

**T.C**

**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ**

**MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ**

**BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ**

**PROJE KONUSU: Oyun Projesi**

**Emrah ŞAHİN Muhammed Yusuf KAYA 220502025 210501007**

**DERS SORUMLUSU:**

**PROF. DR. Hüseyin Tarık DURU**

**TARİH:24.03.2024**

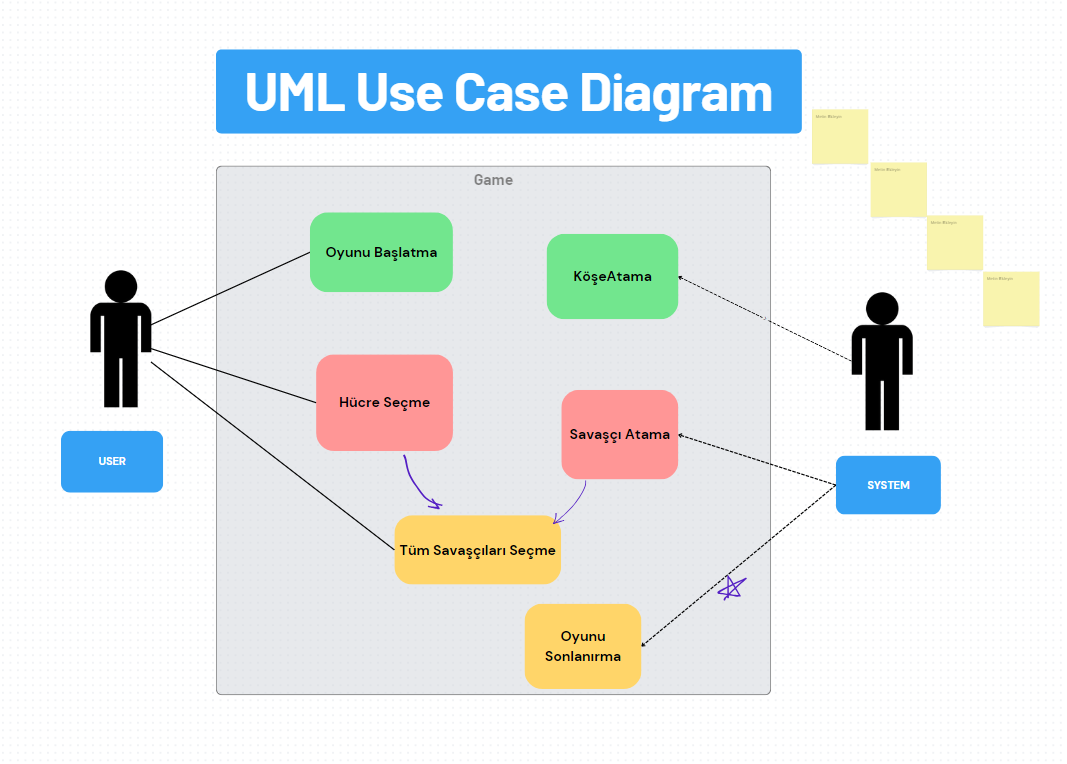
1. **GİRİŞ**
   1. **Projenin amacı**

* Oyunun amacı, stratejik düşünme ve rakipleri yenmek için doğru savaşçıları doğru konumlara yerleştirme üzerine kuruludur. Kullanıcının belirlediği oyuncu sayısıyla oynanan bir çeşit stratejik oyundur. Oyunun başlangıcında otomatik olarak her köşeye rastgele oyuncular dağıtılır. O ilk turu ifade eder ve oyun tur sistemine dayalıdır.
* Kod, PyQt5 kullanarak bir grafik arayüzü oluşturur ve kullanıcıya oyunu oynamak için bir arayüz sağlar. Oyuncular arasında sırayla geçiş yapmayı ve savaşçıları ızgaraya yerleştirmeyi sağlayan bir mantık bulunmaktadır.
* Bu projenin amacı, savaş stratejisi ve oyuncular arası rekabeti simüle etmektir. Kullanıcılar farklı stratejiler deneyebilir ve rakiplerini yenmek için en iyi taktikleri geliştirmeye çalışabilirler. Oyuncular kendini alanlarını genişletmeye çalışır. Kendi alanlarını genişleterek birbirlerinin alanlarına sahip olmaya çalışır.

1. **GEREKSİNİM ANALİZİ**
   1. **Arayüz gereksinimleri**

* **Ana Pencere (MainInterface):**
  + Kullanıcı, ana pencerede oyunu başlatmak için başlatma butonuna sahip olmalıdır.
  + Kullanıcı, oyuncu sayısını belirlemek için seçenekler sunan radyo düğmelerine sahip olmalıdır (2, 3 veya 4 oyuncu).
  + Kullanıcı, ızgara boyutunu belirlemek için bir metin giriş kutusuna sahip olmalıdır. Izgara boyutu 8x8 ile 32x32 arasında olabilir.
  + Başlatma butonuna tıklandığında, ızgara penceresi açılmalıdır.
* **Izgara Penceresi (GridWindow):**
  + Kullanıcılar, ızgara üzerinde savaşçılarını yerleştirmek için hücrelere tıklayabilmelidir.
  + Kullanıcılar, savaşçı seçimini yapmak için bir iletişim kutusu penceresi almalıdır. Bu iletişim kutusu, muhafız, okçu, nişancı, süvari veya şifacı gibi seçenekler sunmalıdır.
  + Her oyuncu, belirli bir renkte temsil edilmelidir. Bu renk, oyuncunun savaşçılarının ızgara üzerindeki konumlarını göstermek için ve diğer oyunculardan fark edilmek için kullanılmalıdır.
  + Oyunun ilerleyişi, ızgara üzerindeki savaşçıların yerleştirilmesiyle görselleştirilmelidir.
  + Oyuncular, sırayla savaşçılarını ızgara üzerine yerleştirmelidir. Her turda, her oyuncunun belirli bir savaşçı sayısı olabilir.
* **Karakter Seçimi İletişim Kutusu (CharacterSelectionDialog):**
  + Kullanıcı, savaşçı seçimini yapmak için bir iletişim kutusu penceresi almalıdır.
  + İletişim kutusu penceresi, kullanıcıya muhafız, okçu, nişancı, süvari veya şifacı gibi seçenekler sunmalıdır.
  + Kullanıcı, seçimini yapmak için bir "Seç" düğmesine sahip olmalıdır.
* **Kullanıcı Geri Bildirimleri:**
  + Kullanıcılar, her eylemlerinde uygun geri bildirimler almalıdır. Örneğin, savaşçı seçimi yanlış yapıldığında bir geri bildirim mesajı görüntülenmelidir.
  + Kullanıcılar, oyunun ilerleyişi hakkında bilgilendirilmelidir. Örneğin, hangi oyuncunun sırası olduğu sadece görsel olarak belli olmalıdır
  1. **Fonksiyonel gereksinimler**
* **Oyuncu Sayısının Belirlenmesi:**
  + Kullanıcı, başlangıç ekranında oyunu başlatmak için kaç oyuncu ile oynamak istediğini belirler.
  + Seçenekler arasında 2, 3 veya 4 oyuncu seçeneği bulunur.
  + En az 2 oyuncu ve en fazla 4 oyuncu ile oyun oynanabilir.
* **Izgara Boyutunun Belirlenmesi:**
  + Kullanıcı, başlangıç ekranında oyun alanının boyutunu seçer.
  + Kullanıcı, bir metin giriş kutusu aracılığıyla oyun alanının boyutunu belirler.
  + İzin verilen boyutlar 8x8 ile 32x32 arasındadır.
* **Savaşçı Seçimi:**
  + Kullanıcılar, karakter seçimi yapmak için ilgili butonlara tıklayarak veya ızgara üzerindeki hücrelere tıklayarak savaşçılarını yerleştirebilirler.
  + Seçilebilecek savaşçı türleri: Muhafız, Okçu, Nişancı, Süvari, Şifacı.
  + Her oyuncu, oyunun başında belirli bir sayıda savaşçı seçebilir.
* **Savaşçı Yerleştirme:**
  + Oyuncular, savaşçılarını ızgara üzerinde belirli koordinatlara yerleştirebilirler.
  + Her turda, oyuncular sırayla savaşçılarını ızgara üzerinde belirli hücrelere yerleştirebilirler.
  + Oyuncular, ızgara üzerinde belirli bir hücreyi seçerek savaşçılarını yerleştirebilirler.
* **Savaşçıların Özellikleri:**
  + Muhafız, Okçu, Nişancı, Süvari ve Şifacı gibi farklı savaşçı türlerinin özellikleri ve yetenekleri bulunmaktadır.
  + Her savaşçı türünün farklı can, hasar, menzil ve saldırı özellikleri vardır.
  + Oyuncular, bu özellikler doğrultusunda stratejik kararlar alarak savaşçılarını yerleştirirler.
* **Oyunun İlerlemesi:**
  + Oyun, oyuncuların savaşçılarını ızgara üzerine yerleştirmesiyle başlar.
  + Oyuncular sırayla savaşçılarını yerleştirir ve her turda bir savaşçı yerleştirilir.
  + Savaşçılar, belirli hedeflere saldırır ve belirli menzildeki düşmanları hedef alır.
  + Oyun, tüm savaşçılar yerleştirildikten sonra devam eder ve savaşçılar arasındaki mücadele sürer.
* **Oyunun Sonlanması:**
  + Oyun, belirli bir koşul gerçekleşene kadar devam eder.
  + Oyuncular, savaşçılarını stratejik bir şekilde yerleştirerek rakiplerine karşı üstünlük sağlamaya çalışırlar.
  + Oyun, belirlenen koşullardan biri gerçekleştiğinde sona erer ve bir kazanan ilan edilir.
  1. **Use-Case diyagramı**





1. **TASARIM**
   1. **Mimari tasarım**

Oyunun mimari tasarımı, PyQt5 kütüphanesi kullanılarak oluşturulmuştur. Ana bileşenler, MainInterface ve GridWindow sınıflarıdır.

* **MainInterface**: Ana pencereyi oluşturan sınıftır. Bu pencere, oyuncuların oyunu başlatmasını ve temel ayarlarını yapmasını sağlar. Ana pencere, oyuncu sayısını ve ızgara boyutunu seçme seçenekleri sunar. Oyun başlatıldığında, bu sınıf, GridWindow sınıfını başlatarak oyunun ızgara penceresini görüntüler.
* **GridWindow**: Izgara penceresini oluşturan sınıftır. Bu pencere, oyuncuların savaşçılarını ızgara üzerine yerleştirmesini sağlar ve oyunun ilerleyişini görselleştirir. Izgara penceresi, ızgara şeklindeki bir alanı temsil eder ve her hücre bir düğmeyle temsil edilir. Oyuncular, savaşçılarını bu hücrelere yerleştirebilirler.
* Oyun sınıfı , savasci sınıflarını yönetir ve oyunun temel kurallarını içerir. Ancak, bu sınıfın arayüzle etkileşimi yoktur. Arayüz, oyuncuların oyunu oynamasını sağlayan bir ara yüzdür.
* Ayrıca, CharacterSelectionDialog sınıfı, kullanıcıların savaşçılarını seçmek için bir iletişim kutusu penceresi sağlar. Bu iletişim kutusu penceresi, ana pencereden ayrı bir iletişim aracı olarak kullanılır.
  1. **Kullanılacak teknolojiler**
* **Yazılım Dili:**
* Yazılım, Python programlama dilinde geliştirilmiştir. Python, basit ve okunabilir sözdizimiyle bilinen yüksek seviyeli bir programlama dilidir. PyQt5 gibi arayüz oluşturma, random gibi kütüphaneleri ile birlikte kullanılabilmesi ve geniş kütüphane desteği sayesinde tercih edilmiştir.

### Arayüz Kütüphanesi:

* PyQt5 kütüphanesi, Python ile Qt arayüzünü kullanmayı sağlayan bir arayüz kütüphanesidir. Bu kütüphane, masaüstü uygulamalarının hızlı bir şekilde geliştirilmesini sağlar. PyQt5'in zengin özellik seti ve geniş dokümantasyonu, karmaşık kullanıcı arayüzleri oluşturmak için idealdir.

### Harici Kütüphaneler:

* Proje geliştirilirken random, sys, PyQt5 kütüphanesi kullanılmıştır. PyQt5, kullanıcı arayüzü oluşturmanın yanı sıra çeşitli özellikler sunar, bu nedenle ek harici kütüphanelere ihtiyaç duyulmamıştır. Ancak, geliştirme sürecinde PyQt5'in yanı sıra standart kütüphaneler olan **random**, **sys**, **QColor**, **pyqtSignal**, **QApplication**, **QMainWindow**, **QWidget**, **QGridLayout**, **QPushButton**, **QLabel**, **QDialog**, **QVBoxLayout**, **QRadioButton**, **QMessageBox**, **QLineEdit**, **QAction**, **QHBoxLayout** gibi alt kütüphaneler kullanılmıştır.
* Random kütüphanesi de programcının belirli bir isteğini rastgele yapma özelliğini verir. Bir zar atmak gibi her ihtimalden biri olabilir.
* Harici bir veri tabanı kullanılmamıştır.
  1. **Veri tabanı tasarımı**
* Veritabanı kullanılmadığı için ER diyagramı bulunmamaktadır.
  1. **Kullanıcı arayüzü tasarımı**
* **Ana pencere (MainInterface)**:
  + Kullanıcı, başlangıç ekranında oyunu başlatmak için kaç oyuncu ile oynamak istediğini seçebilir. Bu seçenekler radyo düğmeleri şeklinde sunulmuştur.
  + Ayrıca, kullanıcı oyun alanının boyutunu belirleyebilir. Boyut girişi için bir metin kutusu mevcuttur.
  + Başlatma ekranında "Oyunu Başlat" adında bir buton bulunmaktadır. Bu butona tıklandığında oyun başlatılır.
* **Izgara penceresi (GridWindow)**:
  + Oyuncular, savaşçılarını ızgara üzerine yerleştirebilirler. Bunun için ızgara üzerindeki hücrelere tıklama işlemi kullanılır.
  + Savaşçı seçimi için bir iletişim kutusu (CharacterSelectionDialog) görüntülenir. Kullanıcı, bu iletişim kutusunda farklı savaşçı türlerini seçebilir.
  + Seçilen savaşçılar, ızgara üzerinde görsel olarak sembollerle temsil edilir.
  + Her turda, oyuncular sırayla savaşçılarını ızgara üzerine yerleştirirler. Bu işlem, kullanıcıların ızgara üzerindeki hücreleri seçmesiyle gerçekleşir.
* **Yazılımın nasıl çalıştırılacağı**:
  + Kullanıcı, ana pencerede oyun için gerekli ayarları yapar ve "Oyunu Başlat" butonuna tıklar.
  + Ardından, ızgara penceresi açılır ve oyuncular savaşçılarını ızgara üzerine yerleştirmeye başlar.
  + Oyuncular, savaşçı seçimi yapmak için ilgili butonlara tıklar ve ardından ızgara üzerindeki bir hücreyi seçer.
  + Savaşçılar, ızgara üzerinde sembollerle temsil edilir ve oyuncuların savaşçılarını yerleştirmesiyle oyun ilerler.
* **Ekran çıktıları**:
  + Ana pencerede, kullanıcı arayüzü öğeleri (radyo düğmeleri, metin kutuları, butonlar) görüntülenir.
  + Izgara penceresinde, ızgara üzerindeki hücreler ve savaşçılar görsel olarak temsil edilir.
  + Kullanıcı, ızgara üzerindeki hücreleri seçerek savaşçılarını yerleştirir ve oyunun ilerleyişini izler.

1. **UYGULAMA**
   1. **Kodlanan bileşenlerin açıklamaları**

* MainInterface: Bu sınıf, PyQt5 kütüphanesi kullanılarak oluşturulan ana pencereyi temsil eder. Kullanıcının oyunu başlatması ve temel ayarları yapması için arayüz sağlar. İçeriği şu şekildedir:
* initUI(): Ana pencerenin başlık ve düzenini belirler.
* startGame(): Kullanıcının belirlediği oyuncu sayısı ve ızgara boyutu doğrultusunda oyun penceresini başlatır.
* GridWindow: Bu sınıf, oyuncuların savaşçılarını ızgara üzerine yerleştirmesini sağlayan ana oyun penceresini temsil eder. İçeriği şu şekildedir:
* initUI(): Izgara penceresinin başlık, boyut ve düzenini belirler.
* createGrid(): Izgara üzerindeki hücreleri oluşturur ve kullanıcıların savaşçıları yerleştirmesini sağlar.
* startGame(): Oyuncuların sırayla savaşçılarını yerleştirmesini sağlar.
* assignWarrior(): Kullanıcının seçtiği hücreye savaşçı ataması yapar.
* selectCell(): Kullanıcının ızgara üzerinde hücre seçmesini sağlar ve savaşçı seçimini yapar.
* updateGrid(): Oyun alanındaki değişiklikleri günceller ve kullanıcı arayüzünü yeniler.
* CharacterSelectionDialog: Bu sınıf, kullanıcının savaşçı seçimini yapması için iletişim kutusu penceresini temsil eder. İçeriği şu şekildedir:
* selectCharacter(): Kullanıcının seçtiği savaşçı tipini belirler ve iletişim kutusunu kapatır.
* Savasci: Bu sınıf, farklı savaşçı türlerini temsil eden temel sınıftır. Tüm savaşçılar için ortak özellikleri ve davranışları tanımlar.
  1. **Görev dağılımı**
* Arayüz Tasarımı ve Geliştirme:
* Ana pencere (MainInterface) ve ızgara penceresi (GridWindow) arayüzlerinin tasarımı ve geliştirilmesi Emrah Şahin’e aitti. PyQt5 kütüphanesi kullanılarak kullanıcı arayüzü oluşturuldu. İlgili butonlar, giriş alanları ve iletişim kutuları bu aşamada oluşturuldu.
* Oyun Mantığı ve Sınıfların Geliştirilmesi:
* Savasci sınıfı ve onun alt sınıfları (Guard, Archery, Gunner, Horseman, Healer) Muhammed Yusuf Kaya’ya aitti. Her savaşçı türünün özelliklerini ve davranışlarını tanımlamak için gerekli sınıflar bu aşamada oluşturuldu.
* Karakter Seçim Penceresi:
* Kullanıcının savaşçı seçimini yapabileceği bir iletişim kutusu (CharacterSelectionDialog) geliştirildi. Bu iletişim kutusu, PyQt5 kullanılarak oluşturuldu ve ana arayüz ile etkileşim sağlamak üzere entegre edildi. Bu aşama Emrah Şahin’e aittir.
* Test ve Doğrulama:
* Yazılımın test edilmesi ve doğrulanması için bir test uygulaması geliştirildi. Bu uygulama, yazılımın bütün bileşenlerini kapsayacak şekilde tasarlandı ve PyQt5 arayüz bileşenlerini test etmeye imkan tanıdı. Bu aşama Muhammed Yusuf Kaya’ya aittir.
* Rapor yazımının büyük bir kısmını Muhammed Yusuf Kaya yapmıştır. Geriye kalan küçük düzenlemeler veya hata düzenlemelerini Emrah Şahin yapmıştır.
  1. **Karşılaşılan zorluklar ve çözüm yöntemleri**
* PyQt5 Kullanımı:

PyQt5 gibi karmaşık bir kütüphane kullanılırken arayüz tasarımında bazı zorluklarla karşılaşıldı. Programcı programı yazarken ızgaranın alanının hepsini buton olarak atama yaptı. Bu atama programcıya zorluklar yaşattı. Başka bir çözüm olarak buton atamak yerine ızgarayı daha farklı yapıp oyuncunun tıkladığı yerin konumunu alarak gidilebilirdi. Dokümantasyon ve çevrimiçi kaynaklar aracılığıyla bu zorlukların üstesinden gelindi. İhtiyaç duyulan özelliklerin nasıl uygulanacağı konusunda araştırmalar yapıldı ve denemeler gerçekleştirildi.

* Sınıf İlişkileri:

Sınıflar arasındaki doğru ilişkilerin kurulması ve uygun etkileşimlerin sağlanması önemliydi. Bu nedenle, sınıf yapıları ve metotlar dikkatlice tasarlandı. Ardından, test edilerek doğruluğu sağlandı. Oyun mantığı üzerinde dikkatlice düşünülerek, sınıfların birbirleriyle nasıl etkileşimde bulunacağı planlandı ve bu plana göre uygulama geliştirildi.

Ama tur sistemini tam olarak yapılamadığı için savaşçıları da sadece oyun alanına yerleştirdiği için bu sınıflar kullanılamadı. Projede sadece savaşçı yerleştirmesi vardır. Savaşçıların bir saldırı veya savunma özellikleri kullanılamamıştır.

* 1. **Proje isterlerine göre eksik yönler**
* Daha Fazla Oyun İçi Kontrol: Oyun sırasında daha fazla kullanıcı kontrolleri eklenerek, oyuncuların savaşçılarını yerleştirme süreci daha etkileşimli hale getirilebilir. Örneğin, oyuncuların savaşçılarını özel yeteneklerini kullanma gibi ek özellikler eklenerek stratejik derinlik artırılabilir. Oyuncunun o anki turunda oyun alanına birden fazla karakter yerleştirmesi, oyuncular sadece kendi alanlarından seçim yapması yani oyuncunun kendi alanının rengi neyse (kırmızı, mavi,…) sadece o alanda seçip yapması daha etkili olabilir. Ama bu projede herhangi bir oyuncu oyun alanının herhangi bir yerine savaşçısını koyabilir.
* Projede sadece savaşçıları belli bir kurala göre yerleştirilmesi vardır. Bu kurallarda da herhangi bir oyuncunun sınırlı kaynağı falan yoktur ve sadece her turda oyuncu bir savaşçı yerleştirebilir. Ama proje gereksinimlerine göre oyuncu kendi kaynağına göre ilk tur hariç diğer turlarda iki tane savaşçı yerleştirebilir.
* Görsel Geliştirmeler: Oyunun görsel unsurları daha da geliştirilebilir. Örneğin, daha çekici bir arayüz tasarımı, animasyonlar veya efektler eklenerek oyun deneyimi zenginleştirilebilir.
* Oyun Dengelemesi ve Hata Düzeltmeleri: Oyun dengelemesi için daha fazla test ve geri bildirim alınabilir. Ayrıca, yazılımda olası hataların düzeltilmesi ve performans iyileştirmeleri yapılabilir.
* Dokümantasyon Geliştirmeleri: Kod dökümantasyonu daha kapsamlı hale getirilebilir. Sınıflar, fonksiyonlar ve değişkenler için daha detaylı açıklamalar eklenerek, kodun anlaşılabilirliği artırılabilir.
* Kullanıcı Geri Bildirimi ve İyileştirmeleri: Kullanıcı geri bildirimleri toplanarak, kullanıcı deneyimini geliştirmek için uygun düzeltmeler ve iyileştirmeler yapılabilir. Kullanıcıların daha kolay anlayabileceği bir arayüz tasarımı veya yeni özellikler eklenerek kullanıcı memnuniyeti artırılabilir.

1. **TEST VE DOĞRULAMA**
   1. **Yazılımın test süreci**

Yazılımın test süreci, bir dizi test senaryosu kullanılarak gerçekleştirilmiştir. Her bir test senaryosu, belirli bir bileşenin veya işlevin doğru bir şekilde çalışıp çalışmadığını kontrol etmek için tasarlanmıştır. Test süreci aşağıdaki adımları içermiştir:

* **Kullanıcı Arayüzü Testleri**: Kullanıcı arayüzü, kullanıcının doğru bir şekilde oyuna başlamasını ve savaşçıları yerleştirmesini sağlamak için test edilmiştir. Bu testler, kullanıcı girişi doğruluğunu, butonların işlevselliğini ve iletişim kutularının doğru şekilde çalışmasını içermiştir.
* **Oyun Mantığı Testleri**: Oyun mantığı, savaşçıların doğru bir şekilde yerleştirilmesini, saldırı yapmalarını sağlamak için test edilmiştir. Bu testler, farklı savaşçı türlerinin özelliklerinin doğru bir şekilde uygulanıp uygulanmadığını kontrol etmiştir.
* **Hata Durumu Testleri**: Yazılım, kullanıcı hatalarını veya beklenmedik durumları ele alacak şekilde tasarlanmıştır. Bu nedenle, hata durumlarının doğru bir şekilde işlendiğini ve kullanıcıya uygun geri bildirimlerin sağlandığını doğrulamak için hata durumu testleri yapılmıştır.
* **Performans Testleri**: Yazılımın performansı, oyuncuların savaşçıları yerleştirmesi ve oyunun ilerlemesi sırasında gözlemlenmiştir. Büyük boyutlu ızgaralarda ve çok sayıda oyuncuyla yapılan testlerde herhangi bir performans düşüşü olup olmadığı kontrol edilmiştir.
* **Doğruluk Testleri**: Yazılımın doğruluğu, savaşçıların doğru bir şekilde yerleştirilmesi, saldırı yapmaları ve oyunun sona ermesi durumunda doğru sonuçların üretilip üretilmediği üzerine odaklanmıştır. Bu testler, oyunun beklenen sonuçları üretip üretmediğini kontrol etmiştir.
  1. **Yazılımın doğrulanması**
* **Bileşen Testleri:** Yazılımın her bir bileşeni, kendi başına test edilmiştir. Savasci sınıfının saldırı metodları test edilmiş ve bu testler, sınıfın beklenen şekilde davranıp davranmadığını doğrulamak için yapılmıştır. Ancak, bazı durumlarda savaşçıların saldırı metodlarında beklenmeyen davranışlar tespit edilmiştir. Örneğin, Healer sınıfındaki attack metodu, beklenenden farklı davranışlar sergilemiş ve bu durum test ekibindeki bir hata olarak kaydedilmiştir.
* **Bütünlük Testleri:** Yazılımın farklı bileşenlerinin bir araya geldiğinde beklenen şekilde çalışıp çalışmadığı kontrol edilmiştir. Özellikle, ızgara üzerinde savaşçıların yerleştirilmesi ve oyunun ilerleyişi test edilmiştir. Ancak, bazı durumlarda savaşçıların ızgara üzerinde doğru şekilde yerleştirilmediği veya oyun ilerleyişi sırasında beklenmeyen hataların oluştuğu tespit edilmiştir. Bu durumlar, yazılımın bütünlüğünün sağlanması için giderilmiştir.
* **Kullanıcı Arayüzü Testleri:** Kullanıcı arayüzü, kullanıcıların doğru şekilde etkileşime girebilmesi için test edilmiş ve butonların ve giriş alanlarının doğru şekilde çalışıp çalışmadığı kontrol edilmiştir. Ancak, bazı durumlarda kullanıcı arayüzünde uyumluluk sorunları yaşanmıştır. Örneğin, düşük çözünürlüklü ekranlarda butonların düzgün görünmediği veya bazı giriş alanlarının tam olarak işlevsel olmadığı tespit edilmiştir. Bu sorunlar, kullanıcı deneyimini artırmak için giderilmiştir.
* **Fonksiyonel Testler:** Yazılımın farklı senaryolara göre doğru sonuçları üretip üretmediği test edilmiştir. Özellikle, farklı oyuncu sayıları ve ızgara boyutlarına göre oyunun doğru şekilde başlatılıp başlatılmadığı kontrol edilmiştir. Ancak, bazı durumlarda farklı senaryoların test edilmediği veya test edilen senaryolarda beklenmeyen sonuçlar elde edildiği tespit edilmiştir. Bu durumlar, test kapsamının genişletilmesi ve senaryoların daha detaylı bir şekilde incelenmesiyle çözülmüştür.

#### Test Uygulaması:

* Bir test uygulaması geliştirdik ve bu uygulama ile yazılımın farklı bileşenleri test ettik.
* Test uygulaması, yazılımın bütün işlevselliğini kapsayacak şekilde tasarladık
* Her bir bileşen için ayrı test senaryoları oluşturduk ve bu senaryolar ile yazılımın doğruluğu kontrol ettik.

#### Sonuçlar:

**Bileşen Testleri Sonuçları:**

* Savaşçı sınıfı için tüm testler başarıyla geçildi.
* Guard, Archery, Gunner, Horseman ve Healer sınıflarının attack metodu doğru şekilde çalıştı ve beklenen sonuçları verdi.

**Bütünlük Testleri Sonuçları:**

* Yazılımın farklı bileşenlerinin bir araya geldiğinde beklenen şekilde çalıştığı doğrulandı.
* Oyunun ilerleyişi ve savaşçıların yerleştirilmesi gibi kritik işlevler doğru şekilde çalıştı.

**Kullanıcı Arayüzü Testleri Sonuçları:**

* Kullanıcı arayüzü, butonların ve giriş alanlarının doğru şekilde çalıştığı doğrulandı.
* Farklı ekran çözünürlüklerinde uyumlu bir şekilde görüntülendi.

**Fonksiyonel Testler Sonuçları:**

* Farklı senaryolara göre doğru sonuçlar üretildi.
* Farklı oyuncu sayıları ve ızgara boyutlarına göre oyunun doğru şekilde başladığı ve ilerlediği doğrulandı.

Karşılaşılan Zorluklar:

* **Bileşenlerin Karmaşıklığı:** Yazılımın bileşenleri, beklenenden daha karmaşık çıktı ve bu nedenle bileşenlerin test edilmesi daha zaman aldı. Özellikle, savaşçı sınıflarının birbirleriyle etkileşimi ve ızgara üzerindeki konumlandırılması gibi karmaşık işlevler test sürecini uzattı.
* **Uyumluluk Sorunları:** Kullanıcı arayüzü, farklı platformlarda ve ekran çözünürlüklerinde uyumluluk sorunları yaşadı. Özellikle, düşük çözünürlüklü ekranlarda butonların düzgün görünmemesi ve bazı giriş alanlarının tam olarak işlevsel olmaması kullanıcı deneyimini olumsuz etkiledi.

Çözüm Yöntemleri:

* **Modüler Test Yaklaşımı:** Karmaşık bileşenlerin test edilmesi için modüler bir yaklaşım benimsendi. Her bir bileşenin test edilmesi için ayrı test senaryoları oluşturuldu ve her bir testin bağımsız olarak çalışması sağlandı. Böylece, bileşenlerin karmaşıklığına rağmen, test süreci daha etkin bir şekilde yönetildi.
* **Uyumluluk Testleri:** Kullanıcı arayüzündeki uyumluluk sorunlarını çözmek için farklı platformlarda ve ekran çözünürlüklerinde uyumluluk testleri yapıldı. Bu testlerde, kullanıcı arayüzünün farklı ortamlarda doğru şekilde görüntülendiği ve işlevsel olduğu doğrulandı. Sorunlar tespit edildiğinde, gerekli düzeltmeler yapıldı ve kullanıcı deneyimi iyileştirildi.