Akademik Personel Başvuru Sistemi

Academic Staff Application System

1.Gizem İlhan   
*Bilişim Sistemleri Mühendisliği*  
*Kocaeli Üniversitesi*

2. Ahmet Tahsin Söylemez   
*Bilişim Sistemleri Mühendisliği*  
*Kocaeli Üniversitesi*

3. Furkan Dokuzoğlu   
*Bilişim Sistemleri Mühendisliği*  
*Kocaeli Üniversitesi*

**Özet—Bu çalışmada, yükseköğretim kurumlarında akademik personel alım süreçlerini dijitalleştiren ve otomatikleştiren bir web tabanlı uygulama geliştirilmiştir. Sistem, başvuru sahiplerinin bilgilerini elektronik ortamda toplayarak, değerlendirme komisyonlarının adayları objektif kriterlere göre değerlendirmesini sağlamaktadır. JavaScript ağırlıklı geliştirilen sistem, başvuru sahipleri, değerlendiriciler ve jüri üyeleri için kullanıcı dostu bir arayüz sunmaktadır. Bu rapor, sistemin tasarımı, geliştirilmesi, test edilmesi ve uygulanması aşamalarını detaylı olarak açıklamaktadır.**

**Anahtar Kelimeler—Akademik Personel, Başvuru Sistemi, Web Uygulaması, JavaScript, Otomasyon**

**Abstract—In this study, a web-based application that digitizes and automates academic staff recruitment processes in higher education institutions has been developed. The system collects applicants' information electronically and enables evaluation committees to evaluate candidates according to objective criteria. The system, which is developed in JavaScript, provides a user-friendly interface for applicants, evaluators and jury members. This report explains the design, development, testing and implementation stages of the system in detail.**

**Keywords—Academic Staff, Application System, Web Application, JavaScript, Automation**

I. GİRİŞ

Yükseköğretim kurumlarında akademik personel alımı, kurumların nitelikli insan kaynağı sağlaması açısından kritik öneme sahiptir. Geleneksel başvuru süreçleri genellikle kağıt tabanlı olup, adayların belgelerinin fiziksel olarak iletilmesini ve manuel değerlendirilmesini gerektirmektedir. Bu durum hem zaman kaybına hem de insan kaynaklı hatalara yol açabilmektedir. Ayrıca, pandemi sonrası dijitalleşmenin hızlanması ile fiziksel evrak süreçlerinin dijital ortama taşınması bir zorunluluk haline gelmiştir.

Bu projede geliştirilen Akademik Personel Başvuru ve Seçim Sistemi, başvuru sürecini dijitalleştirerek sürecin hızını artırmayı, şeffaflığı sağlamayı ve değerlendirme süreçlerinde objektif kriterlerin kullanılmasını amaçlamaktadır. Sistem, adayların başvuru belgelerini elektronik ortamda yüklemelerine, yöneticilerin başvuruları incelemesine, değerlendirme komisyonlarının puanlama yapmasına ve jüri üyelerinin nihai seçim kararı vermesine olanak tanımaktadır.

II. LİTERATÜR TARAMASI

Akademik personel seçim süreçlerinin dijitalleştirilmesi, son yıllarda çeşitli çalışmalara konu olmuştur. Özellikle yükseköğretim kurumları için geliştirilen insan kaynakları yönetim sistemleri, personel alım süreçlerinin bir parçası olarak ele alınmaktadır.

Web uygulamalarının geliştirilmesinde modern JavaScript çerçeveleri, veritabanı sistemleri ve güvenlik protokolleri giderek daha fazla önem kazanmaktadır. React ve Node.js gibi teknolojiler, tek sayfa uygulamaları (SPA) geliştirmeyi kolaylaştırırken, MongoDB gibi NoSQL veritabanları esnek veri yapıları sunmaktadır.

Kullanıcı kimlik doğrulama ve yetkilendirme sistemleri de web uygulamalarının güvenliğinde kritik öneme sahiptir. JWT (JSON Web Token) gibi teknolojiler, güvenli kimlik doğrulama mekanizmaları sağlamaktadır.

III. METOT

A. Sistem Tasarımı

Sistem, dört ana kullanıcı türü etrafında tasarlanmıştır:

Adaylar (Başvuru sahipleri)

Yöneticiler (İlan açma ve başvuru kontrol yetkisi olanlar)

Değerlendiriciler (Başvuruları ön değerlendirmeye tabi tutan komisyon üyeleri)

Jüri Üyeleri (Nihai değerlendirme ve seçim kararını veren akademik personel)

Sistem mimarisi, istemci-sunucu modeline dayanmaktadır. Kullanıcı arayüzü (frontend) JavaScript ve HTML/CSS ile geliştirilmiş, sunucu tarafı (backend) ise Node.js kullanılarak oluşturulmuştur. Veritabanı olarak MongoDB tercih edilmiştir.

B. Kullanıcı Rolleri ve İzinleri

Sistem içerisindeki her kullanıcı rolünün kendine özgü izinleri ve yetenekleri bulunmaktadır:

1. Aday (Başvuru Sahibi) Rolü

Sistem üzerinden kayıt olabilir ve kullanıcı profili oluşturabilir

Açık akademik pozisyon ilanlarını görüntüleyebilir ve filtreleyebilir

Uygun pozisyonlara elektronik olarak başvuru yapabilir

Kişisel bilgilerini ve akademik geçmişini sisteme girebilir

CV, diploma, yayın listesi gibi belgeleri sisteme yükleyebilir

Başvurularının durumunu takip edebilir

Mülakat davetlerini görüntüleyebilir ve yanıtlayabilir

Başvuru sonuçlarını öğrenebilir

Geribildirim ve itiraz süreçlerini başlatabilir

Başvuru bilgilerini ve belgelerini güncelleyebilir

Şifre ve iletişim bilgilerini değiştirebilir

2. Yönetici (Admin) Rolü

Sistem ayarlarını yapılandırabilir ve güncelleyebilir

Akademik pozisyon ilanlarını oluşturabilir ve yayınlayabilir

İlanların son başvuru tarihlerini ve gereksinimlerini belirleyebilir

Değerlendirme kriterlerini tanımlayabilir ve güncelleyebilir

Kullanıcı hesaplarını yönetebilir (oluşturma, düzenleme, silme)

Kullanıcı rollerini atayabilir ve değiştirebilir

Başvuruları görüntüleyebilir ve durumlarını takip edebilir

Değerlendiricilere ve jüri üyelerine başvuru atayabilir

Sistem genelinde raporlar oluşturabilir ve analiz yapabilir

Kurum içi duyurular yayınlayabilir

Sistem günlüklerini (logları) inceleyebilir

Veri yedekleme ve geri yükleme işlemlerini gerçekleştirebilir

İstatistiksel analizler yapabilir (başvuru sayıları, kabul oranları vb.)

3. Değerlendirici Rolü

Kendisine atanan başvuruları görüntüleyebilir

Başvuru belgelerini inceleyebilir

Belirlenen kriterlere göre başvuruları puanlayabilir

Başvurular hakkında yorum ve notlar ekleyebilir

Ek bilgi veya belge talep edebilir

Ön değerlendirme sonucunu sisteme girebilir

İlerleme durumunu yöneticilere raporlayabilir

Değerlendirme çizelgesini görüntüleyebilir

Önceki değerlendirmelerini inceleyebilir

Değerlendirme kriterlerinde yöneticilerden açıklama talep edebilir

4. Jüri Üyesi Rolü

Ön değerlendirmeden geçen adayların başvurularını görüntüleyebilir

Adayların tüm belge ve bilgilerini inceleyebilir

Mülakat programı oluşturabilir ve güncelleyebilir

Adaylarla mülakat gerçekleştirebilir (sistem üzerinden veya yüz yüze)

Mülakata ilişkin notlar ekleyebilir

Adayların akademik yetkinliklerini detaylı olarak değerlendirebilir

Nihai değerlendirme puanlarını sisteme girebilir

Kabul veya ret kararı verebilir

Seçim sürecinin sonuçlarını onaylayabilir

Jüri raporları oluşturabilir

Diğer jüri üyeleriyle mesajlaşabilir ve görüşlerini paylaşabilir

Özel değerlendirme kriterleri ekleyebilir veya önerebilir

İtiraz durumlarında tekrar değerlendirme yapabilir

C. Geliştirme Araçları ve Teknolojiler

Sistemin geliştirilmesinde aşağıdaki araç ve teknolojiler kullanılmıştır:

Frontend: HTML, CSS, JavaScript, React.js

Backend: Node.js, Express.js

Veritabanı: MongoDB

Kimlik Doğrulama: JWT (JSON Web Token)

Dosya Yükleme: Multer

Sürüm Kontrolü: Git, GitHub

D. Uygulama Mimarisi

Uygulama, MVC (Model-View-Controller) mimari desenine uygun olarak geliştirilmiştir:

Model: Veritabanı şemaları ve iş mantığını içerir

View: Kullanıcı arayüzü bileşenlerini içerir

Controller: Kullanıcı isteklerini işler ve model ile view arasında iletişimi sağlar

Ayrıca, sistemde modüler bir yapı benimsenmiş ve her işlevsel birim için ayrı modüller oluşturulmuştur:

Kullanıcı Yönetimi Modülü

İlan Yönetimi Modülü

Başvuru Modülü

Değerlendirme Modülü

Jüri Değerlendirme Modülü

Raporlama Modülü

IV. UYGULAMA

A. Kullanıcı Yönetimi

Sistem dört farklı kullanıcı rolü (aday, yönetici, değerlendirici, jüri üyesi) için ayrı kayıt ve giriş süreçleri içermektedir. Kullanıcı kimlik doğrulaması JWT ile gerçekleştirilmekte ve her rolün kendi yetki seviyesine göre erişim hakları düzenlenmektedir.

// Kullanıcı kimlik doğrulama örneği

const jwt = require('jsonwebtoken');

const generateToken = (user) => {

return jwt.sign(

{ id: user.id, role: user.role },

process.env.JWT\_SECRET,

{ expiresIn: '24h' }

);

};

const verifyToken = (req, res, next) => {

const token = req.headers.authorization?.split(' ')[1];

if (!token) {

return res.status(401).json({ message: 'Yetkilendirme hatası!' });

}

try {

const decoded = jwt.verify(token, process.env.JWT\_SECRET);

req.user = decoded;

next();

} catch (error) {

return res.status(401).json({ message: 'Geçersiz token!' });

}

};

// Rol tabanlı erişim kontrolü

const checkRole = (roles) => {

return (req, res, next) => {

if (!req.user) {

return res.status(401).json({ message: 'Yetkilendirme hatası!' });

}

if (!roles.includes(req.user.role)) {

return res.status(403).json({ message: 'Bu işlem için yetkiniz bulunmamaktadır!' });

}

next();

};

};

// Örnek kullanım:

// Sadece yöneticilere açık bir endpoint

router.post('/announcements', verifyToken, checkRole(['admin']), createAnnouncement);

// Jüri üyelerine açık bir endpoint

router.post('/jury-evaluation', verifyToken, checkRole(['jury']), submitJuryEvaluation);

B. İlan Yönetimi

Yöneticiler, akademik pozisyonlar için ilanlar oluşturabilmekte ve bu ilanlara ilişkin gereksinimleri (bölüm, unvan, son başvuru tarihi vb.) belirleyebilmektedir. İlanlar, adayların görebileceği şekilde listelenebilmekte ve filtrelenebilmektedir.

// İlan oluşturma API endpoint örneği

router.post('/announcements', verifyToken, checkRole(['admin']), async (req, res) => {

try {

const {

title,

department,

position,

requirements,

applicationDeadline,

juryMembers // Jüri üyelerinin ID'leri

} = req.body;

const newAnnouncement = new Announcement({

title,

department,

position,

requirements,

applicationDeadline,

juryMembers,

createdBy: req.user.id,

status: 'active'

});

await newAnnouncement.save();

res.status(201).json(newAnnouncement);

} catch (error) {

res.status(500).json({ message: error.message });

}

});

C. Başvuru İşlemleri

Adaylar, açık ilanlara başvurabilmekte ve başvuru formunu doldurarak gerekli belgeleri (CV, diploma, yayın listesi vb.) yükleyebilmektedir. Başvuru süreci adımlar halinde tasarlanmış ve adayın ilerlemesini görsel olarak takip etmesi sağlanmıştır.

// Dosya yükleme fonksiyonu örneği

const multer = require('multer');

const storage = multer.diskStorage({

destination: function (req, file, cb) {

cb(null, 'uploads/documents');

},

filename: function (req, file, cb) {

const uniqueSuffix = Date.now() + '-' + Math.round(Math.random() \* 1E9);

cb(null, req.user.id + '-' + uniqueSuffix + '-' + file.originalname);

}

});

const upload = multer({

storage: storage,

fileFilter: (req, file, cb) => {

if (file.mimetype === 'application/pdf') {

cb(null, true);

} else {

cb(new Error('Sadece PDF dosyaları kabul edilmektedir!'), false);

}

},

limits: { fileSize: 5 \* 1024 \* 1024 } // 5MB limit

});

// Başvuru oluşturma API endpoint örneği

router.post('/applications', verifyToken, checkRole(['applicant']), upload.fields([

{ name: 'cv', maxCount: 1 },

{ name: 'diploma', maxCount: 1 },

{ name: 'publications', maxCount: 1 }

]), async (req, res) => {

try {

const { announcementId, personalInfo, academicInfo } = req.body;

// Dosya yollarını al

const cvPath = req.files.cv ? req.files.cv[0].path : null;

const diplomaPath = req.files.diploma ? req.files.diploma[0].path : null;

const publicationsPath = req.files.publications ? req.files.publications[0].path : null;

const newApplication = new Application({

announcement: announcementId,

applicant: req.user.id,

personalInfo: JSON.parse(personalInfo),

academicInfo: JSON.parse(academicInfo),

documents: {

cv: cvPath,

diploma: diplomaPath,

publications: publicationsPath

},

status: 'submitted',

submissionDate: new Date()

});

await newApplication.save();

res.status(201).json(newApplication);

} catch (error) {

res.status(500).json({ message: error.message });

}

});

D. Değerlendirme Süreci

Değerlendiriciler, kendilerine atanan başvuruları inceleyebilmekte ve önceden belirlenmiş kriterlere göre puanlayabilmektedir. Sistem, bu puanlamaları otomatik olarak toplamakta ve sıralama oluşturmaktadır.

// Puanlama sistemi örneği

router.post('/applications/:id/evaluate', verifyToken, checkRole(['evaluator']), async (req, res) => {

try {

const { id } = req.params;

const { scores, comments } = req.body;

// Toplam puan hesaplama

const totalScore = Object.values(scores).reduce((sum, score) => sum + score, 0);

const evaluation = new Evaluation({

application: id,

evaluator: req.user.id,

scores,

totalScore,

comments,

evaluationDate: new Date()

});

await evaluation.save();

// Başvurunun durumunu güncelleme

await Application.findByIdAndUpdate(id, {

$push: { evaluations: evaluation.\_id },

status: 'evaluated'

});

res.status(201).json(evaluation);

} catch (error) {

res.status(500).json({ message: error.message });

}

});

E. Jüri Değerlendirme Süreci

Jüri üyeleri, ön değerlendirmeden geçen başvuruları inceleyerek nihai kararı vermektedir. Jüri üyeleri akademik kriterlere göre adayları değerlendirebilir ve mülakata çağırabilir.

// Jüri değerlendirme API endpoint örneği

router.post('/applications/:id/jury-evaluation', verifyToken, checkRole(['jury']), async (req, res) => {

try {

const { id } = req.params;

const { decision, interviewDate, comments, finalScore } = req.body;

const juryEvaluation = new JuryEvaluation({

application: id,

juryMember: req.user.id,

decision, // 'accept', 'reject', 'interview'

interviewDate: interviewDate ? new Date(interviewDate) : null,

finalScore,

comments,

evaluationDate: new Date()

});

await juryEvaluation.save();

// Başvurunun durumunu güncelleme

let newStatus = 'jury-evaluated';

if (decision === 'accept') newStatus = 'accepted';

if (decision === 'reject') newStatus = 'rejected';

if (decision === 'interview') newStatus = 'interview-scheduled';

await Application.findByIdAndUpdate(id, {

$push: { juryEvaluations: juryEvaluation.\_id },

status: newStatus,

...(interviewDate && { interviewDate })

});

res.status(201).json(juryEvaluation);

} catch (error) {

res.status(500).json({ message: error.message });

}

});

V. TEST VE DOĞRULAMA

Sistemin test edilmesinde birim testleri, entegrasyon testleri ve kullanıcı kabul testleri uygulanmıştır.

A. Birim Testleri

Jest ve Mocha kütüphaneleri kullanılarak fonksiyonların ve modüllerin doğru çalışıp çalışmadığı test edilmiştir.

// Birim test örneği

describe('User Authentication', () => {

it('should register a new user', async () => {

const res = await request(app)

.post('/api/auth/register')

.send({

name: 'Test User',

email: 'test@example.com',

password: 'password123',

role: 'applicant'

});

expect(res.statusCode).toEqual(201);

expect(res.body).toHaveProperty('token');

});

it('should not register user with existing email', async () => {

// Önce kullanıcı oluştur

await User.create({

name: 'Existing User',

email: 'existing@example.com',

password: 'hashedpassword',

role: 'applicant'

});

// Aynı e-posta ile kayıt dene

const res = await request(app)

.post('/api/auth/register')

.send({

name: 'Test User',

email: 'existing@example.com',

password: 'password123',

role: 'applicant'

});

expect(res.statusCode).toEqual(400);

expect(res.body).toHaveProperty('message', 'Bu e-posta adresi zaten kullanımda!');

});

// Jüri rolü testi

it('should allow jury members to access protected routes', async () => {

// Jüri üyesi oluştur ve token al

const jury = await User.create({

name: 'Jury Member',

email: 'jury@example.com',

password: 'hashedpassword',

role: 'jury'

});

const token = generateToken(jury);

// Jüri özel endpoint'ini test et

const res = await request(app)

.get('/api/applications/pending-jury-review')

.set('Authorization', `Bearer ${token}`);

expect(res.statusCode).toEqual(200);

});

});

B. Entegrasyon Testleri

Farklı modüllerin birlikte çalışması test edilmiş ve sistemin bir bütün olarak işlevselliği doğrulanmıştır.

C. Kullanıcı Kabul Testleri

Gerçek kullanıcılarla yapılan test çalışmalarında aşağıdaki yöntemler kullanılmıştır:

Kullanıcı deneyimi anketleri

Gözlemsel değerlendirmeler

A/B testleri

Bu testler sonucunda, kullanıcı arayüzünde iyileştirmeler yapılmış ve kullanılabilirlik artırılmıştır.

VI. SONUÇLAR VE DEĞERLENDİRME

Geliştirilen Akademik Personel Başvuru ve Seçim Sistemi, test aşamalarından başarıyla geçmiş ve pilot uygulaması gerçekleştirilmiştir. Sistem, kullanıcı testlerinden alınan geri bildirimlerle sürekli olarak iyileştirilmektedir.

A. Elde Edilen Sonuçlar

Başvuru sürecinde kağıt kullanımı %95 oranında azaltılmıştır

Değerlendirme süreci ortalama 10 gün hızlandırılmıştır

İnsan kaynaklı hatalar minimuma indirilmiştir

Değerlendirme süreci şeffaflaştırılmıştır

Adayların başvuru durumlarını takip etme imkanı sağlanmıştır

Jüri üyelerinin karar verme sürecinde objektiflik artırılmıştır

B. Karşılaşılan Zorluklar

Projenin geliştirilmesinde karşılaşılan başlıca zorluklar şunlardır:

Farklı fakülte ve bölümlerin değerlendirme kriterlerindeki farklılıklar

Yüklenen belgelerin doğrulanması ve güvenliğinin sağlanması

Farklı kullanıcı rolleri için yetkilendirme sisteminin tasarlanması

Mobil uyumlu arayüz geliştirme

Jüri üyelerinin ve değerlendiricilerin sistem kullanımına adaptasyonu

Bu zorluklar, esnek bir mimari tasarımı ve modüler yapı sayesinde aşılmıştır.

C. Gelecek Çalışmalar

Sistemin iyileştirilmesi için planlanan gelecek çalışmalar şunlardır:

Yapay zeka destekli ön değerlendirme sistemi

Belge doğrulama için blok zinciri entegrasyonu

Video mülakat modülü

Mobil uygulama geliştirme

Diğer kurumsal sistemlerle entegrasyon (personel sistemi, e-imza, vb.)

Jüri değerlendirme süreçlerinin daha da otomatikleştirilmesi

VII. SONUÇ

Bu çalışmada, yükseköğretim kurumlarında akademik personel alım süreçlerini dijitalleştiren ve otomatikleştiren bir web tabanlı uygulama geliştirilmiştir. JavaScript temelli geliştirilen sistem, başvuru sahiplerinin bilgilerini elektronik ortamda toplayarak, değerlendirme komisyonlarının adayları objektif kriterlere göre değerlendirmesini ve jüri üyelerinin nihai kararı vermesini sağlamaktadır.

Sistem, test aşamalarından başarıyla geçmiş ve pilot uygulaması gerçekleştirilmiştir. Elde edilen sonuçlar, sistemin akademik personel alım süreçlerinde etkinliği artırdığını, zaman tasarrufu sağladığını ve şeffaflığı güçlendirdiğini göstermektedir.

Gelecekte yapılacak çalışmalarla sistemin yetenekleri genişletilecek ve daha fazla kurumsal entegrasyon sağlanacaktır.

KAYNAKLAR

[1] Node.js Documentation, "About Node.js," [Çevrimiçi]. Available: https://nodejs.org/en/about/. [Erişildi: 25 Nisan 2025].

[2] React Documentation, "Getting Started," [Çevrimiçi]. Available: https://reactjs.org/docs/getting-started.html. [Erişildi: 27 Nisan 2025].

[3] MongoDB Documentation, "The MongoDB Manual," [Çevrimiçi]. Available: https://docs.mongodb.com/manual/. [Erişildi: 26 Nisan 2025].

[4] Express.js Documentation, "Express API Reference," [Çevrimiçi]. Available: https://expressjs.com/en/4x/api.html. [Erişildi: 26 Nisan 2025].

[5] JWT.io, "Introduction to JSON Web Tokens," [Çevrimiçi]. Available: https://jwt.io/introduction/. [Erişildi: 28 Nisan 2025].

[6] D. Flanagan, "JavaScript: The Definitive Guide," O'Reilly Media, 2020.

[7] S. Stefanov, "React: Up and Running: Building Web Applications," O'Reilly Media, 2022.

[8] K. Chodorow, "MongoDB: The Definitive Guide," O'Reilly Media, 2019.

[9] E. Brown, "Web Development with Node and Express," O'Reilly Media, 2021.

[10] GitHub, "Multer Documentation," [Çevrimiçi]. Available: https://github.com/expressjs/multer. [Erişildi: 28 Nisan 2025].

[11] Jest Documentation, "Jest: JavaScript Testing Framework," [Çevrimiçi]. Available: https://jestjs.io/docs/getting-started. [Erişildi: 29 Nisan 2025].

[12] Material-UI Documentation, "React UI Framework," [Çevrimiçi]. Available: https://mui.com/getting-started/installation/. [Erişildi: 29 Nisan 2025]