

# 1. GİRİŞ

Bu rapor, Yazılım İnşası dersi kapsamında geliştirilen **Zar Oyunu (Dice Game)** projesinin final aşamasını ve sonuçlarını kapsamaktadır.

Proje, dönem başında hazırlanan Analiz Raporu ve daha sonra oluşturulan Tasarım Raporu temel alınarak gerçekleştirilmiştir. Geliştirilen uygulama, kullanıcıların karşılıklı olarak oyun oynayabildiği, görsel arayüzü olan ve puan takibi yapabilen web tabanlı bir sistemdir.

Final raporunun temel amacı; sistemin kodlama ve oluşturulma sürecini detaylı biçimde belgelemek, analiz aşamasında belirlediğimiz hedeflerin ne kadarının başarıldığını değerlendirmek, yapılan geliştirmeleri açıklamak ve oyunun nasıl kullanılacağını görsellerle destekleyerek anlatmaktır. Sadece çalışan bir yazılım ortaya koymak değil, aynı zamanda bu yazılımın nasıl bir süreçten geçerek oluşturulduğunu göstermek hedeflenmiştir.

Bu kapsamda raporda şu konular detaylı olarak ele alınmıştır:

- Uygulamanın geliştirilme süreci ve izlenen adımlar,
- Analiz ve tasarım raporlarına ne kadar uyulduğu,
- Sistemin sahip olduğu özellikler ve kullanıcı ekranları,
- Kodlama sırasında karşılaşılan zorluklar ve bunlara bulunan çözümler,
- Sistemin eksik kalan yönleri ve ileride yapılabilecek eklemeler.

Proje boyunca, karmaşık ve anlaşılması zor bir yapı kurmak yerine; düzenli, hatasız çalışan ve kullanıcıların kolayca anlayabileceği bir oyun geliştirmeye öncelik verilmiştir. Hazırlanan bu rapor, projenin sadece son halini değil, tüm üretim hikayesini bütüncül bir bakış açısıyla ortaya koymaktadır.

## 2. GERÇEKLEŞTİRİM SÜRECİ (Revize Edilmiş Güvenli Versiyon)

Zar Oyunu Projesi, modern web standartları (HTML, CSS ve JavaScript) temel alınarak geliştirilmiştir. Uygulama, analiz raporunda belirtilen sistem kısıtlarına (kurulum gerektirmez, tarayıcı tabanlı çalışma) tam uyumlu şekilde tasarlanmış ve kodlanmıştır.

Gerçekleştirim süreci şu adımlar izlenerek yürütülmüştür:

- **Analiz ve Tasarım İncelemesi:** Dönem başında hazırlanan Analiz Raporunda tanımlanan fonksiyonel gereksinimler ve Tasarım Raporunda belirlenen arayüz yapısı ve yerleşim planı detaylı biçimde gözden geçirilmiş; geliştirme sürecinde bu hedeflere sadık kalınması amaçlanmıştır.
- **Proje Kurulumu ve Dosya Yapısı:** Proje, karmaşıklığı önlemek ve düzeni sağlamak adına tek bir dosya yerine, işlevlerine göre ayrılmış (HTML, CSS, JS) bir klasör yapısı oluşturularak başlatılmıştır.
- **Katmanlı Kodlama:** Sistemin daha anlaşılır ve yürütülebilir olması için kodlama işlemi üç temel katmanda gerçekleştirilmiştir:
  - **Yapısal Katman (HTML):** Oyunun iskeletini oluşturan giriş ekranı, oyun tahtası ve sonuç pencereleri oluşturuldu.
  - **Görsel Katman (CSS):** Kullanıcı deneyimini iyileştirmek için renkler, buton tasarımları ve zar animasyonları bu aşamada eklendi.
  - **İş Mantığı Katmanı (JavaScript):** Oyunun kurallarını işleten (zarın rastgele gelmesi, puanın hesaplanması, kazananın belirlenmesi) tüm algoritmalar bu katmanda kodlandı

- **Modülerlik:** Her dosya ve fonksiyon, sadece kendi görevini yapacak şekilde tasarlanarak kod karmaşasının önüne geçildi.
- **Test ve Doğrulama:** Oyunun her bir parçası (isim girme, zar atma, oyun bitirme) geliştirdikten sonra manuel olarak test edildi. Farklı senaryolar (ardışık tıklama, oyun bitince tıklama vb.) denenerek hatalar giderildi.
- **Son Dokunuşlar:** Kullanıcıya gösterilen mesajlar (Kazanma tebriği, sıra bildirimleri) anlaşılır hale getirildi ve oyunun görsel bütünlüğü sağlandı.

Gerçekleştirim süreci yaklaşık 2 hafta sürmüş olup, analiz aşamasında planlanan tüm özellikler sisteme başarıyla entegre edilmiştir.

### 3. ANALİZ VE TASARIM RAPORLARI İLE UYUM

Sistem, analiz raporunda belirlenen gereksinimlere ve tasarım raporundaki arayüz kararlarına tam uyum sağlayacak şekilde geliştirilmiştir.

#### 3.1. Fonksiyonel Gereksinimlerin Karşılama Durumu

Analiz raporunda kodlanan fonksiyonel gereksinimlerin (FR) gerçekleştirim durumu şöyledir:

	Gereksinim Adı	Durum	Açıklama
	Oturum Yönetimi	<b>Karşılandı</b>	İsim girişi, varsayılan atama (Player 1/2) ve ekran geçişi sağlandı.
	Zar Mekanizması ve Algoritma	<b>Karşılandı</b>	Rastgele sayı üretimi (1-6), animasyon ve ses senkronizasyonu uygulandı.
	Oyun Mantığı ve Karar Mekanizması	<b>Karşılandı</b>	Sıra takibi (Turn-Based), skor hesaplama ve galibiyet kontrolü kodlandı.
	Sistem Sıfırlama ve Kontrol	<b>Karşılandı</b>	Oyun yenileme (Refresh) ve hatalı giriş yönetimi sisteme eklendi.

### 3.2. Fonksiyonel Olmayan Gereksinimlerin Karşılanması

Sistemin kalite standartlarını belirleyen teknik kriterler (NFR) şu şekilde hayata geçirilmiştir:

Kod	Kriter	Durum	Açıklama
NFR1	Kullanılabilirlik (Usability)	Karşılandı	Sade arayüz, okunabilir fontlar ve anlık kullanıcı yönlendirmeleri.
NFR2	Performans (Performance)	Karşılandı	Hızlı tepki süresi (<1 sn) ve minimum bellek kullanımı.
NFR3	Güvenilirlik (Reliability)	Karşılandı	Try-catch blokları ile hata yakalama ve akış güvenliği.
NFR4	Taşınabilirlik (Portability)	Karşılandı	Tüm güncel tarayıcılarda ve mobil cihazlarda (Responsive) uyumlu çalışma.
NFR5	Bakım Kolaylığı (maintainability)	Karşılandı	Modüler kod yapısı ve merkezi değişken kullanımı.

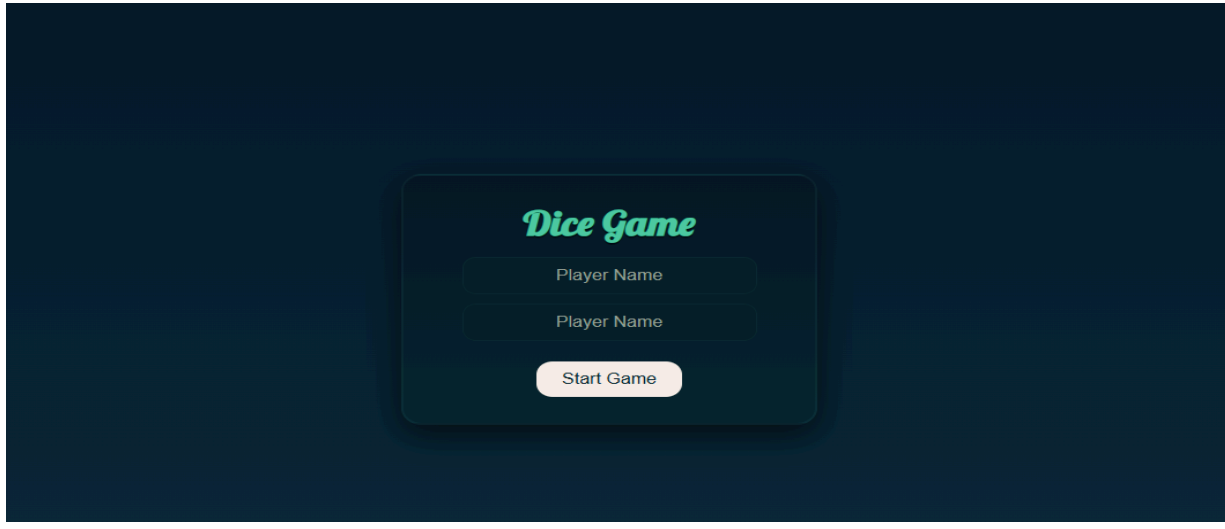
## 4. GERÇEKLEŞTİRİLEN SİSTEM ÖZELLİKLERİ

Bu bölümde, analiz ve tasarım aşamalarından geçerek kodlanan sistemin, son kullanıcıya sunduğu özellikler ve arayüz detayları açıklanmıştır.

### 4.1. Oyuncu Giriş Modülü (Login Screen)

Uygulama başlatıldığında kullanıcıyı karşılayan ilk modüldür. Analiz raporunda belirtilen **FR1** gereksinimi kapsamında geliştirilmiştir.

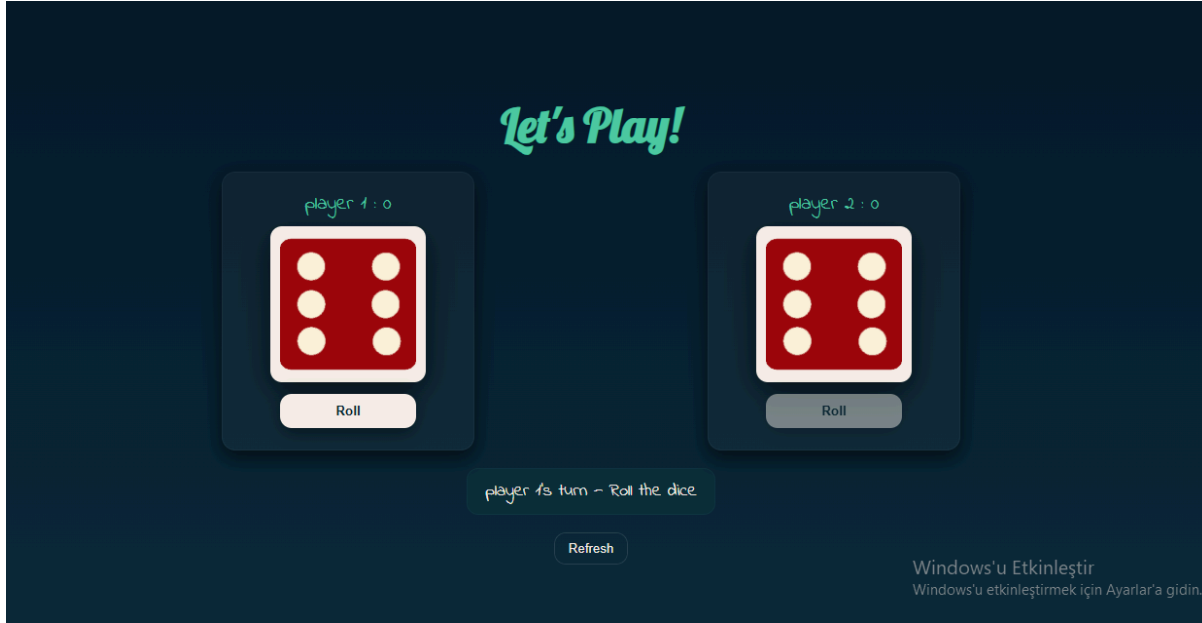
- İşlev:** Kullanıcılardan iki farklı oyuncu ismi talep edilir.
- Varsayılan Değer Kontrolü:** Eğer kullanıcı isim girmeden "Start" butonuna basarsa, sistem hata vermez; otomatik olarak "Player 1" ve "Player 2" isimlerini atar.
- Geçiş Efekt:** İsimler onaylandığında, JavaScript DOM manipülasyonu ile giriş paneli gizlenir ( .hidden sınıfı eklenir) ve ana oyun sahnesi görünür hale gelir.



## 4.2. Oyun Arayüzü ve Kontroller

Oyun sahnesi, rekabeti vurgulamak amacıyla ekranı iki eşit parçaya bölen simetrik bir yapıda tasarlanmıştır (**NFR1 - Kullanılabilirlik**).

- **Oyuncu Panelleri:** Sol tarafta 1. Oyuncu, sağ tarafta 2. Oyuncu yer alır. Her panelde oyuncunun ismi, anlık skoru, zar görseli ve kontrol butonu bulunur.
- **Durum Çubuğu (Status Bar):** Ekranın orta alt kısmında yer alan bu alan, kullanıcıya sürekli geri bildirim verir (Örn: "Sıra Player 1'de", "Player 2 kazandı" vb.).



## 4.3. Zar Mekanizması ve Tur Yönetimi

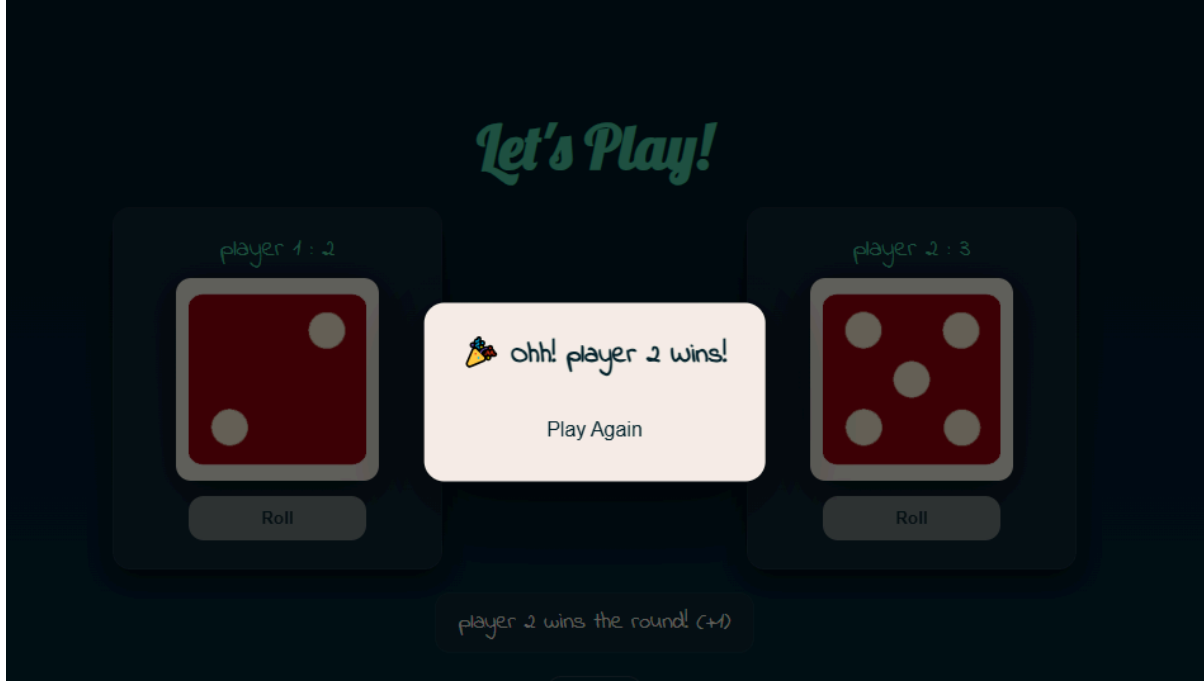
Oyunun çekirdek mekanizmasıdır. Analiz raporundaki **FR2** ve **FR3** maddelerine uygun olarak şu özellikler entegre edilmiştir:

1. **Sıra Kontrolü (Turn-Based):** Sistem, sırası gelmeyen oyuncunun butonunu pasif (disabled) hale getirir. Böylece oyuncuların aynı anda basması veya sıra ihlali yapması engellenir.
2. **Animasyon:** Butona basıldığında CSS ile hazırlanan shake animasyonu 0.6 saniye boyunca çalışır.
3. **Rastgelelik:** JavaScript `Math.random()` fonksiyonu ile 1-6 arasında tamamen rastgele bir sayı üretilir ve ilgili zar görseli ekrana basılır.

#### 4.4. Sonuç Ekranı (Winner Overlay)

Oyun sonsuz değildir; analizde belirlenen "3 puana ulaşan kazanır" kuralı işletilir.

- Bir oyuncu hedef skora ulaştığında arka plan karartılır.
- Ekranın ortasında kazananı tebrik eden dinamik bir pencere (Popup) açılır.
- Bu aşamada oyun durdurulur ve sadece "Play Again" (Tekrar Oyna) seçeneği sunulur.



### 5. KULLANICI KILAVUZU

#### 5.1. Programın Başlatılması

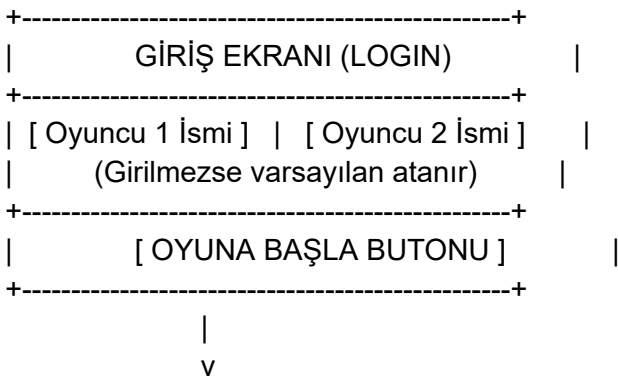
Zar Oyunu, tarayıcı tabanlı (web) bir uygulama olduğu için herhangi bir kurulum gerektirmez.

1. Proje klasörünü bilgisayarınıza indirin.
2. Klasör içerisindeki `index.html` dosyasına çift tıklayın veya dosyayı tarayıcınızın (Chrome, Edge, Firefox vb.) içine sürükleyin.

#### 5.2. Oynanış Senaryosu ve Akış Şeması

Aşağıdaki diyagram, tipik bir oyun sürecindeki kullanıcı ve sistem etkileşimini modellemektedir:

##### Evre 1: Giriş ve Kurulum



(Sistem arayüzü yükler)

|  
v

```
+-----+
| ANA OYUN SAHNESİ |
+-----+
```

**Evre 2: Oyun Döngüsü (Turn-Based Loop)** Bu döngü, bir oyuncu kazanana kadar tekrarlanır:

BAŞLANGIÇ (Sıra 1. Oyuncuda)

|  
v

```
+-----+
| 1. OYUNCU HAMLESİ |
| - "Roll" butonuna basar | ---> (Buton Kilitlenir)
| - Zar animasyonu oynar |
+-----+
```

|  
v

```
+-----+
| 2. OYUNCU HAMLESİ |
| - "Roll" butonuna basar | ---> (Buton Kilitlenir)
| - Zar animasyonu oynar |
+-----+
```

|  
v

```
+-----+
| SİSTEM DEĞERLENDİRMESİ |
| IF (Zar1 > Zar2) -> P1++ |
| IF (Zar2 > Zar1) -> P2++ |
| IF (Eşit) -> Puan Yok |
+-----+
```

|  
v

(Bir oyuncu 3 puana ulaştı mı?)

/ \

[HAYIR] [EVET]

| |

(Başa Dön) (Bitişe Git)

**Evre 3: Oyun Sonu**

|

v

```
+-----+
| KAZANAN EKRANI (POPUP) |
+-----+
|
| 🎉 PLAYER X KAZANDI! 🎉 |
|
| [ TEKRAR OYNA BUTONU ] |
+-----+
```

|  
v

(Skorlar Sıfırlanır)

## 6. KARŞILAŞILAN ZORLUKLAR VE ÇÖZÜMLER

Geliştirme sürecinde karşılaşılan teknik ve mantıksal zorluklar, mühendislik yaklaşımıyla analiz edilmiş ve aşağıdaki yöntemlerle çözüme kavuşturulmuştur.

### 6.1. Eşzamansızlık (Senkronizasyon) Problemi

- **Sorun:** Geliştirme aşamasında, butona basıldığı anda zar animasyonu henüz bitmeden puanın değiştiği ve sonucun ekrana yansıdığı tespit edilmiştir. Bu durum, oyunun gerçeklik algısını bozmaktaydı.
- **Çözüm:** İş Mantığı katmanına setTimeout fonksiyonu ile bir gecikme (delay) mekanizması entegre edildi. Mantıksal hesaplamaların (puan artışı, kazanan kontrolü), görsel animasyon süresi (650 ms) tamamlandıktan sonra tetiklenmesi sağlanarak tam senkronizasyon kuruldu.

### 6.2. Sıra Yönetimi ve Çakışmalar (Concurrency)

- **Sorun:** Oyuncuların sıra kendinde değilken butona basabilmesi veya her iki oyuncunun aynı anda hamle yapmaya çalışması, oyun algoritmasının kilitlenmesine yol açıyordu.
- **Çözüm:** "Durum Kontrolü (State Check)" mekanizması geliştirildi. Sıra 1. oyuncudayken, 2. oyuncunun butonu DOM manipülasyonu ile disabled (pasif) konuma getirildi. Böylece kullanıcı hataları yazılımsal olarak engellendi.

### 6.3. Veri Tutarlılığı ve Hata Yönetimi

- **Sorun:** Ses dosyalarının yüklenemediği durumlarda veya tarayıcı uyumsuzluklarında JavaScript kodunun çalışmayı durdurduğu gözlemlendi.
- **Çözüm:** Kritik fonksiyonlar try-catch blokları içerisine alınarak, ses dosyası oynatılamaz olsa bile oyunun hata vermeden (crash olmadan) akışına devam etmesi sağlandı.

## 7. SONUÇ VE DEĞERLENDİRME

Bu çalışma kapsamında geliştirilen Zar Oyunu Sistemi, yazılım mühendisliği yaşam döngüsünün analiz, tasarım ve gerçekleştirim aşamalarını eksiksiz ve tutarlı biçimde içeren bütüncül bir projedir.

Sistem; nesne yönelimli prensiplere uygun, katmanlı (HTML-CSS-JS), modüler ve geliştirilebilir bir mimari sunmaktadır. Analiz raporunda tanımlanan tüm kullanım senaryoları (FR1-FR4) başarıyla hayata geçirilmiş; tasarım raporunda belirlenen arayüz yapısı ve kullanıcı deneyimi hedefleri kodda birebir uygulanmıştır.

Proje sürecinde:

- Statik bir web sayfasının nasıl dinamik bir uygulamaya dönüştüğü deneyimlenmiş,
- Olay güdümlü (Event-Driven) programlama mantığı pekiştirilmiş,
- Karşılaşılan senkronizasyon ve mantık hataları etkili algoritmik çözümlerle aşılmıştır.

Sonuç olarak; kullanıcı dostu, hatasız çalışan ve teknik gereksinimleri karşılayan bir ürün ortaya konulmuştur. Elde edilen bu tecrübe, gelecekteki daha kapsamlı yazılım projeleri için sağlam bir temel oluşturacaktır.