# ZAR OYUNU
# TASARIM RAPORU (DESIGN REPORT)
# Yazılım İnşası Dersi

# 1. Sistemin Amacı ve Tasarım Hedefleri

## 1.1 Sistemin Amacı

Bu projenin amacı, iki kişilik bir zar oyununu web tabanlı olarak geliştirmek ve kullanıcıların etkileşimli bir oyun deneyimi yaşamasını sağlamaktır. Oyun, sırayla zar atma ve yüksek zar atan oyuncunun puan kazanması mantığı üzerine kurulmuştur. Oyuncular, sırayla zar atarak 3 puana ulaşmaya çalışır; ilk 3 puana ulaşan oyuncu oyunu kazanır.

Bu oyunun geliştirilmesinde öncelikli hedefler şunlardır:

- **Kullanıcı Deneyimi:** Oyuncuların kolayca oyun oynayabilmesi ve skorları takip edebilmesi için basit ve anlaşılır bir arayüz tasarlanmıştır.

- **Taşınabilirlik:** Oyun, farklı cihazlar ve tarayıcılarda sorunsuz çalışacak şekilde HTML, CSS ve JavaScript kullanılarak geliştirilmiştir.

- **Genişletilebilirlik:** İleride yeni özellikler eklenebilmesi için modüler bir yapı uygulanmıştır. Örneğin, farklı zar türleri veya skor hedefleri eklenebilir.

- **Performans:** Animasyonlar, skor hesaplamaları ve oyun mantığı minimum gecikme ile çalışacak şekilde optimize edilmiştir.

## 1.2 Tasarım Hedefleri

Tasarım hedefleri, projenin başarısını ve kullanıcı memnuniyetini doğrudan etkiler.

### 1.2.1 Kullanılabilirlik

Oyun, kullanıcıların karmaşık adımlar olmadan oynayabilmesi için basit bir arayüz sunar. Oyuncuların isim girişi yapması, zar atması ve skor takibi sezgisel bir şekilde yapılabilir. Animasyon ve ses efektleri, kullanıcılara anlık geri bildirim sağlar.

### 1.2.2 Taşınabilirlik

Oyunun tarayıcı tabanlı olması, farklı cihaz ve ekran boyutlarında çalışmasını sağlar. Responsive tasarım sayesinde hem mobil hem de masaüstü kullanıcıları için uyumluluk sağlanmıştır.

### 1.2.3 Genişletilebilirlik

Kod yapısı modüler ve esnek olacak şekilde tasarlanmıştır. Gelecekte yeni zar türleri eklenebilir, skor hedefleri değiştirilebilir veya çok oyunculu mod eklenebilir.

#### 1.2.4 Performans
Oyun, düşük donanım kapasitesine sahip cihazlarda bile hızlı ve akıcı çalışacak şekilde optimize edilmiştir. JavaScript ile oyun mantığı, animasyonlar ve skor hesaplamaları minimum gecikme ile uygulanmıştır.

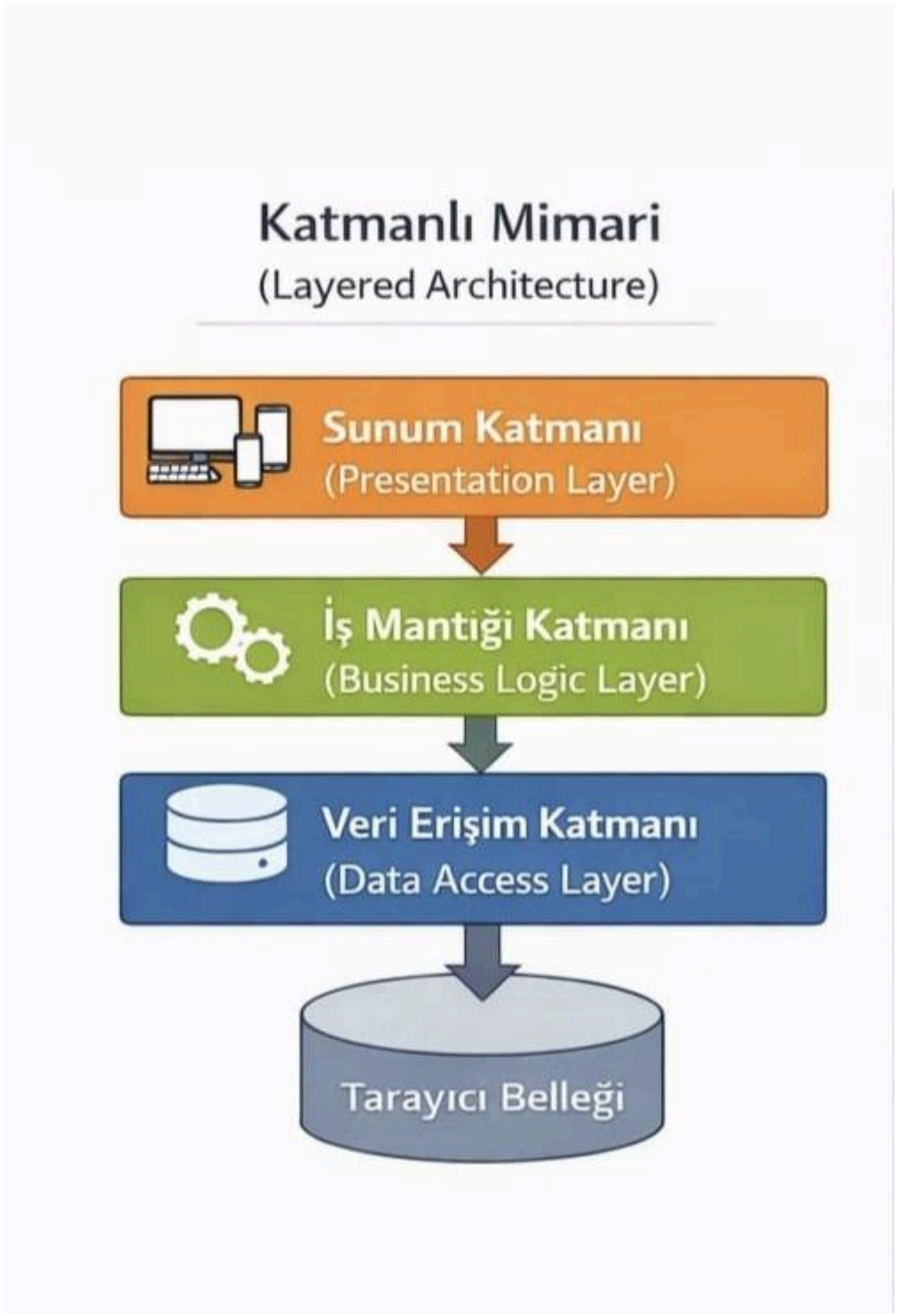
### 1.3 Ödünler ve Tasarım Kriterleri
Projenin tasarımında bazı ödünler ve kriterler belirlenmiştir:

- **Kullanılabilirlik vs Özellik Zenginliği:** Basit ve anlaşılır bir arayüz tercih edilmiş, gereksiz özellikler eklenmemiştir.

- **Geliştirme Süresi vs Genişletilebilirlik:** Proje kısa sürede tamamlanacak şekilde tasarlanmış, ancak kod yapısı ileride genişletilmeye uygun hâle getirilmiştir.

## 2. Sistem Mimarisi

### 2.1 Alt Sistem Ayrıştırma
Sistem, farklı sorumlulukları olan üç ana alt sisteme ayrılmıştır. Bu ayrıştırma, hem geliştirme sürecini kolaylaştırır hem de ileride sistemin genişletilmesini sağlar.




Şekil 2.1: Zar Oyunu Katmanlı Mimari Diyagramı

### 2.1.1 Kullanıcı Arayüzü (UI) Alt Sistemi

Kullanıcı arayüzü, oyuncuların oyunla doğrudan etkileşim kurduğu katmandır. Bu alt sistem HTML ve CSS kullanılarak oluşturulmuş olup aşağıdaki bileşenleri içerir:

- Başlangıç ekranı: Oyuncu isimlerinin girildiği form ve “Başlat” butonu.
- Oyun tahtası: Zar görüntüleri, skor tabloları ve tur durumları.

- Kontrol paneli: Zar atma butonları ve oyunu yeniden başlatma butonu.

- Geri bildirim: Zar animasyonları ve ses efektleri, kazanma popup'ları.

Bu yapı, kullanıcıların sezgisel bir deneyim yaşamasını sağlarken, UI değişiklikleri yapılırken diğer sistemleri etkilemez.

### 2.1.2 İş Mantığı (Business Logic) Alt Sistemi

İş mantığı alt sistemi, oyunun kurallarını ve akışını yönetir. JavaScript kullanılarak geliştirilmiş olup görevleri şunlardır:

- Zar atma işlemlerini yönetmek ve rastgele değer üretmek.

- Tur sırasını belirlemek ve oyuncuların hamlelerini kontrol etmek.

- Skor hesaplamak ve kazananı belirlemek.

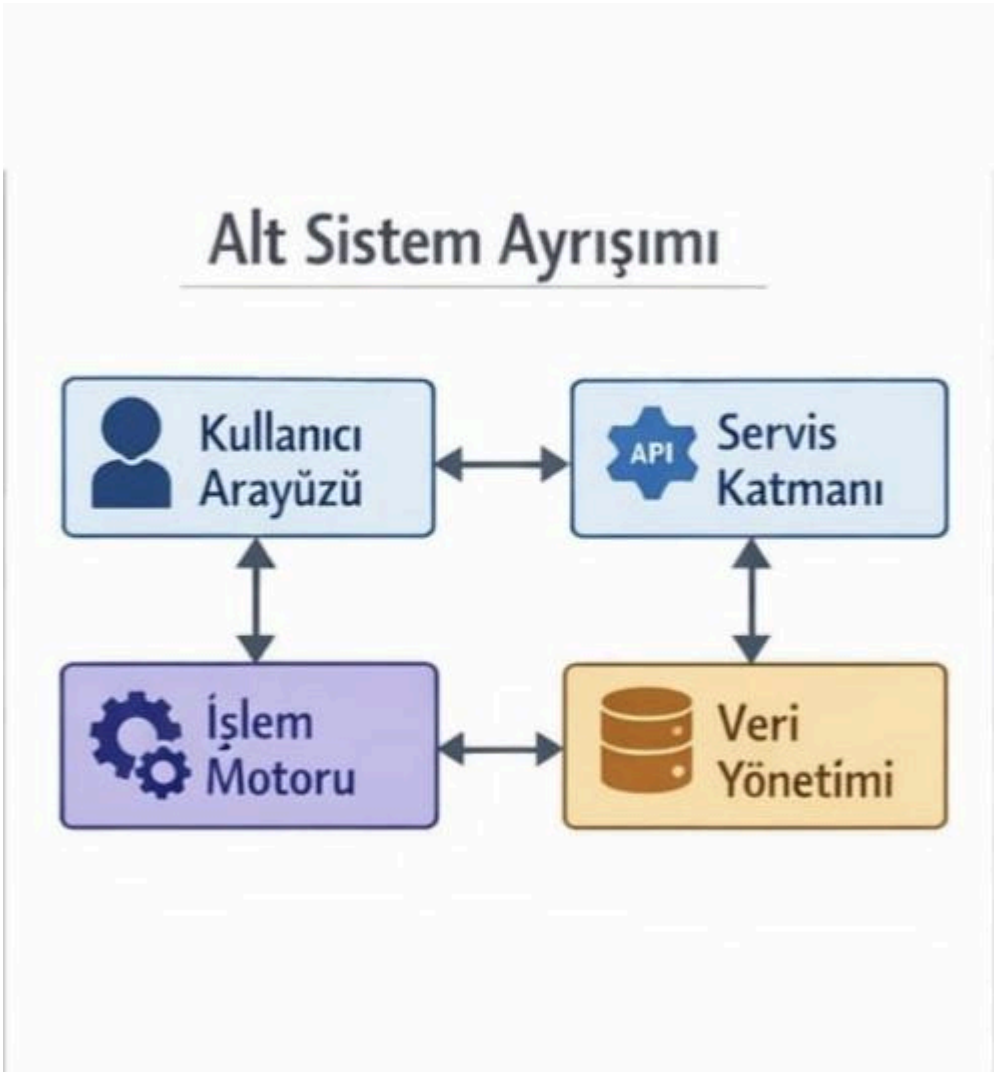
- Durum mesajlarını güncellemek.

İş mantığı UI'den bağımsız olduğundan, ileride yeni kurallar veya oyun modları eklemek kolaydır.

### 2.1.3 Veri Yönetimi Alt Sistemi

Veri yönetimi, oyuncu isimleri ve skorları geçici olarak saklar. Tarayıcı belleği kullanılarak veri tutulur, bu sayede oyunun performansı yüksek ve basit kalır.

Kalıcı veri saklama veya sunucu tabanlı depolama bu projede kullanılmamıştır; ancak sistem, ileride bu özellikler eklenmeye uygun şekilde tasarlanmıştır.



Şekil 2.2: Sistem Alt Sistem Ayrışımı ve Etkileşimi

## 2.2 Donanım/Yazılım Eşleme

Oyun tamamen istemci tarafında çalışacak şekilde tasarlanmıştır. Donanım ve yazılım eşlemesi aşağıdaki gibidir:

### 2.2.1 Sunucu Tarafı

Bu projede sunucu tarafı kullanılmamaktadır; tüm işlemler kullanıcının tarayıcısında gerçekleşir. Bu sayede geliştirme süresi kısalır ve oyun çevrimdışı olarak da çalışabilir.

### 2.2.2 İstemci Tarafı

Tüm UI, iş mantığı ve veri yönetimi istemci tarafında çalışır. Bu yapı sayesinde farklı cihazlar ve tarayıcılarda uyumluluk sağlanır.

## 2.3 Kalıcı Veri Yönetimi

Oyuncu skorları ve isimleri geçici olarak tarayıcı belleğinde tutulur. Proje kapsamında veritabanı kullanılmamıştır; ancak veri yönetimi modüler tasarlanmıştır. Gelecekte skorları kaydetmek veya oyuncu geçmişini takip etmek için veri tabanı eklemek mümkündür.

## 2.4 Erişim Kontrolü ve Güvenlik
Oyun, tek kullanıcı tabanlı ve tarayıcıda çalıştığı için karmaşık güvenlik mekanizmalarına ihtiyaç duymaz. Ancak temel güvenlik önlemleri aşağıda belirtilmiştir:

- Yanlış veya eksik isim girişleri varsayılan isimlerle değiştirilir.
- Kullanıcılar sadece kendi hamlelerini yapabilir; sıra dışı işlemler devre dışı bırakılır.
- Tarayıcı dışında bir veri akışı olmadığı için veri güvenliği büyük ölçüde sağlanmıştır.

## 2.5 Sınır Koşulları
Sistem, aşağıdaki sınır koşullarına göre tasarlanmıştır:

### 2.5.1 Başlatma Koşulları
- Oyuncu isimleri girilmeli veya varsayılan isimler kullanılmalıdır.
- Başlat butonuna basıldığında oyun alanı görünür hâle gelir.

### 2.5.2 Hata Yönetimi
- Oyuncuların zar atma sırası doğru şekilde kontrol edilir.
- Beklenmeyen durumlarda (ör. bir oyuncunun iki kez zar atması) hata mesajı gösterilir ve oyun durumu düzeltilir.

### 2.5.3 Kapanış Koşulları
- Bir oyuncu 3 puana ulaştığında kazanan belirlenir.
- Kazanan popup ile gösterilir ve oyun durur.
- Oyuncu “Yeniden Başlat” butonuna basarak oyunu sıfırlayabilir.

# 3. Alt Sistem Hizmetleri

Sistem, farklı alt sistemler aracılığıyla kullanıcıya hizmet sunar. Bu alt sistemler, oyunun işleyişi ve kullanıcı etkileşimini sağlar. Ana alt sistemler:

- Kullanıcı Arayüzü (UI) Alt Sistemi
- İş Mantığı (Yönetim) Alt Sistemi
- Veri (Model) Alt Sistemi
-

## 3.1 Kullanıcı Arayüzü Alt Sistemi Hizmetleri
Kullanıcı arayüzü alt sistemi, oyuncuların oyunu kullanabilmesini sağlayan tüm görsel ve etkileşimli bileşenleri içerir. Bu alt sistem aşağıdaki hizmetleri sunar:

### 3.1.1 Giriş Ekranı
- Oyuncuların isimlerini girmesi sağlanır.
- Varsayılan isimler atanabilir.
- “Başlat” butonuna basılması ile oyun başlatılır.

### 3.1.2 Oyun Alanı ve Skor Görüntüleme
- Her oyuncu için zar ve skor alanı gösterilir.
- Oyuncu sırası ve tur durumu ekranda anlık olarak güncellenir.

### 3.1.3 Zar Atma ve Animasyonlar
- Zar butonları ile oyuncular sırayla zar atabilir.
- Zar atıldığında animasyon ve ses efektleri oynatılır.
- Hamleler görsel olarak geri bildirim ile sunulur.

### 3.1.4 Durum ve Hata Mesajları
- Tur sonunda kazanan veya beraberlik durumları bildirilir.
- Yanlış işlemler veya sıra hataları kullanıcıya mesaj ile gösterilir.

## 3.2 İş Mantığı (Yönetim) Alt Sistemi Hizmetleri
İş mantığı alt sistemi, oyunun kurallarını ve akışını yöneten ana bileşendir. Bu alt sistem aşağıdaki hizmetleri sunar:

### 3.2.1 Tur ve Puan Yönetimi
- Oyuncuların hamle sırası belirlenir.
- Zar değerleri karşılaştırılır ve puanlar güncellenir.
- İlk 3 puana ulaşan oyuncu kazanır.

### 3.2.2 Oyun Durumu Kontrolü
- Tur sonunda skorları ve oyunun durumunu kontrol eder.
- Kazanan belirlenir ve gerekli popup mesajı tetiklenir.
- Beraberlik veya sıra hatası gibi durumları yönetir.

### 3.2.3 Yeniden Başlatma
- Oyuncular oyunu yeniden başlatabilir.
- Sistem tüm skorları ve oyun durumunu sıfırlar.

## 3.3 Veri (Model) Alt Sistemi Hizmetleri
Veri alt sistemi, oyun sırasında gerekli tüm bilgileri geçici olarak saklar. Bu alt sistem aşağıdaki hizmetleri sunar:

### 3.3.1 Skor ve Oyuncu Bilgileri
- Oyuncu isimleri ve puanlar geçici olarak tarayıcı belleğinde tutulur.
- Oyun sırasında hızlı ve güvenli erişim sağlar.

### 3.3.2 Oyun Geçmişi (Opsiyonel)
- İleride oyun geçmişi veya skor tablosu saklanacak şekilde tasarlanabilir.
- Şimdilik basit yapıda veri geçici olarak tutulur.

### 3.3.3 Veri Bütünlüğü ve Güvenlik
Sadece mevcut tarayıcı oturumu boyunca veri geçerlidir.

Yanlış veya eksik veri girişleri kontrol edilir ve sistem otomatik düzeltme sağlar.

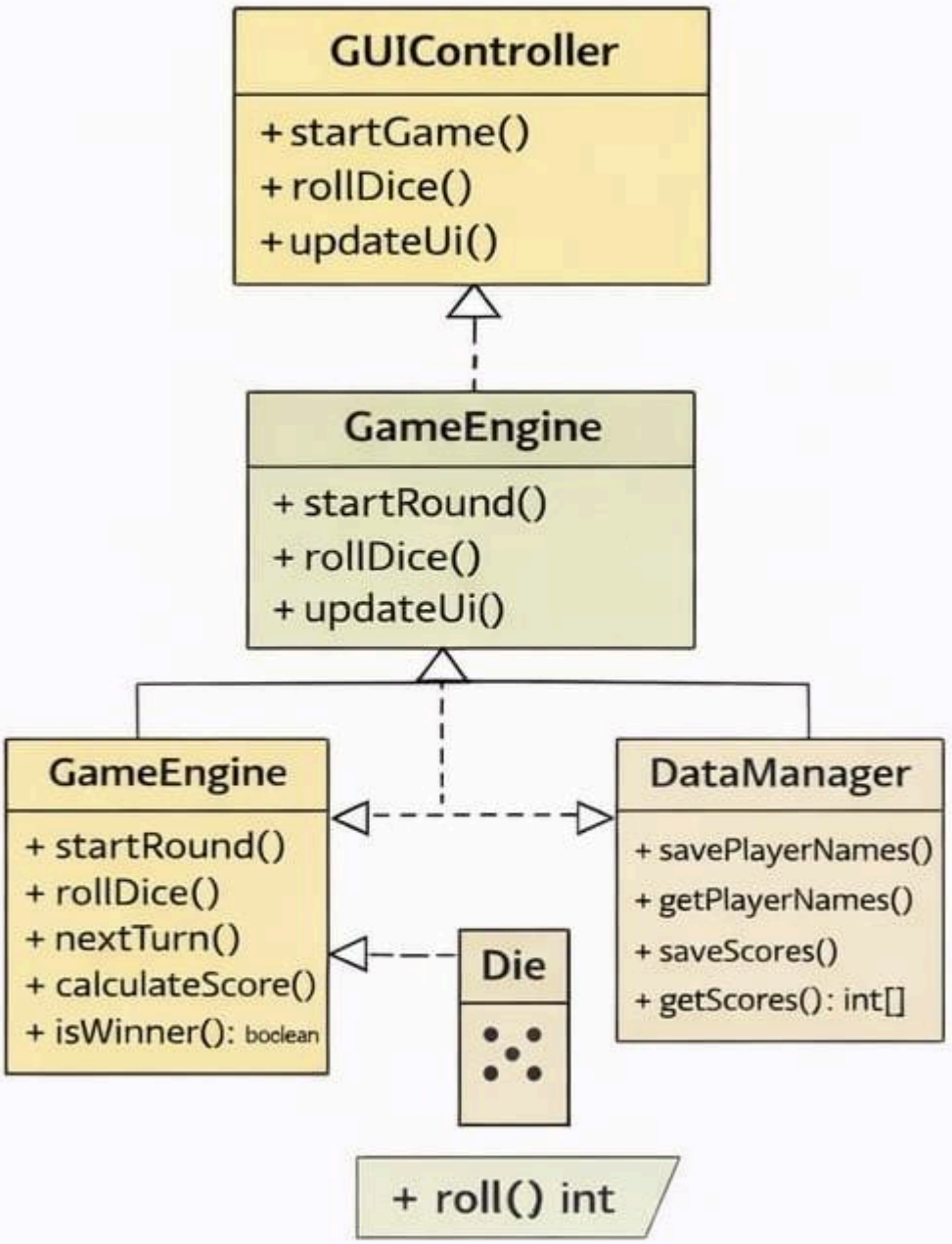
# 4. Düşük Seviyeli Tasarım

Bu bölümde oyunun iç yapısı, sınıf tasarımları ve nesne ilişkileri detaylandırılmaktadır. Zar oyunu, temel olarak kullanıcı arayüzü, oyun mantığı ve veri yönetimi bileşenlerinden oluşur. Nesne yönelimli yaklaşımla her bileşenin sorumlulukları belirlenmiştir.

## 4.1 Sınıf Tasarımı

Oyun üç ana sınıf ve birkaç yardımcı sınıf üzerinden yönetilir:

# Nihai Sınıf Tasarımı (UML)



Şekil 4.1: Zar Oyunu Nihai Sınıf Diyagramı

### 4.1.1 Player (Oyuncu) Sınıfı

- **Sorumluluk**: Oyuncu bilgilerini ve puan durumunu tutar.
- **Özellikler**:
  **name**: Oyuncunun adı
  **score**: Oyuncunun mevcut puanı
- **Metotlar**:
  **rollDice()**: 1-6 arasında rastgele zar değeri üretir
  **addScore():** Tur kazanınca puanı 1 artırır
  **resetScore():** Oyun yeniden başlatıldığında puanı sıfırlar

Bu sınıf oyunun temel veri birimidir ve kullanıcı arayüzü ile iş mantığı arasında köprü görevi görür.

### 4.1.2 Game (Oyun) Sınıfı

- **Sorumluluk**: Oyun akışını ve kuralları yönetir.
- **Özellikler**:
  **player1 ve player2**: Oyuncu nesneleri
  **currentTurn**: Oyuncu sırasını takip eder
  **winTarget**: Kazanmak için gereken puan (ör. 3)
- **Metotlar**:
  **startGame():** Oyunu başlatır
  **roll(player):** Seçilen oyuncu için zar atma işlemini yönetir
  **checkRoundWinner():** İki oyuncu zar attıktan sonra tur kazananını belirler
  **checkGameWinner():** Oyunculardan biri kazanmışsa oyunu sonlandırır
  **resetGame():** Oyun sıfırlanır

Bu sınıf oyun mantığının merkezi olup tüm oyun akışını kontrol eder.

### 4.1.3 Dice (Zar) Sınıfı

- **Sorumluluk**: Zar nesnesi ve rastgele değer üretimini kapsar.
- **Özellikler**:
  **value**: Zarın mevcut değeri
- **Metotlar**:
  **roll():** Zarın yeni değerini üretir
  **getValue():** Zar değerini döndürür

Dice sınıfı, hem oyun mantığını hem de kullanıcı arayüzünü besleyen bağımsız bir nesnedir.

### 4.1.4 Notification (Bildirim) Sınıfı

- **Sorumluluk**: Kullanıcıya oyun durumunu bildirir (tur sonucu, kazanan vs.).
- **Metotlar**:
  **showMessage(message):** Durum mesajını ekrana yazar
  **playSound(sound):** Oyun sırasında ses efekti oynatır

Bu sınıf, UI ile iş mantığı arasındaki geri bildirim mekanizmasını sağlar.

## 4.2 Tasarım Kararları ve Tasarım Desenleri

#### 4.2.1 Singleton Deseni (Game Sınıfı İçin)
- Oyun yalnızca tek bir Game nesnesi üzerinden yönetilir.
- Bu sayede oyun durumu merkezi olarak kontrol edilir ve tutarsızlıklar engellenir.

#### 4.2.2 Factory Deseni (Dice Sınıfı İçin)
- Zar nesneleri dinamik olarak oluşturulur.
- İleride farklı türde zar eklenmek istenirse kolaylık sağlar (ör. 10 yüzlü zar).

#### 4.2.3 Observer Deseni (Notification Sınıfı İçin)
- Oyun durumu değiştiğinde kullanıcı arayüzüne anlık bildirim gönderir.
- Tur bitişi, kazanan belirlenmesi ve hata mesajları bu mekanizma ile iletilir.

### 4.3 Paket Yapısı
Oyunun kodları, anlaşılır ve modüler bir şekilde paketlenmiştir:

- **ui:** Kullanıcı arayüzü sınıfları ve DOM yönetimi (Notification)
- **core**: Oyun mantığı ve ana sınıflar (Game, Player, Dice)
- **model**: Veri saklama ve skor yönetimi
- **sound**: Ses efektleri için yardımcı sınıflar

Bu paket yapısı, kodun okunabilirliğini ve bakımını kolaylaştırır.

### 4.4 Sınıf Arayüzleri
Her ana sınıfın bir arayüzü vardır. Bu sayede sınıflar arası bağımlılık azaltılmıştır:

- **PlayerInterface**: Oyuncu işlemleri (puan ekleme, zar atma)
- **GameInterface**: Oyun akışı metotları (başlat, tur kontrol, kazanma kontrolü)
- **DiceInterface**: Zar işlemleri
- **NotificationInterface**: Bildirim ve ses yönetimi

## 5. Sonuç ve Değerlendirme

Bu rapor kapsamında geliştirilen zar oyunu projesi, basit ama etkili bir iki oyunculu oyun sistemi olarak tasarlanmıştır. Oyuncular sırayla zar atar ve 3 puana ilk ulaşan oyuncu oyunu kazanır. Proje, hem kullanıcı dostu bir arayüz sunmakta hem de oyun mantığını sağlam ve yönetilebilir bir şekilde gerçekleştirmektedir.

### 5.1 Elde Edilen Başarılar
**Kullanıcı Arayüzü:**
- HTML ve CSS kullanılarak temiz, anlaşılır ve estetik bir arayüz tasarlandı.
- Oyuncu isimlerini girebilecekleri bir ekran ve oyun sırasında skorları görüntüleyen bir oyun alanı sağlandı.
- Pop-up bildirimler ve animasyonlar ile kullanıcı deneyimi zenginleştirildi.

**Oyun Mantığı ve İş Akışı:**

- JavaScript ile oyun akışı yönetildi.
- Oyuncuların zar atma sırası ve puan güncellemeleri doğru şekilde kontrol edildi.
- Tur kazananı ve oyun kazananı hesaplanarak kullanıcıya bildirildi.
- Tekrar oynama ve sıfırlama seçenekleri eklendi.

**Nesne Yönelimli Tasarım:**

- Oyuncu (Player), Oyun (Game), Zar (Dice) ve Bildirim (Notification) sınıfları oluşturuldu.
- Tasarım desenleri kullanılarak sistemin genişletilebilir ve bakımı kolay hale getirildi.
- Singleton, Factory ve Observer desenleri ile sınıflar arasındaki ilişkiler ve sorumluluklar netleştirildi.

**Performans ve Kullanılabilirlik:**

- Oyun hızlı çalışmakta ve kullanıcı etkileşimlerine anında yanıt vermektedir.
- Tarayıcı tabanlı olması sayesinde herhangi bir ek yazılım kurulumu gerektirmemektedir.

## 5.2 Karşılaşılan Zorluklar ve Çözümler

- **Tur sırasının yönetimi**: Başlangıçta oyuncuların aynı anda zar atması veya yanlış sıralama problemi yaşandı.
  **Çözüm**: Roll butonları aktif/pasif duruma getirilerek sıra kontrolü sağlandı.
- **Animasyon ve ses senkronizasyonu**: Zar animasyonu ve ses efektlerinin uyumsuzluğu gözlemlendi.
  **Çözüm**: setTimeout kullanılarak animasyon tamamlandıktan sonra ses oynatıldı.
- **Oyun sonu durumunun yönetimi**: Oyuncu kazanır kazanmaz oyun durumu net bir şekilde gösterilmeliydi.
  **Çözüm**: Pop-up ve overlay sistemi ile kazanan oyuncu öne çıkarıldı ve tüm roll butonları devre dışı bırakıldı.

## 5.3 Projenin Genişletilebilirliği

Bu oyun, ileride ek özelliklerle geliştirilebilir:

- **Farklı oyun modları:** Örn. 5 puana ulaşma, tek zar yerine iki zar kullanma.
- **Çevrimiçi çok oyuncu desteği**: WebSocket veya benzeri teknolojilerle iki oyuncu farklı cihazlardan oynayabilir.
- **Gelişmiş skor tablosu ve istatistikler**: Kullanıcıların geçmiş oyun performansları kaydedilebilir.

Bu tasarım sayesinde yeni özellikler eklemek kolaydır; sınıflar ve modüler yapı, değişiklikleri sistemin diğer bölümlerine zarar vermeden uygulamayı sağlar.

## 5.4 Genel Değerlendirme

Bu proje, öğrencilik seviyesinde bir web tabanlı oyun olarak:

- Kullanıcı dostu bir arayüz,
- Net bir oyun mantığı ve akışı,

- Bakımı ve genişletmesi kolay bir nesne yönelimli yapı

sunmaktadır.

Tasarım ve geliştirme süreci boyunca uygulanabilirlik, performans ve kullanılabilirlik ön planda tutulmuş, aynı zamanda oyun deneyimi eğlenceli ve anlaşılır hale getirilmiştir. Sonuç olarak, proje hem eğitim hem de uygulama amacıyla başarılı bir örnek teşkil etmektedir.