Sistem Testi

Klasik Sınav Soruları ve Cevap Anahtarı

Bu PDF, kullanıcı tarafından sağlanan soru-cevap içeriğinin düzenlenmiş hâlidir.

Tarih: 6 Ocak 2026

Bölüm 1 — Sistem Testi Konulu Klasik Sorular (Soru 1-10)

Soru 1: Sistem testinin temel amacını ve onu birim/entegrasyon testlerinden ayıran en temel farkı açıklayınız.

Sistem testinin temel amacı, yazılım ürününü "bir bütün olarak" ele alarak tanımlanmış gereksinimleri, kullanıcı beklentilerini ve kalite standartlarını ne ölçüde karşıladığını uçtan uca senaryolarla doğrulamaktır. Birim ve entegrasyon testlerinden farkı; tek tek bileşenlerin veya modüllerin teknik doğruluğuna değil, tüm sistemin bir araya geldiğinde sergilediği bütünsel davranışa ve gerçek hayat senaryolarına odaklanmasıdır.

Soru 2: Sistem testinde "Kara Kutu" (Black-box) yaklaşımı ne anlama gelir? Bu yaklaşımın test mühendisine sağladığı avantaj nedir?

Kara kutu yaklaşımında test mühendisi sistemin iç kod yapısını, sınıfları veya algoritmaları bilmek zorunda değildir; sadece dışarıdan verilen girdilere karşılık sistemin ürettiği çıktılara ve gözlemlenebilir davranışlara odaklanır. Bu yaklaşımın en büyük avantajı, testi kullanıcı bakış açısına ve iş gereksinimlerine yaklaştırmasıdır. Ayrıca, yazılımın iç kodlaması değişse bile (implementasyon değişikliği), sistemin dış davranışı aynı kaldığı sürece testlerin geçerliliğini korumasını sağlar.

Soru 3: Fonksiyonel ve fonksiyonel olmayan gereksinimler arasındaki farkı birer örnekle açıklayınız.

Fonksiyonel gereksinimler sistemin "ne yapması gerektiğini" (örn: "Kullanıcı geçerli bilgilerle giriş yapabilmelidir"), fonksiyonel olmayan gereksinimler ise sistemin bu işi "nasıl" yapması gerektiğini (örn: "Sistem 1000 eşzamanlı kullanıcıyı desteklemelidir" veya "Veriler şifrelenmelidir") tanımlar.

Soru 4: Sistem test sürecinde "Giriş (Entry)" ve "Çıkış (Exit)" kriterleri neden tanımlanır? Her biri için ikişer örnek veriniz.

Bu kriterler, testin ne zaman başlayıp ne zaman başarıyla tamamlanmış sayılacağına dair nesnel sınırlar çizmek ve tartışmaları azaltmak için tanımlanır.
Giriş Kriteri Örnekleri: Entegrasyon testlerinin tamamlanmış olması, test ortamının hazır olması.
Çıkış Kriteri Örnekleri: Planlanan senaryoların tamamının yürütülmesi, kritik seviyede açık hata kalmaması.

Soru 5: "Risk Temelli Sistem Testi" yaklaşımı nedir? Sınırlı kaynaklara sahip bir projede bu yaklaşım nasıl bir fayda sağlar?

Risk temelli yaklaşım, test eforunun hatanın gerçekleşme olasılığı ve yaratacağı etkinin büyüklüğüne göre en kritik alanlara yönlendirilmesidir. Sınırlı kaynakların (zaman, bütçe, personel) olduğu projelerde, tüm senaryoları test etmek imkansız olduğu için bu yöntem en kritik hataların en erken aşamada yakalanmasını ve kaynakların en yüksek faydayı sağlayacak şekilde kullanılmasını sağlar.

Soru 6: Bir hata (defect) kaydında bulunması gereken temel unsurlar nelerdir? İyi raporlanmamış bir hata kaydı süreci nasıl etkiler?

Bir hata kaydında benzersiz kimlik (ID), açıklayıcı başlık, yeniden üretme adımları, beklenen ve gözlenen sonuçlar, ortam bilgileri (cihaz, tarayıcı vb.) ve varsa log/ekran görüntüleri bulunmalıdır. İyi raporlanmamış hatalar geliştiricilerin hatayı anlayamamasına veya yeniden üretememesine, bu da sürecin tıkanmasına ve zaman kaybına yol açar.

Soru 7: Hata yönetimindeki "Şiddet" (Severity) ve "Öncelik" (Priority) kavramları arasındaki farkı açıklayınız.

Şiddet, hatanın teknik açıdan sistem üzerinde yarattığı etkinin ciddiyetini (örn: sistemin çökmesi veya veri kaybı) ifade eder. Öncelik ise hatanın ne kadar acil düzeltilmesi gerektiğini (örn: canlıya çıkıştan önce mutlaka giderilmesi gerekenler) gösterir. Bazen teknik şiddeti düşük ama kullanıcı deneyimini çok bozan bir hata yüksek önceliğe sahip olabilir.

Soru 8: Test ortamının canlı (production) ortama benzerliğinin kritik olmasının sebepleri nelerdir?

Donanım kapasitesi, veritabanı boyutu ve ağ yapılandırması gibi unsurlar sistemin gerçek davranışını doğrudan etkiler. Eğer test ortamı canlıdan çok farklıysa, sistem testinde alınan başarılı sonuçlar canlı ortamda aynı şekilde gerçekleşmeyebilir ve performans veya uyumluluk sorunları ancak canlıya çıkıldığında fark edilebilir.

Soru 9: Manuel testler ile otomatik testlerin hangi durumlarda tercih edilmesi gerektiğini kısaca karşılaştırınız.

Manuel testler; karmaşık kullanıcı arayüzleri, görsel doğrulama gerektiren durumlar ve ilk kez denenen keşifsel testler için vazgeçilmezdir. Otomatik testler ise özellikle sık tekrarlanan regresyon testlerinde, insan hatasını azaltmak ve testleri çok daha kısa sürede, tekrarlı bir şekilde yürütmek için tercih edilir.

Soru 10: "Test Verisi" tasarımı yapılırken sadece "mutlu yol" (happy path) verilerinin kullanılmamasının nedeni nedir? Hangi veri türleri de dikkate alınmalıdır?

Sadece mutlu yol verileri kullanıldığında, sistemin gerçek hayatta karşılaşacağı hatalı girişlere veya uç durumlara nasıl tepki vereceği ölçülemez. Bu nedenle geçerli verilerin yanı sıra geçersiz veriler, sınır değerler ve büyük hacimli veri senaryoları da test setine dahil edilmelidir.

Bölüm 2 — İleri Seviye Klasik Sorular (Soru 11-20)

Soru 11: "Test Piramidi" kavramını açıklayınız ve sistem testi açısından bu piramidin hangi kısımları daha kritiktir?

Test piramidi, testlerin miktarını ve seviyesini gösteren bir modeldir; tabanda çok sayıda birim testi, ortada daha az sayıda entegrasyon/API testi ve tepede en az sayıda ama kritik UI uçtan uca (E2E) testleri yer alır. Sistem testi perspektifinden bakıldığında piramidin özellikle orta ve üst kısmı (API ve UI E2E testleri) kritik önem taşır çünkü bu seviyeler sistemin bütünsel davranışını temsil eder.

Soru 12: Sistem testlerinde "her şeyi otomatikleştirmek" neden yanlış bir hedeftir? Manuel testin vazgeçilmez olduğu bir alanı örnekleyiniz.

Otomasyonda asıl sanat neyi manuel, neyi otomatik yapacağına bilinçli karar vermektir; her şeyi otomatikleştirmek maliyetli ve verimsiz olabilir. Manuel testler; özellikle ilk kez geliştirilen özelliklerin keşifsel testleri, kullanılabilirlik (UX) değerlendirmeleri ve karmaşık raporlama sonuçlarını yorumlama gibi "insan muhakemesi" gerektiren alanlarda vazgeçilmezdir.

Soru 13: Otomasyona en uygun sistem testi senaryolarının temel özellikleri nelerdir?

Sık tekrarlanan regresyon senaryoları, kritik iş akışlarının "mutlu yol" (happy path) ve temel hata senaryoları, API seviyesindeki karmaşık iş kuralları ile performans ve yük testleri otomasyon için en uygun alanlardır.

Soru 14: Sistem testi kapanış aşamasında hazırlanan "Test Özet Raporu"nda hangi temel metrikler yer almalıdır?

Bu raporda; planlanan ve yürütülen test durumlarının sayısı, başarılı/başarısız test dağılımı, bulunan hataların sayısı ve şiddet dağılımları, hataların modüllere göre dağılımı ve gözlenen genel kalite durumu gibi metrikler bulunmalıdır.

Soru 15: Bir yazılım sürümü için "Kabul" veya "Red" kararı verilirken sadece testlerin geçme oranına bakılmamasının nedeni nedir?

Çünkü sadece sayılar (geçti/kaldı) sistemin riskini tam yansıtmaz. Karar verilirken açık kalan hataların şiddeti, bu hataların iş süreçlerine ve güvenliğe olası etkileri ile canlı ortamda ortaya çıkabilecek risklerin kabul edilebilirliği gibi unsurlar da dikkate alınmalıdır.

Soru 16: Test sürecinin sonunda yapılan "Öğrenilen Dersler" (Lessons Learned) faaliyetinin yazılım geliştirme yaşam döngüsüne katkısı nedir?

Bu faaliyet; nelerin iyi gittiğini, nelerin efor kaybına yol açtığını ve hataların daha erken aşamalarda nasıl yakalanabileceğini belirleyerek hem test sürecinin olgunlaşmasını sağlar hem de gelecekteki projelerin daha verimli planlanmasına katkı sunar.

Soru 17: Regresyon testlerinin sistem testindeki temel amacı nedir ve neden önemlidir?

Regresyon testleri, sistem üzerinde yapılan yeni geliştirmelerin veya değişikliklerin, mevcut ve çalışan fonksiyonları olumsuz etkilemediğini (bozmadığını) doğrulamak için yapılır. Sistemin sürekliliğini ve kararlılığını korumak için kritiktir.

Soru 18: Keşifsel testler (Exploratory Testing) ile senaryo tabanlı testler arasındaki temel fark nedir?

Senaryo tabanlı testler önceden yazılmış adım adım yönergelere dayanırken, keşifsel testlerde test mühendisi o anki öğrenme, test tasarımı ve yürütme faaliyetlerini eş zamanlı yürütür. Keşifsel testler, yapılandırılmış senaryoların gözden kaçırabileceği beklenmedik hataları bulmakta çok etkilidir.

Soru 19: Fonksiyonel olmayan sistem testleri neleri kapsar? Üç örnek veriniz.

Fonksiyonel olmayan testler sistemin "nasıl" çalıştığına odaklanır. Örnekler:
Performans Testi: Sistemin belirli bir yük altında nasıl tepki verdiği.
Güvenlik Testi: Verilerin korunması ve yetkisiz erişim kontrolleri.
Kullanılabilirlik Testi: Yazılımın kullanıcı dostu olup olmadığı.

Soru 20: "Test İzlenebilirliği" (Traceability) ne anlama gelir ve neden bir matris ile takip edilir?

Test izlenebilirliği, her bir gereksinimin hangi test durumuyla doğrulandığını takip etmektir. Bir "İzlenebilirlik Matrisi" kullanılarak, "Her gereksinim gerçekten test edildi mi?" sorusuna yanıt verilir ve test kapsamındaki eksiklikler kolayca tespit edilir.

Bölüm 3 — Derinlemesine Klasik Sorular (1-52)

1): Sistem testinin amacı, “yazılımın çalışması” ile “doğru şeyi doğru şekilde yapması” ayrımında nasıl konumlanır?

Sistem testi yazılımı bir bütün olarak ele alır; tek tek fonksiyonların çalışmasından ziyade, modüller birleşince ortaya çıkan davranışın iş gereksinimleri ve kullanıcı beklentileri ile uyumunu ölçer. Yani “buton çalışıyor mu?” değil, “butona basınca tüm akış doğru iş kuralıyla, doğru veriyle, doğru yan etkilerle tamamlanıyor mu?” sorusuna odaklanır. Örnek: Otoparkta sadece “ücret hesaplama fonksiyonu” değil; abonelik + süre + tarife + hata durumları birlikte ele alınarak uçtan uca doğru ücretlendirme doğrulanır.

2): Sistem testinde “uçtan uca senaryo” neden kritik bir ölçüm noktasıdır?

Çünkü entegrasyon seviyesinde fark edilmeyen sorunlar, gerçek kullanımda belirli sırayla çağrılan bileşenler, yük/ veri hacmi, hata durumları altında ortaya çıkar. Uçtan uca senaryolar; UI/iş mantığı/veri katmanı ve dış servis etkileşimlerini birlikte doğrular; canlıya çıkış riskini azaltır.

3): Sistem testi kapsamı (scope) belirlenmezse hangi iki problem kaçınılmaz olur?

Odak kaybı: Test ekibi kritik akış yerine “her yere biraz dokunma”ya gider; kritik hatalar kaçabilir.
Beklenti yönetimi krizi: Paydaşlar “şu neden test edilmedi?” diye tartışır; çünkü in-scope/out-of-scope net değildir. Kapsam yazılı olmalı; hangi modül bu sürümde değiştiyse o akışların senaryoları derin test edilmelidir.

4): “In-scope / out-of-scope” kararını verirken teknik olarak en doğru yaklaşım nedir?

Değişiklik yapılan alanları ve riskli noktaları (kritik iş akışları) in-scope yapıp; değişmeyen, daha önce yeterince test edilmiş modülleri gerekçesiyle out-of-scope bırakmaktır. Ama out-of-scope kararının gerekçesi açık yazılmalıdır; yoksa test sonunda “eksik test” eleştirisi belirsiz kalır.

5): Sistem sınırı (system boundary) net değilse test tasarımında hangi hata yapılır?

Dış sistemlerin sorumluluğu ile kendi sistemimizin sorumluluğu karışır. Örn. ödeme akışında bankanın onayı dış sistem ise; sistem testi kendi sınırında olan davranışı doğrular; dış sistemler için stub/mock veya sandbox kullanır. Sınır net değilse ya dış sistemi “test ettik sanırız” ya da gereksiz yere gerçek sistem bağımlılığı kurarız.

6): Fonksiyonel ve fonksiyonel olmayan gereksinimleri sistem testinde nasıl birlikte ele alırsın?

Fonksiyonel gereksinim “ne yapmalı?”dır (giriş yapabilmeli, ücret hesaplamalı). Fonksiyonel olmayan gereksinim “nasıl davranmalı?”dır (performans, güvenlik, kullanılabilirlik). Sistem testi fonksiyoneli uçtan uca doğrularken; fonksiyonel olmayanı da çoğu zaman ek test türleriyle (yük, güvenlik, kullanılabilirlik testleri) destekler.

7): “Sistem hızlı olmalı” gereksinimi neden test edilemez? Test edilebilir hâle nasıl getirirsin?

“Hızlı” ölçülebilir değildir; herkesin algısı farklıdır. Test edilebilir yapmak için ölçü konur: “Ortalama cevap süresi 2 sn altında”, “1000 eşzamanlı kullanıcı” gibi net eşikler tanımlanır. Böylece performans/yük testinde pass/fail kararı üretilebilir.

8): Kara kutu yaklaşımı sistem testinde ne demektir? “Yan etki” örnekleri ver.

Testçi iç kodu bilmeden; girdi → çıktı ve gözlemlenebilir davranış üzerinden değerlendirir. Yan etkilere örnek: DB’de kayıt oluşması, e-posta/SMS gönderimi, log kaydı, üçüncü tarafa çağrı, ekranda hata mesajı, yönlendirme.

9): Kara kutu yaklaşımında “iç yapıyı hiç bilmemek” doğru mu?

Tamamen değil. İç mimariyi detay kod düzeyinde bilmek şart değil; ama sistem sınırları, veri akışları, entegrasyon noktalarını bilmek daha anlamlı senaryolar tasarlatır. Testler yine implementasyona bağımlı olmadan yazılmalıdır.

10): Sistem testini “kullanıcı odaklı” yapan şey nedir? Bankacılık transferi örneğiyle açıkla.

Sistem testi, kullanıcıların gerçek iş akışını (use case) temel alır. Transfer örneğinde sadece “transfer kaydı oluştu mu?” değil; giriş → alıcı seçimi → tutar → doğrulama → hatalı giriş uyarıları → başarılı işlem bildirimi → hesap özetinde görünmesi gibi zincirin tamamı doğrulanır.

11): Kullanıcı hatalarına dayanıklılık sistem testinde nasıl ölçülür? 3 örnek durum ver.

Senaryo temelli negatif testlerle:
Zorunlu alan boş bırakma → doğru ve anlaşılır uyarı vermeli
Yanlış format (e-posta/parola) → doğru doğrulama mesajı
Ağ kopması → kullanıcı deneyimi bozulmadan “graceful” hata yönetimi (işlem tekrarı/geri alma)

12): Risk temelli sistem testinde “risk” nasıl tanımlanır? İki bileşeni nedir?

Risk = hata olasılığı × hata etkisi. Etki; veri kaybı, itibar zedelenmesi, finansal zarar, güvenlik açığı gibi sonuçları kapsar.

13): Risk temelli yaklaşımda yüksek riskli alanlar için test seti nasıl farklılaştırılır?

Yüksek riskli alanlarda daha fazla senaryo, daha fazla sınır/hata durumu ve mümkünse otomasyonla sürekli regresyon yapılır. Orta/düşük riskte daha sınırlı kapsam (tipik kullanım) yeterli görülebilir.

14): Risk temelli testin “şeffaflık ve hesap verebilirlik” katkısı nedir?

Test planında “neden bu alanlar yoğun test edildi, neden bazıları daha az?” gerekçesi yazılı olur. Bir hata kaçınca “neden yakalanmadı?” sorusu, risk analizi ve kapsam kararları üzerinden rasyonel açıklanabilir.

15): Test planlamada hedefler ve başarı kriterleri neden ayrı kavramlardır?

Hedef “ne doğrulayacağız?”dır (ör. yeni ücret modülü doğru mu). Başarı kriteri “test sonunda ne olursa başarılı sayarız?”dır (ör. %95 senaryo çalıştı, kritik açık hata kalmadı gibi). Başarı kriteri yoksa test bitişi sürekli tartışma olur.

16): Entry kriterleri niçin “sistem testi başlamadan önce” şarttır? 3 örnek ver.

Sistem testi, hazır olmayan build/ortamda çalışırsa sahte hatalar üretir. Örnek entry kriterleri: entegrasyon testleri geçti, test ortamı kurulu, smoke testler başarılı, test verisi hazır.

17): Exit kriteri yanlış seçilirse ne olur? “Sayıya bakma” hatasını anlat.

Sadece “kaç test geçti?” sayısına bakarsan; kritik şiddette açık hata kalmasına rağmen çıkış kararı verilebilir. Exit; yürütme yüzdesi + açık hata şiddeti + kritik senaryoların kabulü gibi çok boyutlu olmalıdır.

18): Zaman-kaynak-bütçe planlamasında “kapsam” ile nasıl gerçekçi ilişki kurulur?

Kısa sürede çok geniş kapsam hedeflemek yüzeysel teste yol açar. Çözüm: risk temelli önceliklendirme ile kritik alanlara yoğunlaşmak ve otomasyonu regresyon için kullanmak.

19): Gereksinimlerin “test edilebilirlik” incelemesi sırasında hangi 4 özellik aranır?

Gereksinimlerin açık, çelişkisiz, tutarlı ve test edilebilir olması aranır. Muğlak ifadeler netleştirilmezse test tasarımı subjektif olur.

20): İzlenebilirlik matrisi (Requirements → Test Cases) neyi garanti eder? Yoksa ne olur?

Her gereksinimin en az bir test ile ilişkilendiğini gösterir. Yoksa “gözden kaçan gereksinimler” oluşur; test sonunda “bu gereksinim test edilmedi” fark edilmez.

21): Test senaryosu ile test durumu (test case) arasındaki farkı örnekle açıkla.

Senaryo uçtan uca akıştır (“kayıt ol → giriş yap → park başlat”). Test case senaryonun daha detaylı parçasıdır: ID, ön koşul, adımlar, beklenen sonuç, son koşul. Detay, tekrarlanabilirliği sağlar.

22): İyi bir test case’in “son koşul / beklenen sistem durumu” kısmı neden kritiktir?

Sadece ekrandaki mesajı değil, sistemin kalıcı durumunu doğrular (DB’de kayıt, oturum state’i, log, dış çağrı). Yoksa “görüntü doğru ama veri yanlış” hataları kaçabilir.

23): Test verisi tasarımında neden sadece “happy path” yetmez? 5 veri türü say.

Gerçekte invalid girişler, sınır değerler, kombinasyonlar ve büyük veri hacimleri sık görülür. Bu yüzden: valid, invalid, sınır değer, kombinasyon (tarife/süre/abonelik), büyük hacimli veri senaryoları hazırlanır.

24): Test ortamı hazırlanırken dokümante edilmesi gereken 4 şey nedir?

Donanım/altyapı (sunucu/konteyner), işletim sistemi ve sürümleri, DB sürümleri, ağ konfigürasyonu (IP aralıkları, güvenlik duvarı, load balancer). Dokümantasyon yoksa performans/bağlantı/uyumluluk sorunları kök neden analizinde kaybolur.

25): Stub/Mock kullanma kararı hangi durumda avantajdır, hangi durumda risk doğurur?

Dış sisteme erişim yoksa veya maliyet/güvenlik riski varsa avantajdır. Risk: Simülasyon gerçek sistemi yeterince temsil etmezse “test geçti sanıp” canlıda patlar. Bu yüzden simülasyon davranışı gerçek sistemle uyumlu tasarlanmalıdır.

26): Test ortamının canlıya benzer olmaması hangi iki büyük hatayı gizler?

Performans sorunları: Testte hızlı çalışan sorgu, canlıda milyon kayıtla timeout olabilir.
Güvenlik/ağ kısıtları: Testte serbest erişim, canlıda firewall ile engellenebilir.

27): Pre-prod/Staging ortamı niçin “kritik testler” için önerilir?

Çünkü test ortamı ile canlı arasındaki farklar sonuçları yanıltabilir; staging/pre-prod canlıya daha yakın bir ikinci ortam olarak risk azaltır.

28): Manuel sistem testi yürütme adımlarını 6 maddede yaz ve her adımın amacı ne?

Ön koşulları sağla (testin doğru zeminde koşması)
Adımları uygula (senaryoyu işletme)
Çıktı/yan etkiyi gözlemle (gerçek davranışı yakala)
Beklenenle karşılaştır (pass/fail)
Tutarsızlıkta hata kaydı aç (izlenebilirlik)
Sonucu test yönetim aracına işle (raporlama)

29): Otomatik sistem testlerinin regresyonda kritik olmasının 3 nedeni nedir?

Aynı testlerin her sürümde tekrar koşulması, insan hatasının azalması, hız/ölçek avantajı. Regresyonu sürdürülebilir yapar.

30): Test sonuçlarının kayıt altına alınmasında hangi bilgiler zorunlu tutulmalıdır?

Testi kim/ne zaman koştu, sonuç (pass/fail/blocked), başarısızsa ilgili hata kaydı linki, ek kanıtlar (log/screenshot). Bu olmadan geçmişe dönük izlenebilirlik kurulamıyor.

31): Hata raporlama formunda bulunması gereken alanları say ve “neden”ini açıkla.

ID, açıklayıcı başlık, yeniden üretme adımları, beklenen/gözlenen sonuç, ortam bilgisi, log/screenshot. Neden: Geliştirici hatayı üretemezse süreç tıkanır; hata “anlaşılamadığı için” kapanamaz.

32): Hata yaşam döngüsünde “Resolved” ile “Closed” farkı nedir?

Resolved: geliştirici “düzeltildi” der. Closed: testçi retest yapar, gerçekten düzeldiğini doğrular ve kapatır. Aradaki fark doğrulama sorumluluğudur.

33): “Rejected/Not a Bug” kararı hangi iki durumda makuldür?

1) Davranış gereksinimlere uygundur (testçi yanlış yorumlamıştır). 2) Tekrar üretilemez ve kanıt yoktur (ortam/veri belirsiz).

34): Severity ve Priority aynı şey midir? Örnekle çelişen bir durum anlat.

Değildir. Severity teknik etkidir; Priority aciliyettir. Örn. düşük severity (UI yazım hatası) ama ana akışta her kullanıcıyı etkiliyorsa priority yüksek olabilir. Ya da yüksek severity bir hata, geçici iş kuralıyla idare edilebiliyorsa priority bir tık düşebilir.

35): Test kapanışında neden sadece “testler bitti” demek yetmez?

Çünkü amaç; hedeflere ulaşıldı mı, riskler azaldı mı, sistem bir sonraki aşamaya hazır mı sorularına yanıt vermektir. Bu değerlendirme olmadan kabul/erteleme kararı rastgeleleşir.

36): Test özet raporunda hangi metrikler yer almalı ve neye yarar?

Planlanan/yürütülen test sayısı, pass/fail dağılımı, hata sayısı ve şiddet dağılımı, modüle göre dağılım, genel kalite yorumu. Yönetim ve iş birimleri “teslime hazır mı?” kararını bu metriklerle verir.

37): Kabul/Red kararında “açık kalan hatalar” nasıl yorumlanmalı?

Sadece sayıya değil; şiddete, kapsama, iş süreçlerine etkisine, güvenliğe etkisine bakılır. Bazen risk kabul edilerek canlıya çıkılır; bazen kritik hatadan dolayı sürüm ertelenir.

38): Lessons Learned toplantısının sistem testine katkısı nedir?

Sadece hata kapatmak değil; bir sonraki döngüde daha iyi plan/ortam/veri/tasarım üretmektir. “Hangi hatalar daha erken yakalanabilirdi? Ortamda ne iyileşmeli?” gibi geri bildirimle süreç olgunlaşır.

39): Gereksinim temelli fonksiyonel testte test case nasıl “gereksinime bağlanır”? Basit bir örnek ver.

Gereksinim ID (örn. LOGIN-01) ile test case eşleştirilir; test adımları doğrudan gereksinimi doğrular. Bu izlenebilirlik, “şu gereksinim test edildi mi?” sorusuna net cevap üretir.

40): Use case tabanlı testlerde bir testin aynı anda birden fazla gereksinimi sınaması neden normaldir?

Çünkü use case akışı doğası gereği birden fazla fonksiyonu birleştirir (kayıt + aktivasyon + login). Sistem testinde amaç; akışın kullanıcı bakış açısıyla çalışmasıdır.

41): End-to-end testlerin “en geniş kapsamlı” olmasının iki bedeli nedir?

Kurulum/ortam bağımlılığı artar (UI + servisler + DB + dış sistemler).
Hata izolasyonu zorlaşır (hangi bileşen bozdu?).
Bu yüzden E2E yanında daha hedefli testler de gerekir.

42): Performans & yük testinde hangi iki metrik birlikte anlamlıdır? Neden tek metrik yetmez?

Cevap süresi + kaynak kullanımı (CPU/RAM/IO) birlikte anlamlıdır. Çünkü “2 sn altı” görünen sistem, CPU’yu %100’e vurup kısa süre sonra çökebilir; dayanıklılık sorunu çıkar.

43): Stres testi ile dayanıklılık (endurance) testinin farkını gerçekçi örnekle açıkla.

Stres: kapasitenin üstüne çıkarıp kırılma noktasını ve kontrollü hata davranışını görmek. Dayanıklılık: 8-24 saat orta yükte bellek sızıntısı/perf düşüşü var mı diye bakmak.

44): Sistem testinde güvenlik testleri “ne kadar ileri” götürülebilir? Sınır nerede?

Sistem testi içinde kimlik doğrulama/yetkilendirme senaryoları, yetkisiz erişim denemeleri, girdi doğrulama kontrolleri yapılır. Daha ileri sızma/dinamik tarama gibi işler genelde ayrı güvenlik uzmanlığı ve araç gerektirir.

45): Uyumluluk ve taşınabilirlik testleri en çok hangi ürünlerde “kritik” olur?

Web ve mobil uygulamalarda: farklı tarayıcı/OS/cihaz ekranı kombinasyonlarında tutarlılık gerekir. Taşınabilirlik, on-prem → bulut gibi altyapı değişimlerinde önem kazanır.

46): Güvenilirlik ve hata toleransı testinde “graceful degradation” ne demektir?

Dış servis yokken sistemin tamamen çökmesi yerine; anlaşılır hata verip veri bütünlüğünü koruması ve mümkünse otomatik iyileşme (self-healing) davranışı göstermesidir.

47): Regresyon testi neden “sistem testinin her döngüsüyle” bağlantılıdır?

Her sürümde yeni özellik/düzeltme/altyapı değişimi, çalışan fonksiyonları bozabilir. Regresyon “yan etkiyi” yakalamak için kritik senaryoları tekrar koşturur.

48): Tam regresyon - seçimli regresyon - risk temelli regresyon stratejilerini kıyasla.

Tam regresyon: her şeyi her sürümde; pahalı (otomasyonsuz zor).
Seçimli: değişen modüllere temas eden kritik akışlar.
Risk temelli: yüksek riskli alanlar her sürüm; düşük riskli alanlar daha seyrek.

49): Keşifsel test, planlı testlerin yerini tutar mı? Doğru denge nedir?

Tutmaz; izlenebilirlik/tekrar edilebilirlik zayıflar. Denge: ana iskelet planlı testler (gereksinim/senaryo), riskli yeni alanlarda süreli keşifsel oturumlar. Keşifte bulunan önemli akışlar sonradan kalıcı test case'e dönüştürülür.

50): Test verisi türleri: sentetik veri - maskeleme - anonimleştirme. Farkları ve riskleri nelerdir?

Sentetik veri: Tamamen yapay; gizlilik riski yok ama gerçek dağılımı/istisnaları iyi yansıtmayabilir.
Maskeleme: Canlı verinin hassas alanlarını değiştirir; yapı/format korunur ama bazen "orijinale dair iz" kalabilir.
Anonimleştirme: Kişiyle ilişki geri döndürülemez şekilde koparılır; maskeden daha ileri düzeydir. Testte çoğu zaman birlikte kullanılır (anonim canlı kopya + edge case için sentetik ek veri).

51): KVKK/GDPR bağlamında test ortamı neden "en zayıf halka" olabilir?

"Nasıl olsa test" diye güvenlik politikaları gevşetilirse canlıdan direkt veri kopyalanır, erişimler genişler, log/backup'ta hassas veri kalır. Bu yüzden canlı veri zorunluysa önce anonimleştirme/maskeleme, erişimi kısıtlama ve log/backup kontrolü yapılmalıdır.

52): Pseudonym (takma veri) kullanmak neyi çözer, neyi çözmez?

Çözer: Gerçek kişiyi doğrudan tanımlamayı zorlaştırır (isim/TC/telefon yerine takma değer). Çözmez: Eğer ilişkilendirme anahtarları/yan veriler duruyorsa yeniden kimliklendirme riski kalabilir; bu yüzden anonimleştirme gerektiren durumlarda pseudonym tek başına yetmeyebilir.

Bölüm 4 — Süreçler ve Yönetim (Soru 1-10)

Soru 1: Bir sistem test planında "Başarı Kriterleri" (Success Criteria) neler olabilir? Üç örnek vererek açıklayınız.

Başarı kriterleri, test sürecinin hedeflerine ulaşıp ulaşmadığını ölçen somut göstergelerdir. Örnekler:
Planlanan test senaryolarının en az %95'inin başarıyla yürütülmüş olması.
Kritik ve yüksek şiddetli açık hatanın kalmaması.
Toplam hata sayısının belirlenmiş bir eşik değerin altında olması.

Soru 2: Gereksinim İzlenebilirlik Matrisi (Requirements Traceability Matrix - RTM) nedir ve sistem testinde neden kullanılır?

RTM, her bir gereksinimin en az bir test durumuyla (test case) ilişkilendirildiği bir matristir. Temel amacı, "Bu gereksinim gerçekten test edildi mi?" sorusuna yanıt vermek ve test kapsamındaki eksikliklerin kolayca tespit edilmesini sağlamaktır.

Soru 3: Test verisi tasarımı yapılırken dikkate alınması gereken dört temel veri türünü belirtiniz.

Testin kalitesini artırmak için şu veri türleri kullanılmalıdır:
Geçerli (valid) veriler: Sistemin normalde kabul etmesi gereken veriler.
Geçersiz (invalid) veriler: Sistemin hata vermesi beklenen hatalı girişler.
Sınır değerler (boundary values): Kabul ve red sınırındaki uç değerler.
Büyük hacimli veriler: Sistemin yük altındaki davranışını ölçmek için kullanılan veri setleri.

Soru 4: Sistem testinde "Stub" ve "Mock" bileşenler hangi amaçla kullanılır? Bir örnekle açıklayınız.

Bu bileşenler, sistemin entegre olduğu ancak test anında ulaşılamayan veya kullanımı maliyetli olan dış sistemleri simüle etmek/taklit etmek için kullanılır. Örneğin, gerçek bir banka ödeme sistemine erişim mümkün değilse, bankanın beklenen onay mesajını taklit eden bir simülasyon (mock) bileşeni kullanılır.

Soru 5: Manuel sistem testi yürütme adımlarını sırasıyla yazınız.

Süreç şu adımlardan oluşur:
Test durumunun ön koşullarını sağlama.
Test adımlarını sırasıyla uygulama.
Sistemin çıktılarını ve yan etkilerini gözlemleme.
Gözlenen sonuçları beklenen sonuçlarla karşılaştırma.
Hata varsa kayıt oluşturma, yoksa test durumunu "geçti" olarak işaretleme.

Soru 6: Hata Yaşam Döngüsündeki (Bug Lifecycle) "Resolved" (Çözüldü) ve "Closed" (Kapatıldı) statüleri arasındaki fark nedir?

"Resolved" statüsü, geliştiricinin hatayı düzelttiğini ve ilgili sürüme dahil ettiğini beyan ettiği aşamadır. "Closed" ise test mühendisinin düzeltmeyi test edip doğruladıktan sonra hatayı kesin olarak sonlandırdığı aşamadır.

Soru 7: Bir hata kaydı formunda (Bug Report) yer alması gereken teknik detaylar nelerdir?

İyi bir hata kaydında; hatanın benzersiz kimliği (ID), başlığı, hatayı yeniden üretme adımları (steps to reproduce), beklenen ve gözlenen sonuçlar, ortam bilgileri (sürüm, tarayıcı vb.) ve kanıt olarak log veya ekran görüntüleri bulunmalıdır.

Soru 8: Test kapanışı aşamasında hazırlanan "Test Özet Raporu" hangi metrikleri içermelidir?

Raporda; yürütülen testlerin sayısı, başarı/başarısızlık oranları, bulunan hataların şiddet ve modül bazlı dağılımı ile sistemin genel kalite durumu hakkında özet bilgiler yer almalıdır.

Soru 9: Gereksinim temelli fonksiyonel test ile kullanım senaryosu (use case) tabanlı test arasındaki farkı açıklayınız.

Gereksinim temelli testler doğrudan belirli bir maddeyi (örn: "Giriş yapabilmelidir") doğrulamaya odaklanır. Kullanım senaryosu tabanlı testler ise kullanıcının sistemi uçtan uca nasıl kullandığına (örn: "Kayıt ol, aktivasyon yap ve ilk kez giriş yap") odaklanan daha zengin akışları test eder.

Soru 10: Proje sonunda "Canlıya Çıkış" kararı verilirken "Kabul/Red" değerlendirmesi neye göre yapılır?

Bu karar sadece geçen test sayısına bakılarak verilmez. Açık kalan hataların şiddeti, bu hataların iş süreçlerine ve güvenliğe olası etkileri ile canlı ortamda ortaya çıkabilecek risklerin kabul edilebilirliği birlikte değerlendirilir. Bazı durumlarda düşük riskli hatalarla canlıya çıkılabilirken, bazı durumlarda tek bir kritik hata sürümün ertelenmesine neden olabilir.