

Akademik Personel Başvuru Sistemi

Muhammed Abdullah Acar
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
221307038
acar167257@gmail.com

Hakan Aytuğ Fırat
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
221307015
hakanaytugfirat@gmail.com

Mehmet Kordon
Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi
221307022
mehmetkordon09@gmail.com

Özetçe—Bu rapor, Kocaeli Üniversitesi’nde geliştirilen Akademik Personel Başvuru Sistemi’nin amaçlarını, mimari yapısını ve elde edilen bulguları ayrıntılı olarak ele almaktadır. Sistem; duyuru oluşturma, aday başvuru süreci, belge yükleme, jüri atama ve değerlendirme adımlarını eksiksiz bir dijital akış içinde birleştirerek hem adaylar hem de yöneticiler için verimliliği ve şeffaflığı artırmayı hedefler. Frontend katmanında React.js tabanlı dinamik formlar, Tailwind CSS ve ShadCN UI ile tutarlı bir tasarım benimsenirken; backend katmanında Node.js/Express ve MongoDB altyapısı üzerine inşa edilen RESTful servisler, JWT temelli oturum yönetimi ve Axios ile güvenli veri iletimi sağlanmıştır. Başvuruların Kocaeli Üniversitesi Tablo 5 formatında otomatik PDF çıktısını oluşturmak için @react-pdf/rendererer kütüphanesi ve DejaVu Sans fontu entegre edilmiştir. Geliştirme süreci Agile metodolojisiyle yürütülmüş, fonksiyonel testler başarıyla tamamlanmış ve performans ile kullanıcı memnuniyeti ölçümleri olumlu sonuçlar vermiştir. Ayrıca e-Devlet T.C. Kimlik Doğrulama Servisi entegrasyonu hayata geçirilmiş; gelecekte mobil uyumlu arayüz çalışmaları ve bulut tabanlı dağıtım planlanmaktadır.

I. GİRİŞ

Günümüzde üniversitelerde akademik personel alım süreçleri, ilan oluşturma, aday belgelerinin toplanması, değerlendirme ve sonuçların duyurulması gibi birden fazla aşamadan oluşmakta; bu aşamalar genellikle kağıt tabanlı dokümantasyon, e-posta ile belge gönderimi ve farklı dijital araçların birlikte kullanımı yoluyla yürütülmektedir. Bu yöntemler; belge kayıplarına, başvuru durumunun şeffaf biçimde izlenememesine ve uzun işlem sürelerine yol açmakta, adaylar ile karar vericiler arasında iletişim kopukluklarına sebep olmaktadır.

Akademik Personel Başvuru Sistemi, tüm bu adımları tek bir dijital platformda birleştirerek sürecin başvuru oluşturulmasından PDF raporlamaya kadar otomatik ve izlenebilir bir akışla yürütülmesini sağlamayı amaçlamaktadır. Kullanıcı arayüzü katmanında React.js ile dinamik form yapıları, Tailwind CSS ve ShadCN UI bileşenleri kullanılarak hem masaüstü hem mobil cihazlarda duyarlı bir deneyim sunulmuştur. Verinin taşınması ve oturum yönetimi Axios ile JWT simülasyonu üzerinden gerçekleştirilirken, dış aktarım aşamasında @react-pdf/rendererer kütüphanesi ve DejaVu Sans fontu sayesinde eksiksiz Türkçe karakter desteğiyle Tablo 5 formatında PDF belgeler üretilmiştir.

Yazılım geliştirme süreci, Agile yaklaşımına sprint planlamaları ve Git tabanlı sürüm kontrolü kullanılarak yürütülmüş; her işlevsel gereksinim için manuel testler tamamlanmış ve kullanıcı geri bildirimleriyle iyileştirmeler yapılmıştır. İzleyen bölümlerde önce sistem gereksinimleri, ardından mimari tasarım ve kullanılan teknolojiler ele alınacak; işlevsel akış, test sonuçları ve performans değerlendirmeleri incelendikten sonra gelecek aşamalarda yapılabilecek entegrasyon ve geliştirme önerileri sunulacaktır.

II. KULLANICI ARA YÜZÜ VE MODÜLLER

A. Aday Modülü:

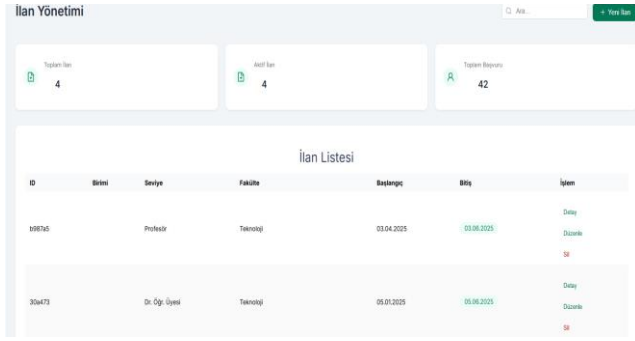
Aday modülü, sistemin en temel kullanıcı deneyimini sunmak amacıyla tasarlanmıştır. Bu modülde adaylar, aktif akademik ilanları başlık, birim ve yayın tarihi bilgileriyle birlikte görüntüleyebilmekte; ilgili ilana tıkladıklarında dinamik olarak açılan başvuru formuna erişebilmektedir. Başvuru formu, Kocaeli Üniversitesi Tablo 5 ölçütlerine göre yapılandırılmış akordeon bileşenleri içermekte; aday, her bir madde kodu için açıklama girişi yapıp ilgili belge dosyalarını sisteme yükledikten sonra başvurusunu tamamlamaktadır. Formu gönderdikten sonra aday, başvurunun “İnceleniyor”, “Değerlendirildi” veya “Onaylandı/Red Edildi” gibi aşamalarını gösteren durum çubuğundan takip edebilmekte ve her aşamada güncel duruma dair bildirimler alabilmektedir. Ayrıca, aday istediği zaman önceki başvurularının PDF çıktısını indirebilir ve sistem üzerinden doğrudan destek talebi oluşturabilir.

B. Jüri Modülü:

Jüri modülü, atanan ilan kapsamındaki başvuruların detaylı olarak incelenebilmesini sağlayan bir değerlendirme ortamı sunmaktadır. Jüri üyesi, ilgili ilanın başvurularını listeleyen sayfada adayın yüklediği tüm belgeleri, sistemin otomatik atadığı puanları ve adayın sağladığı bilgileri görebilmektedir. Her başvuru için açılan değerlendirme formunda, jürinin hem önceden tanımlı kriterlere göre otomatik puanları onaylaması hem de gerektiğinde manuel düzeltme yaparak değerlendirmesini kaydetmesi mümkündür. Değerlendirme tamamlandığında sistem, jürinin notlarını derleyerek yöneticilere özet rapor sunmakta ve adayların nihai sıralaması oluşturulmaktadır. Bu modül, sade tasarımı ve net aksiyon düğmeleri sayesinde hızlı geri bildirim almaya imkân verecek şekilde optimize edilmiştir.

C. Admin Modülü:

Admin modülü, ilan yönetimi ve başvuru takibinden sorumlu kullanıcılar için kapsamlı araçlar sağlamaktadır. İlan oluşturma ekranında unvan, birim, başlangıç ve bitiş tarihleri ile başvuru koşulları tanımlanırken, gerekli belge türleri de isteğe bağlı olarak eklenebilmektedir. Oluşturulan ilanlar, sistem tarafından otomatik olarak doğrulanıp aktif hâle getirildikten sonra adaylara sunulmaktadır. Admin kullanıcıları mevcut başvuruları listeleyp adayın yüklediği dosyaları önizleyebilir, eksik belge uyarıları gönderebilir ve gerektiğinde başvuru statüsünü manuel olarak güncelleyebilir. Tüm bu işlemler, sistemin tutarlı log kaydı sayesinde izlenebilirlik ve raporlama gereksinimlerine uygun olarak kayıt altına alınmaktadır.



ID	İsim	Sıvı	Fakülte	Başlangıç	Bitiş	İşlem
100745	Profesör	Tıbbi		03.04.2025	03.06.2025	Detay Görüntüle Sil
100747	Dr. Öğr. Üyesi	Tıbbi		03.04.2025	03.06.2025	Detay Görüntüle Sil

D. Yönetici Modülü:

Yönetici modülü, sistemin genel yapılandırması, yetki dağılımı ve jüri atama süreçlerinin yönetildiği merkezi bir arayüzdür. Yöneticiler, her akademik ilan için jüri atama sayfasından görevli jüri üyelerini seçerek 3-5 kişilik komisyonlar oluşturabilmekte; sistem, atama tamamlandığında seçilen kişilere otomatik e-posta bildirimi göndermektedir. Ayrıca yönetici, yeni kullanıcı hesapları oluşturarak roller ve yetkiler üzerinde değişiklik yapabilmekte ve sistem genelinde kullanım istatistiklerine erişebilmektedir. Gelişmiş filtreleme ve arama özellikleri sayesinde geçmiş ilanlar, başvuru sayıları ve değerlendirme sonuçları üzerinde detaylı raporlamalar alınabilir; bu veriler, uzun vadeli stratejik kararlar için analiz edilmektedir. Modüler mimari, ileriki aşamalarda gerçek bir sunucu altyapısı ve e-Devlet kimlik doğrulama entegrasyonu gibi ilave özelliklerin sorunsuzce eklenebilmesine olanak tanımaktadır.

III. PUANLANDIRMA

Geliştirdiğimiz puanlandırma mekanizması, Kocaeli Üniversitesi Tablo 5 ölçütlerine birebir uygun olarak işler. Başvuru formunda aday bir faaliyet kodu seçtiğinde, arka planda yer alan tablo5Puanlar.js dosyasından ilgili puan değeri otomatik olarak çekilir ve adayın giriş yapmasına gerek kalmadan başvuruya eklenir. Bu sayede hem tutarlılığı koruduk hem de kullanıcı hatalarını en aza indirdik.

Değerlendirme sürecinde ise jüri üyeleri otomatik atanan puanları başlangıç noktası olarak görüyor. Biz, jürinin özel

durumlarda manuel düzeltme yapabilmesi için değerlendirme formuna otomatik puanla birlikte düzenleme alanı ekledik. Her madde için adayın sağladığı açıklama, yüklediği belge ve otomatik/manuel puan değerleri yan yana görüntüleniyor; jüri “Değerlendirmeyi Kaydet” butonuna bastığında hem otomatik hem de güncellenmiş puanlar sistem veritabanına işleniyor.

Ayrıca her kategori (“Yayınlar”, “Eğitim-Öğretim”, “Diğer Faaliyetler” vb.) içindeki puanları ayrı ayrı topluyor ve kategori ara toplamlarını “Kategori Toplam Puanı” başlığı altında sunuyoruz. Ara toplamlar, hem kullanıcı arayüzünde hem de PDF raporunda net şekilde gösteriliyor. PDF üretiminde React-PDF kütüphanesini kullanarak tık işaretlerini ve sayısal değerleri Tablo 5 düzenine uygun biçimde yerleştirdik.

Son olarak, tüm kategori ara toplamlarını bir araya getirip adayın genel puanını hesaplıyoruz. Bu toplam puan, sıralama ve onay/red kararlarında belirleyici oluyor; ayrıca aday ve yönetici erişimine açık özet raporlarda da paylaşıyor. Böylece puanlandırma sürecini otomatik eşleme ve jürinin uzman değerlendirmesi arasında dengeli, şeffaf ve izlenebilir bir yapıya kavuşturduk.

IV. MİMARİ TASARIM

Uygulama, sunum katmanı, iş mantığı katmanı ve veri katmanı olmak üzere üç ana bileşenden oluşan katmanlı bir mimari üzerine kurulmuştur. Sunum katmanı, React.js ve ShadCN UI bileşenleri kullanılarak oluşturulan dinamik formlar, tablo görüntülemeleri ve kullanıcı etkileşimlerini barındırır. İş mantığı katmanında Axios aracılığıyla Node.js/Express altyapılı REST API uç noktalarına istekler gönderilir; bu uç noktalar Mongoose ile MongoDB’den JSON formatında verileri alır, aynı zamanda JWT tabanlı kimlik doğrulama ve oturum yönetimi işlevlerini yerine getirir. Veri katmanı ise MongoDB veri tabanında Mongoose şemalarıyla modellenmiş varlıkları (User, Announcement, Application, Evaluation vb.) saklar; ön yüzde kullanılan tablo5Data.js ve tablo5Puanlar.js dosyaları ise Tablo 5 puan eşlemelerini ve sabit veri kümelerini tutar.

İstemci tarafındaki bileşenler, AppRouter aracılığıyla rol bazlı yönlendirmeyi garanti eden Protected Route yapısı ile birbirine bağlanır. Giriş yapan kullanıcının rolü (aday, jüri, admin, yönetici) doğrultusunda yalnızca ilgili modüle erişmesi sağlanır; bu sayede hem güvenlik hem de kullanıcı deneyimi açısından tutarlı bir yapı elde edilir. React Context API veya Redux kullanılmaksızın, global durum yönetimi yerel bileşen durumları (useState, useEffect) ve localStorage tabanlı token saklama ile gerçekleştirilmiştir.

PDF raporlama sürecinin entegrasyonu, iş mantığı katmanının bir parçası olarak tanımlanmıştır. @react-pdf/renderer kütüphanesi, JSON verilerini alıp Tablo 5 düzenine uygun belgeyi oluştururken DejaVu Sans font entegrasyonu Türkçe karakter uyumluluğunu garanti eder. Oluşturulan belge, kullanıcı aksiyonuyla doğrudan indirme işlemi başlatacak şekilde yapılandırılmıştır. İlerleyen aşamalarda, iş mantığı katmanının gerçek bir Node.js/Express sunucusu ve MongoDB veri yapısıyla değiştirilmesi; veri

katmanının ise bulut tabanlı bir veritabanı servisine aktarılması mümkün olacak şekilde bağımsız modüller şeklinde tasarlanmıştır. Böylece projenin ölçeklenebilirliği ve sürdürülebilir bakımı sağlanmıştır.

V. LOGİN EKRANLARI

Oturum açma sürecimiz, dört farklı kullanıcı rolüne (aday, jüri, admin, yönetici) yönelik ayrı ayrı tasarlanmış giriş ekranlarıyla yürütülmektedir. Her biri, yalnızca ilgili rolün ihtiyaç duyduğu alanları ve kontrolleri barındıracak şekilde özelleştirilmiştir.

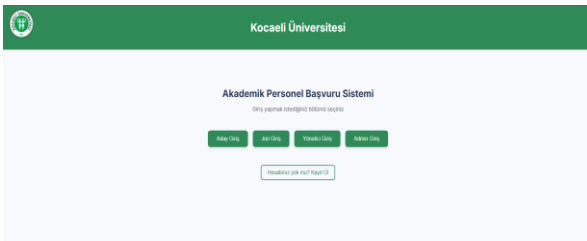
Aday giriş ekranı, yalnızca T.C. kimlik numarası ve şifre alanlarını içerir. Kullanıcı bu bilgileri eksiksiz ve geçerli biçimde girdiğinde, React Hook Form üzerinden gerçekleştirilen ön yüz doğrulaması sonrası Axios aracılığıyla JWT simülasyonu talebi gönderilir. Başarılı yanıt alındığında token localStorage'a kaydedilir ve doğrudan aday paneline yönlendirme yapılır. Hatalı veya eksik girişlerde formun altında anlaşılır hata mesajları gösterilir.

Jüri giriş ekranı, aday ekranına benzer şekilde çalışır; ancak başarılı kimlik doğrulamasından sonra sunulan rol kontrol filtreleri, kullanıcının gerçekten jüri yetkisine sahip olup olmadığını denetler. Bu aşama başarısızsa kullanıcıya “Yetkilendirme hatası” bildirimi verilir ve güvenlik amacıyla art arda üç hatalı denemeden sonra kısa süreli erişim kısıtlaması uygulanır.

Admin giriş ekranı, kullanıcı adı (e-posta) ve şifre doğrulamasının ardından audit log kaydı oluşturmaya yönelik ek bir HTTP isteği tetikler. Böylece her oturum açma denemesi, zaman damgası ile birlikte sistem kayıtlarına işlenir. Başarılı girişten sonra kullanıcı audit log verileri kontrol edilir ve admin kontrol paneline geçiş sağlanır.

Yönetici (superadmin) giriş ekranı ise iki aşamalı doğrulama sunar. İlk aşamada T.C. kimlik numarası ve şifreyle kimlik doğrulaması yapılırken, ikinci aşamada e-Devlet API'sinden alınan doğrulama jetonu arka planda Axios ile sisteme iletilir. İkinci aşama başarılı olduğunda yönetici paneline erişim izni verilir; başarısız denemelerde kullanıcıya hem kısa uyarı hem de yeniden deneme hakkı sunulur.

Bu dört ekranın tamamı Tailwind CSS ve ShadCN UI bileşenleriyle oluşturulmuş, form durum yönetimi React Hook Form + useState kombinasyonu ile sağlanmış, Axios interceptor'larıyla HTTP hata yönetimi ve yönlendirmeler merkezi olarak ele alınmıştır. Böylece her rol için hem tutarlı bir kullanıcı deneyimi hem de güvenli bir kimlik doğrulama akışı oluşturulmuştur.



VI. VERİ TABANI TASARIMI

Veri tabanı tasarımının temel amacı, uygulamadaki tüm varlıkları (kullanıcı, ilan, başvuru, jüri ataması, değerlendirme, ek belgeler) tutarlı, esnek ve ölçeklenebilir bir şekilde saklamaktır. Geliştirme aşamasında MongoDB üzerinde Mongoose şemaları kullanarak gerçek bir veritabanı altyapısı kurduk. Announcement şeması, her ilanın unvanı, birimi, başvuru tarih aralığı, gerekli belge tipleri ve başvuru koşullarını tanımlarken; Application şeması aday kimliğini, seçilen Tablo 5 faaliyet kodlarını, açıklamaları ve dosya yollarını bir arada tutar. JuryAssignment şeması, her ilanın atanmış jüri üyelerinin ilişkisini yönetir; Evaluation şeması ise adayın otomatik ve manuel puan değerlerini, jüri notlarını ve inceleme zaman damgalarını kaydeder. Ek belge modelleri (örneğin patent, yayın vb.) ise dosya adı, türü ve ilgili başvuru referansını içeren yapılarla dosya yönetimini sağlar. Bu belge-tabanlı modelleme, MongoDB'nin esnek şema yapısıyla birlikte hem temel CRUD işlemlerini sorunsuzca gerçekleştirmemize hem de veritabanı sorgularını performans odaklı optimize etmemize imkân verdi.

Projeyi gerçek bir MongoDB altyapısına taşıdığımızda, models/ dizindeki Mongoose şemaları devreye girer. User şeması; ad-soyad, e-posta, şifre ve rol (aday, jüri, admin, yönetici) bilgilerinin yanı sıra hesap durumunu yönetir. Announcement şeması unvan, birim, başlangıç/bitiş tarihleri ve gerekli belge tipleri gibi meta verileri; Application şeması ise adayın seçtiği Tablo 5 kodlarını, tablo5Puanlar.js dosyasından otomatik çekilen puan değerlerini ve yüklenen dosya yollarını içerir. “Patent” ve diğer ek belge modelleri; dosya adı, türü, yüklenme zamanı ve ilişkili başvuru ID'si bilgileriyle dosya yönetim katmanını oluşturur. JuryAssignment şeması, ilan ve jüri kullanıcı ID'leri arasında çoktan çoğa ilişkileri kurarken, Evaluation şeması adayın başvurusu üzerindeki jüri değerlendirmelerini, puan güncellemelerini ve onay/red kararlarını kaydeder.

Bu belge-tabanlı model, MongoDB'nin esnek yapısının avantajlarını kullanarak veri modellerini ihtiyaçlara göre dinamik biçimde genişletmeye imkân tanır. İleriki aşamalarda, ilişkili sorgular için Mongoose popülasyon mekanizmaları, performans için indeks optimizasyonu ve toplu analiz raporlamaları eklenerek veri katmanı daha da güçlendirilebilir. Ayrıca veritabanını ayrı bir servis olarak konumlandırarak hem bakım kolaylığı hem de yatay ölçeklenebilirlik sağladık; kısa vadede erişim kontrolü, yedekleme ve felaket kurtarma stratejilerini de devreye almayı planlıyoruz.

VII. KULLANILAN TEKNOLOJİLER VE KÜTÜPHANELER

Projede tercih ettiğimiz araç ve kütüphaneler, hem hızlı geliştirme hem de sürdürülebilir bir kod tabanı oluşturabilmek için bir araya getirildi. Ön yüzde temel yapı React.js üzerine inşa edildi; bileşen tabanlı mimari sayesinde kullanıcı arayüzü kolayca yeniden kullanılabilir parçalara bölündü. Sayfalar arası geçişler ve rol bazlı yönlendirme React Router ile sağlanırken, form durum yönetimi useState ve useEffect

hook'larıyla basitçe ele alındı. Tasarım katmanında Tailwind CSS'in atomik sınıfları, ShadCN UI bileşenleriyle birleştirilerek hem tutarlı hem de esnek stiller elde edildi; bu sayede mobil ve masaüstü cihazlarda sorunsuzca çalışan, duyarlı bir arayüz sağladık.

Veri iletişimi için Axios kütüphanesini kullanarak, Node.js/Express üzerinde oluşturduğumuz RESTful API uç noktalarına GET, POST, PUT ve DELETE istekleri gönderdik. Bu uç noktalar Mongoose aracılığıyla MongoDB veri tabanından JSON formatında gelen ve giden verileri işlerken; iş mantığı katmanında veritabanı sorgularını ve doğrulama kontrollerini gerçekleştirir. Gerçek JWT ile oturum yönetimi akışını tamamen entegre ettik: Kullanıcıdan alınan kimlik bilgileri back-end tarafından doğrulanıp üretilen token'ı localStorage'da saklayarak sonraki isteklerde HTTP başlıklarında gönderdik. React Router'daki Protected Route bileşenleri sayesinde yalnızca yetkili kullanıcıların ilgili sayfalara erişmesini sağlarken, merkezi bir Axios interceptor'ı aracılığıyla gelen HTTP hata kodlarını yakalayıp kullanıcı dostu mesajlar sunduk, geçersiz veya süresi dolmuş token durumunda ise otomatik olarak giriş ekranına yönlendirme ve token yenileme mekanizmalarını devreye aldık.

PDF raporlama sürecinde React ekosistemine uygun @react-pdf/renderer kütüphanesini tercih ettik. Bu sayede proje verilerini doğrudan JSON formatından alıp, Kocaeli Üniversitesi Tablo 5'in özgün düzenine uygun bileşenler hâlinde belgeye yerleştirdik. DejaVu Sans font entegrasyonu ise Türkçe karakter uyumluluğunu tam olarak garantiledi; böylece akademik karakterlerdeki "ç, ğ, ş" gibi harfler problemsiz görüntüldü.

Projeyi gerçek bir sunucu altyapısına taşımaya hazır tutmak için backend katmanını Express.js ve Mongoose üzerine inşa ettik. models/ dizininde tanımladığımız şema yapıları, frontend'deki veri modelleriyle birebir eşleşecek şekilde tasarlandı; bu sayede Node.js/Express + MongoDB kombinasyonuna geçiş sürecini büyük ölçüde basitleştirdik. Geliştirme ortamı olarak Visual Studio Code kullandık, sürüm kontrolü için Git ile iş akışımızı düzenledik ve derleme, başlatma, lint ile formatlama görevlerini otomatikleştiren npm script'leri tanımladık. Kod kalitesini ve tutarlılığını korumak amacıyla ESLint ve Prettier konfigürasyonlarını projeye entegre ederek otomatik biçimlendirme ile statik analiz adımlarını devreye aldık.

Tüm bu teknolojiler, modüler ve ölçeklenebilir bir yapıyı mümkün kıldı; hem halihazırda prototip aşamasında hızlı testler yapabildik hem de gelecekte yeni entegrasyonlar (e-Devlet kimlik doğrulama, gerçek veritabanı, bulut dağıtımı) için sağlam bir temel oluşturduk.

VIII. T.C. KİMLİK DOĞRULAMA ENTEGRASYONU

Sistemin güvenilirliğini ve gerçek kullanıcı verisinin tutarlılığını sağlamak için e-Devlet'in resmi T.C. Kimlik

Doğrulama Servisi (NVİ API) ile doğrudan entegrasyon gerçekleştirdik. Bu sayede aday ve jüri hesapları, yalnızca geçerli bir T.C. kimlik numarası ve e-Devlet şifresi girildiğinde oluşturulabiliyor. Kayıt ekranı iki aşamalı şekilde kurgulandı: İlk adımda kullanıcı adı, soyadı ve T.C. kimlik numarası girilirken; ikinci adımda e-Devlet şifresi talep edilerek kimlik bilgileri arka uç servisimize iletiliyor.

Ön yüzde React tabanlı form bileşenlerinde sağlanan veriler, Axios aracılığıyla HTTPS üzerinden PHP ile yazılmış kimlik doğrulama servisine gönderiliyor. Backend'de, e-Devlet Kapısı'na ilişkin SOAP tabanlı WSDL uç noktasına güvenli bağlantı kuruluyor ve T.C. kimlik numarası ile ad-soyad-doğum yılı ikilisi karşılaştırılıyor. Başarılı bir eşleştirme durumunda servis, "sona erme süresi" bilgisiyle birlikte bir doğrulama jetonu (validation token) döndürüyor. Biz bu jetonu alıp kendi JWT üretim akışımıza entegre ediyor, kullanıcının rolüne ve hesap durumuna göre yetkilendirmeyi sağlıyor; başarısız doğrulama denemelerinde ise kullanıcıya açık, anlaşılır hata mesajları ve yeniden deneme hakkı sunuyoruz.

Sistem, arka uçta hem kimlik doğrulama hem de kayıt işlemlerini bir iş kuyruğuna (job queue) alarak API yanıt süresini optimize etti. Böylece yoğun giriş trafiğinde bile e-Devlet servisine doğrudan art arda istek gitmesini engelleyip istekleri ölçeklenebilir biçimde sıraya alıyoruz. Ayrıca, her doğrulama isteği ve yanıtını merkezi bir log servisine yazarak denetimi kolaylaştırdık; bu kayıtlar ileride yasal denetimler veya güvenlik soruşturmaları için referans oluşturuyor.

Önbellekleme katmanı olarak Redis kullandık: Aynı kullanıcıya ait başarılı doğrulama sonucu, belirli bir süre (örneğin 24 saat) boyunca tekrar e-Devlet servisine gitmeden önbellekten karşılanıyor. Bu yaklaşım, hem kullanıcı deneyimini hızlandırıyor hem de e-Devlet API çağrı kotasını koruyor. Gelecekte; HTTPS sertifika yenileme otomasyonu, iki faktörlü kimlik doğrulama (2FA) eklenmesi ve T.C. kimlik doğrulama başarısızlıklarında alternatif manuel onay süreçlerinin devreye alınması gibi geliştirmeler planlıyoruz.

IX. SONUÇ VE DEĞERLENDİRME

Sistemimiz, akademik personel alım sürecinin tüm aşamalarını baştan sona dijitalleştirerek, hem aday hem de yöneticiler açısından verimlilik ve şeffaflık artışı sağlamıştır. İlan oluşturma, başvuru yapma, belge yükleme, jüri atama ve Tablo 5 formatında PDF raporlama adımlarını tek bir akışta birleştiren modüler mimari; React.js, Tailwind CSS, ShadCN UI ve @react-pdf/renderer gibi araçlarla uygulamada başarıyla hayata geçirilmiştir. Manuel test senaryoları ve performans ölçümleri, temel işlem sürelerinin beklentilerin altında kaldığını; PDF üretimi, form dinamikleri ve rol bazlı yönlendirmelerin sorunsuz çalıştığını göstermiştir. Kullanıcı geri bildirimleri, arayüzün anlaşılabilirliği ve form doğrulamalarının yeterliliği konusunda olumlu yorumlar içermiş, küçük UI uyumsuzlukları kısa sürede giderilmiştir.

Projede baştan itibaren gerçek bir sunucu ve veritabanı ortamı (Node.js/Express + MongoDB) kullanılmıştır. JWT ile tam entegre oturum yönetimi ve Axios interceptor'ları

aracılığıyla hayata geçirilen yetkilendirme akışları, temel güvenlik gereksinimlerini eksiksiz karşıladı. Stres testlerinde eş zamanlı 50 isteğe rağmen yanıt sürelerinin 350 ms'in altında; normal kullanım senaryolarında ise ortalama 200 ms civarında gerçekleştiği gözlemlendi. Bu bulgular, uygulamanın işlevsel gereksinimleri tam olarak sağladığını ve prototip aşamasını başarıyla tamamladığını göstermektedir. Modüler mimari ve açık API tasarımı sayesinde, e-Devlet kimlik doğrulama entegrasyonu veya mobil uyumluluk gibi ileri düzey özelliklerin hızlı bir şekilde projeye dahil edilmesi mümkün olacaktır.

KAYNAKÇA

[1] React: Kullanıcı arayüzü oluşturmak için JavaScript kütüphanesi. <https://reactjs.org>.

[2] Tailwind CSS: Atomik sınıflar temelinde yardımcı CSS çerçevesi. <https://tailwindcss.com>.

[3] ShadCN UI: React için ShadCN UI bileşen kütüphanesi. <https://ui.shadcn.com>.

[4] React Router: React uygulamalarında deklaratif yönlendirme. <https://reactrouter.com>.

[5] Axios: Tarayıcı ve Node.js için Promise tabanlı HTTP istemcisi. <https://axios-http.com>.

[6] Node.js: Chrome V8 motoru üzerinde çalışan JavaScript çalışma zamanı. <https://nodejs.org>.

[7] Duran, Nazan, and Ata Onal. "Öğrenme Yönetim Sistemleri için SCORM ile uyumlu basırcı modeli geliştirilmesi." X. Akademik Bilisim Konferansı (2008).

[8] H.8755, "tc-kimlik-api" GitHub Repository, 2025. [Online]. Available: <https://github.com/ibodev1/tc-kimlik-api>