla eşleşip eşleşmediğini kontrol etmek ve yazılım sisteminin hatasız olmasını sağlamak
✓için yapılan bir etkinliktir.

Requirements Analysis _ Gereksinimlerin analizi

✓Sadece user story de açıklanan yazılım gereksinimlerini gözden geçirirsiniz.

User Story:

✓Bir kullanıcıya değer sağlayacak işlevselliğin kısa açıklaması
✓İçerik:
✓Başlık
✓Açıklama
✓Acceptance Criteria (Kabul etme kriteri/kabul şartları)

User Story :

Başlık:

➢Kişisel Kimlik Numarasını Girin (PIN)

Description (Açıklama):

➢ATM kullanıcısı olarak, nakit çekebilmem için PIN kodumu girmek istiyorum

Acceptance Criteria:

➢PIN dört basamakli olmalıdır

➢PIN özel karakterlere (sembollere) izin vermemelidir

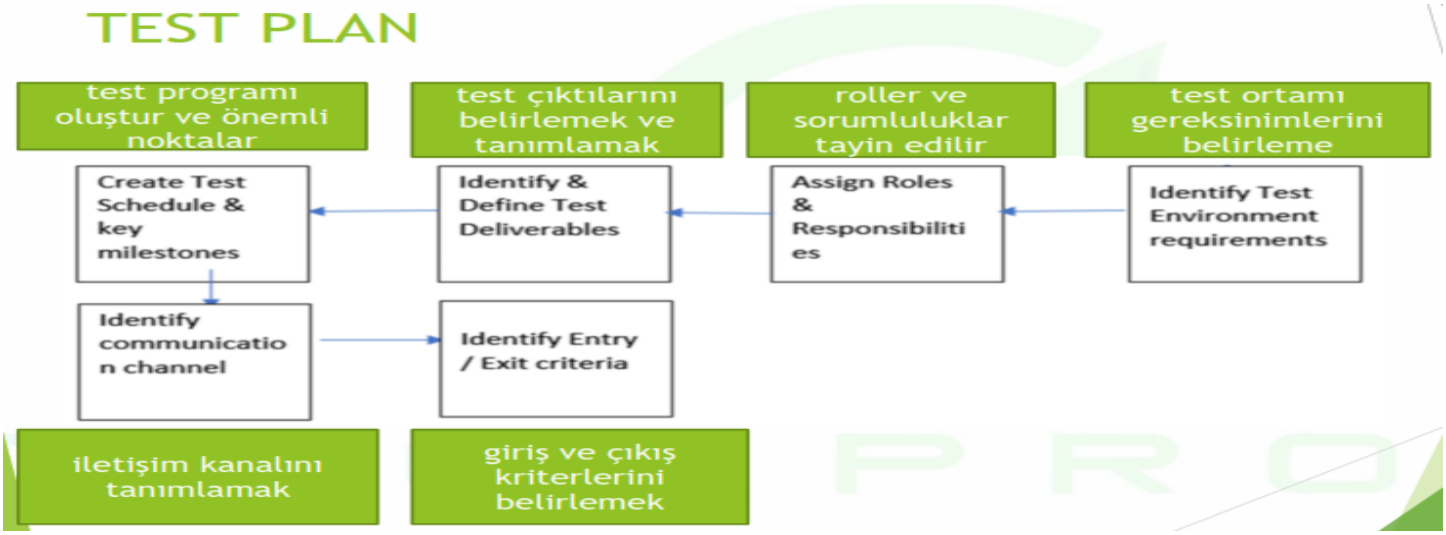
➢PIN'in 30 saniye içinde girilmesi gerekir, aksi takdirde işlem iptal edilir

TEST PLAN

➢Test Planı belgesi, Ürün Açıklaması, Yazılım Gereksinimi Spesifikasyonu (SRS) veya Kullanım Senaryosu Belgelerinden (Use Case Documents) türetilmiştir.

➢Amaçlanan test faaliyetlerinin kapsamını, yaklaşımını, kaynaklarını ve programını açıklayan bir belge.

➢Test Planı belgesi genellikle Test lead veya Test Manager tarafından hazırlanır ve belgenin odak noktası neyin test edileceğini, nasıl test edileceğini, ne zaman test edileceğini ve hangi testi kimin yapacağını açıklamaktır.



Test Case Development_Test Kılıfı Geliştirme/Oluşturma

✓Test verilerini kullanarak testlerinizi yazılımın ayrıntılı gereksinimlerine / tasarımına (bazen kendi hayal gücünüze dayanarak) dayalı olarak detaylandırır ve tasarlarsınız.

What is a test case in software testing? (Yazılım testinde test kılıfı nedir?)

✓• En basit biçimde, bir test kılıfı, bir test cihazının yazılımın gereksinimleri karşılayıp karşılamadığını ve işlevlerini doğru bir şekilde yerine getirip getirmediğini belirlediği bir dizi koşul veya değişkendir.

✓• Test kılıfı, bir tester' ın gerçekleştirdiği tek bir yürütülebilir testtir. Tek tek aşamalar(step) takip edilerek yapilir.

✓• Bir test kılıfını, bir şeyin davranması gerektiği gibi davrandığını doğrulamak için bir dizi adım adım talimat olarak düşünebilirsiniz.

✓test kılıfı ama başarılı ama basarısız neticelenir.

TEST CASE Test kılıfı

A test case often contains: (Bir test kılıfı genellikle şunları içerir)

- Test Case ID: Test kılıfı kimliği (numarası).

- Başlık: özet/ test kilifi amacı.
- Test steps(adımları): Testi gerçekleştirmek için adım adım prosedür.
- Test Data (verileri): Test yapılırken kullanılacak test verileri veya test verilerine bağlantılar.
- Expected result (Beklenen sonuç): yapilan testin beklenen sonucu
- The actual result (Gerçek sonuç): testi yaptıktan sonra ortaya cikan netice.
- Status (statü): Pass or Fail (başarılı veya basarısız)

Who Writes Test Cases? Test kılıflarını kim yazar

 Genellikle KG ekibinden biri test kılıflarını yazar.

 Bu Test kılıflarını hazırlamak için her ekip kendi standart şablonunu kullanır

Test Environment Setup Test Ortamı Kurulumu/ Oluşturulması

 Testinizi nerede ve hangi ortamda gerçekleştireceğinizi bilmelisiniz ve test verilerinize de sahip olmalısınız!

Test Environments Test Ortamı

 Dev (Development)

 Test (Test ortami)

 Stage (sahne)

 Prod (uretim)

Test Execution _ Test uygulaması

 Her şey hazır olduğunda, planlanan tüm functionality (ler) test edebilir ve yürütebilirsiniz.

 Manuel Testerlar testlerini manuel olarak yapar

 Automation (Cross-functional) Tester scriptlerini yazarak test ederler

Test Planning (Test Planlaması)

Neyin test edilmesi gerektiğine dair genel bir fikir topladıktan sonra, testler için 'plan yapılır'.

 Test case => adım adım neyi test edeceğinizi açıklayan senaryolarınızdır

 Test plan =>Test kapsamı, yaklaşım kaynakları ve amaçlanan test
faaliyetlerinin zamanlamasını tanımlayan giriş ve çıkış kriterleri.

 Test execution => Scriptler hazır olduğunda, çalıştırır veya manuel olarak
testinizi gercekleştirirsiniz.

 Test data => functionality (leri) test etmek için kullanmanız gereken veriler

 Acceptance criteria =>functionalityi test etmenin ayrıntılı tanımı

 Expected result => yapılan testin beklenen sonucu

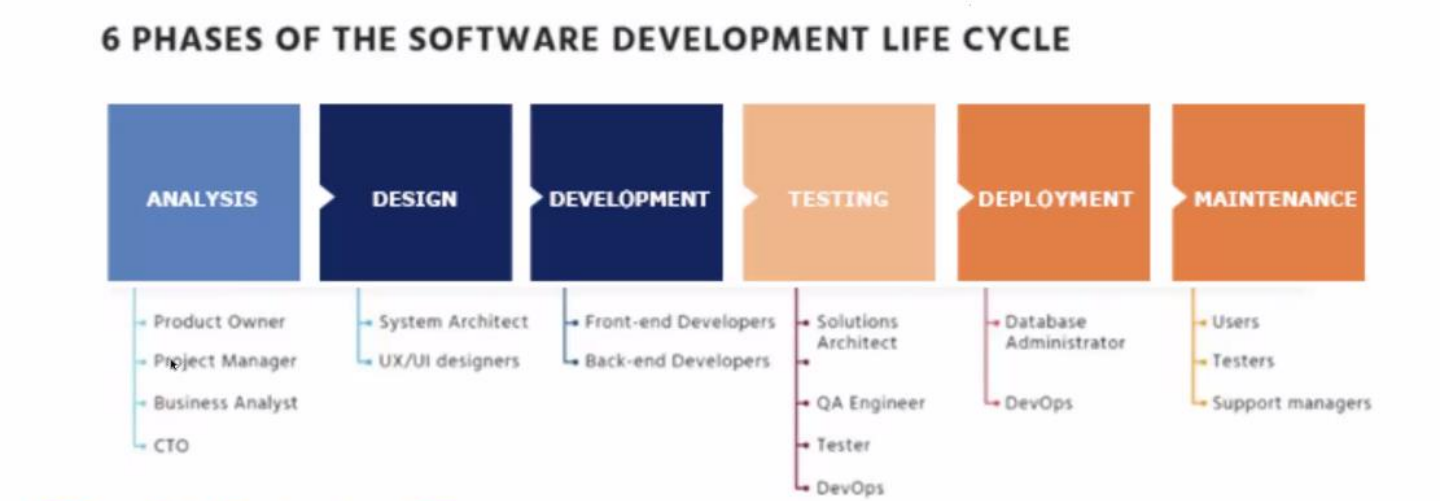
 Actual result => testi yaptıktan sonra ortaya cıkan netice.

 Environments (test yapilan ortamlar (URL)=> Dev,Test,Stage,Prod

Java +Selenium + SDLC => UI(User Interface) tester→frontEnd

Java +API + SDLC => API tester→BackEnd

Java + SQL + JDBC + SDLC→Database testing / Backend



CTO => Chief Technology Officer
UX => User Experience Designer (Kullanıcı Deneyim Tasarımcısı)
UI => User Interface Designer (Kullanıcı Arayüz Tasarımcısı

https://www.visualcapitalist.com/every-minute-internet-2020/
https://www.visualcapitalist.com/map-facebook-path-social-network-domination/