WEB & YAZILIM GELİŞTİRME SERİSİ · MODÜL 15

Kodlamada YZ

Yapay zekâyla kodlamak — sorumlu biçimde: kod asistanları, etkili istem, bağlam ve doğrulama; kod yazma, açıklama, hata ayıklama, inceleme, test ve dokümantasyonda YZ; güvenlik, sırlar/KVKK, lisans, önyargı, aşırı bağımlılık; istem mühendisliği, ajanlar, ekip politikası, etik ve insan denetimi. Özgün diyagramlarla ve 'önce doğrula' ilkesiyle.

İstem · Üretim · Doğrulama · Sorumluluk · Eğitim amaçlıdır

Bu Kitap Hakkında

Bu modül, çağımızın en güçlü geliştirme araçlarından birini — yapay zekâ destekli kodlamayı — sorumlu ve etkili biçimde kullanmayı öğretir. Ofis serisindeki yapay zekâ modülünün geliştirici karşılığıdır ve baştan sona tek bir omurgaya dayanır: YZ önerir, insan doğrular ve karar verir. Modül daha çok kavram, yargı ve sorumlu kullanım üzerine kuruludur; her konuyu özgün diyagramlarla (doğrulama döngüsü, istem anatomisi, araç türleri, güvenlik ve sınır şemaları) ve gerçek örneklerle açıklar. Dört seviye ve yirmi altı bölüm boyunca kodlamada YZ'nin ne olduğundan ve nasıl çalıştığından (örüntüye dayalı tahmin, bu yüzden halüsinasyon), araç türlerine, etkili isteme, bağlam vermeye, üretilen kodu okumaya, halüsinasyon ve doğrulamaya, insan denetimine; günlük geliştirmede kod yazma, açıklama/öğrenme, hata ayıklama, kod incelemesi, test ve dokümantasyona; güvenlik risklerine, sırlar ve veri gizliliğine (KVKK), lisans ve atıfa, önyargı ve sınırlara, aşırı bağımlılığa ve derinlemesine doğrulamaya; istem mühendisliğine, ajanlara, ekip politikasına, iş akışı tasarımına ve etik sorumluluğa kadar uzanır.

Her bölümde konuyu görselleştiren özgün bir diyagram (doğrulama döngüsü, istem anatomisi, araç türü karşılaştırması, güvenlik ve sınır şemaları), gerçek örnekler, 'önce doğrula' kartı ve bir alıştırma yer alır. Sorumlu kullanım baştan sona vurgulanır: çıktıyı doğrulamak (halüsinasyona karşı), sırları ve kişisel veriyi (KVKK) korumak, üretilen kodu güvenlik gözüyle incelemek, lisans belirsizliğine dikkat etmek, önyargı ve sınırları bilmek, aşırı bağımlılıktan kaçınıp beceriyi korumak, ajanları denetlemek ve her çıktının sorumluluğunu sahiplenmek. Modül, sorumlu YZ kullanım ilkelerinden oluşan bir kontrol listesiyle kapanır. Bu, on altı modüllük 'Web & Yazılım Geliştirme' serisinin on beşinci ve FAZ 4'ün ilk modülüdür. Bu seri eğitim amaçlıdır; YZ çıktıları her zaman insan tarafından doğrulanmalıdır.

Web & Yazılım Geliştirme Serisi · Modül 15

İçindekiler

SEVİYE 1

Kodlamada YZ'ye Giriş

Temeller: kodlamada yapay zekâ nedir, nasıl çalışır, araç türleri, etkili istem, bağlam vermek, üretilen kodu okumak, halüsinasyon ve doğrulama, insan denetimi.

BÖLÜM 01

Kodlamada Yapay Zekâ Nedir?

Kodlama yapay zekâsı (YZ kod asistanları), çok büyük miktarda kod üzerinde eğitilmiş ve sana kod önerileri üreten araçlardır. Yazarken tamamlama önerir, soruları yanıtlar, kod yazar veya açıklar. Ama unutma: bir asistandır, bir yetki değil — son karar ve sorumluluk her zaman senindir.

Asistan, ama patron değil

Şema 1.1 — YZ bir hızlandırıcıdır; yönlendiren ve doğrulayan sensin.

- **Ne yapar:** kod önerir, açıklar, hata bulmaya yardım eder, test yazar.
- **Ne yapmaz:** senin yerine düşünmez, anlamaz, sorumluluk almaz.
- **Senin rolün:** yönlendirmek, doğrulamak, karar vermek.

İPUCU

Bu modülün en önemli mesajı baştan net olsun: **YZ güçlü bir araçtır, ama bir araçtır** — onu bir "her şeyi bilen otorite" değil, "hızlı ama bazen yanılan bir stajyer" gibi düşün. Çok iş çıkarabilir, ama her çıktısını **sen kontrol etmelisin**. Bu seride öğrendiğin tüm temeller (HTML/CSS/JS, algoritma, veritabanı, güvenlik, yayınlama) burada paha biçilmez hâle gelir: **YZ'nin önerisini ancak konuyu anlıyorsan değerlendirebilirsin**. Konuyu bilmeyen biri için YZ tehlikelidir (yanlışı fark edemez); konuyu bilen biri için ise muazzam bir hızlandırıcıdır. Bu yüzden YZ, öğrenmenin yerini almaz — onu güçlendirir. Modül boyunca tekrar tekrar göreceğin ilke şudur: **YZ önerir, insan karar verir**. Her bölümdeki "önce doğrula" kartı bu yüzden var.

Önce doğrula KONTROL

Bir YZ asistanı kullandığında, o sana hızla bir taslak (kod, açıklama, öneri) sunar — ama bu taslak doğru, güvenli ve senin amacına uygun olmayabilir. Bu yüzden her çıktıyı okur, anlar ve doğrularsın; öyle kabul edersin. YZ işi hızlandırır, ama doğru olup olmadığının sorumluluğu sende kalır. Bu modül boyunca her konuda bu doğrulama refleksini geliştireceksin.

Alıştırma 8 dk

YZ'nin rolünü kavra:

1. YZ asistanının yaptığı ve yapmadığı şeyleri ikişer madde yaz.
2. "YZ önerir, insan karar verir" ilkesini kendi cümlenle açıkla.
3. Konuyu bilmeyen biri için YZ neden riskli olabilir, belirt.

BÖLÜM 02

Nasıl Çalışır (Genel Bakış)

YZ kod asistanlarının nasıl çalıştığını kabaca anlamak, onlara neden güvenip neden temkinli olman gerektiğini gösterir. Özünde bu araçlar "anlamaz"; çok büyük miktarda metin ve kodda gördükleri örüntülere dayanarak, bir sonraki en olası parçayı tahmin eder.

Tahmin, gerçek bilgi değil

Şema 2.1 — YZ anlamaz; örüntüye göre en olası devamı tahmin eder.

- **Örüntü tanıma:** milyonlarca örnekten kalıpları öğrenir.
- **Tahmin:** bağlama göre en olası devamı üretir.
- **Anlamaz:** doğruyu "bilmez"; olasıyı üretir — bu yüzden yanılabilir.

İPUCU

Bu basitleştirilmiş bakış, YZ'nin hem gücünü hem zayıflığını açıklar. **Gücü:** o kadar çok kod görmüştür ki, yaygın kalıpları (bir döngü nasıl yazılır, bir API nasıl çağrılır) şaşırtıcı isabetle üretir. **Zayıflığı:** "anlamadığı" için, akla yatkın **ama yanlış** şeyler de üretebilir — çünkü amacı "doğru olmak" değil, "olası/akıcı görünmek"tir. İşte **halüsinasyonun** (sonraki bölümlerde) kök sebebi budur: var olmayan bir fonksiyon, yanlış bir parametre veya hatalı bir mantık, tıpkı gerçeği gibi **özgüvenle** sunulabilir. Bu yüzden YZ'nin "kendinden emin" görünmesi, doğru olduğu anlamına gelmez. Ayrıca YZ'nin bilgisi **eğitildiği ana kadarıdır**; çok yeni kütüphaneleri veya değişiklikleri bilmeyebilir. Bunu anlamak, sana en önemli alışkanlığı kazandırır: **özgüvenli bir çıktı bile doğrulanmalıdır**.

Önce doğrula

KONTROL

YZ bir çıktı ürettiğinde, bunu "doğru olduğunu bilerek" değil, "en olası devam bu" diye üretir. Çoğu zaman olası olan ile doğru olan örtüşür ve harika sonuç alırsın; ama bazen örtüşmez — akla yatkın görünen ama yanlış bir kod gelir. Çıktının özgüvenli tonu, doğruluğunun garantisi değildir. Bu yüzden her çıktıyı, "bu gerçekten doğru mu?" diye kontrol edersin.

Alıştırma

10 dk

Çalışma mantığını kavra:

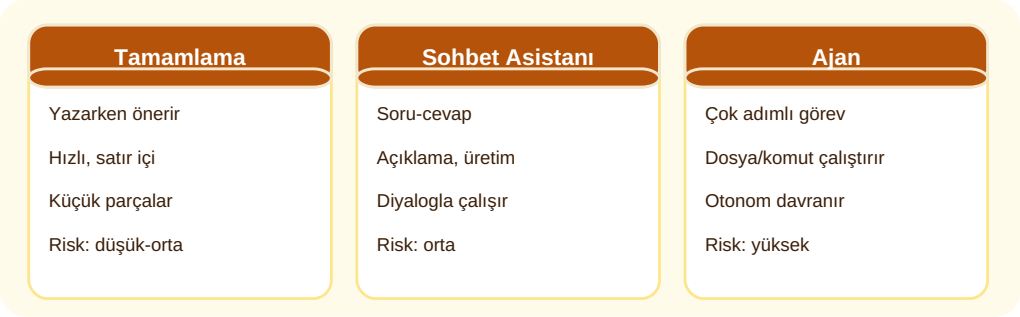
1. YZ'nin "anladığı" mı yoksa "tahmin ettiği" mi, kendi cümlenle açıkla.
2. Bu çalışma biçimi halüsinasyonla nasıl ilişkili, yaz.
3. YZ'nin özgüvenli görünmesinin neden doğruluk anlamına gelmediğini belirt.

BÖLÜM 03

YZ Araç Türleri

Kodlamada YZ tek bir şey değildir; farklı biçimlerde karşına çıkar. En yaygın üçü: yazarken satır içi öneren tamamlama araçları, soru-cevapla çalışan sohbet asistanları ve çok adımlı görevleri yürüten ajanlar. Her birinin gücü ve riski farklıdır.

Üç ana tür



Şema 3.1 — YZ araç türleri: tamamlama, sohbet ve ajan; risk arttıkça denetim artar.

- **Tamamlama:** editörde yazarken satır içi öneri (en yaygın).
- **Sohbet asistanı:** soru sorar, açıklama/kod alırsın.
- **Ajan:** bir görevi kendi başına, çok adımda yürütür — en çok denetim ister.

İPUCU

Üç türün ortak ilkesi aynı: **ne kadar otonom, o kadar dikkat**. **Tamamlama** araçları küçük öneriler sunar (bir satır, bir fonksiyon) — sen yazarken görür ve kabul/ret edersin, kontrol nispeten kolaydır. **Sohbet asistanları** daha büyük parçalar üretir ve diyalog kurarsın; çıktıyı bütün olarak değerlendirmen gerekir. **Ajanlar** en güçlü ama en riskli türdür: bir hedef verirsin, onlar birçok adımı (dosya değiştirme, komut çalıştırma, hatta yayınlama) kendi başlarına yapabilir. Ajanlar zaman kazandırır ama **denetimsiz bırakılırsa** ciddi zarar verebilir (yanlış dosyayı silmek, hatalı kodu uygulamak). Bu yüzden ajanlarla çalışırken **onay kapıları** koymak ve ne yaptıklarını adım adım izlemek şarttır. Hangi türü kullanırsan kullan, "önce doğrula" ilkesi değişmez — yalnızca doğrulamanın kapsamı, aracın otonomisiyle birlikte büyür.

Önce doğrula KONTROL

Bir tamamlama aracı kullandığında, küçük öneriler görür ve her birini hızla onaylar/reddedersin. Bir sohbet asistanından kod istediğinde, gelen bütünü okuyup doğrularsın. Bir ajana görev verdiğinde, onun attığı her adımı (özellikle dosya değiştiren veya komut çalıştıran adımları) izler ve onaylarsın. Araç ne kadar çok şeyi kendi başına yapıyorsa, senin denetimin o kadar dikkatli olmalıdır.

Alıştırma 10 dk

Araçları ayırt et:

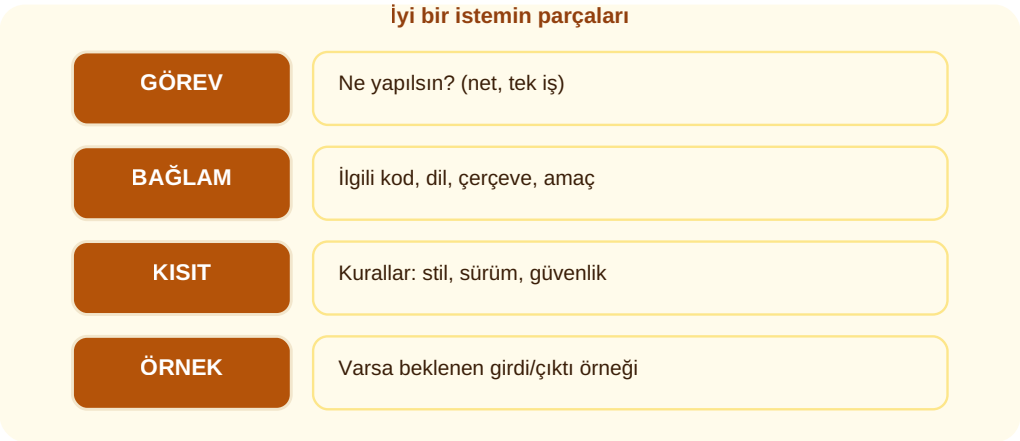
1. Üç YZ araç türünü ve birer kullanım örneğini yaz.
2. Neden ajanlar en çok denetim gerektirir, açıkla.
3. "Ne kadar otonom, o kadar dikkat" ilkesini kendi cümlenle belirt.

BÖLÜM 04

İlk Adım: Etkili İstem (Prompt)

YZ'den iyi sonuç almak, iyi soru sormakla başlar. "İstem" (prompt), YZ'ye ne istediğini anlatma biçimindir. Belirsiz bir istem belirsiz sonuç verir; net, bağlamlı ve sınırları belli bir istem ise isabetli sonuç verir. İyi istem yazmak öğrenilebilir bir beceridir.

İyi bir istemin parçaları



Şema 4.1 — İyi istem: görev + bağlam + kısıt + (varsa) örnek.

Belirsiz vs net istem

```
 # Belirsiz (kötü):
"bir giriş formu yap"

# Net (iyi):
"HTML + CSS ile bir giriş formu yaz. Alanlar:
 e-posta, parola. Mobil uyumlu olsun, etiketler
 erişilebilir olsun, JS doğrulama EKLEME."
```

İPUCU

İyi istemin sırrı, YZ'ye **net bir hedef ve yeterli bağlam** vermektir. Dört parçayı düşün: **Görev** (tam olarak ne yapılsın — tek ve net bir iş), **Bağlam** (hangi dil/çerçeve, ilgili kod, ne amaçla), **Kısıt** (uyulması gereken kurallar: stil, sürüm, güvenlik, "şunu yapma"), ve varsa **Örnek** (beklenen girdi/çıktı). Belirsiz bir istem ("bir şeyler yap"), YZ'yi tahmin etmeye zorlar ve genelde işine yaramayan bir sonuç çıkar. Pratik ipuçları: **küçük ve net** iste (devasa tek bir istem yerine adım adım), **kısıtları açıkça söyle** (özellikle "şunu yapma"ları), ve sonuç istediğin gibi değilse **düzeltici geri bildirimle** tekrar dene ("bu iyi ama şu kısmı şöyle değiştir"). İyi istem yazmak, YZ'yle çalışmanın en pratik becerisidir — ve Seviye 4'te buna "istem mühendisliği" olarak daha derin bakacağız. Ama unutma: en iyi istem bile **doğrulamayı ortadan kaldırmaz**; sadece daha iyi bir başlangıç noktası verir.

Önce doğrula KONTROL

Net, bağlamlı ve sınırları belli bir istem yazdığında, YZ amacına çok daha yakın bir sonuç üretir — daha az gidip gelmeyle istediğine ulaşırsın. Belirsiz bir istem ise genelde işe yaramayan, varsayımlarla dolu bir çıktı verir ve zaman kaybettirir. Ama net istemle gelen iyi görünen sonucu bile doğrularsın: iyi istem, doğru sonucu garanti etmez, yalnızca olasılığını artırır.

Alıştırma 12 dk

İstem yaz:

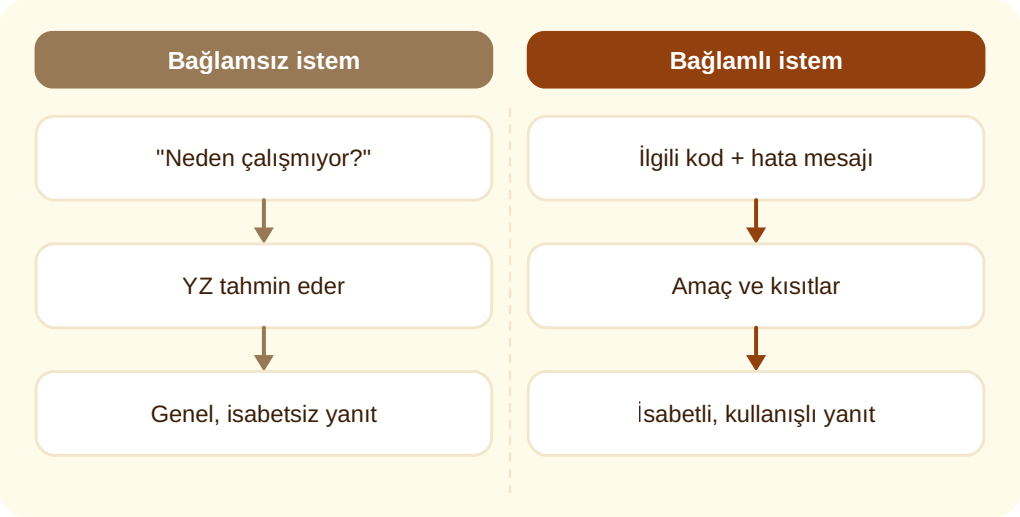
1. Belirsiz bir istem ile aynı isteğin net hâlini karşılaştırmalı yaz.
2. İyi bir istemin dört parçasını (görev/bağlam/kısıt/örnek) örnekle.
3. İstemde "şunu yapma" gibi kısıtların neden faydalı olduğunu açıkla.

BÖLÜM 05

Bağlam Vermek

YZ yalnızca verdiğin bilgiyle çalışır; aklını okuyamaz ve projenin tamamını bilmez. Doğru ve ilgili bağlam vermek (mevcut kod, hata mesajı, amaç, kısıtlar), aldığın sonucun kalitesini doğrudan belirler. Bağlam ne kadar yerinde, sonuç o kadar isabetli.

Bağlam kaliteyi belirler



Şema 5.1 — Bağlam: ne kadar ilgili bilgi verirsen, yanıt o kadar isabetli.

- **İlgili kodu ver:** sorunlu fonksiyon, ilgili dosya.
- **Hata mesajını ver:** tam metniyle (özetleme).
- **Amacı ve kısıtları söyle:** ne yapmaya çalışıyorsun, ne kullanıyorsun.

İPUCU

YZ'nin en sık hayal kırıklığı yaratan davranışı, aslında **yetersiz bağlamdan** kaynaklanır: "neden çalışmıyor?" diye sorarsan, kodu görmeyen YZ tahmin etmek zorunda kalır ve genelde işe yaramaz. Ona **ilgili kodu**, **tam hata mesajını** ve **ne yapmaya çalıştığını** verirsen, çok daha isabetli yardım alırsın. Ancak bağlam verirken **kritik bir denge** vardır: faydalı bağlam ver, ama **hassas bilgi verme**. Sırları (parolalar, API anahtarları), kişisel verileri (KVKK kapsamındakiler) veya şirketin gizli kodunu bir YZ aracına yapıştırmak **ciddi bir güvenlik/gizlilik riskidir** (Seviye 3'te derinleşeceğiz) — çünkü o veri nereye gittiğini ve nasıl kullanıldığını bilmediğin bir sisteme aktarılır. Bu yüzden bağlamı **temizle**: gerçek anahtarları örnek değerlerle değiştir, kişisel verileri çıkar, yalnızca sorunu anlatmaya yetecek kadarını ver. "İlgili ama güvenli bağlam" altın kuraldır.

Önce doğrula

KONTROL

İlgili bağlamı (kod, hata, amaç) verdiğinde, YZ'nin yanıtı genel tahminlerden çıkıp senin durumuna özel, kullanışlı bir hâle gelir — sorunu çok daha hızlı çözersin. Ama bağlamı verirken sırları ve kişisel verileri temizlersin; faydalı olanı paylaşır, hassas olanı korursun. Doğru bağlam isabeti artırır; temiz bağlam ise seni güvende tutar.

Alıştırma

10 dk

Bağlam ver:

1. Bağlamsız ve bağlamlı bir "neden çalışmıyor?" sorusunu karşılaştır.
2. YZ'ye verilecek ilgili bağlam türlerini (kod, hata, amaç) yaz.
3. Bir YZ aracına asla yapıştırmaman gereken bilgi türlerini belirt.

BÖLÜM 06

Üretilen Kodu Okumak ve Anlamak

YZ bir kod ürettiğinde, en tehlikeli şey onu okumadan kabul etmektir. Üretilen kodu satır satır okur, ne yaptığını anlar ve ancak öyle kullanırsın. Anlamadığın bir kodu projene koymak, içinde ne olduğunu bilmediğin bir kutuyu eve sokmak gibidir.

Önce oku, sonra kabul et

Şema 6.1 — Üretilen kodu okumadan kabul etme: oku, anla, sorgula.

İPUCU

"Çalışıyor gibi görünüyor" ile "doğru ve güvenli" arasında büyük fark vardır. YZ'nin ürettiği kod akıcı ve ikna edici görünebilir ama **gizli hatalar**, **güvenlik açıkları** veya **senin durumuna uymayan varsayımlar** içerebilir. Bu yüzden altın kural: **anlamadığın kodu projene koyma**. Üretilen kodu okurken kendine sor: Bu satır tam olarak ne yapıyor? Neden böyle yazılmış? Bir güvenlik riski var mı (Seviye 3)? Gereksiz/şişkin mi? Benim durumuma uygun mu? Eğer bir kısmı anlamıyorsan, YZ'ye **açıklatabilirsin** ("bu satır ne yapıyor?") — bu hem güvenliği artırır hem öğretir. Anlamadan kabul etmenin iki büyük zararı vardır: (1) **gizli hatalar/açıklar** projene sızar, (2) sen **öğrenmezsin** ve bağımlı hâle gelirsin (Seviye 3'teki "aşırı bağımlılık"). Üretilen kodu okumak, YZ'yi bir "kara kutu" olmaktan çıkarıp gerçek bir öğrenme ve hızlanma aracına dönüştürür. Sen kodun sahibisin; ne koyduğunu bilmelisin.

Önce doğrula KONTROL

Üretilen kodu satır satır okuyup anladığında, içine gizli hataların veya güvenlik açıklarının sızmasını engellersin ve aynı zamanda öğrenirsin. Anlamadığın bir parça varsa, kullanmadan önce onu açıklatır veya araştırırsın. Sonuç: projende yalnızca anladığın, doğruladığın ve sahiplendiğin kod bulunur — "çalışıyor gibi" değil, "doğru biliyorum" diyebildiğin kod.

Alıştırma 12 dk

Kodu oku:

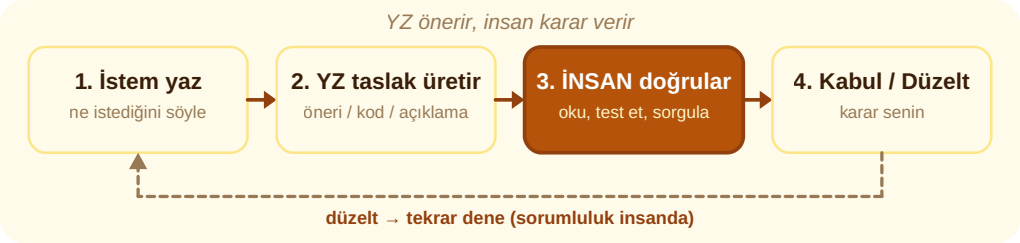
1. YZ üretilen kodu okurken kendine soracağın üç soruyu yaz.
2. "Anlamadığın kodu projene koyma" kuralının iki nedenini açıkla.
3. Anlamadığın bir satırla karşılaşınca ne yaparsın, belirt.

BÖLÜM 07

Halüsinasyon ve Doğrulama

Halüsinasyon, YZ'nin gerçek gibi görünen ama yanlış veya tamamen uydurma bir şey üretmesidir: var olmayan bir fonksiyon, yanlış bir parametre, hatalı bir gerçek. Üstelik bunu özgüvenle yapar. Bu yüzden YZ ile çalışmanın kalbinde tek bir alışkanlık vardır: doğrulamak.

Doğrulama döngüsü



Şema 7.1 — Doğrulama döngüsü: YZ önerir, insan doğrular ve karar verir.

Halüsinasyon örneği: var olmayan bir yöntem

```
// YZ özgüvenle önerebilir:
const sonuc = dizi.bulHepsi(x => x > 5);
// ama JavaScript'te "bulHepsi" YOKTUR.
// Doğru yöntem: dizi.filter(x => x > 5)
// -> Bu yüzden her çağrıyı doğrula.
```

İPUCU

Halüsinasyon, YZ'nin "anlamadan tahmin etme" doğasının (Bölüm 2) doğrudan sonucudur: akla yatkın görünen ama gerçek olmayan bir şey üretir — var olmayan bir kütüphane fonksiyonu, yanlış bir API kullanımı, uydurma bir istatistik veya hatalı bir mantık. En sinsi yanı, bunu **tıpkı doğruymuş gibi, özgüvenle** sunmasıdır. Çözüm tek kelimedir: **doğrulama**. Pratik doğrulama yolları: **kodu çalıştır ve test et** (en kesin yol — gerçekten çalışıyor mu?), **resmî dokümana bak** (bu fonksiyon/parametre gerçekten var mı?), **kendi bilginle kontrol et** (mantık doğru mu?), ve **kritik gerçekleri ikinci bir kaynaktan teyit et**. Asla "YZ söyledi, doğrudur" deme — özellikle kesin bir gerçek, bir komut, bir güvenlik kararı veya bir kütüphane kullanımı söz konusuysa. Diyagramdaki döngü bu modülün özüdür: **İstem → YZ üretir → İNSAN doğrular → kabul/düzelt**. Doğrulama bir engel değil, YZ'yi güvenle hızlı kılan şeydir. Doğrulanmamış YZ çıktısı, kontrol edilmemiş bir varsayımdır.

Önce doğrula

KONTROL

YZ bir çıktı verdiğinde, onu çalıştırarak, resmî dokümanla karşılaştırarak ve kendi bilginle sorgulayarak doğrularsın — özellikle var olmayan fonksiyonlar, yanlış parametreler ve kesin "gerçekler" konusunda. Doğrulama, halüsinasyonların projene sızmasını engeller. Sonuç: YZ'nin hızından yararlanırken, onun yanılma payını insan kontrolüyle kaparsın. Doğrulanmış çıktı güvenli, doğrulanmamış çıktı bir risktir.

Alıştırma

12 dk

Doğrula:

1. "Halüsinasyon" nedir ve neden olur (Bölüm 2 ile bağ), kendi cümlenle yaz.
2. Bir YZ çıktısını doğrulamanın üç pratik yolunu açıkla.
3. Hangi durumlarda doğrulama özellikle kritiktir, örnek ver.

BÖLÜM 08

İnsan Denetimi: Sen Karar Verirsin

Tüm bu modülün üstüne kurulduğu ilke budur: YZ ne kadar yetenekli olursa olsun, son karar ve sorumluluk insandadır. Sen yönlendirir, doğrular, kabul eder veya reddeder ve sonuçtan sorumlu olursun. YZ bir araçtır; onu kullanan ve cevap veren sensin.

Döngüdeki insan

Şema 8.1 — İnsan denetimi: yönlendir, doğrula, karar ver, sorumlu ol.

- **Yönlendiren:** hedefi ve sınırları sen koyarsın.
- **Doğrulayan:** çıktının doğru/güvenli olduğunu sen kontrol edersin.
- **Karar veren:** kabul mü, düzeltme mi, ret mi — sen seçersin.
- **Sorumlu olan:** sonuçtan "YZ söyledi" diye kaçamazsın.

İPUCU

"İnsan döngüde" (human-in-the-loop) ilkesi, sorumlu YZ kullanımının temelidir: YZ önerir, ama **her kritik kararı bir insan onaylar**. Bu yalnızca teknik değil, **etik ve yasal** bir gerçektir: bir kod hata yaparsa, bir güvenlik açığı kullanıcıları zarara uğratırsa veya bir karar birini mağdur ederse, sorumluluk **"YZ söyledi"** diyerek kaçılabilecek bir şey değildir — sorumluluk, o çıktıyı kabul eden insandadır. Bu yüzden YZ'yi, sorumluluğu devredebileceğin bir merci değil, hızlandıran bir asistan olarak gör. Otonomi arttıkça (özellikle ajanlarda) insan denetimi **azalmaz, daha kritik hâle gelir**: ne kadar çok şeyi YZ'ye bırakıyorsan, kontrol kapıların o kadar net olmalı. Bu ilke, bu modülün geri kalanının çerçevesidir — günlük kullanımda (Seviye 2), güvenlik ve gizlilikte (Seviye 3), ve olgun çalışmada (Seviye 4) hep aynı merkez döner: **YZ önerir, insan karar verir ve sorumluluğu taşır**. Bu, YZ'yi güçsüzleştirmez; onu güvenle güçlü kılar.

Önce doğrula

KONTROL

YZ ile çalışırken sen döngünün merkezindesin: hedefi koyar, çıktıyı doğrular, kabul/düzelt/ret kararını verir ve sonuçtan sorumlu olursun. YZ ne kadar otomatik olursa olsun, bu zincir değişmez — yalnızca otonomi arttıkça denetimin daha dikkatli olur. Sonuç: YZ'nin hızını alırsın ama kontrolü ve sorumluluğu elinde tutarsın. "YZ söyledi" bir mazeret değildir; karar ve sonuç senindir.

Alıştırma

10 dk

Denetimi sahiplen:

1. İnsanın YZ döngüsündeki dört rolünü (yönlendir/doğrula/karar/sorumlu) yaz.
2. Neden "YZ söyledi" bir sorumluluk mazereti olamaz, açıkla.
3. Otonomi (örn. ajanlar) arttıkça insan denetimi neden daha kritik olur, belirt.

SEVİYE 2

Günlük Geliştirmede YZ

YZ'yi günlük işte kullanmak: kod tamamlama ve yazma, açıklama ve öğrenme, hata ayıklama, kod incelemesi, test yazımı ve dokümantasyon — her zaman doğrulama refleksiyle.

BÖLÜM 09

Kod Tamamlama ve Yazma

YZ'nin en yaygın kullanımı, kod yazmaya yardımdır: bir satırı tamamlar, bir fonksiyon üretir, tekrarlayan bir kalıbı hızlıca yazar. Doğru kullanıldığında bu büyük bir hız kazancıdır — ama gelen her parçayı okuyup doğrulama kuralı asla değişmez.

Hızlı taslak, dikkatli kabul

Şema 9.1 — Kod yazımı: net istem → taslak → doğrula → entegre.

Üretilen kodu doğrulama refleksi

```
// YZ bir fonksiyon önerdi. Sormadan kabul etme:
// 1) Ne yapıyor? Satır satır anladın mı?
// 2) Kenar durumlar? (boş girdi, sıfır, hata)
// 3) Güvenli mi? (girdi doğrulama, sır yok)
// 4) Test ettin mi? -> Öyle entegre et.
```

İPUCU

Kod yazımında YZ, en çok zaman kazandıran ama en kolay kötüye kullanılan alandır. **İyi kullanım:** tekrarlayan kalıpları (bir döngü iskeleti, bir veri dönüşümü), bildiğin ama yazması sıkıcı şeyleri ve hızlı taslakları YZ'ye yazdırıp **kontrol ederek** hızlanmak. **Kötü kullanım:** anlamadığın karmaşık kodu körü körüne kabul edip projeye gömmek. Pratik denge: YZ'yi **küçük, doğrulanabilir parçalar** için kullan (büyük, anlaşılmaz bir blok yerine). Gelen her parçayı, kendin yazmış gibi sahiplen — çünkü artık senin sorumluluğunda. Özellikle **kenar durumları** (boş girdi, hata, sınır değerler) ve **güvenliği** (girdi doğrulama — Seviye 3) kontrol et; YZ bunları sıklıkla atlar veya basitleştirir. Unutma: YZ ile hızlı kod yazmak, hızlı **doğru** kod yazmak değildir — ikisini buluşturan şey senin doğrulamandır.

Önce doğrula

KONTROL

YZ ile kod yazdığında, dakikalar yerine saniyeler içinde bir taslağa kavuşursun — ama o taslak senin amacına, güvenlik ihtiyaçlarına ve kenar durumlarına tam uymayabilir. Bu yüzden okur, kenar durumları ve güvenliği kontrol eder, test eder ve ancak öyle entegre edersin. Sonuç: YZ'nin hızını alır, ama projene yalnızca anladığın ve doğruladığın kodu koyarsın.

Alıştırma

12 dk

Kodu yazdır ve doğrula:

1. YZ ile kod yazmanın bir iyi ve bir kötü kullanımını yaz.
2. Üretilen kodda neden özellikle kenar durumları kontrol etmelisin, açıkla.
3. "Hızlı kod" ile "hızlı doğru kod" farkını kendi cümlenle belirt.

BÖLÜM 10

Açıklama ve Öğrenme

YZ'nin en değerli ve en güvenli kullanımlarından biri öğrenmedir: anlamadığın bir kodu açıklatmak, bir kavramı sormak, bir hatanın neden olduğunu öğrenmek. Doğru kullanıldığında YZ, sabırlı bir özel öğretmen gibidir — ama öğrettiğini de teyit etmen gerekir.

YZ'yle öğrenmek

Şema 10.1 — Öğrenme: açıklat → anla → kaynaktan teyit et.

İPUCU

Öğrenme, YZ'nin en parlak olduğu alandır çünkü burada **çıktıyı doğrudan üretime koymazsın** — anlamak için kullanırsın, dolayısıyla risk düşüktür ve fayda yüksektir. YZ'ye anlamadığın bir kodu açıklattırabilir ("bu satır ne yapıyor?"), bir kavramı farklı düzeylerde sorabilir ("bunu daha basit anlat"), veya bir hatanın kök sebebini öğrenebilirsin. Bu, sabırlı, yargılamayan ve 7/24 erişilebilir bir öğretmen gibidir. **Ama kritik bir incelik var:** YZ öğretirken de halüsinasyon yapabilir — yanlış bir açıklama, uydurma bir "kural" veya eski bir bilgi verebilir. Bu yüzden öğrendiğin **kritik şeyleri** (özellikle bir gerçeği, bir güvenlik kuralını, bir API davranışını) **resmî kaynaktan teyit et**. En sağlıklı kullanım: YZ'yi "hızlı kavrama" için kullan, sonra önemli noktaları güvenilir dokümanla pekiştir. Bu, hem hızlı öğrenmeni hem yanlış öğrenmemeni sağlar. YZ öğrenmeni hızlandırır; ama neyi öğrendiğinin doğruluğundan yine sen sorumlusun.

Önce doğrula KONTROL

YZ'ye bir kodu veya kavramı açıklattığında, onu kendi başına saatlerce araştırmaktan çok daha hızlı kavrarsın — üstelik anlamadığın yeri tekrar tekrar, utanmadan sorabilirsin. Ama YZ açıklarken de yanılabileceği için, öğrendiğin kritik bilgileri resmî dokümandan teyit edersin. Sonuç: YZ öğrenmeni hızlandırır, teyit ise yanlış öğrenmeni engeller — ikisi birlikte güçlü bir öğrenme döngüsü kurar.

Alıştırma 10 dk

YZ'yle öğren:

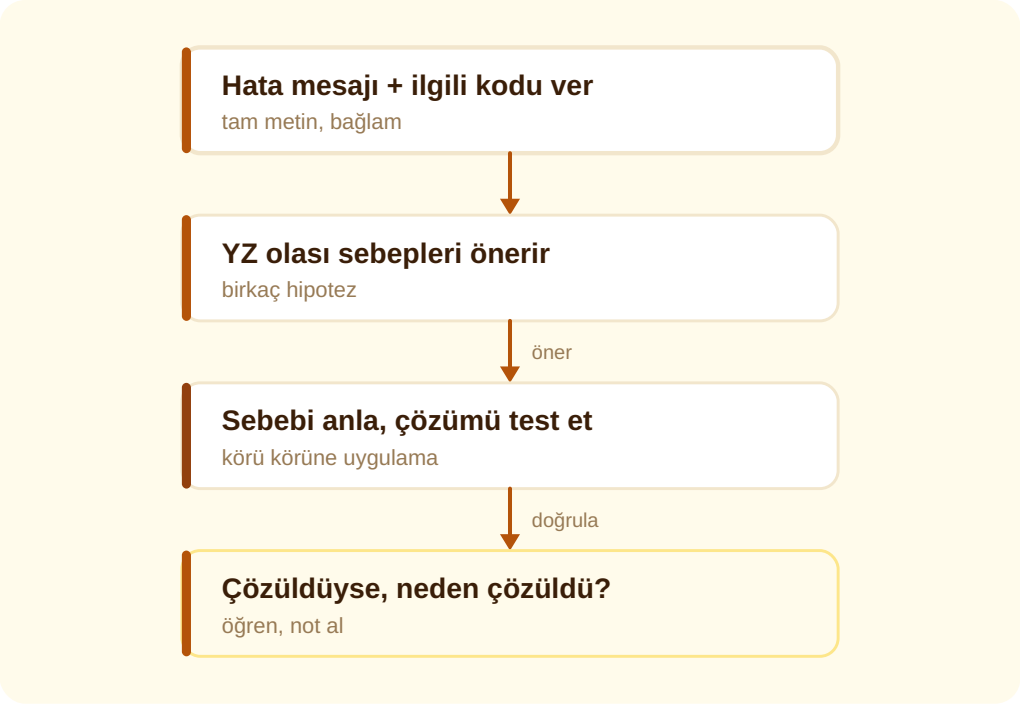
1. YZ'yi öğrenmek için kullanmanın neden düşük riskli olduğunu yaz.
2. YZ'nin öğretirken de yanılabileceğini bir örnekle açıkla.
3. Öğrendiğin kritik bir bilgiyi nasıl teyit edersin, belirt.

BÖLÜM 11

Hata Ayıklama (Debugging)

Bir hatayla karşılaştığında, YZ güçlü bir yardımcı olabilir: hata mesajını ve ilgili kodu verir, olası sebepleri ve çözümleri sorarsın. Ama YZ'nin önerdiği çözümü körü körüne uygulamak yerine, sebebi anlayıp çözümü test ederek ilerlersin.

YZ ile sorun çözme



Şema 11.1 — Hata ayıklama: bağlam ver → hipotez al → anla ve test et.

İyi bir hata ayıklama istemi

```
 # YZ'ye ver:
# 1) TAM hata mesajı (özetleme)
# 2) İlgili kod parçası
# 3) Ne yapmaya çalıştığın + ne beklediğin
# Sonra: önerilen çözümü ANLA ve TEST ET.
```

İPUCU

YZ hata ayıklamada gerçekten yardımcıdır çünkü çok sayıda benzer hata görmüştür ve hızlıca **olası sebepler** önerebilir — bu, "nereye bakacağını bilememe" tıkanıklığını aşmana yardım eder. İyi kullanım için: **tam hata mesajını** ver (özetleme — tam metin altın değerindedir), **ilgili kodu** ekle ve **ne beklediğini** söyle. Ama iki tuzağa dikkat et: **(1) Körü körüne yama:** YZ'nin önerdiği çözüm hatayı "susturabilir" ama gerçek sebebi çözmeyebilir (hatta yeni hata ekleyebilir) — bu yüzden **sebebi anla**, sadece belirtiyi değil. **(2) Doğrulamadan uygulama:** önerilen çözümü test et; gerçekten çalıştığını ve başka bir şeyi bozmadığını gör. En değerli sonuç, hatanın **çözülmesi kadar neden çözüldüğünü anlamandır** — bu seni bir sonraki hatada daha güçlü kılar. YZ'yi bir "düşünme ortağı" gibi kullan (hipotez üreten), kararı sen ver. Ayrıca: hata ayıklarken paylaştığın koda da dikkat — sır veya kişisel veri içeriyorsa temizle (Seviye 3).

Önce doğrula

KONTROL

Bir hatayı YZ'ye tam mesajı ve ilgili koduyla anlattığında, sana hızlıca birkaç olası sebep ve çözüm sunar — özellikle takıldığın noktada büyük bir hızlandırma. Ama önerilen çözümü uygulamadan önce sebebini anlar ve test edersin; böylece belirtiyi susturmak yerine gerçek sorunu çözer ve yeni hata eklemezsin. Sonuç: hatayı hem hızlı çözer hem de neden olduğunu öğrenerek bir sonrakine hazırlanırsın.

Alıştırma

12 dk

Hata ayıkla:

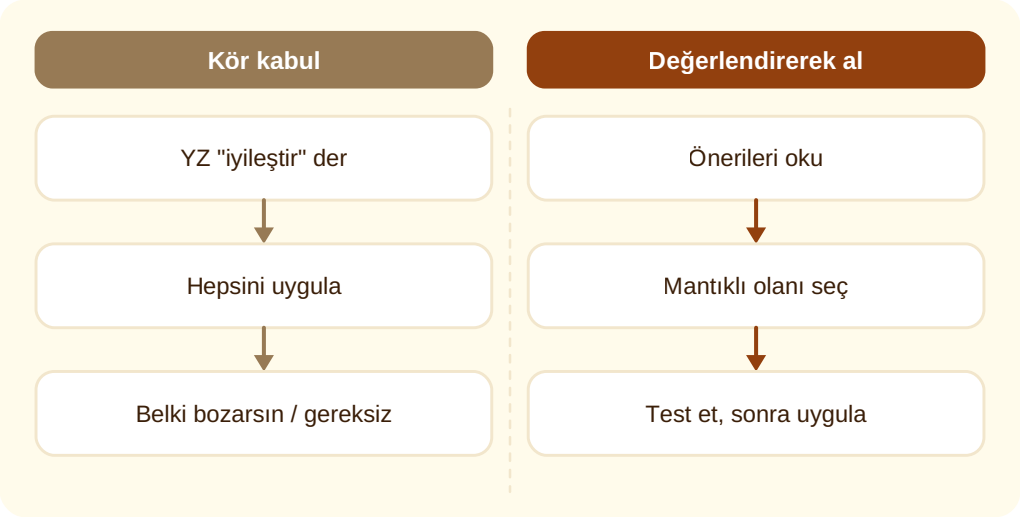
1. İyi bir hata ayıklama istemine neleri eklersin, sırala.
2. "Belirtiyi susturmak" ile "sebebi çözmek" farkını açıkla.
3. Önerilen çözümü neden test etmen gerektiğini belirt.

BÖLÜM 12

Kod İncelemesi ve İyileştirme

YZ, yazdığın kodu inceleyip iyileştirme önerileri sunabilir: daha okunur, daha verimli, daha güvenli hâle getirme. Bu, ikinci bir göz gibi değerlidir — ama YZ'nin her önerisi "iyileştirme" değildir; önerileri değerlendirip sana uygun olanları seçersin.

İkinci göz, ama son söz sende



Şema 12.1 — İnceleme: önerileri değerlendir, körü körüne uygulama.

İPUCU

YZ'yi bir **kod inceleme yardımcısı** olarak kullanmak değerlidir: yazdığın koda "bunu nasıl iyileştirebilirim?", "bir güvenlik riski var mı?", "daha okunur olabilir mi?" diye sorabilirsin. İnsan inceleme (Modül 14'teki PR) yerine geçmez ama onu tamamlar — özellikle tek başına çalışıyorsan değerli bir "ikinci göz" sağlar. Ancak **YZ'nin her önerisi iyi değildir**: bazı önerileri gereksiz karmaşıklık ekler, bazıları senin bağlamına uymaz, bazıları "moda" ama faydasızdır. Bu yüzden önerileri **değerlendirerek** al: her birini "bu gerçekten daha mı iyi yapıyor, neden?" diye sorgula, mantıklı olanı seç, uygulamadan önce test et. Özellikle **güvenlik iyileştirmeleri** için YZ faydalı bir kontrol listesi sunabilir (Seviye 3) ama yine doğrula. İyi bir refactoring (yeniden düzenleme) kuralı: **davranışı değiştirmeden** iyileştir ve her adımdan sonra test et — YZ ile çalışırken bu daha da kritik, çünkü YZ bazen "iyileştirirken" davranışı sessizce değiştirebilir.

Önce doğrula

KONTROL

YZ'ye kodunu incelettiğinde, gözden kaçırdığın iyileştirmeleri (okunabilirlik, verimlilik, güvenlik) görme şansın artar — yalnız çalışırken bile bir "ikinci göz" kazanırsın. Ama her öneriyi değerlendirir, sana uygun olanı seçer ve uygulamadan önce test edersin; çünkü bazı öneriler gereksiz karmaşıklık ekler veya davranışı değiştirir. Sonuç: kodun gerçekten iyileşir, körü körüne uygulamanın getireceği risk olmadan.

Alıştırma

10 dk

İncelet ve değerlendir:

1. YZ kod incelemesinin insan incelemesini neden tamamladığını (yerine geçmediğini) yaz.
2. Her YZ önerisinin neden "iyileştirme" olmayabileceğini açıkla.
3. Refactoring yaparken "davranışı değiştirmeden + test et" kuralını belirt.

BÖLÜM 13

Test Yazımında YZ

Test yazmak önemli ama çoğu zaman ihmal edilen bir iştir (Modül 14). YZ, test taslakları üreterek bu işi hızlandırabilir. Ama YZ'nin ürettiği testlerin gerçekten doğru şeyi test ettiğini ve önemli durumları kapsadığını sen kontrol etmelisin.

Test taslağı, insan kapsamı

Şema 13.1 — Test: YZ taslak üretir, insan kapsamı ve doğruluğu sağlar.

YZ'nin atlayabileceği kenar durumlar

```
 // YZ "mutlu yol" testini yazar:
test("topla(2,3) === 5", ...)
// Sen kenar durumları EKLE:
// - negatif sayılar? sıfır? çok büyük sayı?
// - geçersiz girdi? (metin, boş, null)
```

İPUCU

Test yazımı, YZ'nin gerçekten yardımcı olduğu bir alandır çünkü test kodu genelde **tekrarlayan ve şablon ağırlıklıdır** — YZ hızlıca bir iskelet üretebilir, sen de zihinsel enerjini "neyi test etmeliyim?" sorusuna ayırabilirsin. Ama kritik bir tuzak var: YZ genelde **"mutlu yolu"** (her şeyin yolunda gittiği durumu) test eder ve asıl önemli olan **kenar durumları** (boş girdi, sıfır, negatif, çok büyük değer, geçersiz tür, hata koşulları) sıklıkla atlar — oysa hatalar tam da buralarda saklanır. Bu yüzden YZ'nin ürettiği testleri bir **başlangıç** olarak al, sonra **kapsamı sen tamamla**: "hangi durumda bu kod bozulabilir?" diye düşünüp eksik testleri ekle. Ayrıca dikkat: bir testin **var olması**, doğru şeyi test ettiği anlamına gelmez — YZ bazen yanlış bir beklentiyi test eden, yani hatayı "onaylayan" testler yazabilir. Bu yüzden her testin **doğru davranışı** kontrol ettiğini sen doğrula. Test, kodun güvencesidir; o güvencenin kendisi de güvenilir olmalı.

Önce doğrula

KONTROL

YZ'ye test yazdırdığında, sıkıcı şablon işini hızlıca halleder ve sen asıl önemli soruya — "hangi durumları test etmeliyim?" — odaklanırsın. Ama YZ genelde temel "mutlu yolu" test ettiğinden, kenar ve hata durumlarını sen ekler ve her testin doğru davranışı kontrol ettiğini doğrularsın. Sonuç: testlerin hem hızlı yazılır hem de gerçekten güven verecek kapsama ulaşır — kodun her yayında daha sağlam olur.

Alıştırma

12 dk

Test yazdır:

1. YZ'nin test yazarken genelde hangi durumları atladığını yaz.
2. Bir test fonksiyonu için eklemen gereken üç kenar durumu örnekle.
3. "Bir testin var olması doğru olduğu anlamına gelmez" ne demek, açıkla.

BÖLÜM 14

Dokümantasyon ve Yorumlar

İyi dokümantasyon ve açıklayıcı yorumlar (Modül 14), kodun bakımını kolaylaştırır ama yazması sıkıcıdır. YZ, koduna bakıp doküman ve yorum taslakları üretebilir. Ama bu taslakların kodu doğru anlattığını ve gerçeği yansıttığını kontrol etmek senin işin.

Taslak hızlı, doğruluk insanda

Şema 14.1 — Dokümantasyon: YZ taslak üretir, insan doğruluğu sağlar.

İPUCU

Dokümantasyon ve yorum yazımı, YZ'nin güzel bir yardımcı olduğu ama dikkat gerektiren bir alandır. **İyi tarafı:** YZ, bir fonksiyona bakıp ne yaptığını özetleyen bir açıklama, parametre listesi veya kullanım örneği taslağı üretebilir — bu, "boş sayfa" zorluğunu aşar. **Dikkat edilecek:** YZ kodun **ne yaptığını** tahmin eder; eğer kodu yanlış yorumladıysa, doküman da yanlış olur — ve **yanlış doküman, hiç dokümandan kötüdür** (insanları yanlış yönlendirir). Bu yüzden ürettiği açıklamanın kodu **gerçekten doğru** anlattığını kontrol et. İki tuzak daha: **(1) "Neyi" anlatan gereksiz yorumlar** — YZ bazen kodun aynısını tekrarlayan yorumlar üretir ("i'yi bir artır" gibi); iyi yorum "ne" değil **"neden"**i açıklar. **(2) Bayatlama** — doküman kodla birlikte güncellenmeli; YZ ile hızlı üretmek, güncel tutma sorumluluğunu kaldırmaz. En iyi kullanım: YZ'yi ilk taslak için kullan, sonra **doğruluk, sadelik ve "neden"e odaklanarak** insan eliyle rafine et. Doküman senin adına konuşur; doğru ve net olduğundan emin ol.

Önce doğrula

KONTROL

YZ'ye kodunu belgelettiğinde, açıklama ve yorumların ilk taslağını hızlıca alırsın — boş sayfanın zorluğunu aşarsın. Ama YZ kodu yanlış yorumlayabileceğinden, ürettiği dokümanın gerçeği doğru anlattığını kontrol eder, gereksiz tekrarları atar ve "neden"i öne çıkarırsın. Sonuç: kodun hızlıca belgelenir, ama yalnızca doğru ve işe yarar dokümantasyonla — yanlış yönlendiren değil, yol gösteren kayıtlarla.

Alıştırma

10 dk

Belgelet ve doğrula:

1. Neden "yanlış doküman, hiç dokümandan kötüdür", açıkla.
2. İyi bir yorumun "ne" yerine "neden"i anlatması ne demek, örnekle.
3. YZ ile üretilen dokümanı kontrol ederken nelere bakarsın, yaz.

SEVİYE 3

Sorumlu ve Güvenli Kullanım

YZ'yi güvenle kullanmak: üretilen kodun güvenlik riskleri, sırlar ve veri gizliliği (KVKK), lisans ve atıf, önyargı ve sınırlar, aşırı bağımlılık ve derinlemesine doğrulama.

BÖLÜM 15

Güvenlik: Üretilen Kodun Riskleri

YZ'nin ürettiği kod, çalışıyor görünse bile güvenlik açıkları içerebilir. YZ güvenliği "varsayılan olarak" düşünmez; gördüğü örneklerdeki kalıpları üretir — ve bu örneklerin bir kısmı güvensizdir. Bu yüzden üretilen kodu güvenlik gözüyle incelemek şarttır.

Sık görülen güvenlik riskleri

Şema 15.1 — Üretilen kodda sık görülen güvenlik riskleri.

- **Girdi doğrulama eksiği:** kullanıcı girdisini güvenmek (SQL enjeksiyonu, XSS — Modül 7/8).
- **Savunmasız bağımlılık:** eski veya bilinen açığı olan paket önerme.
- **Sır gömme:** örnek diye koda gerçekçi anahtar/parola yazma.
- **Aşırı yetki:** gereğinden geniş erişim/izin.

İPUCU

Bu, bu serinin güvenlik bilincinin (Modül 7-12-14) YZ tarafıdır ve **kritiktir**: YZ'nin ürettiği kod "çalışıyor" diye **güvenli sanma**. YZ, güvenliği özel olarak istemediğin sürece çoğu zaman düşünmez ve eğitim verisindeki — bir kısmı güvensiz — kalıpları üretir. Sık riskler: **girdi doğrulama eksikliği** (kullanıcı girdisini doğrudan kullanmak → SQL enjeksiyonu, XSS; Modül 7-8'deki parametreli sorgu/temizleme ilkelerini YZ atlayabilir), **savunmasız bağımlılıklar** (YZ eski veya açığı olan bir paket önerebilir — bilgisi güncel olmayabilir), **sır gömme** (örnek diye koda gerçekçi anahtarlar yazma) ve **aşırı yetki**. Korunma: üretilen kodu **güvenlik gözüyle de incele**, YZ'den **açıkça güvenlik iste** ("girdiyi doğrula", "parametreli sorgu kullan"), bağımlılıkları **tara** (Modül 14) ve kritik güvenlik kodunu mutlaka **insan ve araçlarla denetle**. YZ güvenlik için faydalı bir **yardımcı** olabilir (açık aramak, öneri sunmak) ama güvenliğin **sorumlusu** değildir — o sensin. Güvenlik, "çalışıyor mu?"dan ayrı ve daha derin bir sorudur.

Önce doğrula KONTROL

YZ'nin ürettiği kodu güvenlik gözüyle incelediğinde, çalışıyor görünen ama açık içeren parçaları (doğrulanmamış girdi, savunmasız paket, gömülü sır) yakalarsın. YZ'den açıkça güvenlik istemek ve kritik kodu insan/araçlarla denetlemek, bu açıkların üretime sızmasını engeller. Sonuç: YZ'nin hızından yararlanırken, güvenliği — "çalışıyor mu?"dan ayrı bir sorumluluk olarak — sen güvence altına alırsın.

Alıştırma 12 dk

Güvenliği denetle:

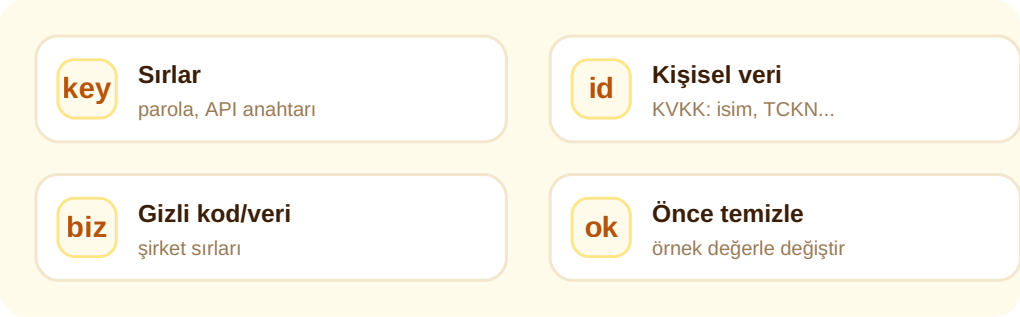
1. Üretilen kodda sık görülen üç güvenlik riskini yaz.
2. "Çalışıyor" ile "güvenli" arasındaki farkı açıkla.
3. YZ'den güvenli kod almak için ne yapabilirsin (açık istek), belirt.

BÖLÜM 16

Sırlar ve Veri Gizliliği (KVKK)

Bir YZ aracına bir şey yazdığında, o veri senin kontrolünden çıkıp başka bir sisteme gider. Bu yüzden sırları (parolalar, anahtarlar), kişisel verileri (KVKK kapsamındakiler) ve gizli şirket kodunu YZ araçlarına vermek ciddi bir risktir. Ne paylaştığına dikkat etmek, sorumlu kullanımın temelidir.

YZ'ye ne vermemeli?



Şema 16.1 — YZ'ye vermeden önce temizle: sır, kişisel ve gizli veri.

Paylaşmadan önce temizle

```
// YAPMA: gerçek sır/kişisel veri yapıştırma
const key = "sk-canli-GERCEK-ANAHTAR";
const eposta = "gercek.kullanici@firma.com";

// YAP: örnek/temsilî değerlerle anlat
const key = "ORNEK_ANAHTAR";
const eposta = "ornek@example.com";
```

İPUCU

Bu, sorumlu YZ kullanımının **en kritik kurallarından biridir** ve doğrudan Modül 7-12-14'teki güvenlik/gizlilik ilkelerinin ve **KVKK**'nın uzantısıdır. Temel gerçek: bir YZ aracına yazdığın her şey, **senin kontrolünden çıkıp** üçüncü bir tarafın sistemine gider; orada nasıl saklandığını, kimin gördüğünü veya gelecekte nasıl kullanılacağını **tam bilemezsin**. Bu yüzden şunları **asla** yapıştırma: **sırlar** (parolalar, API anahtarları, tokenlar), **kişisel veriler** (gerçek isimler, TCKN, e-posta, telefon, adres, sağlık/finans verisi — KVKK kapsamı), ve **gizli şirket kodu/verisi** (ticari sırlar, müşteri verisi). Bunun yerine: kodu/sorunu **temizleyerek** paylaş — gerçek anahtarları `ORNEK_ANAHTAR` ile, kişisel verileri temsilî değerlerle değiştir, yalnızca sorunu anlatmaya yetecek kadarını ver. Kurumsal ortamlarda genelde **YZ kullanım politikaları** vardır (hangi aracın kullanılabileceği, neyin paylaşılabileceği — Seviye 4); bunlara uy. KVKK açısından: kişisel veriyi izinsiz bir YZ'ye aktarmak bir **veri ihlali** olabilir ve yasal sorumluluk doğurur. Kural basit: **emin değilsen, paylaşma.**

Önce doğrula KONTROL

Bir YZ aracına bilgi yazmadan önce temizlediğinde — gerçek anahtarları, kişisel verileri ve gizli kodu örnek değerlerle değiştirdiğinde — sorunu yine anlatabilir ama hassas veriyi kontrolünden çıkarmamış olursun. Bu, hem güvenlik sızıntılarını hem KVKK ihlallerini önler. Sonuç: YZ'den yardım alırken, sırları ve insanların kişisel verilerini korur, yasal ve etik sorumluluğunu yerine getirirsin. Emin olmadığında paylaşmamak en güvenli yoldur.

Alıştırma 12 dk

Veriyi koru:

1. Bir YZ aracına asla vermemen gereken üç tür bilgiyi yaz.
2. Bir kodu YZ ile paylaşmadan önce nasıl "temizlersin", açıkla.
3. Kişisel veriyi izinsiz YZ'ye vermenin KVKK açısından sonucunu belirt.

BÖLÜM 17

Lisans ve Atıf

YZ'nin ürettiği kod nereden geliyor? YZ, çok sayıda kamuya açık kod üzerinde eğitilmiştir; ürettiği kod bazen mevcut kodlara benzeyebilir. Bu, lisans ve telif açısından gri bir alandır. Üretilen kodu kullanırken bu belirsizliğin farkında olmak gerekir.

Belirsizliğin farkında ol

Şema 17.1 — Lisans: üretilen kodun kaynağı belirsiz olabilir; dikkatli ol.

İPUCU

Bu, henüz net kurallara tam oturmamış, gelişen bir alandır; bu yüzden amaç kesin hukuki cevaplar vermek değil, **farkında olmanı** sağlamaktır (ben bir hukukçu değilim; ciddi durumlarda hukuki danışmanlık alınmalı). Temel gerçek: YZ, milyonlarca satır **kamuya açık kod** üzerinde eğitilmiştir ve farklı lisanslara (kimi serbest, kimi kısıtlı) sahip bu kodlardan öğrendiği örüntüleri üretir. Çoğu zaman ürettiği kısa/yaygın parçalar bir sorun teşkil etmez, ama **uzun veya özgün bir bloğu** neredeyse birebir üretmesi teorik olarak mümkündür ve bu, bir başkasının lisanslı kodunu farkında olmadan kullanmana yol açabilir. Pratik yaklaşım: **kısa, yaygın kalıplar** için genelde endişelenme; **büyük, özgün veya kritik** bir blok için, tanıdık/şüpheli geliyorsa kaynağını araştır. **Kurumsal ortamda** mutlaka şirketinin YZ ve lisans **politikasına uy** (bazı kurumlar üretilen kodun kullanımına kurallar koyar). Ayrıca: ürettiğin işte YZ'nin payını **şeffafça** belirtmek (ekip/proje kurallarına göre) iyi bir uygulamadır. Özetle: lisans belirsizliğinin farkında ol, şüphedeysen araştır veya danış.

Önce doğrula

KONTROL

YZ ürettiği kodu kamuya açık kodlardan öğrendiği örüntülerle oluşturur; bu yüzden ürettiği parçanın kaynağı ve lisansı belirsiz olabilir. Kısa, yaygın kalıplar genelde sorun değildir; büyük veya özgün bloklarda ise tanıdık geliyorsa kaynağını araştırır ve kurumunun politikasına uyarsın. Sonuç: YZ'den yararlanırken, başkasının haklarını farkında olmadan ihlal etme riskini azaltır ve şüphede olduğunda doğru kişilere danışırsın.

Alıştırma

10 dk

Lisansa dikkat et:

1. YZ'nin ürettiği kodun kaynağının neden belirsiz olabileceğini yaz.
2. Hangi durumda lisans konusunda daha dikkatli olmalısın (kısa vs büyük blok), açıkla.
3. Kurumsal bir ortamda şüphe duyduğunda ne yaparsın, belirt.

BÖLÜM 18

Önyargı ve Sınırlar

YZ kusursuz değildir ve nesnel de değildir. Eğitim verisindeki önyargıları yansıtabilir, güncel olmayan bilgiler verebilir ve bağlamı sınırlıdır. Bu sınırları bilmek, YZ'ye nerede güvenip nerede temkinli olacağını anlamanı sağlar.

YZ'nin sınırları

Şema 18.1 — YZ'nin sınırları: önyargı, eski bilgi, sınırlı bağlam.

İPUCU

YZ'yi olgunca kullanmak, sınırlarını bilmekle başlar. **(1) Önyargı:** YZ, eğitildiği verideki — yani insanların ürettiği metin ve koddaki — **önyargıları yansıtabilir**. Bu kodda "yaygın ama kötü" alışkanlıkları üretmek olarak görünebileceği gibi, daha geniş bağlamlarda (örneğin insanları sınıflandıran bir sistem yazarken) **ayrımcı sonuçlar** üretmeye de yol açabilir — bu yüzden insanları etkileyen kararlar üreten kodda özellikle dikkatli ol. **(2) Eski bilgi:** YZ'nin bilgisi **eğitildiği ana kadarıdır**; çok yeni kütüphaneleri, sürümleri veya değişiklikleri bilmeyebilir, hatta kaldırılmış/değişmiş bir şeyi hâlâ "doğru" sanabilir. **(3) Sınırlı bağlam:** YZ senin projenin tamamını, ekip kararlarını, iş kurallarını görmez — yalnızca verdiğin kadarını bilir. **(4) Genelleme:** "ortalama" bir çözüm üretir; senin özel, kenar veya alışılmadık durumuna uymayabilir. Bu sınırların hiçbiri YZ'yi işe yaramaz kılmaz — ama **nerede temkinli olacağını** söyler: yeni teknolojiler, insanları etkileyen kararlar, özel/karmaşık bağlamlar ve "herkese uyan" görünen çözümler için fazladan doğrula. Sınırlarını bilmek, aracı daha iyi kullanmaktır.

Önce doğrula KONTROL

YZ'nin önyargı, eski bilgi, sınırlı bağlam ve genelleme gibi sınırları olduğunu bildiğinde, ona körü körüne değil, yerinde güvenirsin: yeni teknolojilerde bilgisinin eskimiş olabileceğini, özel durumunda genel çözümünün uymayabileceğini, insanları etkileyen kararlarda önyargı taşıyabileceğini hesaba katarsın. Sonuç: YZ'yi güçlü olduğu yerde kullanır, sınırlı olduğu yerde fazladan doğrularsın — bu, aracı en olgun biçimde kullanmaktır.

Alıştırma 10 dk

Sınırları tanı:

1. YZ'nin dört sınırını (önyargı, eski bilgi, bağlam, genelleme) yaz.
2. Hangi tür kodda önyargı konusunda özellikle dikkatli olmalısın, açıkla.
3. YZ'nin "eski bilgi" sınırı pratikte nasıl bir soruna yol açabilir, belirt.

BÖLÜM 19

Aşırı Bağımlılık ve Beceri

YZ öğrenmeyi ve üretmeyi hızlandırır — ama bir tuzağı vardır: her şeyi YZ'ye yaptırırsan, kendi becerin körelebilir ve YZ olmadan çalışamaz hâle gelebilirsin. Sağlıklı kullanım, YZ'yi beceriyi köreltmek için değil, geliştirmek için kullanmaktır.

Hızlandırıcı mı, koltuk değneği mi?

Şema 19.1 — YZ'yi beceriyi geliştirmek için kullan, köreltmek için değil.

İPUCU

Bu, özellikle öğrenme aşamasındakiler için kritik bir dengedir. YZ o kadar yardımcıdır ki, **düşünmeyi ona devretme** tuzağına düşmek kolaydır: her sorunu sormak, her kodu anlamadan kabul etmek, hiç zorlanmamak. Sorun şu: **zorlanmak, öğrenmenin nasıl gerçekleştiğidir** — bir problemi kendin çözmeye çalışmak (başarısız olsan bile), beyninde YZ'nin asla kuramayacağı bağlantılar kurar. Her şeyi YZ'ye yaptıran biri, kısa vadede hızlı görünür ama uzun vadede: (1) **temel becerileri gelişmez**, (2) YZ'nin **yanlışını fark edemez** (çünkü kendisi bilmez), (3) YZ olmadan **çaresiz** kalır. Sağlıklı denge: **önce kendin dene/düşün**, sonra YZ'ye danış; üretileni **anlayarak** kullan; ara sıra **YZ olmadan** çalışıp becerini koru; YZ'yi "benim yerime yapan" değil "bana öğreten ve hızlandıran" olarak kullan. Özellikle öğrenirken, bazı şeyleri zahmetli de olsa kendin yapmak **bir yatırımdır**. YZ en çok, **güçlü bir temeli olan** kişinin elinde değer kazanır — çünkü o kişi onu yönlendirebilir, değerlendirebilir ve yanılgılarını yakalayabilir. Amacın YZ'siz çaresiz kalmak değil, YZ'yle **daha güçlü** olmak.

Önce doğrula KONTROL

YZ'yi sağlıklı kullandığında — önce kendin deneyip sonra danışarak, üretileni anlayarak, ara sıra YZ'siz çalışarak — becerin YZ'yle birlikte artar; o seni hızlandıran bir güç olur. Her şeyi anlamadan YZ'ye yaptırdığında ise becerin gelişmez, YZ'nin hatalarını fark edemez ve ona bağımlı hâle gelirsin. Sonuç: doğru kullanımda YZ seni güçlendirir; yanlış kullanımda zayıflatır. Hedef, YZ'yle daha yetkin bir geliştirici olmaktır.

Alıştırma 10 dk

Beceriyi koru:

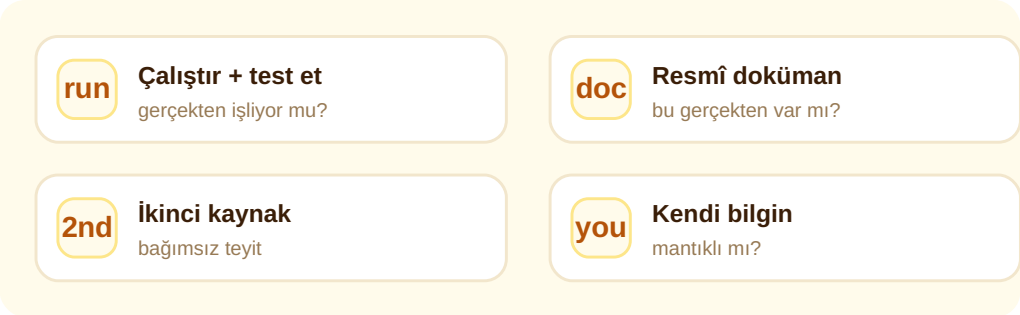
1. Aşırı YZ bağımlılığının üç uzun vadeli zararını yaz.
2. "Zorlanmak, öğrenmenin nasıl gerçekleştiğidir" ne demek, açıkla.
3. YZ'yi beceriyi geliştirmek için kullanmanın iki yolunu belirt.

BÖLÜM 20

YZ Çıktısını Doğrulama (Derinlemesine)

Doğrulama, bu modülün kalbidir — bu bölümde onu derinleştiriyoruz. Bir YZ çıktısını doğrulamanın birden çok yolu vardır ve çıktının türüne (kod, gerçek, karar) göre hangisini kullanacağını bilmek, sorumlu kullanımın özüdür.

Doğrulama araçların



Şema 20.1 — Doğrulama araçları: çalıştır, dokümana bak, teyit et, sorgula.

- **Kod için:** çalıştır ve test et — en kesin doğrulama.
- **Gerçek/iddia için:** resmî dokümana ve ikinci bir kaynağa bak.
- **Karar/öneri için:** kendi muhakemenle ve uzman görüşüyle tart.

İPUCU

Doğrulama "her şeyi her zaman aynı şekilde kontrol et" demek değildir; **çıktının türüne göre doğru aracı seçmektir**. **Kod** için en kesin doğrulama **çalıştırıp test etmektir** — gerçekten işliyor mu, kenar durumlarda ne yapıyor, başka bir şeyi bozuyor mu? **Bir gerçek veya iddia** için (bir fonksiyonun varlığı, bir API'nin davranışı, bir "en iyi uygulama") **resmî dokümana** bak ve mümkünse **ikinci bir bağımsız kaynaktan** teyit et. **Bir karar veya tasarım önerisi** için kendi muhakemeni, deneyimini ve gerekirse bir uzmanın görüşünü kullan — çünkü "en iyi" çözüm bağlama bağlıdır ve YZ senin tüm bağlamını bilmez. **Doğrulamanın derinliği riske göre ölçeklenir:** kritik bir güvenlik kodu, bir ödeme mantığı veya insanları etkileyen bir karar için doğrulama çok daha titiz olmalı; deneme amaçlı küçük bir script için daha hafif olabilir. Altın kural değişmez: **doğrulanmamış YZ çıktısı, bir varsayımdır** — onu bir gerçek gibi kullanmadan önce kanıtla. Bu refleks, seni "YZ'ye kanan" biri olmaktan "YZ'yi ustaca kullanan" biri olmaya taşır.

Önce doğrula

KONTROL

Bir YZ çıktısını doğrularken türüne uygun aracı seçtiğinde — kodu çalıştırıp test ederek, gerçekleri resmî kaynaktan teyit ederek, kararları kendi muhakemenle tartarak — onun güvenilirliğini gerçekten ölçersin. Doğrulamanın titizliğini riske göre ayarlar, kritik kodda çok daha dikkatli olursun. Sonuç: YZ'nin hızını alırken, çıktısını bir varsayım olmaktan çıkarıp kanıtlanmış bir sonuca dönüştürürsün — sorumlu kullanımın özü budur.

Alıştırma

12 dk

Derinlemesine doğrula:

1. Kod, gerçek ve karar için ayrı ayrı hangi doğrulama yöntemini kullanırsın, yaz.
2. Doğrulamanın derinliğinin neye göre ölçeklenmesi gerektiğini açıkla.
3. "Doğrulanmamış YZ çıktısı bir varsayımdır" ilkesini kendi cümlenle belirt.

SEVİYE 4

YZ ile Olgun Çalışmak

YZ'yle olgun çalışmak: iyi istem mühendisliği, ajanlar ve otomasyon, ekip kullanımı ve politika, iş akışı tasarımı, etik ve sorumluluk ile sorumlu kullanım ilkeleri.

BÖLÜM 21

İyi İstem Mühendisliği

İstem mühendisliği, YZ'den tutarlı ve isabetli sonuçlar almak için istemleri ustaca yazma becerisidir. Birkaç teknik, aldığın sonuçların kalitesini dramatik biçimde artırır — ama yine, en iyi istem bile doğrulamayı ortadan kaldırmaz.

İstemi güçlendiren teknikler

Şema 21.1 — İstem mühendisliği: net görev → bağlam/örnek → böl → iyileştir.

İPUCU

İstem mühendisliği "sihirli kelimeler" bulmak değil, **YZ'ye iyi düşünme malzemesi vermektir**. En etkili teknikler: **(1) Net ve tek görev** — bir istemde tek bir iş iste; "şunu yap, ayrıca bunu, bir de onu" demek karışık sonuç verir. **(2) Bağlam ve örnek ver** — ilgili kodu, hedefi ve mümkünse **beklenen girdi/çıktı örneğini** göstermek (buna "örnekle yönlendirme" denir) isabeti çok artırır. **(3) Adımlara böl** — karmaşık bir görevi tek seferde istemek yerine parçalara ayır; YZ'ye "önce şunu planla, sonra uygula" demek ("adım adım düşün") daha tutarlı sonuç verir. **(4) Rol/çerçeve ver** — "deneyimli bir güvenlik gözüyle incele" gibi bir çerçeve, çıktının odağını ayarlar. **(5) İteratif düzelt** — ilk sonuç nadiren mükemmeldir; "bu iyi ama şu kısmı şöyle yap" diyerek rafine et. **(6) Kısıtları açıkça söyle** — "şunu kullanma", "şu sürüme uygun olsun", "güvenli olsun". Bu teknikler sonucu iyileştirir ama bir şeyi değiştirmez: **çıktıyı yine doğrularsın**. İyi istem, doğru cevabı garanti etmez — sadece olasılığını yükseltir ve gidip gelmeyi azaltır. İstem mühendisliği, YZ'yi daha verimli bir ortak yapar; doğrulama ise onu güvenli yapar.

Önce doğrula KONTROL

İstem mühendisliği tekniklerini (net tek görev, bağlam ve örnek, adımlara bölme, iteratif düzeltme, açık kısıtlar) kullandığında, YZ'den ilk seferde çok daha isabetli ve kullanışlı sonuçlar alır, daha az gidip gelirsin. Ama en iyi istem bile çıktının doğruluğunu garanti etmez; onu yine doğrularsın. Sonuç: iyi istemle YZ daha verimli bir ortak olur, doğrulamayla da güvenli kalır — ikisi birlikte olgun kullanımı oluşturur.

Alıştırma 12 dk

İstemini güçlendir:

1. İstem kalitesini artıran üç tekniği yaz ve birer örnek ver.
2. Karmaşık bir görevi neden adımlara bölmek daha iyi sonuç verir, açıkla.
3. "En iyi istem bile doğrulamayı kaldırmaz" ne demek, belirt.

BÖLÜM 22

YZ Ajanları ve Otomasyon

YZ ajanları, bir hedef verdiğinde onu birçok adımda kendi başına yürüten araçlardır: dosya okur/yazar, komut çalıştırır, kararlar verir. Büyük güç sunarlar ama büyük risk de taşırlar — bu yüzden ajanlarla çalışmak, denetimin en kritik olduğu yerdir.

Güçlü ama denetim ister

Şema 22.1 — Ajanlar: otonomi yüksek → denetim ve onay kapıları şart.

İPUCU

Ajanlar, YZ'nin en güçlü ve en riskli biçimidir. Güçlüdür çünkü bir hedefi (örneğin "şu özelliği ekle ve test et") çok adımda **kendi başına** yürütebilir — planlar, dosya değiştirir, komut çalıştırır. Risklidir çünkü **otonomi, hata yapma kapasitesini de büyütür**: yanlış bir adım artık tek bir öneri değil, gerçekleştirilen bir eylemdir (yanlış dosyayı silmek, hatalı kodu uygulamak, istenmeyen bir komut çalıştırmak). Bu yüzden ajanlarla çalışırken denetim **azalmaz, en kritik hâle gelir**. Güvenli kullanım: **onay kapıları koy** (özellikle dosya silen, komut çalıştıran, yayınlayan adımları ajan yapmadan önce sen onayla); **kapsamı sınırla** (ajanın erişimini ve yetkisini gerektiği kadar tut — en az yetki ilkesi); **her adımı izle** (ne yaptığını adım adım takip et, "kara kutu" gibi bırakma); ve **güvenli bir ortamda dene** (geri alınabilir, izole bir alanda — üretimde değil). Ajanlar muazzam zaman kazandırabilir ama "ayarla ve unut" zihniyetiyle değil. Otonomi arttıkça insan sorumluluğu kaybolmaz; yalnızca **denetimin şekli değişir** — tek tek satırları değil, eylemleri ve onları onaylamayı denetlersin. En güçlü araç, en dikkatli eli gerektirir.

Önce doğrula

KONTROL

Bir YZ ajanına görev verdiğinde, o birçok adımı kendi planlayıp gerçekleştirir — bu büyük bir hız sağlar ama her adım gerçek bir eylem olduğundan hata riski de büyür. Onay kapıları koyar (kritik adımları sen onaylarsın), kapsamı sınırlar, her adımı izler ve güvenli/geri alınabilir bir ortamda çalışırsın. Sonuç: ajanın gücünden yararlanırken, eylemlerinin kontrolünü ve sorumluluğunu elinde tutarsın — denetim, otonomiyle birlikte daha da kritikleşir.

Alıştırma

12 dk

Ajanı denetle:

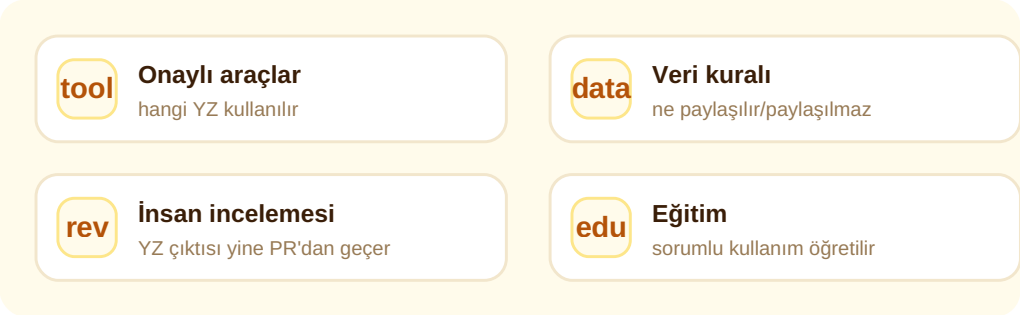
1. Ajanların neden hem güçlü hem riskli olduğunu yaz.
2. Bir ajanla güvenli çalışmak için alacağın üç önlemi açıkla.
3. "Otonomi arttıkça denetim daha kritik olur" ilkesini kendi cümlenle belirt.

BÖLÜM 23

Ekipte YZ Kullanımı ve Politika

YZ'yi tek başına kullanmak ile bir ekipte/kurumda kullanmak farklıdır. Ekipte tutarlılık, güvenlik ve sorumluluk için ortak kurallar (politikalar) gerekir: hangi araçlar kullanılabilir, ne paylaşılabilir, çıktılar nasıl denetlenir. İyi politika, YZ'nin faydasını korurken riskini yönetir.

Ortak kurallar



Şema 23.1 — Ekipte YZ: onaylı araç, veri kuralı, inceleme, eğitim.

İPUCU

Bir ekipte veya kurumda YZ kullanımı, bireysel kullanımdan daha fazla yapı gerektirir çünkü **riskler de paylaşılır** (birinin sır sızdırması tüm kurumu etkiler). İyi bir YZ politikası şunları netleştirir: **Hangi araçlar onaylı?** (her YZ aracı aynı gizlilik/güvenlik garantisini vermez; kurum belirli araçları onaylayabilir, özellikle veri işleme konusunda). **Ne paylaşılabilir, ne paylaşılamaz?** (Seviye 3'teki sır/KVKK kuralları kurumsal düzeyde tanımlanır — müşteri verisi, gizli kod asla; genel sorular serbest olabilir). **Çıktılar nasıl denetlenir?** (YZ ile üretilen kod da insan incelemesinden — Modül 14'teki PR'dan — geçmeli; "YZ yazdı" denetimi atlamaz). **Sorumluluk kimde?** (çıktıyı kullanan kişi sorumludur, net olmalı). **Eğitim:** ekip, sorumlu kullanım (doğrulama, gizlilik, sınırlar) konusunda bilgilendirilmeli — bu modülün öğrettikleri gibi. İyi politika YZ'yi **yasaklamaz** (faydası büyük) ama **çerçeveler**: faydayı serbest bırakırken riski yönetir. Eğer bir kurumda çalışıyorsan, mevcut YZ politikasını öğren ve uy; yoksa, ekibinle birlikte basit bir tane oluşturmak değerli bir adımdır. Ortak kurallar, herkesin YZ'den güvenle yararlanmasını sağlar.

Önce doğrula

KONTROL

Bir ekipte YZ politikası olduğunda — onaylı araçlar, net veri paylaşım kuralları, YZ çıktısının da insan incelemesinden geçmesi ve sorumlu kullanım eğitimi — herkes YZ'den tutarlı ve güvenli biçimde yararlanır. Politika YZ'yi yasaklamaz, çerçeveler: faydasını korurken sır sızıntısı, denetimsiz çıktı ve belirsiz sorumluluk gibi riskleri yönetir. Sonuç: YZ, kurumda bireysel bir risk değil, ortak ve güvenli bir güç hâline gelir.

Alıştırma

10 dk

Politika kur:

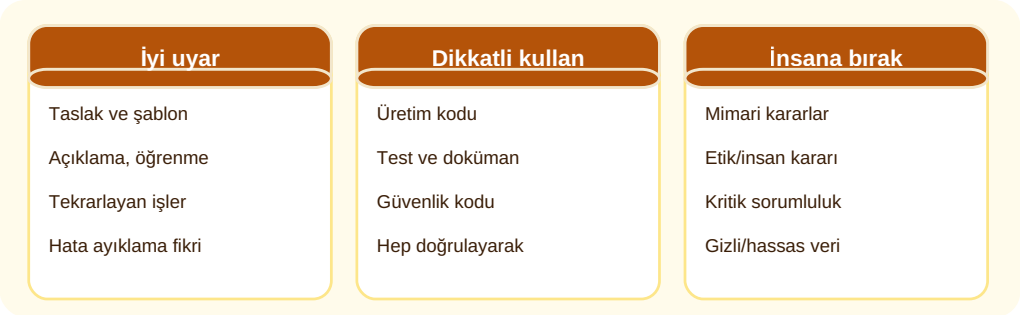
1. Bir ekip YZ politikasının netleştirmesi gereken üç şeyi yaz.
2. "YZ yazdı" demenin neden insan incelemesini atlatmaması gerektiğini açıkla.
3. İyi bir politikanın YZ'yi yasaklamak yerine neden "çerçevelemesi" gerektiğini belirt.

BÖLÜM 24

YZ ile İş Akışı Tasarlamak

YZ her işe uygun değildir; bazı işlerde harika, bazılarında riskli, bazılarında ise tamamen insana bırakılması gereken bir araçtır. Olgun kullanım, YZ'yi nerede kullanıp nerede kullanmayacağını bilerek bir iş akışı tasarlamaktır.

YZ'yi nereye koymalı?



Şema 24.1 — İş akışı: YZ'nin iyi uyduğu, dikkatli ve uygun olmadığı işler.

İPUCU

Olgun bir YZ kullanıcısı, "YZ'yi her yere koymak" ile "YZ'yi hiç kullanmamak" arasında **akıllı bir denge** kurar — işin türüne göre. **YZ'nin iyi uyduğu işler:** taslak/iskelet üretmek, tekrarlayan/şablon kod, açıklama ve öğrenme, hata ayıklama için fikir üretme, dokümantasyon taslağı — yani **insanın yönlendirip doğrulayabileceği, hata maliyeti düşük, hızlandırılabilir** işler. **Dikkatli kullanılacak işler:** üretim kodu, testler, güvenlik kodu — YZ yardımcı olur ama **her çıktı titizlikle doğrulanmalı**. **İnsana bırakılması gereken işler:** mimari ve tasarım kararları (geniş bağlam ve deneyim ister), **etik veya insanları etkileyen kararlar** (sorumluluk insanda olmalı), kritik güvenlik/sorumluluk noktaları ve **gizli/hassas verinin işlendiği** yerler. İyi iş akışı şu soruyu sorar: "Bu adımda YZ hızlandırır mı, yoksa risk mi ekler? Hatası ne kadar pahalı? Doğrulayabilir miyim?" Cevaba göre YZ'yi devreye al veya alma. Amaç, YZ'yi **doğru yerlerde** kullanıp gücünden yararlanmak, **yanlış yerlerde** kullanmaktan kaçınıp riski önlemek. Bu denge, bu modülün tüm öğrettiklerini pratik bir iş akışına dönüştürür.

Önce doğrula KONTROL

YZ'yi iş türüne göre konumlandıran bir akış tasarladığında — taslak/öğrenme/tekrar işlerinde serbest, üretim/test/güvenlik işlerinde dikkatli ve doğrulayarak, mimari/etik/hassas işlerde insana bırakarak — onun gücünden en çok yararlanır, riskini en aza indirirsin. Her adımda "YZ burada hızlandırır mı yoksa risk mi ekler?" diye sorarsın. Sonuç: YZ doğru yerlerde bir hızlandırıcı, yanlış yerlerde devre dışı kalır; akıllı bir denge kurarsın.

Alıştırma 12 dk

İş akışı tasarla:

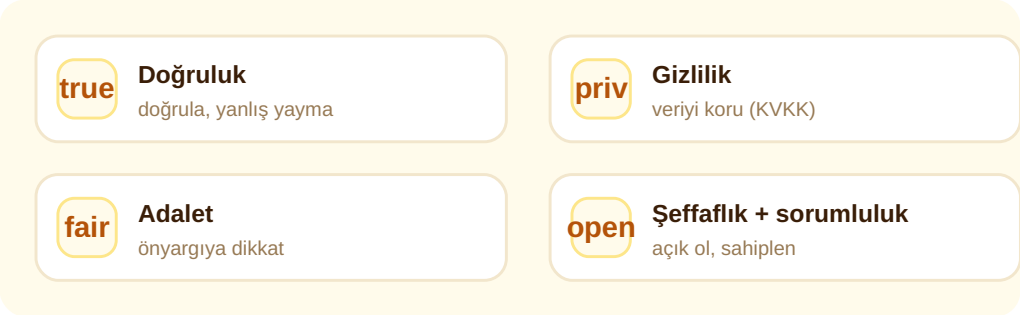
1. YZ'nin iyi uyduğu, dikkatli kullanılacak ve insana bırakılacak işlere birer örnek yaz.
2. Bir işte YZ kullanıp kullanmamaya karar verirken sorman gereken soruları belirt.
3. Neden etik/insan kararları YZ'ye bırakılmamalı, açıkla.

BÖLÜM 25

Etik ve Sorumluluk

YZ ile geliştirme, teknik olduğu kadar etik bir konudur. Ürettiğin yazılım insanları etkiler; YZ kullanman bu sorumluluğu azaltmaz, hatta yeni boyutlar ekler: doğruluk, gizlilik, adalet, şeffaflık. Olgun bir geliştirici, YZ'yi bu etik çerçevede kullanır.

YZ kullanımının etik boyutları



Şema 25.1 — YZ etiği: doğruluk, gizlilik, adalet, şeffaflık ve sorumluluk.

İPUCU

YZ ile geliştirmenin etik boyutu, bu modülün ilkelerini daha geniş bir sorumluluğa bağlar. **Doğruluk:** doğrulamadan YZ çıktısını yaymak (yanlış kod, yanlış bilgi), başkalarını yanlış yönlendirebilir — doğrulama bir etik sorumluluktur, sadece teknik bir adım değil. **Gizlilik:** insanların verisini (KVKK) YZ'ye sorumsuzca vermek, onların mahremiyetini ihlal eder; veriyi korumak başkalarına karşı bir borçtur. **Adalet:** YZ'nin önyargılarını (Seviye 3) fark etmeden, insanları etkileyen sistemlerde kullanmak ayrımcılık üretebilir — özellikle insanları sınıflandıran/değerlendiren kodda adaleti gözet. **Şeffaflık:** uygun olduğunda YZ'nin katkısını açıkça belirtmek (ekip/proje kurallarına göre) dürüstlüktür. **Sorumluluk:** en temel ilke — **ürettiğin her şeyin sorumluluğu sende**, "YZ yaptı" bir mazeret değil. Bunların hepsi, "iyi geliştirici" olmanın YZ çağındaki anlamını oluşturur: hızlı ama dikkatli, güçlü ama sorumlu. YZ muazzam bir araçtır; onu insanlara fayda sağlayacak, zarar vermeyecek biçimde kullanmak senin elinde. Teknik beceriyle etik sorumluluğu birleştiren geliştirici, bu çağın en değerli geliştiricisidir.

Önce doğrula

KONTROL

YZ'yi etik bir çerçevede kullandığında — çıktıyı doğrulayarak (doğruluk), veriyi koruyarak (gizlilik/KVKK), önyargıya dikkat ederek (adalet) ve katkıyı şeffafça belirtip sonucu sahiplenerek (şeffaflık/sorumluluk) — yalnızca çalışan değil, insanlara saygılı ve güvenilir yazılım üretirsin. YZ kullanman bu sorumlulukları azaltmaz; aksine onları daha bilinçli taşımanı gerektirir. Sonuç: teknik beceriyle etik duyarlılığı birleştiren, çağın en değerli türden geliştiricisi olursun.

Alıştırma

12 dk

Etik kullan:

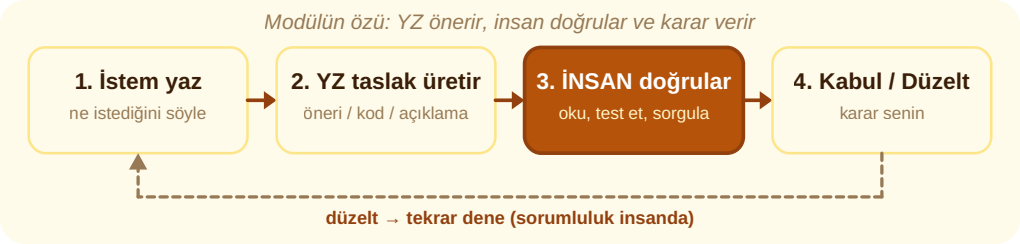
1. YZ kullanımının dört etik boyutunu (doğruluk, gizlilik, adalet, şeffaflık) yaz.
2. Doğrulamanın neden teknik olduğu kadar etik bir sorumluluk olduğunu açıkla.
3. "Ürettiğin her şeyin sorumluluğu sende" ilkesini kendi cümlenle belirt.

BÖLÜM 26

Bitirme: Sorumlu YZ Kullanım İlkeleri

Tüm modülü, YZ'yi sorumlu ve etkili biçimde kullanmak için bir ilkeler bütününde topluyoruz. Bu liste, YZ'yi bir risk değil bir güç hâline getiren alışkanlıkların özetidir — her YZ kullanımında başvurabileceğin bir pusula.

Sorumlu kullanımın özü



Şema 26.1 — Tüm modülün özü tek döngüde: öner → doğrula → karar ver.

Sorumlu YZ kullanım ilkeleri

```
[ ] Önce doğrula: çıktıyı oku, anla, test et
[ ] Halüsinasyonu varsay: özgüven = doğruluk DEĞİL
[ ] İnsan karar verir: sorumluluk sende
[ ] Sır/kişisel veri (KVKK) verme: önce temizle
[ ] Güvenliği ayrı kontrol et: "çalışıyor" yetmez
[ ] Beceriyi koru: anlamadan kabul etme
[ ] Sınırları bil: önyargı, eski bilgi, bağlam
[ ] Ajanları denetle: onay kapıları koy
[ ] Etik sahiplen: doğruluk, gizlilik, adalet
```

İPUCU

Bu ilkeler, modülün tamamının özüdür; her YZ kullanımında bir pusula olarak başvur. Bu modülle birlikte, çağımızın en güçlü geliştirme araçlarından birini **sorumlu biçimde** kullanmayı öğrendin: YZ'nin ne olduğunu ve nasıl çalıştığını (Seviye 1), günlük işte nasıl kullanılacağını (Seviye 2), güvenli ve sorumlu kullanımı (Seviye 3) ve olgun çalışmayı (Seviye 4). Bütünsel kavrayış tek bir cümlede: **YZ önerir, insan doğrular ve karar verir.** YZ, bu seride öğrendiğin tüm temellerin (HTML/CSS/JS'ten algoritma, veritabanı, güvenlik ve yayınlamaya) üstüne oturan bir **hızlandırıcıdır** — ama bu temeller olmadan onu güvenle yönlendiremez veya yanılgılarını yakalayamazsın. İşte bu yüzden YZ, öğrenmenin yerini almaz; **güçlü bir temeli olan kişinin elinde** en değerli hâline gelir. En önemli mesaj baştan beri aynıydı ve şimdi senin alışkanlığın olmalı: her özgüvenli çıktıyı **doğrula**, her hassas veriyi **koru**, her kararı ve sorumluluğu **sahiplen**. Bu refleksleri taşıyan biri için YZ, korkulacak bir tehdit değil, ustaca kullanılacak muazzam bir güçtür. Serinin son modülünde (Bitirme & Portfolyo), tüm bu öğrendiklerini birleştirip kendi projeni inşa edecek ve geliştirici yolculuğunu taçlandıracaksın.

Önce doğrula KONTROL

Bu ilkeleri alışkanlık hâline getirdiğinde — önce doğrulamak, halüsinasyonu varsaymak, insan kararını korumak, veriyi korumak, güvenliği ayrı denetlemek, beceriyi geliştirmek, sınırları bilmek, ajanları denetlemek ve etiği sahiplenmek — YZ senin için bir risk değil, sorumlu biçimde kullandığın muazzam bir güç olur. Hepsinin merkezinde tek bir döngü vardır: YZ önerir, sen doğrular ve karar verirsin. Sonuç: çağın en güçlü araçlarını, en olgun ve güvenli biçimde kullanan bir geliştirici olursun.

Alıştırma 20 dk

İlkelerini benimse:

1. Sorumlu YZ kullanım ilkelerini kendi cümlelerinle bir liste hâlinde yaz.
2. Bu modülde öğrendiğin en önemli üç ilkeyi ve nedenini belirt.
3. "YZ önerir, insan doğrular ve karar verir" döngüsünü bir örnekle anlat.
4. YZ'nin neden "güçlü bir temeli olan kişinin elinde" en değerli olduğunu açıkla.

EK

Kodlamada YZ Terimleri Sözlüğü

En sık kullanılan yapay zekâ destekli geliştirme terimleri. Bir başvuru kaynağı olarak saklayabilirsin.

YZ kod asistanı	`Öneri üreten araç`	Halüsinasyon	`Özgüvenli ama yanlış çıktı`
Doğrulama	`Çıktıyı kontrol etme`	İnsan denetimi	`Son karar insanda`
İstem (prompt)	`YZ'ye ne istediğini söyleme`	Bağlam	`İlgili bilgi (kod, hata, amaç)`
Tamamlama	`Satır içi öneri`	Sohbet asistanı	`Soru-cevap YZ`
Ajan	`Otonom, çok adımlı YZ`	Sır sızıntısı	`Anahtar/veri YZ'ye verme riski`
KVKK	`Kişisel veri koruma`	Lisans/atıf	`Üretilen kodun hakları`
Aşırı bağımlılık	`Beceriyi köreltme riski`	Önce doğrula	`Bu modülün altın kuralı`

Tek bir altın kural

KONTROL

Bu modülün tüm konuları (istem yazma, kod üretme, hata ayıklama, güvenlik, gizlilik, etik) tek bir merkez ilkenin etrafında döner: **YZ önerir, insan doğrular ve karar verir**. YZ güçlü bir hızlandırıcıdır ama "anlamadan tahmin eder", bu yüzden özgüvenle yanılabilir (halüsinasyon). Çıktısını her zaman **okur, anlar, test eder ve sorgularsın**; sırlarını ve kişisel veriyi (KVKK) korursun; güvenlik açıklarını kontrol edersin; ve son kararın — ve sorumluluğun — senin olduğunu unutmazsın. Bu refleksi kazanan biri için YZ muazzam bir güçtür; kazanmayan için ise sessiz bir risktir. İleri seviyelerde günlük kullanımı, güvenliği, etiği ve olgun iş akışını ele alacağız — ama altın kural hep aynı kalacak: **önce doğrula**.