**Sprint 1 - 4. Hafta Raporu**

1. React ile Form Validasyonu ve Hata Yönetimi (Emir Avcı)

* Ne yapıldı:

Form validasyonu eklendi, hata mesajları oluşturuldu, kullanıcı geri bildirimleri eklendi.

* Kullanılan Kodlar:

1. Rezervasyon.js (validasyon eklendi):
2. import React, { useState } from 'react';
3. import './Rezervasyon.css';
4. function Rezervasyon() {
5. const [formData, setFormData] = useState({
6. tarih: '',
7. saat: '',
8. isim: '',
9. telefon: '',
10. email: ''
11. });
12. const [errors, setErrors] = useState({});
13. const [success, setSuccess] = useState(false);
14. const validateForm = () => {
15. const newErrors = {};
16. if (!formData.tarih) {
17. newErrors.tarih = 'Tarih seçimi zorunludur';
18. }
19. if (!formData.saat) {
20. newErrors.saat = 'Saat seçimi zorunludur';
21. }
22. if (!formData.isim) {
23. newErrors.isim = 'İsim alanı zorunludur';
24. }
25. if (!formData.telefon) {
26. newErrors.telefon = 'Telefon alanı zorunludur';
27. } else if (!/^[0-9]{10}$/.test(formData.telefon)) {
28. newErrors.telefon = 'Geçerli bir telefon numarası giriniz';
29. }
30. if (!formData.email) {
31. newErrors.email = 'E-posta alanı zorunludur';
32. } else if (!/\S+@\S+\.\S+/.test(formData.email)) {
33. newErrors.email = 'Geçerli bir e-posta adresi giriniz';
34. }
35. setErrors(newErrors);
36. return Object.keys(newErrors).length === 0;
37. };
38. const handleSubmit = async (e) => {
39. e.preventDefault();
40. if (validateForm()) {
41. try {
42. const response = await fetch('/api/rezervasyon', {
43. method: 'POST',
44. headers: {
45. 'Content-Type': 'application/json',
46. },
47. body: JSON.stringify(formData),
48. });
49. if (response.ok) {
50. setSuccess(true);
51. setFormData({
52. tarih: '',
53. saat: '',
54. isim: '',
55. telefon: '',
56. email: ''
57. });
58. } else {
59. throw new Error('Rezervasyon oluşturulamadı');
60. }
61. } catch (error) {
62. setErrors({ submit: error.message });
63. }
64. }
65. };
66. return (
67. <div className="rezervasyon-container">
68. <h2>Rezervasyon Yap</h2>
69. {success && (
70. <div className="success-message">
71. Rezervasyonunuz başarıyla oluşturuldu!
72. </div>
73. )}
74. {errors.submit && (
75. <div className="error-message">
76. {errors.submit}
77. </div>
78. )}
79. <form onSubmit={handleSubmit}>
80. {/\* ... mevcut form alanları ... \*/}
81. {errors.tarih && <span className="error">{errors.tarih}</span>}
82. {errors.saat && <span className="error">{errors.saat}</span>}
83. {errors.isim && <span className="error">{errors.isim}</span>}
84. {errors.telefon && <span className="error">{errors.telefon}</span>}
85. {errors.email && <span className="error">{errors.email}</span>}
86. </form>
87. </div>
88. );
89. }
90. export default Rezervasyon;
91. Node.js ile Hata Yönetimi ve Middleware (Kayra Doğan)

* Ne yapıldı:

Hata yönetimi middleware'i eklendi, API güvenliği artırıldı.

* Kullanılan Kodlar:

1. middleware/errorHandler.js:

1const errorHandler = (err, req, res, next) => {

1. console.error(err.stack);
2. if (err.name === 'ValidationError') {
3. return res.status(400).json({
4. error: 'Validation Error',
5. details: Object.values(err.errors).map(e => e.message)
6. });
7. }
8. if (err.name === 'MongoError' && err.code === 11000) {
9. return res.status(400).json({
10. error: 'Duplicate Error',
11. message: 'Bu zaman dilimi için zaten bir rezervasyon mevcut'
12. });
13. }
14. res.status(500).json({
15. error: 'Server Error',
16. message: 'Bir hata oluştu, lütfen daha sonra tekrar deneyin'
17. });
18. };
19. module.exports = errorHandler;

2- middleware/validateRezervasyon.js:

1. const validateRezervasyon = (req, res, next) => {
2. const { tarih, saat, isim, telefon, email } = req.body;
3. if (!tarih || !saat || !isim || !telefon || !email) {
4. return res.status(400).json({
5. error: 'Validation Error',
6. message: 'Tüm alanlar zorunludur'
7. });
8. }
9. if (!/^[0-9]{10}$/.test(telefon)) {
10. return res.status(400).json({
11. error: 'Validation Error',
12. message: 'Geçerli bir telefon numarası giriniz'
13. });
14. }
15. if (!/\S+@\S+\.\S+/.test(email)) {
16. return res.status(400).json({
17. error: 'Validation Error',
18. message: 'Geçerli bir e-posta adresi giriniz'
19. });
20. }
21. next();
22. };
23. module.exports = validateRezervasyon;
24. MongoDB Veritabanı Optimizasyonu (Emir Demirtaş)

* Ne yapıldı:

Veritabanı performansı iyileştirildi, indeksler güncellendi.

* Kullanılan Kodlar:

1. models/Rezervasyon.js (optimizasyonlar eklendi):
2. // ... mevcut şema tanımı ...
3. // Performans optimizasyonları
4. rezervasyonSchema.index({ createdAt: -1 }); // Tarihe göre sıralama için
5. rezervasyonSchema.index({ durum: 1, tarih: 1 }); // Durum ve tarihe göre sorgulama için
6. // Veri doğrulama
7. rezervasyonSchema.pre('save', function(next) {
8. if (this.isNew) {
9. this.createdAt = new Date();
10. }
11. next();
12. });
13. // ... model export ...
14. Test ve Dokümantasyon (Çağrı Doğan)

* Ne yapıldı:

API testleri yazıldı, dokümantasyon güncellendi.

* Kullanılan Kodlar:

1. tests/rezervasyon.test.js:
2. const request = require('supertest');
3. const app = require('../app');
4. const Rezervasyon = require('../models/Rezervasyon');
5. describe('Rezervasyon API', () => {
6. it('yeni rezervasyon oluşturmalı', async () => {
7. const rezervasyon = {
8. tarih: '2024-03-20',
9. saat: '19:00',
10. isim: 'Test Kullanıcı',
11. telefon: '5551234567',
12. email: 'test@example.com'
13. };
14. const res = await request(app)
15. .post('/api/rezervasyon')
16. .send(rezervasyon);
17. expect(res.status).toBe(201);
18. expect(res.body.isim).toBe(rezervasyon.isim);
19. });
20. });

2- docs/API.md (güncellendi):

1. # API Dokümantasyonu
2. ## Hata Kodları
   1. 400: Validation Error
   2. 401: Unauthorized
   3. 404: Not Found
   4. 500: Server Error
3. ## Rate Limiting
4. API istekleri dakikada 100 istek ile sınırlandırılmıştır.