

# Chain Prompting Yöntemi ile Asal Sayı Kontrolü

## Giriş

Bu projede, kullanıcıdan alınan sayının asal olup olmadığını kontrol eden bir Python fonksiyonu geliştirilmiştir. Fonksiyon, kullanıcıdan aldığı sayıyı kontrol eder, geçersiz girişler için hata mesajları döndürür ve kullanıcı nın girdiği sayının asal olup olmadığını belirler. Ayrıca, fonksiyon birden fazla sayıyı kontrol edebilme özelliğine de sahiptir. Raporda, her adımda yapılan değişiklikler ve fonksiyonun işleyişi ayrıntılı olarak açıklanacaktır.

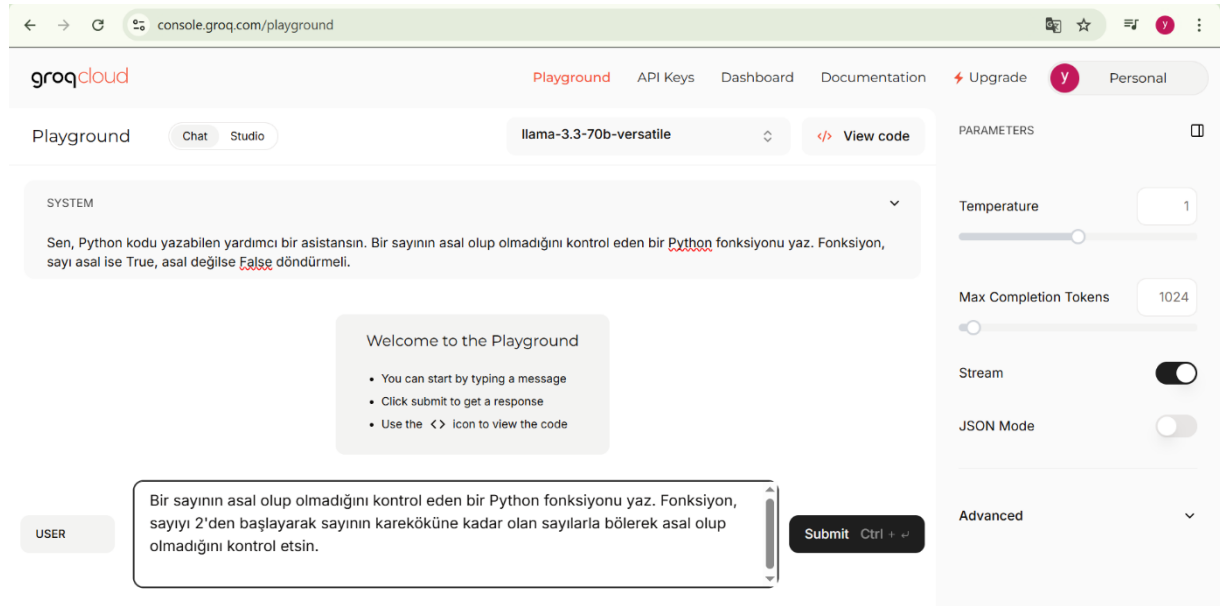
## Kullanılan Ortam ve Model

- Kullanılan Ortam: Groq Playground (<https://console.groq.com/playground>)
- Kullanılan Model: Llama-3.3-70b-Versatile

## Amaç

Bu ödevde, Chain Prompting yöntemi ile modelin bir sayının asal olup olmadığını belirlemek için verilen adımların her birini iyileştirmek hedeflenmiştir. Her adımda, modelin önceki çıktısı, bir sonraki adımdaki yeni soruyu şekillendirmek amacıyla kullanılarak, sonuçların doğruluğu ve kalitesi artırılmaya çalışılmıştır.

## Adım 1: Temel Asal Sayı Kontrolü Fonksiyonunun Oluşturulması



Şekil 1 - Adım 1 : System ve User Prompt

System Propmt :

*Sen, Python kodu yazabilen yardımcı bir asistansın. Bir sayının asal olup olmadığını kontrol eden bir Python fonksiyonu yaz. Fonksiyon, sayı asal ise True, asal değilse False döndürmeli.*

User Propmt :

*Bir sayının asal olup olmadığını kontrol eden bir Python fonksiyonu yaz. Fonksiyon, sayıyı 2'den başlayarak sayının kareköküne kadar olan sayılarla bölerek asal olup olmadığını kontrol etsin.*

```
def asal_mi(sayı):
    """
    Bir sayının asal olup olmadığını kontrol eden fonksiyon.

    Args:
        sayı (int): Kontrol edilecek sayı.

    Returns:
        bool: Sayı asal ise True, asal değilse False.
    """
    if sayı <= 1: # 1 ve daha küçük sayılar asal değildir
        return False
    for i in range(2, int(math.sqrt(sayı)) + 1): # 2'den başlayarak kareköke kadar Controlled
        if sayı % i == 0: # Sayı i ile tam bölünüyor mu?
            return False # Sayı asal değil
    return True # Sayı asal
```

Temperature 1

Max Completion Tokens 1024

Stream ☒

JSON Mode ☐

Şekil 2 - Adım1 : Temel Asal Sayı Kontrol Fonksiyon

Şekil 2’de ilk adımda elde ettiğimiz temel asal sayı kontrol fonksiyonu yer almaktadır. Bu fonksiyon, geçersiz girişleri kontrol etmekte ve sayı 0 veya 1 olduğunda doğru hata mesajlarını döndürmektedir. Ayrıca, asal sayılar için geçerli kontrol mekanizması (2’den kareköküne kadar olan bölme işlemi) uygulanmıştır.

## Adım 2: Kullanıcıdan Sayı Girişi Alınarak Asallık Kontrolü

User Propmt :

*Yukarıdaki fonksiyonu geliştir. Kullanıcı negatif bir sayı girdiğinde veya 0/1 gibi geçersiz bir giriş yaparsa uygun bir hata mesajı döndürsün.*

```
def asal_mi(sayı):
    """
    Bir sayının asal olup olmadığını kontrol eden fonksiyon.

    Args:
        sayı (int): Kontrol edilecek sayı.

    Returns:
        bool: Sayı asal ise True, asal değilse False.
        str: Geçersiz giriş yapılmış ise hata mesajı.
    """
    # Geçersiz giriş kontrolü
    if not isinstance(sayı, int):
        return "Hata: Girdi bir sayı olmalıdır."
    if sayı < 0:
        return "Hata: Asal sayılar pozitifdir. Lütfen pozitif bir sayı giriniz."
    if sayı <= 1:
        return "Hata: 0 ve 1 asal sayı değildir."
```

gropcloud Playground API Keys Dashboard Documentation Upgrade Personal

Playground Chat Studio llama-3.3-70b-versatile View code

PARAMETERS

Temperature 1

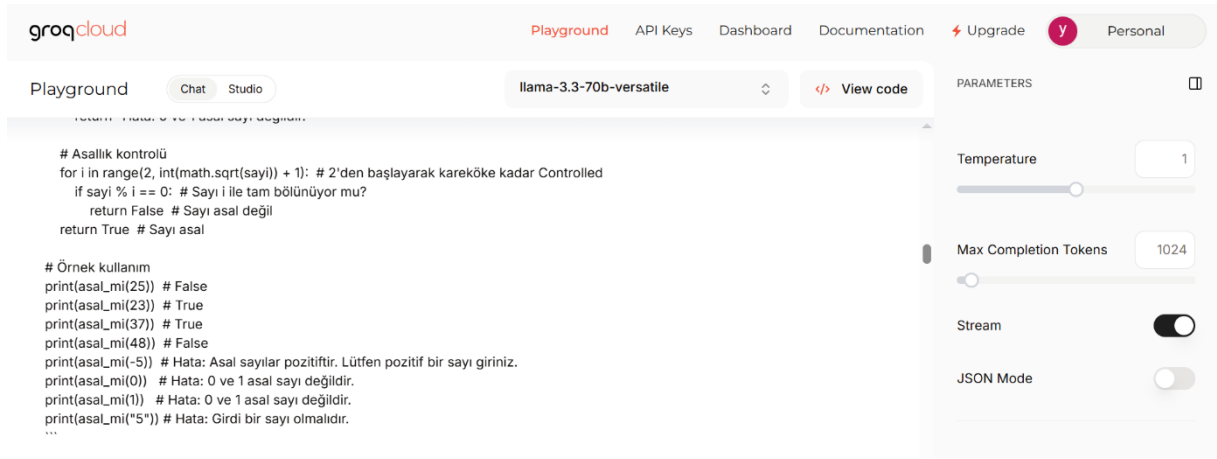
Max Completion Tokens 1024

Stream ☒

JSON Mode ☐

Advanced

Şekil 3 - Adım 2 : Kullanıcıdan Sayı Girişi Alınarak Asallık Kontrolü



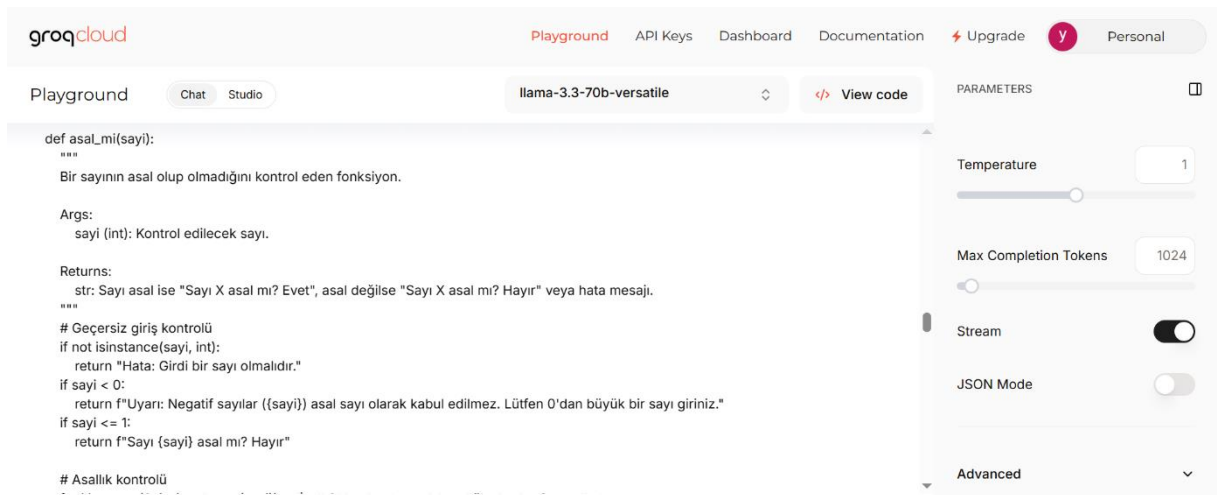
Şekil 4 - Adım 2 : Kullanıcıdan Sayı Girişi Alınarak Asallık Kontrolü(Devamı)

Şekil 3 ve 4’de gördüğümüz ikinci adımda geliştirilen fonksiyonda, kullanıcıdan bir sayı girmesi isteniyor ve bu sayının asal olup olmadığı kontrol ediliyor. Eğer kullanıcı negatif bir sayı veya geçersiz veri girerse, hata mesajları döndürülmektedir. Bu, kullanıcının doğru veri girmesini sağlamak için gereklidir.

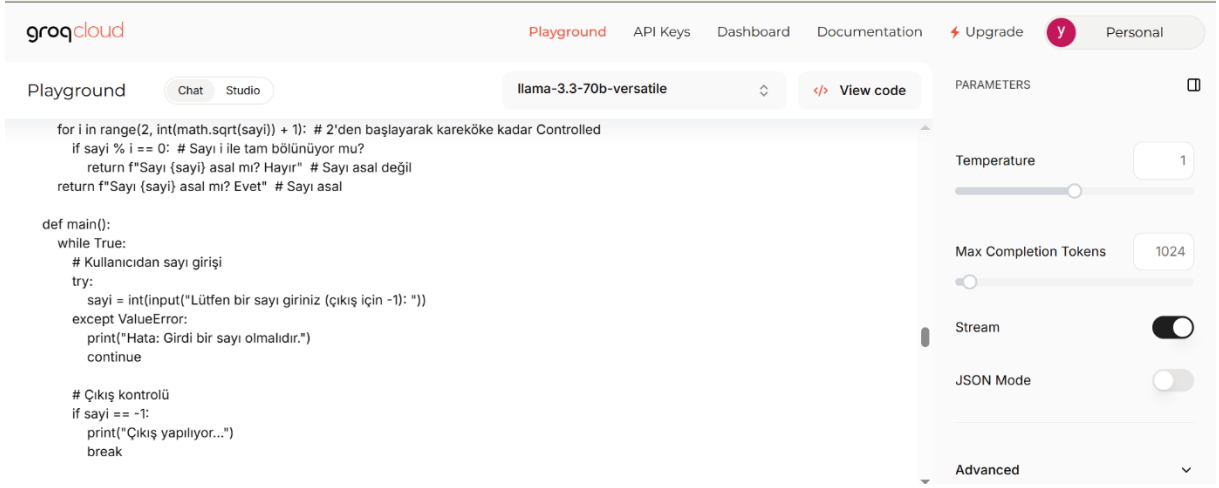
### Adım 3: Chain Prompting ile Asal Sayı Kontrolünün İyileştirilmesi

#### User Propmt :

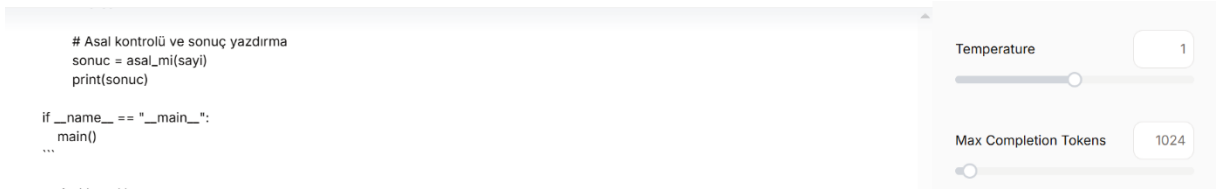
*Yukarıdaki fonksiyonu geliştir. Fonksiyon, kullanıcıdan alınan sayının asal olup olmadığını kontrol ettikten sonra, sayının asal olup olmadığını yazdırmak yerine, "Sayı X asal mı? Evet/Hayır" şeklinde bir formatta geri dönsün. Ayrıca, negatif sayılarla yapılan işlemlerde kullanıcıyı uyararak bir açıklama ekleyin ve fonksiyona `while` döngüsü ekleyerek birden fazla sayı kontrol edilmesine imkan verin. Her sayının sonucunu yazdıran bir fonksiyon haline getirin.*



Şekil 5 - Adım 3: Chain Prompting ile Asal Sayı Kontrolünün İyileştirilmesi



Şekil 6 - Adım 3: Chain Prompting ile Asal Sayı Kontrolünün İyileştirilmesi



Şekil 7 - Adım 3: Chain Prompting ile Asal Sayı Kontrolünün İyileştirilmesi

Bu adımda, ilk fonksiyon üzerinde geliştirmeler yapıldı. Kullanıcıyı negatif sayılarla ilgili uyarın bir açıklama eklenmiş ve fonksiyon, sayının asal olup olmadığını daha detaylı bir şekilde geri döndürmeye başlamıştır. Her sayıya özel mesajlar ile kullanıcı bilgilendirilmektedir. Ayrıca, kullanıcıya birden fazla sayı girebilme imkânı sunulmuş, while döngüsü ile sürekli bir giriş işlemi sağlanmıştır.

#### Adım 4: Modelin Verilen Adımlarda Sonuçları Doğru Bir Şekilde Döndürmesi

##### System Propmt:

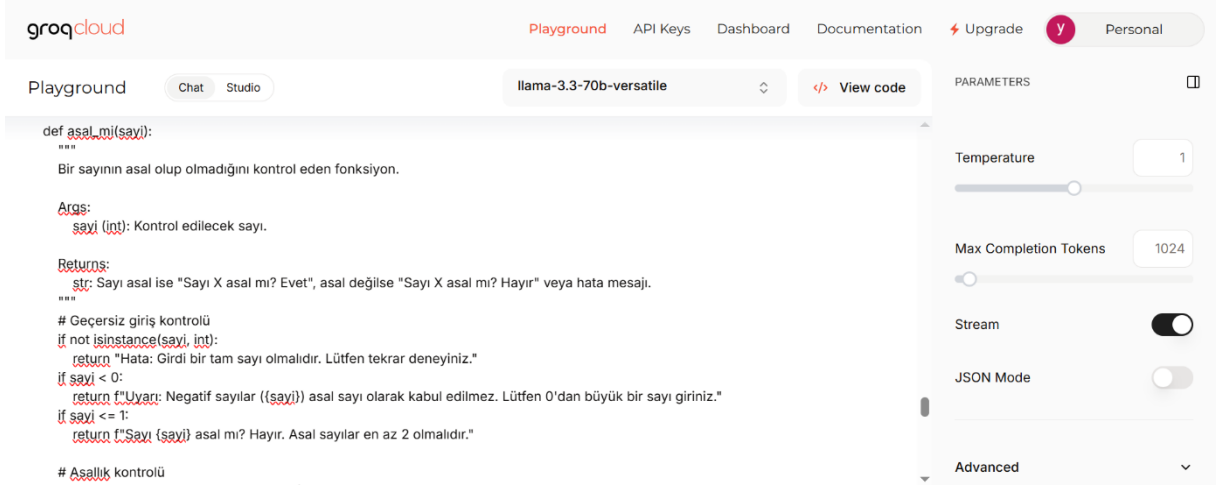
*Kullanıcının sayıyı girmesini bekleyin, ardından her sayı için aşağıdaki kontrolleri yapın:*

- 1. Eğer sayı geçersizse (örneğin, negatif bir sayı ya da string), uygun bir hata mesajı gösterin.*
- 2. Eğer sayı asal değilse, sayının asal olmadığına dair geri bildirimde bulunun.*
- 3. Eğer sayı pozitif ve geçerliyse, asal mı değil mi olduğunu belirten bir mesaj verin. Ayrıca negatif sayılar için kullanıcıyı bilgilendirecek bir açıklama ekleyin.*
- 4. Kullanıcı bir sayı girdikten sonra bir sonucu görmelidir.*

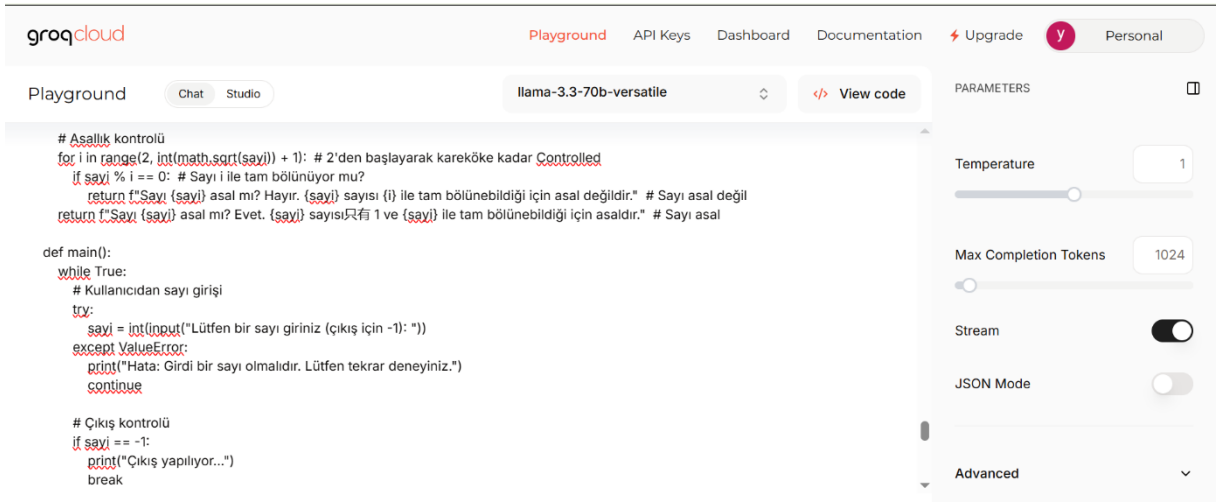
##### User Propmt :

*Yukarıdaki fonksiyonu geliştir. Şimdi, fonksiyon bir sayı girdikten sonra sadece "Sayı X asal mı? Evet/Hayır" sonucunu yazmakla kalsın, aynı zamanda her sayı için girilen sayının asal olup*

olmadığını kontrol ettikten sonra, sayının asal olup olmadığını belirtmekle birlikte, negatif sayılar için kullanıcıyı bilgilendirici açıklama da eklesin. Ayrıca fonksiyon, kullanıcı bir sayı girdiğinde, pozitif sayılarla ve geçersiz girişlerle ilgili daha ayrıntılı hata mesajları verecek şekilde tekrar yapılandırılmalı.



Şekil 8 - Adım 4: Modelin Verilen Adımlarda Sonuçları Doğru Bir Şekilde Döndürmesi



Şekil 9 - Adım 4: Modelin Verilen Adımlarda Sonuçları Doğru Bir Şekilde Döndürmesi

```
# Asal kontrolü ve sonuç yazdırma
sonuç = asal_mi(sayi)
print(sonuç)

if __name__ == "__main__":
    main()
...
```

Şekil 10 - Adım 4: Modelin Verilen Adımlarda Sonuçları Doğru Bir Şekilde Döndürmesi

Bu adımda, Chain Prompting kullanarak fonksiyon üzerinde önemli geliştirmeler yapıldı. Kullanıcıdan alınan sayının asal olup olmadığı kontrol edilirken, önceki adımlardan farklı olarak, negatif sayılar için

uyarı mesajları eklendi ve geçersiz girişlerde daha ayrıntılı hata mesajları sunuldu. Ayrıca, sayı pozitifse ve geçerliyse, asal olup olmadığına dair daha açıklayıcı bir yanıt verildi.

Kullanılan user ve system promptlarıyla birlikte fonksiyonun daha iyi çalışması sağlandı. Kullanıcı bir sayı girdiğinde, fonksiyon artık sadece sayının asal olup olmadığını belirtmekle kalmayıp, aynı zamanda negatif sayılar ve geçersiz girişler için bilgilendirici açıklamalar da sunmaktadır.

## **Model Çıktı Karşılaştırması ve Analizi**

İlk sürümde, temel asal sayı kontrol fonksiyonu kullanılmıştır. Bu sürümde model, geçersiz girişleri kontrol etmekte ve sayılar 0 veya 1 olduğunda doğru hata mesajlarını döndürmektedir. Ayrıca, asal sayılar için geçerli kontrol mekanizması (2'den kareköküne kadar olan bölme işlemi) uygulanmıştır. Ancak bu sürümde, yalnızca temel doğrulama yapılmakta ve kullanıcıya çok fazla bilgi sunulmamaktadır. Negatif sayılar ve geçersiz girişler konusunda açıklamalar eksiktir. Bu sürüm, doğruluk açısından temel gereksinimleri karşılarsa da, kullanıcı deneyimi bakımından daha gelişmiş özelliklere ihtiyaç duyulmaktadır.

İkinci sürümde ise, modelin çıktıları daha kullanıcı dostu hale getirilmiştir. Kullanıcıdan alınan sayıya dair negatif sayılar ve geçersiz verilerle ilgili daha ayrıntılı hata mesajları eklenmiştir. Bu sürümde, kullanıcı doğru veri girmediğinde, model ona yönelik bilgilendirici mesajlar sunmakta ve hatalı verilerle ilgili uyarılar yapmaktadır. Bu iyileştirmeler sayesinde, modelin doğruluğu artırılmış ve kullanıcı deneyimi iyileştirilmiştir. Ancak bu sürümde, sayının asal olup olmadığına dair verilen geri bildirim yine temel düzeyde kalmaktadır. Bu nedenle, daha ayrıntılı açıklamalar ve daha fazla etkileşimli geri bildirimler hala gereklidir.

Üçüncü sürümde, modelin işleyişi daha da geliştirilmiştir. Kullanıcı, negatif sayılar hakkında uyarı alırken, fonksiyon sayının asal olup olmadığını daha detaylı bir şekilde geri döndürmektedir. Her sayıya özel mesajlar eklenmiş ve kullanıcıya birden fazla sayı girme imkânı sağlanmıştır. Bu, kullanıcının etkileşimini artırarak, modelin daha esnek ve kullanıcı dostu olmasına olanak tanımıştır. Ayrıca, while döngüsü kullanılarak sürekli bir giriş işlemi sağlanmış ve kullanıcıdan gelen veriler daha verimli bir şekilde işlenmiştir. Bu sürüm, kullanıcı deneyimini iyileştirirken, modelin doğruluğunu da artırmıştır.

Dördüncü sürümde ise, Chain Prompting kullanılarak fonksiyon daha da geliştirildi. Bu sürümde, modelin çıktıları daha kapsamlı hale gelmiştir. Negatif sayılar için uyarılar ve geçersiz girişler için daha ayrıntılı hata mesajları eklenmiştir. Ayrıca, sayının asal olup olmadığına dair daha açıklayıcı ve bilgilendirici geri bildirimler sunulmuştur. Kullanıcı, negatif sayılar veya geçersiz girişler girdiğinde bilgilendirilirken, geçerli bir pozitif sayı girildiğinde asal olup olmadığı hakkında detaylı bilgi verilmiştir. Bu sürüm, kullanıcıdan alınan veriye daha fazla dikkat edilmesini sağlamak ve kullanıcı deneyimini olumlu bir şekilde etkilemektedir. Ayrıca, Chain Prompting ile yapılan iyileştirmeler, modelin esnekliğini artırmış ve kullanıcı geri bildirimlerini daha etkili bir şekilde işlemesini sağlamıştır.

Sonuç olarak, her bir sürümde yapılan geliştirmelerle birlikte modelin doğruluğu, kullanıcı deneyimi ve esnekliği artırılmıştır. Dördüncü sürüm, tüm bu iyileştirmeleri en iyi şekilde birleştirerek, kullanıcı geri bildirimlerini etkili bir şekilde işleyen ve kullanıcı hatalarını minimize eden bir çözüm sunmaktadır.