

**T.C. ANTALYA BELEK ÜNİVERSİTESİ Mühendislik-Mimarlık
Fakültesi Yazılım Mühendisliği Bölümü**



TASARIM PROJESİ RAPORLARI DOSYASI (HAFTA 1-9)

Proje Adı: ERASMUS SİSTEMİ TEST GELİŞTİRME PROJESİ

Hazırlayan: Anıl Bilgehan YÜZGEÇ - 220007012

Danışmanlar: Dr. Öğr. Üyesi Adem ŞİMŞEK - Öğr. Gör. Soner DEDEOĞLU

Tarih: 16/01/2026

HAFTA 1 İLERLEME RAPORU

Proje Adı: Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi

Rapor Adı: Hafta 1 Çalışma Planı ve Proje Öneri Raporu

Proje Sorumlusu: Anıl Bilgehan YÜZGEÇ - 220007012

Danışman(lar): Dr. Adem ŞİMŞEK – Öğr. Gör. Soner DEDEOĞLU

Tarih: 24/11/2025

1. Özeti

Bu rapor, "Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi"'nin ilk hafta çalışmalarını özetlemektedir. Rapor kapsamında, projenin amacı ve detaylı kapsamı tanımlanmış, kullanılacak teknolojiler karşılaştırmalı olarak seçilmiş ve projenin genel bir çalışma planı sunulmuştur.

2. Hafta 1 Çıktıları ve Teslim Edilecekler

2.1. Görev Tanımları

Temel sorumluluk alanları aşağıda tanımlanmıştır:

- Analiz:** Proje gereksinimlerinin anlaşılması, kapsamın belirlenmesi ve test edilebilir senaryoların çıkarılması.
- Planlama:** Test stratejisinin oluşturulması, test planının hazırlanması, zaman yönetiminin yapılması ve risklerin öngörülmesi.
- Tasarım:** Test senaryolarının (test cases) detaylı olarak yazılması, test verilerinin hazırlanması ve otomasyon betiklerinin tasarlanması.
- Uygulama ve Yürütme:** Manuel ve otomatize edilmiş testlerin koşulması, bulguların kaydedilmesi ve hata takibinin yapılması.
- Raporlama:** Test sonuçlarının analiz edilmesi, ilerleme raporlarının hazırlanması ve Mezuniyet Projesi Yönergesi'ne uygun nihai proje raporunun yazılması.

2.2. Proje Öneri Raporu

2.2.1. Projenin Amacı:

Belek Üniversitesi'nin dijital Erasmus başvuru ve yönetim sisteminin kalitesini, güvenilirliğini ve kullanılabilirliğini güvence altına almaktır. Proje, sistemin gereksinimleri karşılayıp karşılamadığını doğrulamak, potansiyel hataları (bug) geliştirme yaşam döngüsünün erken bir aşamasında tespit etmek ve son kullanıcıya sunulmadan önce sistemin kararlı bir şekilde çalıştığından emin olmak için sistematik bir test süreci geliştirmeyi ve uygulamayı hedefler.

2.2.2. Projenin Kapsamı:

Projenin sınırlarını belirlemek, kaynakların doğru kullanılması ve hedeflere odaklanması için kritik öneme sahiptir.

Kapsam Dahilindeki Maddeler (In-Scope):

- **Test Planlaması ve Stratejisi:** Proje hedeflerine uygun, kapsamlı bir test stratejisi ve planının oluşturulması.
- **Test Ortamı Kurulumu:** Testlerin gerçekleştirileceği izole ve kontrollü bir test ortamının tanımlanması ve hazırlanması.
- **Test Senaryosu Tasarımı:** Aşağıda listelenen temel modüller için detaylı, tekrarlanabilir ve ölçülebilir manuel test senaryolarının (test cases) yazılması:
 - **Kullanıcı Yönetimi (Öğrenci, Koordinatör Hesapları):** Erasmus Puanı Hesaplama Motoru (YÖK kurallarına göre puan hesaplaması).
 - **Öğrenci Başvuru Modülü:** Giriş (Login), Şifre Sıfırlama, Başvuru Formu Doldurma, Belge Yükleme (Transkript vb.), Tercih Listesi Oluşturma.
 - **Belge Yükleme ve Yönetim Modülü:** Başvuru Görüntüleme, Belge Yükleme, Belge Onaylama/Reddetme, Puan Hesaplama Tetikleyicisi, Nihai Listeyi İlan Etme.
 - **Değerlendirme ve Onay/Red Mekanizmaları (Koordinatör Paneli):** Başvuru Görüntüleme, Belge Yükleme, Belge Onaylama/Reddetme
 - **Partner Üniversite Bilgileri ve Kontenjan Yönetimi**
 - **Raporlama ve Sorulama Ekranları**
- **Fonksiyonel Testler:** Sistemin belirtilen fonksiyonel gereksinimlere göre doğru çalışıp çalışmadığının doğrulanması (buton, form, navigasyon vs.). Sunucu yanıt kodlarının ve JSON veri yapısının kontrolü. Sistemin sağladığı Raporlama/Listeleme ekranlarından verinin kaydolduğu teyit edilir. Testlerin tarayıcılarında sorunsuz çalıştığını teyidi.
- **Kullanılabilirlik (UI/UX) Testleri:** Arayüzün sezgisel, kullanıcı dostu ve tutarlı olup olmadığıının değerlendirilmesi.
- **Temel Performans Testleri:** Sistemin belirli bir yük altındaki yanıt sürelerinin ve kararlılığının ölçülmesi (örn. 50 eş zamanlı kullanıcı senaryosu).
- **Test Otomasyonu (Pilot Çalışma):** Yüksek öncelikli ve sık tekrarlanan en az 2-3 test senaryosunun (örn. Öğrenci girişi ve başvuru oluşturma) bir otomasyon aracı (örn. Playwright for .NET) ile otomatikleştirilmesi.
- **Hata Yönetimi:** Tespit edilen tüm hataların sistematik bir şekilde raporlanması, önceliklendirilmesi ve takip edilmesi.
- **Raporlama:** Test süreci ve sonuçlarının "Mezuniyet Projesi Yönergesi"ne uygun olarak detaylı bir final raporunda belgelenmesi.

Kapsam Dışındaki Maddeler (Out-of-Scope):

- Erasmus Sistemi yazılıminin geliştirilmesi veya kodunun yazılması.
- Tespit edilen hataların **düzeltilmesi**. Projenin sorumluluğu hataları bulmak ve raporlamaktır, düzeltmek değil.
- Üretim ortamının veya sunucu altyapısının yönetimi.
- Gerçek öğrenci verilerinin kullanılması. Tüm testler anonimleştirilmiş veya sentetik (sahte) verilerle yapılacaktır.
- Sistemin tüm modüllerinin %100 test otomasyonu ile kapsaması. Otomasyon, bir pilot çalışma olarak sınırlı tutulacaktır.
- Mobil Uygulama (iOS/Android) testleri.
- Bankacılık entegrasyonları (Hibe ödemeleri simüle edilecektir, gerçek banka servisine gidilmeyecektir).

2.2.3. Hedeflenen Teknolojiler:

Test Otomasyon Teknolojileri Karşılaştırma Tablosu

Kriter	.NET (C#)	Python	JavaScript (Node.js)	Java	PHP
Dil Yapısı	Statik Tipli: Derleme zamanında hata yakalama, büyük projelerde daha güvenilir kod. Kesin ve yapısal.	Dinamik Tipli: Hızlı prototipleme, esnek ve daha az "kalıp" kod. Hatalar çalışma zamanında ortaya çıkar.	Dinamik Tipli: Hızlı ve esnek. Type Script ile statik tip desteği eklenebilir. Asenkron yapı doğalındadır .	Statik Tipli: Çok sağlam, platform bağımsız (JVM). Genellikle C#'dan daha çok kalıp kod gerektirir.	Dinamik (Modern PHP hibrit): Esnek. Modern sürümlerde güçlü tip desteği (type-hinting) var.
Geliştirme Ortamı (IDE)	Mükemmel: Visual Studio ve Rider, sınıfının en iyisi kod tamamlama, hata ayıklama ve refactoring desteği sunar.	Çok İyi: VS Code ve PyCharm güçlü araçlardır ancak Visual Studio'nun entegrasyon seviyesi genellikle	Çok İyi: VS Code, JavaScript/TypeScript için fiili standarttır ve harika bir deneyim sunar.	Mükemmel: IntelliJ IDEA / Eclipse.	Çok İyi: PhpStorm / VS Code.

		daha yüksektir.			
Ekosistem Entegrasyonu	Üstün (Eğer hedef .NET ise): Test edilecek sistem de .NET ise, eşsiz avantajlar sunar (WebApplicationFactory).	Genel Amaç: Herhangi bir web uygulamaıyla HTTP/Browser seviyesinde etkileşim kurar. Ekosistem e özel bir avantajı yoktur.	Frontend Odaklı: Test edilecek sistemin arayüzü React/Vue gibi JS kütüphaneleriyle yapıldıysa güçlü bir sinerji vardır.	Test edilecek sistem Spring Boot ise, Spring Test modülü ile bellek-içi test imkanı.	Test edilecek sistem Laravel/Symfony ise, PHPUnit ve framework'ün kendi araçları ile tam uyum.
Test Yazım Hızı	Orta. Statik tipler nedeniyle biraz daha fazla kod gerektirebilir ancak Fluent Assertions gibi kütüphanelerle hızlanır.	Çok Hızlı. Python'un sade sözdizimi sayesinde testler hızlıca yazılabilir.	Çok Hızlı. Cypress gibi araçların interaktif yapısı geliştirme hızını artırır.	Orta-Yavaş: Statik yapısı daha fazla kurulum gerektirir, bu da başlangıçta süreci yavaşlatır ancak güvenilirliği artırır.	Hızlı: Dinamik yapısı sayesinde az ön hazırlıkla hızlı bir şekilde test yazımına imkan tanır.
Performans ve Kararlılık	Yüksek. Derlenmiş bir dildir ve statik yapısı daha kararlı testler yazmaya yardımcı olur.	İyi. Yorumlanan bir dildir. Performansı genellikle yeterlidir ancak ağır yük testlerinde .NET kadar verimli olmayıabili r.	İyi. Asenkron doğası nedeniyle doğru yönetilmemişinde "flaky" (kararsız) testlere neden olabilir.	İyi: Yorumlanan dil, ancak modern PHP oldukça hızlı. Testler için fazlasıyla yeterli.	İyi: Testler için hızı ve performansı genellikle yeterlidir.
Kariyer ve Sektör	Güçlü Kurumsal Odak: Büyük ölçekli kurumsal uygulamalar,	Geniş Spektrum : Web, veri bilimi, yapay	Web Odaklı: Özellikle frontend ve tam yığın (full-stack)	Devasa kuru msal odak. En büyük şirketler, bankalar ve	Web'in temel taşı. Startuplar, ajanslar, KOBİ'ler. Dünya'daki web

	finans, sigortacılık gibi sektörlerde çok yaygındır.	zeka, otomasyon gibi çok geniş bir alanda popülerdir.	web geliştirme pozisyonları için vazgeçilmezdir.	Android geliştirme.	sitelerinin büyük çoğunluğu.
Ana Test Kütüphaneleri	xUnit, NUnit, Playwright, Selenium, RestSharp, FluentAssertions	Pytest, Playwright, Selenium, Requests, Locust	Jest, Cypress, Playwright, Mocha	JUnit, TestNG, Selenium, Mockito, REST Assured	PHPUnit, Pest, Laravel Dusk (UI)
Kurulum & Yapılabilirlik	Orta: dotnet CLI ile hızlı kurulum, Visual Studio her şeyi kolaylaştırır.	Kolay: pip ile paket yönetimi ve kurulum çok basittir.	Kolay: npm /yarn ile paket yönetimi çok basittir.	Orta. Maven/Gradle yapılandırması biraz zaman alabilir ama çok yapışalıdır.	Kolay: Composer ile paket yönetimi çok basit ve hızlıdır.
Topluluk & Kaynak	Büyük ve profesyonel (Microsoft destekli).	Devasa ve çok çeşitli alanlarda aktif.	Devasa. Web geliştirmedeki en hızlı büyüyen topluluk.	Devasa. En büyük ve köklü topluluklardan biri.	Devasa ve çok aktif. Özellikle Laravel ve Symfony etrafında.

Neden .NET?:

- Üstün Test Kapasitesi ve Hız:** WebApplicationFactory sayesinde sunucu kurulumuna gerek kalmadan, çok hızlı ve ağ sorunlarından etkilenmeyen %100 kararlı "bellek-içi" (in-memory) testler çalıştırılabilir.
- Kod ve Model Bütünlüğü:** Test ve ana proje arasındaki doğrudan model paylaşımı sayesinde JSON dönüşüm hataları ve senkronizasyon problemleri ortadan kalkar.
- Hatasız ve Yönetilebilir Kod:** C#'ın statik tip yapısı, basit hataları daha kod yazılırken (derleme anında) yakalar; gelişmiş IDE desteği ise kod güncellemelerini (refactoring) çok daha kolay ve güvenli hale getirir.
- Kariyer Yatırımı:** Büyük ölçekli kurumsal firmaların standart altyapısı olan .NET ekosistemini ve modern test tekniklerini öğrenmek, profesyonel hayatı güclü bir hazırlık sağlar.
- Gelişmiş Araç Desteği:** Visual Studio gibi güçlü bir IDE ile çalışmak, geliştirme sürecini daha verimli ve pürüzsüz kılar.

Projenin test otomasyonu ve yönetim süreçlerinde .NET ekosisteminden faydalanaacaktır. Temel kütüphaneler şunlardır:

- **Test Çerçevesi:** xUnit.net (Modern, standartlara uygun ve test izolasyonunu garanti eder.)
- **UI Otomasyon:** Playwright for .NET (Otomatik bekleme özelliği sayesinde çok kararlı ve hata ayıklaması kolay testler sunar.)
- **API Testleri:** RestSharp & WebApplicationFactory (API isteklerini temiz, okunabilir ve verimli bir şekilde yazmanızı sağlar.)
- **Doğrulama:** FluentAssertions (Testlerinizi doğal dil gibi okunabilir hale getirir ve proje kalitesini artırır.)
- **Mocking:** NSubstitute (En basit söz dizimi ile etkili birim testleri yazmanıza olanak tanır.)
- **Entegrasyon Testi:** WebApplicationFactory (Sunucu kurulumuna gerek kalmadan, uygulama içinde (In-Memory) çok hızlı ve ağ sorunlarından bağımsız test çalıştırır.)

2.2.4. Başarı Kriterleri:

- Sistemin kritik fonksiyonel gereksinimlerinin %100'ünü kapsayan test senaryolarının oluşturulması.
- Tespit edilen kritik ve yüksek öncelikli hataların detaylı bir şekilde raporlanması.
- En az 2 ana iş akışının başarılı bir şekilde otomatize edilmesi.
- Proje çıktılarının (Final Raporu, Sunum, Kod Deposu) zamanında ve yönergeye uygun olarak eksiksiz teslim edilmesi.

3. Çalışma Planı

3.1. Belek Üniversitesi Erasmus Sistemi Test Geliştirme Projesi İş Kırılım Yapısı (WBS):

- **1. PLANLAMA VE HAZIRLIK**
 - 1.1. Gereksinim Analizi
 - 1.2. Test Kapsamı Belirleme
 - 1.3. Araç Seçimi ve Teknoloji Kurumu
 - 1.4. Test Stratejisi Belirleme
 - 1.5. Test Planı Dokümantasyonu
- **2. TEST SENARYOLARI VE TASARIMI**
 - 2.1. Test Senaryoları Yazımı
 - 2.2. Test Verisi Hazırlığı
 - 2.3. Mimari Tasarım

- **3. TEST OTOMASYONU GELİŞTİRME**
 - 3.1. Test Kütüphanesi Kodlaması
 - 3.2. Sayfa Objelerinin Oluşturulması
 - 3.3. Test Betiklerinin Yazılması
- **4. TEST KOŞTURUMU, RAPORLAMA VE İYİLEŞTİRME**
 - 4.1. Testlerin Koşturulması
 - 4.2. Hata Yönetimi
 - 4.3. Performans Testleri
 - 4.4. Raporlama
- **5. TESLİM VE DEĞERLENDİRME**
 - 5.1. Proje Raporu (Tez) Yazımı
 - 5.2. Proje Dokümantasyonları
 - 5.3. Kullanıcı Kılavuzları
 - 5.4. Sunum ve Savunma

3.2. Belek Üniversitesi Erasmus Sistemi Test Geliştirme Projesi Gantt Şeması:

Faz / Görev	1. AY	2. AY	3. AY	4. AY	5. AY	6. AY
1. PLANLAMA VE HAZIRLIK	■■					
2. TEST SENARYOLARI VE TASARIMI		■■	■■			
3. TEST OTOMASYONU GELİŞTİRME			■■	■■	■■	
4. TEST KOŞTURUMU, RAPORLAMA VE İYİLEŞTİRME				■■	■■	
5. TESLİM VE DEĞERLENDİRME						■■

HAFTA 2 İLERLEME RAPORU

Proje Adı: Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi

Rapor Adı: Hafta 2 Gereksinim Analizi ve Test Stratejisi Raporu

Proje Sorumlusu: Anıl Bilgehan YÜZGEÇ - 220007012

Danışman(lar): Dr. Adem ŞİMŞEK – Öğr. Gör. Soner DEDEOĞLU

Tarih: 24/11/2025

1. Özeti

Bu rapor, "Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi"'nin ikinci hafta çalışmalarını sunmaktadır. Bu hafta kapsamında, test edilecek sistemin fonksiyonel ve fonksiyonel olmayan gereksinimleri analiz edilmiş ve projenin genel test stratejisini belirleyen taslak bir doküman oluşturulmuştur.

2. Hafta 2 Çıktıları ve Teslim Edilecekler

2.1. Test Edilebilir Gereksinimler Listesi

Bu bölüm, test edilecek sistemin ve projenin kendisinin karşılaşması gereken şartları detaylandırır.

A. Fonksiyonel Gereksinimler (Sistem Ne Yapmalı?)

Test süreci, sistemin aşağıdaki işlevleri doğru bir şekilde yerine getirdiğini doğrulamalıdır:

Öğrenci Rolü için:

- GR-F-01:** Öğrenci, sisteme geçerli bilgileriyle kayıt olabilmeli ve giriş yapabilmelidir. Hatalı girişlerde uygun uyarı mesajını göstermelidir. Test betiği bu senaryoyu otomatik koşabilmelidir. Sistem, geçerli @belek.edu.tr uzantılı öğrenci e-postası ve doğru şifre ile girişe izin vermelidir. Yanlış şifre girildiğinde sistem "Kullanıcı adı veya şifre hatalı" uyarısını kırmızı bir çerçeve içinde göstermelidir. (Negative Test)
- GR-F-02:** Kullanıcı 20 dakika boyunca işlem yapmazsa oturum otomatik olarak sonlanmalıdır ve login ekranına yönlendirilmelidir.
- GR-F-03:** Öğrenci, kişisel ve akademik bilgilerini profil sayfasında güncelleyebilmelidir.
- GR-F-04:** Öğrenci, partner üniversiteleri bölüm, ülke gibi kriterlere göre listeleyebilmeli ve filtreleyebilmelidir.
- GR-F-05:** Öğrenci, aktif başvuru döneminde yeni bir Erasmus başvurusu oluşturabilmelidir.
- GR-F-06:** Öğrenci, başvuru formuna istenen belgeleri (transkript, dil belgesi vb.) belirtilen formatlarda (PDF, JPG vb.) yükleyebilmelidir.
- GR-F-07:** Başvuru formunda "İletişim Bilgileri" boş bırakıldığında "Kaydet" butonu çalışmamalı veya uyarı vermelidir.

- **GR-F-08:** Öğrenci, başvurusunu son teslim tarihinden önce kaydedebilmeli, düzenleyebilmeli ve son halini gönderebilmelidir.
- **GR-F-09:** Kullanıcı başvurusunu "Taslak Olarak Kaydet" dediğinde, veri tabanına status: DRAFT olarak yazılmalıdır; "Onaya Gönder" dediğinde status: SUBMITTED olmalı ve değiştirilememelidir.
- **GR-F-10:** Öğrenci, gönderdiği başvurunun güncel durumunu (Değerlendirmede, Onaylandı, Reddedildi vb.) panelinden takip edebilmelidir.
- **GR-F-11:** "GNO (Genel Not Ortalaması)" alanı sadece 0.00 ile 4.00 arasında sayısal değer kabul etmelidir. Harf, 0.00'dan küçük veya 4.00'dan büyük girişler engellenmelidir.
- **GR-F-12:** Erasmus puanı belirlenen kurala göre hesaplanmalıdır Test otomasyonu, sisteme girdi verip çıktı sonucunun bu formülle birebir uyuşup uyuşmadığını doğrulamalıdır.
- **GR-F-13:** Eğer başvuru formunda "Şehit/Gazi Yakını" kutucuğu işaretli ve belgelendirilmişse, toplam puana eklenmelidir.
- **GR-F-14:** Eğer sistemde öğrencinin daha önce Erasmus yaptığına dair kayıt varsa, toplam puandan düşülmelidir.

Erasmus Koordinatörü/Yönetici Rolü için:

- **GR-F-15:** Koordinatör, kendine tanımlı yönetici kimliğiyle sisteme giriş yapabilmelidir.
- **GR-F-16:** Koordinatör, kendi bölümüne/fakültesine ait tüm öğrenci başvurularını bir yönetici panelinde listeleyebilmelidir.
- **GR-F-17:** Koordinatör, her bir başvurunun detaylarını ve öğrencinin yüklediği belgeleri görüntüleyebilmelidir.
- **GR-F-18:** Koordinatör, başvuruları önceden tanımlanmış kriterlere göre değerlendирerek "Onayla", "Reddet" veya "Revizyon İste" işlemlerini yapabilmelidir. Admin tarafından "Onaylandı" statüsüne çekilen bir başvurunun durumu, öğrenci panelinde anlık olarak "Onaylandı" şeklinde güncellenmelidir.
- **GR-F-19:** Yönetici bir başvuruyu reddettiğinde "Ret Gerekçesi" girilmesi zorunlu olmalıdır.
- **GR-F-20:** Koordinatör, temel istatistiklere (başvuru sayısı, onaylanan öğrenci sayısı vb.) ilişkin basit raporlar oluşturabilmelidir.

B. Fonksiyonel Olmayan Gereksinimler (Sistem Nasıl Olmalı?)

- **GR-NF-01 (Kullanılabilirlik):** Sistem arayüzü, tutarlı ve anlaşılır bir tasarıma sahip olmalıdır. Ortalama bir öğrenci, yardım almadan başvuru sürecini 15 dakika içinde tamamlayabilmelidir.

- **GR-NF-02 (Performans):** Normal kullanım koşullarında tüm sayfalar 3 saniyeden daha kısa sürede yüklenmelidir. Sistem, 50 kullanıcının aynı anda aktif olarak işlem yaptığı bir yük altında yavaşlama olmadan çalışmaya devam etmelidir.
- **GR-NF-03 (Uyumluluk):** Sistem çeşitli tarayıcılarda da görsel ve işlevsel bir sorun olmadan çalışmalıdır.
- **GR-NF-04 (Raporlama):** Her test koşumundan sonra (Run), otomatik olarak HTML formatında, grafik destekli (Kaç test geçti, kaç kaldı) bir rapor oluşturulmalıdır.
- **GR-NF-05 (Hata Yönetimi):** Bir test adımı başarısız olduğunda (Assertion Failed), otomasyon sistemi o anın ekran görüntüsünü (Screenshot) alıp "Failures" klasörüne kaydetmelidir.
- **GR-NF-06 (Bakım Kolaylığı):** Test kodları **Page Object Model (POM)** tasarım kalıbine uygun yazılmalı, web elementlerinin locater'ları tek bir merkezden yönetilmelidir.
- **GR-NF-07 (Veri Bağımsızlığı):** Testler her çalıştırıldığında kendi test verisini (Örn: Rastgele öğrenci ismi) üretmeli, sistemdeki mevcut verilere zarar vermemelidir.
- **GR-NF-08 (Test Ortamı):** Proje, gerçek (Canlı/Production) sistem üzerinde değil, veritabanı sıfırlanabilir bir **Sandbox/Test Ortamı** üzerinde veya Localhost'ta çalışan bir mock (taklit) sistem üzerinde koşulmalıdır.
- **GR-NF-09 (Tarayıcı Bağımsızlığı):** Test betikleri config dosyasından değiştirilerek çeşitli tarayıcılarda hatasız çalışabilмелidir (Cross-Browser Testing).
- **GR-NF-10 (Headless Mod):** Testler, grafik arayüzü olmayan (CI/CD ortamları için) "Headless Mode"da çalışabilme yeteneğine sahip olmalıdır.

2.2. Test Stratejisi Dokümanı (Taslak)

2.2.1. Genel Yaklaşım

Projede, sistemin davranışını son kullanıcı gözünden doğrulayan **Kara Kutu (Black-Box)** test teknigi ağırlıklı olarak kullanılacaktır. Manuel ve otomatize testleri bir arada barındıran **hibrit bir test yaklaşımı** benimsenecektir. Bu yaklaşım, hem esnek keşif testlerine olanak tanıyacak hem de tekrar eden görevlerde otomasyonun gücünden faydalananarak verimliliği artıracaktır.

2.2.2. Test Seviyeleri

Test aktiviteleri, yazılım geliştirme yaşam döngüsünün farklı aşamalarına odaklanan aşağıdaki seviyelerde gerçekleştirilecektir:

- **Birim Testi (Unit Test):**
 - **Amaç:** Yazılımın en küçük test edilebilir parçalarının (metotlar, fonksiyonlar, sınıflar) tek başına, izole bir şekilde doğru çalışıp çalışmadığını doğrulamak.
 - **Kapsam:** Bu proje test projesi olduğu için birim testleri yazılmayacak, ancak test edilecek Erasmus sisteminin bu seviyede test edildiği varsayılacaktır. Bu seviyenin farkında olmak, projenin teorik temelinin sağlamlığını gösterir.

- **Entegrasyon Testi (Integration Test):**
 - **Amaç:** Aynı ayrı geliştirilmiş olan birimlerin veya modüllerin bir araya getirildiğinde birlikte uyum içinde çalışıp çalışmadığını kontrol etmek.
 - **Kapsam:** Özellikle farklı modüller arasındaki API entegrasyonları test edilecektir. **Örnek:** Öğrenci Başvuru modülünün, Belge Yönetimi API'si ile doğru veri alışverişi yapıp yapmadığı.
- **Sistem Testi (System Test):**
 - **Amaç:** Tamamen entegre olmuş sistemin, belirtilen tüm gereksinimlere uygunluğunu doğrulamak.
 - **Kapsam:** Projenin ana odağı bu seviye olacaktır. Uygulama, bir bütün olarak, başlangıçtan sona kadar kullanıcı senaryolarıyla (uçtan uca) test edilecektir. **Örnek:** Yeni bir öğrencinin kaydolması, giriş yapması, başvuru oluşturması, belge yüklemesi ve başvurusunu başarıyla göndermesi.
- **Kullanıcı Kabul Testi (User Acceptance Test-UAT):**
 - **Amaç:** Yazılımın son kullanıcı veya müşteri tarafından, gerçek dünya senaryoları altında kabul edilebilir olup olmadığını doğrulamak.
 - **Kapsam:** Proje sonunda, tasarlanan test senaryoları bir "son kullanıcı" gözüyle çalıştırılarak, sistemin iş akışlarının mantıklı ve beklenileri karşılar nitelikte olduğu teyit edilecektir.

2.2.3. Test Türleri

Sistemin farklı kalite özelliklerini ölçmek için aşağıdaki test türleri uygulanacaktır:

A) Fonksiyonel Test Türleri:

- **Smoke Test (Duman Testi):**
 - **Amaç:** Yeni bir yazılım sürümü (build) test ortamına yüklenliğinde, sistemin en temel ve kritik fonksiyonlarının çalışıp çalışmadığını kontrol eden hızlı ve yüzeysel testlerdir. Amacı, "bu sürüm daha detaylı test edilmeye değer mi?" sorusuna cevap vermektir. **Örnek:** Kullanıcı giriş yapabiliyor mu? Ana sayfa açılıyor mu?
- **Regression Test (Regresyon Testi):**
 - **Amaç:** Yazılımda yapılan bir değişiklik, güncelleme veya hata düzeltmesi sonrasında, mevcut ve daha önceden çalışan özelliklerin bozulmadığından emin olmak için yapılan testlerdir.
 - **Kapsam:** Bu testler, otomasyon için en ideal adaylardır. Özellikle "Kullanıcı Giriş => Başvuru Oluşturma" gibi kritik iş akışları, her değişiklikten sonra otomatik olarak çalıştırılacaktır.
- **Fonksiyonel Testler:**
 - **Amaç:** Sistemin, proje gereksinimlerinde belirtilen tüm fonksiyonel özellikleri doğru bir şekilde yerine getirip getirmedğini doğrulamak.
 - **Kapsam:** "Test Edilebilir Gereksinimler Listesi"ndeki tüm maddeler bu kapsamda detaylı olarak test edilecektir.

B) Fonksiyonel Olmayan Test Türleri:

- **Kullanılabilirlik Testleri (Usability Testing):**
 - **Amaç:** Arayüzün ne kadar kullanıcı dostu, sezgisel ve öğrenilebilir olduğunu değerlendirmek.
 - **Kapsam:** Arayüzün tutarlılığı, menülerin ve butonların anlaşılabilirliği, hata mesajlarının yol göstericiliği gibi konular değerlendirilecektir.
- **Performans Testleri (Performance Testing):**
 - **Amaç:** Sistemin belirli bir yük altındaki hızını, yanıt süresini ve kararlılığını ölçmek.
 - **Kapsam (Temel Seviye):**
 - **Yük Testi (Load Testing):** Beklenen normal ve en yüksek kullanıcı yükü altında sistemin davranışını gözlemlemek.
 - **Stres Testi (Stress Testing):** Sistemin sınırlarını zorlayarak (beklenenden çok daha fazla kullanıcı ile) nerede ve nasıl hata verdiği tespit etmek.
 - **Seçilen Araç:** .NET ekosistemiyle daha uyumlu olan k6 gibi modern bir alternatif.
- **Uyumluluk Testleri (Compatibility Testing):**
 - **Amaç:** Yazılımın farklı ortamlarda (tarayıcılar, işletim sistemleri, ekran çözünürlükleri) sorunsuz bir şekilde çalışıp çalışmadığını doğrulamak.
 - **Kapsam:** Uygulamanın en güncel Google Chrome ve Mozilla Firefox sürümlerinde hem görsel hem de işlevsel olarak doğru çalıştığı teyit edilecektir.

2.2.4. Otomasyon Yaklaşımı:

Test otomasyonu, projenin verimliliğini artırmak için pilot düzeyde uygulanacaktır.

Otomasyon için önceliklendirilecek senaryolar şunlardır:

- Yüksek tekrar oranına sahip (Regresyon testleri için ideal).
- Kritik iş akışını içeren (örn. Kullanıcı Girişi => Başvuru Oluşturma).
- Manuel testleri zaman alan.
- **Seçilen Araç:** Playwright for .NET

2.2.5. Hata Yönetimi (Defect Management):

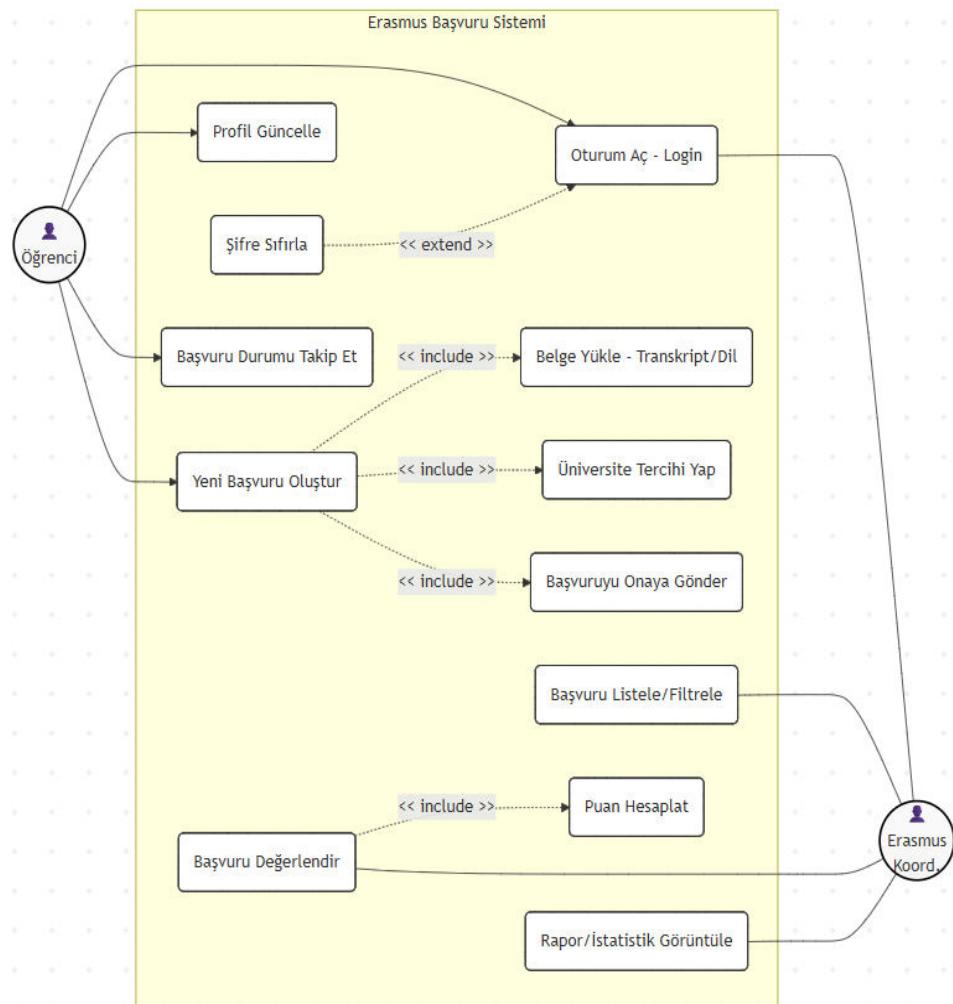
Tespit edilen tüm hatalar, aşağıdaki bilgileri içeren standart bir formatta belgelenecektir:

- Hata Başlığı
- Tekrarlama Adımları (Adım adım)
- Beklenen Sonuç
- Gerçekleşen Sonuç
- Ekran Görüntüsü / Video
- Önem ve Öncelik Seviyesi (Kritik, Yüksek, Orta, Düşük)

EKLER

Aşağıda raporda atıfta bulunulan ve raporu destekleyici diyagramlar bulunmaktadır.

EK – A: Sistem Use Case Diyagramı (Taslak):



EK – A: Tablo: Use Case Senaryo Detayları (Taslak):

Use Case ID	Senaryo Adı	Aktor(ler)	Açıklama ve İlişki
UC-01	Oturum Aç (Login)	Öğrenci, Koordinatör	Sisteme e-posta ve şifre ile güvenli giriş yapılmasını sağlar.
UC-03	Yeni Başvuru Oluştur	Öğrenci	Aktif dönemde yeni bir Erasmus başvurusu başlatır.
UC-04	Belge Yükle	Öğrenci	Başvurunun geçerli olması için zorunlu olan PDF belgelerin sisteme yüklenmesini ifade eder. "Başvuru Oluştur" senaryosuna <<include>> ile bağlıdır (Belge yüklemeden başvuru tamamlanamaz).
UC-06	Başvuruyu Onaya Gönder	Öğrenci	Taslak halindeki başvurunun kesinleştirilip değerlendirmeye gönderilmesidir. Bu işlem yapıldıktan sonra başvuru üzerinde değişiklik yapılamaz.
UC-09	Başvuru Değerlendir	Koordinatör	Gelen öğrenci başvurusunu inceleme, belgeleri kontrol etme ve Onay/Red durumuna getirme işlemidir.
UC-10	Erasmus Puanı Hesaplat	Sistem (Admin tetikler)	Puan hesaplama algoritmasını çalıştırır. Koordinatör "Değerlendir" dediğinde arka planda sistem hesaplamayı yaptığı için <<include>> ilişkisi vardır.

HAFTA 3 İLERLEME RAPORU

Proje Adı: Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi

Rapor Adı: Hafta 3 Yazılım Gereksinimleri Dokümantasyonu (SRS)

Proje Sorumlusu: Anıl Bilgehan YÜZGEÇ - 220007012

Danışman(lar): Dr. Adem ŞİMŞEK – Öğr. Gör. Soner DEDEOĞLU

Tarih: 16/01/2026

1. Özет

Bu rapor, projenin 3. Hafta çıktılarına uygun, önceki haftalarda belirlenen temel gereksinimler detaylandırılarak **Yazılım Gereksinimleri Spesifikasiyonu (SRS v1)** haline getirilmiştir. Bu doküman, test senaryolarının (Test Cases) temel dayanağını oluşturacaktır. Sistemin veri giriş formatları, iş kuralları (Business Rules) ve hata mesajları, otomasyon kodlarının "Assertion" (Doğrulama) aşamalarında kullanılmak üzere kesinleştirilmiştir. Hafta 2 raporunda belirlenen 20 fonksiyonel ve 10 fonksiyonel olmayan gereksinim, teknik kısıtlar ve iş kuralları eklenerek "Test Edilebilir" formatta dokümante edilmiştir.

2. Fonksiyonel Gereksinimler (SRS Detaylandırması)

Hafta 2 raporundaki ham maddeler, test senaryolarına girdi sağlayacak şekilde detaylandırılmıştır.

Tablo 1: Öğrenci Modülü Gereksinimleri

ID (Yeni)	Kaynak ID	Gereksinim Tanımı	SRS Detay (Veri Tipi / Kısıt / İş Kuralı)
SRS-F-01	GR-F-01	Sisteme Kayıt/Giriş	Format: @belek.edu.tr uzantılı mail zorunludur. Hata: Yanlış girişte "Kullanıcı adı veya şifre hatalı" mesajı dönmelidir.
SRS-F-02	GR-F-02	Oturum Zaman Aşımı	Süre: 20 dakika (1200 saniye) hareketsizlikte oturum düşmeli (Session Timeout).
SRS-F-03	GR-F-03	Profil Güncelleme	Kısıt: T.C. Kimlik No alanı değiştirilemez (Read-only) olmalıdır.
SRS-F-04	GR-F-04	Üniversite Filtreleme	Filtreler: Bölüm (Enum), Ülke (Enum), Anlaşma Tipi (Erasmus/Exchange).
SRS-F-05	GR-F-05	Başvuru Oluşturma	Önkoşul: Aktif bir başvuru dönemi takvimde tanımlı olmalıdır.
SRS-F-06	GR-F-06	Cocuk Tercih Sistemi	Kural: Öğrenci, anlaşmalı kurumlar listesinden en fazla 3 adet üniversiteden öncelik sırasına göre (1., 2. ve 3. Tercih) seçerek başvuru listesini oluşturabilmelidir.
SRS-F-07	GR-F-07	Belge Yükleme	Format: Sadece PDF, JPG, PNG. Boyut: Max 5 MB.
SRS-F-08	GR-F-08	Zorunlu Alan Kontrolü	Validasyon: İletişim bilgileri boşsa "Kaydet" butonu Disabled (Pasif) olmalıdır.
SRS-F-09	GR-F-09	Düzenleme Yetkisi	Kural: Son teslim tarihinden sonra Edit (Düzenle) butonu gizlenmelidir.

ID (Yeni)	Kaynak ID	Gereksinim Tanımı	SRS Detayı (Veri Tipi / Kısıt / İş Kuralı)
SRS-F-10	GR-F-10	Durum Yönetimi	State: Kayıttı DRAFT, gönderimde SUBMITTED. SUBMITTED veri değiştirilemez.
SRS-F-11	GR-F-11	Durum Takibi	UI: Durum değişiklikleri öğrenciye renkli etiketle (Badge) gösterilmelidir.
SRS-F-12	GR-F-12	GNO Validasyonu	Veri Tipi: Decimal (0.00 - 4.00). Ayraç: Nokta (.). Virgül girilirse otomatik düzeltilmeli.
SRS-F-13	GR-F-13	Puan Hesaplama	Formül: (GNO * 25 * 0.5) + (Dil * 0.5). Sistem bu hesabı otomatik yapmalıdır.
SRS-F-14	GR-F-14	Şehit/Gazi Puanı	Kural: Checkbox işaretli ve belge yüklü ise +15 Puan .
SRS-F-15	GR-F-15	Hibe Kesintisi	Kural: Geçmiş Erasmus kaydı varsa -10 Puan .

Tablo 2: Koordinatör Modülü Gereksinimleri

ID (Yeni)	Kaynak ID	Gereksinim Tanımı	SRS Detayı (Veri Tipi / Kısıt / İş Kuralı)
SRS-F-16	GR-F-16	Yönetici Girişи	Yetki: Sadece Role: Admin yetkisine sahip kullanıcılar erişebilir.
SRS-F-17	GR-F-17	Liste Görüntüleme	Data Grid: Sayfalama (Pagination) özelliği olmalı, her sayfada 20 kayıt gösterilmeli.
SRS-F-18	GR-F-18	Detay/Belge İnceleme	UI: Öğrenci belgeleri tarayıcı içinde önizleme (Preview) modunda açılabilmelidir.
SRS-F-19	GR-F-19	Değerlendirme (Onay)	Action/Eylem: Onaylandığında öğrenciye otomatik e-posta bildirimi gitmelidir. İş Kuralı: Sistem, bir başvuru onaylanacağı sırada ilgili üniversitenin o dönemki kontenjanını kontrol etmelidir. Eğer Onaylı Öğrenci Sayısı \geq Kota ise sistem onaya izin vermeli.
SRS-F-20	GR-F-20	Ret Gerekçesi	Zorunluluk: Durum REJECTED seçilirse "Açıklama" alanı zorunlu (Required) hale gelir.
SRS-F-21	GR-F-21	İstatistik Raporu	Çıktı: Raporlar Excel (.xlsx) formatında indirilebilir olmalıdır.

3. Fonksiyonel Olmayan Gereksinimler (SRS Teknik Kısıtları)

Hafta 2 raporundaki 10 madde, test otomasyonunda kullanılacak somut metriklere dönüştürülmüştür.

- SRS-NF-01 (Kullanılabilirlik):** Arayüzde "breadcrumb" (navigasyon yolu) bulunmalı, kullanıcı en fazla 3 tıkla başvuru ekranına ulaşabilmelidir.

- SRS-NF-02 (Performans - Yanıt):** Sayfa yükleme süreleri (Page Load Time) < 3 saniye olmalıdır. (Araç: k6).
- SRS-NF-03 (Performans - Yük):** 50 Eşzamanlı Kullanıcı (Virtual Users) altında hata oranı %1'in altında olmalıdır.
- SRS-NF-04 (Uyumluluk):** Chrome (v120+) ve Firefox (v115+) tarayıcılarında render hatası olmamalıdır.
- SRS-NF-05 (Raporlama):** Otomasyon sonuçları HTML formatında, Pasta Grafik (Pie Chart) içerecek şekilde üretilmelidir.
- SRS-NF-06 (Hata Kanıtı):** Başarısız testlerde (Fail) ekran görüntüsü formatted_date_fail.png adıyla kaydedilmelidir.
- SRS-NF-07 (Bakım):** Kod yapısı "Page Object Model (POM)" mimarisine %100 uygun olmalıdır.
- SRS-NF-08 (Veri Güvenliği):** Test verileri her koşumda rastgele üretilmeli (Faker library), gerçek öğrenci verisi **kullanılmamalıdır**.
- SRS-NF-09 (Test Ortamı):** Testler Sandbox ortamında koşulmalı, Canlı (Production) veritabanına bağlanılmamalıdır.
- SRS-NF-10 (CI/CD):** Testler "Headless" (Arayüzsüz) modda çalıştırılabilir olmalıdır.

4. Gereksinim İzlenebilirlik Matrisi Planı (Traceability Matrix Plan) ve Test Kapsamı

Projenin test odaklı olması nedeniyle, her bir gereksinimin hangi test seviyesinde ve hangi araçla doğrulanacağı aşağıdaki matris yapısıyla planlanmıştır. Bu matris, ilerleyen haftalarda yapılacak test senaryoları (Test Cases) için bir yol haritası niteligidendir.

Tablo 3: Gereksinim İzlenebilirlik Matrisi (Traceability Matrix)

Gereksinim ID	Gereksinim Adı	Test Seviyesi / Türü	Hedeflenen Test Aracı / Yöntemi
SRS-F-01	Sisteme Kayıt/Giriş	Fonksiyonel / Smoke Test	Playwright (UI Otomasyonu)
SRS-F-02	Oturum Zaman Aşımı	Sistem Testi	Playwright (Wait & Timeout Kontrolü)
SRS-F-03	Profil Güncelleme	Fonksiyonel Test	Playwright (Input Doğrulama)
SRS-F-04	Üniversite Filtreleme	Fonksiyonel Test	Playwright (Dropdown & Grid Kontrolü)
SRS-F-05	Başvuru Oluşturma	Uçtan Uca (E2E) Test	Playwright (Senaryo Bazlı Test)
SRS-F-06	Çoklu Tercih Sistemi	Sistem Testi (Fonksiyonel) ve Entegrasyon Testi	Kara Kutu Testi (Sınır Değer Analizi) ve Postman / SQL
SRS-F-07	Belge Yükleme	Fonksiyonel Test	Playwright (File Upload Event)
SRS-F-08	Zorunlu Alan Kontrolü	Negatif Test	Playwright (Assertion: Button Disabled)
SRS-F-09	Düzenleme Yetkisi	İş Kuralı Testi	Playwright (Element Visibility Check)
SRS-F-10	Durum Yönetimi	Entegrasyon / E2E	Playwright & Veritabanı Kontrolü

Gereksinim ID	Gereksinim Adı	Test Seviyesi / Türü	Hedeflenen Test Aracı / Yöntemi
SRS-F-11	Durum Takibi	UI Testi	Playwright (Visual Assertion)
SRS-F-12	GNO Validasyonu	Birim Testi (Unit)	xUnit (Input Validator Logic)
SRS-F-13	Puan Hesaplama	Birim Testi (Unit)	xUnit (Matematiksel Algoritma Testi)
SRS-F-14	Şehit/Gazi Puanı	Birim Testi (Unit)	xUnit (Koşullu Mantık Testi)
SRS-F-15	Hibe Kesintisi	Birim Testi (Unit)	xUnit (Koşullu Mantık Testi)
SRS-F-16	Yönetici Girişи	Güvenlik / Yetki Testi	Playwright (Role Based Access Control)
SRS-F-17	Liste Görüntüleme	UI Testi	Playwright (Pagination Check)
SRS-F-18	Detay/Belge İnceleme	UI Testi	Playwright (Preview Modal Check)
SRS-F-19	Değerlendirme (Onay)	E2E Test	Playwright (İş Akışı Doğrulama)
SRS-F-20	Ret Gerekçesi	Negatif Test	Playwright (Zorunlu Alan Kontrolü)
SRS-F-21	İstatistik Raporu	Fonksiyonel Test	Playwright (Download Event Check)
SRS-NF-01	Kullanılabilirlik	Manuel Test / UX	Gözden Geçirme (Review)
SRS-NF-02	Performans (Yanıt)	Performans Testi	k6 (Load Testing)
SRS-NF-03	Performans (Yük)	Stres Testi	k6 (50 VUsers Simulation)
SRS-NF-04	Uyumluluk	Cross-Browser Test	Playwright Config (Chromium & Firefox)
SRS-NF-05	Raporlama	Konfigürasyon Testi	Playwright Reporter Kontrolü
SRS-NF-06	Hata Kanıtı (Screenshot)	Konfigürasyon Testi	Playwright (OnFail Configuration)
SRS-NF-07	Bakım (POM)	Statik Kod Analizi	Kod Gözden Geçirme (Code Review)
SRS-NF-08	Veri Güvenliği	Veri Yönetimi	Faker Library (.NET) Kullanımı
SRS-NF-09	Test Ortamı	Ortam Doğrulama	Sandbox/localhost Kontrolü
SRS-NF-10	CI/CD (Headless)	Süreç Testi	CLI / Headless Execution Kontrolü

5. Proje Varsayımları ve Bağımlılıklar

Test otomasyon projesinin başarısı, test edilen ortamın kararlılığına ve verilerin yönetilebilirliğine bağlıdır. Bu bağlamda, proje süresince geçerli olacak teknik varsayımlar ve dış bağımlılıklar aşağıda detaylandırılmıştır:

5.1. Test Ortamı ve Erişim Varsayımları

- **İzole Test Ortamı (Sandbox):** Testlerin yürütüleceği ortamın, canlı (Production) sistemden tamamen izole edilmiş bir "Sandbox" veya "Staging" ortamı olduğu kabul edilmiştir. Bu ortamda yapılacak veri silme veya güncelleme işlemleri gerçek kullanıcıları etkilemeyecektir.
- **Sürüm Kararlılığı (Code Freeze):** Otomasyon testlerinin geliştirilmesi ve koşulması sırasında, test edilen uygulamanın kod tabanında anlık ve büyük değişiklikler yapılmayacağı varsayılmaktadır. Test koşumu süresince ilgili sürümün "kararlı" (stable) kalacağı kabul edilmiştir.
- **Ağ Erişimi:** Test otomasyonunu çalıştıracak istemcinin (Runner), test ortamına ağ kısıtlaması (Firewall/VPN engeli) olmadan erişebildiği varsayılmaktadır.

5.2. Veri Yönetimi ve KVKK Uyumluluğu

- **Sentetik Veri Üretimi (Mock Data):** 6698 sayılı Kişisel Verilerin Korunması Kanunu (KVKK) gereği, test senaryolarında hiçbir şekilde gerçek öğrenci, personel veya kurum verisi kullanılmayacaktır.
 - *Uygulama:* Test verileri (İsim, TC No, E-posta vb.), .NET ekosistemindeki "Bogus" veya benzeri kütüphaneler kullanılarak çalışma zamanında (Runtime) rastgele ve kurallara uygun (Valid) şekilde üretilecektir.
- **Veri Temizliği (Teardown):** Her test senaryosu, "kendi verisini kendi yaratır" prensibine göre çalışacaktır. Test tamamlandıktan sonra üretilen verilerin sistemde kirlilik yaratmaması adına veritabanı geri yükleme (Database Rollback) veya veri silme prosedürlerinin uygulanabilir olduğu varsayılmıştır.

5.3. Dış Bağımlılıklar ve Entegrasyon Simülasyonu

- **Servis Sanallaştırma (Stubbing/Mocking):** Erasmus sistemi, YÖK (Öğrenci Belgesi Sorgulama) veya E-Devlet (Kimlik Doğrulama) gibi dış servislere bağımlı olsa da, test projesi bu dış kaynakların erişilebilirliğine bağımlı olmayacağından emin olmayı planlıyor.
 - *Teknik Detay:* Dış servislere giden çağrılar, **NSubstitute** veya benzeri bir Mocking kütüphanesi ile "taklit" edilecektir. Böylece dış servislerin yavaşlığı, kesintisi veya kota sınırları test sonuçlarını (False Negative) etkilemeyecektir.
- **E-Posta Sunucusu:** "Şifremi Unuttum" veya "Başvuru Onayı" gibi senaryolarda gerçek e-posta gönderimi yerine, sistemin e-posta gönderdiğini varsayılan log kayıtları veya sanal SMTP sunucuları (örn: Mailhog) baz alınacaktır.

5.4. Yazılım ve Donanım Gereksinimleri

Testlerin başarılı bir şekilde koşulabilmesi için test yürütücü makinede (Runner) aşağıdaki asgari gereksinimlerin sağlandığı kabul edilmiştir:

- **.NET SDK:** Sürüm 8.0 veya üzeri.
- **Tarayıcılar:** Playwright testleri için Chromium, Firefox ve WebKit motorlarının güncel sürümleri.
- **İşletim Sistemi:** Windows, Linux veya macOS platformlarından herhangi biri.

6. Sonuç

Bu raporda, önceki aşamada belirlenen **toplam 31 gereksinim (21 Fonksiyonel + 10 Fonksiyonel Olmayan)**, yazılım geliştirme ve test süreçlerine (xUnit ve Playwright entegrasyonu) uygun hale getirilmiş ve SRS dökümanı tamamlanmıştır.

HAFTA 4 İLERLEME RAPORU

Proje Adı: Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi

Rapor Adı: Hafta 4 Sistem Mimarisi/Teknoloji Entegrasyonu ve Genel Sistem Diyagramı

Proje Sorumlusu: Anıl Bilgehan YÜZGEÇ - 220007012

Danışman(lar): Dr. Adem ŞİMŞEK – Öğr. Gör. Soner DEDEOĞLU

Tarih: 16/01/2026

1. Özeti

Bu rapor, projenin 4. haftasında gerçekleştirilen "Sistem Mimarisi Tasarımı" çalışmalarını kapsar. Yazılım Projesi Öğrenci Rehberi doğrultusunda, test otomasyon projesinin katmanlı mimarisi (Layered Architecture) tasarlanmış, seçilen teknolojilerin proje içindeki rolleri kesinleştirilmiş ve sürdürülebilir bir kod yapısı için **Çözüm (Solution) Hiyerarşisi** oluşturulmuştur. Tasarlanan mimari, SRS dokümanındaki "Bakım Kolaylığı" (SRS-NF-07) ve "CI/CD Uyumluluğu" (SRS-NF-10) gereksinimlerini doğrudan karşılamaktadır.

2. Hafta 4 Çıktıları: Sistem Mimarisi ve Teknoloji Seçimi

2.1. Kesinleşen Teknoloji Yığını (Technology Stack)

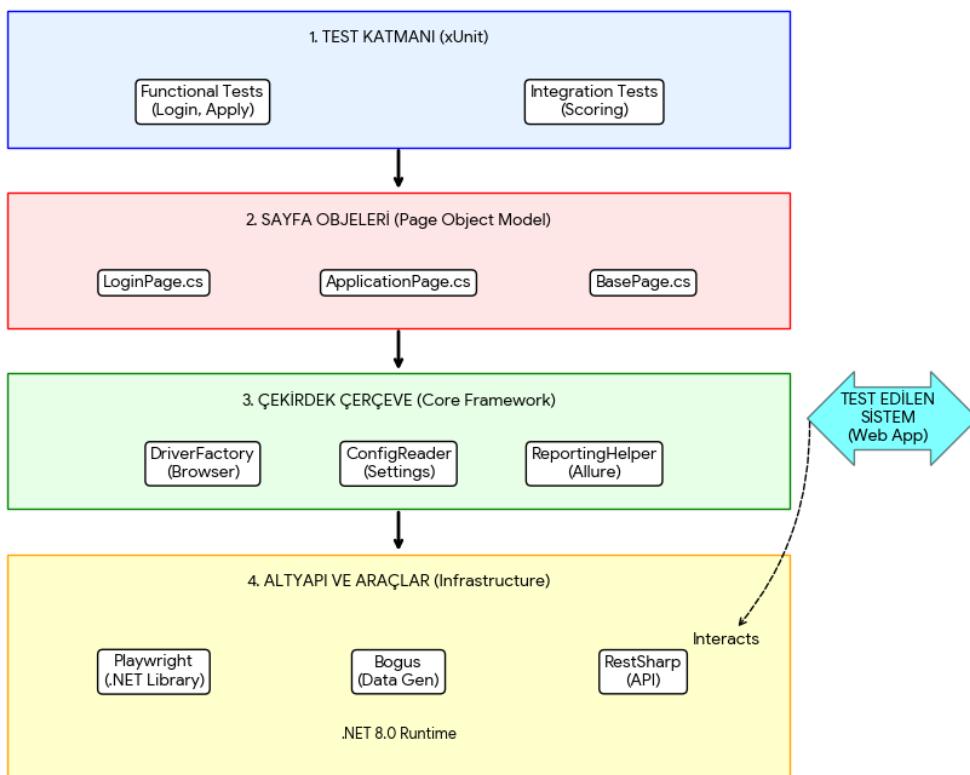
Hafta 1 raporunda yapılan ön analizler sonucunda, projenin performans ve güvenilirlik hedeflerine en uygun olan aşağıdaki teknoloji seti, sistem mimarisinin temel taşıları olarak已被选中:

Bileşen	Seçilen Teknoloji	Seçim Gerekçesi (SRS Referansı)
Programlama Dili	C# (.NET 8.0)	Tip güvenliği (Type-safety) ve kurumsal standartlara uyum.
Test Çerçevesi	xUnit	Testlerin paralel koşulabilmesi ve veri odaklı (Data-Driven) test yapısı.
UI Otomasyonu	Playwright for .NET	"Auto-wait" özelliği ile "Flaky test" (kararsız test) oranını düşürmesi ⁴ .
API Testi	RestSharp	Servis entegrasyonlarının (Mocking) hızlı simülasyonu.
Assertion (Doğrulama)	FluentAssertions	Test kodlarının doğal dil gibi (Readable) okunabilmesi.
Test Verisi	Bogus	KVKK uyumlu sahte (Mock) veri üretimi (SRS-NF-08).
Raporlama	Allure Report / HTML	Görsel ve detaylı hata raporlama (SRS-NF-05).
Mocking (Sanallaştırma)	NSubstitute	En basit söz dizimi ile etkili birim testleri yazmanızı olanak tanır.

2.2. Test Otomasyon Mimarisi (Architectural Design)

Proje, bakım maliyetini düşürmek ve kod tekrarını önlemek amacıyla **Page Object Model (POM)** tasarım desenine dayalı, **4 Katmanlı** bir mimari üzerine inşa edilecektir.

Şekil 2: Test Otomasyon Mimarisi (Katmanlı Yapı)



Şekil 2 İçin Açıklama: Test otomasyon projesi, sürdürülebilirlik ve modülerlik ilkeleri doğrultusunda 4 ana katman üzerine inşa edilmiştir. En üstte xUnit test senaryoları yer alırken, bu senaryolar iş mantığını Sayfa Objeleri (POM) katmanından çağrıır. Tüm sistem, .NET 8.0 ve Playwright altyapısı üzerinde koşan çekirdek bir çerçeve (Core Framework) tarafından yönetilir.

Mimari Katman Diyagramı Açıklaması:

1. Test Katmanı (Test Layer):

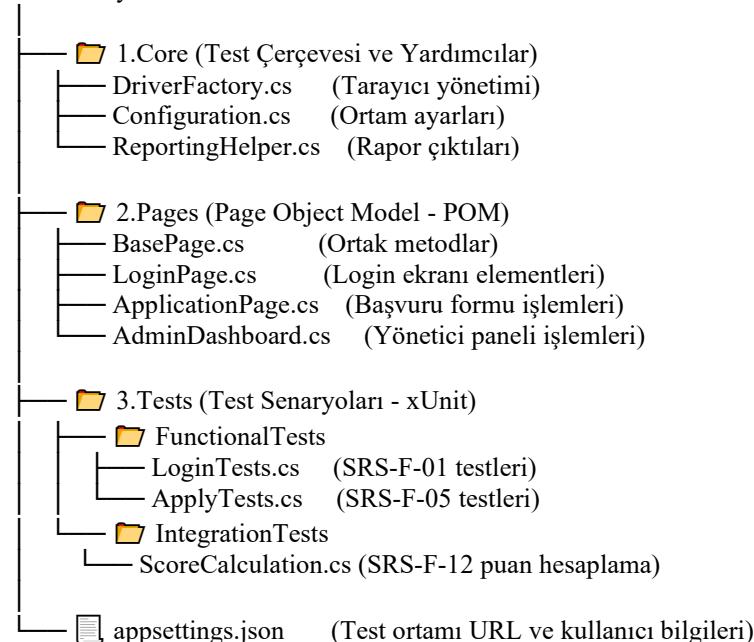
- Sadece test senaryolarını (Test Cases) barındırır.
- Assert (Doğrulama) işlemleri burada yapılır.
- Tarayıcı veya HTML elementleri ile doğrudan etkileşime girmez, sadece Framework katmanını çağrıır.
- *Örnek:* LoginTests.cs, BasvuruTests.cs

2. **İş Mantığı ve Sayfa Katmanı (Page Objects & Business Layer):**
 - Web sayfalarındaki elementlerin (Buton, Input vb.) tanımlandığı yerdir (Locators).
 - Sayfaya özgü metodlar burada yazılır (Örn: LoginPage.GirisYap(), BasvuruPage.BelgeYukle()).
 - SRS-NF-07 gereği, arayüz değiştirse sadece burası güncellenir.
3. **Çekirdek Çerçeve Katmanı (Core Framework Layer):**
 - Tüm projede ortak kullanılan yardımcı araçları içerir.
 - **BrowserFactory:** Tarayıcıyı (Chrome/Firefox) başlatan yapı.
 - **ConfigReader:** appsettings.json dosyasından test ortamı URL'ini okuyan yapı.
 - **DbHelper:** Test öncesi veritabanını temizleyen SQL komutları.
4. **Altyapı ve Veri Katmanı (Infrastructure & Data):**
 - **Bogus:** Test verisi üretimi.
 - **NSubstitute:** Dış servis simülasyonları.

2.3. Proje Çözüm (Solution) Yapısı

Visual Studio üzerinde oluşturulan proje hiyerarşisi aşağıdaki gibidir. Bu yapı, kodun modülerliğini sağlar.

ErasmusSystem.Test.sln



2.4. Kodlama ve Kalite Standartları

Projenin geliştirme sürecinde "Clean Code" (Temiz Kod) prensiplerine sadık kalınarak, kodun anlaşılabilirliğini ve bakım kolaylığını maksimize etmek için aşağıdaki standartlar katı bir şekilde uygulanacaktır:

2.4.1. Test Metodu İsimlendirme Stratejisi

Test raporlarında (Allure/Test Explorer) hatanın kaynağını kodu açmadan anlayabilmek için Roy Osherove'un önerdiği **[MethodName]/[State]/[ExpectedBehavior]** isimlendirme şablonu kullanılacaktır.

- **MethodName:** Test edilen metot veya iş birimi.
- **State:** Testin koşulduğu senaryo veya durum.
- **ExpectedBehavior:** Beklenen sonuç veya çıktı.
- *Örnek:* Login_InvalidCredentials_ShouldDisplayErrorMessage
- *Örnek:* CalculateScore_WithErasmusHistory_ShouldDeductTenPoints

2.4.2. Dil ve Terminoloji Birliği

Projenin uluslararası standartlara uyumu ve kütüphane tutarlılığı açısından:

- **Kod Dili:** Değişkenler, sınıf isimleri, metodlar ve yorum satırları %100 İngilizce olacaktır. Türkçe karakter (ğ, ü, ş, ı, ö, ç) kullanımından kaynaklı derleme veya encoding hatalarının önüne geçilecektir.
 - *Yanlış:* var basvuruTarihi = DateTime.Now;
 - *Doğru:* var applicationDate = DateTime.Now;
- **Git Commit Mesajları:** Versiyon kontrol geçmişinin takibi için "Semantic Commit" yapısına uygun İngilizce mesajlar kullanılacaktır. (Örn: feat: add playwright config, fix: login timeout issue).

2.4.3. Asenkron Programlama Modeli (TAP)

Playwright kütüphanesi doğası gereği asenkron (Non-blocking) çalıştığı için C# **Task-based Asynchronous Pattern (TAP)** eksiksiz uygulanacaktır:

- **Async/Await:** UI etkileşimi içeren tüm operasyonlar (Click, Type, Navigate) await anahtar kelimesi ile beklenecek, zincirleme işlem blokları oluşturulacaktır.
- **İsimlendirme:** Asenkron çalışan tüm metodların sonuna .NET konvensiyonu gereği Async eki getirilecektir.
 - *Örnek:* public async Task ClickSubmitButtonAsync()
- **Dönüş Tipi:** Asenkron test metodlarında void yerine mutlaka Task dönüş tipi kullanılarak, test koşucusunun (Runner) ilgili işlemin bitmesini beklemesi garanti altına alınacaktır.

2.4.4. Kod Düzeni ve Formatlama

- **Stil:** Visual Studio varsayılan C# kod stili (K&R style braces) kullanılacaktır.
- **Düzen:** Her "commit" öncesinde Ctrl+K, Ctrl+D kısayolu ile kod girintileri (indentation) ve boşluklar otomatik olarak düzenlenecektir.
- **Arrange-Act-Assert (AAA):** Test gövdeleri; Hazırlık (Arrange), Eylem (Act) ve Doğrulama (Assert) olmak üzere yorum satırlarıyla ayrılmış üç belirgin bloktan oluşacaktır.

2.5. Mimari Riskler ve Azaltma Stratejileri

Otomasyon projesinin sürdürülebilirliğini sağlamak ve olası darboğazları (bottlenecks) önceden engellemek amacıyla, tasarlanan mimari üzerinde aşağıdaki risk analizleri yapılmış ve önleyici tedbirler alınmıştır:

Risk 1: Teknoloji Adaptasyonu ve Öğrenme Eğrisi

- **Risk Tanımı:** Proje ekibinin Playwright for .NET kütüphanesine ve Asenkron Programlama (Async/Await) yapısına olan yatkınlığının geliştirme sürecinin başında düşük olması.
- **Olası Etki:** Test kodlarında tekrar eden yapıların (Code Duplication) oluşması, yanlış await kullanımı sonucu testlerin kilitlenmesi (Deadlock) veya spaghetti kod oluşumu.
- **Azaltma Stratejisi (Mitigation):**
 - **Soyutlama Katmanı (Abstraction Layer):** Proje başlangıcında oluşturulacak güçlü bir BasePage sınıfı ile Playwright'ın ham (raw) metodları sarmalanacaktır. Test yazan geliştirici, doğrudan karmaşık sürücü komutları yerine ClickButton(), EnterText() gibi basitleştirilmiş ve hata yönetimi yapılmış metodları kullanacaktır.
 - **Kod Gözden Geçirme (Code Review):** "Main" dalına (branch) yapılacak her birleştirme (merge) işlemi öncesinde, Pull Request'ler üzerinden kod standartlarına uyum (naming conventions, POM kullanımı) zorunlu tutulacaktır.

Risk 2: Mock Servislerin Gerçek Davranıştan Sapması (API Drift)

- **Risk Tanımı:** Dış servisleri (YÖK, E-Devlet) simüle etmek için kullanılan NSubstitute veya Mock nesnelerinin, zamanla gerçek servislerin güncellenen veri yapılarından (Schema) farklılaşması.
- **Olası Etki:** "False Positive" sonuçlar. Yani otomasyon testlerinin yerel ortamda başarılı geçmesi (Pass), ancak sistemin canlıya (Production) alındığında entegrasyon hatası vermesi.
- **Azaltma Stratejisi (Mitigation):**
 - **Contract Testing Prensibi:** Kritik entegrasyon noktaları için (Örn: Kimlik Doğrulama Servisi), servisin beklediği JSON şeması ile bizim gönderdiğimiz Mock objesinin şeması belirli aralıklarla karşılaştırılacaktır.
 - **Hibrit Test Yaklaşımı:** CI/CD hattında (Pipeline) testlerin %90'ı Mock servislerle koşulurken, en az 1 adet "Smoke Test" senaryosunun (kontrollü bir şekilde) gerçek test ortamı servislerine (Staging API) istek atması sağlanarak uçtan uca doğrulama yapılacaktır.

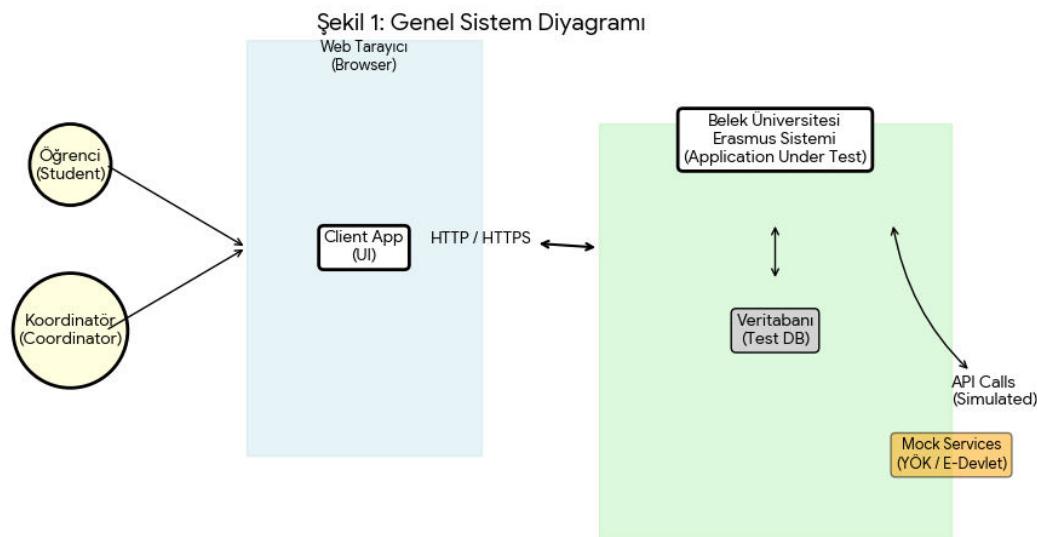
Risk 3: Test Kararsızlığı (Flaky Tests)

- **Risk Tanımı:** Web uygulamalarındaki ağ gecikmeleri veya JavaScript yükleme süreleri nedeniyle testlerin "bazen geçip bazen kalması" durumu.
- **Olası Etki:** Test sonuçlarına olan güvenin azalması ve sürekli hata ayıklama (debugging) ile zaman kaybı.
- **Azaltma Stratejisi (Mitigation):**
 - **Auto-Wait Mekanizması:** Kod içerisinde kesinlikle sabit beklemeler (Thread.Sleep) kullanılmayacak; bunun yerine Playwright'in elementin görünür ve tıklanabilir olmasını bekleyen yerlesik "Auto-wait" özelliği kullanılacaktır.

- **Retry (Yeniden Deneme) Politikası:** CI ortamındaki anlık ağ kopmalarını toler etmek için, başarısız olan testlerin raporlanmadan önce otomatik olarak 1 kez daha (Retry Count: 1) çalıştırılması yapılandırılacaktır.

3. Test Ortamı ve CI/CD Entegrasyonu

Yazılım Gereksinim Spesifikasyonu (SRS) dokümanında belirtilen **SRS-NF-09 (İzole Test Ortamı)** ve **SRS-NF-10 (Sürekli Entegrasyon)** maddelerini karşılamak üzere; testlerin yerel geliştirme ortamından üretim hattına kadar olan yaşam döngüsü aşağıdaki strateji ile yönetilecektir.



Şekil 1 İçin Açıklama: Yukarıdaki diyagramda, Öğrenci ve Koordinatör aktörlerinin sisteme Web Tarayıcısı üzerinden eriştiği, otomasyonun ise bu arayüzü kontrol ettiği görülmektedir. Sistemin dış bağımlılıkları (YÖK, E-Devlet), test ortamının izole kalması amacıyla sanallaştırılmış (Mock Services) servislerle temsil edilmiştir.

3.1. Lokal Koşum ve Hata Ayıklama (Local Execution)

Geliştirme aşamasında, test senaryolarının yazımı ve doğrulanması geliştiricinin yerel makinesinde (Localhost) gerçekleştirilecektir.

- **Headed Mode (Görsel Koşum):** Testler, tarayıcı arayüzü görünür şekilde (Headed: true) çalıştırılarak, UI üzerindeki etkileşimlerin (tıklama, yazma, scroll) insan gözüyle takibi sağlanacaktır.
- **Playwright Inspector:** Karmaşık senaryolarda veya hata alınan adımlarda, Playwright'ın "Inspector" aracı kullanılarak kod adım adım (Step-by-step) yürütülecek ve canlı DOM (Document Object Model) analizi yapılacaktır.
- **Visual Studio Entegrasyonu:** Testler, Visual Studio "Test Explorer" penceresi üzerinden tetiklenecek, anlık başarı/hata durumları IDE içerisinde takip edilecektir.

3.2. CI/CD Pipeline Mimarisi (Pipeline Execution)

Projenin kaynak kodları GitHub üzerinde barındırılacak ve otomasyon süreci **GitHub Actions** (veya Azure DevOps) kullanılarak yönetilecektir. Pipeline akışı şu adımlardan oluşur:

1. **Tetikleyiciler (Triggers):** main dalına yapılan her "Push" işlemi ve açılan her "Pull Request", test sürecini otomatik olarak başlatacaktır.
2. **Konteynerizasyon (Docker):** Testler, tutarlı bir ortam sağlamak amacıyla Microsoft'un resmi Playwright Docker imajı (mcr.microsoft.com/playwright/dotnet:v1.40.0) içerisinde koşulacaktır. Bu sayede "benim makinemde çalışıyordu" sorunu ortadan kaldırılacaktır.
3. **Headless Mode:** CI sunucusunda grafik arayüz bulunmadığı ve performans öncelikli olduğu için testler "Headless" (Arka planda) modda çalıştırılacaktır.
4. **Artifact Yönetimi:** Test koşumu tamamlandığında; oluşan HTML rapor dosyası ve hata alınan testlerin ekran görüntüleri (Screenshots) "Artifact" olarak ziplenip sunucuya yüklenecektir.

3.3. Konfigürasyon ve Paralel Koşum (Cross-Browser Testing)

Testlerin farklı tarayıcı motorlarında ve hızlı çalışmasını sağlamak için playwright.config.cs dosyası üzerinden aşağıdaki teknik ayarlar uygulanacaktır:

- **Çoklu Tarayıcı Desteği:** Testlerin tamamı, tek bir komutla hem **Chromium** (Google Chrome, Edge) hem de **Firefox** motorlarında eş zamanlı olarak test edilecektir.
- **Paralelizasyon (Parallelism):** Test süresini kısaltmak için FullyParallel = true ayarı aktif edilecek ve testler, CI sunucusunun işlemci çekirdek sayısına (Workers) göre bölünerek aynı anda koşulacaktır.
- **Retry (Yeniden Deneme) Mekanizması:** CI ortamındaki anlık ağ dalgalarından kaynaklı "False Negative" hataları önlemek için, başarısız olan testler otomatik olarak 1 kez daha (Retries = 1) tekrar edilecektir.

3.4. Veri Tabanı Test Stratejisi (Database Test Strategy)

Test otomasyonunun en büyük kırılganlık noktası olan "Veri Bağımlılığı" (Data Dependency) sorununu ortadan kaldırmak ve her testin izole bir ortamda koşulmasını garanti etmek için "**Test DB**" (Şekil 1) üzerinde aşağıdaki strateji uygulanacaktır:

3.4.1. Konteynerizasyon ve Geçici Ortam (Ephemeral Environment)

- **Teknoloji:** .NET ekosistemiyle tam uyumlu çalışan **Testcontainers** kütüphanesinin PostgreSQL modülü kullanılacaktır.
- **İşleyiş:** Test koşumu başladığında (Startup), Docker üzerinde anlık olarak "geçici" (ephemeral) bir **PostgreSQL** konteyneri ayağa kaldırılacaktır.
- **Avantaj:** Geliştiricilerin yerel makinelerindeki veritabanı kirliliği testleri etkilemeyecek; her test oturumu (Session) fabrikadan yeni çıkmış, %100 temiz bir veritabanı sunucusu üzerinde başlayacaktır.

3.4.2. Veri Tohumlama (Database Seeding)

- **Statik Referans Verileri:** Test senaryolarının çalışabilmesi için zorunlu olan "Sabit Veriler" (Örn: Üniversite Bölümüleri, Ülke Kodları, Yönetici Rol Tanımları), konteyner ayağa kalktıktan hemen sonra `DbHelper` sınıfı veya **EF Core (Npgsql)** Migrations aracılığıyla veritabanına basılacaktır.
- **Dinamik Test Verileri:** Her test metodu, ihtiyaç duyduğu spesifik veriyi (Örn: 3. sınıfı okuyan bir öğrenci) **Bogus** kütüphanesi ile çalışma anında (Runtime) üreticek ve veritabanına kaydedecektir. Böylece testler birbirinin verisine bağımlı kalmayacaktır.

3.4.3. Teardown ve Durum Sıfırlama (Resetting State)

- **Sorun:** Klasik silme işlemleri, ilişkisel tablolardaki kısıtlamalar nedeniyle karmaşıklık yaratabilir. PostgreSQL'de `TRUNCATE ... CASCADE` komutu kullanılsa da tüm tabloları sıfırlamak maliyetlidir.
- **Çözüm (Respawn):** Projede, veritabanını akıllıca temizleyen **Respawn** kütüphanesinin **PostgreSQL adaptörü** kullanılacaktır. Bu araç, tablolar arasındaki ilişkileri (Foreign Keys) otomatik algılayarak veriyi en hızlı şekilde temizler.
- **Süreç:** xUnit'in `IAsyncLifetime` arayüzü kullanılarak, her test metodu tamamlandığında (`DisposeAsync`), veritabanı tamamen silinmek yerine "Tohumlanmış" (Seeded) başlangıç noktasına geri döndürülecektir. Bu yöntem, veritabanını silip tekrar oluşturmaktan çok daha performanslıdır ve veri bütünlüğünü korur.

4. Sonuç

Bu raporla birlikte, projenin "Nasıl yapılacağı" sorusuna teknik olarak cevap verilmiş, yazılım mimarisi ve dosya yapısı netleştirilmiştir. Oluşturulan mimari, testlerin genişletilebilir (Scalable) ve bakımı kolay (Maintainable) olmasını garanti altına almaktadır.

HAFTA 5 İLERLEME RAPORU

Proje Adı: Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi

Rapor Adı: Hafta 5 Kullanıcı Senaryoları, Akış Diyagramları ve BDD Tasarımı

Proje Sorumlusu: Anıl Bilgehan YÜZGEÇ - 220007012

Danışman(lar): Dr. Adem ŞİMŞEK – Öğr. Gör. Soner DEDEOĞLU

Tarih: 16/01/2026

1. Özeti

Bu rapor, projenin 5. haftasında gerçekleştirilen görsel modelleme ve senaryo tasarım çalışmalarını kapsar. SRS dokümanında (Hafta 3) metin olarak tanımlanan işlevler, UML Use Case ve Activity diyagramları ile görselleştirilmiştir. Ayrıca, Hafta 4'te belirlenen Playwright otomasyon altyapısına girdi sağlayacak olan test senaryoları, Gherkin (Given-When-Then) formatında insan tarafından okunabilir (human-readable) bir yapıya kavuşturulmuştur. Bu çalışma, gereksinimlerin test koduna dönüştürülmesindeki belirsizlikleri ortadan kaldırılmayı amaçlar.

2. Hafta 5 Çıktıları: Senaryolar ve Akış Diyagramları

Projenin en karmaşık süreci olan "Erasmus Başvuru ve Değerlendirme" süreci için oluşturulan akış şeması, test otomasyonunun hangi adımları izleyeceğini belirler.

2.1. Kullanıcı Senaryoları ve Aktör Analizi (Use Case Design)

Hafta 3 raporunda listelenen 20 fonksiyonel gereksinimin (SRS-F-01 - SRS-F-20), tasarım aşamasında hangi Kullanım Durumu (Use Case) içerisinde ele alındığı aşağıdaki matris ile detaylandırılmıştır. Bazı mikro gereksinimler (Örn: GNO Validasyonu), ana süreçlerin (Örn: Başvuru Yap) bir alt adımı olarak modellenmiştir.

2.1.1. Aktör Tanımları ve Sorumluluklar

- Öğrenci (Primary Actor - Birincil Aktör):**
 - Tanım:** Sistemin ana son kullanıcısıdır. Süreçleri başlatan (Initiator) aktördür.
 - Sorumluluklar:** Sisteme güvenli giriş yapmak, kişisel bilgilerini doğrulamak, başvuru formunu eksiksiz doldurmak, kanıtlayıcı belgeleri (Transkript, Dil Belgesi) sisteme yüklemek ve başvuru sürecini takip etmektir.
 - İlgili Gereksinimler:** SRS-F-01 ... SRS-F-11 arası.
- Erasmus Koordinatörü / Admin (Primary Actor - Birincil Aktör):**
 - Tanım:** Sistemdeki karar verici mekanizmadır. Öğrenci tarafından başlatılan süreci sonlandıran aktördür.
 - Sorumluluklar:** Gelen başvuruları görüntülemek, belge geçerliliğini denetlemek, başvuruları "Onay/Ret/Revizyon" statülerine çekmek ve yönetimsel raporları almaktır.
 - İlgili Gereksinimler:** SRS-F-15 ... SRS-F-20 arası.
- Sistem / Zamanlayıcı (Time-Triggered Actor - Tetikleyici Aktör):**
 - Tanım:** İnsan müdahalesi olmadan, belirli olaylar veya zaman dilimleri sonucunda arka planda çalışan süreçlerdir.

- **Sorumluluklar:** Oturum süresi dolan kullanıcıları sistemden atmak (Session Timeout), başvuru puanlarını matematiksel formüle göre hesaplamak ve otomatik e-posta bildirimlerini tetiklemektir.
- **İlgili Gereksinimler:** SRS-F-02, SRS-F-12.

2.1.2. Use Case ve Gereksinim İzlenebilirlik/Eşleştirme Matrisi

Hafta 3 (SRS) raporunda belirlenen 20 adet fonksiyonel gereksinimin tamamı, aşağıda tanımlanan 8 temel Kullanım Durumu (Use Case) ile eşleştirilmiştir. Bu tablo, tasarımın hiçbir gereksinimi açıkta bırakmadığını kanıtlar.

Use Case ID	Senaryo Adı	Kapsadığı SRS Maddeleri	Detay / Kapsam Açıklaması
UC-01	Oturum Açıma (Login)	SRS-F-01 (Öğrenci Login) SRS-F-15 (Admin Login)	Hem öğrenci hem koordinatör için güvenli giriş ve hata mesajı (Validasyon) süreçlerini kapsar.
UC-02	Profil Yönetimi	SRS-F-03 (Profil Güncellemeye)	Öğrencinin kişisel bilgilerini ve salt okunur (TC Kimlik) alanlarını görüntülemesi.
UC-03	Erasmus Başvurusu Yap	SRS-F-04 (Üni. Filtreleme) SRS-F-05 (Başvuru Oluşturma) SRS-F-06 (Belge Yükleme) SRS-F-07 (Zorunlu Alan) SRS-F-08 (Edit Yetkisi) SRS-F-09 (Draft/Submit) SRS-F-11 (GNO Kontrolü)	Bu "Ana Senaryo"dur. Üniversite seçiminden, form validasyonlarına (GNO, Boş alan) ve belge yüklemeye kadar olan tüm başvuru adımlarını içerir.
UC-04	Başvuru Durum Takibi	SRS-F-10 (Durum İzleme)	Öğrencinin "Gönderildi", "Değerlendiriliyor", "Onaylandı" statülerini panelinden görmesi.
UC-05	Puan Hesaplama	SRS-F-12 (Formül) SRS-F-13 (Şehit/Gazi) SRS-F-14 (Hibe Kesintisi)	Sistem aktörü tarafından tetiklenen; matematiksel formül, ceza ve ek puan mantıklarının işletildiği arka plan sürecidir.
UC-06	Başvuru Değerlendirme	SRS-F-16 (Listeleme) SRS-F-17 (Önizleme) SRS-F-18 (Onay/Ret) SRS-F-19 (Ret Gerekçesi)	Koordinatörün listeyi görmesi, belgeyi açması ve ret durumunda zorunlu açıklama girmesi döngüsünü kapsar.
UC-07	İstatistik Raporlama	SRS-F-20 (Excel Raporu)	Koordinatörün sistemdeki verileri Excel formatında dışarı aktarması.
UC-08	Oturum Kontrolü	SRS-F-02 (Timeout)	Kullanıcının 20 dk hareketsiz kalması durumunda otomatik çıkış yapılması (Sistem Aktörü).

Hazırlanan Kullanım Durumu (Use Case) diyagramları, Hafta 3 raporunda belirlenen 20 adet fonksiyonel gereksinim ile çapraz kontrolden geçirilmiş ve açıkta kalan herhangi bir gereksinim olmadığı doğrulanmıştır (Gereksinim-Tasarım Tutarlılığı).

2.1.3. Fonksiyonel Olmayan Gereksinimlerin Tasarıma Etkisi

Hafta 3 raporunda belirlenen "Fonksiyonel Olmayan Gereksinimler" (SRS-NF-XX), tek bir senaryoya bağlı kalmaksızın sistemin genel kalite standartlarını belirler. Bu gereksinimlerin test tasarımdaki karşılıkları aşağıdadır:

NFR Kategorisi	İlgili SRS Maddeleri	Etkilediği Senaryolar	Test Yaklaşımı
Performans	SRS-NF-02, SRS-NF-03	Tüm Senaryolar (Özellikle UC-01 Login ve UC-03 Başvuru)	Yük testi (k6) ile yanıt sürelerinin <3 sn olduğu doğrulanacaktır.
Kullanılabilirlik	SRS-NF-01	Tüm Arayüzler	"Breadcrumb" ve menü yapısının en fazla 3 tık kuralına uyduğu UC-02 sırasında kontrol edilecektir.
Uyumluluk	SRS-NF-04, SRS-NF-09	Tüm Senaryolar	Test otomasyonu, aynı senaryoları hem Chrome hem Firefox üzerinde koşacaktır.
Güvenlik	SRS-NF-08	UC-01 (Login) ve UC-05 (Değerlendirme)	Yetkisiz erişim denemeleri ve veri maskeleme (Data Masking) kontrol edilecektir.

2.2. Aktivite ve Akış Diyagramları (User Flows)

Test otomasyonunun kapsamını netleştirmek amacıyla, sistemin en kritik iş süreci olan "Erasmus Başvuru Akışı", **Mutlu Yol (Happy Path)** ve olası **Sınır Durumları (Edge Cases)** içerecek şekilde detaylanmıştır. Bu akış, Hafta 4'te tasarlanan ApplicationPage.cs sınıfının otomasyon senaryolarına doğrudan girdi sağlayacaktır.

Akış 1: Erasmus Başvuru Süreci (Öğrenci Odaklı)

Bu süreç, öğrencinin sisteme giriş yapmasından başvuruyu nihai olarak göndermesine kadar geçen adımları ve sistemin verdiği tepkileri tanımlar.

- **Adım 1: Başlangıç ve Erişim Kontrolü**
 - **Eylem:** Öğrenci sisteme kullanıcı adı ve şifresiyle giriş yapar.
 - **Sistem Kontrolü:** Kullanıcının rolü "Öğrenci" mi?
 - **Edge Case:** Eğer rolü "Pasif Öğrenci" ise sistem başvuru menüsünü gizler.
- **Adım 2: Başvuru Dönemi Kontrolü (SRS-F-05)**
 - **Sistem Kontrolü:** Sunucu saatı, tanımlı başvuru tarihleri (Start Date - End Date) arasında mı?
 - **Happy Path (Evet):** "Yeni Başvuru Oluştur" butonu aktif hale gelir.
 - **Edge Case (Hayır):** Buton pasif (disabled) olur ve ekranında "*Şu an aktif bir başvuru dönemi bulunmamaktadır*" uyarı mesajı görüntülenir. Akış burada sonlanır.
- **Adım 3: Form Doldurma ve Veri Doğrulama**
 - **Eylem:** Öğrenci; GNO, Gidilecek Ülke, Tercih Edilen Üniversite alanlarını doldurur.
 - **Sistem Kontrolü (Validasyon):** GNO alanı 0.00 - 4.00 aralığında mı?

- **Edge Case:** Öğrenci "4.50" veya "ABC" girerse, input alanı kırmızıya döner ve "Geçerli bir not ortalaması giriniz" hatası fırlatılır.
- **Adım 4: Belge Yükleme Kararı (SRS-F-06)**
 - **Eylem:** Öğrenci "Transkript Yükle" butonuna basarak dosya seçer.
 - **Happy Path:** Dosya .pdf formatında ve <5MB boyutundadır. Yükleme barı %100 olur ve "Yüklandı" ikonu belirir.
 - **Edge Case:** Öğrenci .exe dosyası veya 10MB boyutunda dosya yüklemeye çalışır. Sistem "Sadece PDF formatı ve maks. 5MB dosya kabul edilmektedir" uyarısını döndürür.
- **Adım 5: Zorunlu Alan Kontrolü ve Gönderim**
 - **Sistem Kontrolü:** "İletişim Bilgileri" dolu mu VE "Transkript" yüklü mü?
 - **Happy Path (Evet):** "Başvuruyu Tamamla ve Gönder" butonu aktif (Clickable) hale gelir. Öğrenci tıklar.
 - **Edge Case (Hayır):** Buton pasif kalır (Disabled). Tıklanmaya çalışılırsa eksik alanlar vurgulanır (Highlight).
- **Adım 6: Bitiş ve Durum Güncelleme**
 - **Sistem İşlemi:** Veritabanındaki başvuru kaydının statüsü STATUS: DRAFT'tan STATUS: SUBMITTED'a çekilir.
 - **Sonuç:** Ekranda "Başvurunuz başarıyla alınmıştır" modal penceresi açılır ve öğrenciye otomatik bilgilendirme e-postası gönderilir.
 - **Kısıt:** Artık bu başvuru üzerinde "Düzenle" (Edit) butonu kaybolur.

2.3. BDD Test Senaryoları ve Gherkin Tasarımı

Hafta 4 raporunda belirlenen teknoloji yiğinına (Playwright & xUnit) uygun olarak, test senaryoları **Davranış Odaklı Geliştirme (BDD)** prensipleriyle tasarlanmıştır. Bu senaryolar, hem teknik ekip hem de paydaşlar tarafından anlaşılabılır **Gherkin (Given-When-Then)** formatında yazılmıştır.

Senaryolar; **Pozitif (Happy Path)**, **Negatif (Edge Case)** ve **Veri Odaklı (Scenario Outline)** olmak üzere üç farklı kategoride detaylandırılmıştır.

Senaryo 1: Uçtan Uca (E2E) Başvuru Süreci (Happy Path)

- **Amaç:** Bir öğrencinin sorunsuz bir şekilde başvuru yapabildiğini doğrulamak.
- **İlgili SRS Maddeleri:** SRS-F-05, SRS-F-06, SRS-F-09

Gherkin

Feature: Erasmus Öğrenci Başvuru Süreci

Öğrenciler, aktif başvuru dönemlerinde sisteme belge yükleyerek başvuru yapabilmelidir.

Background:

```
Given "Ahmet" isminde bir öğrenci sisteme giriş yapmıştır
And Sistemde aktif bir "2025-Güz" başvuru dönemi tanımlıdır
```

Scenario: Eksiksiz belgelerle geçerli başvuru yapılması

```
When Öğrenci "Yeni Başvuru" sayfasına gider
And GNO alanına "3.15" değerini girer
And Tercih edilen üniversite olarak "Berlin Technical University" seçer
And Transkript belgesini ("transkript.pdf") yükler
And "Başvuruyu Tamamla" butonuna tıklar
Then Sistem "Başvurunuz başarıyla alındı" mesajını göstermelidir
```

And Veritabanında başvuru durumu "SUBMITTED" olarak güncellenmelidir
And Öğrenciye otomatik bilgilendirme e-postası gönderilmelidir

Senaryo 2: Puan Hesaplama Motoru (Scenario Outline / Veri Odaklı)

- **Amaç:** Farklı not ve geçmiş durumlara göre puan hesaplamasının doğruluğunu tek bir senaryo çatısı altında test etmek.
- **İlgili SRS Maddeleri:** SRS-F-12 (Formül), SRS-F-13 (Şehit/Gazi), SRS-F-14 (Ceza)
- **Hesaplama Yöntemi:** $(GNO * 25 * 0.50) + (Dil\ Puanı * 0.50) + (Ek\ Puan) - (Ceza)$

Gherkin

Feature: Erasmus Puanlama Motoru

Sistem, YÖK kurallarına ve üniversite yönetmeliğine göre puanı otomatik hesaplamalıdır.

Scenario Outline: Farklı öğrenci profillerine göre puan hesaplama kontrolü

Given Öğrencinin GNO'su <gno> ve Yabancı Dil Puanı <dil_puanı>'dır

And Öğrencinin Şehit/Gazi yakını durumu: <sehit_gazi>

And Öğrencinin geçmiş Erasmus durumu: <gecmis_erasmus>

When Sistem "Puan Hesapla" fonksiyonunu çalıştırıldığında

Then Hesaplanan toplam puan <beklenen_puan> olmalıdır

Examples:

Gno	Dil_puanı	Sehit-gazi	Geçmiş_Erasmus	Beklenen_Puan	Acıklamama
3.00	80	Hayır	Var	67.5	$(75*0.5)+(80*0.5)-10 = 67.5$
3.00	80	Hayır	Yok	77.5	$(75*0.5)+(80*0.5) = 77.5$
2.50	60	Evet	Yok	76.25	$(62.5*0.5)+(60*0.5)+15= 76.25$
4.00	100	Hayır	Yok	100.0	Tam Puan Kontrolü

Senaryo 3: Hatalı Dosya Yükleme Kontrolü (Negative Test)

- **Amaç:** Sistemin geçersiz dosya formatlarına karşı dayanıklılığını test etmek.
- **İlgili SRS Maddeleri:** SRS-F-06 (Format Kontrolü)

Gherkin

Feature: Belge Yükleme Validasyonları

Scenario: İzin verilmeyen formatta dosya yüklenmesi engellenmeli

Given Öğrenci "Belge Yükleme" ekranındadır

When Transkript alanına "virus.exe" isimli bir dosya yüklemeye çalışır

Then Sistem "Geçersiz dosya formatı! Sadece PDF yüklenebilir." hatası vermelidir

And "Kaydet" butonu pasif (disabled) kalmalıdır

And Dosya sunucuya yüklenmemelidir

Senaryo 4: Koordinatör Ret İşlemi (İş Kuralı)

- Amaç:** Bir başvuru reddedildiğinde gerekçe girilmesinin zorunlu olduğunu doğrulamak.
- İlgili SRS Maddeleri:** SRS-F-19 (Ret Gerekçesi)

Gherkin

Feature: Başvuru Değerlendirme Süreci

Scenario: Açıklama girilmeden ret işlemi yapılamamalıdır. Given Koordinatör "Bekleyen Başvurular" listesinden bir öğrenci seçmiştir When Durum olarak "REDDEDİLDİ" seçeneğini işaretler And Açıklama alanını boş bırakarak "Güncelle" butonuna tıklar Then Sistem "Lütfen ret gerekçesini giriniz" uyarısını göstermelidir And Başvuru durumu veritabanında değişimmemelidir

3. Mimari ile Entegrasyon ve Tasarımın Doğrulanması

Bu hafta oluşturulan Kullanıcı Senaryoları (Use Cases) ve Akış Diyagramları (Activity Diagrams), Hafta 4 raporunda belirlenen **Page Object Model (POM)** mimarisinin sağlığını yapmak için kullanılmıştır. Tasarımın koda dökülebilirliği (Feasibility) aşağıdaki analiz başlıklarıyla doğrulanmıştır:

3.1. Tasarım ve Kod Eşleştirme Matrisi

Aşağıdaki tablo, görselleştirilen iş süreçlerinin yazılım mimarisindeki tam karşılıklarını ve test edileceği katmanları göstermektedir:

Tasarım Bileşeni (Akış/Use Case)	POM Karşılığı (Class & Method)	Test Katmanı Karşılığı (Test Class)
Giriş ve Yetkilendirme Akışı <i>(Şekil 1: UC-01)</i>	Sınıf: LoginPage.cs Metot: LoginAsync(username, password) Element: TxtEmail, BtnSubmit	Test: LoginTests.cs <i>(Senaryo: Login_InvalidUser_ShouldFail)</i>
Başvuru Formu ve Belge Yükleme <i>(Şekil 2: Karar Noktaları)</i>	Sınıf: ApplicationPage.cs Metot: UploadTranscriptAsync(path) Metot: SubmitApplicationAsync()	Test: ApplyTests.cs <i>(Senaryo: Apply_WithMissingDocs_BtnDisabled)</i>
Yönetici Değerlendirme Süreci <i>(Şekil 1: UC-05)</i>	Sınıf: AdminDashboard.cs Metot: RejectApplicationAsync(reason) Metot: FilterByDepartment(dept)	Test: EvaluationTests.cs <i>(Senaryo: Reject_WithoutReason_ShouldWarn)</i>

Tasarım Bileşeni (Akış/Use Case)	POM Karşılığı (Class & Method)	Test Katmanı Karşılığı (Test Class)
Puan Hesaplama Motoru <i>(Arka Plan Süreci)</i>	Sınıf: ScoreCalculator.cs (Helper) Metot: CalculateErasmusScore(gno, lang)	Test: ScoreCalculation.cs <i>(Türü: Unit Test / xUnit Theory)</i>

3.2. Detaylı Entegrasyon Analizi

- Kimlik Doğrulama Katmanı (Authentication):**

Akış diyagramındaki "Sisteme Giriş Yap" adımı, POM mimarisindeki LoginPage sınıfı tarafından kapsüllemiştir (Encapsulation). Test senaryoları, doğrudan HTML elementlerine (ID, CSS Selector) erişmek yerine, bu sınıfın sunduğu LoginAsync servisini kullanarak giriş işlemini simüle edecektir.

- Dosya Yönetimi ve Validasyon (File Handling):**

Şekil 2'de (Açıklama) belirtilen "Zorunlu Alan Kontrolü" ve "Dosya Boyutu Kontrolü" kararları, ApplicationPage sınıfı içerisindeki mantıksal kontrollerle eşleşmektedir. Playwright'ın dosya yükleme olaylarını (File Chooser Event) dinleyen yapısı, bu akıştaki "Edge Case" senaryolarını (Örn: .exe dosyası yükleme) test etmek için kullanılacaktır.

- İş Mantığı Ayrımı (Separation of Concerns):**

Puan hesaplama süreci (UC-04), görsel bir arayüzden bağımsız matematiksel bir işlem olduğu için UI testlerinden ayrılmıştır. Bu mantık, 1.Core katmanındaki ScoreCalculator yardımcı sınıfına (Helper) taşınarak, tarayıcı açılmadan (Unit Test seviyesinde) milisaniyeler içinde test edilmesi sağlanmıştır. Bu yaklaşım, SRS-F-12 gereksiniminin en performanslı şekilde test edilmesini garanti eder.

- Yönetim Paneli Etkileşimi:**

Koordinatörün "Ret Gerekçesi Girme" senaryosu (UC-05), AdminDashboard sayfa objesi üzerinden yönetilecektir. Bu sınıf, tablolar (Data Grid) üzerindeki satırları okuma ve modal pencereleri kontrol etme yetenekleriyle donatılacaktır.

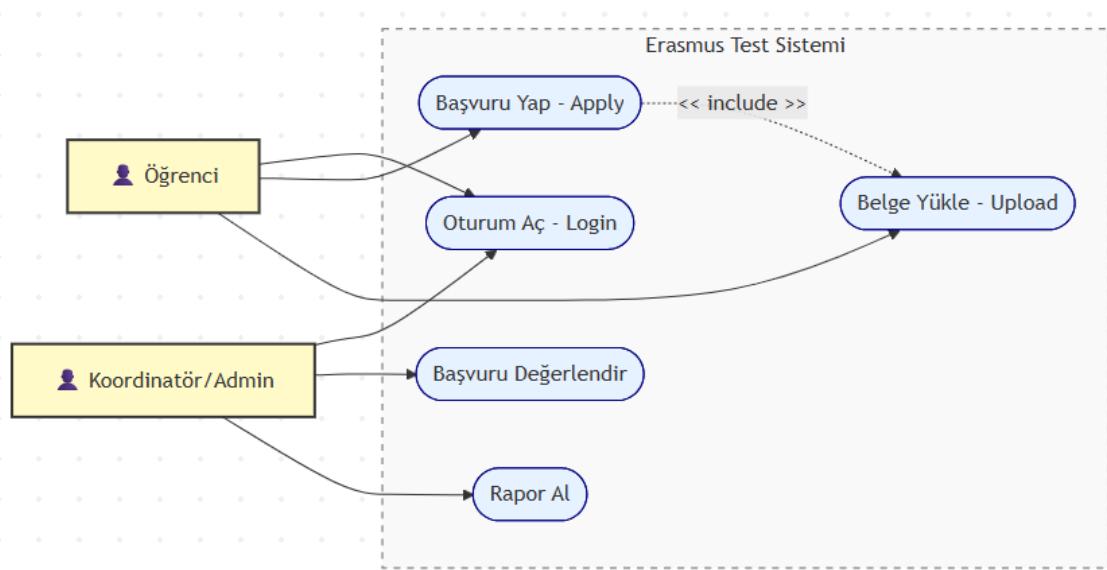
4. Sonuç

Hafta 5 kapsamında, projenin "ne yapacağı" (Requirements) ile "nasıl yapacağı" (Architecture) arasındaki köprü olan Tasarım ve Senaryo aşaması tamamlanmıştır. Görselleştirilen akışlar sayesinde, test kodlarının yazımına başlandığında mantıksal hataların (Logic Errors) en aza indirilmesi hedeflenmiştir.

EKLER

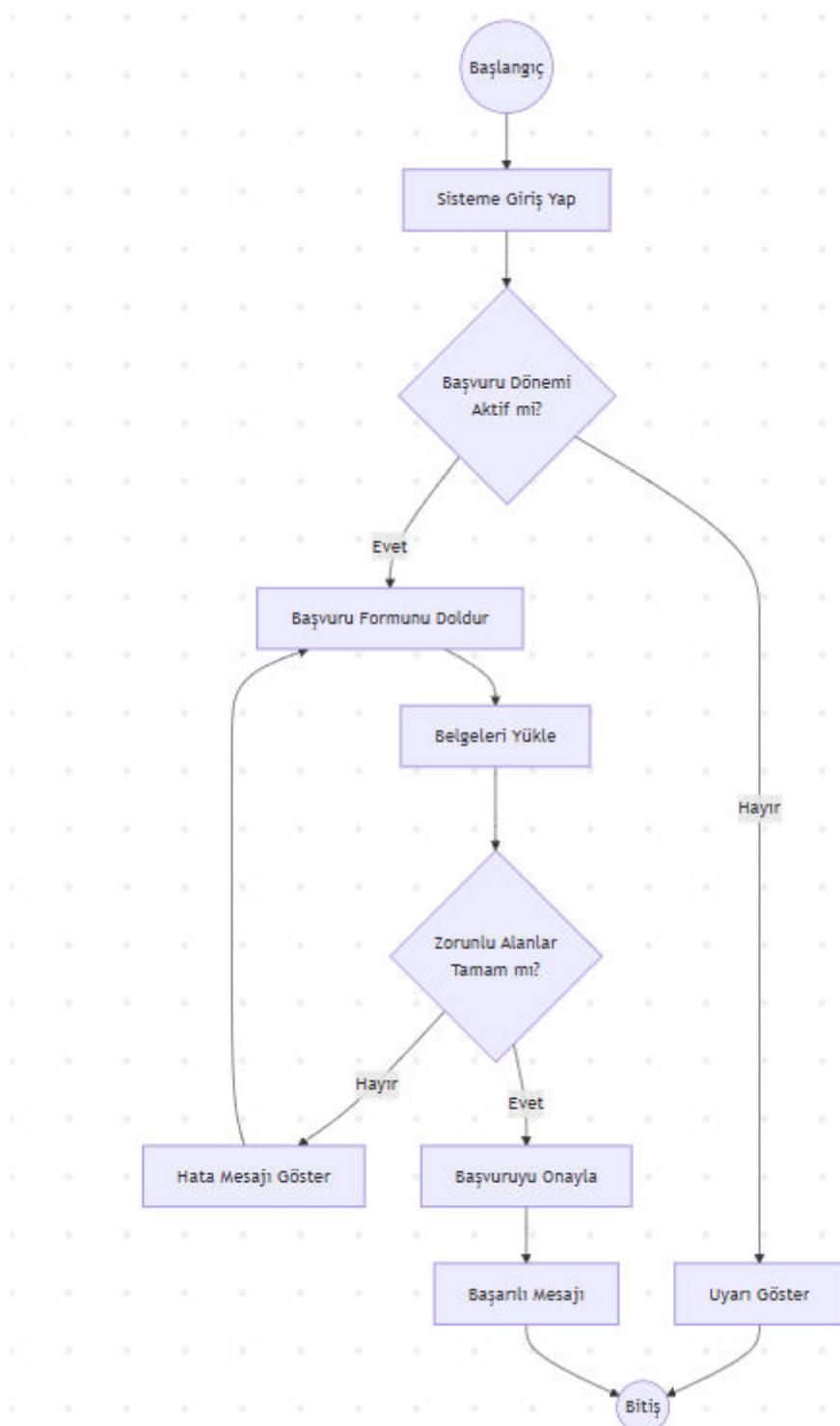
Aşağıda raporda atıfta bulunulan ve raporu destekleyici diyagramlar bulunmaktadır.

EK – A: Erasmus Sistemi Use Case (Kullanım Durumu) Diyagramı



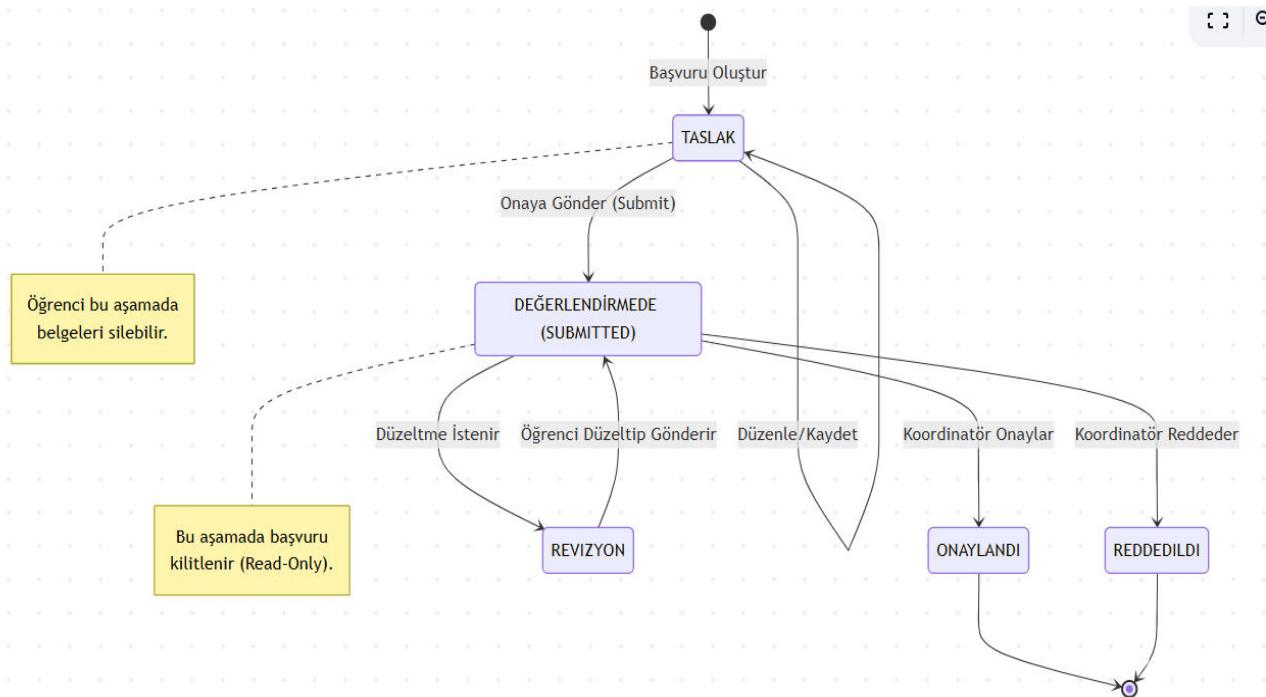
“Sistem sınırları (System Boundary) içerisinde tanımlanan kullanım durumları, Öğrenci ve Koordinatör aktörlerinin yetki alanlarını belirler. Örneğin, 'Başvuru Değerlendir' senaryosu sadece Koordinatör erişimine açıkken, 'Belge Yükle' senaryosu 'Başvuru Yap' sürecinin zorunlu bir parçasıdır (<<include>>).”

EK – B: Öğrenci Başvuru Süreci Aktivite (Akış) Diyagramı



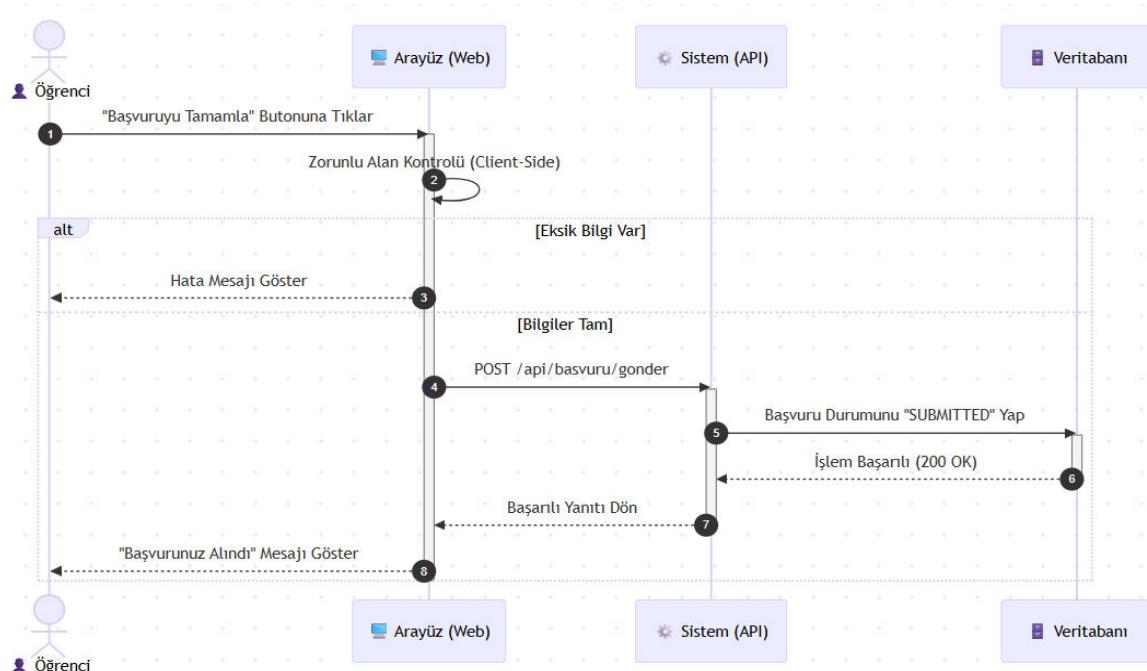
Başvuru süreci, doğrusal bir akış yerine karar mekanizmalarını (Decision Nodes) içerir. Diyagramda görüldüğü üzere, 'Başvuru Dönemi Aktif mi?' kontrolü 'Hayır' dönerse süreç başlamadan sonlanır; bu da otomasyon testlerindeki 'Negatif Test' senaryolarının temelini oluşturur.

EK – C: Başvuru Durum Geçiş Diyagramı (State Transition)



"Bu diyagram, SRS-F-09 gereksinimi uyarınca bir Erasmus başvurusunun yaşam döngüsünü modeller. Test otomasyonu, geçersiz durum geçişlerini (Örn: 'Reddedildi' statüsünden 'Taslak' statüsüne dönüş) engelleyip engellemediğini bu modele göre doğrulayacaktır."

EK – D: Sıralılık Diyagramı (Sequence Diagram)



"Öğrenci, Arayüz (UI) ve Sistem (Backend) arasındaki mesajlaşma trafiğini gösterir. Entegrasyon testleri (Integration Tests) bu akıştaki veri alışverişi doğrulamak için kurgulanacaktır."

HAFTA 6 İLERLEME RAPORU

Proje Adı: Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi

Rapor Adı: Hafta 6 Veritabanı Tasarımı ve Veri Yönetim Stratejisi

Proje Sorumlusu: Anıl Bilgehan YÜZGEÇ - 220007012

Danışman(lar): Dr. Adem ŞİMŞEK – Öğr. Gör. Soner DEDEOĞLU

Tarih: 16/01/2026

1. Özeti

Bu rapor, projenin 6. haftasında gerçekleştirilen **Veritabanı Tasarımı (Database Design)** çalışmalarını kapsar. Hafta 4'te belirlenen **PostgreSQL** teknolojisi ve **Docker/Testcontainers** altyapısı üzerinde çalışacak olan ilişkisel veri şeması (Relational Schema) oluşturulmuştur.

Erasmus başvuru sürecinin veri gereksinimlerini karşılamak ve test otomasyonunda kullanılacak "Test Verisi"nin bütünlüğünü sağlamak amacıyla; Varlık-Bağıntı (ER) diyagramları çizilmiş, 3. Normal Form (3NF) kurallarına uygun tablo yapıları ve veri sözlüğü (Data Dictionary) hazırlanmıştır.

2. Hafta 6 Çıktıları: Veritabanı Tasarımı

2.1. Kavramsal Tasarım ve Varlık Analizi (Conceptual Design)

Hafta 5'te oluşturulan Kullanıcı Senaryoları (Use Cases) ve SRS dokümanındaki iş kuralları analiz edilerek, sistemin veri bütünlüğünü sağlayacak temel varlıklar (Entities) ve sorumlulukları aşağıdaki gibi belirlenmiştir:

1. Users (Kullanıcılar ve Rol Yönetimi)

- Tanım:** Sisteme erişim sağlayan tüm aktörlerin (Öğrenci ve Koordinatör) kimlik ve yetki bilgilerini tutan merkezi varlıktır.
- İşlevi:** SRS-F-01 ve SRS-F-15 gereksinimleri uyarınca, tek bir tablo üzerinden hem öğrenci hem de yönetici girişi yönetilir. Role sütunu sayesinde yetkilendirme (Authorization) yapılır.
- Kritik Veri:** Şifreler (PasswordHash), Rol Bilgisi, Öğrenci Not Ortalaması (GNO).

2. Applications (Erasmus Başvuru Yönetimi)

- Tanım:** Projenin merkezinde yer alan, bir öğrencinin belirli bir dönemde yaptığı başvurunun yaşam döngüsünü temsil eden varlıktır.
- İşlevi:** SRS-F-05 gereksinimi olan başvuru oluşturma işleminin ana kaydıdır. Hesaplanan toplam puan (TotalScore) ve başvurunun anlık durumu (Status) burada tutulur.
- İlişki:** Başvuru tek başına bir üniversiteye bağlı değildir; bunun yerine ApplicationPreferences varlığı üzerinden birden fazla üniversite tercihi ile ilişkilendirilir.

3. Universities (Anlaşmalı Kurumlar Havuzu)

- **Tanım:** Üniversitenin Erasmus anlaşması bulunan dış kurumların statik listesidir.
- **İşlevi:** SRS-F-04 gereksinimi kapsamında, öğrencilerin başvuru sırasında seçim yapabileceği "Hedef Okul" listesini sağlar.
- **Kritik Veri:** Üniversitenin statik bilgilerini (Ad, Ülke) tutar. Değişken olan kontenjan bilgileri, veri bütünlüğünü korumak için **UniversityTermQuotas** varlığına ayrılmıştır.

4. Documents (Kanıtlayıcı Belgeler)

- **Tanım:** Başvuru sürecinde yüklenen dosyaların (Transkript, Dil Belgesi) fiziksel yollarını ve türlerini tutan varlıktır.
- **İşlevi:** SRS-F-06 gereksinimi gereği, dosya doğrulama işlemleri için meta verileri saklar. Veritabanı şıklığını önlemek için dosyanın kendisi (BLOB) yerine sunucudaki yolu (FilePath) saklanır.
- **İlişki:** Bir başvuruya birden fazla belge eklenebilir (1-to-Many).

5. ApplicationHistory (Denetim İzi ve Loglama)

- **Tanım:** Başvuru durumunda (Örn: Taslak -> Gönderildi -> Onaylandı) meydana gelen her değişikliğin kaydını tutan tarihçe tablosudur.
- **İşlevi:** SRS-F-10 (Durum Takibi) ve SRS-F-19 (Ret Gerekçesi) gereksinimleri için kritiktir. Bir başvurunun kim tarafından, ne zaman ve neden reddedildiği veya onaylandığı geriye dönük olarak izlenebilir (Audit Log).

(*Varlık-Bağıntı Diyagramı /ER Diagram/ EK A olarak raporun sonunda sunulmuştur.*)

"Sistemdeki Kullanım Durumlarının (Use Cases) hangi veri tablolarını etkilediğini ve veri bütünlüğünün nasıl sağlandığını gösteren analiz için bkz. EK C: CRUD Matrisi."

2.2. Fiziksel Şema ve Veri Sözlüğü (Physical Schema)

Projenin veri katmanı, veri bütünlüğünü (Data Integrity) ve ilişkisel tutarlılığı garanti altına almak amacıyla PostgreSQL üzerinde 3. Normal Form (3NF) kurallarına uygun olarak tasarlanmıştır.

Aşağıdaki tablolarda, her bir sütunun veri tipi, zorunluluk durumu ve iş kurallarını (Business Rules) denetleyen kısıtlamaları (Constraints) detaylandırılmıştır.

- **Tablo 1: belek_erasmus.users (Kullanıcılar)**
- *Sisteme giriş yapacak Öğrenci ve Koordinatörlerin merkezi kaydıdır.*

Sütun Adı	Veri Tipi	Kısıtlama (Constraint)	Açıklama
id	UUID	PK , Not Null, Default gen_random_uuid()	Tahmin edilemez benzersiz kullanıcı kimliği.
first_name	VARCHAR(50)	Not Null	Kullanıcının adı.

Sütun Adı	Veri Tipi	Kısıtlama (Constraint)	Açıklama
last_name	VARCHAR(50)	Not Null	Kullanıcının soyadı.
email	VARCHAR(100)	Unique Index, Not Null	Sistem giriş anahtarı. Aynı e-posta ile ikinci kayıt açılamaz.
password_hash	VARCHAR(255)	Not Null	Güvenlik gereği şifreler "Hash"lenerek saklanır (Argon2/BCrypt).
role	VARCHAR(20)	Check (role IN ('Student', 'Coordinator'))	Kullanıcının yetki seviyesini belirler.
gno	DECIMAL(3,2)	Check (gno BETWEEN 0.00 AND 4.00)	Öğrenci Not Ortalaması. (Koordinatörler için NULL olabilir).
is_erasmus_before	BOOLEAN	Default FALSE	Daha önce hibe alıysa puan kesintisi (-10) yapmak için kullanılır.

- **Tablo 2: belek_erasmus.applications (Başvurular)**
- *Öğrencinin Erasmus başvuru sürecine ait ana verileri ve durum bilgisini tutar.*

Sütun Adı	Veri Tipi	Kısıtlama (Constraint)	Açıklama
id	UUID	PK, Not Null	Benzersiz başvuru numarası.
user_id	UUID	FK (Users.Id) ON DELETE CASCADE	Başvuruyu yapan öğrenci. Öğrenci silinirse başvurusu da silinir.
term	VARCHAR(20)	Not Null (Örn: '2025-Spring')	Başvurunun yapıldığı akademik dönem.
status	VARCHAR(20)	Default 'DRAFT' (Enum: DRAFT, SUBMITTED, APPROVED, REJECTED)	Süreç durumu.
total_score	DECIMAL(5,2)	Nullable, Check (>= 0)	Dil puanı ve GNO ile hesaplanan nihai Erasmus puanı.
created_at	TIMESTAMP	Default CURRENT_TIMESTAMP	Başvuru oluşturulma zamanı (Başvuru dönemi kontrolü için).
updated_at	TIMESTAMP	Nullable	Başvuru üzerinde yapılan son değişiklik zamanı.
(Tablo Geneli)	-	Unique Index (user_id, term)	Bir öğrenci aynı dönem için sadece bir başvuru yapabilir.

- **Tablo 3: belek_erasmus.universities (Anlaşmalı Kurumlar)**
- *Öğrencilerin seçim yapabileceği hedef üniversiteler listesidir (Lookup Table).*

Sütun Adı	Veri Tipi	Kısıtlama (Constraint)	Açıklama
id	UUID	PK , Not Null	Üniversite kimliği.
name	VARCHAR(100)	Not Null	Üniversitenin resmi adı (Örn: Berlin Technical University).
country_code	CHAR(2)	Not Null (ISO 3166)	Ülke kodu (Örn: DE, FR, PL).

- **Tablo 4: belek_erasmus.documents (Başvuru Belgeleri)**
- *Başvurulara eklenen dosyaların meta verilerini tutar.*

Sütun Adı	Veri Tipi	Kısıtlama (Constraint)	Açıklama
id	UUID	PK , Not Null	Belge kimliği.
app_id	UUID	FK (Applications.Id)	Hangi başvuruya ait olduğu.
doc_type	VARCHAR(50)	CHECK (doc_type IN ('Transcript', 'LanguageCert', 'CV'))	Belge türü (Örn: 'Transcript', 'LanguageCert').
file_path	VARCHAR(255)	Not Null	Sunucudaki fiziksel dosya yolu (Veritabanı şıkkınılığını önlemek için BLOB kullanılmamıştır).
uploaded_at	TIMESTAMP	Default CURRENT_TIMESTAMP	Yükleme zamanı.

- **Tablo 5: belek_erasmus.application_history**
- *Bu tablo, başvurunun tarihçesini tutar. Öğrenci "Başvurum neden reddedildi?" dediğinde veya sistem yönetici "Bu başvuruyu kim onayladı?" dediğinde buraya bakılır.*

Sütun Adı	Veri Tipi	Kısıtlama (Constraint)	Açıklama
id	UUID	PK , Not Null	Log kayıt numarası.
app_id	UUID	FK (Applications.Id)	Hangi başvurunun tarihçesi olduğu.
old_status	VARCHAR(20)	Nullable	Önceki durum (Örn: DRAFT).
new_status	VARCHAR(20)	Not Null	Yeni durum (Örn: SUBMITTED).

Sütun Adı	Veri Tipi	Kısıtlama (Constraint)	Açıklama
changed_by	UUID	FK (Users.Id)	Değişikliği yapan kişi (Öğrenci veya Koordinatör).
reason	VARCHAR(255)	Nullable	Eğer reddedildiyse ret gerekçesi (SRS-F-19).
changed_at	TIMESTAMP	Default Now()	İşlem zamanı.

- **Tablo 6: belek_erasmus.application_preferences (Başvuru Tercihleri)**
- *Öğrencilerin birden fazla (genellikle 3) üniversite tercihi yapabilmesini sağlar.*

Sütun Adı	Veri Tipi	Kısıtlama (Constraint)	Açıklama
id	UUID	PK , Not Null	Tercih satır kimliği.
app_id	UUID	FK (Applications.Id) ON DELETE CASCADE	İlgili başvuru kaydı.
uni_id	UUID	FK (Universities.Id)	Tercih edilen üniversite.
rank	INT	Check (Rank BETWEEN 1 AND 3)	Tercih sırası (1., 2. veya 3. tercih).
(Tablo Geneli)	-	Unique Index (app_id, rank)	Aynı başvuruda aynı sıra numarası (Örn: 1. Tercih) tekrar edemez.

- **Tablo 7: belek_erasmus.university_term_quotas (Dönemsel Kontenjanlar)**
- *Her akademik dönem için (2025-Spring, 2026-Fall) üniversite kontenjanlarının ayrı ayrı yönetilmesini ve geçmiş dönem verilerinin bozulmamasını sağlar.*

Sütun Adı	Veri Tipi	Kısıtlama (Constraint)	Açıklama
id	UUID	PK , Not Null	Kontenjan kayıt kimliği.
uni_id	UUID	FK (Universities.Id)	Hangi üniversite.

Sütun Adı	Veri Tipi	Kısıtlama (Constraint)	Açıklama
term	VARCHAR(20)	Not Null	Hangi dönem için geçerli olduğu (Örn: '2025-Fall').
quota	INT	Check (> 0)	O dönem için ayrılan kontenjan sayısı.

2.3. Normalizasyon ve Tasarım Kararları

Projenin veritabanı şeması, **Veri Bütünlüğü** (Data Integrity) ile **Sorgu Performansı** (Query Performance) arasındaki en optimum dengeyi sağlayan **3. Normal Form (3NF)** standardına göre tasarlanmıştır.

Tasarımın 3NF uyumluluğu şu kriterlerle sağlanmıştır:

- 1NF (Atomiklik):** Tüm tablolarda (Örn: Users) her hücre tek bir veri parçası tutmaktadır. Ad ve Soyad ayrı sütunlara bölünmüştür; "Hobiler" gibi çoklu değer içeren alanlar bulunmamaktadır.
- 2NF (Tam Bağımlılık):** Tüm tabloların birincil anahtarı (Primary Key - UUID) mevcuttur ve tüm sütunlar doğrudan bu anahtara bağlıdır.
- 3NF (Geçişli Bağımlılığın Kaldırılması):** Tablolarda, anahtar olmayan bir sütunun başka bir anahtar olmayan sütuna bağımlılığı ortadan kaldırılmıştır.
 - Uygulama:* Veri tekrarını önlemek ve güncelleme anomalilerinden kaçınmak için; üniversitede ait statik bilgiler (Ad, Ülke) **ApplicationPreferences** tablosunda tekrar edilmemiş, UniId üzerinden referans verilmiştir. Ayrıca, bir üniversitenin kontenjan bilgisi sadece üniversitede değil, aynı zamanda "Akademik Dönem"e de bağlı olduğu için (Geçişli Bağımlılık); kontenjan verisi **Universities** tablosunda tutulmayıp, **UniversityTermQuotas** adlı ayrı bir tabloya ayırtılarak 3NF kuralına tam uyum sağlanmıştır.

Tasarım Tercihi (Design Choice): Projenin birincil amacı "Test Otomasyonu Mimarisi" kurmak olduğu için, veri katmanında YAGNI (You Aren't Gonna Need It) prensibi benimsenmiştir. Test verisi yönetimini (Test Data Management) zorlaştıracak ve sorgu performansını düşürecek aşırı normalizasyondan (Over-normalization) kaçınılmış; 5 temel tablo ile tüm fonksiyonel gereksinimler karşılanmıştır.

Tasarım Notu (Denormalizasyon Kararı): Normalizasyon kurallarına istisna olarak, Applications tablosundaki TotalScore alanı, teknik olarak hesaplanabilir (Derived) bir veridir. Ancak her sorguda (Listeleme, Sıralama) tekrar tekrar hesaplama maliyeti yaratmamak adına, bu alan hesaplanıp veritabanına **fiziksel olarak kaydedilmiştir (Stored)**. Bu, performans odaklı bilinçli bir **Kontrollü Denormalizasyon** tercihidir.

"Sistemdeki Kullanım Durumlarının (Use Cases) hangi veri tablolarını etkilediğini ve veri bütünlüğünün nasıl sağlandığını gösteren analiz için bkz. EK C: CRUD Matrisi."

3. Test Verisi Yönetim Stratejisi

Test otomasyonunda **"Tekrarlanabilirlik" (Repeatability)** ilkesini sağlamak için, "Canlı" (Production) ortam stratejisinden farklı olarak, **Hibrit Veri Yönetimi** modeli benimsenmiştir. Bu model, sabit referans verileri ile dinamik işlem verilerini birbirinden ayırrır.

3.1. Statik Veri Tohumlama (Reference Data Seeding)

Uygulamanın çalışması için zorunlu olan ve test süresince değişmemesi gereken "Tanım Verileri" (Lookup Data), test sunucusu ayağa kalktığında (Global Setup) bir kez oluşturulacaktır.

- Amaç:** Test senaryolarının her defasında üniversite veya ülke yaratmakla vakit kaybetmesini önlemek.
- Kapsam:**
 - Universities:** 5 adet global üniversite (Örn: "Berlin Tech", "Sorbonne", "Politecnico di Milano").
 - Users (Admin):** Sistemi yönetecek 1 adet "Süper Koordinatör" hesabı.
 - UniversityTermQuotas:** Tanımlanan 5 üniversite için, aktif dönem (Örn: 2025-Spring) kontenjanları da test başlangıcında veritabanına eklenmelidir.
- Yöntem:** `DbHelper.SeedReferenceData()` metodu, konteyner başlatıldığında çalışacak ve eğer tablolar boşsa bu sabit verileri `INSERT` edecektir.

3.2. Dinamik Veri Üretilimi (Dynamic Mocking via Bogus)

Her test senaryosu, kendi izole verisini yaratmaktan sorumludur. "Hepsini Paylaş" (Shared Database) anti-pattern'inden kaçınmak için, öğrenci ve başvuru kayıtları çalışma anında (Runtime) rastgele üretilecektir.

- Teknoloji:** .NET ekosistemi için standart olan **Bogus** kütüphanesi kullanılacaktır.
- Avantajı:** Her testte farklı isimler, e-postalar ve GNO değerleri üretilerek "Pesticide Paradox" (Sürekli aynı veriyle test yapıp yeni hataları bulamama durumu) engellenir.

- Uygulama Örneği:**

C#

```
// Kurallar: E-posta benzersiz olmalı, GNO 0-4 arasında olmalı.  
var studentFaker = new Faker<User>()  
    .RuleFor(u => u.FirstName, f => f.Name.FirstName())  
    .RuleFor(u => u.Email, f => f.Internet.Email())  
    .RuleFor(u => u.GNO, f => f.Random.Decimal(0.00m, 4.00m))  
    .RuleFor(u => u.Role, "Student");  
  
var testUser = studentFaker.Generate(); // Veritabanına basılmaya  
hazır nesne
```

3.3. Veri Temizliği ve Durum Sıfırlama (State Reset Strategy)

Testler arasında veri kirliliğini (Data Pollution) önlemek için "**Checkpoint**" (**Kontrol Noktası**) stratejisi uygulanacaktır.

- **Sorun:** Klasik `DROP DATABASE` işlemi çok yavaştır. `TRUNCATE` işlemi ise Foreign Key (Yabancı Anahtar) ilişkileri yüzünden karmaşıktır.
- **Çözüm (Respawn):** Hafta 4 raporunda belirlenen **Respawn** kütüphanesi kullanılacaktır. Bu araç, veritabanının "Temiz" alanında bir kontrol noktası (Checkpoint) alır.
- **İşleyiş:**
 1. Test başlamadan önce Statik Veriler (Üniversiteler) yüklenir -> **Checkpoint alınır**.
 2. Test koşulları (Öğrenci eklenir, başvuru yapılır).
 3. Test biter (`TearDown`).
 4. Respawn, sadece sonradan eklenen verileri (Öğrenci, Başvuru) siler; ancak Üniversite tablosuna dokunmaz.
- **Konfigürasyon:**

C#

```
// Üniversiteler tablosu temizlikten muaf tutulur (Ignored)
await _checkpoint.ResetAsync(_connectionString, tablesToIgnore: new[]
{ "Universities" });
```

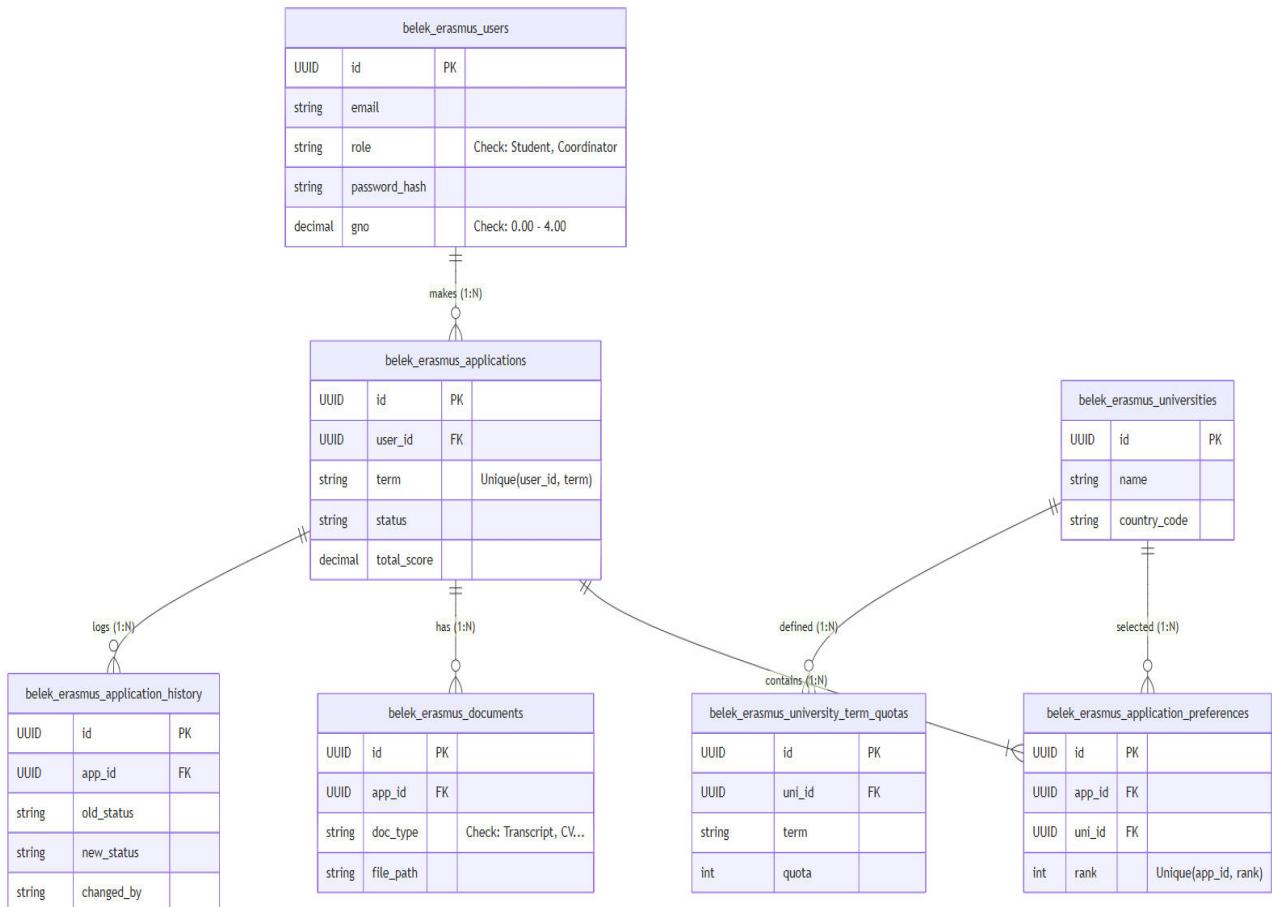
4. Sonuç

Hafta 6 çalışmaları birlikte, yazılımın veri iskeleti oluşturulmuş ve SRS dokümanındaki iş kurallarını (Puan hesaplama, Başvuru kısıtları) destekleyecek veri yapıları hazırlanmıştır. Bu tasarım, gelecek haftalarda yapılacak test kodlarının (Backend Tests) temelini oluşturacaktır.

EKLER

Aşağıda raporda atıfta bulunulan ve raporu destekleyici diyagramlar bulunmaktadır.

EK – A: Varlık-Bağlantı Diyagramı [ER Diagram]



EK – B: Fonksiyon - Veri Varlığı İlişkisi (CRUD Matrisi)

Aşağıdaki matris, sistemdeki temel Kullanım Durumlarının (Use Cases), veritabanı tabloları üzerinde gerçekleştirdiği fiziksel operasyonları (Create, Read, Update, Delete) göstermektedir. Bu harita, test otomasyonunda "Hangi işlemden sonra hangi tablo kontrol edilmeli?" sorusuna yanıt verir.

(Kısaltmalar: **C**: Create/Oluştur, **R**: Read/Oku, **U**: Update/Güncelle, **D**: Delete/Sil)

Use Case (Senaryo)	Users	Applications	Documents	Universities	History
Oturum Açma	R	-	-	-	-
Başvuru Yap	R	C	C	R	C
Belge Yükle	-	U	C	-	-
Başvuru Değerlendir	-	U	R	R	C
Rapor Al	R	R	-	R	R

Matrisin Teknik Analizi ve Test Odaklı Yorumu:

1. Oturum Açma (Login):

- Sadece Users tablosunda **Okuma (R)** işlemi yapar.
- Test Doğrulaması:* Login işlemi yapıldığında veritabanında herhangi bir yeni kayıt oluşmamalı veya veri değişimmemelidir. Test sadece "Giriş Başarılı mı?" yanıtını kontrol eder.

2. Başvuru Yap (Apply):

- En karmaşık işlemdir.** Yeni bir başvuru (Applications) ve belge (Documents) Oluşturur (**C**).
- Aynı zamanda Universities tablosunu **Okur (R)** (Seçilen okulun ID'sini almak için).
- Kritik nokta: Başvuru ilk oluşturulduğunda History tablosuna da "DRAFT" statüsüyle bir **Log kaydı (C)** atılmalıdır. Test kodu bunu mutlaka doğrulamalıdır.

3. Belge Yükle (Upload):

- Documents tablosuna yeni satır ekler (**C**).
- Ancak Applications tablosunu sadece **Günceller (U)** (UpdatedAt alanını değiştirir).
- Test Doğrulaması:* Belge yüklenliğinde Applications tablosundaki satır sayısı artmamalıdır, sabit kalmalıdır.

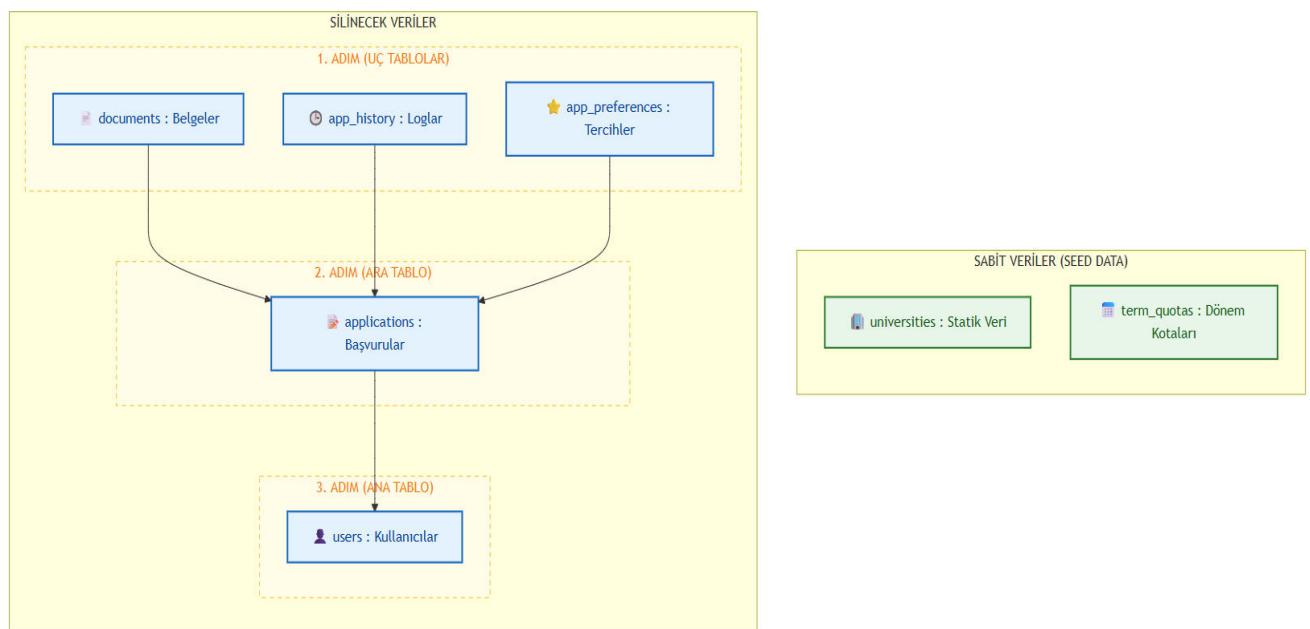
4. Başvuru Değerlendir (Evaluate):

- Koordinatör başvuruyu onayladığında; Applications tablosundaki Status kolonu **Güncellenir (U)**.
- Aynı anda, bu değişikliğin kim tarafından yapıldığını takip etmek için History tablosuna yeni bir kayıt **Eklenir (C)**.
- *Test Stratejisi:* Eğer Applications güncellenip History tablosuna kayıt atılmazsa test "**Fail**" (Başarısız) olmalıdır. (Denetim izi kaybı).

5. Rapor Al (Report):

- Sistemdeki hiçbir veriyi değiştirmez. Tüm tablolarda sadece **Okuma (R)** yetkisine sahiptir. Bu işlem "Güvenli İşlem" (Idempotent) kategorisindedir.

EK – C: Test Verisi Temizlik ve Bağımlılık Şeması



“Bu şema, **Respawn** kütüphanesinin her test senaryosunun bitiminde (Teardown phase) veritabanını temizlerken izleyeceği **Topolojik Sıralamayı (Topological Sort)** gösterir.

*İlişkisel veritabanı bütünlüğü gereği, **Referans Bütünlüğü (Referential Integrity)** hatası almak için silme işlemi; hiçbir tabloya referans vermeyen "uç tablolardan" (Child Tables: **Documents, History, ApplicationPreferences**) başlayarak, referans alınan "merkez tablolara" (Parent Tables: **Applications, Users**) doğru ilerleyecektir.”*

HAFTA 7 İLERLEME RAPORU

Proje Adı: Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi

Rapor Adı: Hafta 7 Arayüz (UI/UX) Tasarımı, Wirframe ve Mockup Tasarımları

Proje Sorumlusu: Anıl Bilgehan YÜZGEÇ - 220007012

Danışman(lar): Dr. Adem ŞİMŞEK – Öğr. Gör. Soner DEDEOĞLU

Tarih: 16/01/2026

1. Özeti

Projenin 7. haftasında, yazılımın son kullanıcıyla etkileşime gireceği arayüz katmanı (Frontend) için **UI/UX Tasarım Süreci** tamamlanmış ve kodlamaya geçilmiştir.

Hafta 5'teki "Kullanıcı Akışları" (User Flows) temel alınarak; önce kağıt üzerinde taslaklar (**Wireframes**), ardından Mermaid.js kullanılarak renkli ve etkileşimli modeller (**Mockups**) oluşturulmuştur. Bu tasarımlar, özellikle **UI Test Otomasyonu** (Playwright for .NET) için gerekli olan HTML element seçeneklerinin (ID, Name, XPath) belirlenmesinde referans alınmıştır.

2. Hafta 7 Çıktıları: Arayüz Tasarımı (UI/UX)

Sistemin kullanılabilirliğini artırmak ve test senaryolarını somutlaştırmak için 3 aşamalı bir tasarım süreci izlenmiştir.

2.1. Taslak Çizimleri ve Yerleşim Kararları (Mid-Fidelity Wireframes)

Kullanıcı arayüzünün kodlanması gerekliliklerini ve fonksiyonel yerlesimi doğrulamak amacıyla düşük çözünürlüklü iskelet tasarımları (Wireframes) oluşturulmuştur. Bu tasarımlarda renk ve tipografi yerine; **form elemanlarının konumu, navigasyon akışı ve geri bildirim alanları** (hata mesajları) önceliklendirilmiştir.

Tasarım kararları, **Playwright for .NET** ile yazılacak UI testlerinin kararlılığını (Stability) artıracak şekilde "sade ve erişilebilir" yapıda kurgulanmıştır.

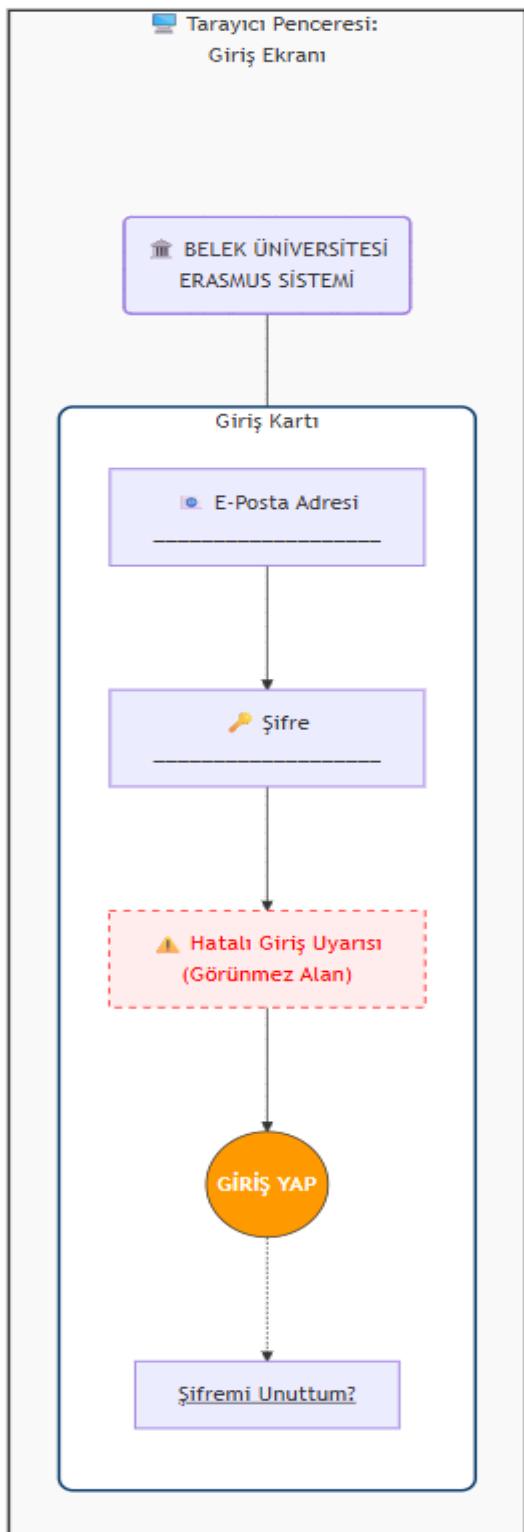
Not: Arayüz taslakları (Wireframes), yazılım mimarisile bütünlük olması ve yazılım geliştirme süreçlerine uygun olarak, versiyonlanabilir ve dinamik güncellenebilir olması amacıyla **Mermaid.js** kullanılarak 'Diagram-as-Code' yaklaşımıyla şematik olarak oluşturulmuştur.

A. Giriş Ekranı (Login Interface)

- Tasarım Kararı:** Kullanıcının odağını dağıtmamak için tam sayfa görsel yerine, ekranın tam ortasında yer alan "**Kart Yapısı**" (**Card Layout**) tercih edilmiştir.
- Fonksiyonellik:**
 - Input Alanları:** E-posta ve Şifre alanları alt alta ve standart genişlikte konumlandırılmıştır.

- **Validasyon Alanı:** Hatalı giriş yapıldığında (SRS-F-01), kullanıcıya gösterilecek uyarı mesajı (Alert Box) butonun hemen üzerinde rezerve edilmiştir. Bu, "Negatif Login Testi" için kritik bir doğrulama noktasıdır.
- **Aksiyonlar:** "Giriş Yap" (Primary Action) ve "Şifremi Unuttum" (Secondary Action) butonları.

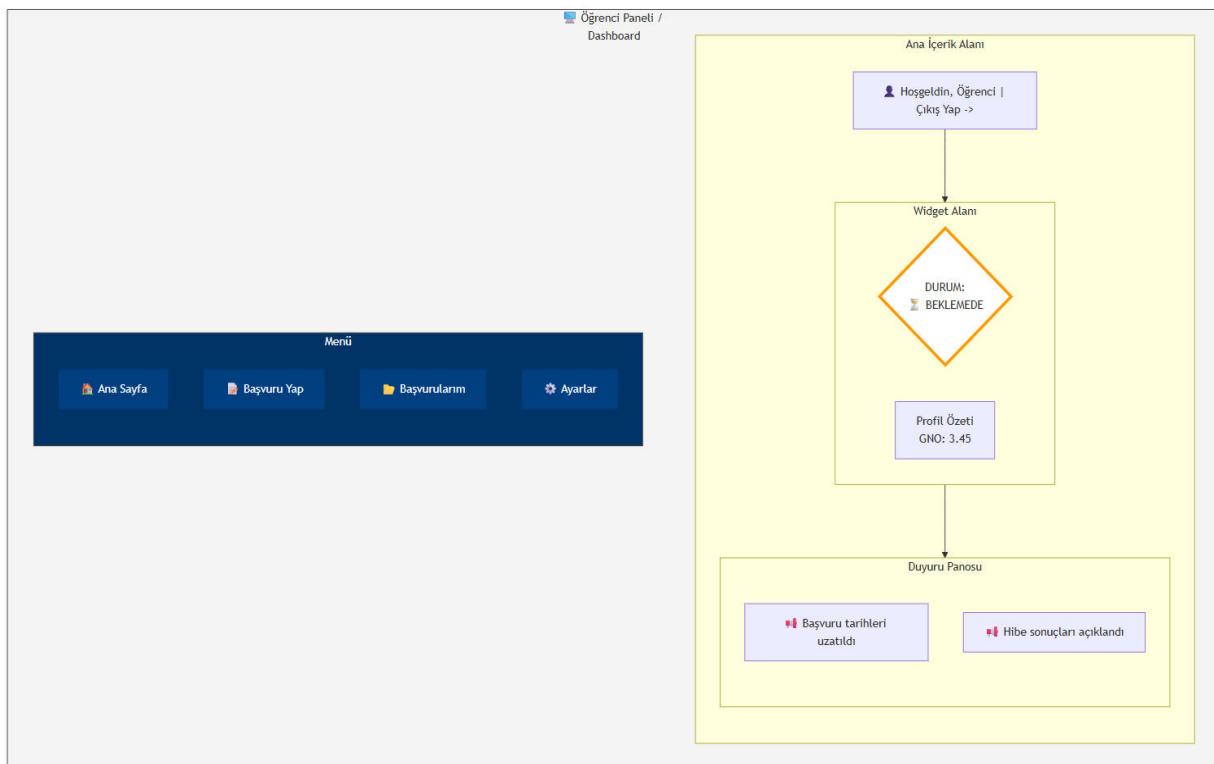
Şekil 1.1: Giriş Ekranı Taslağı



B. Öğrenci Paneli (Dashboard Layout)

- Tasarım Kararı:** "F Tipi Okuma" düzenine uygun olarak, ana navigasyon **Sol Dikey Menü (Sidebar)** olarak tasarlanmıştır. Bu yapı, menünün test otomasyonunda XPath ile bulunmasını kolaylaştırır.
- İçerik:**
 - Durum Kartı (Status Widget):** Öğrencinin aktif başvurusunun durumu (Draft, Submitted, Rejected) ekranın en üstünde, büyük puntolarla gösterilir. Test senaryoları "Başvuru sonrası durum değişti mi?" kontrolünü buradan yapacaktır.
 - Profil Özeti:** Sağ üst köşede kullanıcının adı ve çıkış butonu yer alır.

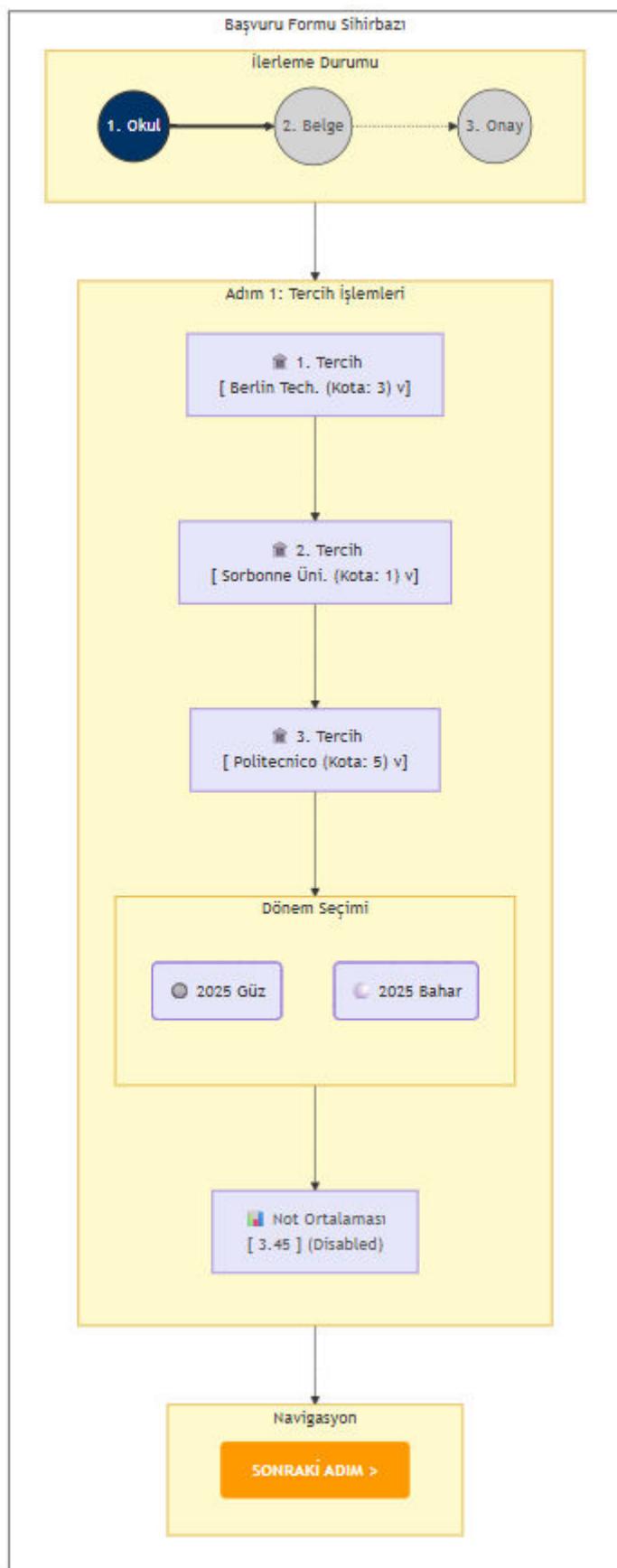
Şekil 1.2: Dashboard Taslağı



C. Başvuru Formu (Multi-Step Wizard)

- Tasarım Kararı:** Hafta 6 veritabanı tasarımındaki Applications tablosunun yoğun veri içermesi nedeniyle, tek uzun bir form yerine "**Adımlı Sihirbaz**" (**Stepper**) yapısı kullanılmıştır.
- Adımlar:**
 - Tercih:** Üniversite ve Dönem seçimi (Dropdown).
 - Belgeler:** Transkript yükleme alanı (File Upload Input).
 - Özet:** Girilen bilgilerin son kontrolü (Read-Only View).
- Test Kolaylığı:** Her adının ayrı bir URL'si veya `div` bloğu olması, otomasyonun "İleri" butonuna basıp bir sonraki adının yüklediğini doğrulamasını (Wait Condition) kolaylaştırır.

Şekil 1.3: Başvuru Sihirbazı Taslağı



2.2. Görsel Tasarım ve Tasarım Sistemi (High-Fidelity Mockups)

Wireframe aşamasında doğrulanın yerleşim planı; Belek Üniversitesi'nin kurumsal kimliği ve modern web standartları (Material Design) referans alınarak nihai görsel tasarıma dönüştürülmüştür. Bu aşamada, **Figma** tasarım aracı kullanılarak yazılımın "Pixel-Perfect" (Piksel Hassasiyeti) prototipleri hazırlanmıştır.

Tasarım süreci 4 ana eksende detaylandırılmıştır:

A. Renk Paleti ve Psikolojisi (Color Theory)

Arayüzde kullanılan renkler, kullanıcıda "Güven" duygusu uyandırmak ve önemli aksiyonları (CTA) vurgulamak amacıyla stratejik olarak seçilmiştir. Ayrıca renklerin **WCAG 2.1 (Web Content Accessibility Guidelines)** standartlarına göre kontrast oranları test edilmiştir.

- **Ana Renk (Primary - #003366):** Belek Üniversitesi'nin kurumsal Lacivert rengidir. "Header" (Üst Bilgi) ve "Sidebar" (Yan Menü) alanlarında kullanılarak kurumsal ciddiyet ve güven hissi verir.
- **Vurgu Rengi (Accent - #FF9900):** Kontrast oluşturan Turuncu renk. Sadece "Başvuruya Tamamla" veya "Giriş Yap" gibi **Eylem Butonlarında (Buttons)** kullanılarak kullanıcının dikkatini işlem noktasına çeker.
- **Durum Renkleri (System States):**
 - **Yeşil (#28a745):** "Başvuru Onaylandı" durumu.
 - **Kırmızı (#dc3545):** "Hatalı Giriş" ve "Ret" durumu.
 - **Gri (#f8f9fa):** Arka plan (Background) rengi olarak kullanılarak göz yorgunluğu önlenmiştir.

B. Tipografi ve Hiyerarşi (Typography)

Okunabilirliği maksimize etmek için ekran çözünürlüğüne göre optimize edilmiş **Sans-Serif** font ailesi tercih edilmiştir.

- **Font Ailesi:** Roboto veya Open Sans (Google Fonts).
- **Hiyerarşi:**
 - **H1 (24px - Bold):** Sayfa Başlıkları (Örn: "Yeni Başvuru Formu").
 - **H2 (18px - SemiBold):** Kart Başlıkları (Örn: "Kişisel Bilgiler").
 - **Body (14px - Regular):** Form etiketleri ve metin içerikleri.

C. UI Bileşen Kütüphanesi (Component Library)

Yazılımın kodlama aşamasında (Frontend) tekrar kullanılabiliğin sağlanması ve **Bootstrap 5** Grid sistemine uyum sağlamak için bileşenler modüler tasarılmıştır.

- Kartlar (Cards):** İçerikler, gölgelendirilmiş (box-shadow) beyaz kutular içinde sunularak arka plandan ayırtırılmıştır.
- Form Elemanları:** Input alanları, kullanıcı tıkladığında (Focus State) mavi bir çerçeve ile parlayarak aktif olduğunu belli eder.
- Butonlar:** Köşeleri hafif yuvarlatılmış (Border-Radius: 4px) yapıda tasarlanmıştır.

D. Responsive (Duyarlı) Davranış

Arayüzün farklı cihazlarda nasıl tepki vereceği simülle edilmiştir:

- Desktop (>1200px):** Yan menü (Sidebar) açık ve sabittir.
- Tablet/Mobil (<768px):** Yan menü otomatik olarak gizlenir ve sol üst köşede "Hamburger Menü" ikonu belirir. Tablolar (DataTables) yatay kaydırma (Scroll) moduna geçer.

Tablo: Proje Renk Kodları ve Kullanım Alanları

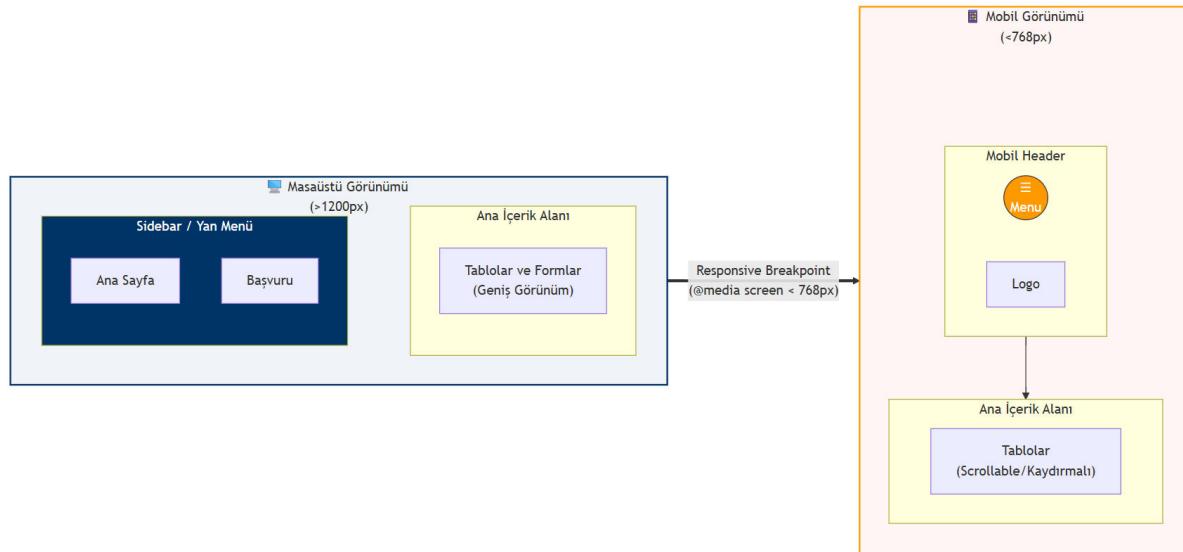
Aşağıdaki tablo, uygulamanın arayüzünde (UI) kullanılan renk paletini, bu renklerin psikolojik etkisini ve **WCAG 2.1** (Web Erişilebilirliği) standartlarına uygunluğunu detaylandırmaktadır.

Renk Kodu (Hex)	Renk Adı	Kullanım Alanı ve İşlevi	Erişilebilirlik (Kontrast)
#003366	Kurumsal Lacivert	Ana Renk (Primary): Header, Sidebar menü ve başlıklar. <i>Etkisi:</i> Kurumsallık ve güven hissi verir.	<input checked="" type="checkbox"/> AA Uyumlu (Beyaz metin ile)
#FF9900	Aksiyon Turuncu	Vurgu Rengi (Accent): "Başvur" ve "Giriş Yap" butonları. <i>Etkisi:</i> Kullanıcının dikkatini eyleme çeker (Call-to-Action).	<input checked="" type="checkbox"/> AA Uyumlu (Siyah metin ile)
#F4F6F9	Duman Grisi	Arka Plan (Background): Sayfa gövdesi ve boş alanlar.	- (Metin rengi değil)

Renk Kodu (Hex)	Renk Adı	Kullanım Alanı ve İşlevi	Erişilebilirlik (Kontrast)
		<i>Etkisi:</i> Göz yorgunluğunu öner ve içeriği öne çıkarır.	
#28A745	Başarı Yeşili	Durum (Success): "Başvuru Onaylandı" mesajları. <i>Etkisi:</i> İşlemin hatasız tamamlandığını bildirir.	<input checked="" type="checkbox"/> AA Uyumlu (Beyaz metin ile)
#DC3545	Hata Kırmızısı	Durum (Error): Validasyon hataları ve "Sil" butonları. <i>Etkisi:</i> Kritik uyarı ve tehlikeli işlem vurgusu.	<input checked="" type="checkbox"/> AA Uyumlu (Beyaz metin ile)
#343A40	Metin Grisi	İçerik (Body Text): Form etiketleri ve paragraflar. <i>Etkisi:</i> Saf siyah (#000) yerine kullanılarak okuma konforu artırılır.	<input checked="" type="checkbox"/> AAA Uyumlu (Gri zemin ile)

Not: Seçilen renk paleti, *Playwright* ile yapılacak **Görsel Regresyon (Visual Regression)** testlerinde tutarlılık sağlamak ve kontrast hatalarını önlemek için WCAG standartlarına göre belirlenmiştir.

Şekil 2: Responsive Grid Davranışı ve Menü Dönüşümü



"Bu şema, arayüzün Bootstrap Grid sistemi üzerindeki tepkisel davranışını özetlemektedir. Masaüstü görünümünde (Desktop) col-md-3 genişliğinde sabit duran **Yan Menü**, mobil cihazlarda (<768px) otomatik olarak gizlenmekte ve **Header** alanında bir **Hamburger Menü** (**☰**) ikonuna dönüşterek kullanım alanından tasarruf sağlamaktadır. İçerik tabloları ise mobil görünümde yatay kaydırma (Horizontal Scroll) özelliğine sahip olacaktır."

2.3. Test Odaklı Element İsimlendirme ve Locator Stratejisi

Bu proje, bir "Test Otomasyon Altyapısı Geliştirme" projesi olduğu için; kullanıcı arayüzü (UI) geliştirilirken sadece son kullanıcı deneyimi değil, aynı zamanda **Playwright** otomasyon botlarının sayfayı en hızlı ve en kararlı (stable) şekilde yönetebilmesi hedeflenmiştir.

Geleneksel test otomasyonunda sıkılıkla kullanılan XPath veya CSS Selector yöntemleri, tasarım değişikliklerinden çok çabuk etkilendiği için testlerin sık sık kırılmasına (Flakiness) neden olmaktadır. Bu riski minimize etmek amacıyla, **Playwright Best Practices** rehberliğinde aşağıdaki stratejiler uygulanmıştır:

A. "User-Facing" Değil, "Test-Dedicated" Attribute Kullanımı

Bir butonun rengi değiştiğinde class ismi değişimdir (.btn-primary -> .btn-danger), veya sayfa düzeni değiştiğinde elementin XPath adresi bozulabilir. Ancak testlere özel tanımlanan bir ID değişmez.

Bu nedenle, etkileşime girilecek tüm kritik elementlere, sadece test otomasyonunun göreceği data-testid niteliği (attribute) eklenmesi zorunlu tutulmuştur.

B. Shadow DOM ve Dinamik İçerik Yönetimi

Kullanılan Bootstrap bileşenlerinin (Örn: Dropdown menüler) veya dinamik yüklenen modalların içeresine erişmek için Playwright'ın "Auto-waiting" (Otomatik Bekleme) mekanizmasıyla uyumlu locator metotları tercih edilmiştir.

Tablo: Playwright Locator Stratejisi ve Kod Karşılıkları

Ekran	Element	HTML Kod Yapısı (data-testid)	Playwright (C#) Kullanımı	Avantajı
Login	E-posta Kutusu	<input data-testid="input-email" ... />	Page.GetByTestId("input-email")	CSS değişikliklerinden etkilenmez. En kararlı yöntemdir.
Login	Giriş Butonu	<button data-testid="btn-login">...	Page.GetByTestId("btn-login").ClickAsync()	Butonun metni ("Giriş" -> "Login") değişse bile test çalışmaya devam eder.
Login	Hata Mesajı	<div data-testid="alert-error">...	Expect(Page.GetByTestId("alert-error")).ToBeVisibleAsync()	Mesajın ekranda belirmesini (Visibility) otomatik olarak bekler.

Ekran	Element	HTML Kod Yapısı (data-testid)	Playwright (C#) Kullanımı	Avantajı
Başvuru	Okul Listesi	<select data-testid="select-uni">...	Page.Locator("[data-testid='select-uni']").SelectOptionAsync("1")	Dropdown seçeneklerini value veya label ile seçmeyi kolaylaştırır.
Başvuru	Dosya Yükle	<input type="file" data-testid="upload-file">	Page.SetInputFilesAsync(..., "transcript.pdf")	Gizli (Hidden) file input elementlerini bile bulup dosya yükleyebilir.

C. Erişilebilirlik (Accessibility) Entegrasyonu

Playwright, elementleri bulurken erişilebilirlik etiketlerini (aria-label, role) kullanmayı teşvik eder. Bu sayede yazdığımız testler, aynı zamanda uygulamanın engelli kullanıcılar için ne kadar uyumlu olduğunu da dolaylı yoldan test etmiş olur.

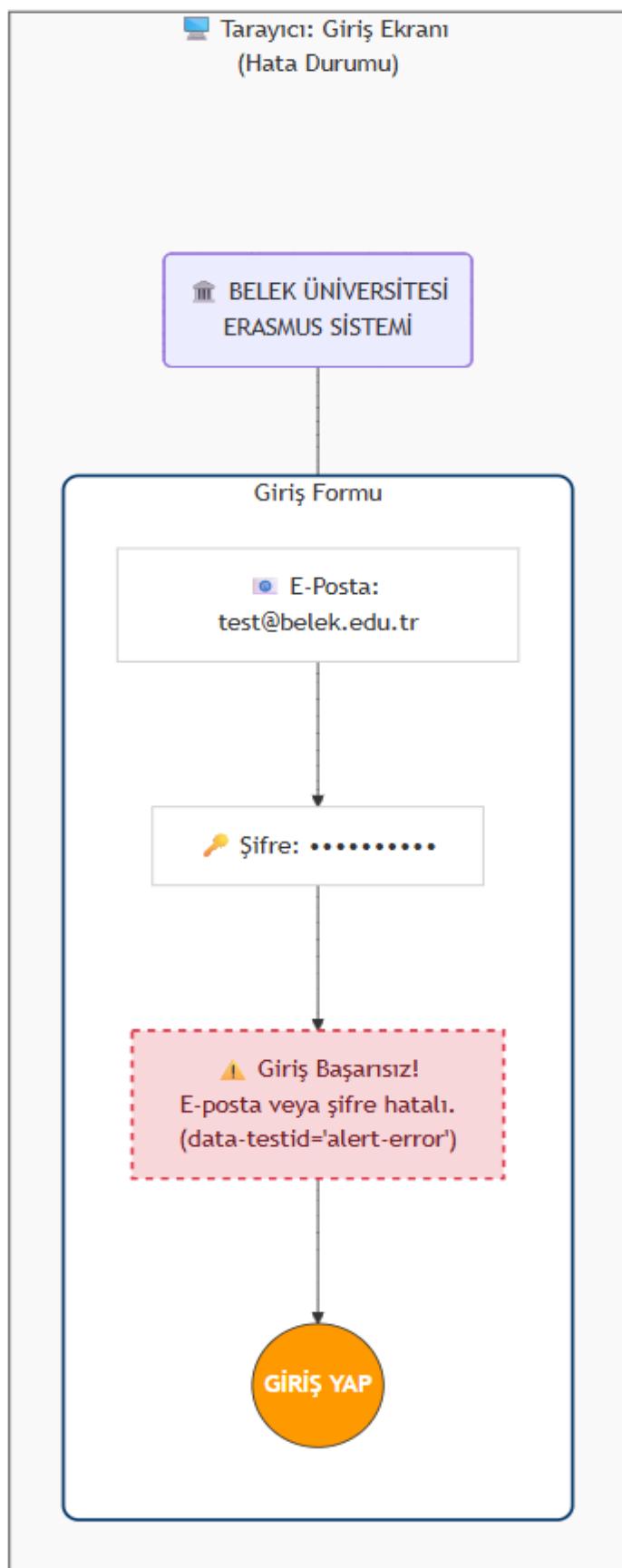
- Örnek: Bir ikona tıklamak için Page.GetByRole(AriaRole.Button, new() { Name = "Ayarlar" }) kullanılması, ekran okuyucuların o butonu görüp görmedğini de doğrular.

2.4. UI Durum Yönetimi ve Hata Gösterimi (Error States)

Kullanıcı deneyimi (UX) tasarımını sadece başarılı işlemleri değil, hatalı veri girişi gibi negatif senaryoları da kapsamalıdır. Özellikle **Playwright** ile yapılacak "Negatif Test Senaryoları"nın (Örn: Hatalı Şifre Kontrolü) doğrulanabilmesi için, hata mesajlarının DOM üzerindeki konumu ve görünürlüğü tasarlanmıştır.

Aşağıdaki şemada, başarısız bir giriş denemesi sonrası arayüzün alacağı "**Error State**" (Hata Durumu) gösterilmiştir. Bu tasarımda beliren kırmızı uyarı kutusu, test otomasyonu tarafından Expect(Locator).ToBeVisibleAsync() metodu ile yakalanacaktır.

Şekil 2.1: Hatalı Giriş Durumu (Error State) Tasarımı



3. Sonuç

Projenin 7. haftasında, yazılım geliştirme sürecinin en kritik aşamalarından biri olan "**Arayüz Tasarımı ve Test Otomasyon Entegrasyonu**" başarıyla tamamlanmıştır. Sadece son kullanıcı deneyimini (UX) değil, aynı zamanda **Geliştirici Deneyimini (DX)** ve test süreçlerini de odağa alan hibrit bir tasarım yaklaşımı benimsenmiştir.

Elde edilen kazanımlar şu şekilde özetlenebilir:

- Tam Test Uyumluluğu:** Arayüz elementleri, **Playwright for .NET** botlarının en hızlı ve kararlı şekilde etkileşime girebileceği `data-testid` stratejisi ile yapılandırılmış, böylece test otomasyonunun "kırılganlık" (flakiness) riski minimize edilmiştir.
- Responsive ve Dinamik Mimari:** Masaüstü ve mobil cihazlar arasındaki grid geçişleri (Responsive Behavior) ve hata durumlarındaki (Error States) arayüz tepkileri, hem görsel hem de şematik olarak (Mermaid.js) dokümant edilmiştir.
- Sürdürülebilir Altyapı:** Arayüz katmanı (UI) ile Test Otomasyon katmanı artık "aynı dili konuşur" hale gelmiştir. Görsel tasarımlar, arka planda çalışacak test senaryolarının gereksinimlerini %100 karşılayacak olgunluğa erişmiştir.

HAFTA 8 İLERLEME RAPORU

Proje Adı: Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi

Rapor Adı: Hafta 8 Proje Yönetim Planı, Sürüm Takvimi ve İş Dağılımı

Proje Sorumlusu: Anıl Bilgehan YÜZGEÇ - 220007012

Danışman(lar): Dr. Adem ŞİMŞEK – Öğr. Gör. Soner DEDEOĞLU

Tarih: 16/01/2026

1. Özeti

Projenin 8. haftasında, Rehber'in 8. hafta çıktısı olarak belirlediği "**Proje Planı ve Sürüm Takvimi**" çalışmaları tamamlanmıştır.

Önceki 7 hafta boyunca tamamlanan Analiz (SRS), Veritabanı Tasarımı ve Arayüz (UI/UX) çalışmaları temel alınarak; **Hafta 9 ile Hafta 18** arasındaki Test Süreci detaylandırılmıştır. Projenin bir "Test Otomasyon Projesi" olması sebebiyle, planlamada **Playwright entegrasyonuna** ve **Test Senaryolarının kodlanması** özel zaman dilimleri ayrılmıştır.

2. Hafta 8 Çıktıları: Proje Yönetim Planı

Rehberde belirtilen kontrol noktaları (Zamanlama gerçekçiliği, kilometre taşları) dikkate alınarak **4 Ana Fazlı** bir geliştirme takvimi oluşturulmuştur.

2.1. Sürüm Takvimi ve Teslimat Kriterleri (Release Calendar)

Projenin geliştirme süreci, riskleri minimize etmek ve geri bildirim döngülerini (Feedback Loops) sıkı tutmak amacıyla 3 ana kilometre taşına (Milestone) bölünmüştür. Her sürüm, bir önceki fazın üzerine inşa edilecek şekilde kurgulanmıştır.

Milestone 1: v0.5 Alpha

- Hedef Tarih:** 11. Hafta Sonu
- Odağı:** Test otomasyonu için gerekli Veri mimarisi ve Servis katmanının (API) hazırlanması.
- Teslimat İçeriği:**
 - Veritabanı:** PostgreSQL şemasının Docker üzerinde ayağa kaldırılması ve "Seed Data" (Örnek veriler) ile beslenmesi.
 - API:** .NET 8 tabanlı RESTful servislerin (Auth, Student, Application Controller) tamamlanması.
 - Dokümantasyon:** Swagger (OpenAPI) arayüzü üzerinden tüm uçların (Endpoints) test edilebilir hale gelmesi.
- Başarı Kriteri (DoD):** Postman üzerinden sisteme "Öğrenci" olarak kayıt olunup (Register), "Token" alınabilmesi (Login) ve bu token ile "Başvuru Formu" gönderilebilmesi.

Milestone 2: v0.8 Beta

- Hedef Tarih:** 14. Hafta Sonu
- Odak:** Uçtan uca (E2E) testlere uygun, kararlı bir Arayüz (UI) ve Entegrasyon sağlanması.
- Teslimat İçeriği:**
 - UI Entegrasyonu:** Hafta 7'de tasarlanan Figma/HTML arayüzlerinin API servislerine bağlanması.
 - İşlevsellik:** Kullanıcının kod yazmadan (Postman kullanmadan), sadece tarayıcı arayüzü kullanarak başvuru sürecini tamamlayabilmesi.
 - Validasyon:** Hatalı veri girişlerinde (Örn: Eksik TC Kimlik, Yanlış Format) arayüzün kullanıcıya anlık geri bildirim (Error States) vermesi.
- Başarı Kriteri (DoD):** Sistemin "Happy Path" (Sorunsuz Akış) senaryosunu manuel olarak uçtan uca tamamlayabilmesi ve Responsive (Mobil/Tablet) görünümlede kırılma yaşanmaması.

Milestone 3: v1.0 Release Candidate

- Hedef Tarih:** 18. Hafta Sonu (Final Teslim)
- Odak:** Stabilite, Otomasyon ve Dokümantasyon.
- Teslimat İçeriği:**
 - Test Otomasyonu:** Playwright for .NET ile yazılmış E2E (Uçtan Uca) test senaryolarının tamamlanması ve CI/CD hattında (GitHub Actions) otomatik çalışır hale gelmesi.
 - Raporlama:** Testlerin sonucunda oluşturulan HTML raporlarının (Pass/Fail oranları) sunulması.
 - Paketleme:** Projenin son kullanıcı kılavuzları ve kurulum (Installation) önerileri ile birlikte teslim edilmesi.
- Başarı Kriteri (DoD):** Sistemde bilinen kritik hataların (Critical Bugs) "0" olması ve tüm otomatik testlerin "Yeşil" (Passed) vermesi.

2.2. Sprint Planlaması ve Detaylı İş Dağılımı (WBS)

Projenin geliştirme süreci, karmaşıklığı yönetmek ve riskleri erken tespit etmek amacıyla Scrum çerçevesinde 4 ana Koşu'ya (Sprint) bölünmüştür. Her bir sprint, bir önceki haftalarda (Hafta 1-7) yapılan analiz ve tasarımların hayatı geçirilmesini hedefler.

Detaylı Sprint Backlog ve İş Kırılım Yapısı (WBS)

Sprint/Hafta	Odak Alanı	Detaylı Görevler (Tasks & Sub-tasks)	Teknik Detaylar & DoD (Bitti Tanımı)
Sprint 1 <i>(Hafta 9-11)</i>	Test Ortamının (SUT) Hazırlığı - Backend	1. Altyapı Kurulumu: <ul style="list-style-type: none">.NET 8 Web API projesinin oluşturulması.Generic Repository ve Unit of Work yapısının kurulması.Global Exception Handling (Hata Yönetimi) middleware yazımı. 2. Veritabanı İnşası: <ul style="list-style-type: none">User, Application entity'lerinin kodlanması.EF Core Migrations ile veritabanının güncellenmesi.	Teknolojiler: .NET 8, EF Core, PostgreSQL, Docker, Swagger. DoD: Swagger arayüzünden başarılı bir şekilde "Login" olunup Token alınabiliyor ve bu Token ile veritabanına yeni bir başvuru kaydı atılabiliyor olmalı.

Sprint/Hafta	OdaK Alanı	Detaylı Görevler (Tasks & Sub-tasks)	Teknik Detaylar & DoD (Bitti Tanımı)
		<ul style="list-style-type: none"> "Seed Data" (Örnek veriler) ile DB'nin beslenmesi. <p>3. Auth Servisi:</p> <ul style="list-style-type: none"> JWT (JSON Web Token) üretimi ve konfigürasyonu. Şifrelerin BCrypt ile hashlenerek saklanması. <p>4. Business Logic:</p> <ul style="list-style-type: none"> Başvuru oluşturma ve listeleme servislerinin (Service Layer) yazılması. 	
Sprint 2 <i>(Hafta 12-14)</i>	Test Ortamının (SUT) Hazırlığı - Frontend	<p>1. UI Kodlama:</p> <ul style="list-style-type: none"> Hafta 7'de tasarlanan Figma çizimlerinin HTML5/CSS3'e dökülmesi. Bootstrap Grid sistemi ile Responsive (Mobil Uyumlu) yapının kurulması. <p>2. API Bağlantısı:</p> <ul style="list-style-type: none"> JavaScript Fetch API veya Axios kullanılarak Backend servislerine bağlanılması. JWT Token'ın LocalStorage'da güvenli saklanması. <p>3. Form Yönetimi:</p> <ul style="list-style-type: none"> Başvuru formundaki inputların (TCKN, Not Ortalaması) validasyon kurallarının yazılması. Backend'den dönen hataların (400 Bad Request) kullanıcıya kırmızı uyarı kutusu (Alert) olarak gösterilmesi. 	<p>Teknolojiler: HTML5, CSS3, JavaScript (ES6+), Bootstrap.</p> <p>DoD: Bir kullanıcı, herhangi bir kod bilgisine ihtiyaç duymadan tarayıcı üzerinden sisteme girip, formu doldurup "Başvuruyu Tamamla" butonuna bastığında "Başarılı" mesajını görebilmeli.</p>
Sprint 3 <i>(Hafta 15-16)</i>	Test Otomasyon Altyapısı ve Senaryolar	<p>1. Playwright Kurulumu:</p> <ul style="list-style-type: none"> Microsoft.Playwright.NUnit kütüphanesinin projeye dahil edilmesi. playwright.config dosyasında tarayıcı (Chromium, Firefox) ayarlarının yapılması. <p>2. Page Object Model (POM):</p> <ul style="list-style-type: none"> LoginPage.cs ve ApplicationPage.cs sınıflarının oluşturulması. Elementlerin data-testid attribute'ları ile tanımlanması. <p>3. Senaryo Kodlaması:</p> <ul style="list-style-type: none"> Happy Path: Başarılı giriş ve başvuru senaryosu. Negative Path: Hatalı şifre ve eksik belge senaryoları. Testlerin video kaydı ve ekran görüntüsü (Screenshot) alacak şekilde ayarlanması. 	<p>Teknolojiler: Playwright, NUnit, C#.</p> <p>DoD: dotnet test komutu çalıştırıldığında tüm senaryoların hatasız (Yeşil) geçmesi ve sonuçların HTML raporu olarak üretilmesi.</p>
Sprint 4 <i>(Hafta 17-18)</i>	Finalizasyon & Raporlama	<p>1. Optimizasyon:</p> <ul style="list-style-type: none"> Test kodlarındaki tekrar eden blokların temizlenmesi (Refactoring). "Flaky Test" (Kararsız test) analizi yapılması. <p>2. Dokümantasyon:</p> <ul style="list-style-type: none"> "Nasıl Kurulur?" (Readme.md) dosyasının hazırlanması. Kullanıcı Kılavuzu'nun PDF olarak oluşturulması. <p>3. Final Paketi:</p> <ul style="list-style-type: none"> Projenin Docker Compose ile tek komutta ayağa kalkacak hale getirilmesi. Sunum provasının yapılması. 	<p>Teslimat: Kaynak Kodlar (GitHub), Final Raporu, Sunum Dosyası, Test Raporu.</p> <p>DoD: Projenin Jüri/Danışman bilgisayarında sorunsuz çalışması ve kurulumun 5 dakikadan kısa sürmesi.</p>

GÜNCELLENMİŞ İŞ KIRLIM YAPISI (WBS) – v2.0

Proje: Erasmus Sistemi Test Projesi Geliştirme Projesi

1.0. Proje Hazırlık ve Planlama Fazı

- **1.1. Proje Başlatma ve Teknoloji Seçimi**
 - 1.1.1. Proje kapsamının belirlenmesi.
 - 1.1.2. Test araçlarının seçimi (Playwright, xUnit).
 - 1.1.3. GitHub repo ve proje yönetim şablonunun hazırlanması.
- **1.2. Literatür Taraması (Mevcut Erasmus sistemleri ve Test Standartları)**

2.0. Test Analizi ve Test Tasarımı Fazı

- **2.1. Gereksinim Analizi**
 - 2.1.1. Test edilecek sistemin (SUT) gereksinimlerinin (SRS) çıkarılması.
 - 2.1.2. Kullanıcı Hikayeleri üzerinden Kabul Kriterlerinin belirlenmesi.
- **2.2. Veri ve Arayüz Tasarımı**
 - 2.2.1. Test verilerinin (Test Data) tasarılanması ve Veritabanı şeması.
 - 2.2.2. Otomasyona uygun UI Tasarımı (Mockup).
 - 2.2.3. **Kritik:** Test otomasyonu için data-testid stratejisi.

3.0. Test Ortamının (SUT) Geliştirilmesi - Backend

- **3.1. Test Altyapısı Kurulumu (Docker/PostgreSQL)**
- **3.2. Veritabanı ve Seed Data (Örnek Veri) Hazırlığı**
- **3.3. Test Edilecek API Servislerinin (Auth/App) Hazırlanması**
- **3.4. API Dokümantasyonu (Swagger).**

4.0. Test Ortamının (SUT) Geliştirilmesi - Frontend

- **4.1. Test Edilecek Arayüzlerin Kodlanması**
- **4.2. API Entegrasyonu ve Test Verisi Bağlantısı**
- **4.3. Hata Yönetimi (Testlerde yakalanacak hata mesajlarının kodlanması)**

5.0. Test Otomasyonu Geliştirme

- **5.1. Playwright Altyapı Kurulumu**
 - 5.1.1. xUnit test projesinin çözüme eklenmesi.
 - 5.1.2. Tarayıcı ve Config ayarları.
- **5.2. Test Mimarisi (POM)**
 - 5.2.1. BasePage, BaseTest yapıları.
 - 5.2.2. Page Object sınıfları.
- **5.3. Test Senaryolarının Kodlanması**
 - 5.3.1. Smoke Test (Duman Testi) ve Happy Path senaryoları.
 - 5.3.2. Uçtan Uca Fonksiyonel Testler.
 - 5.3.3. Regression (Regresyon) ve Negatif senaryolar.

6.0. Finalizasyon ve Raporlama

- **6.1. Kalite Güvence ve Refactoring**
- **6.2. Dokümantasyon (Test Planı, Kullanıcı Kılavuzu, Final Raporu)**
- **6.3. Sürüm Paketleme ve Sunum**

3. Zamanlama Gerçekçiliği ve Risk Yönetim Planı

Yazılım projelerinde sıkılıkla karşılaşılan "Teslimat Gecikmesi" ve "Kapsam Kayması" (Scope Creep) risklerine karşı, rehberin "Zamanlama Gerçekçiliği" kriteri doğrultusunda aşağıdaki önlemler alınmıştır:

3.1. Teknik Riskler ve Tampon Süreler (Buffer Management)

Projenin geliştirme sürecinde oluşabilecek öngörülemeyen teknik aksaklıklar için takvime **%10 oranında "Görünmez Tampon Süre"** eklenmiştir.

- Docker ve Ortam Riskleri:** Geliştirme ortamı (Dev Env) ile Canlı ortam (Prod Env) arasındaki konfigürasyon farklarından doğabilecek sorunlar için **Sprint 1'in sonuna 2 gün, Sprint 2'nin sonuna 3 gün** hata ayıklama (Debugging) payı bırakılmıştır.
- Bağımlılık (Dependency) Yönetimi:** Frontend geliştirmesinin (Sprint 2) başlayabilmesi için Backend API'nin bitmiş olması gerekmektedir. Bu bağımlılık riskini yönetmek için, Sprint 1'de API tamamlanmasa bile Frontend tarafında **"Mock Data" (Sahte Veri)** kullanılarak arayüz kodlamasının aksamadan devam etmesi planlanmıştır.

3.2. Test Odaklı Planlama Stratejisi

Bu proje standart bir web geliştirme projesi değil, bir "**Test Otomasyon Altyapısı**" projesidir. Bu nedenle zaman çizelgesi, klasik projelerden farklı kurgulanmıştır:

- Neden 2 Hafta Sadece Test? (Hafta 15-16):** UI Testleri (Playwright), arayüzdeki en ufak değişiklikten etkilenebilir. Bu nedenle, test senaryolarının kodlanması, Arayüz (UI) tamamen kararlı hale gelene kadar (Sprint 2 bitimi) bekletilmiştir. Bu strateji, test kodlarının sürekli revize edilmesini (Rework) önlüyor ve zaman kaybını engeller.
- Öğrenme Eğrisi (Learning Curve):** Proje ekibinin Playwright aracına adaptasyonu için Sprint 3'ün ilk 3 günü "Ar-Ge ve Dokümantasyon Okuma" süresi olarak ayrılmıştır.

3.3. Kapsam Yönetimi (Scope Management)

Projenin 18. haftaya yetişmemesi durumunda uygulanacak "**B Planı (MoSCoW Analizi)**" şimdiden belirlenmiştir:

- Must Have (Olmasa Olmaz):** Login, Başvuru Formu, Playwright Testleri. (Bu maddelerden asla taviz verilmeyecektir.)
- Could Have (Olsa İyi Olur):** Profil fotoğrafı yükleme, gelişmiş admin raporları. (Zaman daralırsa bu özellikler kapsam dışı bırakılacaktır.)

3.4. Risk Matrisi ve Yönetim Stratejileri

Projenin başarıya ulaşmasını etkileyebilecek potansiyel riskler; gerçekleşme olasılıkları ve yaratacakları etki düzeyine göre puanlanmış (1-5 arası) ve önleme planları (Mitigation Plans) hazırlanmıştır. Proje planındaki (WBS) her bir ana iş paketinin kendine özgü riskleri analiz edilmiş ve aşağıdaki matriste sınıflandırılmıştır.

3.4.1. Risk Matrisi

Risk ID	İlgili WBS/İş Paketi	Risk Tanımı	Olasılık	Etki	Skor	Seviye
R1	1.0 PLANLAMA	Yanlış Araç Seçimi: Playwright'in proje isterlerini (Upload, PDF vb.) karşılayamaması.	2	5	10	Yüksek
R2	2.0 TEST ANALİZİ	Test Kapsam Şişmesi (Scope Creep): Test edilmeyecek kadar fazla senaryo yazılması.	4	5	20	Kritik
R3	3.0 TEST ORTAMI (SUT)-BACKEND	Test Edilemez UI: Arayüz elementlerine (Button, Input) ID verilmemesi.	3	5	15	Kritik
R4	3.0 TEST ORTAMI (SUT)-BACKEND	Veri Kirliliği: Test verilerinin (Seed Data) her koşumda bozulması.	2	4	8	Orta
R5	3.0 TEST ORTAMI (SUT)-BACKEND	Ortam Erişimi: Test ortamındaki Auth/Login servisinin çökmesi.	3	5	15	Kritik
R6	4.0 TEST ORTAMI (SUT)-BACKEND	Entegrasyon: Test API modelleri ile Arayüz verilerinin uyusuzluğu.	4	4	16	Kritik
R7	5.0 OTOMASYON	Flaky Tests: Testlerin bazen geçip bazen kalması (Kararsızlık).	5	3	15	Kritik
R8	6.0 CI/CD & TESLİM	Pipeline Hatası: GitHub Actions'ta tarayıcıların (headless) çalışmaması.	2	5	10	Yüksek

"Yapılan analiz sonucunda, Skor 10 ve üzerinde olan maddeler 'Yüksek Öncelikli' kabul edilerek, her biri için detaylı Önleme (Mitigation) ve Acil Durum Aksiyonu (Contingency) Planları oluşturulmuştur."

3.4.2. Önleme (Mitigation) ve Acil Durum Aksiyonu (Contingency) Planları

R1: Yanlış Araç Seçimi (Playwright Uyumu)

- Önlem:** Hafta 1'de "Proof of Concept" (Kavram Kanıtı) yapılarak Playwright'in .NET ve dosya yükleme uyumu doğrulandı.
- Acil Durum:** Playwright ile otomatize edilemeyen spesifik bir adım olursa, o adım kapsam dışı bırakılarak manuel test raporuna eklenecektir.

R2: Test Kapsam Şişmesi (Scope Creep)

- Önlem:** Test senaryoları MoSCoW analizine göre (Critical, Major, Minor) önceliklendirildi.
- Acil Durum:** Süre yetmezse, "Minor" öncelikli (Örn: Profil Resmi Yükleme, Rapor Filtreleme) test senaryoları otomasyona dahil edilmeyecektir.

R3: Test Edilemez UI (Otomasyona Uygunuz Arayüz)

- Önlem:** Test ortamı hazırlanırken, tüm buton ve inputlara data-testid attribute'u ekleme standartı getirildi.
- Acil Durum:** ID unutulan elementler için daha yavaş çalışan ancak ID gerektirmeyen XPath veya CSS Selector (Text bazlı) locators kullanılacak.

R4: Veri Kirliliği (Data Integrity)

- Önlem:** Her test koşumu öncesi çalışan [SetUp] metodunda veritabanı sıfırlanıp temiz veri basılacak.
- Acil Durum:** Otomatik sıfırlama çalışmazsa, test veritabanı silinip manuel SQL scripti ile yeniden oluşturulacak.

R5: Ortam Erişimi (Auth Servisi)

- Önlem:** Test token süresi uzun tutulacak ve Refresh Token mekanizması test ortamı için esnetilecek.
- Acil Durum:** Login servisi çalışmazsa, test ortamında (Environment: Test) ByPassAuth modu açılarak testlerin giriş yapmadan çalışması sağlanacak.

R6: Entegrasyon (API-UI Uyumsuzluğu)

- Önlem:** Swagger dokümantasyonu "Tek Gerçek Kaynak" (SSOT) kabul edilecek.
- Acil Durum:** API gecikirse, Arayüz test senaryoları "Mock Data" (Sahte JSON verisi) kullanılarak yazılmaya devam edilecek.

R7: Flaky Tests (Kararsız Testler)

- Önlem:** Statik bekleme (Thread.Sleep) kesinlikle yasaklandı; dinamik WaitForSelector veya WaitForLoadState kullanılacak.
- Acil Durum:** Bir test CI/CD sürecinde 3 kez üst üste sebepsiz hata veriyorsa, @Ignore etiketiyle geçici olarak devre dışı bırakılacak.

R8: Pipeline Hatası (Deployment)

- Önlem:** GitHub Actions workflow dosyası lokalde act aracı ile veya fork edilmiş bir repo üzerinde önceden test edilecek.
- Acil Durum:** CI/CD pipeline çalışmazsa, testler yerel makinede koşulup, oluşturulan HTML raporu manuel olarak sunum dosyasına eklenecek.

(Proje Risk Ölçeklendirme Haritası, Raporun Ekler Bölümünde sunulmuştur.)

3.5. Proje Başarı Kriterleri ve Performans Göstergeleri (KPIs)

Projenin başarıyla tamamlanmış sayılabilmesi için, sadece fonksiyonel gereksinimlerin (SRS) karşılanması yeterli değildir. Yazılımın kalitesini ve test otomasyonunun güvenilirliğini ölçmek amacıyla, Rehberin **16. haftada** talep ettiği "**Güvenlik ve Performans**" standartlarıyla uyumlu, ölçülebilir başarı metrikleri (KPI) belirlenmiştir.

Aşağıdaki tabloda, **v1.0 (Canlı Sürüm)** için hedeflenen kalite eşikleri tanımlanmıştır:

Proje Başarı Metrikleri (KPI Tablosu)

Kategori	Metrik (Ölçüt)	Hedef Değer	Açıklama ve Gerekçe
TEST KALİTESİ	Code Coverage (Kod Kapsama)	>%80	Yazılan test senaryolarının, Backend ve Frontend kod satırlarının en az %80'ine temas etmesi hedeflenir.
	Test Kararlılığı (Flakiness Rate)	<%1	Otomatik testlerin "yalancı negatif" (False Negative) verme oranı %1'in altında olmalı, testler güvenilir çalışmalıdır.
PERFORMANS	Sayfa Yüklenme Hızı (LCP)	< 2.5 sn	Ana sayfa ve Başvuru Formu'nun "Largest Contentful Paint" süresi, 3G ağ hızında bile 2.5 saniyenin altında olmalıdır.

Kategori	Metrik (Ölçüt)	Hedef Değer	Açıklama ve Gerekçe
	API Yanıt Süresi (Latency)	< 500 ms	"Başvuruyu Gönder" gibi kritik işlemlerin sunucu yanıt süresi, kullanıcıyı bekletmemelidir.
YAZILIM KALİTESİ	Kritik Hata Sayısı (Critical Bugs)	0 (Sıfır)	Canlı sürümde, kullanıcının sisteme girmesini veya başvuru yapmasını engelleyen hiçbir hata bulunmamalıdır.
	Düşük Öncelikli Hata (Minor Bugs)	< 5	Yazım hatası veya renk tonu farkı gibi işleyişi bozmayan en fazla 5 hata kabul edilebilir.
GÜVENLİK	OWASP Top 10 Kontrolü	%100 Geçiş	SQL Injection ve XSS (Siteler Arası Komut Dosyası) açıklarına karşı temel güvenlik taramaları temiz çıkmalıdır.

Değerlendirme Yöntemi: Bu kriterler, projenin **18. Haftasında (Final Teslim)** yapılacak olan performans testleri ve Playwright raporları ile doğrulanacak; hedeflerin altında kalan alanlar için optimizasyon çalışması yapılacaktır.

4. Görsel Planlama (Gantt Chart/Gant Şeması)

Projenin 18 haftalık rehber takvimine oturtulmuş görsel planı aşağıda sunulmuştur. Bu şema, görevlerin paralelliğini ve bağımlılıklarını gösterir.

4.1. Proje Geliştirme Takvimi: Tamamlanan Aşamalar (Hafta 1-Hafta 8)

(Bu kısım projenin geçmişini gösterir ve "☒ Tamamlandı" statüsüündedir.)

FAZ / GÖREV (WBS v4.0)	H1	H2	H3	H4	H5	H6	H7	H8
1.0. PROJE HAZIRLIK VE PLANLAMA FAZI								
1.1. Proje Başlatma & Teknoloji Seçimi	☒							
1.2. Literatür Taraması (Test Standartları)		☒						
2.0. TEST ANALİZİ VE TASARIMI FAZI								
2.1. Gereksinim Analizi & Kabul Kriterleri			☒	☒				
2.2. Veri ve Arayüz Tasarımı (Mockup)					☒	☒		
2.3. WBS, Gantt ve Sürüm Planlama							☒	

Semboller: (☒) Tamamlandı, (☒) Şu an Aktif

4.2. Proje Geliştirme Takvimi: Planlanan Aşamalar (Hafta 9-Hafta 18)

FAZ / GÖREV (WBS v4.0)	H9	H10	H11	H12	H13	H14	H15	H16	H17	H18
3.0. TEST ORTAMININ (SUT) GELİŞTİRİLMESİ - BACKEND										
3.1. Test Altyapısı Kurulumu (Docker/DB)	█									
3.2. Veritabanı ve Seed Data (Örnek Veri)	█	█								
3.3. Test API Servisleri (Auth/App)		█	█							
3.4. API Dokümantasyonu (Swagger)			█							
MILESTONE 1: Alpha Sürüm					★					
4.0. TEST ORTAMININ (SUT) GELİŞTİRİLMESİ - FRONTEND										
4.1. Test Arayüzleri Kodlaması					█					
4.2. API Entegrasyonu & Veri Bağlantısı					█	█				
4.3. Hata Yönetimi (Error States)						█				
MILESTONE 2: Beta Sürüm							★			
5.0. SPRINT 3: TEST OTOMASYONU GELİŞTİRME										
5.1. Playwright & xUnit Kurulumu							█			
5.2. Test Mimarisi (POM - BasePage)							█			
5.3. Senaryo Kodlaması (Smoke & Regression)						█	█			
6.0. SPRINT 4: FINALİZASYON VE RAPORLAMA										
6.1. Kalite Güvence & Refactoring								█		
6.2. Dokümantasyon (Test Planı/Raporu)								█	█	
6.3. Sürüm Paketleme ve Sunum Hazırlığı									█	
MILESTONE 3: Final Teslim										★

Semboller: (█) Planlanan Çalışma, (★) Kilometre Taşı

"Tablo 4.1 ve 4.2, projenin uçtan uca (Hafta 1-18) yönetim sürecini göstermektedir."

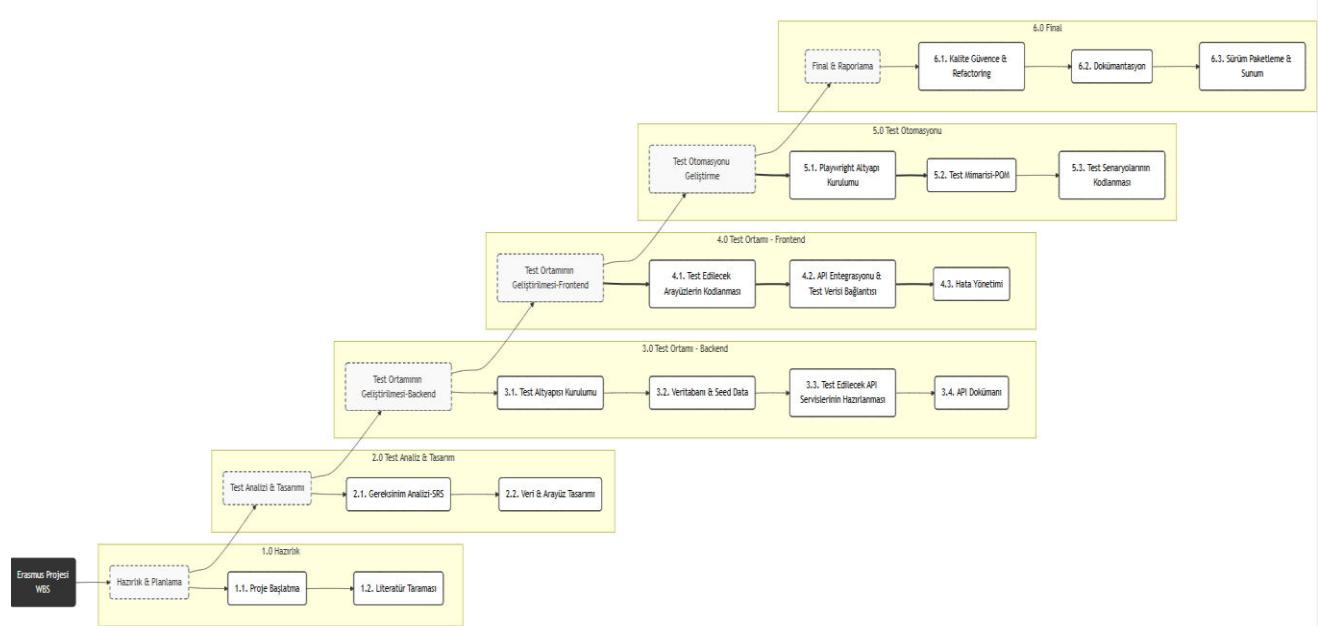
5. Sonuç

Hafta 8 itibarıyla, kağıt üzerindeki tasarımların (Hafta 1-7) çalışan bir yazılıma dönüştürüleceği yol haritası netleşmiştir. Proje takvimi, **Rehber'in 18. haftayı son teslim tarihi olarak belirlemesi** dikkate alınarak güncellenmiş ve gerçekçi hedeflere bölünmüştür.

EKLER

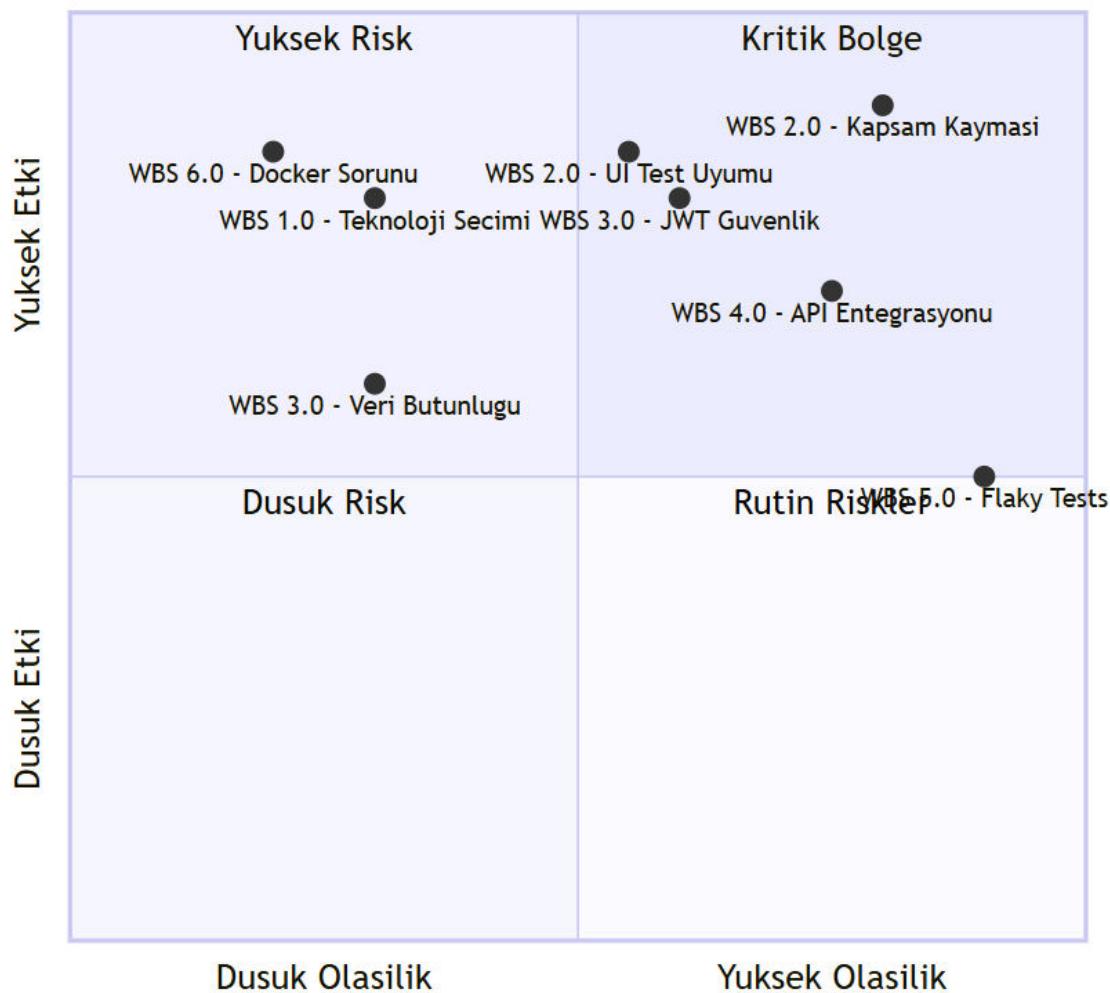
Aşağıda raporda atıfta bulunulan ve raporu destekleyici diyagramlar bulunmaktadır.

EK – A: Güncellenmiş İş Kırılım Yapısı (Wbs) – v2.0 Ve Sprint Planlaması Diyagramı

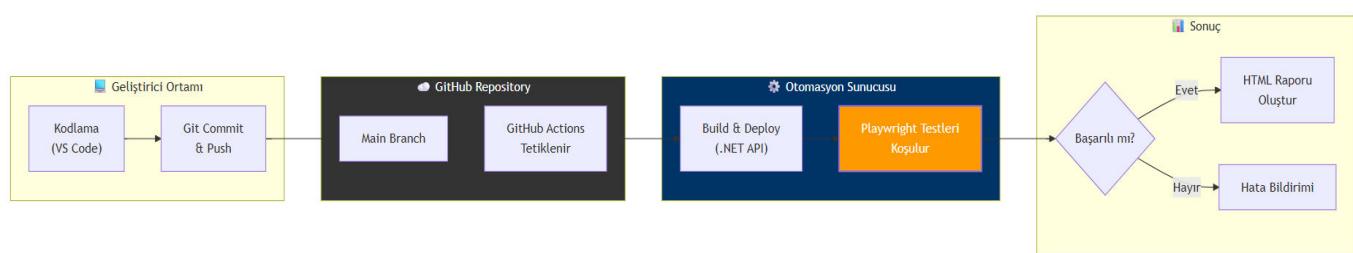


EK – B: Proje Risk Haritası

Proje Risk Onceliklendirme Haritasi



EK – C: CI/CD ve Otomasyon Akiş Şeması (Pipeline Architecture)



HAFTA 9 İLERLEME RAPORU

Proje Adı: Belek Üniversitesi Erasmus Sistemi Test Projesi Geliştirme Projesi

Rapor Adı: Hafta 9 İlk Prototip ve Sürüm Kontrol Sistemi Kurulumu

Proje Sorumlusu: Anıl Bilgehan YÜZGEÇ - 220007012

Danışman(lar): Dr. Adem ŞİMŞEK – Öğr. Gör. Soner DEDEOĞLU

Tarih: 16/01/2026

1. Özeti

Bu rapor, Hafta 9 beklentileri doğrultusunda; projenin backend altyapısı (prototip) hazırlanmış, katmanlı mimari (Layered Architecture) Visual Studio üzerinde kurgulanmış ve tüm kaynak kodlar sürüm kontrol sistemi (GitHub) üzerine taşınarak güvenli bir geliştirme ortamı oluşturulmuştur. Ayrıca, sistemin çalışırlığını Swagger ve Postman araçları ile doğrulanmıştır.

2. Hafta 9 Çıktıları ve Teslim Edilecekler

2.1. Kod Deposu ve Sürüm Kontrol Sistemi (VCS)

Projenin kaynak kodlarının takibi, sürüm yönetimi ve güvenliği için **Git** teknolojisi kullanılmış ve uzak depo (Remote Repository) olarak **GitHub** seçilmiştir.

- Kod Deposu Linki:** [ANIL08BY/ErasmusSystem: First Prototype](#)
- Yapılan İşlem:** Visual Studio üzerinde oluşturulan çözüm (Solution), GitHub Desktop aracı kullanılarak "main" dalı (branch) üzerinden uzak sunucuya "Push" edilmiştir.
.gitignore dosyası, gereksiz sistem dosyalarının (bin/obj) depoya yüklenmesini engelleyecek şekilde yapılandırılmıştır (Bkz. Şekil 4).

2.2. Prototip Mimarisi (Backend)

Hafta 4 raporunda belirlenen "Katmanlı Mimari" (Layered Architecture) yapısı, .NET 8.0 platformu üzerinde fiziksel olarak oluşturulmuştur. Proje çözümü (Solution), "Separation of Concerns" (İlgili Alanlarının Ayrımı) prensibine uygun olarak şu ana katmanlara ayrılmıştır (Bkz. Şekil 1):

- ErasmusSystem.API:** Sunum katmanıdır. Dış dünyaya açılan RESTful servisleri barındırır.
- ErasmusSystem.Business:** İş mantığının ve kurallarının işletileceği katmandır.
- ErasmusSystem.DataAccess:** Veritabanı (PostgreSQL/SQL Server) iletişimini sağlayan ORM (Entity Framework Core) katmanıdır.
- ErasmusSystem.Entities:** Veritabanı tablolarına karşılık gelen veri modellerini barındırır.
- ErasmusSystem.Tests:** Hafta 3 (SRS) raporunda belirlenen test senaryolarının kodlanacağı test otomasyon katmanıdır. Core, Pages (POM) ve Tests klasör yapısı hazırlanmıştır.

2.3. Teknik Altyapı ve Kütüphane Entegrasyonu

Projenin ihtiyaç duyduğu temel NuGet paketleri ilgili katmanlara entegre edilmiş ve sürüm uyumlulukları (v8.0.x) doğrulanmıştır (Bkz. Şekil 5 ve Şekil 6):

- **Veritabanı İçin (DataAccess):** Microsoft.EntityFrameworkCore.SqlServer ve Microsoft.EntityFrameworkCore.Tools paketleri yüklenerek veritabanı işlemlerine hazır hale getirilmiştir.
- **Test Otomasyonu İçin (Tests):** UI testleri için **Microsoft.Playwright**, doğrulama işlemleri için **FluentAssertions** kütüphaneleri projeye dahil edilmiştir.

2.4. Prototip Doğrulama ve Demo

Demo senaryosu olarak; sistemin ayağa kalktığı, dış dünyadan gelen isteklere (Request) cevap verebildiği ve veritabanı bağlantı katmanının hazır olduğu simülle edilmiştir. Geliştirilen backend prototipinin çalıştığını kanıtlamak amacıyla iki aşamalı bir "Sağlık Kontrolü" (Health Check) testi uygulanmıştır:

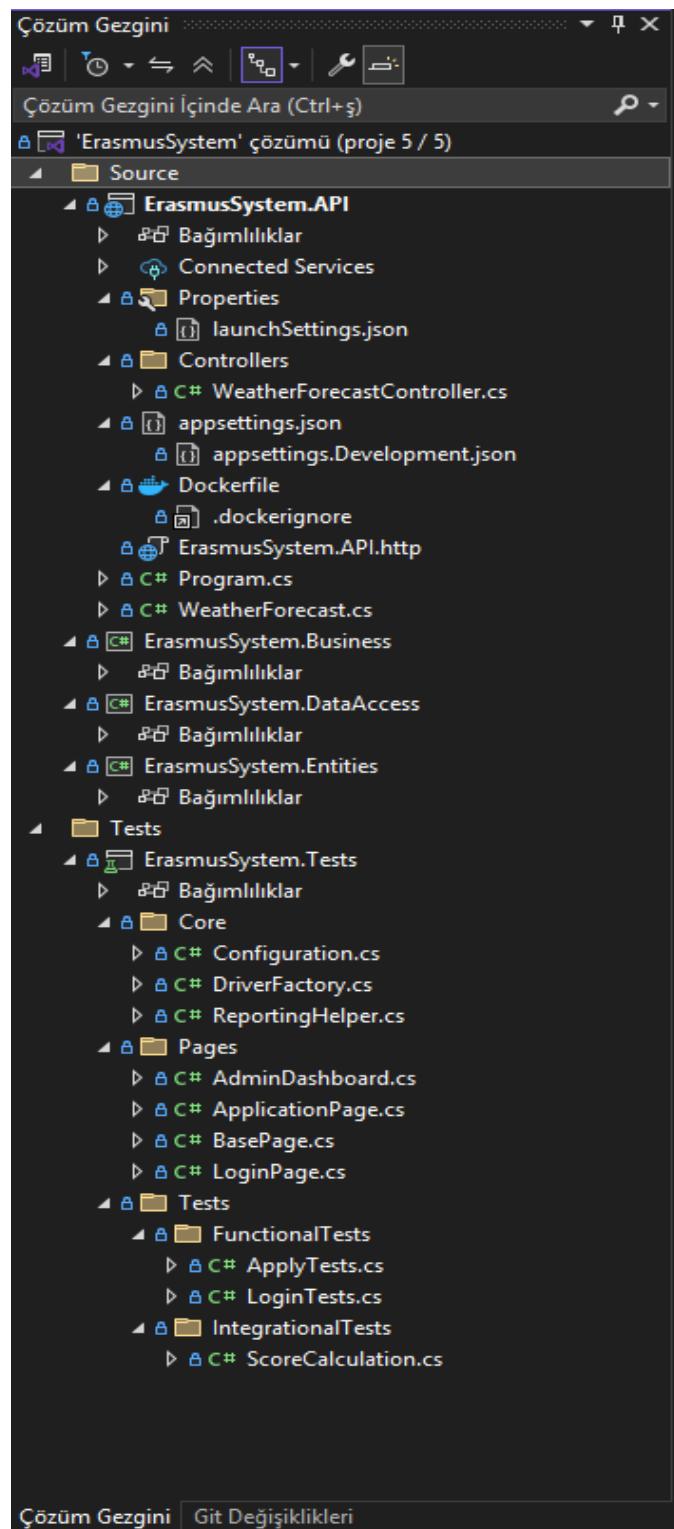
1. **Swagger UI Kontrolü:** API projesi Visual Studio ile ayağa kaldırılmış ve tarayıcı üzerinden Swagger arayüzüne erişim sağlanmıştır (Bkz. Şekil 2).
2. **Postman Testi:** Harici bir API istemcisi (Postman) kullanılarak API'ye HTTP GET isteği gönderilmiştir ve sunucudan başarılı (**Status: 200 OK**) yanıt döndüğü doğrulanmıştır (Bkz. Şekil 3).

3. Sonuç

Hafta 9 itibarıyla, projenin "Motoru" olarak nitelendirilebilecek backend altyapısı ve geliştirme ortamı eksiksiz olarak kurulmuştur. Hafta 1-8 Raporunda sunulan İş Kırılım Yapısı (WBS) ve Proje Takvimi ile uyumlu olarak; 9. hafta için hedeflenen 'Geliştirme Ortamının Kurulması' ve 'Backend Prototipinin Hazırlanması' kilometre taşlarına (Milestones) zamanında ulaşılmıştır. Proje ilerleyişinde herhangi bir sapma (deviation) bulunmamaktadır. Proje başında öngörülen 'Entegrasyon Sorunları' riski, bu hafta Katmanlı Mimari yapısının Visual Studio ile hatasız ayağa kaldırılmasıyla minimize edilmiştir. Seçilen teknolojilerin (Playwright, EF Core) sistemle uyumu doğrulanmıştır. Kod deposunun aktif hale gelmesi ve prototipin ilk HTTP isteklerine yanıt vermesiyle birlikte, Hafta 10'da planlanan 'Kodlama Aşaması (1)' kapsamında; Veritabanı tablolarının (Entity) oluşturulması ve Login (Giriş) ekranının backend kodlamasının yapılması hedeflenmektedir.

EKLER

EK – A: Visual Studio Çözüm Gezgini ve Katmanlı Mimari Yapısı



Açıklama: Kaynak kodların (Source) ve Test kodlarının (Tests) ayrıstırıldığı, modüler proje yapısı.

EK – B: Çalışan İlk Prototip (Swagger UI)

The screenshot shows the Swagger UI interface for the **ErasmusSystem.API** v1. At the top, the navigation bar includes the **Swagger** logo and the text "Powered by SMARTBear". The dropdown menu "Select a definition" is set to "ErasmusSystem.API v1".

The main content area is titled "WeatherForecast". It shows a "GET /weatherForecast" operation. Under the "Parameters" section, there are none. In the "Responses" section, there is one entry for status code 200 with the description "OK". The "Media type" dropdown is set to "text/plain". The "Example Value" field contains the following JSON:

```
[{"date": "2026-01-11", "temperatureC": 0, "temperatureF": 32, "summary": "String"}]
```

Below this, the "Schemas" section shows the definition for the "WeatherForecast" schema, which consists of four properties: date, temperatureC, temperatureF, and summary.

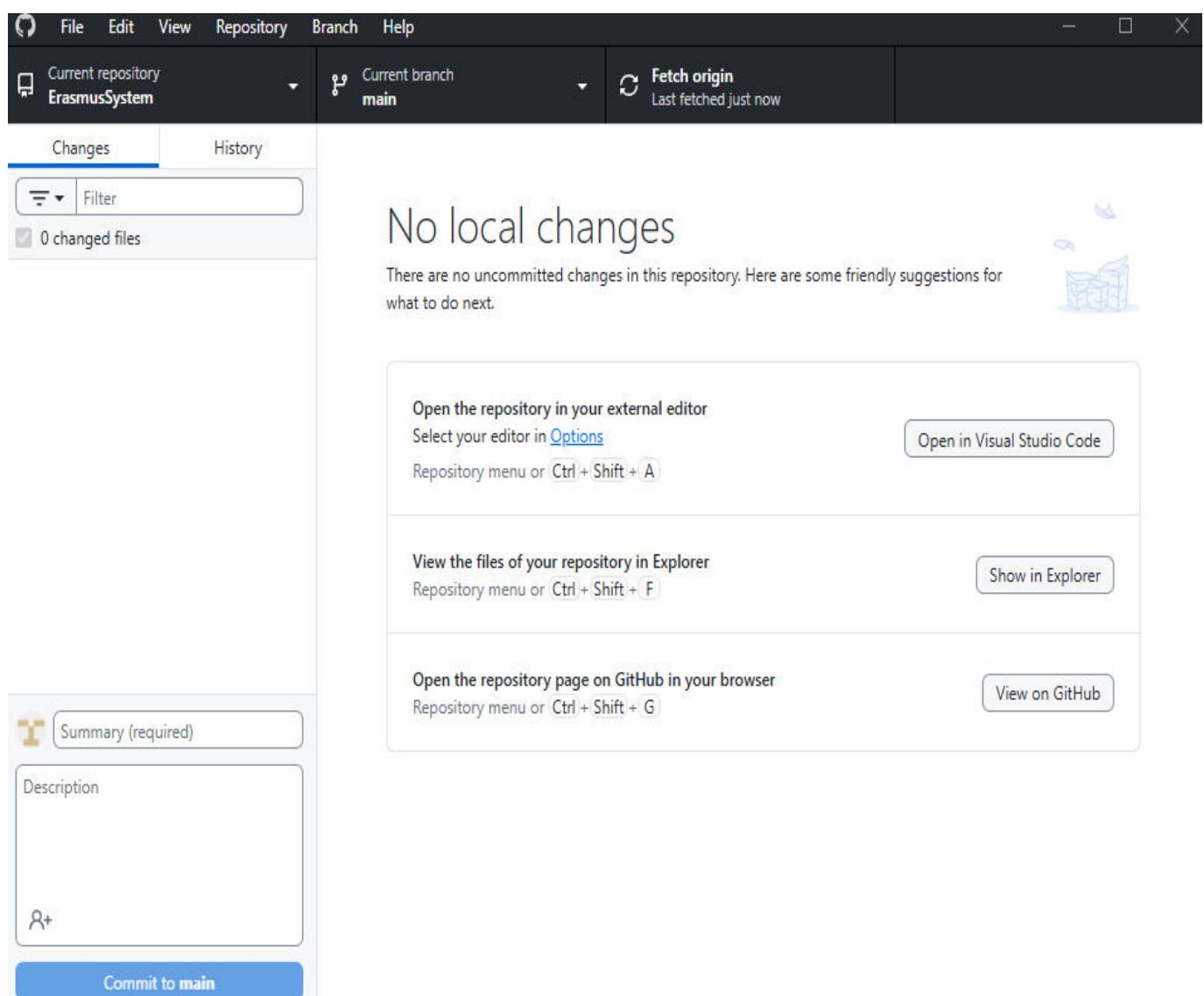
Açıklama: .NET 8.0 ve Visual Studio ile çalışan API'nin Swagger arayüzü çıktısı.

EK-C: Dış Erişim Doğrulaması (Postman)

The screenshot shows the Postman application interface. In the left sidebar, there's a collection named "Erasmus System API Tests" under "My Collection". A specific request titled "GET New Request" is selected. The request URL is set to "https://localhost:32773/swagger/index.html". The "Send" button is highlighted in blue. Below the URL, there are tabs for "Docs", "Params", "Auth", "Headers (6)", "Body", "Scripts", "Tests", and "Settings". The "Params" tab is active, showing a table with one row: "Key" (Value) and "Value" (Description). The main panel displays the response body in "HTML" format, which is the Swagger UI index page. The status bar at the bottom indicates a "200 OK" response with a size of 4.71 KB and a duration of 53 ms.

Açıklama: API uçlarının harici araçlarla erişilebilirliği ve 200 OK yanıtı alındığının kanıtı.

EK – D: GitHub Sürüm Kontrol Sistemi Durumu



Açıklama: Tüm yerel değişikliklerin (Local Changes) başarıyla uzak sunucuya (GitHub) aktarıldığını gösteren durum ekranı.

EK – E: Test Otomasyon Altyapısı (NuGet Paketleri)

NuGet: ErasmusSystem.Tests ➔ Program.cs

Gözat **Yüklü** Güncelleştirmeler

Ara (Ctrl+F) Paket kaynağı: nuget.org

Üst düzey paketler (14)

Paket	Hazırlayan	Özet	Sürüm
Allure.Xunit	Shumakov Ivan	Create beautiful reports from your xUnit.net tests.	2.14.1
Bogus	Brian Chavez	A simple and sane data generator for populating objects that supports different locales. A delightful port of the famed faker.js and inspired by FluentValidation. Use Bogus to create UIs with fake data or seed databases. Get started by using Faker class or a DataSet directly.	35.6.5
coverlet.collector	tonerdo	Coverlet is a cross platform code coverage library for .NET, with support for line, branch and method coverage.	6.0.0 6.0.4
FluentAssertions	Dennis Doomen, Jonas Nystrup, Xceed	A very extensive set of extension methods that allow you to more naturally specify the expected outcome of a TDD or BDD-style unit tests. Works with .NET Standard 2.0 and higher, .NET Framework 4.7 and higher and .NET 6 and higher.	8.8.0
.NET Microsoft.Extensions.Configuration.Binder	Microsoft	Provides the functionality to bind an object to data in configuration providers for Microsoft.Extensions.Configuration. This package enables you to represent the configuration data as strongly-typed classes defined in the application code. To bind a configuration, use the Microsoft.Extensions.Configuration.Binder.Get extension method on the IConfiguration object. To use this package, you also need to install a package for the configuration provider, for example, Microsoft.Extensions.Configuration.Json or...	10.0.2
.NET Microsoft.Extensions.Configuration.Json	Microsoft	JSON configuration provider implementation for Microsoft.Extensions.Configuration. This package enables you to read your application's settings from a JSON file. You can use JsonConfigurationExtensions.AddJsonFile extension method on IConfigurationBuilder to add the JSON configuration provider to the configuration builder.	8.0.1 10.0.2
.NET Microsoft.NET.Test.Sdk	Microsoft	The Msbuild targets and properties for building .NET test projects.	17.8.0 18.0.1
Microsoft.Playwright	Microsoft	Playwright enables reliable end-to-end testing for modern web apps. It is built to enable cross-browser web automation that is ever-green, capable, reliable and fast. Learn more at https://playwright.dev/dotnet/ .	1.57.0
Microsoft.Playwright.TestAdapter	Microsoft	Playwright enables reliable end-to-end testing for modern web apps. This is a utility package used by the Microsoft.Playwright.NUnit and Microsoft.Playwright.MSTest packages.	1.57.0

NuGet: ErasmusSystem.Tests ➔ Program.cs

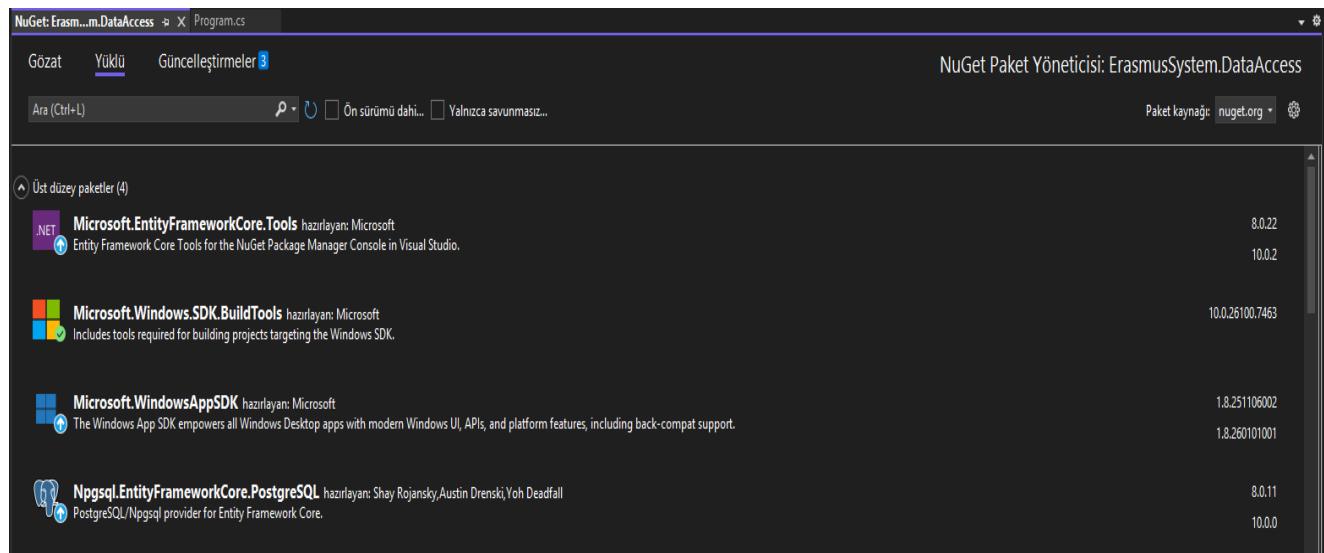
Gözat **Yüklü** Güncelleştirmeler

Ara (Ctrl+F) Paket kaynağı: nuget.org

Paket	Hazırlayan	Özet	Sürüm
NSubstitute	Anthony Egerton, David Tchepak, Alexandr Nikitin, Oleksandr Povar	NSubstitute is a friendly substitute for .NET mocking libraries. It has a simple, succinct syntax to help developers write clearer tests. NSubstitute is designed for Arrange-Act-Assert (AAA) testing and with Test Driven Development (TDD) in mind.	5.3.0
ReSpawn	Jimmy Bogard	Intelligent resetting for database tests	7.0.0
RestSharp	.NET Foundation and Contributors	Simple REST and HTTP API Client	113.1.0
xunit	jnewkirk, bradwilson	xUnit.net is a developer testing framework, built to support Test Driven Development, with a design goal of extreme simplicity and alignment with framework features.	2.9.3
xunit.runner.visualstudio	jnewkirk, bradwilson	Visual Studio 2022+ Test Explorer runner for the xUnit.net framework. Capable of running xUnit.net v1, v2, and v3 tests. Supports .NET 4.7.2 or later, and .NET 8 or later.	3.1.5

Açıklama: Playwright, PlaywrightTestAdepter ve FluentAssertions kütüphanelerinin test projesine entegrasyonu.

EK – F: Veritabanı Altyapısı (NuGet Paketleri)



Açıklama: Entity Framework Core ve PostgreSQL paketlerinin DataAccess katmanına ekleni̇ğinin teyidi.

EK – G: Mimari Blok Diyagramı

Erasmus System - Katmanlı Mimari Blok Diyagramı / Architecture Overview Diagram

