



ISTQB®

International Software
Testing Qualifications Board

ISTQB Foundation Level v4.0

YAZILIM TESTİ EL KİTABI TÜRKÇE KAYNAK

Recep ALICI

Project Manager / QA Engineer



alicirecepp@gmail.com



[coderalici](https://www.instagram.com/coderalici)

ÖNSÖZ

Merhaba, ben **Recep ALICI**.

Yazılım testine ilk başladığımda, test sürecinin aslında sadece "hata bulmak" olmadığını fark etmem biraz zaman aldı. Zamanla öğrendim ki iyi bir test süreci, bir yazılımın gerçekten güvenilir, stabil ve kullanıcı dostu olmasını sağlayan en kritik aşamalardan biri. Hataları önceden tespit etmek, sistemin dayanıklılığını artırmak ve en önemlisi, kullanıcıya kusursuz bir deneyim sunmak test mühendislerinin en büyük görevi.

Bu kitapçığı hazırlarken aklımdaki en büyük amaç, **yazılım testine merak duyan herkesin** bu alana sağlam bir giriş yapmasını sağlamak ve **ISTQB sınavına hazırlananlara** yardımcı olmaktı. İçeriği mümkün olduğunca anlaşılır ve sade tutmaya çalıştım. Teknik konuları sadece teorik olarak değil, **gerçek hayattan örneklerle ve pratik sorularla destekleyerek** anlatmayı hedefledim.

Eğer sen de ISTQB sınavına hazırlanıyorsan ya da yazılım testi hakkında daha fazla bilgi edinmek istiyorsan, doğru yerdesin! Umarım bu kitapçık, hem sınav sürecinde hem de test dünyasındaki yolculuğunda sana rehber olur.

Şimdiden başarılar dilerim!

Ve unutma:

İyi bir test mühendisi, sadece hataları bulan değil, kaliteyi garanti eden kişidir!

Recep ALICI

Project Manager / QA Engineer

alicirecepp@gmail.com

İçindekiler

1. ISTQB NEDİR? NEDEN ISTQB? (WHAT IS THE ISTQB / WHY ISTQB?)	14
ISTQB Nedir?	14
Neden ISTQB?	14
2. YAZILIM TESTİNİN TEMELLERİ (TEST BASICS)	15
Açıklama	15
Yazılım Testinin Temel Amaçları	15
Yazılım Test Türleri	15
Örnek	15
Pratik Sorular	16
Cevaplar	16
3. YAZILIM NEDİR? (WHAT IS SOFTWARE?)	17
Yazılımın Tanımı	17
1. Sistem Yazılımı	17
2. Uygulama Yazılımı	17
Yazılımın Önemi	17
Örnek Senaryo	17
Pratik Sorular	17
Cevaplar	18
4. YAZILIM TESTİ NEDİR? (WHAT IS SOFTWARE TESTING?)	19
Yazılım Testinin Tanımı	19
Yazılım Testinin Önemi	19
Yazılım Test Süreci	19
Örnek Senaryo	19
Pratik Sorular	20
Cevaplar	20
5. HATA VE HATA YÖNETİMİ (ERROR & BUG MANAGEMENT)	21
Hata Nedir?	21
Hata Yönetimi Süreci	21
Örnek Senaryo	21
Pratik Sorular	22
Cevaplar	22
6. YAZILIM TESTİNİN GEREKLİLİĞİ (WHY IS TESTING NECESSARY?)	23
Yazılım Testinin Önemi	23
Yazılım Testinin Sağladığı Faydalar	23
Örnek Senaryo	23

Pratik Sorular	23
Cevaplar	24
7. YAZILIM TESTİNİN ANA HEDEFLERİ (MAIN OBJECTIVES OF TESTING)	25
Yazılım Testinin Temel Hedefleri	25
Test Sürecinin Ana Hedefleri	25
Örnek Senaryo	25
Pratik Sorular	25
Cevaplar	26
8. YAZILIM TESTİNİN YEDİ PRENSİBİ (SEVEN PRINCIPLES OF TESTING)	27
Yazılım Testinin Temel Prensipleri	27
Örnek Senaryo	27
Pratik Sorular	28
Cevaplar	28
9. TEMEL TEST SÜREÇLERİ (BASIC TEST PROCESSES)	29
Test Süreçlerinin Önemi	29
Temel Test Süreçleri Aşamaları	29
Örnek Senaryo	29
Pratik Sorular	30
Cevaplar	30
10. YAZILIM TEST SÜRECİ – PLANLAMA VE KONTROL (SOFTWARE TESTING PROCESS - PLANNING AND CONTROL)	31
Test Planlama ve Kontrolün Önemi	31
Test Planlama Sürecinin Temel Adımları	31
Test Kontrol Sürecinin Temel Adımlar	31
Örnek Senaryo	32
Test planlama sürecinde:	32
Test kontrol sürecinde:	32
Pratik Sorular	32
Cevaplar	32
11. YAZILIM TEST SÜRECİ - ANALİZ VE TASARIM (SOFTWARE TESTING PROCESS - ANALYSIS AND DESIGN)	33
Analiz ve Tasarım Aşamasının Önemi	33
Analiz ve Tasarım Sürecinin Temel Adımları	33
Örnek Senaryo	33
Pratik Sorular	34
Cevaplar	34
12. YAZILIM TEST SÜRECİ – GERÇEKLEŞTİRME VE YÜRÜTME (SOFTWARE TESTING PROCESS - EXECUTION AND IMPLEMENTATION)	35

Gerçekleştirme ve Yürütme Aşamasının Önemi	35
Gerçekleştirme ve Yürütme Sürecinin Temel Adımları	35
Örnek Senaryo	36
Pratik Sorular	36
Cevaplar	36
13. YAZILIM TEST SÜRECİ - DEĞERLENDİRME (SOFTWARE TESTING PROCESS - EVALUATION).....	37
Test Değerlendirme Aşamasının Önemi.....	37
Test Değerlendirme Sürecinin Temel Adımları	37
Örnek Senaryo	37
Pratik Sorular	38
Cevaplar	38
14. YAZILIM TEST SÜRECİ – TESTİN SONLANDIRILMASI (SOFTWARE TESTING PROCESS – ENDING THE TEST).....	39
Testin Sonlandırılması Aşamasının Önemi	39
Testin Sonlandırılması Sürecinin Temel Adımları.....	39
Örnek Senaryo	39
Pratik Sorular	40
Cevaplar	40
15. YENİ TEST SÜREÇLERİ VE STANDARTLARI (NEW TEST PROCESS AND STANDARDS).....	41
Yeni Test Süreçlerinin Önemi.....	41
Güncel Test Süreçleri ve Standartları	41
Örnek Senaryo	41
Pratik Sorular	42
Cevaplar	42
16. TEST PSİKOLOJİSİ (TEST PSYCHOLOGY)	43
Test Psikolojisinin Önemi	43
Test Psikolojisinin Temel Unsurları	43
Örnek Senaryo	43
Pratik Sorular	44
Cevaplar	44
17. TEST FAALİYETLERİ VE GÖREVLERİ (TESTING ACTIVITIES AND TASKS).....	45
Test Sürecinin Önemi.....	45
Test Faaliyetleri ve Görevleri	45
Örnek Senaryo	45
Pratik Sorular	46
Cevaplar	46
18. YAZILIM TESTİ İŞ ÜRÜNLERİ (SOFTWARE TESTING BUSINESS PRODUCTS).....	47

Test İş Ürünlerinin Önemi	47
Yazılım Testi İş Ürünleri	47
Örnek Senaryo	47
Pratik Sorular	48
Cevaplar	48
19. YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ BOYUNCA TEST (TESTING THROUGHOUT THE SOFTWARE LIFECYCLE)	49
Testin Yazılım Geliştirme Sürecindeki Rolü	49
Yazılım Geliştirme Yaşam Döngüsündeki Test Aşamaları	49
Örnek Senaryo	50
Pratik Sorular	50
Cevaplar	50
20. YAZILIM TEST SEVİYELERİ (SOFTWARE TEST LEVELS)	51
Test Seviyelerinin Önemi	51
Yazılım Test Seviyeleri	51
Örnek Senaryo	52
Pratik Sorular	52
21. TEST TÜRLERİ (TEST TYPES)	53
Test Türlerinin Önemi	53
Temel Test Türleri	53
Örnek Senaryo	53
Pratik Sorular	54
Cevaplar	54
22. BAKIM TESTİ (MAINTENANCE TESTING)	55
Bakım Testinin Önemi	55
Bakım Testi Türleri	55
Örnek Senaryo	55
Pratik Sorular	56
Cevaplar	56
23. STATİK TESTLER (STATIC TESTS)	57
Statik Testlerin Önemi	57
Statik Test Türleri	57
Pratik Sorular	58
Cevaplar	58
24. İNCELEMELER (REVIEWS)	59
İncelemelerin Önemi	59
İnceleme Türleri	59
Pratik Sorular	60

Cevaplar	60
25. ROLLER VE SORUMLULUKLAR (ROLES AND RESPONSIBILITIES)	61
Yazılım Test Sürecindeki Roller ve Sorumlulukların Önemi	61
Temel Test Rollerini	61
Örnek Senaryo	61
Pratik Sorular	62
Cevaplar	62
26. STATİK KOD ANALİZİ (STATIC CODE ANALYSIS)	63
Statik Kod Analizinin Önemi	63
Statik Kod Analizinin Avantajları	63
Statik Kod Analiz Araçları	63
Örnek Senaryo	64
Pratik Sorular	64
Cevaplar	64
27. TEST TASARIM TEKNİKLERİ (TEST DESIGN TECHNIQUES)	65
Test Tasarım Tekniklerinin Önemi	65
Test Tasarım Teknikleri Türleri	65
1. Kara Kutu Test Teknikleri (Black Box Testing)	65
2. Beyaz Kutu Test Teknikleri (White Box Testing)	65
3. Deneyime Dayalı Test Teknikleri (Experience-Based Testing)	66
Örnek Senaryo	66
Pratik Sorular	67
Cevaplar	67
28. RİSK TEMELLİ TEST YAKLAŞIMLARI (RISK-BASED TESTING APPROACHES)	68
Risk Temelli Test Nedir?	68
Risk Değerlendirme Kriterleri	68
Risk Temelli Testin Avantajları	68
Risk Temelli Test Süreci	69
Örnek Senaryo	69
Pratik Sorular	70
Cevaplar	70
29. TEST YÖNETİMİ (TEST MANAGEMENT)	71
Test Yönetimi Nedir?	71
Test Yönetiminin Ana Bileşenleri	71
Test Yönetiminin Önemi	72
Örnek Senaryo	72
Pratik Sorular	73

Cevaplar	73
30. TEST PLANLAMA VE TAHMİNLEME (TEST PLANNING AND ESTIMATION).....	74
Test Planlama ve Tahminleme Nedir?	74
Test Planlamanın Ana Bileşenleri.....	74
Test Tahminleme Yöntemleri	75
Örnek Senaryo	75
Pratik Sorular	76
Cevaplar	76
31. TEST SÜRECİNİN TAKİBİ VE KONTROLÜ (TEST PROGRESS SURVEILLANCE AND CONTROL)	77
Test Sürecinin Takibi ve Kontrolü Nedir?	77
Test Takibi ve Kontrolünün Ana Bileşenleri.....	77
Test Takibi ve Kontrolünün Önemi	78
Örnek Senaryo	78
Pratik Sorular	79
Cevaplar	79
32. DENEYİME DAYALI TEST TASARIM TEKNİKLERİ (EXPERIENTIAL BASED TEST DESIGN TECHNIQUES).....	80
Deneyime Dayalı Test Tasarım Teknikleri Nedir?	80
Deneyime Dayalı Test Teknikleri Türleri.....	80
Deneyime Dayalı Testlerin Avantajları	80
Örnek Senaryo	81
Pratik Sorular	82
Cevaplar	82
33. YAPILANDIRMA YÖNETİMİ (CONFIGURATION MANAGEMENT)	83
Yapılandırma Yönetimi Nedir?	83
Yapılandırma Yönetiminin Ana Bileşenleri	83
Yapılandırma Yönetiminin Önemi	83
Örnek Senaryo	84
Pratik Sorular	85
Cevaplar	85
34. RİSK VE TEST (RISK AND TESTING)	86
Risk ve Test Nedir?	86
Risk Türleri	86
✓ Ürünle İlgili Riskler (Product Risks)	86
✓ Proje Riskleri (Project Risks).....	86
✓ İş Riskleri (Business Risks)	86
Risk Bazlı Testin Avantajları.....	86

Risk Değerlendirme Süreci	87
1. Risk Tanımlama (Risk Identification)	87
2. Risk Analizi (Risk Analysis)	87
3. Risk Azaltma (Risk Mitigation)	87
4. Sürekli İzleme (Continuous Monitoring)	87
Örnek Senaryo	87
Pratik Sorular	88
Cevaplar	88
35. OLAY/HATA YÖNETİMİ (INCIDENT/ERROR MANAGEMENT)	89
Olay/Hata Yönetimi Nedir?	89
Olay/Hata Yönetim Sürecinin Adımları	89
1. Olayın Tespiti (Incident Identification)	89
2. Olayın Kayıt Altına Alınması (Incident Logging)	89
3. Olayın Önceliklendirilmesi (Incident Prioritization)	89
4. Olayın Analiz Edilmesi (Incident Analysis)	89
5. Çözüm ve Düzeltme (Resolution & Fixing)	90
6. Doğrulama ve Yeniden Test (Verification & Re-testing)	90
7. Olayın Kapatılması (Incident Closure)	90
Olay/Hata Yönetiminin Önemi	90
★ Hataların Takip Edilebilirliğini Sağlar	90
★ Etkili Sorun Giderme Sağlar	90
★ Zaman ve Kaynak Yönetimini Geliştirir	90
Örnek Senaryo	90
★ Olay Tespiti:	90
★ Kayıt Altına Alma:	90
★ Önceliklendirme:	90
★ Analiz ve Düzeltme:	91
★ Doğrulama:	91
★ Olay Kapatma:	91
Bu süreç sayesinde:	91
Pratik Sorular	91
Cevaplar	91
36. TEST ARAÇ TÜRLERİ (TEST TOOL TYPES)	92
Test Araç Türleri Nedir?	92
Test Araç Türleri ve Kullanım Alanları	92
1. Test Yönetim Araçları (Test Management Tools)	92
2. Hata Takip ve Yönetim Araçları (Bug Tracking & Defect Management Tools)	92

3. Otomasyon Test Araçları (Automation Testing Tools)	92
4. Performans Test Araçları (Performance Testing Tools)	92
5. Güvenlik Test Araçları (Security Testing Tools)	92
6. Statik Analiz Araçları (Static Analysis Tools)	92
Test Araçlarının Avantajları	93
★ Zaman Tasarrufu Sağlar	93
★ Hata Takibini Kolaylaştırır	93
★ Daha Güvenilir Sonuçlar Elde Edilir	93
★ Performans ve Güvenlik Açısından Daha Güçlü Testler Sağlar	93
Örnek Senaryo	93
★ Test Yönetimi İçin:	93
★ Hata Takibi İçin:	93
★ Otomasyon İçin:	93
★ Performans İçin:	93
★ Güvenlik İçin:	93
Bu süreç sayesinde:	94
Pratik Sorular	94
Cevaplar	94
37. TEST ARAÇLARININ ETKİN KULLANIMI: POTANSİYEL AVANTAJLAR VE RİSKLER (EFFECTIVE USE OF TEST TOOLS: POTENTIAL BENEFITS AND RISKS) Test Araçlarının Etkin Kullanımı Nedir?	95
Test Araçlarının Potansiyel Avantajları	95
★ Zaman ve Maliyet Tasarrufu	95
★ Hata Tespitinde Doğruluk	95
★ Veri Toplama ve Raporlama	95
★ Tekrarlanabilirlik ve Tutarlılık	95
★ Sürekli Entegrasyon ve Dağıtım (CI/CD) Desteği	95
Test Araçlarının Potansiyel Riskleri	96
△ Yanlış Araç Seçimi	96
△ Yüksek Bakım Maliyeti	96
△ Yanlış Pozitif ve Yanlış Negatif Sonuçlar	96
△ Eğitim ve Yetkinlik Eksikliği	96
△ Bağımlılık	96
Örnek Senaryo	96
★ Avantaj	96
★ Risk	96

✦ Çözüm	96
Bu süreç sayesinde:	96
Test Araçlarının Etkin Kullanımı İçin En İyi Uygulamalar	97
Pratik Sorular	97
Cevaplar	97
38. ETİK KURALLAR (ETHICAL RULES)	98
Etik Kurallar Nedir?	98
Etik Kuralların Temel İlkeleri	98
1. Dürüstlük ve Tarafsızlık	98
2. Gizliliğe Saygı	98
3. Mesleki Yetkinlik ve Sürekli Gelişim	98
4. Yasalara ve Standartlara Uyum	98
5. Bağımsızlık	98
6. Yanıltıcı Bilgiden Kaçınma	99
Etik Kuralların Önemi	99
✦ Güvenilirliği Artırır	99
✦ Veri Güvenliğini Sağlar	99
✦ Profesyonellik Sağlar	99
Örnek Senaryo	99
✦ Gizliliğe Saygı Gösterme:	99
✦ Yetkili Birimlere Bildirme:	99
✦ Yanıltıcı Bilgiden Kaçınma:	99
Pratik Sorular	100
Cevaplar	100
39. AGİLE TEST SÜREÇLERİ (AGILE TESTING PROCESSES)	101
Agile Test Süreçleri Nedir?	101
Agile Testin Temel Özellikleri	101
1. Kısa Teslimat Süreçleri	101
2. Otomasyon ve Sürekli Entegrasyon (CI/CD)	101
3. Scrum ve Kanban ile Entegrasyon	101
4. Ekip İş Birliği	101
5. Adaptif Test Süreci	101
Örnek Senaryo	102
1. Sprint Bazlı Testler:	102
2. Sürekli Entegrasyon ile Test:	102
3. Kullanıcı Geri Bildirimine Dayalı Testler:	102
Bu süreç sayesinde:	102

Pratik Sorular	102
Cevaplar	102
40. RİSK TEMELLİ TEST YAKLAŞIMLARI (RISK-BASED TESTING APPROACHES)	103
Risk Temelli Test Nedir?	103
Risk Değerlendirme Kriterleri.....	103
1. Hata Olasılığı.....	103
2. İşletme Etkisi	103
3. Uyumluluk ve Entegrasyon Riskleri	103
4. Güvenlik ve Performans Gereksinimleri	103
Risk Temelli Testin Avantajları.....	103
✓ Önceliklendirme Sağlar	103
✓ Test Kapsamını Optimize Eder	103
✓ Kullanıcı Deneyimini Artırır	104
Risk Temelli Test Süreci	104
1. Riskleri Belirleme	104
2. Risklerin Değerlendirilmesi	104
3. Test Planı Hazırlama	104
4. Testlerin Yürütülmesi	104
5. Sürekli Gözden Geçirme	104
Örnek Senaryo	104
Pratik Sorular	105
Cevaplar	105
41. TEST OTOMASYONU (TEST AUTOMATION)	106
Test Otomasyonu Nedir?	106
Test Otomasyonunun Temel Avantajları	106
✦ Zaman Tasarrufu	106
✦ Tutarlılık ve Tekrarlanabilirlik	106
✦ Erken Hata Tespiti	106
✦ Kapsamlı Test Yapılmasını Sağlar	106
✦ Regresyon Testlerinin Otomatikleştirilmesi	106
Test Otomasyon Araçları ve Kullanım Alanları	106
1. Selenium	106
2. Appium	107
3. JMeter.....	107
4. TestComplete.....	107
5. Cypress	107
Ne Zaman Test Otomasyonu Kullanılmalıdır?	107

Örnek Senaryo	107
Bu süreç sayesinde:	107
Pratik Sorular	108
Cevaplar	108
42. YAPAY ZEKA VE MAKİNE ÖĞRENMESİ İLE TEST OTOMASYONU (AI AND MACHINE LEARNING IN TEST AUTOMATION)	109
Yapay Zeka Destekli Test Otomasyonu Nedir?	109
Amaç:.....	109
AI Destekli Test Araçları	109
AI'nin Test Senaryosu Oluşturmadaki Rolü.....	109
1. Öğrenme Yeteneği	109
2. Dinamik Test Senaryoları	109
3. Hata Tahmini.....	110
4. Sürekli Güncellenen Testler	110
Örnek Senaryo	110
Sonuç:	110
Pratik Sorular	110
Cevaplar	110
43. DEVOPS TEST SÜREÇLERİ (DEVOPS TESTING PROCESSES)	111
DevOps Test Süreçleri Nedir?	111
DevOps'un Amacı:	111
DevOps Testin Temel Özellikleri	111
1 Sürekli Test (Continuous Testing)	111
2 Ekipler Arasında İş Birliği	111
3 Shift-Left ve Shift-Right Yaklaşımları.....	111
DevOps ve QA (Kalite Güvencesi) İlişkisi	112
✓ Test Otomasyonu	112
✓ Geri Bildirim Döngüsü	112
✓ Güvenlik ve Uygulama Sağlığı	112
DevOps Sürecinde Kullanılan Test Otomasyon Araçları	112
Örnek Senaryo	112
✓ Sürekli Entegrasyon (CI) Kullanımı	112
✓ Sürekli Teslimat (CD) ile Canlı Ortama Geçiş	112
✓ Yük Testleri (JMeter Kullanımı)	112
✓ Güvenlik Testleri.....	113
Bu süreç sayesinde:	113
Pratik Sorular	113

Cevaplar	113
44. MODERN TEST ARAÇLARI VE TEKNİKLERİ (MODERN TESTING TOOLS AND TECHNIQUES)	114
Modern Test Araçları	114
Risk Bazlı Test Yönetimi ve Güncel Test Standartları (ISO/IEC 29119).....	114
Risk Bazlı Test Yönetimi Adımları:.....	114
ISO/IEC 29119 Test Standartları Nedir?	114
Örnek Senaryo	115
✦ Cypress Kullanımı:.....	115
✦ TestContainers Kullanımı:.....	115
✦ Risk Bazlı Test Yönetimi:.....	115
Sonuç:	115
Cevaplar	115
45. SÜREKLİ TEST (CONTINUOUS TESTING) VE SHIFT-LEFT TESTING YAKLAŞIMLARI.....	116
Sürekli Test (Continuous Testing) Nedir?.....	116
Sürekli Testin Temel Faydaları	116
✦ Hataların Erken Tespiti	116
✦ CI/CD Süreçleriyle Uyum	116
✦ Manuel Test Yükünü Azaltır	116
✦ Gerçek Kullanıcı Simülasyonu.....	116
✦ Daha Kaliteli ve Güvenilir Yazılım.....	116
Shift-Left Testing Nedir?	116
✦ Shift-Left Testing'in Avantajları	117
Örnek Senaryo:	117
E-Ticaret Uygulaması için Sürekli Test & Shift-Left Testing	117
✦ Sürekli Test:	117
✦ Shift-Left Testing:	117
Bu süreç sayesinde:	117
Pratik Sorular	117
Cevaplar	117

1. ISTQB NEDİR? NEDEN ISTQB?

(WHAT IS THE ISTQB / WHY ISTQB?)

ISTQB Nedir?

ISTQB (International Software Testing Qualifications Board), yazılım testi alanında uluslararası geçerliliği olan ve yazılım test mühendislerinin bilgi seviyelerini standartlaştıran bir sertifikasyon kuruluşudur. 2002 yılında kurulmuş olup, yazılım testi konusunda küresel anlamda ortak bir anlayış oluşturmayı amaçlar. ISTQB, farklı seviyelerde sertifikasyon sağlayarak test uzmanlarının mesleki yeterliliklerini artırmalarına yardımcı olur.

ISTQB'nin temel misyonu, yazılım testi süreçlerini daha yapılandırılmış, verimli ve hatasız hale getirmek için en iyi uygulamaları ve test metodolojilerini tanımlamaktır. Bu doğrultuda, ISTQB sertifikasyonları, yazılım testi konusundaki terminolojiyi ve süreçleri küresel düzeyde uyumlu hale getirmeye katkıda bulunur.

Neden ISTQB?

Günümüz yazılım dünyasında, kalite güvencesi ve yazılım testi, başarılı projelerin temel taşlarından biri haline gelmiştir. ISTQB sertifikasyonu, hem bireysel test mühendisleri hem de şirketler için çeşitli avantajlar sunar:

- **Uluslararası Geçerlilik:** ISTQB sertifikaları dünya genelinde kabul görmekte olup, yazılım test mühendislerinin küresel ölçekte iş bulma şansını artırır.
- **Standardizasyon:** Yazılım testi süreçlerinde ortak bir dil ve yapı oluşturarak, farklı ekipler ve şirketler arasında uyumluluğu artırır.
- **Kariyer Gelişimi:** ISTQB sertifikaları, yazılım test mühendislerinin kariyerlerinde ilerlemeleri için önemli bir basamak oluşturur.
- **Maliyet ve Zaman Tasarrufu:** Standart test süreçleri ve metodolojileri sayesinde, şirketler test süreçlerini daha verimli hale getirerek maliyetleri düşürebilir.
- **Sürekli Gelişim:** ISTQB, gelişen teknolojiye paralel olarak sertifikasyon içeriklerini güncelleyerek test uzmanlarının en yeni teknik ve araçları öğrenmesini sağlar.

ISTQB'nin sunduğu sertifikalar sayesinde test süreçleri daha sistematik hale gelir, yazılım kalitesi artar ve hataların erken tespit edilmesiyle yazılım geliştirme süreçleri daha verimli hale getirilir.

2. YAZILIM TESTİNİN TEMELLERİ (TEST BASICS)

Açıklama

Yazılım testi, bir yazılımın belirlenen gereksinimlere uygun olup olmadığını değerlendirmek, hataları tespit etmek ve kullanıcı beklentilerini karşılayıp karşılamadığını doğrulamak amacıyla gerçekleştirilen **sistematik bir süreçtir**. Test süreçleri, **yazılımın daha güvenilir hale gelmesine** ve **geliştirme sürecinde maliyetlerin düşürülmesine** yardımcı olur.

Yazılım Testinin Temel Amaçları

- Hataları Bulmak:** Yazılım içinde bulunan eksiklikleri ve hataları belirlemek.
- Kaliteyi Artırmak:** Yazılımın güvenilir, stabil ve hatasız olmasını sağlamak.
- Doğrulama ve Geçerleme (Verification & Validation):** Yazılımın hem teknik gereksinimlere hem de kullanıcı beklentilerine uygun olup olmadığını değerlendirmek.
- Güvenliği Sağlamak:** Kullanıcı bilgilerini ve sistem güvenliğini koruyacak testler gerçekleştirmek.
- Kullanıcı Deneyimini İyileştirmek:** Yazılımın kullanım kolaylığını ve performansını değerlendirmek.

Yazılım Test Türleri

- Fonksiyonel Testler:** Yazılımın belirlenen işlevleri doğru yerine getirip getirmediğini test eder.
- Fonksiyonel Olmayan Testler:** Performans, güvenlik, kullanılabilirlik ve uyumluluk gibi kriterleri değerlendirir.
- Yapısal Testler:** Yazılımın iç yapısını ve kod kalitesini analiz eder.
- Regresyon Testleri:** Yazılımda yapılan değişikliklerin mevcut sistem üzerinde olumsuz etkileri olup olmadığını test eder.

Örnek

Bir **çevrimiçi ödeme sistemi** geliştirdiğimizi düşünelim. Bu sistemin **güvenli** ve **hatasız** çalıştığını doğrulamak için aşağıdaki testleri uygulayabiliriz:

- Fonksiyonel Test:** Kullanıcı, doğru giriş bilgileriyle sisteme erişebilmelidir. Yanlış şifre girildiğinde sistem uygun bir hata mesajı göstermelidir.
- Performans Testi:** Aynı anda **1000'den fazla kullanıcının giriş yapması** veya **ödeme işlemi gerçekleştirmesi** durumunda sistemin yanıt süresi ölçülmelidir.
- Güvenlik Testi:** Kullanıcı hesap bilgilerinin **şifrelenip şifrelenmediği** ve **yetkisiz erişimlerin engellenip engellenmediği** test edilmelidir.
- Kullanılabilirlik Testi:** Kullanıcıların **ödeme sürecinde zorlanmadan işlem yapabildiği** doğrulanmalıdır.

Bu test süreçleri sayesinde sistemin **hatasız, güvenilir ve kullanıcı dostu** olması sağlanır.

Pratik Sorular

1. **Fonksiyonel ve fonksiyonel olmayan testler arasındaki fark nedir?**
2. **Regresyon testi neden önemlidir?**
3. **Bir yazılımın güvenliği nasıl test edilir?**

Cevaplar

1. **Fonksiyonel testler**, yazılımın belirlenen işlevlerini doğru yerine getirip getirmediğini test ederken; **fonksiyonel olmayan testler**, performans, güvenlik ve kullanılabilirlik gibi yazılımın kalite özelliklerini değerlendirir.
2. **Regresyon testi**, yazılımda yapılan değişikliklerin mevcut sistem üzerinde olumsuz etkileri olup olmadığını belirlemek için yapılır. Böylece yeni güncellemeler eski fonksiyonları bozmaz.
3. **Yazılımın güvenliği**, güvenlik testleriyle ölçülür. Bu testlerde kullanıcı verilerinin **şifrelenmesi**, yetkisiz erişimlerin engellenmesi ve olası saldırılara karşı sistemin dayanıklılığı test edilir.



3. YAZILIM NEDİR? (WHAT IS SOFTWARE?)

Yazılımın Tanımı

Yazılım, bilgisayar sistemlerinde belirli görevleri yerine getirmek için kullanılan programlar, komut dizileri ve ilgili verilerdir. Yazılım, **donanım** ile birlikte çalışarak sistemin işlevselliğini sağlar ve kullanıcıya belirli hizmetler sunar.

Yazılım, iki temel kategoriye ayrılır:

1. Sistem Yazılımı

- Bilgisayarın temel işlevlerini yöneten yazılımlar
- Örnekler: işletim sistemleri, sürücüler, sistem araçları

2. Uygulama Yazılımı

- Kullanıcıların belirli görevleri yerine getirmesi için geliştirilen yazılımlar
- Örnekler: ofis programları, oyunlar, e-ticaret uygulamaları

Yazılımın Önemi

- **Günlük hayatın** birçok alanında süreçleri hızlandırır ve otomatikleştirir.
- İş süreçlerini daha verimli hale getirerek **üretkenliği artırır**.
- **Eğitim, sağlık, finans, ulaşım** gibi kritik sektörlerde etkin çözümler sunar.

Örnek Senaryo

Bir hastane yönetim sistemi yazılımı ele alalım:

- **Sistem Yazılımı:** Hasta verilerini saklayan **veritabanı yönetim sistemi** ile entegre çalışır.
- **Uygulama Yazılımı:**
 - Hasta randevularını yönetir.
 - Doktorların hasta geçmişine erişmesini sağlar.
 - Faturalandırma işlemlerini yürütür.

Bu sayede **hastaneler** hasta kayıtlarını daha düzenli yönetebilir, **hastalar** randevularını kolayca alabilir ve **doktorlar** hasta bilgilerine hızlı erişim sağlayabilir.

Pratik Sorular

1. Sistem yazılımı ile uygulama yazılımı arasındaki temel fark nedir?
2. Yazılımın günlük hayatta en kritik olduđu üç sektör nedir?
3. Bir e-ticaret sisteminde sistem ve uygulama yazılımlarının rolü nedir?

Cevaplar

1. Sistem yazılımı, bilgisayarın temel işlevlerini yönetirken (işletim sistemleri, sürücüler), uygulama yazılımı kullanıcıların belirli görevleri yerine getirmesine yardımcı olur (ofis programları, oyunlar).
2. Yazılımın en kritik olduđu sektörler:
 - Sağlık (hastane yönetim sistemleri)
 - Finans (banka uygulamaları)
 - Ulaşım (navigasyon sistemleri)
3. Bir e-ticaret sisteminde:
 - Sistem yazılımı, sunucu işletim sistemi ve veritabanı yönetimini sağlar.
 - Uygulama yazılımı, kullanıcıların ürün araması, sipariş vermesi ve ödeme yapması gibi işlemleri yürütür.



4. YAZILIM TESTİ NEDİR? (WHAT IS SOFTWARE TESTING?)

Yazılım Testinin Tanımı

Yazılım testi, bir yazılımın belirlenen gereksinimlere uygun çalışıp çalışmadığını değerlendirmek, hataları tespit etmek ve yazılımın kalitesini artırmak amacıyla gerçekleştirilen **sistematik bir süreçtir**. Yazılım testi, **yazılım geliştirme yaşam döngüsünün** ayrılmaz bir parçasıdır ve hataları **erken aşamada** tespit ederek geliştirme maliyetlerini düşürmeye yardımcı olur.

Yazılım Testinin Önemi

- **Hata Tespiti:** Yazılım geliştirme sürecinde olası hataları erkenden tespit ederek giderilmesini sağlar.
- **Kalite Güvencesi:** Kullanıcılara güvenilir ve stabil bir yazılım sunulmasını sağlar.
- **Kullanıcı Deneyimi:** Kullanıcıların yazılımı sorunsuz bir şekilde kullanmasını ve memnuniyetini artırır.
- **Güvenlik:** Veri ihlallerini ve güvenlik açıklarını önleyerek sistemin korunmasını sağlar.

Yazılım Test Süreci

Yazılım testi genellikle aşağıdaki aşamalardan oluşur:

1. **Test Planlama ve Stratejisi:** Test kapsamı, kullanılacak test araçları ve hedefler belirlenir.
2. **Test Senaryolarının Hazırlanması:** Yazılımın farklı yönlerini kapsayan test senaryoları ve test vakaları oluşturulur.
3. **Test Ortamının Kurulması:** Yazılımın test edileceği donanım ve yazılım ortamı hazırlanır.
4. **Testlerin Yürütülmesi:** Hazırlanan test vakaları çalıştırılır ve sonuçlar değerlendirilir.
5. **Hata Raporlama ve Düzeltme:** Tespit edilen hatalar raporlanır ve geliştiricilere iletilerek düzeltilmesi sağlanır.
6. **Testlerin Kapanışı:** Test sonuçları analiz edilerek, yazılımın yayınlanmaya hazır olup olmadığına karar verilir.

Örnek Senaryo

Bir çevrimiçi banka uygulaması geliştirdiğimizi düşünelim. Bu uygulama üzerinden kullanıcılar:

- Para transferi yapabilir
- Bakiye sorgulayabilir
- Kredi başvurusu yapabilir

Bu sistemin **hatasız** çalıştığını doğrulamak için uygulanabilecek bazı testler:

- **Fonksiyonel Test:** Kullanıcı giriş yaptıktan sonra **doğru bakiyeyi görüntüleyebiliyor mu?**

- **Performans Testi:** Aynı anda binlerce kullanıcının giriş yapması durumunda sistemin yanıt süresi nasıl etkileniyor?
- **Güvenlik Testi:** Kullanıcıların hesap bilgileri yetkisiz kişiler tarafından erişilebilir mi?
- **Kullanılabilirlik Testi:** Kullanıcılar uygulama içindeki işlemleri kolayca yapabiliyor mu?

Bu test süreçleri sayesinde sistemin güvenilirliği artırılır, hatalar en aza indirilir ve kullanıcı deneyimi iyileştirilir.

Pratik Sorular

1. Yazılım test sürecinin temel aşamaları nelerdir?
2. Fonksiyonel testler ile güvenlik testleri arasındaki fark nedir?
3. Performans testlerinin amacı nedir?

Cevaplar

1. Yazılım test süreci; test planlama, test senaryolarının hazırlanması, test ortamının kurulması, testlerin yürütülmesi, hata raporlama ve düzeltme, test kapanışı aşamalarından oluşur.
2. Fonksiyonel testler, yazılımın belirlenen işlevlerini doğru şekilde yerine getirip getirmediğini doğrularken; güvenlik testleri, yazılımın güvenlik açıklarını belirleyerek veri güvenliğini sağlamaya odaklanır.
3. Performans testleri, sistemin yüksek kullanıcı trafiği ve ağır yük altında nasıl çalıştığını analiz eder. Hız, yanıt süresi ve ölçeklenebilirlik gibi kriterleri değerlendirerek sistemin dayanıklılığını test eder.

5. HATA VE HATA YÖNETİMİ (ERROR & BUG MANAGEMENT)

Hata Nedir?

Bir yazılım hatası (bug), yazılımın beklenmeyen veya yanlış bir şekilde çalışmasına neden olan bir kusurdur. Hatalar, **kodlama, tasarım veya gereksinim aşamalarında** yapılan hatalar sonucunda ortaya çıkabilir. **Yazılım testi**, bu hataları erken aşamada tespit ederek yazılımın **güvenilirliğini artırmayı** hedefler.

Hata yönetimi süreci, hataların **tespit edilmesi, raporlanması, düzeltilmesi ve tekrar test edilmesini** içeren sistematik bir yaklaşımdır.

Hata Yönetimi Süreci

- Hata Tespiti:** Yazılım testi sırasında veya kullanıcı geri bildirimleriyle hatalar tespit edilir.
- Hata Kaydı:** Bulunan hata, hata takip sistemi (**örneğin, Jira, Bugzilla, TFS**) kullanılarak detaylı bir şekilde kaydedilir.
- Hata Sınıflandırması:** Hatanın önemi ve etkisi belirlenir. Genellikle şu seviyelere ayrılır:
 - Kritik:** Sistemin çalışmasını tamamen durduran hatalar.
 - Yüksek:** Ana işlevleri etkileyen ciddi hatalar.
 - Orta:** Kullanıcı deneyimini etkileyen ama temel işlevselliğe zarar vermeyen hatalar.
 - Düşük:** Küçük görsel veya metinsel hatalar.
- Hata Düzeltme:** Geliştirici ekibi, hatayı düzelterek yeni bir yazılım versiyonu oluşturur.
- Tekrar Test (Re-testing):** Test mühendisleri, hatanın gerçekten giderildiğini doğrulamak için tekrar test yapar.
- Regresyon Testi:** Hata düzeltildiğinde, yazılımın **diğer bölümlerinin etkilenip etkilenmediği** kontrol edilir.

Örnek Senaryo

Bir e-ticaret uygulamasında, kullanıcılar **sepete eklenen ürünleri satın almak istediklerinde ödeme ekranına geçiş yapamadıklarını** fark ediyorlar.

- Hata Tespiti:** Test mühendisleri, hata senaryolarını çalıştırarak sorunu doğrular.
- Hata Kaydı:** Hata, **Jira gibi bir hata takip aracında** detaylı bir şekilde kaydedilir.
- Hata Sınıflandırması:** Hata, kullanıcıların **alışveriş yapmasını engellediği için Kritik Seviye** olarak belirlenir.
- Hata Düzeltme:** Geliştiriciler, **ödeme modülündeki bir kod hatasını** tespit edip düzeltir.
- Tekrar Test:** Test mühendisleri, **ödeme ekranının artık düzgün çalıştığını** doğrular.
- Regresyon Testi:** Ödeme sürecinde yapılan değişikliklerin, **sistemin diğer bölümlerine zarar vermediği** kontrol edilir.

Bu süreç tamamlandığında, **hata yönetimi başarılı bir şekilde gerçekleştirilmiş olur** ve yazılımın **kalitesi artırılır**.

Pratik Sorular

1. Yazılım hatası (bug) nedir?
2. Hata yönetimi süreci hangi aşamalardan oluşur?
3. Hata sınıflandırması neden önemlidir?

Cevaplar

1. Yazılım hatası, bir yazılımın beklenen şekilde çalışmamasına neden olan bir kodlama veya tasarım kusurudur.
2. Hata yönetimi süreci; hata tespiti, hata kaydı, hata sınıflandırması, hata düzeltme, tekrar test (re-testing) ve regresyon testi aşamalarından oluşur.
3. Hata sınıflandırması, hangi hataların öncelikli olarak düzeltilmesi gerektiğini belirlemek için önemlidir ve yazılım kalitesini artırmaya yardımcı olur.



6. YAZILIM TESTİNİN GEREKLİLİĞİ (WHY IS TESTING NECESSARY?)

Yazılım Testinin Önemi

Yazılım testi, hataları tespit etmek, yazılımın **güvenilirliğini artırmak** ve son kullanıcıya **kaliteli bir ürün** sunmak için gereklidir. **Hataların erken tespit edilmesi**, yazılımın **maliyetini ve zaman kaybını** azaltırken, **güvenlik açıklarını** da minimize eder.

Yazılım testleri olmadan, hatalar üretimde ortaya çıkabilir ve bu durum **müşteri memnuniyetsizliğine** ve **finansal kayıplara** yol açabilir.

Yazılım Testinin Sağladığı Faydalar

1. **Maliyet ve Zaman Tasarrufu:** Hataların **erken tespit edilmesi**, daha düşük maliyetli çözümler sunar.
2. **Güvenilirlik ve Kalite:** Kullanıcıların **güvenini sağlamak** ve yazılımın **stabil çalışmasını** garanti altına almak için testler yapılır.
3. **Güvenlik:** Özellikle **finans ve sağlık** gibi kritik sistemlerde, güvenlik açıklarını önlemek için test süreci **hayati önem taşır**.
4. **Müşteri Memnuniyeti:** Sorunsuz bir yazılım, müşteri deneyimini iyileştirir ve **marka itibarını** korur.

Örnek Senaryo

Bir mobil bankacılık uygulaması geliştirdiğimizi düşünelim. Kullanıcılar bu uygulama ile:

- Para transferi yapabilir
- Hesap bakiyesini görüntüleyebilir
- Fatura ödemeleri gerçekleştirebilir

Eğer bu uygulamada **para transferi sırasında kullanıcının bakiyesinden yanlış miktarda para düşülürse**, bu ciddi bir hata olur.

- Eğer bu hata **test sürecinde tespit edilmez** ve kullanıcılar tarafından fark edilirse, **banka büyük bir güven kaybına uğrar ve hukuki sorunlarla karşılaşabilir**.
- **Test süreci**, bu tür hataların erken aşamada bulunmasını sağlayarak **banka ve müşteriler için güvenli ve hatasız bir hizmet sunulmasını** garanti eder.

Pratik Sorular

1. Yazılım testi neden gereklidir?
2. Yazılım testinin sağladığı temel faydalar nelerdir?
3. Hata tespiti neden erken aşamalarda yapılmalıdır?

Cevaplar

1. Hataları erken tespit ederek maliyetleri düşürmek, yazılımın güvenilirliğini artırmak ve müşteri memnuniyetini sağlamak için gereklidir.
2. Maliyet tasarrufu, güvenlik, kaliteyi artırma ve müşteri memnuniyetini yükseltme gibi faydalar sağlar.
3. Erken hata tespiti, hata düzeltme maliyetlerini azaltır ve üretim ortamına hatalı yazılımın geçmesini önler.



7. YAZILIM TESTİNİN ANA HEDEFLERİ (MAIN OBJECTIVES OF TESTING)

Yazılım Testinin Temel Hedefleri

- Yazılımın kalitesini artırmak
- Hataları erken tespit etmek
- Yazılımın kullanıcı gereksinimlerine uygun olup olmadığını doğrulamak

Test süreci, yazılımın beklenen sonuçları üretip üretmediğini ve belirlenen gereksinimlere uygun olup olmadığını değerlendirmeye yardımcı olur.

Test Sürecinin Ana Hedefleri

1. **Hataları Bulmak**
 - Yazılım geliştirme sürecinde hataları **mümkün olduğunca erken aşamada** tespit etmek ve düzeltilmesini sağlamak.
2. **Kaliteyi Artırmak**
 - Yazılımın **fonksiyonel ve fonksiyonel olmayan gereksinimleri** karşıladığını doğrulamak.
3. **Güvenilirlik Sağlamak**
 - Yazılımın **beklenen işlevleri doğru bir şekilde yerine getirdiğini ve performans açısından yeterli olduğunu** test etmek.
4. **Riskleri Azaltmak**
 - Test süreci, **olası hataları önceden tespit ederek** sistem arızalarının önüne geçmeye yardımcı olur.

Örnek Senaryo

Bir hastane randevu sistemi geliştirdiğimizi düşünelim. Bu sistem, **hasta kayıt işlemleri, randevu oluşturma ve doktor atama** gibi işlemleri gerçekleştirmektedir.

Test sürecinde kontrol edilmesi gereken ana hedefler:

✓ **Hata Tespiti:** Hasta randevu alırken **yanlış doktor veya tarih ataması** yapılıyor mu? ✓ **Kalite Kontrolü:** Kullanıcılar **randevularını kolayca görebiliyor ve değiştirebiliyor** mu? ✓ **Güvenilirlik:** Sistem, **yoğun hasta başvurularında hala stabil çalışıyor** mu? ✓ **Risk Yönetimi:** Sistem çökerse, **randevu bilgileri kaybolur** mu?

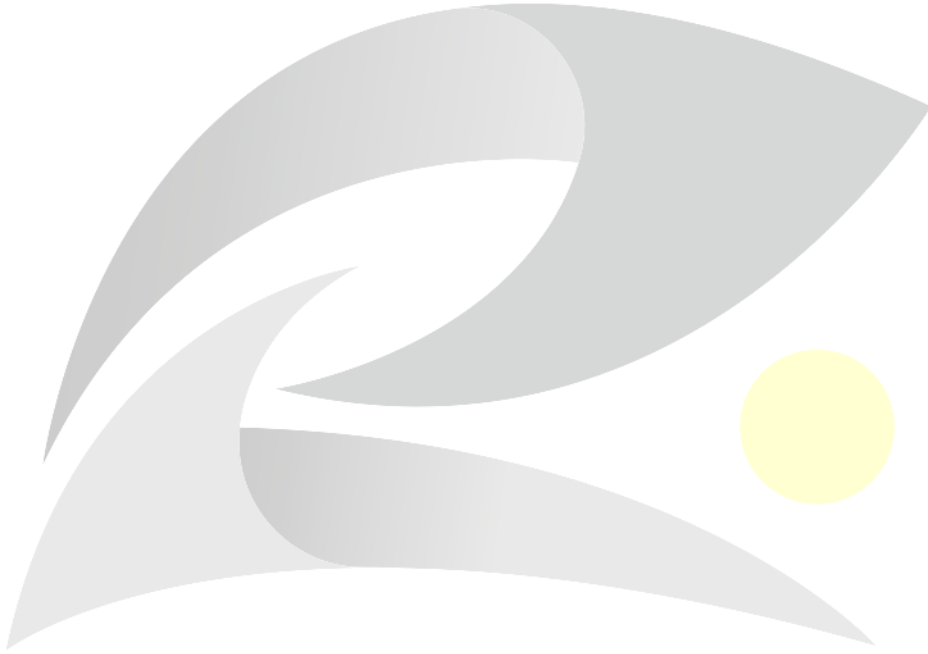
Bu süreç sayesinde, **hastane sistemi güvenilir, hatasız ve kullanıcı dostu** hale getirilmiş olur.

Pratik Sorular

1. **Yazılım testinin temel hedefleri nelerdir?**
2. **Yazılım testi neden kaliteyi artırır?**
3. **Yazılım testinin yazılım geliştirme sürecindeki rolü nedir?**

Cevaplar

1. **Hataları bulmak, kaliteyi artırmak, güvenilirliği sağlamak, riskleri azaltmak.**
2. **Yazılımın gereksinimlere uygun çalıştığını ve hata içermediğini doğruladığı için kaliteyi artırır.**
3. **Yazılımın beklenen işlevleri yerine getirip getirmediğini belirleyerek geliştiricilere geri bildirim sağlar.**



8. YAZILIM TESTİNİN YEDİ PRENSİBİ (SEVEN PRINCIPLES OF TESTING)

Yazılım Testinin Temel Prensipleri

Yazılım testleri belirli temel prensiplere dayanır. Bu prensipler, test sürecinin **daha verimli ve etkili** bir şekilde yürütülmesini sağlar.

ISTQB tarafından belirlenen yedi test prensibi şunlardır:

1. **Testler hataların varlığını gösterir, yokluğunu değil**
 - Test süreci, **hataları belirlemek** için gerçekleştirilir ancak test edilen yazılımın **tamamen hatasız** olduğunu kanıtlamak mümkün değildir.
2. **Kapsamlı test mümkün değildir**
 - **Tüm olası girişleri ve senaryoları** test etmek **pratik olarak imkansızdır**. Bu nedenle, test senaryoları **belirlenmiş kriterlere göre seçilmelidir**.
3. **Erken test önemlidir**
 - Hataların yazılım geliştirme sürecinin **erken aşamalarında tespit edilmesi**, hata düzeltme maliyetini ve süreyi azaltır.
4. **Hatalar kümelenmiştir**
 - Çoğu hata, **yazılımın belirli bölümlerinde yoğunlaşır**. Bu nedenle, **risk temelli test** yaklaşımı benimsenmeli ve **kritik alanlar öncelikli olarak** test edilmelidir.
5. **Pesticide paradoksu**
 - **Aynı testler tekrar tekrar çalıştırıldığında**, yeni hatalar **keşfedilemeyebilir**. Test senaryolarının **düzenli olarak gözden geçirilmesi ve yenilenmesi** gereklidir.
6. **Test, bağlama bağlıdır**
 - Her yazılım projesi **farklıdır** ve test süreçleri de **projenin gereksinimlerine ve bağlamına göre** uyarlanmalıdır.
7. **Yanlış yokluğu hatanın olmadığını göstermez**
 - Bir yazılımın **tüm testlerden geçmesi**, onun **kullanıcı gereksinimlerini tam olarak karşıladığı** anlamına gelmez. **Kullanıcı ihtiyaçları ve beklentileri** test sürecinin dışında değerlendirilmelidir.

Örnek Senaryo

Bir **uçak bileti rezervasyon sistemi** geliştirdiğimizi düşünelim. Kullanıcıların:

- ✓ **Doğru uçuş bilgilerini seçmesi**
- ✓ **Sistemin doğru bilet fiyatını göstermesi**
- ✓ **Rezervasyon işleminin tamamlanması** gerekir.

Eğer test süreci ihmal edilirse, şu hatalar ortaya çıkabilir:


- ✗ **Belirli tarihlerde yanlış fiyatlar gösterilebilir.**
- ✗ **Rezervasyon tamamlanmadan işlem yarıda kesilebilir.**
- ✗ **Ödeme işlemi sırasında hatalar oluşabilir.**

Bu tür hatalar test sürecinde fark edilmezse, **müşteri şikayetlerine ve şirketin itibar kaybetmesine** neden olabilir. **Yazılım testleri**, bu tür kritik hataları **önceden tespit ederek** sistemin sorunsuz çalışmasını sağlar.

Pratik Sorular

1. Yazılım testinin yedi prensibi nelerdir?
2. Kapsamlı test neden mümkün değildir?
3. Pesticide paradoksu nedir ve nasıl önlenebilir?

Cevaplar

1. Testler hataların varlığını gösterir, kapsamlı test mümkün değildir, erken test önemlidir, hatalar kümelenmiştir, pesticide paradoksu, test bağlama bağlıdır, yanlışlığı yokluğu hatanın olmadığını göstermez.
 2. Tüm olası giriş kombinasyonlarını test etmek zaman ve maliyet açısından imkansızdır.
 3. Aynı testleri sürekli çalıştırmak yeni hataların bulunmasını zorlaştırır. Test senaryoları düzenli olarak güncellenmeli ve yeni test teknikleri uygulanmalıdır.
- 

9. TEMEL TEST SÜREÇLERİ (BASIC TEST PROCESSES)

Test Süreçlerinin Önemi

Yazılım testi, belirli süreçlere göre gerçekleştirilir. **Temel test süreçleri**, test sürecinin başarılı bir şekilde yürütülmesi için gerekli tüm aşamaları kapsar.

Bu süreç, **testin planlanmasından sonuçların değerlendirilmesine kadar** olan aşamaları içerir ve yazılımın **hatasız bir şekilde çalışmasını** sağlamayı amaçlar.

Temel Test Süreçleri Aşamaları

- Test Planlama ve Kontrol**
 - Test sürecinin kapsamı, stratejisi, zaman çizelgesi ve kaynakları belirlenir.
 - Kontrol aşaması, planın uygulanmasını izlemek ve gerektiğinde iyileştirmeler yapmak için gerçekleştirilir.
- Test Analizi ve Tasarım**
 - Test edilecek unsurlar belirlenir ve uygun test senaryoları oluşturulur.
 - Test vakaları, gereksinimlere ve test stratejisine uygun şekilde hazırlanır.
- Test Gerçekleştirme ve Yürütme**
 - Planlanan test senaryoları uygulanır ve test sonuçları gözlemlenir.
 - Hatalar tespit edilirse raporlanır ve düzeltme sürecine yönlendirilir.
- Test Değerlendirme ve Raporlama**
 - Test sonuçları analiz edilir ve test sürecinin başarısı değerlendirilir.
 - Hata raporları incelenir ve testlerin ne kadar etkili olduğu belirlenir.
- Testin Sonlandırılması**
 - Test süreçleri tamamlandıktan sonra bulgular belgelenir.
 - Test ortamı kapatılır ve test sürecinin genel değerlendirmesi yapılır.

Örnek Senaryo

Bir **online alışveriş uygulaması** geliştirdiğimizi düşünelim. Kullanıcıların **ürün seçip sipariş vermesi ve ödeme işlemlerini gerçekleştirmesi** gerekir.

Eğer test yapılmazsa:

- ✗ Yanlış ürün gösterilebilir.
- ✗ Ödeme ekranı hata verebilir.
- ✗ Stokta olmayan bir ürün sipariş edilebilir.

Bu tür hataları önlemek için **temel test süreçleri** uygulanır:

- ✓ **Test Planlama:** Sipariş verme süreci test kapsamına dahil edilir.
- ✓ **Test Tasarımı:** Test senaryoları hazırlanır (örneğin, stokta olmayan bir ürünü sipariş etmeye çalışmak).
- ✓ **Test Yürütme:** Kullanıcı işlemleri simüle edilerek testler gerçekleştirilir.
- ✓ **Test Raporlama:** Hata olup olmadığı raporlanır.
- ✓ **Testin Sonlandırılması:** Tüm testler tamamlandıktan sonra süreç kapatılır.

Bu süreçler sayesinde yazılımın hatasız çalışması sağlanır ve müşteri memnuniyeti artırılır.

Pratik Sorular

1. Temel test süreçleri nelerdir?
2. Test planlamasının amacı nedir?
3. Test değerlendirme süreci neden önemlidir?

Cevaplar

1. Test planlama ve kontrol, test analizi ve tasarım, test gerçekleştirme ve yürütme, test değerlendirme ve raporlama, testin sonlandırılması.
2. Test stratejisi oluşturmak, kapsam belirlemek, kaynakları tahsis etmek ve sürecin kontrolünü sağlamak.
3. Test sonuçlarının analizi, hata raporlarının gözden geçirilmesi ve belirlenen kriterlere göre değerlendirilmesi yazılımın kalitesini artırır.

10. YAZILIM TEST SÜRECİ – PLANLAMA VE KONTROL

(SOFTWARE TESTING PROCESS - PLANNING AND CONTROL)

Test Planlama ve Kontrolün Önemi

Test sürecinin en kritik aşamalarından biri olan **test planlama ve kontrol**, yazılımın **test stratejisini belirlemek** ve **test süreçlerini etkin bir şekilde yönetmek** için yapılır.

- **Planlama**, test kapsamını, test ortamını, kaynakları ve zamanlamayı içerir.
- **Kontrol**, test sürecinin ilerleyişini izleme ve gerektiğinde ayarlamalar yapmayı kapsar.

Bu süreç, testlerin **zamanında, bütçe dahilinde ve verimli bir şekilde** gerçekleştirilmesini sağlamayı amaçlar.

Test Planlama Sürecinin Temel Adımları

1. **Test Kapsamının Belirlenmesi**
 - Test edilecek özellikler ve işlevler belirlenir.
 - **Fonksiyonel ve fonksiyonel olmayan testler** için hangi bileşenlerin test edileceği tanımlanır.
2. **Test Stratejisinin Tanımlanması**
 - Kullanılacak **test seviyeleri** (birim, entegrasyon, sistem, kabul testi) ve **test teknikleri** seçilir.
 - **Manuel ve otomatik testlerin** nasıl uygulanacağı belirlenir.
3. **Kaynak Planlaması**
 - Test için gerekli insan gücü, donanım ve yazılım kaynakları tahsis edilir.
 - Test mühendisleri, geliştiriciler ve diğer ilgili ekiplerin rolleri netleştirilir.
4. **Zaman Çizelgesi Oluşturma**
 - Testlerin hangi aşamalarda gerçekleştirileceği belirlenir.
 - **Kritik testler** için belirlenen **son teslim tarihleri** oluşturulur.
5. **Risk Analizi ve Önceliklendirme**
 - Test edilecek alanların risk seviyesi değerlendirilir.
 - **Kritik işlevler** için önceliklendirme yapılır.

Test Kontrol Sürecinin Temel Adımlar

1. **Testlerin İlerleyişini İzleme**
 - Belirlenen plana göre testlerin ilerleyişi takip edilir.
 - **Hedeflenen tarihlere ve kriterlere** ne kadar uygun ilerlendiği gözlemlenir.
2. **Test Sonuçlarının Değerlendirilmesi**
 - Testlerde bulunan hatalar ve yapılan testlerin **başarı oranları** analiz edilir.
 - **Hata yoğunluğu olan alanlar** belirlenir ve raporlanır.
3. **Gerekirse Planın Güncellenmesi**
 - Test sürecinde aksaklıklar yaşanırsa test planı yeniden düzenlenir.

- Geliştiricilerden gelen geri bildirimler doğrultusunda test senaryoları güncellenir.
4. **Kaynak Kullanımının Yönetilmesi**
- Test ekipleri ve test ortamları verimli bir şekilde yönetilir.
 - Kaynak israfını önlemek için test görevleri optimize edilir.

Örnek Senaryo

Bir otel rezervasyon sistemi geliştirdiğimizi düşünelim.

Test planlama sürecinde:

- ✓ **Test Kapsamı:** Kullanıcı giriş yapmadan **rezervasyon yapabiliyor mu?** Rezervasyon sonrası **onay e-postası** gönderiliyor mu?
- ✓ **Test Stratejisi:** Manuel ve otomatik testlerin kombinasyonu uygulanacak.
- ✓ **Kaynak Planlaması:** Test mühendisleri, geliştiriciler ve test ortamı belirlenir.
- ✓ **Zaman Çizelgesi:** İlk testler bir hafta içinde tamamlanmalı.
- ✓ **Risk Analizi:** En kritik işlevler öncelikli olarak test edilmeli.

Test kontrol sürecinde:

- ◆ Testler ilerledikçe süreç takip edilir.
- ◆ Zaman çizelgesine uyulmazsa veya kritik hatalar tespit edilirse test planı güncellenir. ◆ Test ortamları ve ekip kaynakları optimize edilerek süreç verimli hale getirilir.

Bu süreçler sayesinde, otel rezervasyon sistemi hatasız bir şekilde çalışır ve müşteri deneyimi iyileştirilmiş olur.

Pratik Sorular

1. Test planlama sürecinin temel bileşenleri nelerdir?
2. Test kontrolünün amacı nedir?
3. Test planlaması sırasında hangi faktörler göz önünde bulundurulmalıdır?

Cevaplar

1. Test kapsamı belirleme, test stratejisi oluşturma, kaynak planlaması, zaman çizelgesi oluşturma, risk analizi ve önceliklendirme.
2. Test sürecinin etkin bir şekilde yönetilmesini sağlar ve kaynak israfını önler.
3. Proje gereksinimleri, test kaynakları, zaman kısıtlamaları, yazılımın karmaşıklığı.

11. YAZILIM TEST SÜRECİ - ANALİZ VE TASARIM

(SOFTWARE TESTING PROCESS - ANALYSIS AND DESIGN)

Analiz ve Tasarım Aşamasının Önemi

Analiz ve tasarım aşaması, test sürecinin en kritik adımlarından biridir. Bu aşamada:

- Test senaryoları oluşturulur,
- Test stratejisi belirlenir,
- Gerekli test verileri hazırlanır.

Test süreçlerinin doğru planlanması, testlerin etkinliğini artırarak hata tespit oranını yükseltir.

Analiz ve Tasarım Sürecinin Temel Adımları

1. Test Koşullarının Belirlenmesi
 - Test edilecek işlevler tanımlanır.
 - Yazılımın hangi özellikleri ve bileşenlerinin test edileceği belirlenir.
2. Test Senaryolarının Oluşturulması
 - Test edilecek durumlar ve adımlar tanımlanır.
 - Gerçekleştirilecek testlerin ayrıntılı senaryoları oluşturulur.
3. Test Verilerinin Hazırlanması
 - Test sürecinde kullanılacak veriler tanımlanır ve yönetilir.
 - Gerçekçi ve anlamlı test verileri seçilir.
4. Test Ortamının Tanımlanması
 - Testlerin gerçekleştirileceği ortam belirlenir.
 - Test cihazları, ağ yapılandırmaları, veri tabanları ve diğer bileşenler hazırlanır.
5. Test Tasarım Tekniklerinin Seçilmesi
 - Kara kutu testi, beyaz kutu testi gibi test teknikleri belirlenir.
 - Fonksiyonel ve fonksiyonel olmayan test yaklaşımları seçilir.

Örnek Senaryo

Bir online bankacılık sistemi geliştirdiğimizi düşünelim. Kullanıcıların para transferi yaparken doğru alıcıya ve doğru tutarda işlem yaptığından emin olunması gerekir.

- ✓ Test Koşulları: Kullanıcı giriş yapmadan para transferi yapabiliyor mu? Yanlış IBAN girildiğinde hata mesajı gösteriliyor mu?
- ✓ Test Senaryoları: Başarılı ve başarısız para transferi için test senaryoları oluşturulur.
- ✓ Test Verileri: Gerçekçi kullanıcı hesapları ve farklı para miktarları ile test verileri hazırlanır.

- ✓ **Test Ortamı:** Bankanın canlı ortamına benzer bir test ortamı oluşturulur.
- ✓ **Test Teknikleri:** Sınır değeri analizi ve karar tablosu testi gibi yöntemler uygulanır.

Bu süreçler sayesinde, bankacılık işlemlerinde hata riski en aza indirilerek kullanıcıların güvenli işlemler yapması sağlanır.

Pratik Sorular

1. Test analiz ve tasarım aşamasında hangi faaliyetler gerçekleştirilir?
2. Test senaryoları nasıl oluşturulur?
3. Test teknikleri nasıl belirlenir?

Cevaplar

1. Test koşullarının belirlenmesi, test senaryolarının oluşturulması, test verilerinin hazırlanması, test ortamının tanımlanması, test tekniklerinin seçilmesi.
2. Test edilecek fonksiyonlar belirlenerek, test verileri ve beklenen sonuçlar tanımlanır.
3. Testin türüne göre uygun test teknikleri (eşdeğer bölümlere ayırma, sınır değeri analizi, karar tablosu vb.) belirlenir.

12. YAZILIM TEST SÜRECİ – GERÇEKLEŞTİRME VE YÜRÜTME

(SOFTWARE TESTING PROCESS - EXECUTION AND IMPLEMENTATION)

Gerçekleştirme ve Yürütme Aşamasının Önemi

Gerçekleştirme ve yürütme aşaması, testlerin planlandığı şekilde uygulanmasını ve yazılımın beklenen gereksinimleri karşılayıp karşılamadığını doğrulamayı içerir.

Bu aşama şunları kapsar:

- Test senaryolarının yürütülmesi,
- Test sonuçlarının kaydedilmesi,
- Olası hataların raporlanması ve tekrar test edilmesi.

Bu süreç, yazılımın gerçek kullanıcı senaryolarına göre nasıl çalıştığını değerlendirmek için kritik bir adımdır.

Gerçekleştirme ve Yürütme Sürecinin Temel Adımları

1. **Test Ortamının Hazırlanması**
 - Testlerin çalıştırılacağı donanım, yazılım ve ağ yapılandırmalarının doğrulanması.
 - Gerekli test verilerinin yüklenmesi ve ortamın stabil hale getirilmesi.
2. **Test Senaryolarının Çalıştırılması**
 - Önceden oluşturulmuş test vakaları çalıştırılır.
 - Testlerin hem manuel hem de otomatik olarak yürütülmesi sağlanır.
3. **Sonuçların Kaydedilmesi**
 - Testlerin çıktıları belgelenir ve beklenen sonuçlarla karşılaştırılır.
 - Başarılı ve başarısız test durumları detaylı olarak raporlanır.
4. **Hata Raporlama**
 - Tespit edilen hatalar hata takip sistemine (*Jira, Bugzilla vb.*) kaydedilir.
 - Hata türü belirlenir (*Kritik, Yüksek, Orta, Düşük*).
5. **Testlerin Tekrar Çalıştırılması (Re-testing)**
 - Hata düzeltildikten sonra, test tekrar çalıştırılarak sorunun gerçekten çözüldüğü doğrulanır.
6. **Regresyon Testi**
 - Yapılan değişikliklerin yazılımın diğer bölümlerini etkileyip etkilemediği test edilir.
 - Önceden çalışan fonksiyonların bozulmadığına emin olunur.

Örnek Senaryo

Bir e-ticaret platformunda, kullanıcıların sepete ürün ekleyip ödeme yapması gerektiğini düşünelim.

- ✓ **Test Ortamı Hazırlanır:** Gerçek kullanıcı senaryolarına benzer bir test ortamı kurulur.
- ✓ **Test Senaryoları Çalıştırılır:** Kullanıcı giriş yapmadan sepete ürün eklemeye çalışır, eksik ödeme bilgileri girildiğinde hata mesajı alınıp alınmadığı kontrol edilir.
- ✓ **Sonuçlar Kaydedilir:** Her testin başarılı olup olmadığı raporlanır.
- ✓ **Hata Raporlanır:** Sepete eklenen ürünlerin toplam fiyatının yanlış hesaplandığı tespit edilirse, hata kaydedilir.
- ✓ **Testler Tekrar Çalıştırılır:** Hata giderildikten sonra yeniden test edilir.
- ✓ **Regresyon Testi Yapılır:** Sepetle ilgili bir hata düzeltilirken, ödeme sayfasının etkilenip etkilenmediği test edilir.

Bu süreçler sayesinde:

- ◆ **Kullanıcıların sorunsuz alışveriş yapması sağlanır.**
- ◆ **Yazılımın kalitesi artırılır ve hata oranı minimuma indirilir.**

Pratik Sorular

1. Test yürütme sürecinde hangi adımlar takip edilir?
2. Hata raporlama süreci nasıl işler?
3. Test ortamı neden önemlidir?

Cevaplar

1. Test ortamının hazırlanması, test senaryolarının çalıştırılması, sonuçların kaydedilmesi, hata raporlama, tekrar test (re-testing) ve regresyon testi aşamalarını içerir.
2. Tespit edilen hata, hata yönetim sistemine kaydedilir, hata sınıflandırılır, düzeltilir ve yeniden test edilir.
3. Gerçek ortamı simüle etmek ve doğru test sonuçları elde etmek için gereklidir.

13. YAZILIM TEST SÜRECİ - DEĞERLENDİRME (SOFTWARE TESTING PROCESS -EVALUATION)

Test Değerlendirme Aşamasının Önemi

Değerlendirme aşaması, test sürecinin etkinliğini analiz etmeyi ve yazılımın belirlenen kalite gereksinimlerini karşılayıp karşılamadığını belirlemeyi amaçlar.

Bu süreç, testlerin doğruluk ve eksiksizlik açısından incelenmesini kapsar ve testlerin ne kadar başarılı olduğunu analiz etmek için kritik bir adımdır.

Test Değerlendirme Sürecinin Temel Adımları

1. Test Sonuçlarının Analizi
 - Testlerin başarı oranları ve bulunan hataların analizi yapılır.
 - Test edilen özelliklerin gereksinimlere uygun çalışıp çalışmadığı değerlendirilir.
2. Hata Raporlarının İncelenmesi
 - Tespit edilen hatalar gözden geçirilir.
 - Çözüm süreçleri incelenir ve giderilen hataların gerçekten düzeltilip düzeltilmediği kontrol edilir.
3. Kalite Kriterlerine Göre Değerlendirme
 - Yazılımın belirlenen kalite standartlarını karşılayıp karşılamadığı test edilir.
 - Fonksiyonel, performans ve güvenlik gereksinimlerinin karşılandığı doğrulanır.
4. Test Kapsamının Değerlendirilmesi
 - Uygulanan testlerin yeterliliği kontrol edilir.
 - Eksik veya gözden kaçan test senaryoları tespit edilir.
5. Geri Bildirim ve Sürekli İyileştirme
 - Test sürecinden elde edilen bilgilerle, gelecekteki test süreçlerinin iyileştirilmesine yönelik öneriler sunulur.
 - Geliştirici ekip ve test mühendisleri arasında geri bildirim paylaşımı yapılır.

Örnek Senaryo

Bir online eğitim platformunun test sürecini değerlendirdiğimizi düşünelim. Testlerin tamamlanmasının ardından:

- ✓ Test Sonuçları Analiz Edilir: Kullanıcı kayıt işlemi sırasında tespit edilen hataların çözüme kavuşturulup kavuşturulmadığı kontrol edilir.
- ✓ Hata Raporları Gözden Geçirilir: En fazla hata içeren alanlar belirlenir (örneğin, ödeme sayfasında yaşanan sorunlar).
- ✓ Kalite Kriterleri Değerlendirilir: Platformun yük testi sonuçlarına göre belirlenen performans hedeflerine ulaşp ulaşmadığı analiz edilir.
- ✓ Test Kapsamı İncelenir: Test edilen senaryoların yeterli olup olmadığı gözden geçirilir.

- ✓ **Geri Bildirimler Toplanır:** Kullanıcı geri bildirimleri ve test ekibinin önerileri doğrultusunda, gelecekte yapılacak test süreçleri iyileştirilir.

Bu aşama sayesinde:

- ◆ **Test süreci daha etkin hale getirilir.**
- ◆ **Yazılımın kalitesi güvence altına alınır.**
- ◆ **Gelecekteki test süreçleri için sürekli iyileştirme sağlanır.**

Pratik Sorular

1. Test değerlendirme aşamasında hangi faaliyetler gerçekleştirilir?
2. Test sürecinin başarı kriterleri nelerdir?
3. Test değerlendirme raporu ne içerir?

Cevaplar

1. Test sonuçlarının analiz edilmesi, hata raporlarının incelenmesi, belirlenen kalite kriterlerine göre değerlendirme yapılması.
2. Testlerin tamamlanma yüzdesi, hataların bulunma oranı, kullanıcı gereksinimlerinin karşılanma düzeyi.
3. Test senaryolarının başarı durumu, tespit edilen hatalar, test sürecinin genel değerlendirmesi.

14. YAZILIM TEST SÜRECİ – TESTİN SONLANDIRILMASI

(SOFTWARE TESTING PROCESS – ENDING THE TEST)

Testin Sonlandırılması Aşamasının Önemi

Testin sonlandırılması aşaması, testlerin tamamlandığını ve yazılımın belirlenen kriterlere uygun olup olmadığını doğrulamak için yapılan son kontrolleri içerir.

Bu aşamada:

- Test sonuçları gözden geçirilir ve raporlanır.
- Açık kalan testler analiz edilir.
- Gelecekteki test süreçleri için geri bildirim toplanarak sürekli iyileştirme sağlanır.

Bu süreç tamamlandıktan sonra, yazılım yayına hazır hale gelir ve test ekibi yeni projelere yönlendirilir.

Testin Sonlandırılması Sürecinin Temel Adımları

1. Test Ortamının Temizlenmesi
 - Test sırasında oluşturulan geçici veriler, test hesapları ve test araçları kaldırılır.
 - Test ortamı gereksiz verilerden arındırılır.
2. Test Raporlarının Hazırlanması
 - Test sürecinin sonuçları raporlanır ve paydaşlarla paylaşılır.
 - Başarılı ve başarısız test senaryoları detaylandırılır.
3. Açık Kalan Testlerin Değerlendirilmesi
 - Henüz tamamlanmamış veya çözilememiş hatalar analiz edilir.
 - Düşük öncelikli hatalar bir sonraki sürüme bırakılabilir.
4. Öğrenilen Derslerin Belirlenmesi
 - Test sürecinde karşılaşılan zorluklar ve iyileştirme fırsatları belirlenir.
 - Gelecekteki test stratejileri için öneriler oluşturulur.
5. Test Sürecinin Resmi Kapanışı
 - Test faaliyetleri sona erdirilir ve kapanış toplantısı düzenlenir.
 - Test ekibi yeni projelere yönlendirilir.

Örnek Senaryo

Bir mobil ödeme uygulamasının test sürecinin tamamlandığını düşünelim.

- ✓ Test Ortamı Temizlenir: Test sırasında oluşturulan sahte hesaplar ve test verileri silinir.
- ✓ Test Raporu Hazırlanır: Ödeme işlevselliğinin başarı oranı ve tespit edilen hatalar hakkında detaylı bir rapor hazırlanır.

- ✓ **Açık Kalan Testler İncelenir:** Düşük öncelikli veya sonraki sürümlerde düzeltilecek hatalar belirlenir.
- ✓ **Öğrenilen Dersler Kaydedilir:** Test sürecinde karşılaşılan zorluklar ve iyileştirme fırsatları belirlenir.
- ✓ **Süreç Kapatılır:** Test ekibi, sürecin başarıyla tamamlandığını belirten bir kapanış toplantısı düzenler.

Bu süreç sayesinde:

- ◆ Yazılımın belirlenen kalite standartlarına uygun olup olmadığı doğrulanır.
- ◆ Gelecekteki test süreçleri daha verimli hale getirilir.
- ◆ Yazılım, hatasız ve sorunsuz bir şekilde yayına alınır.

Pratik Sorular

1. Testin sonlandırılması aşamasında hangi işlemler gerçekleştirilir?
2. Test kapatma raporu neyi içerir?
3. Neden test sonlandırma aşaması önemlidir?

Cevaplar

1. Test ortamının temizlenmesi, test raporlarının tamamlanması, açık kalan testlerin değerlendirilmesi, öğrenilen derslerin belirlenmesi, test sürecinin resmi olarak kapanması.
2. Test faaliyetlerinin özetlenmesi, tespit edilen hataların durumları, test süreçlerinin genel performans analizi.
3. Test sürecinin tamamlandığını ve yazılımın yayına hazır olduğunu doğrulamak için gereklidir.

15. YENİ TEST SÜREÇLERİ VE STANDARTLARI (NEW TEST PROCESS AND STANDARDS)

Yeni Test Süreçlerinin Önemi

Yazılım test süreçleri, yazılım dünyasındaki gelişmelere uyum sağlamak için sürekli olarak güncellenmektedir. Yeni teknolojiler, metodolojiler ve otomasyon sistemleri, test süreçlerinin daha hızlı, güvenilir ve verimli hale gelmesini sağlar.

Günümüzde Agile, DevOps, yapay zeka destekli testler ve otomasyon gibi modern yaklaşımlar, test süreçlerini optimize ederek yazılım kalitesini artırmaktadır.

Güncel Test Süreçleri ve Standartları

- Agile ve DevOps Test Süreçleri**
 - Sürekli entegrasyon ve sürekli teslimat (CI/CD) ile uyumlu test yöntemleri kullanılır.
 - Sık dağıtımlar ve hızlı geri bildirim süreçleri desteklenir.
- Yapay Zeka Destekli Testler**
 - Test süreçlerini optimize etmek için yapay zeka (AI) ve makine öğrenmesi (ML) teknikleri kullanılır.
 - Hata tespiti ve tahmini analizler sayesinde test süreçleri daha verimli hale gelir.
- Otomatik Test Süreçleri**
 - Manuel test süreçlerini hızlandırmak için test otomasyon araçları (Selenium, Cypress, Appium vb.) kullanılır.
 - Sürekli test (Continuous Testing) sayesinde yazılım geliştirme sürecinin her aşamasında test yapılabilir.
- ISO/IEC 29119 Test Standartları**
 - Yazılım test süreçlerinin uluslararası kabul görmüş standartlara uygun olarak yürütülmesini sağlar.
 - Test planlaması, dokümantasyonu, yürütme ve yönetim süreçlerini kapsar.
- Risk Temelli Test Yaklaşımları**
 - En kritik alanlara odaklanarak test kapsamı optimize edilir.
 - Test senaryoları, yazılımın en çok risk taşıyan bölümlerine göre önceliklendirilir.

Örnek Senaryo

Bir mobil bankacılık uygulaması geliştiren bir şirketi düşünelim.

Geleneksel manuel testler yerine:

- ✓ Otomatik test senaryoları ve yapay zeka destekli test araçları kullanılarak test süreçleri hızlandırılır.
- ✓ CI/CD sistemleri ile entegre testler yapılarak, her yeni sürümde hatalar otomatik olarak

tespit edilir.

- ✓ **Risk temelli test yaklaşımı kullanılarak, en kritik güvenlik ve fonksiyonel testler önceliklendirilir.**
- ✓ **ISO/IEC 29119 test standartlarına uygun test dokümantasyonu hazırlanır.**

Bu sayede:

- ◆ **Test süreci daha hızlı ve verimli hale gelir.**
- ◆ **Hata tespit oranı yükselir ve yazılım kalitesi artar.**
- ◆ **Geliştirme süreci kesintiye uğramadan testler sürekli olarak yürütülür.**

Pratik Sorular

1. **Yeni test süreçlerinin amacı nedir?**
2. **ISO/IEC 29119 test standartları neyi kapsar?**
3. **Otomasyon test süreçlerinin avantajları nelerdir?**

Cevaplar

1. **Yeni teknolojilere uyum sağlamak, test süreçlerini daha verimli hale getirmek ve yazılım kalitesini artırmak.**
2. **Yazılım test süreçlerini tanımlayan uluslararası bir standarttır ve test dokümantasyonu, yönetimi ve yürütme süreçlerini kapsar.**
3. **Daha hızlı test yürütme, hata tespit sürecini hızlandırma, tekrarlanabilirlik ve maliyet tasarrufu sağlar.**

16. TEST PSİKOLOJİSİ

(TEST PSYCHOLOGY)

Test Psikolojisinin Önemi

Test psikolojisi, yazılım testi sürecinde test mühendislerinin nasıl düşündüğünü ve psikolojik faktörlerin test süreçlerine nasıl etki ettiğini inceler.

Yazılım testi süreci:

- Sabır,
- Detaylara dikkat,
- Eleştirel düşünme gibi beceriler gerektirir.

Test mühendislerinin motivasyonu ve test sürecine yaklaşımları, testlerin başarısını doğrudan etkileyebilir.

Test Psikolojisinin Temel Unsurları

1. Eleştirel Düşünme
 - Test uzmanları, yazılımı kullanıcının bakış açısından analiz etmeli ve olası hata senaryolarını belirlemelidir.
 - Her olasılığı sorgulayan bir bakış açısı ile testleri planlamalıdır.
2. Dikkat ve Sabır
 - Test süreçleri bazen tekrarlayıcı olabilir.
 - Test mühendisleri, en küçük detayları bile gözden kaçırmamalıdır.
3. Bağımsızlık
 - Test uzmanları, geliştirme ekibinden bağımsız çalışarak yazılımı objektif bir şekilde değerlendirmelidir.
 - Geliştirme ekibinin etkisinde kalmadan, yazılımın gerçek performansını analiz etmelidirler.
4. Ekip Çalışması ve İletişim
 - Hataların belirlenmesi ve düzeltilmesi sürecinde, geliştiricilerle etkili iletişim kurmak gerekir.
 - Yanlış anlaşılmalara önüne geçmek için test sonuçları iyi bir şekilde açıklanmalıdır.
5. Olumsuzluk Yanılsaması ile Başa Çıkma
 - Test mühendisleri sürekli hata aradıkları için bazen geliştirme ekibiyle ters düşebilirler.
 - Bu yüzden pozitif bir iş ilişkisi kurarak yapıcı geri bildirim vermek önemlidir.

Örnek Senaryo

Bir test mühendisi, bir mobil uygulamanın kullanıcı deneyimini test ederken, belirli bir ekranda düğmelerin yanlış hizalandığını fark eder.

- Geliştirici ekibi, bu hatanın önemli olmadığını düşünebilir.
- Ancak test mühendisi, bu düzensiz hizalamanın kullanıcıların güvenilirlik algısını nasıl etkileyebileceğini değerlendirerek hatanın düzeltilmesi için ısrarcı olur.
- Bu tür durumlar, test uzmanlarının eleştirel düşünme becerilerini ve iletişim yeteneklerini kullanmasını gerektirir.

Bu örnek, test psikolojisinin test mühendisleri için neden kritik olduğunu gösterir.

Pratik Sorular

1. Test psikolojisi neden önemlidir?
2. Test mühendislerinin karşılaştığı temel psikolojik zorluklar nelerdir?
3. Test mühendisleri nasıl motive edilmelidir?

Cevaplar

1. Test uzmanlarının dikkatli ve odaklanmış olmasını sağlar, test süreçlerinin daha verimli yürütülmesine katkıda bulunur.
2. Tekrarlanan test süreçleri nedeniyle motivasyon kaybı, zaman baskısı altında çalışma, kullanıcı beklentileriyle çelişen test bulguları.
3. Test uzmanlarına eğitim ve gelişim fırsatları sunmak, başarılı test sonuçlarını teşvik etmek, test süreçlerinin çeşitlendirilmesini sağlamak.

17. TEST FAALİYETLERİ VE GÖREVLERİ

(TESTING ACTIVITIES AND TASKS)

Test Sürecinin Önemi

Test süreci, belirli faaliyetler ve görevlerden oluşur. Testlerin planlanmasından, uygulanmasına, hata raporlamasından sonuçların değerlendirilmesine kadar birçok aşama içerir.

Başarılı bir test süreci için bu görevlerin her biri dikkatlice yürütülmelidir.

Test Faaliyetleri ve Görevleri

1. Test Planlama
 - Test sürecinin kapsamını, stratejisini ve zaman çizelgesini belirleme.
 - Hangi test türlerinin uygulanacağını ve hangi kaynakların kullanılacağını tanımlama.
2. Test Tasarımı
 - Test senaryolarını ve test vakalarını oluşturma.
 - Fonksiyonel ve fonksiyonel olmayan gereksinimlere göre test planları hazırlama.
3. Test Ortamının Hazırlanması
 - Testlerin gerçekleştirileceği yazılım, donanım ve ağ ortamlarının oluşturulması.
 - Gerçek kullanım senaryolarına en uygun test ortamını simüle etme.
4. Testlerin Uygulanması
 - Test senaryolarının manuel veya otomatik test araçlarıyla yürütülmesi.
 - Testlerin çıktılarının değerlendirilmesi ve hata bulgularının incelenmesi.
5. Hata Raporlama
 - Bulunan hataların detaylı bir şekilde dokümente edilmesi.
 - Hata takip araçları (*Jira, Bugzilla vb.*) kullanılarak hataların takibinin sağlanması.
6. Test Sonuçlarının Analizi
 - Testlerin başarı durumlarının gözden geçirilmesi.
 - İyileştirme önerileri ve yazılım kalitesi ile ilgili geri bildirim sağlanması.

Örnek Senaryo

Bir online yemek siparişi uygulamasının test sürecini düşünelim.

- ✓ **Test Planlama:** Kullanıcıların sipariş verme, ödeme yapma ve sipariş takibi işlemlerinin test edileceği belirlenir.
- ✓ **Test Tasarımı:** “Sipariş oluştur butonu tıklandığında doğru fiyat hesaplanıyor mu?” gibi test senaryoları oluşturulur.
- ✓ **Test Ortamı Hazırlanır:** Farklı cihaz ve tarayıcılarda uygulamanın test edilmesi için uygun test ortamları kurulur.
- ✓ **Testlerin Uygulanması:** Kullanıcı senaryoları manuel ve otomatik test araçlarıyla yürütülür.
- ✓ **Hata Raporlama:** Ödeme ekranında hata tespit edilirse, hata ayrıntılı bir şekilde

raporlanır ve geliştiricilere iletilir.

- ✓ **Sonuçların Analizi:** Test sonuçları değerlendirilerek uygulamanın stabil olup olmadığına karar verilir.

Bu süreçler sayesinde yazılımın istenilen kalitede çalışması sağlanır ve kullanıcı deneyimi en üst seviyeye çıkarılır.

Pratik Sorular

1. Yazılım test sürecinin temel faaliyetleri nelerdir?
2. Test planlama aşamasında hangi adımlar takip edilir?
3. Hata raporlama sürecinin önemi nedir?

Cevaplar

1. Test planlama, analiz ve tasarım, test ortamının hazırlanması, test yürütme, hata raporlama ve test değerlendirme.
2. Test kapsamı belirlenir, test stratejisi oluşturulur, kaynaklar atanır ve riskler değerlendirilir.
3. Hata raporlama, tespit edilen hataların düzeltilmesini sağlamak ve yazılımın kalitesini artırmak için gereklidir.

18. YAZILIM TESTİ İŞ ÜRÜNLERİ

(SOFTWARE TESTING BUSINESS PRODUCTS)

Test İş Ürünlerinin Önemi

Yazılım test süreci sonucunda çeşitli iş ürünleri ortaya çıkar. Bu iş ürünleri:

- Test sürecinin kayıt altına alınmasını sağlar,
- Test süreçlerinin izlenebilirliğini artırır,
- Yazılım geliştirme sürecinde kaliteyi ölçmeye yardımcı olur.

Test iş ürünleri, test süreçlerinin başarısını değerlendirmek ve testlerin etkinliğini analiz etmek için kritik öneme sahiptir.

Yazılım Testi İş Ürünleri

1. **Test Planı**
 - Test sürecinin stratejisini, kapsamını, zamanlamasını ve kaynaklarını içeren belge.
 - Testin nasıl yürütüleceğini tanımlar.
2. **Test Senaryoları ve Test Vakaları**
 - Yazılımın belirli bölümlerinin test edilmesi için hazırlanan detaylı adımlar.
 - Fonksiyonel ve fonksiyonel olmayan gereksinimlere göre testlerin oluşturulması.
3. **Test Ortamı Dokümantasyonu**
 - Testlerin gerçekleştirileceği donanım ve yazılım ortamının tanımlandığı belge.
 - Test ortamının doğru yapılandırıldığından emin olmak için kullanılır.
4. **Hata Raporları**
 - Test sırasında tespit edilen hataların detaylarını içeren kayıtlar.
 - Hata takip araçları (*Jira, Bugzilla vb.*) kullanılarak yönetilir.
5. **Test Sonuç Raporları**
 - Testlerin başarı durumunu, gerçekleştirilen testleri ve tespit edilen hataları özetleyen rapor.
 - Hangi testlerin başarılı veya başarısız olduğunu gösterir.
6. **Test Kapanış Raporu**
 - Test sürecinin sonunda, test süreçlerinin özetlendiği ve elde edilen sonuçların açıklandığı belge.
 - Yazılımın yayına hazır olup olmadığına karar vermek için kullanılır.

Örnek Senaryo

Bir e-ticaret platformunun test sürecinde üretilen iş ürünlerini düşünelim:

- ✓ **Test Planı:** Ödeme işlemleri, ürün listeleme özellikleri ve kullanıcı kaydının test edileceği belirlenir.
- ✓ **Test Senaryoları:** “Kullanıcı, ödeme ekranına gelip kredi kartı bilgilerini eksik girerse hata mesajı gösterilmeli” gibi test vakaları oluşturulur.
- ✓ **Test Ortamı Dokümantasyonu:** Farklı tarayıcılar ve mobil cihazlarda testlerin

yapılacağı belirtilir.

- ✓ **Hata Raporları:** Ödeme ekranında yanlış tutar hesaplandığı tespit edilir ve hata kaydedilir.
- ✓ **Test Sonuç Raporları:** 100 test vakasından 95'inin başarılı olduğu, 5 hatanın bulunduğu belirtilir.
- ✓ **Test Kapanış Raporu:** Test sürecinin tamamlandığı, kritik hataların giderildiği ve yazılımın yayına hazır olduğu raporlanır.

Bu iş ürünleri sayesinde:

- ◆ Test süreci daha şeffaf hale gelir.
- ◆ Test süreçlerinin izlenebilirliği artar.
- ◆ Test sonuçları net bir şekilde raporlanır ve analiz edilir.

Pratik Sorular

1. Yazılım testi iş ürünleri nelerdir?
2. Test dokümantasyonunun önemi nedir?
3. Test iş ürünleri nasıl yönetilir?

Cevaplar

1. Test planları, test senaryoları, test ortamı dokümantasyonu, hata raporları, test sonuç raporları, test kapanış raporları.
2. Test dokümantasyonu, sürecin şeffaf olmasını sağlar ve gelecekteki test süreçleri için referans oluşturur.
3. Test iş ürünleri, sürüm kontrol sistemleri ve test yönetim araçları ile yönetilir.

19. YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ BOYUNCA TEST (TESTING THROUGHOUT THE SOFTWARE LIFECYCLE)

Testin Yazılım Geliştirme Sürecindeki Rolü

Yazılım geliştirme süreci boyunca farklı aşamalarda yapılan testler, hataların erken tespit edilmesini sağlayarak yazılım kalitesini artırır.

- Her aşamada test uygulanması, geliştirme maliyetlerini düşürerek daha verimli bir süreç sağlar.
- Erken test yaklaşımı, kullanıcı deneyimini iyileştirerek daha güvenilir ve hatasız bir yazılım ortaya çıkmasına yardımcı olur.

Yazılım Geliştirme Yaşam Döngüsündeki Test Aşamaları

1. Gereksinim Analizi Aşaması
 - Test uzmanları, yazılımın gereksinimlerini analiz ederek test edilebilirlik açısından değerlendirme yapar.
 - Eksik veya çelişkili gereksinimler belirlenerek erken aşamada düzeltilir.
2. Tasarım Aşaması
 - Test senaryoları oluşturulur, test stratejileri belirlenir ve risk analizleri yapılır.
 - Fonksiyonel ve fonksiyonel olmayan test senaryoları hazırlanır.
3. Geliştirme Aşaması
 - Birim testler gerçekleştirilerek her modül bağımsız olarak test edilir.
 - Kodun doğru çalıştığından emin olunarak hatalar erken aşamada tespit edilir.
4. Entegrasyon Aşaması
 - Birlikte çalışan modüller test edilerek bileşenler arasındaki hatalar tespit edilir.
 - API, veri akışı ve sistem entegrasyonları doğrulanır.
5. Sistem Testi Aşaması
 - Yazılımın bütün olarak çalışıp çalışmadığı test edilir.
 - Performans, güvenlik, yük ve kullanılabilirlik testleri yapılır.
6. Kabul Testi Aşaması
 - Yazılımın müşteri gereksinimlerini karşılayıp karşılamadığı doğrulanır.
 - Gerçek kullanıcı senaryolarına göre testler gerçekleştirilir.
7. Bakım Testi
 - Canlıya alınan yazılımın güncellemeleri ve düzeltmeleri test edilir.
 - Yeni eklenen özelliklerin mevcut sistemle uyumlu olup olmadığı kontrol edilir.

Örnek Senaryo

Bir mobil alışveriş uygulaması geliştirdiğimizi düşünelim. Test süreci şu aşamalardan geçer:

- ✓ **Gereksinim Analizi:** Kullanıcılar ödeme yaparken hangi özelliklerin olması gerektiği belirlenir.
- ✓ **Tasarım Aşaması:** “Ödeme işlemi tamamlandığında kullanıcıya onay mesajı gösterilmeli” gibi test senaryoları hazırlanır.
- ✓ **Geliştirme Aşaması:** Ödeme işlemiyle ilgili kod geliştirilirken, geliştirici birim testleri gerçekleştirir.
- ✓ **Entegrasyon Aşaması:** Ödeme işleminin sipariş yönetim sistemiyle uyumlu çalıştığı test edilir.
- ✓ **Sistem Testi:** Tüm uygulama bir bütün olarak test edilir ve performansı değerlendirilir.
- ✓ **Kabul Testi:** Kullanıcıların gerçek dünya senaryolarında uygulamayı test etmesi sağlanır.
- ✓ **Bakım Testi:** Yeni bir ödeme yöntemi eklendiğinde, sistemde mevcut özelliklerin hala sorunsuz çalıştığı test edilir.

Bu süreçler sayesinde:

- ◆ Yazılımın kullanıcı beklentilerini karşılaması sağlanır.
- ◆ Hatalar erken aşamada tespit edilir ve maliyet düşürülür.
- ◆ Yazılımın yüksek kalite standartlarını karşılaması garanti edilir.

Pratik Sorular

1. Yazılım geliştirme yaşam döngüsünde testin rolü nedir?
2. Yazılım testinin erken aşamalarda yapılmasının avantajları nelerdir?
3. Yazılım geliştirme süreçlerinde testin sürekli uygulanması neden önemlidir?

Cevaplar

1. Test, yazılımın tüm aşamalarında uygulanarak hataların erken tespit edilmesini sağlar.
2. Hataların erken tespiti maliyetleri düşürür ve yazılımın kalitesini artırır.
3. Sürekli test, yazılımın gelişim sürecinde yüksek kalite standartlarını karşılamasına yardımcı olur.

20. YAZILIM TEST SEVİYELERİ (SOFTWARE TEST LEVELS)

Test Seviyelerinin Önemi

Yazılım testi, yazılım geliştirme sürecinde belirli seviyelerde gerçekleştirilir.

Her test seviyesi:

- Belirli bir amaca hizmet eder
- Yazılımın farklı yönlerini değerlendirir
- Sistemin gereksinimlere uygunluğunu doğrulamak için uygulanır

Bu seviyeler sayesinde, yazılımın her aşamasında hatalar tespit edilerek güvenilirliği artırılır.

Yazılım Test Seviyeleri

1. **Birim Testi (Unit Testing)**
 - Yazılımın en küçük bileşenlerini (*fonksiyonlar, metotlar, sınıflar*) bağımsız olarak test etmeyi amaçlar.
 - Genellikle geliştiriciler tarafından yazılım kodunun erken aşamalarında gerçekleştirilir.
 - Hata tespitini erken yapmak için genellikle otomatik test araçları (*JUnit, NUnit, pytest vb.*) kullanılır.
2. **Entegrasyon Testi (Integration Testing)**
 - Birbirine bağlı bileşenlerin birlikte düzgün çalışıp çalışmadığını kontrol eder.
 - API, veritabanı ve servis entegrasyonları gibi sistemin farklı bileşenleri test edilir.
 - Birbirinden bağımsız geliştirilen modüllerin başarılı bir şekilde iletişim kurduğundan emin olunur.
3. **Sistem Testi (System Testing)**
 - Yazılımın tüm bileşenlerinin entegre şekilde test edilmesini sağlar.
 - Kullanıcı gereksinimlerine, işlevselliğe ve performansa odaklanır.
 - Farklı senaryolar ve test teknikleri (*fonksiyonel test, performans testleri vb.*) uygulanır.
4. **Kabul Testi (Acceptance Testing)**
 - Yazılımın müşteri veya son kullanıcılar tarafından onaylanması için yapılan testtir.
 - Gerçek dünya senaryolarında yazılımın beklenen şekilde çalışıp çalışmadığı doğrulanır.
 - İki temel türü vardır:
 - Alfa Testi: Yazılım şirketi içinde kullanıcılar tarafından yapılan testler.
 - Beta Testi: Gerçek kullanıcılar tarafından yapılan testler.

Örnek Senaryo

Bir çevrimiçi uçak bileti rezervasyon sisteminin test sürecini düşünelim:

✓ Birim Testi:

- “Ödeme hesaplama fonksiyonu doğru çalışıyor mu?” gibi küçük bileşenler test edilir.
- Kodun en küçük parçaları bağımsız olarak değerlendirilir.

✓ Entegrasyon Testi:

- “Ödeme modülü, banka sistemine doğru verileri iletiyor mu?” gibi birden fazla bileşenin uyumu test edilir.
- Modüller arasında doğru veri akışının olup olmadığı kontrol edilir.

✓ Sistem Testi:

- “Kullanıcılar tüm rezervasyon sürecini sorunsuz tamamlayabiliyor mu?” gibi tüm sistemin bir bütün olarak çalıştığı test edilir.
- Yazılımın işlevselliği, performansı ve güvenliği değerlendirilir.

✓ Kabul Testi:

- “Gerçek kullanıcılar uygulamayı kullanırken herhangi bir sorunla karşılaşılıyor mu?” sorusuna yanıt aranarak müşteri gereksinimlerinin karşılandığı doğrulanır.
- Gerçek dünya koşullarında test yapılır ve yazılımın canlıya alınmaya hazır olup olmadığı belirlenir.

Bu seviyeler sayesinde:

- ◆ Yazılım, geliştirme aşamasından itibaren sistematik bir şekilde test edilir.
- ◆ Olası hatalar erken aşamalarda yakalanarak maliyetler düşürülür.
- ◆ Son kullanıcıların sorunsuz bir deneyim yaşaması sağlanır.

Pratik Sorular

1. Yazılım test seviyeleri nelerdir?
2. Birim testi ve entegrasyon testi arasındaki fark nedir?
3. Kabul testinin amacı nedir?
- 4.

Cevaplar

1. Birim testi, entegrasyon testi, sistem testi ve kabul testi.
2. Birim testi, bireysel bileşenleri test ederken entegrasyon testi, bu bileşenlerin birlikte çalışmasını doğrular.
3. Kabul testi, yazılımın müşteri gereksinimlerini karşıladığını doğrulamak için yapılır.

21. TEST TÜRLERİ

(TEST TYPES)

Test Türlerinin Önemi

Yazılım testleri, test edilen özelliklere ve yazılım geliştirme sürecindeki aşamalara bağlı olarak **farklı** türlere ayrılır.

Test türleri:

- Yazılımın işlevselliğini, güvenliğini, performansını ve kullanılabilirliğini değerlendirmeye yardımcı olur.
- Farklı test türleri kullanılarak yazılımın çeşitli yönleri test edilir ve olası hatalar en aza indirilir.

Temel Test Türleri

1. **Fonksiyonel Testler**
 - Yazılımın belirlenen gereksinimlere uygun çalışıp çalışmadığını değerlendirir.
 - Örnek: Giriş formunun doğru veri kabul edip etmediğini test etmek.
2. **Fonksiyonel Olmayan Testler**
 - Performans, güvenlik, kullanılabilirlik ve erişilebilirlik gibi kalite özelliklerini test eder.
 - Örnek: Web sitesinin yüksek trafik altında nasıl çalıştığını ölçmek.
3. **Yapısal (Beyaz Kutu) Testler**
 - Yazılımın iç yapısını ve kod kalitesini değerlendirir.
 - Örnek: Kodun tüm dallarının (if-else gibi) test edildiğini doğrulamak.
4. **Değişiklikle İlgili Testler**
 - Yazılımda yapılan değişikliklerin diğer özellikleri etkileyip etkilemediğini test eder.
 - Örnek: Regresyon testi ile, yeni bir güncellemenin eski fonksiyonları bozup bozmadığını kontrol etmek.

Örnek Senaryo

Bir e-ticaret sitesini test ettiğimizi düşünelim.

- ✦ **Fonksiyonel Test:** “Sepete ekle” butonunun doğru çalışıp çalışmadığını kontrol etmek.
- ✦ **Performans Testi:** Yoğun kullanıcı trafiğinde web sitesinin yavaşlayıp yavaşlamadığını test etmek.
- ✦ **Güvenlik Testi:** Kullanıcı şifrelerinin güvenli bir şekilde saklandığını doğrulamak.
- ✦ **Kullanılabilirlik Testi:** Kullanıcıların sipariş verme sürecini rahatça tamamlayıp tamamlayamadığını incelemek.
- ✦ **Regresyon Testi:** Sepet sistemine yeni bir özellik eklendiğinde, mevcut ödeme sisteminin hala sorunsuz çalıştığını kontrol etmek.

Bu test türleri, yazılımın işlevselliğini ve güvenilirliğini artırarak son kullanıcı deneyimini geliştirir.

Pratik Sorular

1. Yazılım test türleri nelerdir?
2. Fonksiyonel ve fonksiyonel olmayan testler arasındaki fark nedir?
3. Değişiklikle ilgili testler nelerdir?

Cevaplar

1. Fonksiyonel testler, fonksiyonel olmayan testler, yapısal testler ve değişikliklerle ilgili testler.
2. Fonksiyonel testler, yazılımın belirli işlevlerinin beklendiği gibi çalıştığını doğrularken, fonksiyonel olmayan testler güvenlik, performans gibi kalite özelliklerini değerlendirir.
3. Regresyon testi, yeniden test (re-testing) ve bakım testi gibi testleri içerir.



22. BAKIM TESTİ (MAINTENANCE TESTING)

Bakım Testinin Önemi

Bakım testi, yazılımın canlı sistemde çalışırken yapılan değişikliklerden etkilenip etkilenmediğini doğrulamak için gerçekleştirilir.

Bu testler:

- Yazılım güncellemeleri, hata düzeltmeleri veya yeni özellik eklemeleri sonrası yapılır.
- Yazılımın mevcut işlevlerinin değişikliklerden dolayı bozulmadığını doğrulamak için kritik öneme sahiptir.

Bakım testleri olmadan, yapılan değişiklikler istenmeyen hatalara yol açabilir ve sistemin kararlılığı bozulabilir.

Bakım Testi Türleri

- Hata Düzeltme Testi (Corrective Maintenance Testing)**
 - Yazılımda tespit edilen hataların düzeltilmesi sonrası yapılan testlerdir.
 - Örnek: Yanlış hata mesajı veren bir form düzeltildikten sonra, formun hatasız çalıştığı test edilir.
- Uyarılama Testi (Adaptive Maintenance Testing)**
 - Yazılımın yeni ortam veya sistemlere uyarlanmasını doğrulamak için gerçekleştirilir.
 - Örnek: Bir uygulamanın yeni bir işletim sistemi sürümünde çalışabilir olup olmadığı test edilir.
- Geliştirme Testi (Perfective Maintenance Testing)**
 - Mevcut yazılımın performansını veya kullanılabilirliğini iyileştirmek amacıyla yapılan testlerdir.
 - Örnek: Sayfa yüklenme süresi azaltıldıktan sonra, yeni hızın sistemde herhangi bir soruna yol açıp açmadığı test edilir.
- Önleyici Test (Preventive Maintenance Testing)**
 - Olası hataları önlemek ve sistemin gelecekte daha sağlam çalışmasını sağlamak için yapılan testlerdir.
 - Örnek: Gelecekte oluşabilecek güvenlik açıklarını önlemek için ek test senaryoları oluşturulur.

Örnek Senaryo

Bir bankacılık uygulamasının mobil sürümüne “Yüz Tanıma ile Giriş” özelliği eklendiğini düşünelim.

Bu durumda bakım testleri şu şekilde uygulanır:

- ✓ **Hata Düzeltme Testi:** Önceden raporlanan bir hata (örneğin, giriş ekranında yanlış hata mesajı gösterilmesi) düzeltildikten sonra, sistemin doğru çalıştığı doğrulanır.
- ✓ **Uyarlama Testi:** Uygulamanın yeni bir işletim sistemi sürümünde sorunsuz çalışıp çalışmadığı test edilir.
- ✓ **Geliştirme Testi:** Yüz tanıma giriş hızının artırılması sonrası, girişin sorunsuz çalıştığı kontrol edilir.
- ✓ **Önleyici Test:** Gelecekte oluşabilecek güvenlik açıklarını önlemek için ek test senaryoları uygulanır.

Bakım testleri sayesinde:

- ◆ Sistemin sürekliliği korunur.
- ◆ Yapılan değişikliklerin mevcut özellikleri bozmadığı garanti edilir.
- ◆ Son kullanıcı deneyimi iyileştirilir ve güvenlik sağlanır.

Pratik Sorular

1. Bakım testleri neden önemlidir?
2. Bakım testleri ne zaman gerçekleştirilir?
3. Regresyon testi ve bakım testi arasındaki fark nedir?

Cevaplar

1. Yazılımın değişikliklerden olumsuz etkilenmediğini doğrulamak için gereklidir.
2. Yazılım güncellemeleri, hata düzeltmeleri ve yeni özellikler eklendiğinde gerçekleştirilir.
3. Regresyon testi, tüm sistemin eski sürümlerle uyumlu olup olmadığını kontrol ederken, bakım testi değişikliklerin yazılımın genel işleyişine etkisini test eder.

23. STATİK TESTLER (STATIC TESTS)

Statik Testlerin Önemi

Statik testler, yazılımın çalıştırılmadan analiz edilmesini sağlayan test teknikleridir.

- Kod inceleme, belge inceleme ve statik analiz tekniklerini içerir.
- Yazılım geliştirme sürecinde erken aşamalarda hataların tespit edilmesine yardımcı olur.
- Geliştirme maliyetlerini düşürerek yazılımın daha güvenilir hale gelmesini sağlar.

Bu testler sayesinde, yazılım kodu daha yazılmadan veya yazım aşamasında hatalar belirlenebilir ve düzeltilir.

Statik Test Türleri

1. **Dokümantasyon İnceleme**
 - Yazılım gereksinimleri, tasarım belgeleri ve test planlarının doğruluğu ve tutarlılığı kontrol edilir.
 - Yanlış veya eksik gereksinimlerin önceden tespit edilmesini sağlar.
2. **Kod İnceleme**
 - Geliştiriciler tarafından yazılan kod manuel veya otomatik araçlar kullanılarak gözden geçirilir.
 - Kodun belirlenen standartlara uygun olup olmadığı değerlendirilir.
3. **Statik Kod Analizi**
 - Otomatik analiz araçları ile kod taranır ve potansiyel hatalar tespit edilir.
 - Güvenlik açıkları ve kötü kodlama pratikleri belirlenebilir.
 - Örnek araçlar: *SonarQube*, *Checkstyle*, *Lint*.
4. **Yürütmesiz Testler**
 - Kodun çalıştırılmadan mantıksal hatalarının ve eksikliklerinin tespit edilmesi amaçlanır.
 - Kodun belirli kurallara ve standartlara uygunluğu gözden geçirilir.

Örnek Senaryo

Bir e-ticaret sitesinin sipariş yönetim modülünü geliştirdiğimizi düşünelim.

Siparişlerin yanlış hesaplanmasını önlemek için statik testler şu şekilde uygulanabilir:

✓ Dokümantasyon İnceleme:

- Sipariş fiyatlandırma kurallarının gereksinim dokümanına uygun olup olmadığı kontrol edilir.

✓ Kod İnceleme:

- Geliştiriciler, ödeme hesaplama fonksiyonunun doğruluğunu manuel olarak gözden geçirir.

✓ Statik Kod Analizi:

- Kod analiz araçları kullanılarak güvenlik açıkları ve hatalar tespit edilir.

✓ Yürütmesiz Testler:

- Kodun belirlenen standartlara uygun olup olmadığı kontrol edilir.

Bu testler sayesinde:

- ◆ Yazılım geliştirme sürecinde erken aşamalarda hatalar tespit edilir ve düzeltilir.
- ◆ Geliştirme süreci daha verimli hale getirilir.
- ◆ Kod kalitesi artırılarak bakım maliyetleri düşürülür.

Pratik Sorular

1. Statik testlerin avantajları nelerdir?
2. Statik analiz ve kod inceleme arasındaki fark nedir?
3. Statik testler hangi aşamalarda kullanılır?

Cevaplar

1. Erken hata tespiti, geliştirme maliyetlerinin düşürülmesi ve kaliteyi artırma gibi avantajlara sahiptir.
2. Statik analiz, otomatik araçlarla kodu incelerken, kod inceleme manuel olarak ekip üyeleri tarafından yapılır.
3. Gereksinim analizi, tasarım aşaması ve kod yazımı sırasında uygulanabilir.

24. İNCELEMELER (REVIEWS)

İncelemelerin Önemi

İncelemeler, yazılım geliştirme sürecinde oluşturulan belgelerin, kodların ve diğer iş ürünlerinin hataları erken aşamada tespit etmek amacıyla gözden geçirilmesi sürecidir.

- Statik test tekniklerinden biri olan incelemeler, yazılımın çalıştırılmasına gerek kalmadan hata tespit edilmesini sağlar.
- Yazılım kalitesini artırmak, maliyetleri düşürmek ve geliştirme sürecini daha verimli hale getirmek için kullanılır.

Bu süreçler sayesinde, test aşamasına geçmeden önce hatalar belirlenerek erken aşamada düzeltilir.

İnceleme Türleri

1. **Gayri Resmi İnceleme (Informal Review)**
 - Resmi kurallar olmadan, ekip içindeki kişiler tarafından yapılan hızlı gözden geçirme sürecidir.
 - Belge veya kodun basit bir kontrolü ile hataların belirlenmesini sağlar.
2. **Eş Düzey İnceleme (Peer Review)**
 - Takım üyelerinin birbirlerinin çalışmalarını değerlendirdiği, yapılandırılmış bir inceleme türüdür.
 - Geliştiriciler ve test mühendisleri, birbirlerinin dokümanlarını veya kodlarını kontrol ederek eksiklikleri belirler.
3. **Teknik İnceleme (Technical Review)**
 - Yazılımın teknik gereksinimlere uygunluğunu kontrol etmek için teknik uzmanlar tarafından yapılan detaylı inceleme sürecidir.
 - Performans, güvenlik ve teknik standartlara uygunluk açısından değerlendirme yapılır.
4. **Denetim (Audit)**
 - Bağımsız bir grup tarafından yapılan inceleme olup, süreçlerin standartlara ve uyumluluk gerekliliklerine uygun olup olmadığını kontrol eder.
 - Şirket içi politikalar, düzenleyici standartlar veya sektör gereksinimleri doğrultusunda yapılır.
5. **Resmi İnceleme (Formal Review)**
 - Önceden belirlenmiş kurallara göre gerçekleştirilen ve inceleme sonucunda bir rapor oluşturulan yapılandırılmış inceleme sürecidir.
 - İnceleme sonuçları raporlanarak yönetime sunulur.

Örnek Senaryo

Bir mobil bankacılık uygulaması geliştirdiğimizi düşünelim.

Yeni bir ödeme özelliği eklenmiştir ve bu özelliğin güvenli, hatasız ve gereksinimlere uygun çalışmasını sağlamak için inceleme süreçleri şu şekilde uygulanır:

✓ Gayri Resmi İnceleme:

- Geliştiriciler, kendi aralarında kodu hızlıca gözden geçirir ve temel hataları düzeltir.

✓ Eş Düzey İnceleme:

- Geliştirici ekibi, birbirlerinin kodlarını kontrol ederek mantıksal hataları tespit eder.

✓ Teknik İnceleme:

- Güvenlik uzmanları, ödeme sisteminin güvenlik gereksinimlerini karşılayıp karşılamadığını değerlendirir.

✓ Denetim:

- Bağımsız bir ekip, uygulamanın şirketin kalite ve güvenlik standartlarına uygun olup olmadığını denetler.

✓ Resmi İnceleme:

- İnceleme sonuçları raporlanarak yönetim ekibine sunulur.

Bu süreçler sayesinde:

- ◆ Test aşamasına geçmeden önce büyük hatalar tespit edilerek düzeltilir.
- ◆ Daha güvenli ve hatasız bir yazılım geliştirilmiş olur.
- ◆ İlerleyen aşamalarda hata düzeltme maliyetleri düşürülür.

Pratik Sorular

1. İncelemelerin temel amacı nedir?
2. Teknik inceleme ile eş düzey inceleme arasındaki fark nedir?
3. Denetim (Audit) süreci neden bağımsız bir ekip tarafından yapılır?

Cevaplar

1. İncelemelerin temel amacı, yazılım geliştirme sürecinde erken aşamada hataları tespit etmek, maliyetleri düşürmek ve kaliteyi artırmaktır.
2. Teknik inceleme, uzmanlar tarafından detaylı teknik analiz içerirken, eş düzey inceleme ekip üyelerinin birbirlerinin çalışmalarını değerlendirdiği daha genel bir süreçtir.
3. Denetim süreci bağımsız bir ekip tarafından yapılır, çünkü objektif bir değerlendirme sağlamak ve standartlara uyumluluğu güvence altına almak gereklidir.

25. ROLLER VE SORUMLULUKLAR

(ROLES AND RESPONSIBILITIES)

Yazılım Test Sürecindeki Roller ve Sorumlulukların Önemi

Yazılım testi sürecinde, farklı rollerin belirlenmesi ve her bir rolün sorumluluklarının açıkça tanımlanması, test sürecinin başarılı bir şekilde yürütülmesi için kritik öneme sahiptir.

- Her rol, yazılımın hatasız ve kaliteli olmasını sağlamak için belirli görevleri yerine getirir.
- Test süreçlerinin etkinliği, doğru kişiler tarafından yürütülen doğru görevlerle doğrudan ilişkilidir.

Temel Test Roller

1. **Test Yöneticisi (Test Manager)**
 - Test sürecinin genel planlamasını yapar, kaynakları yönetir ve test stratejilerini belirler.
 - Test takvimi oluşturur ve testlerin zamanında tamamlanmasını sağlar.
 - Test ekibinin performansını izler ve yönetime test süreçleri hakkında geri bildirim verir.
2. **Test Analisti (Test Analyst)**
 - Test senaryolarını ve test vakalarını hazırlar.
 - Manuel veya otomatik testleri uygular.
 - Test sonuçlarını değerlendirir ve hataları raporlar.
3. **Test Otomasyon Mühendisi (Test Automation Engineer)**
 - Test süreçlerini otomatikleştirmek için yazılım test araçlarını kullanır.
 - Yinelenen test süreçlerini hızlandırarak manuel test yükünü azaltır.
 - Test scriptleri yazar ve çalıştırarak yazılımın stabilitesini sürekli olarak kontrol eder.
4. **Geliştirici (Developer)**
 - Kod yazım aşamasında birim testlerini gerçekleştirir.
 - Test ekipleriyle iş birliği yaparak tespit edilen hataları düzeltir.
 - Test uzmanlarından gelen hata raporlarına göre kod güncellemelerini gerçekleştirir.
5. **İş Analisti (Business Analyst)**
 - Test sürecinin yazılım gereksinimlerine uygun olup olmadığını kontrol eder.
 - İş kurallarına uygunluğun sağlanmasını denetler.
6. **Kullanıcı Kabul Testi Uzmanı (User Acceptance Tester - UAT Tester)**
 - Son kullanıcı bakış açısıyla testleri gerçekleştirir.
 - Yazılımın müşteri beklentilerini karşılayıp karşılamadığını doğrular.

Örnek Senaryo

Bir mobil bankacılık uygulamasının test sürecini ele alalım:

✓ Test Yöneticisi:

- Test sürecini planlar, test stratejisini belirler ve kaynakları organize eder.

✓ **Test Analisti:**

- Kullanıcı giriş, para transferi ve hesap hareketleri gibi senaryolar için test vakaları oluşturur.

✓ **Test Otomasyon Mühendisi:**

- Para transferi işlemlerinin otomatik testlerini yazar ve yürütür.

✓ **Geliştirici:**

- Yeni bir özellik eklenmeden önce birim testleri uygular ve hataları düzeltir.

✓ **İş Analisti:**

- Bankacılık kurallarına uygunluk açısından testlerin doğruluğunu kontrol eder.

✓ **Kullanıcı Kabul Testi Uzmanı:**

- Son kullanıcı gözüyle uygulamanın beklentileri karşılayıp karşılamadığını değerlendirir.

Bu rollerin belirlenmesi ve etkin bir şekilde yürütülmesi sayesinde:

- ◆ Test süreçleri verimli ve eksiksiz bir şekilde tamamlanır.
- ◆ Yazılımın hem teknik hem de iş gereksinimlerine uygunluğu doğrulanır.
- ◆ Son kullanıcı deneyimi en iyi hale getirilerek kaliteli bir ürün sunulur.

Pratik Sorular

1. Test yöneticisinin başlıca görevleri nelerdir?
2. Test analistinin test sürecindeki rolü nedir?
3. Test otomasyon mühendisinin manuel testlerden farkı nedir?

Cevaplar

1. Test yöneticisi, test süreçlerini planlar, kaynakları yönetir, test stratejisini belirler ve test ekibinin performansını izler.
2. Test analisti, test senaryolarını oluşturur, test vakalarını hazırlar, manuel testleri yürütür ve tespit edilen hataları raporlar.
3. Test otomasyon mühendisi, test süreçlerini otomatikleştirerek tekrar eden testleri hızlandırır ve manuel test yükünü azaltır, böylece test süreçleri daha verimli hale gelir.

26. STATİK KOD ANALİZİ (STATIC CODE ANALYSIS)

Statik Kod Analizinin Önemi

Statik kod analizi, yazılımın **çalıştırılmadan**, kaynak kodunun belirli standartlara uygunluğunu değerlendiren bir test tekniğidir.

- Yazılım geliştirme sürecinde hataları erken aşamada tespit etmek için kullanılır.
- Statik analiz araçları, kodda bulunan güvenlik açıklarını, performans sorunlarını ve kodlama hatalarını otomatik olarak tespit etmeye yardımcı olur.
- Bu sayede yazılımın kalitesi artırılır ve geliştirme maliyetleri düşürülür.

Statik Kod Analizinin Avantajları

✓ Erken Hata Tespiti:

- Yazılım çalıştırılmadan önce kod hatalarının bulunmasını sağlar.

✓ Kod Kalitesini Artırır:

- Kodun belirli standartlara uygun olmasını sağlar ve kod okunabilirliğini artırır.

✓ Güvenlik Açıklarının Azaltır:

- Güvenlik açıklarını erken aşamada tespit ederek olası siber saldırılara karşı koruma sağlar.

✓ Daha Az Manuel Test İhtiyacı:

- Geliştiricilerin manuel test yükünü azaltarak zaman kazandırır.

Statik Kod Analiz Araçları

Statik kod analizi için kullanılan popüler araçlardan bazıları şunlardır:

◆ SonarQube:

- Kod kalitesini analiz etmek ve güvenlik açıklarını tespit etmek için kullanılan güçlü bir araçtır.

◆ Checkstyle:

- Java projelerinde kod stilini analiz etmek için kullanılır.

◆ PMD:

- Kod hatalarını ve potansiyel sorunları belirlemek için kullanılan açık kaynaklı bir araçtır.

◆ FindBugs:

- Java kodundaki hataları tespit eden bir analiz aracıdır.

Örnek Senaryo

Bir bankacılık uygulamasında kullanıcı giriş bilgilerini işleyen bir yazılım modülü geliştirdiğimizi düşünelim.

Statik kod analizi şu şekilde uygulanabilir:

- ✓ **SonarQube** kullanılarak analiz yapılır ve kodda güvenlik açıkları aranır.
- ✓ **Checkstyle** ile kodun belirlenen stil rehberine uygun olup olmadığı kontrol edilir.
- ✓ **PMD** kullanılarak gereksiz kod blokları ve performans sorunları tespit edilir.
- ✓ **Geliştiriciler**, analiz sonuçlarına göre kodlarını revize eder ve güvenlik açıklarını kapatır.

Bu süreç sayesinde:

- ◆ Yazılımın kalitesi artırılır.
- ◆ Güvenlik açıkları minimize edilir.
- ◆ Geliştirme maliyetleri düşürülerek daha verimli bir yazılım geliştirme süreci sağlanır.

Pratik Sorular

1. Statik kod analizi neden önemlidir?
2. Statik kod analizi ile dinamik testler arasındaki temel fark nedir?
3. Hangi popüler araçlar statik kod analizi için kullanılır?

Cevaplar

1. Statik kod analizi, yazılım çalıştırılmadan önce kod hatalarını tespit etmeye ve kod kalitesini artırmaya yardımcı olur. Bu sayede hatalar erken aşamada düzeltilir ve maliyetler düşürülür.
2. Statik kod analizi, yazılımın çalıştırılmadan incelenmesi sürecidir, dinamik testler ise yazılımın çalıştırılarak hata tespit edilmesini içerir. Statik analiz daha erken aşamada yapılırken, dinamik testler yazılımın çalıştırıldığı ortamda gerçekleştirilir.
3. Statik kod analizi için kullanılan popüler araçlar arasında SonarQube, Checkstyle, PMD ve FindBugs bulunmaktadır.

27. TEST TASARIM TEKNİKLERİ

(TEST DESIGN TECHNIQUES)

Test Tasarım Tekniklerinin Önemi

Test tasarım teknikleri, yazılımın belirlenen gereksinimlere uygun olup olmadığını doğrulamak için test senaryolarını oluşturma sürecinde kullanılan yöntemlerdir.

- Test süreçlerini daha sistematik hale getirerek test kapsamını artırır.
- Farklı test teknikleri, yazılımın farklı açılardan değerlendirilmesini sağlar.
- Hata yakalama oranını artırarak yazılımın daha kaliteli olmasına yardımcı olur.

Test Tasarım Teknikleri Türleri

1. Kara Kutu Test Teknikleri (Black Box Testing)

Yazılımın iç yapısını bilmeden, sadece girdiler ve çıktılar değerlendirilerek yapılan testlerdir.

✓ Eşdeğer Bölümleme (Equivalence Partitioning)

- Girdi değerlerini benzer özelliklere sahip bölümlere ayırarak test etme yöntemidir.
- **Örnek:** Yaş alanı için 0-17 (çocuk), 18-64 (yetişkin), 65+ (yaşlı) şeklinde bölümlendirme yapılarak test senaryoları oluşturulur.

✓ Sınır Değer Analizi (Boundary Value Analysis)

- En yüksek ve en düşük sınır değerlerinin test edilmesine odaklanır.
- **Örnek:** Bir sistemde geçerli giriş sayısı 1 ile 100 arasındaysa, testler 0, 1, 100 ve 101 değerleriyle yapılır.

✓ Karar Tablosu Testi (Decision Table Testing)

- Koşullar ve çıktılar arasındaki mantıksal ilişkileri analiz ederek test senaryoları oluşturur.
- **Örnek:** Ödeme ekranında, farklı kart türleri ve limit durumları için doğrulamalar yapılır.

2. Beyaz Kutu Test Teknikleri (White Box Testing)

Yazılımın iç yapısını ve kod akışını baz alarak gerçekleştirilen testlerdir.

✓ Akış Diyagramı Analizi (Control Flow Testing)

- Kodun kontrol akışını doğrulamak için yapılan testtir.
- **Örnek:** Bir if-else koşulunun her iki durumunun test edilmesi.

✓ Birim Testi (Unit Testing)

- **Kodun en küçük bileşenlerini bağımsız olarak test etme sürecidir.**
 - **Örnek:** Bir hesaplama fonksiyonunun her girdiye karşı doğru sonucu verdiğini kontrol etmek.
-

3. Deneyime Dayalı Test Teknikleri (Experience-Based Testing)

Test uzmanlarının geçmiş deneyimlerine dayanarak test senaryolarını belirlemesiyle gerçekleştirilir.

✓ Hata Tahmini (Error Guessing)

- Test uzmanlarının önceki projelerden edindikleri tecrübeler doğrultusunda hata tahmini yaparak test vakaları oluşturmalarıdır.
- **Örnek:** Daha önce benzer sistemlerde karşılaşılan hatalara göre özel test senaryoları oluşturulması.

✓ Keşif Testi (Exploratory Testing)

- Önceden tanımlı test senaryoları olmadan, sistemin farklı açılardan test edilmesi sürecidir.
 - **Örnek:** Bir web sitesinde, kullanıcı gibi dolaşarak beklenmeyen hataları keşfetmek.
-

Örnek Senaryo

Bir e-ticaret platformunda sipariş verme sürecini test ettiğimizi düşünelim.

✦ Eşdeğer Bölümleme:

- Ödeme ekranında geçerli ve geçersiz kredi kartı numaraları farklı kategorilere ayrılarak test edilir.

✦ Sınır Değer Analizi:

- Ücretsiz kargo sınırı 500 TL ise, 499 TL ve 500 TL'lik siparişlerle test gerçekleştirilir.

✦ Keşif Testi:

- Kullanıcı senaryoları öngörülmeden, sistemin nasıl davrandığını görmek için test uzmanı tarafından doğrudan testler yapılır.

Bu yöntemler sayesinde:

- ✦ Yazılımın farklı açılardan test edilmesi sağlanır.
 - ✦ Olası hatalar daha erken aşamada tespit edilir.
 - ✦ Sistematik ve kapsamlı test süreçleri uygulanır.
-

Pratik Sorular

1. **Kara Kutu Testi ile Beyaz Kutu Testi arasındaki temel fark nedir?**
2. **Sınır Değer Analizi hangi durumlarda kullanılır?**
3. **Keşif Testi neden önemlidir?**

Cevaplar

1. **Kara Kutu Testi**, yazılımın iç yapısını bilmeden dışsal davranışlarını test ederken, **Beyaz Kutu Testi** yazılımın iç kod akışına odaklanarak gerçekleştirilen bir test türüdür.
2. **Sınır Değer Analizi**, sistemin kritik eşik noktalarında nasıl çalıştığını test etmek için kullanılır. Örneğin, maksimum veya minimum giriş değerlerinin kontrol edilmesi gereken senaryolarda uygulanır.
3. **Keşif Testi**, önceden belirlenmiş test senaryoları olmadan sistemin beklenmeyen durumlarını keşfetmek için kullanılır. Böylece, test uzmanı manuel olarak test yaparak sistemdeki potansiyel hataları tahmin edebilir.



28. RİSK TEMELLİ TEST YAKLAŞIMLARI (RISK-BASED TESTING APPROACHES)

Risk Temelli Test Nedir?

Risk temelli test (Risk-Based Testing - RBT), test faaliyetlerinin en kritik ve yüksek risk taşıyan alanlara odaklanmasını sağlayan bir yaklaşımdır.

- Yazılım geliştirme sürecinde her modül aynı öneme sahip değildir.
- Bazı bileşenlerin hata yapma ihtimali daha yüksek ve yazılımın genel işleyişine etkisi daha büyük olabilir.
- Bu nedenle, riskleri değerlendirerek test süreçlerini önceliklendirmek, yazılım kalitesini artırmada etkili bir yöntemdir.

Risk Değerlendirme Kriterleri

✓ Hata Olasılığı

- Daha önce hata barındıran veya teknik olarak karmaşık bileşenler yüksek risk taşır.

✓ İşletme Etkisi

- Kullanıcı deneyimine doğrudan etki eden veya kritik iş süreçlerini barındıran bölümler öncelikli test edilmelidir.

✓ Uyumluluk ve Entegrasyon Riskleri

- Farklı sistemlerle entegrasyon içeren bölümler daha fazla test edilmelidir.

✓ Güvenlik ve Performans Gereksinimleri

- Hassas verilerin işlendiği veya yüksek kullanıcı trafiği beklenen alanlarda daha kapsamlı testler gereklidir.

Risk Temelli Testin Avantajları

✦ Önceliklendirme Sağlar

- En kritik fonksiyonlar için daha fazla test kaynağı ayrılmasını sağlar.

✦ Test Kapsamını Optimize Eder

- Gereksiz testleri azaltarak zaman ve maliyet tasarrufu sağlar.

✦ Kullanıcı Deneyimini Artırır

- Kullanıcıya en çok etki eden alanlara odaklanarak müşteri memnuniyetini yükseltir.

Risk Temelli Test Süreci

1 Riskleri Belirleme

- Yazılımın en kritik bileşenleri belirlenir.

2 Risklerin Değerlendirilmesi

- Hata olasılığı ve işletme etkisi göz önünde bulundurularak risk seviyeleri belirlenir.

3 Test Planı Hazırlama

- Risk seviyesine göre test öncelikleri belirlenir.

4 Testlerin Yürütülmesi

- En yüksek riskli bileşenler öncelikli test edilir.

5 Sürekli Gözden Geçirme

- Yazılım geliştikçe risk analizleri güncellenir ve test kapsamı optimize edilir.

Örnek Senaryo

Bir banka mobil uygulaması için risk temelli test sürecini ele alalım:

✦ Risk Analizi:

- Kullanıcı giriş ekranı, para transferi işlemleri ve şifre sıfırlama gibi kritik işlemler en yüksek risk grubuna alınır.

✦ Test Önceliklendirme:

- Hata olasılığı yüksek ve güvenlik açısından kritik alanlar öncelikli olarak test edilir.

✦ Gerçekleştirme:

- Kullanıcı girişi ve ödeme işlemleri için penetrasyon testleri ve yük testleri uygulanarak sistemin güvenilirliği sağlanır.

Bu süreç sayesinde:

- ◆ En kritik sistemler öncelikli olarak test edilerek büyük hataların üretime çıkmadan önce bulunması sağlanır.
- ◆ Zaman ve maliyet tasarrufu yapılarak en önemli bölümlere daha fazla odaklanılır.
- ◆ Yazılımın genel güvenilirliği ve performansı artırılır.

Pratik Sorular

1. Risk temelli testin en büyük avantajı nedir?
2. Test sürecinde riskler nasıl belirlenir?
3. Risk seviyesi yüksek olan bileşenler hangi kriterlere göre önceliklendirilir?

Cevaplar

1. Risk temelli test, en kritik fonksiyonlara öncelik vererek yazılımın güvenilirliğini artırır ve test kaynaklarının verimli kullanılmasını sağlar.
2. Yazılım bileşenlerinin hata olasılığı, işlevsellik, kullanıcı etkisi, entegrasyon noktaları ve güvenlik gibi faktörler analiz edilerek belirlenir.
3. Hata geçmişi, kullanıcıya etkisi, entegrasyon karmaşıklığı ve güvenlik gereksinimleri göz önünde bulundurularak test öncelikleri belirlenir.

29. TEST YÖNETİMİ

(TEST MANAGEMENT)

Test Yönetimi Nedir?

Test yönetimi, yazılım test süreçlerinin planlanması, uygulanması, izlenmesi ve kontrol edilmesi süreçlerini kapsayan bir disiplindir.

- Başarılı bir test yönetimi, testlerin etkin bir şekilde yürütülmesini sağlar.
- Yazılımın kalite güvencesini artırarak hataların erken tespit edilmesine yardımcı olur.
- Zaman ve maliyet tasarrufu sağlarken, kullanıcı deneyimini iyileştirmeye katkıda bulunur.

Test Yönetiminin Ana Bileşenleri

✓ Test Planlama

- Test sürecinin kapsamı, hedefleri ve stratejileri belirlenir.
- Hangi modüllerin test edileceği, test ortamları ve test araçları tanımlanır.

✓ Test İzleme ve Kontrol

- Testlerin ilerleyişi düzenli olarak izlenir.
- Planlanan test süreçlerinde gecikme veya sorunlar varsa, stratejiler güncellenir.

✓ Test Raporlama

- Test süreçleri hakkında yönetime bilgi veren detaylı raporlar hazırlanır.
- Hata oranları, test başarısı ve tamamlanan test senaryoları gibi veriler analiz edilir.

✓ Test Kaynak Yönetimi

- Test ekipleri, test ortamları ve test araçlarının yönetilmesini içerir.
- Doğru kaynakların tahsis edilmesi, test süreçlerinin verimli yürütülmesi açısından kritik öneme sahiptir.

✓ Risk Yönetimi

- Yazılımın kritik alanları belirlenerek test öncelikleri oluşturulur.
- Yüksek riskli alanlara daha fazla test kaynağı ayrılarak olası büyük hatalar önlenir.

Test Yönetiminin Önemi

✦ Sistematik Süreçler

- Test süreçlerinin belirli bir metodolojiye göre yürütülmesini sağlar.

✦ Zaman ve Maliyet Tasarrufu

- Kaynakların verimli kullanımına yardımcı olur.

✦ Daha Kaliteli Yazılım

- Planlı test süreçleri, yazılımın hatalardan arındırılmasına katkı sağlar.

Örnek Senaryo

Bir online yemek sipariş uygulaması geliştirdiğimizi düşünelim.

Bu sistemin test yönetimi şu adımlarla gerçekleştirilebilir:

✦ Test Planlama:

- Kullanıcı kayıt, sipariş verme ve ödeme süreçlerini kapsayan test senaryoları belirlenir.

✦ Test İzleme:

- Gerçekleştirilen testlerin başarı oranları düzenli olarak takip edilir.

✦ Risk Yönetimi:

- Ödeme sistemleri gibi kritik işlevler için ekstra güvenlik testleri uygulanır.

Bu süreçler sayesinde:

- ✦ Yemek sipariş uygulaması kullanıcıya sunulmadan önce hatasız ve güvenilir bir şekilde çalıştığından emin olunur.
 - ✦ En kritik alanlar test edilerek müşteri memnuniyeti artırılır.
 - ✦ Test süreçleri optimize edilerek zaman ve maliyet tasarrufu sağlanır.
-

Pratik Sorular

1. Test yönetiminin en önemli bileşenlerinden biri nedir ve neden önemlidir?
2. Risk yönetimi neden test sürecinde kritik bir rol oynar?
3. Test planlaması sırasında hangi temel unsurlar belirlenmelidir?

Cevaplar

1. Test planlama, test yönetiminin en önemli bileşenlerinden biridir. Testlerin kapsamı, hedefleri ve stratejileri belirlenerek test sürecinin sistematik bir şekilde yürütülmesi sağlanır.
2. Risk yönetimi, yazılımın en kritik bölümlerine öncelik verilmesini sağlar. Böylece en büyük etkiye sahip olabilecek hatalar öncelikli olarak test edilir ve düzeltilir.
3. Test planlamasında test kapsamı, test stratejisi, kaynak tahsisi, risk analizi ve zaman çizelgesi gibi temel unsurlar belirlenmelidir.



30. TEST PLANLAMA VE TAHMİNLEME

(TEST PLANNING AND ESTIMATION)

Test Planlama ve Tahminleme Nedir?

Test planlama, yazılım test sürecinin kapsamını, hedeflerini, stratejisini ve kaynaklarını belirleme sürecidir.

Test tahminleme, test sürecinin ne kadar süreceğini, kaç kaynak gerektirdiğini ve hangi maliyetlerle gerçekleştirileceğini öngörmeyi içerir.

- Başarılı bir test planlama ve tahminleme süreci, proje yönetimi açısından kritik öneme sahiptir.
- Test süreçlerinin zamanında ve verimli bir şekilde tamamlanmasını sağlar.
- Test kapsamı ve kaynakların doğru belirlenmesi, maliyetlerin kontrol altında tutulmasına yardımcı olur.

Test Planlamanın Ana Bileşenleri

✓ Test Kapsamı

- Hangi fonksiyonların ve bileşenlerin test edileceği belirlenir.
- Kapsam dışı alanlar da tanımlanarak test süreçleri netleştirilir.

✓ Test Stratejisi

- Kullanılacak test teknikleri ve araçları belirlenir.
- Manuel veya otomatik testlerin hangi aşamalarda kullanılacağı tanımlanır.

✓ Kaynak Planlaması

- Test süreci için gerekli insan gücü, test ortamları ve test araçları belirlenir.
- Test ekiplerinin görev dağılımı yapılır.

✓ Risk Yönetimi

- Yazılımın en kritik bileşenleri belirlenerek test öncelikleri oluşturulur.
- Risk bazlı test teknikleri kullanılarak en önemli alanlara daha fazla odaklanılır.

✓ Zaman Çizelgesi

- Testlerin hangi zaman diliminde tamamlanacağı belirlenir.
- Test aşamalarına uygun bir takvim oluşturulur.

Test Tahminleme Yöntemleri

✦ Deneyime Dayalı Tahminleme (Experience-Based Estimation)

- Önceki projelerden edinilen bilgilere dayanarak test süresi ve kaynak ihtiyacı belirlenir.
- Daha önce benzer projelerde harcanan zaman ve kaynaklar temel alınarak tahmin yapılır.

✦ Analitik Tahminleme (Analytical Estimation)

- Test süreçleri için matematiksel modelleme ve metrikler kullanılarak tahmin yapılır.
- Örneğin, bir modül için yapılan ortalama test süresi hesaplanarak tüm test süresi tahmin edilir.

✦ Üstten Aşağıya ve Altan Yukarıya Yaklaşım (Top-Down & Bottom-Up Estimation)

✓ Üstten Aşağıya Yaklaşım (Top-Down Approach):

- Projenin genel test süresi belirlenir ve daha sonra detaylandırılır.
- Test süresi, projenin genel zaman çizelgesine uygun şekilde hesaplanır.

✓ Altan Yukarıya Yaklaşım (Bottom-Up Approach):

- Her test aktivitesi için ayrı ayrı tahmin yapılır.
- Tüm aktivitelerin toplam süresi hesaplanarak genel test süresi belirlenir.

Örnek Senaryo

Bir e-ticaret platformunun test sürecini planladığımızı düşünelim.

Aşağıdaki adımlarla test planlama ve tahminleme sürecini gerçekleştirebiliriz:

✦ Test Kapsamı:

- Ürün listeleme, ödeme işlemleri ve kullanıcı giriş sistemleri test edilecek.

✦ Test Stratejisi:

- Hem manuel hem de otomasyon testleri uygulanacak.

✦ Kaynak Planlaması:

- 3 test mühendisi, 1 test yöneticisi ve 2 otomasyon test uzmanı atanacak.

✦ Risk Yönetimi:

- Ödeme sistemi en kritik bileşen olarak belirlenerek, kapsamlı güvenlik testleri yapılacaktır.

✦ Tahminleme:

- Geçmiş projelere dayanarak test sürecinin 6 hafta süreceği ve 500 saat iş gücü gerektireceği öngörülmüştür.

Bu süreçler sayesinde:

- ✦ Test süreci daha verimli ve sistematik bir şekilde yönetilir.
- ✦ Kaynak kullanımı optimize edilerek maliyetler kontrol altında tutulur.
- ✦ Test süresinin daha doğru tahmin edilmesi sağlanarak proje zamanlaması etkili bir şekilde yönetilir.

Pratik Sorular

1. Test planlamasında en kritik bileşenlerden biri nedir ve neden önemlidir?
2. Deneyime dayalı test tahminleme nasıl yapılır?
3. Üstten aşağıya ve alttan yukarıya test tahminleme arasındaki fark nedir?

Cevaplar

1. Test kapsamı, test planlamasının en kritik bileşenlerinden biridir. Hangi fonksiyonların ve bileşenlerin test edileceğini belirler ve kapsam dışı alanları tanımlar. Bu, test süreçlerinin yönlendirilmesi açısından büyük önem taşır.
2. Deneyime dayalı test tahminleme, geçmiş projelerden elde edilen bilgiler kullanılarak test sürecinin süresi ve kaynak ihtiyacının öngörülmesi sürecidir. Daha önce benzer projelerde harcanan zaman ve kaynaklar temel alınarak tahmin yapılır.
3. Üstten aşağıya tahminleme, projenin genel test süresinin belirlenmesiyle başlar ve daha sonra detaylandırılır. Altan yukarıya tahminleme ise her bir test aktivitesi için ayrı ayrı tahmin yapılmasını ve sonunda tüm aktivitelerin toplam süresinin hesaplanmasını içerir.

31. TEST SÜRECİNİN TAKİBİ VE KONTROLÜ

(TEST PROGRESS SURVEILLANCE AND CONTROL)

Test Sürecinin Takibi ve Kontrolü Nedir?

Test sürecinin takibi ve kontrolü, test süreçlerinin ilerleyişini izleme, belirlenen hedeflere ulaşıp ulaşılmadığını değerlendirme ve gerektiğinde düzeltici önlemler almayı içeren kritik bir test yönetimi faaliyetidir.

- Testlerin zamanında ve etkin bir şekilde yürütülmesini sağlar.
- Test sürecinin durumunu izleyerek proje yöneticilerine ve paydaşlara doğru bilgiler sunar.
- Olası riskleri ve gecikmeleri erken tespit ederek, iyileştirici aksiyonların alınmasını sağlar.

Test Takibi ve Kontrolünün Ana Bileşenleri

✓ Test Metriği Belirleme

- Test ilerlemesini ölçmek için kullanılan metrikler tanımlanır.
- Örneğin: Başarıyla geçen test vakalarının oranı, hataların bulunma ve düzeltilme süresi.

✓ Test Raporlama

- Test sürecinin ilerleyişi hakkında detaylı raporlar oluşturulur.
- Hataların tespit edilme oranı, düzeltilme süresi gibi kritik veriler takip edilir.

✓ Risk Yönetimi

- Kritik hataların zamanında tespit edilmesini sağlamak için risk bazlı test yaklaşımları uygulanır.
- Örneğin, ödeme sistemleri veya güvenlik ile ilgili alanlar öncelikli olarak test edilir.

✓ Test Sürecinin Güncellenmesi

- Test süreçleri sürekli olarak iyileştirilir ve gerektiğinde test stratejisi güncellenir.
- Yeni özellikler veya müşteri geri bildirimleri doğrultusunda test planları revize edilir.

Test Takibi ve Kontrolünün Önemi

✦ Şeffaflık Sağlar

- Test sürecinin şeffaflığını artırır ve yönetime doğru bilgiler sunar.

✦ Kaynakların Daha Verimli Yönetilmesini Sağlar

- Test ekiplerinin performansını artırır ve test süreçlerini daha etkin hale getirir.

✦ Olası Gecikmeleri ve Riskleri Erken Aşamada Tespit Eder

- Projede yaşanabilecek gecikmeleri önceden tahmin ederek düzeltici önlemler alınmasını sağlar.

Örnek Senaryo

Bir online uçak bileti rezervasyon sistemi için test sürecini ele alalım.

Test takibi ve kontrolü şu adımlarla yürütülebilir:

✦ Test Metriği Belirleme:

- Hata tespit edilen ve düzeltilen test vakalarının yüzdesi hesaplanır.
- Örneğin, %85 başarı oranına ulaşılması hedeflenebilir.

✦ Test Raporlama:

- Test ilerleyişi haftalık raporlarla yönetime sunulur.
- Hata trendleri, başarısız test oranları raporlanarak yönetime sunulur.

✦ Risk Yönetimi:

- Kullanıcıların ödeme işlemi sırasında yaşadığı olası hatalar, öncelikli olarak test edilir ve çözüme kavuşturulur.

✦ Test Sürecinin Güncellenmesi:

- Kullanıcı geri bildirimleri doğrultusunda test senaryoları güncellenerek yeni testler planlanır.
- Örneğin, mobil cihazlarda sık yaşanan bir hata için ek test senaryoları oluşturulur.

Bu süreçler sayesinde:

- ✦ Sistem daha güvenli ve sorunsuz çalışır.
- ✦ Test ekibi daha verimli çalışarak kaynak yönetimi daha etkili yapılır.
- ✦ Test süreçleri daha ölçülebilir ve optimize edilebilir hale gelir.

Pratik Sorular

1. Test sürecinin takibi ve kontrolünün en önemli bileşeni nedir ve neden önemlidir?
2. Test metrikleri neden gereklidir ve hangi metrikler kullanılabilir?
3. Test sürecinde risk yönetiminin rolü nedir?

Cevaplar

1. Test metrikleri belirlemek, test sürecinin takibi için en önemli bileşenlerden biridir. Testlerin ilerleyişini ölçmek ve gerektiğinde süreçleri optimize etmek için kritik rol oynar.
2. Test metrikleri, test sürecinin ne kadar etkili olduğunu anlamak için gereklidir. Kullanılabilecek metrikler arasında, tamamlanan test vakalarının yüzdesi, hata düzeltme süresi ve başarısız test oranı gibi ölçümler bulunur.
3. Risk yönetimi, test sürecinde kritik öneme sahip hataların önceliklendirilmesini sağlar. Örneğin, yüksek etkiye sahip bir hatanın erken aşamada tespit edilmesi, son kullanıcı deneyimini korumak için gereklidir.



32. DENEYİME DAYALI TEST TASARIM TEKNİKLERİ (EXPERIENTIAL BASED TEST DESIGN TECHNIQUES)

Deneyime Dayalı Test Tasarım Teknikleri Nedir?

Deneyime dayalı test tasarım teknikleri, test uzmanlarının geçmiş deneyimlerinden yararlanarak test senaryoları oluşturmalarını içeren bir yaklaşımdır.

- Bu teknikler, yazılımın potansiyel zayıf noktalarını tahmin etmeye yardımcı olur.
- Geçmiş projelerde karşılaşılan hatalara dayanarak daha etkili testler gerçekleştirilmesini sağlar.
- Özellikle planlanmamış veya bilinmeyen senaryoların keşfedilmesine yardımcı olur.

Deneyime Dayalı Test Teknikleri Türleri

✓ Hata Tahmini (Error Guessing)

- Test uzmanlarının önceki projelerden edindikleri deneyimlere dayanarak en olası hataları tahmin etmesi.
- **Örnek:** Bir form alanına yanlış formatta veri girildiğinde sistemin nasıl tepki verdiğini test etmek.

✓ Keşif Testi (Exploratory Testing)

- Önceden belirlenmiş test senaryoları olmadan, test uzmanlarının yazılımı serbest bir şekilde test ederek hata bulmaya çalışması.
- **Örnek:** Bir e-ticaret sitesinde sipariş verme sürecinin beklenmedik senaryolarla nasıl çalıştığını manuel olarak incelemek.

✓ Check List Kullanımı (Checklist-Based Testing)

- Önceki testlerden elde edilen bilgilerin bir kontrol listesi halinde toplanarak, yeni projelerde test senaryolarının belirlenmesi.
- **Örnek:** Geçmiş projelerde yaşanan güvenlik açıklarının tekrar edilmediğini kontrol etmek için bir kontrol listesi oluşturmak.

Deneyime Dayalı Testlerin Avantajları

✦ Esneklik

- Önceden belirlenmiş test senaryolarına bağlı kalmadan yazılımın farklı yönleri keşfedilebilir.

✦ Hızlı Sonuçlar

- Kapsamlı planlama gerektirmeden hızlı bir şekilde test yapılabilir.

✦ Gerçek Kullanıcı Davranışlarını Simüle Etme

- Kullanıcıların karşılaşabileceği potansiyel sorunlar, test uzmanlarının deneyimleriyle tespit edilebilir.

Örnek Senaryo

Bir mobil bankacılık uygulamasının test sürecini düşünelim.

✦ Hata Tahmini:

- Kullanıcıların yanlış şifre girdiğinde sistemin nasıl tepki verdiği test edilir.
- Yanlış şifre girişlerinin kaç kere yapılabilceği ve bloke sürecinin doğru çalışıp çalışmadığı kontrol edilir.

✦ Keşif Testi:

- Farklı cihaz ve internet hızlarında mobil uygulamanın nasıl çalıştığını anlamak için test uzmanı manuel olarak farklı senaryoları dener.
- Örneğin, uygulama zayıf internet bağlantısında nasıl davranıyor? Yavaş tepki süresi kullanıcı deneyimini etkiliyor mu?

✦ Check List Kullanımı:

- Önceki projelerde yaşanan güvenlik açıklarının tekrar edilmediğini kontrol etmek için bir kontrol listesi oluşturulur.
- Örneğin, kullanıcı oturumu zaman aşımına uğradığında, sistemin doğru şekilde çıkış yapıp yapmadığı test edilir.

Bu teknikler sayesinde:

- ✦ Test süreci daha verimli hale getirilir.
- ✦ Test uzmanları geçmişte karşılaşılan hataları yeniden önleyebilir.
- ✦ Gerçek kullanıcı deneyimine daha yakın testler gerçekleştirilir.

Pratik Sorular

1. Deneyime dayalı test tekniklerinin en büyük avantajı nedir?
2. Keşif testi nasıl gerçekleştirilir?
3. Hata tahmini testi hangi durumlarda kullanılır?

Cevaplar

1. Deneyime dayalı test tekniklerinin en büyük avantajı, geçmiş deneyimlerden yararlanarak olası hataların önceden tahmin edilmesini sağlamaktır. Böylece test süreci daha verimli hale gelir ve olası sorunlar daha hızlı tespit edilir.
2. Keşif testi, önceden tanımlanmış test senaryoları olmadan, test uzmanının yazılımı serbest bir şekilde test ederek beklenmedik hataları tespit etmesiyle gerçekleştirilir. Bu test genellikle manuel olarak yapılır.
3. Hata tahmini testi, geçmiş projelerde karşılaşılan hatalara dayanarak belirli bir yazılım bileşeninin hataya yatkın olup olmadığını kontrol etmek için kullanılır. Örneğin, bir mobil bankacılık uygulamasında ödeme adımları, hata tahmini testi ile kapsamlı şekilde değerlendirilir.



33. YAPILANDIRMA YÖNETİMİ (CONFIGURATION MANAGEMENT)

Yapılandırma Yönetimi Nedir?

Yapılandırma yönetimi, yazılım geliştirme süreçlerinde **kod, belgeler, test senaryoları** ve diğer bileşenlerin düzenli olarak **takip edilmesini ve kontrol edilmesini** sağlayan bir süreçtir.

- **Amaç**, yazılımın doğru ve tutarlı bir şekilde geliştirilmesini, test edilmesini ve dağıtılmasını sağlamaktır.
- **Geliştirme sürecinde değişikliklerin** etkili bir şekilde yönetilmesine ve izlenmesine yardımcı olur.
- **Birden fazla ekip üyesinin** aynı projede çalışmasını kolaylaştırır ve sürüm takibini düzenli hale getirir.

Yapılandırma Yönetiminin Ana Bileşenleri

1. **Sürüm Kontrolü (Version Control)**
 - Yazılımın farklı versiyonlarının düzenli olarak takip edilmesini sağlar.
 - Geliştiriciler, yapılan değişiklikleri kaydedebilir ve gerektiğinde eski sürümlere geri dönebilir.
2. **Değişiklik Yönetimi (Change Management)**
 - Kod, dokümantasyon veya yapılandırma değişikliklerinin kaydedilmesini ve takip edilmesini içerir.
 - Yeni özellik ekleme, hata düzeltme veya sistem iyileştirme gibi değişiklikler belgelenir.
3. **Yapı Yönetimi (Build Management)**
 - Yazılım bileşenlerinin, bağımlılıkların ve çalışma ortamlarının yönetilmesini sağlar.
 - Sistemin farklı ortamlarda tutarlı çalışmasını sağlamak için otomatik yapı yönetim sistemleri kullanılır.
4. **Test ve Dağıtım Süreçleri (Test and Deployment Processes)**
 - Her yeni sürüm öncesinde belirli testlerin çalıştırılmasını içerir.
 - Yazılımın canlıya alınmadan önce hatalardan arındırıldığını ve sistem ile uyumlu olduğunu doğrular.

Yapılandırma Yönetiminin Önemi

✦ Tutarlılık Sağlar

- Yazılımın her sürümünün belgelenmiş ve takip edilebilir olmasını sağlar.

✦ Hata İzleme Kolaylığı Sağlar

- Hataların hangi sürümde ve hangi değişiklikte ortaya çıktığını tespit etmeyi kolaylaştırır.

✦ Ekip Çalışmasını Destekler

- Farklı ekipler arasında deęişikliklerin senkronize edilmesine yardımcı olur.

✦ Dağıtım Sürecini Güçlendirir

- Canlı ortama hatasız ve kontrollü bir şekilde yazılım sürümleri sunulmasını sağlar.

Örnek Senaryo

Bir mobil bankacılık uygulaması geliştirdiğimizi düşünelim.

✓ Sürüm Kontrolü:

- Git veya SVN gibi bir sürüm kontrol sistemi kullanılarak kod deęişiklikleri takip edilir.
- Geliştiriciler, hangi deęişikliklerin yapıldığını görebilir ve gerektiğinde eski sürümlere geri dönebilir.

✓ Deęişiklik Yönetimi:

- Yeni bir özellik eklendiğinde, bu deęişikliğin neden yapıldığı ve nasıl çalıştığı belgelenir.
- Örneğin, "QR Kod ile Ödeme" özelliği eklendiğinde, kod deęişiklikleri versiyon kontrolüne kaydedilir.

✓ Yapı Yönetimi:

- Mobil uygulamanın bağımlı olduğu API sürümleri ve üçüncü taraf kütüphaneler yapılandırılır.
- Örneğin, bir güncellemeyle yeni bir güvenlik modülü eklendiğinde, eski sistemlerle uyumlu olup olmadığı test edilir.

✓ Test ve Dağıtım:

- Her yeni sürüm öncesinde, önceden belirlenen test senaryoları çalıştırılarak hatalar tespit edilir ve sürüm onaylanır.
- CI/CD (Continuous Integration/Continuous Deployment) süreçleri kullanılarak otomatik dağıtım yapılır.

Bu süreçler sayesinde:

- ✦ Yazılım geliştirme ve dağıtım süreci daha düzenli, güvenli ve hatasız hale getirilir.
- ✦ Ekipler arasında deęişikliklerin koordinasyonu sağlanarak iş birliği artırılır.
- ✦ Gerçek kullanıcı ortamında stabil bir uygulama sunulmasına yardımcı olur.

Pratik Sorular

1. Yapılandırma yönetiminin temel amacı nedir?
2. Sürüm kontrolü neden yazılım geliştirme süreçlerinde kritik bir rol oynar?
3. Yapılandırma yönetiminde hangi araçlar kullanılabilir?

Cevaplar

1. Yapılandırma yönetiminin temel amacı, yazılımın bileşenlerini düzenli ve takip edilebilir şekilde yöneterek sürüm uyumluluğunu ve kalite kontrolünü sağlamaktır.
2. Sürüm kontrolü, yazılımın farklı versiyonlarını takip etmeye yardımcı olur. Geliştiriciler, hangi değişikliklerin yapıldığını görebilir ve gerektiğinde önceki sürümlere geri dönebilir.
3. Yapılandırma yönetiminde kullanılan bazı popüler araçlar şunlardır: Git, Subversion (SVN), Jenkins, Ansible, Chef ve Puppet.



34. RİSK VE TEST (RISK AND TESTING)

Risk ve Test Nedir?

Risk bazlı test, yazılım projelerinde **en kritik unsurları önceliklendirerek** test süreçlerini yönetmeye yardımcı olan bir yaklaşımdır.

- **Risk**, yazılımın başarısızlığa uğrama olasılığı ve bu başarısızlığın etkisinin birleşimidir.
- **Risk bazlı test stratejisi**, en yüksek riske sahip bileşenlerin daha kapsamlı test edilmesini sağlar.
- **Bu yaklaşım**, yazılımın güvenilirliğini artırarak kritik hataların önceden tespit edilmesine yardımcı olur.

Risk Türleri

✓ Ürünle İlgili Riskler (Product Risks)

- Yazılımın hatalar içermesi, performans sorunları, güvenlik açıkları gibi **teknik riskler**.
- **Örneğin**, bir e-ticaret platformunda **ödeme sisteminin çökmesi**.

✓ Proje Riskleri (Project Risks)

- **Zamanında teslim edilmeme, bütçe aşımı, yetersiz kaynak kullanımı** gibi organizasyonel riskler.
- **Örneğin**, planlanan test sürecinin gecikmesi veya bütçe kısıtlamaları nedeniyle bazı testlerin eksik yapılması.

✓ İş Riskleri (Business Risks)

- Yazılımın **müşteri ihtiyaçlarını karşılamaması veya piyasada başarısız olması**.
- **Örneğin**, kullanıcı dostu olmayan bir mobil uygulamanın düşük benimsenme oranına sahip olması.

Risk Bazlı Testin Avantajları

★ Öncelik Belirleme

- En kritik alanlara odaklanarak test sürecinin daha verimli hale gelmesini sağlar.

★ Zaman ve Maliyet Tasarrufu

- Daha az kritik bileşenler için gereksiz test yükünü azaltır.

★ Hata Etkisini Minimize Etme

- Yüksek risk taşıyan alanlarda olası hataların etkisini azaltarak **iş sürekliliğini sağlar**.

Risk Değerlendirme Süreci

1. Risk Tanımlama (Risk Identification)

- Yazılım bileşenlerinin **ne kadar kritik olduğunu belirleme**.
- **Örneğin**, kullanıcı giriş işlemi, ödeme sistemi ve veri güvenliği **en kritik alanlar olabilir**.

2. Risk Analizi (Risk Analysis)

- **Olasılık ve etkileri değerlendirerek risk seviyesinin belirlenmesi**.
- **Örneğin**, ödeme sisteminin başarısız olma olasılığı ve bu durumun müşteri kaybına etkisi analiz edilir.

3. Risk Azaltma (Risk Mitigation)

- **Yüksek riskli alanlar için daha yoğun test planı oluşturma**.
- **Örneğin**, güvenlik testleri ve yük testleri uygulanarak risk en aza indirilmeye çalışılır.

4. Sürekli İzleme (Continuous Monitoring)

- **Risk değerlendirmesini test süreci boyunca güncelleme**.
- **Örneğin**, yazılımda yapılan güncellemeler sonrası yeni risklerin oluşup oluşmadığı değerlendirilir.

Örnek Senaryo

Bir e-ticaret platformunda **ödeme sistemi** en kritik bileşenlerden biridir. **Risk bazlı test** şu şekilde uygulanabilir:

✦ Risk Tanımlama:

- **Ödeme sisteminin başarısız olması**, müşteri kayıplarına ve itibar kaybına yol açabilir.

✦ Risk Analizi:

- **Sistemin çökme ihtimali ve etkisi değerlendirilerek risk skoru belirlenir**.
- **Örneğin**, ödeme işleminin başarısız olma olasılığı yüksekse ve bu büyük bir müşteri kaybına yol açıyorsa, bu risk **yüksek öncelikli** olarak değerlendirilir.

✦ Risk Azaltma:

- **Ödeme işlemleri için kapsamlı fonksiyonel ve yük testleri gerçekleştirilir**.
- **Örneğin**, sistemin aynı anda **binlerce kullanıcının ödeme yapmasına** nasıl tepki verdiği test edilir.

✦ Sürekli İzleme:

- **Ödeme sistemindeki olası değişiklikler test süreçlerinde sürekli olarak gözden geçirilir**.

- **Örneğin**, ödeme sağlayıcısı değiştirildiğinde, sistemin yeni entegrasyonlarla çalıştığı doğrulanır.

Bu yaklaşımlar sayesinde:

- ◆ **Ödeme sisteminin sorunsuz çalışması sağlanarak müşteri deneyimi iyileştirilir.**
- ◆ **Yazılımın iş ve teknik risklere karşı daha dayanıklı olması sağlanır.**
- ◆ **Olası hatalar üretime geçmeden önce tespit edilir.**

Pratik Sorular

1. Risk bazlı testin temel amacı nedir?
2. Proje riskleri neden yazılım geliştirme sürecinde önemlidir?
3. Risk değerlendirmesi nasıl gerçekleştirilir?

Cevaplar

1. Risk bazlı testin temel amacı, yazılımın en kritik bileşenlerine öncelik vererek test süreçlerini optimize etmektir. Böylece en büyük etkiye sahip olabilecek hatalar önceden tespit edilebilir.
2. Proje riskleri, yazılım geliştirme sürecinin başarısını doğrudan etkileyebilir. Örneğin, zamanında teslim edilmeyen bir yazılım, müşteri memnuniyetsizliğine ve mali kayıplara yol açabilir.
3. Risk değerlendirmesi, önce risklerin tanımlanması, ardından olasılık ve etkilerinin analiz edilmesiyle gerçekleştirilir. Yüksek riskli bileşenlere daha fazla test kaynağı ayrılarak risk azaltılır.

35. OLAY/HATA YÖNETİMİ

(INCIDENT/ERROR MANAGEMENT)

Olay/Hata Yönetimi Nedir?

Olay/hata yönetimi, yazılım test sürecinde tespit edilen hataların kayıt altına alınmasını, analiz edilmesini, çözülmesini ve yeniden test edilmesini içeren bir süreçtir.

- Bu süreç, yazılımın **kalite güvencesini** sağlamaya yardımcı olur ve **test sürecinin verimli yürütülmesini** sağlar.
- **Test mühendisleri**, yazılım hatalarını tanımlayarak **geliştirici ekibin** bu hataları düzeltmesini sağlar.
- **Kritik hataların zamanında tespit edilip düzeltilmesi**, yazılımın **stabilitesini** ve **güvenilirliğini** artırır.

Olay/Hata Yönetim Sürecinin Adımları

1. Olayın Tespiti (Incident Identification)

- Test sırasında veya canlı sistemde bir hata meydana geldiğinde tespit edilir.
- **Örnek:** Bir mobil bankacılık uygulamasında para transferi yapılamıyorsa, bu bir hata olarak rapor edilir.

2. Olayın Kayıt Altına Alınması (Incident Logging)

- Hata türü, etkisi, tekrar oluşturma adımları gibi bilgiler **detaylı bir şekilde doküman**te edilir.
- **Jira veya Bugzilla** gibi hata yönetim araçlarına kaydedilir.

3. Olayın Önceliklendirilmesi (Incident Prioritization)

- Hataya neden olan olayın yazılım üzerindeki **etkisine** göre sınıflandırılır:
 - **Düşük (Low):** Kullanıcı deneyimini etkileyen ancak işlevselliğe zarar vermeyen hatalar.
 - **Orta (Medium):** Belirli bir modülü etkileyen ancak sistem genelinde büyük sorun yaratmayan hatalar.
 - **Yüksek (High):** Önemli işlevleri etkileyen ancak alternatif çözümler sunulabilen hatalar.
 - **Kritik (Critical):** Sistem çökmesine, veri kaybına veya güvenlik açıklarına neden olan hatalar.

4. Olayın Analiz Edilmesi (Incident Analysis)

- Hatanın **nedeni ve kapsamı** analiz edilir.
- **Kodda veya sistem bileşenlerinde** hata oluşturan unsur tespit edilir.

5. Çözüm ve Düzeltme (Resolution & Fixing)

- **Geliştiriciler hatayı düzeltir** ve yeni bir sürüm oluşturur.
- Hata düzeltildiğinde **test ekibine bildirim** yapılır.

6. Doğrulama ve Yeniden Test (Verification & Re-testing)

- Hata düzeltildikten sonra ilgili testler tekrar çalıştırılarak düzeltmenin başarılı olup olmadığı kontrol edilir.
- **Regresyon testi** uygulanarak diğer işlevlerin etkilenip etkilenmediği değerlendirilir.

7. Olayın Kapatılması (Incident Closure)

- Hata giderildiğinde ve testlerden başarıyla geçtiğinde olay kapatılır.
- Gelecekte benzer hataların önlenmesi için belgeler güncellenir.

Olay/Hata Yönetiminin Önemi

✦ Hataların Takip Edilebilirliğini Sağlar

- Yazılımın kalitesini artırmak için hataların **kaydedilmesi ve yönetilmesi** önemlidir.

✦ Etkili Sorun Giderme Sağlar

- Test süreçlerinde bulunan hataların **etkili bir şekilde analiz edilmesini ve düzeltilmesini** sağlar.

✦ Zaman ve Kaynak Yönetimini Geliştirir

- **Kritik hatalara öncelik vererek** test sürecinin **verimli yürütülmesine** yardımcı olur.

Örnek Senaryo

Bir bankacılık mobil uygulaması geliştirdiğimizi düşünelim.

✦ Olay Tespiti:

- Para transferi yapmaya çalışan kullanıcılar **hata mesajı alıyor**.

✦ Kayıt Altına Alma:

- Hata mesajı, kullanıcı senaryoları ve tekrar oluşturma adımları test yönetim aracına kaydedilir.

✦ Önceliklendirme:

- Hata, **finansal kayıplara yol açabileceği için “kritik”** olarak işaretlenir.

✦ Analiz ve Düzeltme:

- Geliştirici ekibi hatanın nedenini analiz eder ve bir düzeltme yapar.
- Örnek: Hatanın veri tabanı bağlantı probleminden kaynaklandığı tespit edilir.

✦ Doğrulama:

- Güncellenmiş sürüm tekrar test edilerek hatanın düzeltilip düzeltilmediği kontrol edilir.

✦ Olay Kapatma:

- Hata başarılı bir şekilde giderildiğinde test ekibi tarafından olay kapatılır.

Bu süreç sayesinde:

- Hata etkili bir şekilde yönetilmiş ve kullanıcıların finansal kayıp yaşamadan uygulamayı sorunsuz kullanması sağlanmıştır.
- Benzer hataların gelecekte önlenmesi için süreç belgelenmiştir.
- Yazılımın genel kalitesi artırılmıştır.

Pratik Sorular

1. Olay/hata yönetimi sürecinin en önemli adımı nedir ve neden?
2. Hata önceliklendirme neden gereklidir ve nasıl yapılır?
3. Hata yönetim araçlarına iki örnek verin ve kullanım amaçlarını açıklayın.

Cevaplar

1. Hata kayıt altına alma en önemli adımdır çünkü hatanın etkili bir şekilde analiz edilmesi, düzeltilmesi ve yeniden test edilmesi için doğru dokümantasyon gereklidir.
2. Hata önceliklendirme, en kritik hatalara öncelik verilmesini sağlar. Önceliklendirme, hatanın yazılım üzerindeki etkisine ve ortaya çıkma olasılığına göre düşük, orta, yüksek veya kritik olarak sınıflandırılarak yapılır.
3. Jira ve Bugzilla hata yönetim araçlarına örnektir. Jira, yazılım geliştirme süreçlerinde hataları takip etmek ve projeleri yönetmek için kullanılır. Bugzilla ise açık kaynaklı bir hata takip sistemi olup, hataların raporlanması ve izlenmesi için kullanılır.

36. TEST ARAÇ TÜRLERİ (TEST TOOL TYPES)

Test Araç Türleri Nedir?

Test araçları, yazılım test süreçlerini daha verimli ve hızlı hale getirmek için kullanılan yazılımlardır.

- Manuel test sürecini destekleyebilir, test otomasyonunu sağlayabilir ve hata yönetimini kolaylaştırabilir.
- Test sürecinin farklı aşamalarına göre çeşitlenen birçok farklı test aracı bulunmaktadır.
- Bu araçlar, yazılımın işlevselliğini, performansını, güvenliğini ve genel kalitesini artırmaya yardımcı olur.

Test Araç Türleri ve Kullanım Alanları

1. Test Yönetim Araçları (Test Management Tools)

- Test planlarını, test senaryolarını ve ilerleme raporlarını yönetir.
- Örnek Araçlar: TestRail, HP ALM, Zephyr

2. Hata Takip ve Yönetim Araçları (Bug Tracking & Defect Management Tools)

- Yazılım hatalarının kaydedilmesini, izlenmesini ve yönetilmesini sağlar.
- Örnek Araçlar: Jira, Bugzilla, Redmine

3. Otomasyon Test Araçları (Automation Testing Tools)

- Manuel test sürecini hızlandırarak testlerin otomatik çalıştırılmasını sağlar.
- Örnek Araçlar: Selenium, Appium, TestComplete

4. Performans Test Araçları (Performance Testing Tools)

- Yazılımın yük altında nasıl çalıştığını test eder.
- Örnek Araçlar: JMeter, LoadRunner, Gatling

5. Güvenlik Test Araçları (Security Testing Tools)

- Yazılımın güvenlik açıklarını tespit eder.
- Örnek Araçlar: OWASP ZAP, Burp Suite, Acunetix

6. Statik Analiz Araçları (Static Analysis Tools)

- Yazılım kodunun çalıştırılmadan analiz edilmesini sağlar.
- Örnek Araçlar: SonarQube, Checkstyle, PMD

Test Araçlarının Avantajları

✦ Zaman Tasarrufu Sağlar

- Otomatik testler, manuel testlere göre daha hızlı çalıştırılabilir.

✦ Hata Takibini Kolaylaştırır

- Hata yönetim araçları sayesinde hatalar kolayca raporlanabilir ve takip edilebilir.

✦ Daha Güvenilir Sonuçlar Elde Edilir

- Test otomasyon araçları, insan hatasını minimize eder.

✦ Performans ve Güvenlik Açısından Daha Güçlü Testler Sağlar

- Performans testleri, sistemin yoğun yük altında nasıl çalışacağını önceden belirlemeye yardımcı olur.

Örnek Senaryo

Bir e-ticaret platformunda ödeme işlemlerini test ettiğimizi düşünelim. Aşağıdaki test araçlarını kullanabiliriz:

✦ Test Yönetimi İçin:

- **TestRail kullanarak** test senaryolarını ve test planlarını yönetiriz.

✦ Hata Takibi İçin:

- **Jira kullanarak** ödeme sırasında oluşan hataları kaydeder ve geliştiricilere iletiriz.

✦ Otomasyon İçin:

- **Selenium kullanarak** ödeme işlemlerini manuel yerine otomatik test ederiz.

✦ Performans İçin:

- **JMeter kullanarak** ödeme sisteminin yoğun yük altında nasıl çalıştığını test ederiz.

✦ Güvenlik İçin:

- **OWASP ZAP ile** ödeme sistemindeki güvenlik açıklarını tararız.

Bu süreç sayesinde:

- ◆ Ödeme işlemlerinin hatasız ve güvenli bir şekilde çalıştığından emin oluruz.
- ◆ Test sürecini daha verimli hale getiririz.
- ◆ Potansiyel hataları daha erken aşamada tespit edebiliriz.

Pratik Sorular

1. Test otomasyon araçlarının avantajları nelerdir?
2. Performans testi neden gereklidir ve hangi araçlar kullanılır?
3. Hata takip araçlarının yazılım geliştirme sürecindeki önemi nedir?

Cevaplar

1. Test otomasyon araçları, manuel test süreçlerini hızlandırır, hata oranını düşürür ve tekrar eden testleri otomatikleştirerek zaman kazandırır.
2. Performans testi, yazılımın yüksek yük altında nasıl çalıştığını analiz etmek için gereklidir. Kullanılan araçlar arasında JMeter, LoadRunner ve Gatling bulunmaktadır.
3. Hata takip araçları, yazılım geliştirme sürecinde hataların kaydedilmesini, önceliklendirilmesini ve çözülmesini sağlar. Bu araçlar sayesinde hatalar sistematik bir şekilde izlenebilir ve ekipler arasında etkili bir iletişim sağlanır.

37. TEST ARAÇLARININ ETKİN KULLANIMI: POTANSİYEL AVANTAJLAR VE RİSKLER (EFFECTIVE USE OF TEST TOOLS: POTENTIAL BENEFITS AND RISKS)

Test Araçlarının Etkin Kullanımı Nedir?

Test araçlarının etkin kullanımı, yazılım test süreçlerini otomatikleştirmek, izlemek, analiz etmek ve iyileştirmek için çeşitli test araçlarını en verimli şekilde kullanmayı ifade eder.

- Doğru seçilmesi ve uygulanması, yazılım kalitesini artırabilir.
- Ancak, yanlış kullanım, beraberinde riskler de getirebilir.
- Bu nedenle test araçlarının seçimi, uygulanması ve yönetimi dikkatli bir şekilde yapılmalıdır.

Test Araçlarının Potansiyel Avantajları

★ Zaman ve Maliyet Tasarrufu

- Otomasyon araçları, manuel test sürecini hızlandırarak daha az kaynak ile daha fazla test yapılmasını sağlar.

★ Hata Tespitinde Doğruluk

- Test araçları, manuel testlere göre daha tutarlı ve doğru sonuçlar verir.

★ Veri Toplama ve Raporlama

- Test sonuçları sistematik toplanarak detaylı analiz ve raporlar oluşturulur.
- Örnek: Jira, Bugzilla gibi hata takip araçları hataların kolayca kaydedilip raporlanmasını sağlar.

★ Tekrarlanabilirlik ve Tutarlılık

- Otomasyon testleri, testlerin tutarlılıkla tekrar edilmesini sağlar.

★ Sürekli Entegrasyon ve Dağıtım (CI/CD) Desteği

- Test araçları, Jenkins, GitHub Actions gibi platformlarda otomatik testlerin sürekli entegrasyonunu sağlar.

Test Araçlarının Potansiyel Riskleri

⚠ Yanlış Araç Seçimi

- Proje gereksinimlerine uygun olmayan araç seçimi, zaman ve maliyet kaybına neden olur.

⚠ Yüksek Bakım Maliyeti

- Sürekli değişen projelerde, otomatik testlerin bakım maliyeti artabilir.

⚠ Yanlış Pozitif ve Yanlış Negatif Sonuçlar

- Hatalı test sonuçları, yanlış analizlere yol açabilir.

⚠ Eğitim ve Yetkinlik Eksikliği

- Araçların yanlış kullanımı, test sonuçlarının doğruluğunu etkileyebilir.

⚠ Bağımlılık

- Sadece otomasyon testlerine güvenmek, manuel keşif testlerini göz ardı edebilir.

Örnek Senaryo

Bir e-ticaret sitesinde ödeme işlemlerinin test edilmesini ele alalım:

★ Avantaj

- **Selenium ile test otomasyonu**, manuel tekrar eden testlerin otomatikleştirilmesiyle zaman tasarrufu sağlar.

★ Risk

- Yanlış otomasyon stratejisi, sık değişen bileşenler nedeniyle yüksek bakım maliyeti oluşturabilir.

★ Çözüm

- Ekiplerin eğitimini sağlamak, doğru test stratejisi belirlemek ve manuel testleri ihmal etmemek.

Bu süreç sayesinde:

- ➤ Doğru stratejiler ile test araçlarının avantajlarından tam faydalanılır.
- ➤ Potansiyel riskler önceden tespit edilerek minimize edilir.
- ➤ Testlerin doğruluğu ve etkinliği artar.

Test Araçlarının Etkin Kullanımı İçin En İyi Uygulamalar

- ✓ Projeye uygun araçları seçmek
- ✓ Otomasyon ve manuel testleri dengeli kullanmak
- ✓ Test senaryolarını sürekli güncellemek
- ✓ Test mühendislerine gerekli eğitimleri vermek
- ✓ Otomatik sonuçları manuel doğrulamalarla desteklemek

Pratik Sorular

1. Test araçlarının yanlış seçilmesi hangi riskleri doğurabilir?
2. Otomasyon testlerinin avantajlarından biri olan zaman tasarrufu nasıl sağlanır?
3. Test araçlarının yanlış kullanımı nedeniyle ortaya çıkabilecek en büyük risk nedir?

Cevaplar

1. Yanlış test aracı seçimi, proje gereksinimlerine uygun olmayan bir test sürecine yol açabilir ve verimsizliğe neden olabilir.
2. Otomasyon testleri, sık tekrarlanan manuel testleri otomatikleştirerek zaman tasarrufu sağlar.
3. Yanlış kullanım, hatalı sonuçlara ve yanlış pozitif/negatif durumlara yol açarak, yazılımın hatalı piyasaya sürülmesine sebep olabilir.

38. ETİK KURALLAR (ETHICAL RULES)

Etik Kurallar Nedir?

Etik kurallar, yazılım testi süreçlerinde test mühendislerinin uyması gereken **mesleki standartları ve sorumlulukları** kapsar.

Bu kurallar:

- Test süreçlerinin güvenilirliğini ve tarafsızlığını sağlamak
- Müşteri gizliliğini korumak
- Adil ve profesyonel test uygulamalarını teşvik etmek

amacıyla belirlenmiştir.

Etik Kuralların Temel İlkeleri

1. Dürüstlük ve Tarafsızlık

- Test sonuçları objektif bir şekilde değerlendirilmeli ve raporlanmalıdır.
- Yanıltıcı bilgilerden kaçınılmalı ve doğrular açıkça belirtilmelidir.

2. Gizliliğe Saygı

- Müşteri verileri korunmalı ve yetkisiz erişime karşı güvence altına alınmalıdır.
- Gizlilik ihlali, şirketin ve müşterinin güvenini zedeler.

3. Mesleki Yetkinlik ve Sürekli Gelişim

- Test mühendisleri, yeni test tekniklerini ve araçlarını öğrenmeli, sürekli kendilerini geliştirmelidir.
- Test dünyasındaki değişimleri takip ederek daha etkin test süreçleri uygulanmalıdır.

4. Yasalara ve Standartlara Uyum

- Test süreçleri, ulusal ve uluslararası yasal düzenlemelere uygun şekilde yürütülmelidir.
- GDPR, KVKK gibi veri gizliliği yasalarına dikkat edilmelidir.

5. Bağımsızlık

- Test ekipleri, diğer birimlerin baskısından etkilenmeden bağımsız hareket etmelidir.
- Geliştirme ekibinden gelen baskılar test sonuçlarını değiştirmemelidir.

6. Yanıltıcı Bilgiden Kaçınma

- Test sonuçları eksiksiz, doğru ve açıkça sunulmalıdır.
- Bir hatanın önemini küçültmek veya abartmak, test sürecinin güvenilirliğini zedeler.

Etik Kuralların Önemi

✦ Güvenilirliği Artırır

- Test süreçlerinin doğruluğunu garanti eder.
- Yönetim ve geliştiriciler için güvenilir test verileri sağlar.

✦ Veri Güvenliğini Sağlar

- Müşteri ve kullanıcı verilerinin korunmasını garanti eder.
- Hukuki yaptırımları ve veri sızıntılarını önler.

✦ Profesyonellik Sağlar

- Test mühendislerinin mesleki sorumluluklarını yerine getirmesini sağlar.
- Şirket içi etik standartlarının korunmasına katkıda bulunur.

Örnek Senaryo

Bir bankacılık sisteminin test sürecinde, test mühendisi **müşteri hesaplarına erişim sağlayan bir güvenlik açığı bulur**. Etik kurallara uygun olarak:

✦ Gizliliğe Saygı Gösterme:

- Güvenlik açığı halka açık paylaşılmamalı.
- Kullanıcı verileri korunmalıdır.

✦ Yetkili Birimlere Bildirme:

- Güvenlik açığı banka yöneticileri ve güvenlik ekibine raporlanmalı.
- Teknik ekiple hızlı çözüm için paylaşılmalı.

✦ Yanıltıcı Bilgiden Kaçınma:

- Açığın gerçekçi bir raporu sunulmalı.
- Yönetim ekibi gerçek risklerle bilgilendirilmeli.

Bu süreç sayesinde:

- ◆ Güvenlik açığı etik kurallar çerçevesinde yönetilir.
- ◆ Müşteri bilgileri korunur.
- ◆ Güvenilir ve profesyonel bir test süreci sağlanır.

Pratik Sorular

1. Etik kuralların test mühendisleri için en önemli ilkesi nedir ve neden?
2. Test mühendisleri neden mesleki gelişimlerine önem vermelidir?
3. Gizlilik ihlalinin yazılım testi süreçlerine etkisi nedir?

Cevaplar

1. Dürüstlük ve tarafsızlık, çünkü test sonuçlarının manipüle edilmesi veya yanıltıcı olması, yazılım kalitesini ve güvenilirliğini ciddi şekilde zedeler.
2. Yazılım geliştirme ve test süreçleri sürekli geliştiği için, test mühendisleri güncel bilgi ve becerilere sahip olarak test süreçlerini daha etkin ve doğru yönetebilirler.
3. Gizlilik ihlali, müşteri güven kaybına, hukuki yaptırımlara ve şirket itibarının zarar görmesine yol açabilir. Müşteri ve kullanıcı verilerini korumak bu nedenle kritik önemdedir.

39. AGILE TEST SÜREÇLERİ (AGILE TESTING PROCESSES)

Agile Test Süreçleri Nedir?

Agile test, yazılım geliştirme sürecinin bir parçası olarak erken ve sürekli geri bildirim sağlamak için kullanılan bir yaklaşımdır.

- Geleneksel testlerde testler genellikle geliştirme tamamlandıktan sonra uygulanırken,
- Agile test sürecinde testler **her iterasyonda (sprint bazlı)** yapılır.
- Bu sayede hatalar **daha erken tespit edilir ve hızlıca düzeltilir**.

Agile Testin Temel Özellikleri

1. Kısa Teslimat Süreçleri

- Testler, **sprint bazlı** gerçekleştirilir.
- Her iterasyonda testler çalıştırılarak **yazılımın kalitesi korunur**.

2. Otomasyon ve Sürekli Entegrasyon (CI/CD)

- **Test otomasyonu**, tekrarlayan testleri hızlandırır.
- **Sürekli entegrasyon süreçleriyle (CI/CD)** kod her değiştiğinde test edilir.

3. Scrum ve Kanban ile Entegrasyon

- Agile metodolojilerle uyumlu çalışır.
- **Scrum, Kanban gibi yönetim sistemlerinde** testler sprint bazlı yönetilir.

4. Ekip İş Birliği

- Geliştiriciler, test uzmanları ve iş birimi **birlikte çalışır**.
- Hatalar **daha hızlı tespit edilir ve düzeltilir**.

5. Adaptif Test Süreci

- Gereksinimler değiştikçe **test süreçleri de buna uyum sağlar**.
- Esnek ve **değişime açık bir test süreci uygulanır**.

Örnek Senaryo

Bir **mobil yemek sipariş uygulaması** geliştirildiğini düşünelim. Agile test süreçleri şu şekilde uygulanabilir:

1. Sprint Bazlı Testler:

- İlk sprint'te kullanıcı giriş ekranı test edilir.
- İkinci sprint'te menü görüntüleme fonksiyonu test edilir.
- Üçüncü sprint'te sipariş tamamlama testi yapılır.

2. Sürekli Entegrasyon ile Test:

- Her kod değişikliği sonrası **otomatik testler çalıştırılır**.
- Bu sayede **hatalar erken tespit edilir** ve anında düzeltilir.

3. Kullanıcı Geri Bildirimine Dayalı Testler:

- Gerçek kullanıcılar tarafından rapor edilen sorunlar **anında yeni sprint'te çözülür**.
- Bu testler, **sürekli iyileştirme sürecine katkı sağlar**.

Bu süreç sayesinde:

- Her iterasyonda **daha kaliteli ve hatasız bir yazılım oluşturulur**.
- Test süreçleri daha hızlı ve esnek hale gelir.
- Son kullanıcı memnuniyeti ve ürün kalitesi artırılır.

Pratik Sorular

1. Agile testin geleneksel test süreçlerinden farkı nedir?
2. Agile test sürecinde geliştiriciler ve test mühendisleri nasıl birlikte çalışır?
3. Agile test sürecinde otomasyonun rolü nedir?

Cevaplar

1. Agile test, kısa döngülerle sürekli test yapar ve geliştirme ekibiyle entegre çalışır. Geleneksel testlerde ise testler geliştirme sürecinin sonunda gerçekleştirilir.
2. Agile süreçlerde geliştiriciler ve test mühendisleri sürekli iletişim halinde olup,
 - Testler geliştirme sürecinin her aşamasında uygulanır.
 - Hatalar anında raporlanır ve sprint içinde düzeltilir.
3. Otomasyon, Agile test sürecinde tekrar eden testleri hızlandırır.
 - Sürekli entegrasyon (CI/CD) sayesinde hatalar erken aşamada tespit edilir.
 - Kod her değiştiğinde otomatik test çalıştırılır, böylece manuel test yükü azalır.

40. RİSK TEMELLİ TEST YAKLAŞIMLARI (RISK-BASED TESTING APPROACHES)

Risk Temelli Test Nedir?

Risk temelli test (Risk-Based Testing - RBT), test faaliyetlerinin en kritik ve yüksek risk taşıyan alanlara odaklanmasını sağlayan bir yaklaşımdır.

- Yazılım geliştirme sürecinde her modül aynı öneme sahip değildir.
- Bazı bileşenlerin hata yapma ihtimali daha yüksektir ve yazılımın genel işleyişine etkisi büyüktür.
- Bu nedenle, riskleri değerlendirerek test süreçlerini önceliklendirmek, yazılım kalitesini artırmada etkili bir yöntemdir.

Risk Değerlendirme Kriterleri

1. Hata Olasılığı

- Daha önce hata barındıran veya teknik olarak karmaşık bileşenler yüksek risk taşır.

2. İşletme Etkisi

- Kullanıcı deneyimine doğrudan etki eden veya kritik iş süreçlerini barındıran bölümler öncelikli test edilmelidir.

3. Uyumluluk ve Entegrasyon Riskleri

- Farklı sistemlerle entegrasyon içeren bölümler, diğer sistemleri bozma ihtimali yüksek olduğundan daha fazla test edilmelidir.

4. Güvenlik ve Performans Gereksinimleri

- Hassas verilerin işlendiği veya yüksek kullanıcı trafiği beklenen alanlarda daha kapsamlı testler gereklidir.

Risk Temelli Testin Avantajları

✓ Önceliklendirme Sağlar

- En kritik fonksiyonlar için daha fazla test kaynağı ayrılmasını sağlar.

✓ Test Kapsamını Optimize Eder

- Gereksiz testleri azaltarak zaman ve maliyet tasarrufu sağlar.

✓ Kullanıcı Deneyimini Artırır

- Kullanıcıya en çok etki eden alanlara odaklanarak müşteri memnuniyetini yükseltir.

Risk Temelli Test Süreci

1. Riskleri Belirleme

- Yazılımın en kritik bileşenleri belirlenir.
(Örneğin: Ödeme işlemleri, giriş doğrulama sistemleri vb.)

2. Risklerin Değerlendirilmesi

- Hata olasılığı ve işletme etkisi göz önünde bulundurularak risk seviyeleri belirlenir.
(Örneğin: Güvenlik açıkları içeren modüller, daha sık hata barındıran sistemler vb.)

3. Test Planı Hazırlama

- Risk seviyesine göre test öncelikleri belirlenir.
(Örneğin: Kritik bileşenler daha sık test edilir.)

4. Testlerin Yürütülmesi

- En yüksek riskli bileşenler öncelikli test edilir.

5. Sürekli Gözden Geçirme

- Yazılım geliştikçe risk analizleri güncellenir ve test kapsamı optimize edilir.

Örnek Senaryo

Bir banka mobil uygulaması için risk temelli test sürecini ele alalım:

- **Risk Analizi:** Kullanıcı giriş ekranı, para transferi işlemleri ve şifre sıfırlama gibi kritik işlemler en yüksek risk grubuna alınır.
- **Test Önceliklendirme:** Hata olasılığı yüksek ve güvenlik açısından kritik alanlar öncelikli olarak test edilir.
- **Gerçekleştirme:** Kullanıcı girişi ve ödeme işlemleri için penetrasyon testleri ve yük testleri uygulanarak sistemin güvenilirliği sağlanır.

Bu süreç sayesinde:

- En kritik sistemler öncelikli olarak test edilerek büyük hataların üretime çıkmadan önce bulunması sağlanır.
- Yazılım kalitesi ve güvenilirliği artırılır.
- Zaman ve kaynak tasarrufu yapılır.

Pratik Sorular

1. Risk temelli testin en büyük avantajı nedir?
2. Test sürecinde riskler nasıl belirlenir?
3. Risk seviyesi yüksek olan bileşenler hangi kriterlere göre önceliklendirilir?

Cevaplar

1. Risk temelli test, en kritik fonksiyonlara öncelik vererek yazılımın güvenilirliğini artırır ve test kaynaklarının verimli kullanılmasını sağlar.
2. Yazılım bileşenlerinin hata olasılığı, işlevsellik, kullanıcı etkisi, entegrasyon noktaları ve güvenlik gibi faktörler analiz edilerek belirlenir.
3. Hata geçmişi, kullanıcıya etkisi, entegrasyon karmaşıklığı ve güvenlik gereksinimleri göz önünde bulundurularak test öncelikleri belirlenir.



41. TEST OTOMASYONU

(TEST AUTOMATION)

Test Otomasyonu Nedir?

Test otomasyonu, yazılım test süreçlerini manuel testlerden daha hızlı ve verimli hale getirmek için test araçları kullanarak testlerin otomatikleştirilmesini ifade eder.

- Tekrar eden testlerin otomatik çalıştırılması,
- Hata tespitinin hızlandırılması,
- Yazılım kalitesinin artırılması amaçlanır.

Test Otomasyonunun Temel Avantajları

★ Zaman Tasarrufu

- Manuel testlere kıyasla daha hızlı çalıştırılarak **test süresi kısaltılır**.

★ Tutarlılık ve Tekrarlanabilirlik

- Her test senaryosu **aynı koşullarda tekrar tekrar çalıştırılabilir**.

★ Erken Hata Tespiti

- Hataların **geliştirme sürecinin erken aşamalarında tespit edilmesini sağlar**.

★ Kapsamlı Test Yapılmasını Sağlar

- Çok sayıda test senaryosu **daha hızlı ve geniş çapta uygulanabilir**.

★ Regresyon Testlerinin Otomatikleştirilmesi

- Önceden çalışan fonksiyonların, yeni kod değişikliklerinden **etkilenip etkilenmediğini test etmek için kullanılır**.

Test Otomasyon Araçları ve Kullanım Alanları

1. Selenium

- **Web uygulamalarının otomatik testinde kullanılır.**
- **Özellikleri:** Farklı programlama dillerini destekler (Java, Python, C# vb.), birçok tarayıcıda test çalıştırabilir.
- **Kullanım Alanları:** Fonksiyonel testler, regresyon testleri.

2. Appium

- **Mobil uygulama testleri için kullanılır.**
- **Özellikleri:** Hem Android hem de iOS uygulamalarını destekler.
- **Kullanım Alanları:** Mobil uygulama fonksiyonel testleri.

3. JMeter

- **Performans ve yük testlerinde kullanılır.**
- **Özellikleri:** Yük testleri yaparak sistemin büyük trafik altında nasıl davrandığını test eder.
- **Kullanım Alanları:** Web uygulamalarının yük testleri, API testleri.

4. TestComplete

- **Masaüstü, web ve mobil uygulamalar için test otomasyonu sağlar.**
- **Özellikleri:** Kullanıcı dostu arayüzü ile kod yazmadan test senaryoları oluşturulabilir.
- **Kullanım Alanları:** GUI testleri, fonksiyonel testler.

5. Cypress

- **Modern web uygulamalarını test etmek için kullanılır.**
- **Özellikleri:** Hızlı test yazımı ve çalıştırma süresi.
- **Kullanım Alanları:** Front-end testleri, entegrasyon testleri.

Ne Zaman Test Otomasyonu Kullanılmalıdır?

- **Sık Tekrarlanan Testler:** Regresyon testleri gibi sürekli tekrar edilen testlerde.
- **Büyük ve Karmaşık Projeler:** Manuel testlerin yetersiz kaldığı büyük projelerde.
- **Zaman Kritik Projeler:** Yazılımın hızlı geliştirilmesi gereken durumlarda.
- **Yük ve Performans Testleri:** Manuel olarak gerçekleştirilmesi zor olan büyük ölçekli yük ve stres testlerinde.

Örnek Senaryo

Bir online alışveriş sitesi düşünelim:

- **Selenium** kullanarak farklı tarayıcılarda kullanıcı giriş işlemleri otomatik test edilebilir.
- **TestComplete** ile ödeme işlemleri test edilebilir.
- **JMeter** ile 1000 kullanıcının aynı anda alışveriş yapması simüle edilebilir.

Bu süreç sayesinde:

- **Test süreleri kısılır,**
- **Hatalar daha hızlı tespit edilir,**
- **Sistemin performansı değerlendirilir.**

Pratik Sorular

1. Test otomasyonunun manuel testlere göre en büyük avantajı nedir?
2. Hangi durumlarda test otomasyonu tercih edilmelidir?
3. Test otomasyonu hangi test türlerinde kullanılabilir?

Cevaplar

1. Test otomasyonu, manuel testlere göre daha hızlı çalışır ve tutarlı sonuçlar sağlar, böylece hata tespiti daha hızlı yapılır.
2. Sürekli tekrar eden testler, büyük ve karmaşık projeler, zaman kritik projeler ve performans testleri için test otomasyonu tercih edilmelidir.
3. Fonksiyonel testler, regresyon testleri, yük ve performans testleri, güvenlik testleri gibi birçok farklı test türünde kullanılabilir.



42. YAPAY ZEKA VE MAKİNE ÖĞRENMESİ İLE TEST OTOMASYONU (AI AND MACHINE LEARNING IN TEST AUTOMATION)

Yapay Zeka Destekli Test Otomasyonu Nedir?

Yapay zeka (AI) ve **makine öğrenmesi (ML)** destekli test otomasyonu, test süreçlerini optimize etmek ve otomasyon seviyesini artırmak için AI teknolojilerini kullanarak testlerin daha akıllı hale getirilmesini sağlar.

- **Geleneksel test otomasyonu:** Önceden tanımlanmış test senaryoları üzerinden çalışır.
- **AI destekli test otomasyonu:**
 - Test verilerinden öğrenir.
 - Dinamik test senaryoları üretir.
 - Hataları öngörebilir.
 - Test sürecini otomatik optimize eder.

Amaç:

- Test süreçlerini hızlandırmak
- Hata tespitini iyileştirmek
- Bakım maliyetlerini azaltmak
- Sürekli entegrasyon (CI/CD) süreçlerini desteklemek

AI Destekli Test Araçları

Araç	Ne İşe Yarar?	Öne Çıkan Özellikler
Testim	AI destekli otomatik test senaryoları oluşturur.	Otomatik test optimizasyonu
Applitools	Görsel test otomasyonu yapar.	UI değişikliklerini tespit eder.
Mabl	Regresyon testlerini otomatize eder.	Akıllı hata tahmini yapar.
Functionize	Kodsuz test otomasyonu sunar.	AI destekli test senaryosu üretir.

AI'nin Test Senaryosu Oluşturmadaki Rolü

1. Öğrenme Yeteneği

- Geçmiş test verilerinden öğrenerek yeni test senaryoları üretir.

2. Dinamik Test Senaryoları

- Yazılımın değişen bileşenlerine göre testleri günceller.
- En kritik alanlara odaklanır.

3. Hata Tahmini

- Test sonuçlarını analiz eder.
- Olası hata alanlarını belirleyerek test kapsamını iyileştirir.

4. Sürekli Güncellenen Testler

- Geleneksel otomasyon testleri manuel güncellenirken, AI destekli testler otomatik olarak kendini yeniler.

Örnek Senaryo

Bir bankacılık uygulamasında **AI destekli test otomasyonu** ile hataları azaltmak mümkün olabilir:

- **Hata Tahmini:** AI, önceki test sonuçlarını analiz ederek, kullanıcı giriş ekranındaki olası hataları tahmin edebilir.
- **Otomatik Test Senaryosu Üretimi:** Kullanıcı hareketlerini takip eder ve dinamik test senaryoları oluşturur.
- **Özelleştirilmiş Test Kapsamı:** AI, sistemin en kritik bölümlerine öncelik vererek test kapsamını optimize eder.

Sonuç:

- Daha az manuel müdahale
- Daha hızlı test süreçleri
- Daha az hata ve daha yüksek yazılım kalitesi

Pratik Sorular

1. Yapay zeka destekli test otomasyonu ile geleneksel test otomasyonu arasındaki temel fark nedir?
2. AI destekli test araçları test süreçlerini nasıl optimize eder?
3. AI'nin test senaryosu oluşturmadaki en büyük avantajı nedir?

Cevaplar

1. Geleneksel test otomasyonu önceden tanımlanmış test senaryoları kullanırken, AI destekli testler verilerden öğrenerek dinamik test senaryoları üretebilir ve kendini güncelleyebilir.
2. AI destekli test araçları, hata tahmini yaparak en kritik test senaryolarına öncelik verir, test senaryolarını sürekli günceller ve manuel test yükünü azaltarak zaman tasarrufu sağlar.
3. AI destekli testler, yazılım değişikliklerini algılayarak test senaryolarını otomatik güncelleyebilir ve test sürecini dinamik hale getirebilir. Böylece bakım maliyetlerini düşürür ve test sürecini hızlandırır.

43. DEVOPS TEST SÜREÇLERİ (DEVOPS TESTING PROCESSES)

DevOps Test Süreçleri Nedir?

DevOps test süreci, Sürekli Entegrasyon (CI) ve Sürekli Teslimat (CD) ile testlerin otomatik ve sürekli yapılmasını hedefleyen bir yaklaşımdır.

- Agile metodolojisini tamamlayıcıdır.
- Geliştirme, test ve operasyon ekipleri arasındaki bariyerleri kaldırır.
- Test süreçlerini hızlandırır ve yazılım kalitesini sürekli kılar.

DevOps'un Amacı:

- Geliştirme ve test süreçlerini otomatikleştirmek.
- Hataları erken aşamada tespit etmek.
- Sürekli entegrasyon ve teslimat süreçlerini sağlamak.

DevOps Testin Temel Özellikleri

1 Sürekli Test (Continuous Testing)

- Her kod değişikliği anında test edilir.
- CI/CD süreçleri ile entegre çalışır.

2 Ekipler Arasında İş Birliği

- Geliştirme, test ve operasyon ekipleri bir arada çalışır.
- Bariyerler kaldırılarak hızlı geri bildirim sağlanır.

3 Shift-Left ve Shift-Right Yaklaşımları

- **Shift-Left Testing ←:**
 - Testlerin yazılım geliştirme sürecinin erken aşamalarında başlamasını sağlar.
 - Erken tespit edilen hatalar daha düşük maliyetle düzeltilir.
 - Birim ve entegrasyon testlerine daha fazla ağırlık verilir.
- **Shift-Right Testing →:**
 - Testlerin canlı ortama kadar sürekli devam etmesini hedefler.
 - Gerçek kullanıcı verilerine dayalı testler yapılır.
 - Performans ve güvenlik testleri ön plandadır.

DevOps ve QA (Kalite Güvencesi) İlişkisi

✓ Test Otomasyonu

- Tüm testlerin otomatikleştirilmesi ile CI/CD süreçlerine entegre çalışmasını sağlar.

✓ Geri Bildirim Döngüsü

- Test sonuçları anında analiz edilir.
- Hatalar anında düzeltilerek tekrar test edilir.

✓ Güvenlik ve Uygulama Sağlığı

- Canlı ortama çıkan kodların sürekli güvenlik ve performans testlerinden geçirilmesini sağlar.

DevOps Sürecinde Kullanılan Test Otomasyon Araçları

Araç	Ne İşe Yarar?
Jenkins	CI/CD süreçlerinde sürekli entegrasyon sağlar.
Selenium	Web uygulamalarını test etmek için kullanılır.
Appium	Mobil uygulamalar için test otomasyonu sunar.
Kubernetes	Mikro hizmetlerin test ve dağıtım süreçlerini yönetir.
Docker	Test ortamlarını sanallaştırarak farklı platformlarda test imkanı sunar.
JMeter	Performans ve yük testlerini gerçekleştirir.

Örnek Senaryo

Bir online bankacılık uygulamasında, her yeni kod değişikliği CI/CD süreçleriyle nasıl test edilir?

✓ Sürekli Entegrasyon (CI) Kullanımı

- Kod değişiklikleri **Jenkins veya GitHub Actions** tarafından otomatik test edilir.
- Geliştiriciler kodlarını her yüklediğinde, testler anında çalıştırılır.

✓ Sürekli Teslimat (CD) ile Canlı Ortama Geçiş

- Tüm testleri geçen kod değişiklikleri otomatik olarak canlı ortama gönderilir.
- **Docker ve Kubernetes** ile mikro hizmetler yönetilir.

✓ Yük Testleri (JMeter Kullanımı)

- Yoğun banka işlemleri simüle edilir ve sistemin stabil çalışması sağlanır.

✓ Güvenlik Testleri

- Canlı ortama alınan kodlar **OWASP ZAP** ve **Burp Suite** gibi araçlarla güvenlik açısından test edilir.

Bu süreç sayesinde:

- Daha hızlı geliştirme süreçleri sağlanır.
- Hatasız ve güvenli yazılım teslimatı gerçekleşir.
- Sürekli güncellenen ve stabil çalışan uygulamalar oluşturulur.

Pratik Sorular

1. CI/CD'nin test sürecine etkisi nasıldır?
2. Shift-Left ve Shift-Right test yaklaşımlarının farkları nelerdir?
3. DevOps sürecinde otomasyonun önemi nedir?

Cevaplar

1. CI/CD, kodun sürekli test edilmesini ve hataların erken tespit edilmesini sağlar. Böylece geliştirme süreci hızlanır ve yazılımın kalitesi artar.
2. Shift-Left erken test yaparak hataları geliştirme sürecinde azaltmaya odaklanırken, Shift-Right canlı sistemlerdeki kaliteyi izleyerek hataları belirler.
3. Otomasyon, DevOps sürecinde testlerin sürekli entegrasyonla uyumlu çalışmasını sağlar ve manuel test yükünü azaltır.

44. MODERN TEST ARAÇLARI VE TEKNİKLERİ (MODERN TESTING TOOLS AND TECHNIQUES)

Modern Test Araçları

Yazılım test süreçlerinde kullanılan test araçları sürekli gelişmekte ve daha modern, hızlı ve etkili çözümler sunmaktadır. Günümüzde öne çıkan test araçları şunlardır:

Test Aracı	Kullanım Alanı	Avantajları
Cypress	Web UI Testleri	Hızlı, stabil, kolay entegrasyon
Playwright	Web UI Testleri	Farklı tarayıcı desteği, modern framework entegrasyonu
TestContainers	Entegrasyon Testleri	Hafif, Docker tabanlı test ortamları
Katalon Studio	API & Mobil Testler	Kodsuz test desteği, geniş özellikler
Postman	API Testleri	Kolay kullanım, entegrasyon testi desteği

Risk Bazlı Test Yönetimi ve Güncel Test Standartları (ISO/IEC 29119)

Risk Bazlı Test Yönetimi, yazılım test sürecinde en kritik alanlara öncelik verilerek risklerin minimize edilmesini sağlayan bir yaklaşımdır.

- ✓ Hata olasılığı yüksek alanlara öncelik verilir.
- ✓ Kullanıcıya doğrudan etki edebilecek bölümler daha fazla test edilir.
- ✓ Test kaynakları daha verimli kullanılır.

Risk Bazlı Test Yönetimi Adımları:

- Risklerin Belirlenmesi:** Hata olasılığı yüksek olan bileşenler analiz edilir.
- Önceliklendirme:** En kritik alanlar belirlenerek test önceliği verilir.
- Test Planının Oluşturulması:** Risk seviyelerine göre test stratejileri şekillendirilir.
- Test Sürecinin Yürütülmesi:** Belirlenen senaryolar öncelikli olarak test edilir.
- Sürekli İzleme ve Güncelleme:** Risk analizleri ve test süreçleri güncellenir.

ISO/IEC 29119 Test Standartları Nedir?

ISO/IEC 29119, uluslararası yazılım test standartlarını belirleyen bir çerçevedir.

- ✓ Test süreçlerini standartlaştırır.
- ✓ Test yönetimi ve test dokümantasyonu için en iyi uygulamaları sunar.
- ✓ Test mühendislerine rehberlik eder ve kaliteyi artırır.

Örnek Senaryo

Bir **finans uygulamasının** test süreçlerinde modern test araçlarını ve risk bazlı test yönetimini nasıl uygularız?

✦ Cypress Kullanımı:

- Kullanıcı giriş ekranı ve ödeme akışı için otomatik test senaryoları oluşturulur.

✦ TestContainers Kullanımı:

- API ve veri tabanı entegrasyonları için Docker tabanlı test ortamları oluşturulur.

✦ Risk Bazlı Test Yönetimi:

- Ödeme işlemleri kritik alan olarak belirlenir ve hata riski yüksek senaryolar (örneğin yanlış hesap numarası girme) öncelikli test edilir.

Sonuç:

- ✓ Test süreçleri hızlanır.
- ✓ Olası hatalar daha erken tespit edilir.
- ✓ Yazılımın güvenilirliği artar.

Pratik Sorular

1. Cypress ve Playwright gibi modern test araçlarının en büyük avantajı nedir?
2. Risk bazlı test yönetimi neden önemlidir?
3. ISO/IEC 29119 standardı yazılım test süreçlerine nasıl katkı sağlar?

Cevaplar

1. Cypress ve Playwright, modern web testlerinde büyük avantaj sağlar. Çünkü hızlıdır, kolay entegrasyon sunarlar ve farklı tarayıcı desteği sağlarlar.
2. Risk bazlı test yönetimi, en kritik ve yüksek hata riski taşıyan alanlara odaklanarak kaynakların daha verimli kullanılmasını sağlar ve yazılımın güvenilirliğini artırır.
3. ISO/IEC 29119 standardı, test süreçlerini uluslararası düzeyde standartlaştırarak, test yönetimi, test süreçleri ve test dokümantasyonunun en iyi uygulamalar çerçevesinde yürütülmesini sağlar.

45. SÜREKLİ TEST (CONTINUOUS TESTING) VE SHIFT-LEFT TESTING YAKLAŞIMLARI

Sürekli Test (Continuous Testing) Nedir?

Sürekli Test, yazılım geliştirme sürecinde her aşamada testlerin otomatik olarak çalıştırılmasını ve kalite değerlendirmesinin sürekli yapılmasını sağlayan bir yaklaşımdır.

- CI/CD süreçleriyle uyumludur.
- Test otomasyonu kullanarak hızlı hata tespiti sağlar.
- Yazılımın güvenilirliğini ve kalitesini artırır.

Sürekli Testin Temel Faydaları

✦ Hataların Erken Tespiti

- Kod değişiklikleri anında test edilir, hatalar hızlıca düzeltilir.

✦ CI/CD Süreçleriyle Uyum

- Otomasyon testleri sürekli entegrasyon süreçlerine entegre edilir.

✦ Manuel Test Yükünü Azaltır

- Otomasyon sayesinde test süreçleri hızlanır.

✦ Gerçek Kullanıcı Simülasyonu

- Performans ve yük testleri sürekli olarak yapılır.

✦ Daha Kaliteli ve Güvenilir Yazılım

- Hatalar erken tespit edilir, test süreçleri iyileştirilir.

Shift-Left Testing Nedir?

Shift-Left Testing, test süreçlerini yazılım geliştirme sürecinin erken aşamalarına kaydırarak hataların daha erken tespit edilmesini sağlayan bir yaklaşımdır.

- Geleneksel testlerde testler geliştirme sürecinin sonunda gerçekleştirilir.
- Shift-Left ile testler erken aşamada başlar ve geliştiricilerle test mühendisleri birlikte çalışır.

✦ Shift-Left Testing'in Avantajları

- **Erken Test, Düşük Maliyet:** Hatalar erken aşamada bulunur ve maliyet azalır.
- **Sürekli Geri Bildirim:** Geliştiriciler hızlı iyileştirmeler yapabilir.
- **Geliştirme ve Test Entegrasyonu:** Test mühendisleri ve geliştiriciler birlikte çalışır.
- **Daha Hızlı Dağıtım:** Erken test edilen yazılım, hızlıca üretime alınır.

Örnek Senaryo:

E-Ticaret Uygulaması için Sürekli Test & Shift-Left Testing

Bir e-ticaret uygulaması geliştiriyoruz ve test süreçlerini aşağıdaki gibi uygulayabiliriz:

✦ Sürekli Test:

- **CI/CD süreçlerine entegre edilen Cypress veya Selenium** kullanılarak her kod değişikliğinde testlerin otomatik çalıştırılması.

✦ Shift-Left Testing:

- Geliştiriciler, kodlarını yazdıktan sonra hemen birim testlerini çalıştırır.
- Kod değişiklikleri, CI/CD süreçleriyle anında test edilir.

Bu süreç sayesinde:

- ♠ Hatalar erken tespit edilir.
- ♠ Yazılım geliştirme süreci hızlanır.
- ♠ Test maliyetleri azalır.

Pratik Sorular

1. Sürekli testin CI/CD süreçleriyle entegrasyonu nasıl sağlanır?
2. Shift-Left Testing yaklaşımının en büyük avantajı nedir?
3. Sürekli test sürecinde otomasyonun rolü nedir?

Cevaplar

1. Sürekli test, CI/CD süreçlerine entegre edilen test otomasyon araçlarıyla sağlanır ve her kod değişikliğinde otomatik test çalıştırılır.
2. Shift-Left Testing, testleri erken aşamaya kaydırarak hataların düşük maliyetle düzeltilmesini ve yazılımın hızlı dağıtımını sağlar.
3. Sürekli test sürecinde otomasyon, manuel test yükünü azaltarak testlerin hızlı ve güvenilir çalıştırılmasını sağlar, hataların anında tespit edilip düzeltilmesine imkan verir.

