# Sprint 1 - 6. Hafta Raporu

1-Emir Avcı (Kimlik Doğrulama Sistemi)

Ne Yapıldı:

JWT tabanlı kimlik doğrulama için middleware ve giriş endpoint’i yazıldı. Middleware, gelen isteklerdeki JWT token’ı doğrular. Giriş endpoint’i ise kullanıcıya token üretir.

Kod: javascript

1. // JWT Middleware
2. const jwt = require('jsonwebtoken');
3. const auth = async (req, res, next) => {
4. try {
5. const token = req.header('Authorization').replace('Bearer ', '');
6. const decoded = jwt.verify(token, process.env.JWT\_SECRET);
7. req.user = decoded;
8. next();
9. } catch (error) {
10. res.status(401).json({ message: 'Lütfen giriş yapın' });
11. }
12. };
13. module.exports = auth;
14. ```
15. ```javascript
16. // Kullanıcı Girişi API
17. const express = require('express');
18. const router = express.Router();
19. const jwt = require('jsonwebtoken');
20. const User = require('../models/User');
21. router.post('/login', async (req, res) => {
22. try {
23. const { email, password } = req.body;
24. const user = await User.findOne({ email });
25. if (!user || !(await user.comparePassword(password))) {
26. return res.status(401).json({ message: 'Geçersiz email veya şifre' });
27. }
28. const token = jwt.sign({ id: user.\_id }, process.env.JWT\_SECRET, {
29. expiresIn: '24h'
30. });
31. res.json({ token, user: { id: user.\_id, email: user.email, role: user.role } });
32. } catch (error) {
33. res.status(500).json({ message: error.message });
34. }
35. });
36. module.exports = router;

2-Kayra Doğan (Admin Paneli)

Ne Yapıldı:

Admin panelinin ana görünümü ve istatistik kartları React ile oluşturuldu. Panelde toplam rezervasyon, bekleyen onay ve günlük rezervasyon sayıları gösteriliyor.

Kod: javascript

1. import React, { useState, useEffect } from 'react';
2. import axios from 'axios';
3. import { Line } from 'react-chartjs-2';
4. const AdminDashboard = () => {
5. const [stats, setStats] = useState({
6. totalRezervasyonlar: 0,
7. bekleyenOnaylar: 0,
8. gunlukRezervasyonlar: 0
9. });
10. useEffect(() => {
11. fetchStats();
12. }, []);
13. const fetchStats = async () => {
14. try {
15. const response = await axios.get('/api/admin/stats');
16. setStats(response.data);
17. } catch (error) {
18. console.error('İstatistikler yüklenirken hata:', error);
19. }
20. };
21. return (
22. <div className="p-6">
23. <h2 className="text-2xl font-bold mb-6">Admin Paneli</h2>
24. <div className="grid grid-cols-3 gap-4">
25. <div className="bg-white p-4 rounded-lg shadow">
26. <h3 className="text-lg font-semibold">Toplam Rezervasyon</h3>
27. <p className="text-2xl">{stats.totalRezervasyonlar}</p>
28. </div>
29. <div className="bg-white p-4 rounded-lg shadow">
30. <h3 className="text-lg font-semibold">Bekleyen Onaylar</h3>
31. <p className="text-2xl">{stats.bekleyenOnaylar}</p>
32. </div>
33. <div className="bg-white p-4 rounded-lg shadow">
34. <h3 className="text-lg font-semibold">Günlük Rezervasyonlar</h3>
35. <p className="text-2xl">{stats.gunlukRezervasyonlar}</p>
36. </div>
37. </div>
38. </div>
39. );
40. };
41. export default AdminDashboard;

3- Muhammed Emir Demirtaş (Bildirim Sistemi)

Ne Yapıldı:

Rezervasyon onay/red bildirimleri için e-posta gönderim servisi yazıldı. Nodemailer ile SMTP üzerinden e-posta gönderimi sağlanıyor.

Kod: javascript

1. const nodemailer = require('nodemailer');
2. const transporter = nodemailer.createTransport({
3. host: process.env.SMTP\_HOST,
4. port: process.env.SMTP\_PORT,
5. secure: true,
6. auth: {
7. user: process.env.SMTP\_USER,
8. pass: process.env.SMTP\_PASS
9. }
10. });
11. const sendRezervasyonEmail = async (rezervasyon) => {
12. const mailOptions = {
13. from: process.env.SMTP\_FROM,
14. to: rezervasyon.email,
15. subject: 'Rezervasyon Onayı',
16. html: `
17. <h1>Rezervasyonunuz Onaylandı</h1>
18. <p>Tarih: ${rezervasyon.tarih}</p>
19. <p>Saat: ${rezervasyon.saat}</p>
20. <p>Durum: ${rezervasyon.durum}</p>
21. `
22. };
23. await transporter.sendMail(mailOptions);
24. };
25. module.exports = { sendRezervasyonEmail };

4-Çağrı Doğan (Kullanıcı Yönetimi)

Ne Yapıldı:

Kullanıcı verilerini saklamak için MongoDB şeması ve roller bazında erişim kontrolü için middleware yazıldı.

Kod: javascript

1. // Kullanıcı Modeli
2. const mongoose = require('mongoose');
3. const bcrypt = require('bcryptjs');
4. const userSchema = new mongoose.Schema({
5. email: {
6. type: String,
7. required: true,
8. unique: true
9. },
10. password: {
11. type: String,
12. required: true
13. },
14. role: {
15. type: String,
16. enum: ['user', 'admin'],
17. default: 'user'
18. }
19. });
20. userSchema.pre('save', async function(next) {
21. if (this.isModified('password')) {
22. this.password = await bcrypt.hash(this.password, 8);
23. }
24. next();
25. });
26. userSchema.methods.comparePassword = async function(password) {
27. return bcrypt.compare(password, this.password);
28. };
29. module.exports = mongoose.model('User', userSchema);
30. ```
31. ```javascript
32. // Kullanıcı Yetkilendirme Middleware
33. const checkRole = (roles) => {
34. return (req, res, next) => {
35. if (!req.user) {
36. return res.status(401).json({ message: 'Yetkilendirme gerekli' });
37. }
38. if (!roles.includes(req.user.role)) {
39. return res.status(403).json({ message: 'Bu işlem için yetkiniz yok' });
40. }
41. next();
42. };
43. };
44. module.exports = checkRole;